On the Privacy of Frequently Visited User Locations

Zohaib Riaz, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems
University of Stuttgart, Germany
Email: {*zohaib.riaz, frank.duerr, kurt.rothermel*}@ipvs.uni-stuttgart.de

***Abstract*—With the fast adoption of location-enabled devices, Location-based Applications (LBAs) have become widely popular. While LBAs enable highly useful concepts such as geo-social networking, their use also raises serious privacy concerns as it involves sharing of location data with non-trusted third parties. In this respect, we propose an approach that protects the frequently visited locations of users, e.g., a bar, against inferences from long-term monitoring of their location data. Such inferences equate a privacy leak as they reveal a user's personal behavior and interests to possibly malicious non-trusted parties.**

To this end, we first present a study of a dataset of location check-ins to show the existence of this threat among users of LBAs. We then propose our approach to protect visit-frequency of the users to different locations by distributing their location data among multiple third-party Location Servers. This distribution not only serves to avoid a single point of failure for privacy in our system, it also allows the users to control which LBA accesses what information about them. We also describe a number of possible attacks against our privacy approach and evaluate them on real-data from the check-ins dataset. Our results show that our approach can effectively hide the frequent locations while supporting good quality-of-service for the LBAs.

I. Introduction

With growing use of smart-phones, Location Based Applications (LBAs) have also thrived in the past decade. Not only do LBAs benefit end-users by providing them with valuable functionality such as location sharing with friends (e.g. via Foursquare), they also enable the application providers to generate revenue by translating the collected location data into high-end services, such as real-time traffic information (e.g. Waze).

However, sharing location data with third-party LBAs is known to raise personal privacy concerns among users. Obviously, a pair of geo-location coordinates may represent significant private information when analyzed along with contextual information, such as the publishing time and the visited place type. Moreover, this context is not hard to acquire given rapid increase in popularity of geo-social networks, such as Foursquare (over 55 million users), where publishing location equates to announcing presence at semantically well-categorized, real-world venues, e.g., bars, restaurants, hospitals, etc. Since users may frequently share their location, explicitly or along with geo-tagged content such as tweets and photos, an adversary could collect and analyze contextually rich movement trails over time, and thus, can make undesired inferences about users' habits, interests, and inclinations etc.

While many works have offered to defend location privacy in use of LBAs, protecting visits to sensitive *semantic locations*[1], e.g., hospitals, has only been a recent focus [1]–[3]. However, these approaches focus on avoiding privacy breaches associated with individual visits only. We argue that frequent visits to seemingly non-sensitive locations, e.g., a bar, may also represent special personal behavior/interests, and if unprotected, amount to a privacy leak. The underlying privacy threat is well-described by the following observation made in a United States court-case where a suspect was tracked using a GPS device for a month by the police. The court ruled: *"A person who knows all of another's travels can deduce whether he is a weekly church goer, a heavy drinker, ..."* and continuing to finish with *"... and not just one such fact about a person, but all such facts."* [4].

Against the above motivated threat, we propose an approach to avoid the uncontrolled release of users' frequent semantic locations to LBAs. We leverage the finding from works such as [5] that users share their location differently with various classes of audiences, e.g., friends, employers, etc. Therefore, our approach allows users to define a set of representative personas, i.e., portrayals of their personality, e.g., professional, social, family, etc., which they wish to present to the various LBAs. It then ensures that an LBA may only make those inferences by analyzing a user's location data stream which match the traits represented by the shared persona. A naive way to achieve this would be to fully deny an LBA the access to any location context that does not match its persona. However, doing so reasonably leaks information to the LBA that the user is hiding certain information. Instead, our approach chooses to selectively share location updates with LBAs for non-matching context such that they do not convey strong user interest. Consequently, our approach allows a users to, for example, appear "normal" to their employer albeit their drinking habits, or avoid annoying ads even though they shop frequently.

In designing our approach, we also account for another important privacy threat that arises particularly in the use of location sharing LBAs. These LBAs typically build upon a back-end *Location Server* (LS), which manages user-reported locations and implements access-control mechanisms to enforce strictly authorized access to reported locations by LBAs. However, we believe that trusting a third-party LS for location-data security is naive since data-breach events at popular data-houses where sensitive customer information is hacked, leaked, or stolen are significantly prevalent [6]. Therefore, in

[1]semantic place-categories associated with venues, e.g., home, work etc.

Published in Proceedings of the 17th International Conference on Mobile Data Management: MDM'16; Porto, Portugal, June 13-16, 2016

our proposal, we store privacy-sensitive information on *non-trusted* server infrastructures. To this end, we distribute a user's location data among multiple non-trusted LSs from different providers. This distribution is done such that any LS stores only a *portion* of the privacy-sensitive information to limit the data revealed to an attacker if the server is compromised (no single-point-of-failure with respect to privacy). Thus, it is impossible for a single LS to build a precise profile of a user.

With many available back-end LS providers (cloud-based as well as self-hosted), e.g., [7]–[10], we believe that our proposal is highly practical. Moreover, as our approach does not modify location data but rather only distributes it among LSs, it can act as a lower layer for other privacy preserving approaches such as obfuscation [1]–[3] for the protection of individual visits.

Overall, our contributions are as follows. First, we establish the relevance of protecting frequent locations to our daily lives by studying a real-world dataset of location check-ins. Second, we propose the first approach for protecting frequent user-locations. Finally, we evaluate our approach on the check-in dataset for the attained privacy for the users (against both, a user's location history-aware as well as a population's location history-aware adversary) and quality-of-service (QoS) for the LBAs. Our results show that our approach not only disables powerful attacks against protected frequent user locations, it also supports good QoS for LBAs.

Discussions in the rest of the paper are organized as follows. First, we introduce and study a dataset of location check-ins to show the privacy threat of frequent locations. In Sections III and IV, we present the system model and the problem statement for our approach respectively. In Section V, we explain our frequency-protection algorithm. Finally, before our conclusion, we evaluate our algorithm for the attained privacy and QoS in Sections VI and VII respectively.

II. Exploring the dataset

In order to study the relevance of the threat posed by frequent locations in our daily lives, we analyzed a dataset of check-ins, based on geo-tagged posts from Twitter's public feed, collected by Cheng et. al. [11] during a period of 5 months from late September 2010 till Jan 2011. In this section, we will discuss our pre-processing of the data as well as provide the evidence of prevalence of frequent locations among a large number of users in the dataset.

A. The Dataset and its Pre-processing

Overall, the dataset consists of $22,506,721$ check-in entries from $225,098$ users. Apart from user IDs and latitude-longitude pairs, each entry also contains tweet text and a venue-ID. A high percentage of check-ins (53%) were from users who had linked their Foursquare accounts to Twitter. From these, we selected a subset of $10,306$ users who had a minimum of 1 check-in per day and a total reporting time spanning at least 30 days. These users were selected from across the United States, to where 36% of all Foursquare users in the dataset belong, to increase our chances of finding a semantically-labeled venue for their check-ins at Foursquare.

To acquire semantic labels for the check-ins, we used Foursquare's free API [12]. However, we simplified their elaborate hierarchy of semantic labels to obtain a high level set of semantic locations, as shown in Figure 1. These high-level semantic locations (referred as locations for the rest of paper) were intended to be intuitively linkable to users' personality traits. Moreover, we ignore check-ins at user residences as this information does not represent user's interests. Similarly, check-ins at *Professional Places* and *Education* institutions are also ignored unless they represent special user-behavior, e.g., check-ins at work-place after 6 pm.

After the above filtering of uninteresting check-ins, we were left with an average of 141 check-ins and a reporting period of 114 days per user. For the rest of the document, we will refer to these users as the *population*.

B. Evidence of threat

For identifying frequent user locations, we want to determine whether a user's visit-frequency to a location is abnormally high compared to other users in the population. To this end, we rely on the notion of Percentile Rank (PR). Given a dataset of single attribute values $X = \{x_1, ..., x_n\}$, the percentile rank of a particular value x_i in the dataset determines the percentage of dataset values which are less than or equal to x_i.

For each user u_j in the population $\boldsymbol{U}$, we summarize all of their check-ins to the set of semantic locations, $\boldsymbol{S} = \{s_1, ..., s_{14}\}$, from Fig. 1, into a set of visit-frequencies as:

$$\boldsymbol{f}^{u_j} = \{f^{u_j}_{s_1}, ..., f^{u_j}_{s_{14}}\} \tag{1}$$

Next, we determine their frequency-rank profile as:

$$\boldsymbol{r}^{u_j} = \{r^{u_j}_{s_1}, ..., r^{u_j}_{s_{14}}\} \tag{2}$$

where each entry $r^{u_j}_{s_i}$ represents percentile rank of $f^{u_j}_{s_i}$ among non-zero frequencies of other users in $\boldsymbol{U}$ to the semantic location s_i. Given the frequency-rank profile of u_j, we can now determine the set of locations, $\boldsymbol{C} \subset \boldsymbol{S}$, which the user visits more frequently than a certain proportion, th_{crtl}, of the population, i.e.:

$$\boldsymbol{C} = \{c_1, ..., c_n\} \ s.t. \ c_i \in \boldsymbol{S} \wedge r^{u_j}_{c_i} > th_{crtl} \tag{3}$$

For the rest of the paper, we term all frequent locations in set $\boldsymbol{C}$ as user's *critical locations*. Similarly, the threshold th_{crtl}

No.	Category Name	No.	Category Name
1.	Arts & Entertainment	8.	*(Professional Places)*
2.	*(Education)*	9.	Professional Schools
3.	Food	10.	Medical Center
4.	Tea/Coffee/Juices	11.	Spiritual
5.	Nightlife	12.	Shop & Service
6.	Outdoor & Recreation	13.	Financial Institutions
7.	Athletics & Sports	14.	Fitness

Fig. 1. The high-level semantic locations $\boldsymbol{S}$ from our check-ins dataset.

is called the *criticality threshold* and represents a user-tunable parameter in our protection algorithm.

Following the above described steps, we determine the critical locations of all users in $\boldsymbol{U}$. Figure 2 shows the histogram of number of critical locations found per user when criticality threshold is set to 90 and 80. Note that even with th_{crtl} as high as 90, approximately 4000 users have one, and around 1500 users have two critical locations. The plot of cumulative percentage, on the other hand, shows that at $th_{crtl} = 90$, only 40%, and at $th_{crtl} = 80$, only 16% of the population do not have any critical location. Also note that the mass of the frequency-distributions shifts to the right when th_{crtl} is changed from 90 to 80 because more users have visit-frequencies ranking higher than 80% of the population. In short, Fig. 2 shows that the population exhibits significant prevalence of critical locations, even with values of criticality threshold as high as 80 and 90.

Figure 3(a) also shows actual visit frequencies at various percentiles in our population for all locations in $\boldsymbol{S}$. Note, for example, that while 10 trips per month equate $70th$ percentile in "Shop & Service" category, only 7 trips are enough to reach the $95th$ percentile for "Medical Center". Alternatively, in part (b) of Fig. 3, we plot the cumulative distribution of users against visit-frequencies for a subset of locations in $\boldsymbol{S}$. Note that the cumulative proportion on the y-axis of the figure is equivalent to the percentile rank of the corresponding frequency on the x-axis. This figure suggests even more evidently that each location has a characteristic cumulative distribution function (CDF) of percentile ranks over visit frequencies which naturally makes them differently sensitive to a given visiting-frequency value (see the various percentiles at 10 visits/month in Fig. 3(b)).

From the above study, it can be generally concluded that high-visit frequencies expose users to easy inference about their interests by placing them in top ranks among the population. Consequentially, we will describe our proposal against such inference in the next section which aims to protect the critical user locations. Note that for the rest of the paper, we assume that the characteristic CDFs of locations in $\boldsymbol{S}$, as seen above in Fig. 3(b), are publicly known and thus, are available as functions of the visit-frequency parameter to our algorithm as well as the adversary. This is a reasonable assumption since these CDFs are intuitively easy to understand and verify, and may be obtained from a dataset like ours or by conducting a public survey. Moreover, these CDFs, as functions of visit-frequency, depict an aggregate behavior of the population and thus, do not convey information about particular users. Similar assumptions about the availability of various aggregated characteristics of locations are common in the location privacy literature such as knowledge of place popularities in [1], [13] and staying durations in [2].

Thus, for a given location category s_i, our algorithm as well as an adversary may determine the rank r of a certain visit-frequency value f and vice versa as follows:

$$r = CDF_{s_i}(f) * 100 \quad \text{and} \quad f = CDF_{s_i}^{-1}(r/100) \qquad (4)$$

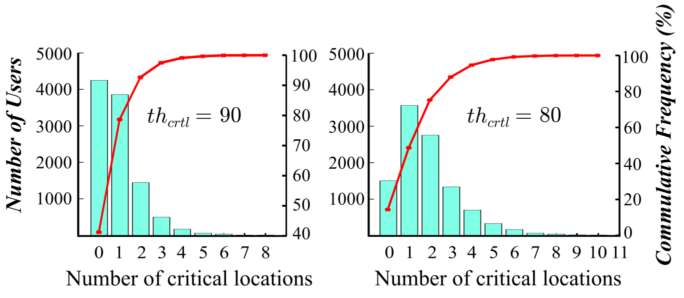

Fig. 2. Distributions of number of critical locations among the users in the dataset for criticality threshold of 90 and 80.

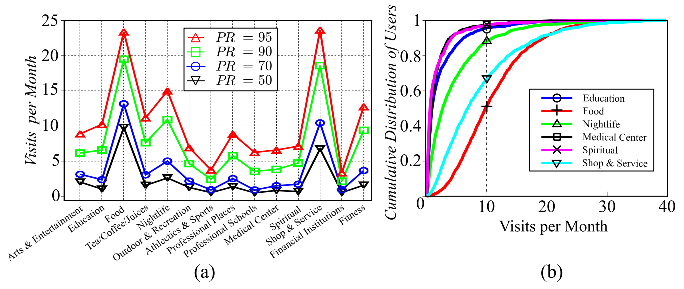

Fig. 3. (a) Monthly visit-frequencies for semantic locations at various percentiles in our dataset. (b) Cumulative distribution of users (equivalent to percentile ranks) against visit-frequencies for a subset of locations in $\boldsymbol{S}$. Note that a given frequency value implies different percentile ranks for different locations.

III. System Model

For our approach, the system comprises three components, namely: the mobile device, the Location Servers (LSs), and the Location-Based Applications (LBAs).

The **mobile device** is capable of determining its location using positioning technologies such as GPS. It runs our trusted *Location-Privacy (LP)* system service which has exclusive access to the captured location data. The LP-service performs location updates to the LSs as governed by our visit-frequency protection algorithm. Note that in our system location updates are user-triggered only, i.e., the LP-service does not update the user location autonomously. Moreover, the LP-service requires to know the underlying location semantics for each update. To this end, we assume that the LP-service can access an (offline) map of places which extends to, at least, the regions of the user's daily movement. We also assume that the communication channel between the mobile device and the LSs is secure.

In our system, **Location Servers (LSs)** are provided by different non-trusted providers and are responsible for managing the user location updates. For instance, LSs could be provided by different self-hosted [10] or cloud-based providers [7]–[9]. Using multiple such non-trusted LSs, we implement a distributed location service. For each user, the LP-service distributes their location updates among $(n + 1)$ LSs where n is a user-defined parameter. Moreover, the LSs implement access-control mechanisms for enabling authorized access to a user's location data by the LBAs. While the LBAs query about a user's latest location update only, we assume that the LSs may store the past location updates received from all users, e.g., for data-mining purposes. Therefore, data-breaches at the LSs pose a privacy threat. Furthermore, since the LS providers are not trusted, we assume that they may collude with each other to undermine user privacy.

Finally, the LP-service allows the user to define personas, e.g., work, family, social, etc., and assign them to different **Location Based Applications (LBAs)**. Based on these personas, our LP-service specifies, for each LS, those LBAs which are allowed to access the user's data. Therefore, LBAs acquire the user's last updated location by querying their accessible LSs.

IV. Problem Statement

We now formally define the requirements from our (visit-) frequency protection algorithm for ensuring privacy of critical user locations.

The goal of our algorithm is to ensure that location data shared with any LBA matches its user-assigned persona. If a user's critical locations are not part of the assigned-persona, then the LBA in question should not be able to infer from the location data, shared over long period of time, that the user visits these locations with high frequency.

More precisely, a *persona* $P_i \in \boldsymbol{P}$ comprises a subset of those semantic locations from $\boldsymbol{S}$ (see Fig.1) for which the user feels comfortable to share location data in full with the LBAs irrespective of whether these locations are critical or not. The complement set P_i', however, represents those locations from $\boldsymbol{S}$ which should be protected if critical, i.e., if the user's visit-frequency for these locations is higher than the user-defined criticality threshold th_{crtl} (see Eq. 3).

Functional Requirement: Formally, an LBA with persona P_i may build an *observed* frequency-profile $\boldsymbol{f}_{obs}$ of the user by aggregating their accessible location updates. From $\boldsymbol{f}_{obs}$, the LBA determines the set of observed critical locations $\boldsymbol{C}_{obs}$. Given the set of the user's actual critical locations $\boldsymbol{C}_u$, the functional requirement from our algorithm is:

$$\boldsymbol{C}_{obs} \bigcap (P_i' \cap \boldsymbol{C}_u) = \varnothing \tag{5}$$

In words, the observed critical locations should not contain any critical location which was left out in the persona P_i.

Adversary Model: In our system, a "*weak*" adversary may take the role of an LBA that combines location information from their authorized LSs (access-rights granted by LP-service) as well as from any compromised/colluding LSs. We also assume that the weak adversary knows our algorithm, the number of total critical locations of the user n, and the set value of criticality threshold th_{crtl}. Moreover, "*strong*" adversaries may possess additional auxiliary attack knowledge which we will gradually introduce in the next two sections for ease of readability. In general, the adversary may use their knowledge to perform probabilistic attacks for determining the still unknown critical locations of the user against which we define the following requirement.

Privacy Requirement: Assume a user u who visits a total of $m \leq |\boldsymbol{S}|$ different semantic locations out of which $n \leq m$ are critical, i.e., $|\boldsymbol{C}_u| = n$. If an adversary already knows $k < n$ critical locations, then their attack probability P_{attack} of finding the remaining $(n-k)$ critical locations should not exceed the probability of random selection α_{rand}:

$$P_{attack}(k) \leq \alpha_{rand}(k) \text{ where } \alpha_{rand}(k) = \frac{(n-k)}{(m-k)} \tag{6}$$

In other words, the prior knowledge of k critical locations should not help the adversary to distinguish the still unknown critical among the remaining $(m-k)$ user locations.

In order to protect the critical locations, our algorithm hides a portion of location updates from the LBAs. However, this might reduce their Quality of Service (QoS). Therefore, another design consideration for our algorithm is to maximize the QoS for the LBAs, i.e., by hiding minimal number of location updates from the LBAs while still ensuring privacy for critical locations. Accordingly, we quantify QoS as the average proportion of the all location updates of the user that are shared with the LBAs.

V. The Frequency Protection Algorithm

In this section, we present two versions of our visit-frequency protection approach, namely, a basic and an advanced version, which differ in the considered adversary knowledge. As the name implies, the basic version is designed against the weak adversary mentioned in Section IV.

A. Overview of the Basic Algorithm

The fundamental idea of our privacy approach is that each of the n LSs, denoted as, $LS_i \in LS_{1..n}$, only stores a limited set of location visits that, if aggregated into a frequency profile, can only reveal a small number of (without loss of generality in our case exactly one) critical locations. Thus, compromising one LS out of $LS_{1..n}$ will only reveal a user profile with exactly one critical location instead of the complete and precise user profile with all critical locations. Consequently, no LS is a single point of failure in terms of privacy. Moreover, one LS, denoted as LS_0, stores a completely *safe profile* that does not reveal any critical information. This safe profile can be refined selectively by adding information from other LSs from $LS_{1..n}$. Therefore, in our system, all LBAs are allowed to access LS_0. By additionally granting the LBAs access rights to certain LSs from $LS_{1..n}$, the LP-service can give individual LBAs access to certain critical locations which are permitted by their persona without revealing other critical profile information. In this way, our algorithm can meet the functional requirement in Eq. 5. For an adversary, getting access to the complete profile is very hard since this would require to compromise all LSs. Moreover, the approach implements "graceful degradation of privacy": the number of critical locations revealed increases linearly with the number of compromised LS.

More precisely, for a user's critical location $c_i \in \boldsymbol{C_u}$, our algorithm protects the actual visit-frequency by distributing its visits among LS_0 and LS_i, such that the visits at LS_0 alone do not reveal c_i as critical. As an example, Fig. 4

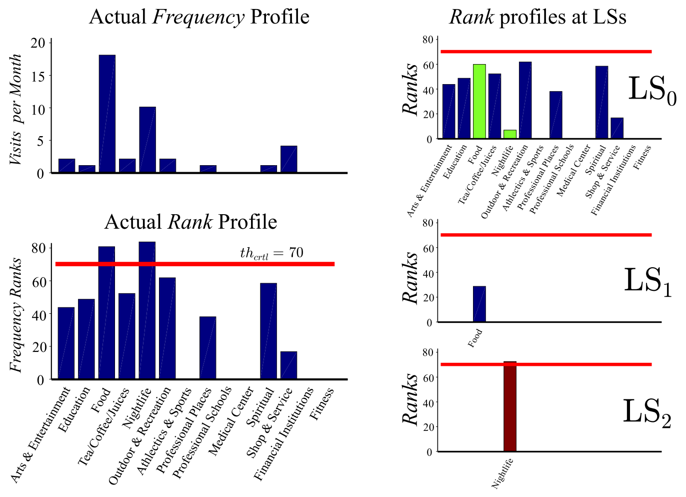

Fig. 4. Left half: the *actual* frequency (top) and rank profiles (bottom) of the user. From the rank-profile, *Food* and *Nightlife* are critical locations given $th_{crtl} = 70$. Right half: the rank-profiles after distribution of location data on 3 LSs by our algorithm.

shows on the left the actual frequency-profile of a hypothetical user. Having set $th_{crtl} = 70$, their actual rank-profile, given underneath the frequency-profile, shows that the user has two critical locations, namely *Food* and *Nightlife*. When using our algorithm, the resulting inferred rank-profiles of the user at the LSs are shown on the right half of the figure. While LS_0 sees all user locations, it cannot determine anyone of them as critical, i.e., the user profile is safe. As for LS_1 and LS_2, they can at most determine one critical location from its rank.

Generally, it can be seen, as shown by this example, that by accessing at least LS_0, all LBAs can acquire a significant proportion of user's location data thus promising high average QoS. Moreover, since a high number of users in our dataset have few critical locations (upto three for almost 90% of users for $th_{crtl} = 80$ in Fig. 2), we believe that two to four LSs suffice for most users (one LS per critical location and one LS for the safe user profile). Nevertheless in Section VIII, we also discuss the special case where the number of critical locations of a user exceed the number of available LSs.

Next, we first discuss how our algorithm, while running on any single user's device, performs a one-time determination of the user's set of critical locations $\boldsymbol{C_u}$, and later, the steps taken to ensure their protection.

B. Determination of a user's critical locations

For determining the critical location set $\boldsymbol{C_u}$, our algorithm first performs continuous background monitoring of the user's visited locations for a short period of time, e.g., a week or two (during which no location updates are published). Note that such prior background monitoring is a popular requirement for all state-of-the-art privacy-preserving approaches that model user mobility with, for example, Markov chains, prior location distributions, etc. such as [3], [14], [15]. At the end of the monitoring period, our algorithm uses the logged visits to first approximate the *actual* visit-frequency profile $\boldsymbol{f}_a$ of the user (see Eq. 1). Then, the corresponding rank-profile $\boldsymbol{r}_a$ for this frequency-profile is determined by computing the rank of each individual visit-frequency in $\boldsymbol{f}_a$ using Eq. 4. Since Eq. 4 exploits the publicly known characteristic CDFs of the semantic locations for rank determination (as discussed in Section II-B), the rank-profile can be computed *independently* on the user's device, i.e., without requiring any real-time aggregation of visit-frequencies of other users. With the computed rank-profile $\boldsymbol{r}_a$, the set $\boldsymbol{C_u}$ of critical locations of the user is determined as per the user-specified value of criticality-threshold th_{crtl} (see Eq. 3).

C. The Basic Algorithm

With the critical locations set $\boldsymbol{C}_u$ of the user known to our algorithm, we now explain its basic execution.

Since LS_0 is meant to hold a completely safe user profile (not revealing any critical locations), our algorithm needs to publish the user trips to each critical location $c_i \in \boldsymbol{C_u}$ at a reduced frequency to LS_0. The remaining trips for each c_i are then published to its corresponding LS_i.

To this end, our algorithm modifies the user's actual rank-profile $\boldsymbol{r}_a$ into a desired rank-profile $\boldsymbol{r}_d$ where the ranks of critical locations are reduced to lie below the criticality threshold th_{crtl}. Note here that the method used to compute $\boldsymbol{r}_d$ can pose a privacy risk. Considering Fig. 4 again for an example, let us first assume the case that our algorithm decides for both critical locations (*Food* and *Nightlife*), a desired rank of 70. Although, this value of desired rank is safe ($\leq th_{crtl} = 70$), it invites attack from the adversary who can infer critical locations as the ones closest in rank to th_{crtl}. More generally, if the method used to determine the desired ranks is deterministic, the adversary can, by knowledge of our algorithm, find out all critical locations of the user.

Therefore, our algorithm selects for each $c_i \in \boldsymbol{C_u}$ the desired ranks $\boldsymbol{r}_d$ in a non-deterministic, random fashion. To be more precise, it sorts the frequency-ranks of non-critical locations to form the set $\boldsymbol{r}_{nc}$. Then, it randomly selects a consecutive pair of ranks from the union $\{0 \cup \boldsymbol{r}_{nc} \cup th_{crtl}\}$, as the range over which to randomly select the rank for c_i. Against the determined set $\boldsymbol{r}_d$, we also calculate the corresponding set of desired frequencies $\boldsymbol{f}_d$ using the notion of inverse percentile.

For performing user-triggered location updates during execution, our algorithm (see Algorithm 1) always publishes *non-*

Algorithm 1 Perform user-triggered location update

```
1: procedure PERFORM_LOCATION_UPDATE
2:     $(l_t, s_t) \leftarrow$ Get_Location_And_Its_Semantics()
3:     if $s_t \in \boldsymbol{C_u}$ AND Rand()> $f^d_{s_t}/f^a_{s_t}$ then
4:         Update: $l_t \rightarrow LS_i$                ▷ since $s_t = c_i$
5:     else
6:         Update: $l_t \rightarrow LS_0$
7:     end if
8:     Schedule_Fake_Events()    ▷ for Advanced Algorithm
9: end procedure
```

critical visits to LS_0. As for the visits to critical locations c_i, LS_0 is updated with a probability of $f^d_{c_i}/f^a_{c_i}$, i.e., with the *ratio of desired to actual visit frequency* of user to c_i. The remaining visits for c_i are published to LS_i.

From the perspective of the LBAs, they may acquire a user's last updated location from their authorized LSs in two ways, i.e., by subscribing at these LSs for location notifications, or, by querying them explicitly. In the later case, each LS replies with its last received location update. The LBA can then determine the latest among the received location updates by comparing their time stamps.

D. The Advanced Algorithm

The advanced algorithm extends the basic one against "strong" adversaries. Particularly, these adversaries, in the role of malicious LBAs running on a user's device, can also sniff the communication channel used by our LP-service. While the communication content, i.e., the location data, is protected by encryption, such an adversary is, nevertheless, able to determine the time and the destination LS for all location updates. Note that this capability is realistic since LBAs that monitor network traffic statistics of other LBAs already exist, e.g., the "Network Connections" app for Android.

For finding the critical locations, the adversary is most interested in the *critical* location updates that are sent to $LS_{1...n}$. To limit attack possibilities however, our algorithm hides the destination LS for critical updates by performing a **uniform update**, i.e., by sending syntactically similar messages with fake data to the rest of LSs among $LS_{1...n}$. Note that while these fake messages do not corrupt actual location data at the LSs (are identified and discarded), they disable the adversary from knowing which inaccessible LS actually received the location update since a message is sent to all of them.

As our protection algorithm runs, the adversary aggregates the location update history of the user as a stream, $\boldsymbol{H} = \{o_{t_1}, o_{t_2}, e_{t_3}, o_{t_4}, e_{t_5}, o_{t_6}, ...\}$, comprising observable events $o_{t_j} \in \boldsymbol{O}$ and unobservable events $e_{t_j} \in \boldsymbol{E}$. For an adversary Adv_k that can access k LSs and LS_0, observable events o_{t_j} are those which are published at these LSs. In contrast, unobservable events e_{t_j} are location updates for those $(n-k)$ critical locations which are published to the rest of the LSs. However, since Adv_k knows the update times for all $e_i \in \boldsymbol{E}$, they may attempt to identify user's critical locations. We analyze this attack and then discuss the corresponding defense offered by our algorithm.

Attack: At first, Adv_k builds a mobility model Ω from observable events $\boldsymbol{O}$. Using Ω, Adv_k may attempt to identify the actual visited semantic location s_i represented by an unobservable event e_t at time t by maximizing the conditional probability $P(s_i|e_t, \Omega)$ over $s_i \in \boldsymbol{S}$. By Bayes' rule:

$$P(s_i|e_t, \Omega) = \frac{P(e_t|s_i, \Omega)P(s_i|\Omega)}{P(e_t|\Omega)} \tag{7}$$

Ignoring the denominator, which is same for all locations, Adv_k's goal is to maximize the numerator, i.e., the product of the likelihood $P(e_t|s_i, \Omega)$ and prior $P(s_i|\Omega)$ over $\boldsymbol{S}$.

Considering first the **prior** $P(s_i|\Omega)$, it represents *the total probability of visiting s_i compared to all locations in $\boldsymbol{S}$ according to* Ω. Since Adv_k computes Ω from events in $\boldsymbol{O}$ only, the computation of prior is erroneous as the number of events in $\boldsymbol{O}$ for critical locations are artificially reduced by our algorithm. In Fig. 4 for example, from LS_0, the approximated prior for *Nightlife* would turn out to be small relative to all other locations due to severe clipping of its frequency.

As for the **likelihood** term $P(e_t|s_i, \Omega)$, it represents *the probability of visiting s_i at time t (e.g., particular hour of day) compared to all other times as per* Ω. Unlike the prior, whose computation depends heavily on information about other locations in $\boldsymbol{S}$, the likelihood is a relatively intrinsic probability distribution for each location s_i. Therefore, it can be approximated correctly by the adversary, even for critical locations, by accessing LS_0 only. For example, if a user shops more frequently on Saturdays, then this trend is kept intact even after a proportion of all shopping trips are probabilistically hidden by our algorithm.

Although the prior $P(s_i|\Omega)$ is erroneous, the likelihood term may still be informative to the adversary Adv_k. For instance, with Ω representing visit-time distributions over $\boldsymbol{S}$, events $e_{day,hour}$ may be correctly identified as trips to a particular location s_j since no other location is visited by user at $t = \{day, hour\}$, e.g., only church visits every Sunday noon. For Ω modeling correlated location updates with a Markov chain, the same scenario may occur if s_j is uniquely visited after another location, e.g., only visiting shopping centers after visiting an ATM. As a general conclusion, the mobility model Ω can increase adversary's knowledge of user's critical locations by narrowing down their search space to only a subset of locations $S' \subset \boldsymbol{S}$ for which likelihoods are non-zero for event e_t. This attack can, therefore, increase attack probability $P_{attack}(k)$ for the adversary to a value more than the probability of random selection $\alpha_{rand}(k)$ (see Eq. 6).

Defense: To prevent the above attack, our algorithm must ensure that *the nature of a location s_i being critical or not is independent of the set of events $\mathbf{E}$ seen by the adversary*. Enforcing this independence implies that all locations of the user, apart from the already known critical ones, are equally likely candidates for the unknown critical locations, thus, conforming to the probability $\alpha_{rand}(k)$ in Eq. 6.

To implemet the above requirement, our algorithm generates an additional fake set of unobservable events $\boldsymbol{E}'$ for each visited location s_i such that s_i appears critical to the adversary, even if it is non-critical. More precisely, the sum of actual and fake events for each location amount to a frequency-rank which is equivalent to the maximum rank r^{max} in user's rank-profile. Like events in $\boldsymbol{E}$, all $e'_t \in \boldsymbol{E}'$ are *uniform updates*, although fake. This means that none of the messages sent to the LSs for each event e'_t contain any real location update. Note that the adversaries cannot syntactically differentiate between a real or fake unobserved event. To generate $\boldsymbol{E}'$, the function "Schedule_Fake_Events" (see Algorithm 2) is called after every user-triggered update (see Algorithm 1) or previously scheduled fake update (see Algorithm 3). This

Algorithm 2 Schedule fake update events

```
1: procedure SCHEDULE_FAKE_EVENTS
2:     Update $\Omega$, $\boldsymbol{f}_a$, $\boldsymbol{r}_a$, $\boldsymbol{f}_{fake}$
3:     $\boldsymbol{f}^{\triangle} \leftarrow$ Get_Fake_Frequency_Profile($\boldsymbol{f}_a$, $\boldsymbol{f}_{fake}$, $\boldsymbol{r}_a$)
4:     $\forall_{s_i \in \boldsymbol{S}} : t_{next} \leftarrow$ Sample $P(t|s_i, \Omega)$ for $t \in (t_{now}, 2T]$
5:     $\forall_{s_i \in \boldsymbol{S}}$ : Enqueue_Scheduled_Event ($s_i$, $t_{next}$)
6: end procedure
```

Algorithm 3 Generate fake update events

```
1: procedure GENERATE_FAKE_EVENT
2:     Uniform Update "fake" $\rightarrow LS_{1...n}$
3:     Schedule_Fake_Events()
4: end procedure
```

function first determines the delta frequency $\boldsymbol{f}^{\triangle}$ representing the number of fake uniform update events for each location s_i that still need to be published in order to make s_i appear critical. This is achieved by subtracting the user's actual frequency $\boldsymbol{f}_a$ and current fake event frequency $\boldsymbol{f}_{fake}$ from the required fake frequency $\boldsymbol{f}_{req}$. Here $\boldsymbol{f}_{req}$ represents frequency to each location s_i corresponding to the maximum rank r^{max} in user's rank profile. Next, the "Schedule_Fake_Events" function schedules the next fake event e'_{next} for each location s_i with frequency $f^{\triangle}_{s_i} \in \boldsymbol{f}^{\triangle}$. With time-period $T = 1/f^{\triangle}_{s_i}$, it schedules e'_{next} at time t_{next}, with respect to current time t_{now}, in the time range $(t_{now}, t_{now} + 2T]$. More specifically, t_{next} is sampled in the range $t \in (t_{now}, t_{now} + 2T]$ using normalized likelihood $P(t|s_i, \Omega)$ as the sampling probability. Consequentially, the timings of fake events in $\boldsymbol{E}'$ for all locations s_i comply with adversary's mobility model of the user Ω, thus making them indistinguishable from actual unobservable events for s_i in $\boldsymbol{E}$.

Note that, for our algorithm, computing the likelihood $P(t|s_i, \Omega)$ is straightforward. For instance, with Ω being temporal distribution of visits to s_i, the likelihood can be computed by normalizing visit counts over hours in a week. Moreover, for modeling all adversaries Adv_k, with $k \in [1, n]$, it suffices for our algorithm to compute these likelihoods (and Ω) from observable events at LS_0 only since $Adv_{k\in[1,n]}$ also learn likelihoods for unknown critical locations from LS_0.

VI. Security Analysis

In our system, the adversary Adv_k attacks user's location data, for determining their critical locations, in two fundamental forms: (1) as the user's update history $\boldsymbol{H}$, (2) as observed frequency profile $\boldsymbol{f}_{obs}$ in comparison with frequency profiles of other users in the population. Note that the later form of attack knowledge, i.e., the adversary's awareness of other users' movement behavior, is ignored by most existing location privacy mechanisms.

As for the attacks based on the update history $\boldsymbol{H}$, we have shown, in the previous section, that our algorithm guarantees privacy against them. In this regard, our algorithm limits the number of observable events $\boldsymbol{O} \in \boldsymbol{H}$ for the adversary (cf. Section V-C) as well as corrupts the unobservable events $\boldsymbol{E} \in \boldsymbol{H}$ by addition of fake ones (cf. Section V-D). However, this privacy guarantee incurs additional communication overhead, due to generation of fake events as evaluated in Section VII-A.

Considering now the second form of attack, we wish to answer the following question: *Can Adv_k identify the critical locations in the user's observed profile $\boldsymbol{f}_{obs}$ by their general knowledge of the frequency-profiles from other users?* Note that for this attack, we are assuming that the strong adversary additionally possesses a large dataset of location check-ins like ours. Based on this auxiliary knowledge, Adv_k may for example know that, in general, people who often visit shopping centers also visit food places frequently. Then if a user's observed profile $\boldsymbol{f}_{obs}$ only satisfies the first part of this relation, then Adv_k has reason to believe that food places are this user's critical location. Analyzing the strength of this kind of attack requires its formulation as a machine-learning problem where a learner can educate itself about the correlations in the visit-frequencies of different locations by analyzing frequency-profiles from many users. Then, it can possibly predict critical locations for an unseen user from their observed profile $\boldsymbol{f}_{obs}$. Next, we show the detailed formulation of this attack and evaluate its strength on our check-in dataset.

Machine-learning based attacks: To simplify the following explanation, we assume a target user with observed frequency-profile, $\boldsymbol{f}_{obs,n=1}$, where only one visited location is critical. The goal of the adversary is to identify this unknown critical location from $\boldsymbol{f}_{obs,n=1}$. In this regard, the adversary may pursue two representative methodologies from machine learning, namely, Classification and Regression. However, due to space limitation, we will only discuss classification here. A thorough analysis with regression can be found in [16].

To perform the attack, the adversary attempts to learn a classifier which identifies the correct critical location given an observed frequency profile. Such a classifier needs to be trained on example observed-profiles from many other users where exactly one critical location is protected by our algorithm (supervised-learning). More specifically, each observed profile $\boldsymbol{f}_{obs,n=1}$ and its corresponding critical location s_i form a training example. During training, the classification algorithm learns to distinguish between observed profiles for different critical locations in $\boldsymbol{S}$. Thus, after training, the classifier can be used to predict the critical location for a given, previously unseen, observed profile.

However, as described in Section V-C, our algorithm modifies the observed frequencies for critical locations in a random fashion. By this, it obfuscates correlations in frequencies of different locations, which obviously degrades the predictability of these frequencies. Our evaluations below show that existing machine learning techniques cannot predict frequencies with a sufficiently high accuracy due to this obfuscation.

For our experiments, we applied our frequency-protection algorithm (c.f. Section V) to protect the frequency profiles of 3539 users in our check-in dataset who had exactly one critical location ($n = 1$). From these users, we created a training-dataset where their protected frequency-profiles and the corresponding actual critical location formed the training

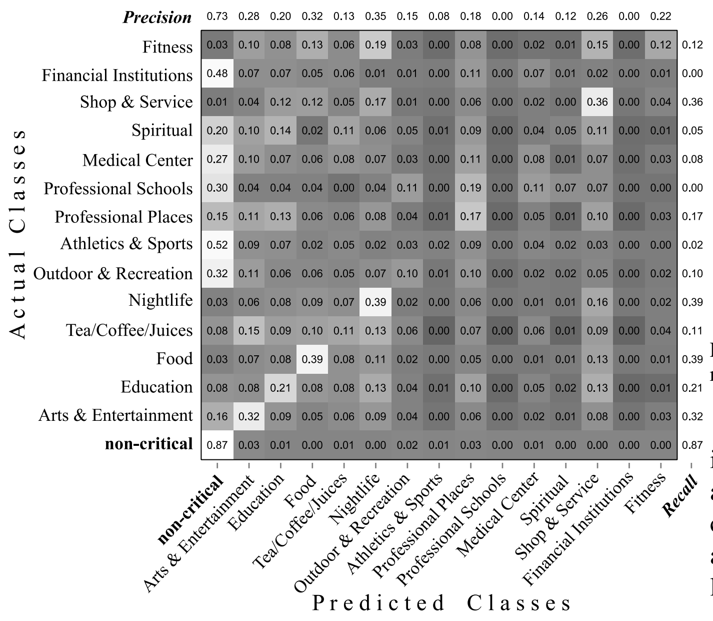

Fig. 5. Confusion Matrix for the Random Forest Classifier. Note the high prediction accuracy of classification (87%) for the non-critical frequency-profiles in the left-bottom corner of the matrix.

examples. We trained and tested two popular classification algorithms on this dataset, a multi-class Random Forest classifier (RF) and a Support-Vector Machine (SVM). While RF classifier performs prediction based on an ensemble of learned decision-trees, the SVM attempts to project the input data into a higher dimensional space where different classes in the data are easily separable. Both the algorithms are known for their good classification performance [17]. However, our best tuned versions of both classifiers (RF: 5000 decision-trees with 5 variables tried per tree-splits, SVM: radial kernel, cost= 50, gamma= 0.005, 10-fold cross-validation, class imbalance taken into account for both RF and SVM) resulted in a similar, but low overall classification accuracy of 25%.

A possible reason for this low identification rate of critical locations from observed profiles could be the wrong choice of RF and SVM classifiers for modeling the correlations in the frequency-profile data. To examine this possibility, we performed another experiment where the above created dataset was appended with frequency profiles of those users who did not have any critical locations. For these users, the corresponding output label of "non-critical" was used. After re-training the RF and SVM classifiers on the modified dataset, the classification accuracy of both classifiers turned out to be similarly high for the "non-critical" class ($\sim$ 87%) whereas the classification accuracy for identifying critical locations was again low, i.e., 22%. This result asserts that while the RF and SVM classifiers are actually well-suited to identify the frequency correlations in the unchanged data, the random modification of visit-frequencies for critical locations by our algorithm results in their poor predictability for the adversary.

The detailed confusion matrix for this evaluation is shown in Fig. 5. Here, the diagonal entries give the classification accuracy of each class. Apart from the "non-critical" class, all classes have at most 39% accurate predictions. Therefore, an adversary cannot use a classification attack to detect critical locations from the observed user profiles.

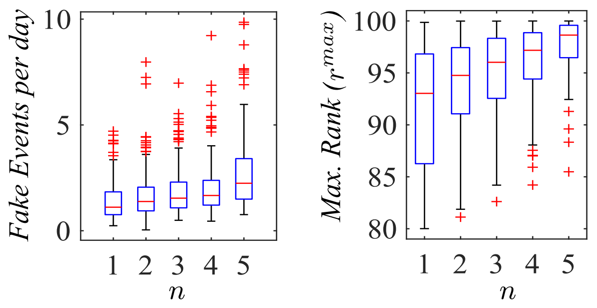

Fig. 6. Number of fake update events per day (left) and maximum frequency ranks (right) vs. n (number of user's critical locations)

VII. Evaluation of Communication Cost and QoS

For evaluations of communication cost and QoS, we chose a select set of 1228 long-history users from 10306 users in our check-in dataset who had more than 250 check-ins. We now present the results of application of our frequency-protection algorithm on the location histories of these users while setting the criticality threshold th_{crtl} to 80 and allowing a maximum of 5 critical locations.

A. Communication Cost

As discussed in Section V-D, our algorithm generates fake unobservable events $e'_t \in \boldsymbol{E}'$ for avoiding attacks based on location update history $\boldsymbol{H}$. For each user, we quantify the communication cost as the fake-event rate, i.e., number of fake events that were generated per day by our algorithm. In our evaluations, we use visit-time distributions as users' mobility model since check-ins from users were, on-average, infrequent (less than 2 a day). Figure 6 shows the results. The left half of the figure shows the boxplot of fake-event rate for users with varying number of total critical locations n. As n increases from 1 to 5, the fake-event rate increases almost linearly from a median of 1 to 2 events per day. This increase is explained by the distribution of maximum rank r^{max} values in user-rank profiles (see right half of Fig. 6). With more critical locations (increasing n), the chances of having a higher value of maximum rank r^{max} also rise. Therefore, correspondingly greater number of fake events are generated for each location to appear at the rank r^{max}. However, the overall low fake event rate of $1-2$ events per day for users in our dataset represents an affordable price for the privacy guarantee against the update history based attacks presented in Section V-D.

B. Quality of Service (QoS) for LBAs

For quantifying QoS, we evaluate the proportion of location updates that are still accessible to the LBAs when access to k out of $(n+1)$ LSs is *denied*. Note that $k=n$, i.e., only one

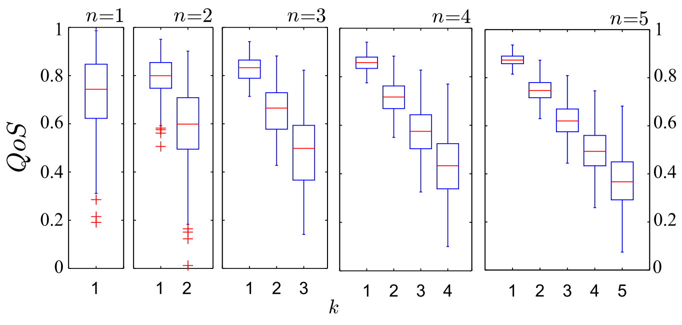

Fig. 7. QoS (proportion of accessible location updates) vs. k (number of LSs out of n to which access is denied).

accessible LS out of n, always implies access to LS_0 in our scenario since any LBA can be given access to LS_0 without revealing any critical location.

For each user, we computed the average proportion of accessible updates over all combinations of $n-k$ accessible LSs for $k \in [1,(n-1)]$. Figure 7 shows the achieved QoS for all users grouped by n, i.e., the total number of critical locations. For all values of n in general, it can be seen that for lower values of k (1 and 2), LBAs are still informed about $\sim 80\%$ and $\sim 70\%$ of the location updates, respectively, for the majority of the population. With increasing values of k, however, the LBAs get to access less and less of all location updates thus highlighting the tradeoff between achievable QoS and the privacy of critical locations. However, since a high proportion of the population users had 2 or less total critical locations (see Fig. 2), we believe that the QoS for LBAs used by most of the population would be high.

VIII. The Case of Limited Number of LSs

We now briefly discuss how our approach may be extended to handle the case where the number of critical locations of a user exceed the number of available LSs. In such a case, some of the critical locations of a user remain unprotected. We propose two solutions to address this problem. While both of our solutions meet the privacy requirement of implementing user-defined personas (see Eq. 5), they differ in whether a single LS is allowed to learn about more than one critical location of the user or not.

LSs may learn more than one critical location: For protecting the remaining critical locations of a user, this solution allows the publishing of more than one critical location per LS. By exploiting the access control mechanisms that are implemented by the LSs, fine-grained access to the critical locations stored on the same LS can be defined for different LBAs. Thus the user-defined personas for the LBAs can be fully implemented as per Eq. 5.

The drawback of this solution, however, is that an LS may give up more than one critical location of the user to the adversary if it is malicious or is compromised. The next solution aims to avoid this problem.

LSs learn at most a single critical location: Similar to the first solution, this solution also publishes multiple critical locations at the LSs to accommodate for high number of critical locations of the user. However, we require that the user publishes *at most one critical location per LS per user-identity*. Consequentially, a user must possess distinct fake identities (one per critical location stored on the LS) and that these identities are shared with the LBAs in order to allow them to access some or all of the user data. Note that in this case, the LS may be able to correlate two fake-ids of a user by matching the network-based identifiers, such as IP address, from the communication channel. Therefore, we assume that the mobile device communicates with the LSs using an anonymization service such as Tor [18]. Note that this assumption does not disable the location-update timing based attacks discussed in Section V-D since the adversary can still observe the timing of the outgoing updates from our LP-service. Therefore, the defense proposed in Section V-D is still necessary.

It may apparently seem that by using distinct fake-ids for all critical locations and the safe-profile, a user can place all of their location data at a single LS without any privacy threat. However, this reasoning has a serious pitfall. An LS can easily correlate the fake-ids of the critical locations to that of the safe-profile by matching the visited venues in the critical visits to the venues in the safe-profile. This necessitates that the safe-profile should be stored on a separate LS where no critical location is stored. Note that this venue-correlation attack cannot be directly applied to two or more critical locations that are stored on the same LS because the venues of these critical locations are different from each other since they belong to different semantic categories. Thus it is possible to store multiple critical locations at a single LS. However, to mitigate any correlation attacks that may be possible within a single LS and to increase the robustness against colluding/compromised LSs, data of critical locations should be distributed among the maximum number of available LSs.

IX. Related Work

Uptil today, location privacy has received a lot of attention from the research community as surveyed in [19]. However, we limit our discussion to those approaches which take location semantics into account for privacy preservation.

Most schemes offering semantic location privacy build upon the idea of location obfuscation [20]. In its basic form, obfuscation replaces the actual location to be shared with an LBA by a region whose size determines the tradeoff between privacy and quality of service. Building on top of the region-size privacy metric, later works also include a number of semantically heterogeneous locations, e.g., school, shopping center, etc., inside the obfuscation region to hide the actual semantic context from the adversary [2], [21]. To incorporate personalization, Damiani et al. [1] propose a probabilistic model of space for the generation of obfuscation regions which respect user preferences regarding their sensitive semantic locations. Their approach ensures that the probability of the actual user location lying inside a sensitive place is limited to under a desired level relative to the total probability of being located inside the whole obfuscation region. Yigitoglu et al. [13] extend this model to an urban setting with road-network and represent location probabilities inside a region by relative popularities of actual venues.

While still protecting individual visits only, other works also consider adversary's knowledge of user's location update history [3], [14], [15]. These approaches model user mobility as Markov chains, and for protecting privacy, either obfuscate the sensitive locations [14] or replace them with dummy locations [15].

However, the above privacy mechanisms are unsuitable for the protection of visit-frequency information. For instance, obfuscating frequent trips may still allow an adversary to estimate the user's visit-frequency to different regions on the map. Consequentially, if highly frequented regions are not sufficiently heterogeneous in terms of location semantics, e.g., a shopping district, the adversary may be able to identify user's critical locations. In general, obfuscation approaches can also be attacked by techniques related to automatic semantic labeling of user visits such as [22]. Using machine-learning, these techniques exploit the contextual information, such as visit timings, nearby businesses, etc., to identify the actual visited semantic location. In contrast, our approach blocks out any contextual information about critical visits by publishing them to those LSs for which the adversary does not have access authorization. Moreover, by explicit inclusion of multiple non-trusted LSs in the system model, our approach mitigates the threat of a single point of failure for privacy under data-breaches. In this regard, our work is similar to that of [23]. However, their work also protects single location updates only while not considering location semantics or visit-frequency information. Finally, the problem of hiding individual visit-frequency statistics has also been addressed in offline publishing of check-in history statistics, e.g., for venue recommendations [24], [25]. These works require trusted servers for their implementations and cannot be used for online location sharing.

X. Conclusion

In this paper, we have presented a new attack to user privacy based on the analysis of visiting frequencies of location traces. By analyzing real data, we have shown that an attacker can derive user profiles including private information like their interests from such traces. To counter such attacks, we have proposed an approach that hides critical information from attackers including malicious location servers and location-based applications. Our evaluations show that our approach hides critical information successfully while preserving sufficient quality of information for the location-based applications.

For future, one possible extension of our work is to consider additional contextual information than just location for hiding users' interests. This may especially be beneficial in the geo-social networking scenario where user's posts may contain frequent mentions of their interests, e.g., as hashtags, and thus, require additional measures for their privacy.

Acknowledgment

This work is a part of the PriLoc project (Privacy-Aware Location Management) of the University of Stuttgart funded by German Research Foundation (DFG) grant RO 1086/15-1.

References

[1] M. L. Damiani, E. Bertino, and C. Silvestri, "The probe framework for the personalized cloaking of private locations," *Trans. Data Privacy*, vol. 3, no. 2, pp. 123–148, 2010.

[2] B. Lee, J. Oh, H. Yu, and J. Kim, "Protecting location privacy using location semantics," in *Proc. of KDD*. ACM, 2011, pp. 1289–1297.

[3] G. Theodorakopoulos, R. Shokri, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Prolonging the hide-and-seek game: Optimal trajectory privacy for location-based services," in *Proc. of WPES*. ACM, 2014, pp. 73–82.

[4] "United States of America v. Antoine Jones," [Online]. Available: http://tinyurl.com/q5rxpkf.

[5] H. Cramer, M. Rost, and L. E. Holmquist, "Performing a check-in: Emerging practices, norms and 'conflicts' in location-sharing using foursquare," in *Proc. of MobileHCI*. ACM, 2011, pp. 57–66.

[6] "World's biggest data breaches & hacks — information is beautiful," [Online]. Available: http://tinyurl.com/lgyx9lc.

[7] "App42 Cloud APIs: Backend as a Service," [Online]. Available: http://api.shephertz.com/.

[8] "Backendless: Backend as a Service Platform," [Online]. Available: https://backendless.com/.

[9] "Heroku: Building Location Based Apps with Heroku PostGIS," [Online]. Available: http://tinyurl.com/pd42b5d.

[10] "Geocoda: Geocoding and Spatial Database," [Online]. Available: https://geocoda.com/.

[11] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui, "Exploring Millions of Footprints in Location Sharing Services," in *Proc. of ICWSM*. Menlo Park, CA, USA: AAAI, 2011.

[12] "foursquare for Developers," [Online]. Available: https://developer.foursquare.com/.

[13] E. Yigitoglu, M. L. Damiani, O. Abul, and C. Silvestri, "Privacy-preserving sharing of sensitive semantic locations under road-network constraints," in *Proc. of MDM*, 2012, pp. 186–195.

[14] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: optimal strategy against localization attacks," in *Proc. of the CCS*. ACM, 2012, pp. 617–627.

[15] C. Ardagna, G. Livraga, and P. Samarati, "Protecting privacy of user information in continuous location-based services," in *Proc. of CSE*, 2012, pp. 162–169.

[16] Z. Riaz and K. Rothermel, "On the privacy of frequently visited user locations," Distributed Systems Department, University of Stuttgart, Tech. Rep., 2015. [Online]. Available: http://tinyurl.com/nor5bjs

[17] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. of ICML*. ACM, 2006, pp. 161–168.

[18] "Tor Project: Anonymity Online," [Online]. Available: https://www.torproject.org/.

[19] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Personal Ubiquitous Computing*, vol. 18, pp. 163–175, 2014.

[20] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Pervasive Computing*, ser. LNCS. Springer Berlin Heidelberg, 2005, vol. 3468, pp. 152–170.

[21] M. Xue, P. Kalnis, and H. Pung, "Location diversity: Enhanced privacy protection in location based services," in *Location and Context Awareness*, ser. LNCS. Springer Berlin Heidelberg, 2009, vol. 5561, pp. 70–87.

[22] J. Krumm, D. Rouhana, and M.-W. Chang, "Placer ++: Semantic place labels beyond the visit," in *Proc. of PerCom*, 2015, pp. 11–19.

[23] F. Dürr, P. Skvortsov, and K. Rothermel, "Position sharing for location privacy in non-trusted systems," in *Proc. of PerCom*, 2011, pp. 189–196.

[24] D. Riboni and C. Bettini, "Differentially-private release of check-in data for venue recommendation," in *Proc. of PerCom*, March 2014, pp. 190–198.

[25] J.-D. Zhang, G. Ghinita, and C.-Y. Chow, "Differentially private location recommendations in geosocial networks," in *Proc. of MDM*, vol. 1, July 2014, pp. 59–68.