Protecting Against Velocity-based, Proximity-based and External Event Attacks in Location-Centric Social Networks

GABRIEL GHINITA, University of Massachusetts Boston, USA
MARIA LUISA DAMIANI, University of Milan, Italy
CLAUDIO SILVESTRI, University of Venice, Italy
ELISA BERTINO, Purdue University, USA

Mobile devices with positioning capabilities allow users to participate in novel and exciting location-based applications. For instance, users may track the whereabouts of their acquaintances in location-aware social networking applications, e.g., Foursquare. Furthermore, users can request information about landmarks in their proximity. Such scenarios require users to report their coordinates to other parties, which may not be fully trusted. Reporting precise locations may result in serious privacy violations, such as disclosure of lifestyle details, sexual orientation, etc. A typical approach to preserve location privacy is to generate a *cloaking region (CR)* that encloses the user position. However, if locations are continuously reported, an attacker can correlate CRs from multiple timestamps to accurately pinpoint the user position within a CR.

In this work, we protect against a broad range of attacks that breach location privacy using knowledge about: *(i)* maximum user velocity; *(ii)* external events that may occur outside the process of self-reporting locations (e.g., social network posts tagged by peers); and *(iii)* information about mutual proximity between users. Assume user u who reports two consecutive cloaked regions A and B. We consider two distinct protection scenarios: in the first case, the attacker does not have information about the sensitive locations on the map, and the objective is to ensure that u can reach *some* point in B from *any* point in A. In the second case, the attacker knows the placement of sensitive locations, and the objective is to ensure that u can reach *any* point in B from *any* point in A. We propose spatial and temporal cloaking transformations to preserve user privacy, and we show experimentally that privacy can be achieved without significant quality of service deterioration.

Categories and Subject Descriptors: H.2.0 [**General**]: Security, integrity, and protection; H.2.8 [**Database applications**]: Spatial databases and GIS

General Terms: Design, Experimentation, Security

Additional Key Words and Phrases: Location Privacy, Location-aware Social Networks

1. INTRODUCTION

The latest generation of social networking applications (e.g., Foursquare, Facebook Places) enable users to share information about their geo-spatial context. Participants connect to the network using mobile devices with positioning capabilities, and are interested in finding friends that are currently in their geographical proximity. For instance, Alice may use such a service to ask a nearby friend to join her for dinner, or to find on-going events close to her location. Many other similar application scenarios exist, in which users can benefit from sharing their location data. However, serious location privacy concerns arise, which need to be addressed for such applications to gain wide-spread popularity.

Consider that Alice is scheduled for a medical appointment at a hospital situated in the down-town area. Immediately after her appointment, she plans to go to a shopping mall nearby, and would like to know if any of her acquaintances who are currently in the down-town area are interested in joining her. Nevertheless, Alice does not want to disclose her exact coordinates (i.e., hospital), because other service users may learn that she suffers from a medical condition. However, she has no objection in letting her buddies know that she is in the down-town area, or within the boundaries of a region spanning several city blocks. Therefore, a coarser-grained *cloaking region (CR)* may be safe to disclose, as long as certain user-specified privacy constraints are satisfied. On the other hand, CRs should not be excessively large, since this would affect the *accuracy* of location-dependent services.

Location cloaking [Gruteser and Grunwald 2003; Gedik and Liu 2005; Gruteser and Liu 2004; Mokbel et al. 2006; Kalnis et al. 2007; Damiani et al. 2010] is a commonly-used approach to protect the privacy of users that access location-based services. Exact coordinates are replaced with a CR which encloses the user and satisfies a privacy constraint. Privacy requirements are specified by

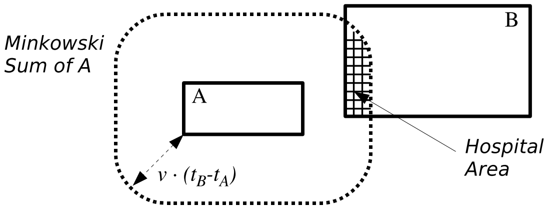


Fig. 1. Location privacy is breached using maximum velocity information

the user's profile, and typically implement a privacy paradigm, such as spatial k-anonymity (SKA) [Kalnis et al. 2007]. SKA is the most prominent location privacy paradigm proposed so far, and aims to protect the privacy of users who ask spatial queries, such as "find the nearest restaurant to my location". In such scenarios, the objective is to protect the exact *identity* of the user who is issuing the query, and the constraint imposed is that each CR must contain at least k distinct users. This way, the probability of identifying the querying user is bounded above by $1/k$. However, in our application context the identity of the user is known, and the objective is to protect the exact location of the user. Furthermore, in areas where the density of users is high (e.g., down-town) k users can be found in close proximity to each other, and the CR can have a small extent (e.g., all k users could be inside the hospital). Therefore, SKA is not applicable to the considered scenario.

A more appropriate protection model is the one from [Gruteser and Liu 2004; Damiani et al. 2010], where the aim is to prevent an attacker from pinpointing exact user coordinates. For instance, in the PROBE system [Damiani et al. 2010] all locations on the map are represented as *features*, and each feature has a type. Certain feature types are sensitive (e.g., hospitals, bars), whereas others are innocuous (e.g., shopping centers, parks). Each user defines his/her own privacy profile, which specifies sensitivity thresholds with respect to each feature type. PROBE generates CRs that cover a mix of sensitive and innocuous regions, such that the association probability between the user and sensitive features is bounded below the specified threshold. However, previous work does not address *linkage* attacks, which can be easily staged in practice by correlating CRs reported at consecutive timestamps.

Velocity-based Attacks. The first type of privacy threat we address in this paper is that of *velocity-based linkage attacks* that rely on knowledge about maximum velocity to pinpoint the exact user coordinates within a reported cloaking region. Consider the example of Figure 1, where Alice reports her (cloaked) location as she moves. We only show two consecutive time snapshots, with corresponding CRs A (issued at time t_A) and B (issued at time t_B, where $t_B > t_A$). Assume that Alice has set her current on-line status to *"Visiting shops in the down-town area"*. An attacker can infer with high probability that Alice is currently walking, hence her velocity can be no higher than 5 km/h. Alternatively, if Alice's status is *"Out for a bicycle trip"*, her speed can be bound to at most 20 km/h. The attacker first determines the Minkowski sum [de Berg et al. 2000] around CR A with enlargement $v \cdot (t_B - t_A)$, where v is the inferred maximum user velocity. Next, the Minkowski enlargement is intersected with the CR B, and the attacker infers that Alice must be situated in the hatched sub-region of B, since she could not have physically reached any farther position. If a hospital building is situated in the hatched region, then the attacker can infer that Alice has a medical appointment, compromising her privacy.

We consider two different protection scenarios:

(i) *Preventing disclosure of exact user coordinates.* This protection scenario aims to prevent attackers from using reported locations to stalk, or physically assault a service user.

(ii) *Bounding the association probability of a user with a sensitive feature.* The objective of this protection scenario is to prevent attackers from learning private details about a user's health condition, religious affiliation, etc.

We formalize both attack models, and propose solutions that generate CRs which are not vulnerable to linkage attacks. We also take into account the resource limitations of mobile users. Specifically, we consider solutions where CRs are generated in an off-line phase, and no significant overhead is incurred by the user to compute CRs on-line. Reporting of pre-defined regions is temporally cloaked, in order to prevent linkage attacks. We also introduce techniques that can generate CRs on-line, if enough resources are available to the user (or if some trusted service is employed for this purpose). The advantage of the latter approach is that CRs are customized to the current user location, leading to better accuracy of provided services.

External Event-based Attacks. Even if the CRs of a user are safe with respect to the self-reported history of cloaked locations, an adversary may still breach location privacy by correlating the self-reported history with external events, such as geo-tagged social media posts. Such posts may be created by the users themselves, or by their peers. The posting of geo-tagged objects such as images, videos, etc., can be used in conjunction with anonymized user whereabouts to violate location privacy. We propose a novel algorithm that addresses this threat, and achieves an interesting trade-off between quality of service and user satisfaction. Our algorithm blocks the publication of geo-tagged items when they pose an immediate privacy threat, but on the other hand over-provisions the amount of protection such that the total amount of blocked publication occurrences is minimized, hence improving user experience.

Mutual Proximity Attacks. We also consider the case where a user wishes to keep private her mutual proximity relationship with another user. Specifically, pairs of users coordinate their anonymized updates, such that an adversary is not able to infer that the two came in close proximity to each other (i.e., they had a secret meeting). This additional level of protection is achieved without the two involved users having to share any additional location information to each other, except for what they would have released in the absence of the additional proximity constraint, and the immediate information they gain from meeting each other.

The rest of the paper is organized as follows: In Section 2, we investigate related work. We formalize the two alternative attack models and protection scenarios in Section 3, and we introduce the system architecture in Section 4. In Section 5, we present our defense strategies consisting of spatial and temporal transformations. In Section 6, we extend spatial and temporal transformations to withstand more complex attacks such as attacks based on external events (Section 6.1) and proximity-based attacks (Section 6.2). We evaluate experimentally the proposed techniques in Section 7, and we conclude with directions for future research in Section 8.

2. RELATED WORK

This section discusses research results closely related to our work. For a more general survey on location privacy in the context of location-based services and mobile applications we refer the reader to [Krumm 2009; Ghinita 2013; Damiani 2014].

Location cloaking was extensively studied in the context of private spatial queries. Typically, users ask nearest-neighbor queries to servers that own databases with points of interest (e.g., restaurants). However, users wish to keep their exact locations private. In [Kido et al. 2005; Yiu et al. 2008], the querying user discloses one or more fake locations to the server. However, these locations could still fall within sensitive areas. Furthermore, attacks through correlation of multiple reported locations are not addressed. A considerable number of location privacy solutions [Gruteser and Grunwald 2003; Gedik and Liu 2005; Kalnis et al. 2007; Mokbel et al. 2006] rely on the spatial k-anonymity (SKA) paradigm, and generate CRs that contain at least k distinct users. However, the focus of all these approaches is on protecting user *identity*, not location. As a result, it is still possible that the resulting cloaking regions have small extent. Furthermore, the CRs may be completely enclosed within sensitive areas on the map. The position paper in [Shokri et al. 2010] summarizes SKA limitations.

More relevant to our work is the protection model in [Damiani et al. 2010; Damiani et al. 2011; Yigitoglu et al. 2012], which aims to hide exact user coordinates, and to prevent association with sensitive locations. In the PROBE system [Damiani et al. 2010], users define their own privacy profiles, by specifying maximum thresholds of association with sensitive feature types. Our privacy model for the scenario of an attacker with background knowledge on map locations is similar to [Damiani et al. 2010]. An alternative model to PROBE is introduced in [Xu and Cai 2009], where a feeling-based measure of privacy protection is proposed. Specifically, the work in [Xu and Cai 2009] defines safety with respect to a single CR based on the level of popularity of enclosed regions. Intuitively, the more popular (i.e., frequently visited) a region is, the more safe it is. A metric that is based on entropy is used to quantify safety. The advantage of this approach is that it does not require users to specify thresholds, which increases usability. We emphasize that, the method in [Xu and Cai 2009] focuses on a single CR, and does not address correlation across multiple updates. Hence, the method is orthogonal to our velocity-based attack protection approach, and in fact our proposed techniques for enforcing consecutive CR safety can be used in conjunction with the single-CR safety condition specified in [Xu and Cai 2009].

Another category of approaches addresses private location queries by encrypting user coordinates. For instance, the work in [Khoshgozaran and Shahabi 2007] employs a geometrical transformation to map locations to the one-dimensional space, and processes queries in the transformed domain. The technique in [Ghinita et al. 2008] employs cryptographic private information retrieval (PIR) protocols, and provides strong privacy guarantees. In an off-line phase, the database of points of interest is organized according to the type of query supported (e.g., nearest-neighbor). At query time, a cryptographic protocol is executed that allows the user to retrieve the requested objects. However, this method is not suitable for the studied problem, since it assumes static data, whereas in our case the user locations (which are the objects of interest) change frequently. Furthermore, PIR incurs high computational and communication overhead.

The closest to our work is the method in [Cheng et al. 2006], where a random cloaking region that encloses the user is generated. The resulting area represents an *uncertainty* region, which prevents the attacker from learning the exact user location. The authors also discuss linkage attacks based on knowledge about maximum velocity, and they propose two solutions: *patching* and *delaying*. Patching reports the union between the current CR and the one reported in the previous timestamp. However, the resulting area may not be contiguous, and can grow very large. Furthermore, the method can be easily reverse-engineered, since the attacker can know that the union was performed due to an imminent vulnerability to linkage. Delaying may incur severe service deterioration due to dropped service requests, as we show in our experimental evaluation. In contrast to [Cheng et al. 2006], we also *postdate* requests, which provides zero request drop ratio, and we take into account scenarios when the attacker has prior knowledge about the placement of sensitive regions on the map. Another feature of our privacy mechanism is that it can be applied in different and possibly constrained spatial contexts. For example, in [Yigitoglu et al. 2012] the technique is deployed to protect the privacy of users moving along a road network. A simpler approach to semantic location cloaking over road networks is presented in [Li et al. 2016]. In this case, the protection goal is to prevent exclusively the semantic homogeneity attacks over road networks through semantic l-diversity, while velocity-dependent threats are ignored.

The line of work in [Shokri et al. 2011; Shokri et al. 2012; Theodorakopoulos et al. 2014] assumes a Bayesian adversary and proposes formal techniques for location protection. Specifically, a probabilistic privacy metric to quantify privacy is introduced in [Shokri et al. 2011]. In [Shokri et al. 2012], protecting privacy is formulated as an optimization problem, where the goal is to maximize the distance between the location guessed by an adversary and the actual user location, while minimizing the loss in quality of service. In [Theodorakopoulos et al. 2014], the approach from [Shokri et al. 2012] is extended to cope with correlated location updates. It is assumed that the background knowledge of the adversary can be fully captured using a Markov model, and a Stackelberg game strategy is employed for protection. The protection algorithm constructs a state graph, and assumes that every movement and action of the user and adversary is modeled as a transition in this graph. The computational cost of the solution is very high, and all computations must be done offline, in advance, for every possible trajectory and adversary action. In the experiments in [Theodorakopoulos et al. 2014], only a small number of locations is used, in the order of 200. Furthermore, the assumption that every single action can be fit to the graph representation, and most importantly, that every action can be eventually quantified using a numerical metric, may be difficult to meet in practice. In contrast, our approach is computationally light, the algorithms are executed online, and we do not require strict conditions on the model of movement, adversary actions, etc.

Several privacy threats due to user co-location are discussed in [Freni et al. 2010; Ruiz-Vicente et al. 2011; Olteanu et al. 2014]. In particular, in [Freni et al. 2010], the attacks take place when geo-tagged resources and identities are explicitly shared by users who obfuscate their location before disclosing it to the members of the community and to the service provider. Such attacks exploit two kinds of information: the linkages that exist among the locations of multiple users, and the fact that co-located users exhibit different privacy profiles, thus their cloaked regions are of different size, leading to privacy breaches. The protection technique from [Freni et al. 2010] relies on a centralized trusted server. Upon a concurrent service request from a set of users, the trusted server collects the users' privacy profiles and computes a cloaked region in compliance with the privacy preferences of all users, possibly adding necessary corrections to prevent linkage attacks. The use of a centralized trusted server has a number of shortcomings in terms of cost, performance, and security. In our work, we present a more flexible approach that does not require any trusted server. In addition, we address two novel privacy threats that have not been addressed in previous work, namely protection in the presence of external asynchronous events, and hiding mutual proximity. The latter threat in particular is specifically challenging for geo-social networks, as acknowledged in [Ruiz-Vicente et al. 2011].

Finally, the recently-proposed concept of geo-indistinguishability [Chatzikokolakis et al. 2013; Andrés et al. 2013] provides a mechanism to randomly perturb locations, and offers quantitative measures for the probability of an adversary to recover the real location from a reported one. Geo-indistinguishability is inspired from the powerful model of differential privacy (DP) [Dwork 2006], which in recent years became the de-facto standard for privacy-preserving data publishing. However, while borrowing some of the syntactic transformations of differential privacy, the work in [Chatzikokolakis et al. 2013; Andrés et al. 2013] does not also inherit the powerful protection semantics of DP, which only permits access to data through a statistical query interface, and prevents an adversary from learning whether a particular data item is included in a dataset or not. In effect, DP is not applicable to operational tasks, as our problem setting requires. Geo-indistinguishability, on the other hand, allows one direct access to perturbed data, and does offer some guarantees against exact location disclosure, but does not address velocity-based attacks in the case of repeated updates from the same user. In addition, if the user's location is frequently reported, the privacy degrades due to the correlations between locations [Chatzikokolakis et al. 2014].

3. PRIVACY AND THREAT MODELS

3.1. Preliminaries

Consider mobile user u who follows a trajectory

$$T = \{(p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n)\},$$

where $p_i = (x_i, y_i)$ is the two-dimensional point location of user u at timestamp t_i. A location *snapshot* at a given time t is defined as the tuple $s = (p, t)$. Denote by $\mathcal{S}$ the set of location snapshots associated with the user trajectory, $\mathcal{S} = \{s_i\}_{1 \leq i \leq n}$. These snapshots may be equidistant in time (i.e., $|t_i - t_{i+1}| = |t_i - t_{i-1}|$), for instance as a result of periodic location updates from a GPS device. However, this is not a requirement, and we consider that the time duration between two consecutive snapshots is arbitrary, and is decided by the user.

Users *continuously* report location information corresponding to each snapshot in $\mathcal{S}$. For instance, in a social networking application, participants constantly update their location so they can be tracked by their friends. In the case of location-based queries, users report location data to a server to retrieve nearby points of interest. In the latter case, each snapshot has an associated query parameter, specifying the query type (e.g., nearest-neighbor) as well as the requested object type (e.g., restaurant). To simplify the terminology, we denote by *request* the event of location reporting in both cases.

Due to privacy considerations, snapshots are not disclosed in their original form. Instead, at each timestamp t_i, the user location is protected using a cloaking region CR_i that replaces exact coordinates. The CR is generated according to the user privacy profile (Section 3.4 properly defines privacy requirements). Furthermore, to prevent linkage attacks, the resulting CR may be reported at a timestamp t'_i which is different than t_i specified in $\mathcal{S}$. We denote the *reported* set of cloaked location snapshots by $\mathcal{S}' = \{(CR_i, t'_i)\}_{1 \leq i \leq n}$.

The set $\mathcal{S}'$ represents the *attacker view*, i.e., the attacker has access to all CRs reported by a user, as well as their reporting timestamps. However, the attacker does *not* know the set $\mathcal{S}$. The attacker also has knowledge about the maximum user velocity v. Our objective is to prevent the inference of additional location information that is not included in $\mathcal{S}'$. Privacy is protected through spatial transformations (i.e., carefully choosing the CR extents), as well as temporal transformations (i.e., deciding when to report CRs).

3.2. Quality of Service

Users report location information in order to facilitate meaningful interactions with their friends, or with entities that provide services tailored to the users' geo-spatial context. Therefore, privacy protection should maintain a good quality of service provided to users. Ideally, the location reporting should be performed in a timely manner (i.e., $|t_i - t'_i|$ should be minimized), and the resulting CRs should have small spatial extent (subject to fulfillment of privacy constraints). Typically, the exact user location p_i is enclosed by CR_i, to provide consistency with the user's geo-spatial context. However, as will be discussed in Section 5.1, we allow users to report some past location if doing so allows them to obtain good quality service without compromising privacy.

Given the two sets of original ($\mathcal{S}$) and reported ($\mathcal{S}'$) snapshots, we define four metrics that characterize the loss in service quality due to privacy protection:

— *CR size*: CRs with large areas may decrease the usability of the reported information. The CR size metric Q_{CR} is defined as the average area of reported CRs

$$Q_{CR} = \frac{1}{n} \cdot \sum_{i=1}^{n} Area(CR_i)$$

— *time error*: the proposed privacy-preserving solutions may report a location snapshot at a different timestamp than its original one. Specifically, snapshots can be *delayed*, i.e., $t'_i > t_i$. Time error Q_T is defined as

$$Q_T = \frac{1}{n} \cdot \sum_{i=1}^{n} |t_i - t'_i|$$

Note that, if the original requests have equi-distant timestamps t_i, an attacker may attempt to use the delay information to compromise privacy. In such a case, the original request sequence can be modified such that the distance between consecutive timestamps is randomized.

— *space error*: when users report CRs built around past locations, it may happen that the current user location p_i falls outside the reported cloaked location CR_i. Disclosing information that is not completely up-to-date may still be useful (e.g., a user may learn the address of a nearby restaurant, even if this is not the closest restaurant). We measure such loss of accuracy using the space error metric, formally defined as:

$$Q_S = \frac{1}{n} \cdot \sum_{i=1}^{n} d(p_i, CR_i)$$

where d measures the Euclidean distance between p_i and its closest point in CR_i.

— *failure ratio*: it is possible that certain snapshots are never reported, e.g., due to the impossibility of finding a CR that satisfies the privacy requirements. Failure ratio is defined as

$$FR = \frac{Dropped\ Requests}{Total\ Requests}$$

All other metrics are computed only for reported CRs.

In addition to the performance metrics, we also consider an additional constraint placed by the user on the maximum delay for location reporting. For instance, if a user asks a location query, s/he may be willing to wait only for a relatively short time to get the answer (e.g., a typical acceptable response time may be 5sec). On the other hand, when sharing location data with friends, the users may accept a larger delay (e.g., 60sec). We define the $MaxDelay$ parameter that specifies the maximum amount of time that a request can be delayed. Specifically, if ($t'_i - t_i > MaxDelay$), then the i^{th} request is considered as failed.

3.3. Distance Metrics for Cloaking Regions

The example of Figure 1 showed how certain geometrical properties of consecutive CRs can be exploited by an attacker. We introduce two distance metrics between cloaking regions, the *Hausdorff* distance and the *point-pairwise* distance, that are fundamental to the studied attack models and proposed defenses. We measure the distance between two-dimensional points p' and p'' using the Euclidean distance, denoted by $d(p', p'')$.

Hausdorff Distance. Consider two cloaking regions[1] A and B. The Hausdorff distance [Atallah 1998] between CRs A and B is formally defined as:

$$d_{haus}(A, B) = \max\{h(A, B), h(B, A)\},$$

where

$$h(A, B) = \max_{p' \in A} \min_{p'' \in B} d(p', p'')$$

$h(A, B)$ represents the *direct* Hausdorff distance, which measures the maximum distance between *any* point in A to *some* point in B. $h(B, A)$ is symmetrically defined. Figure 2 shows an example of distance calculation. The largest distance between any point in A and some point in B is equal to the distance from the left side of A to the left side of B, hence $h(A, B) = 9$. Similarly, $h(B, A) = 12$, hence $d_{haus}(A, B) = \max(9, 12) = 12$.

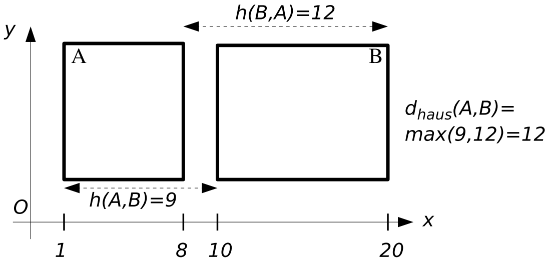


Fig. 2. Example of Hausdorff distance computation

Since rectangles are convex shapes, the Hausdorff distance between two rectangles can be efficiently evaluated, by computing the Euclidean distances between the rectangles' corners and sides.

[1] We consider rectangular CRs only. However, Hausdorff distance applies to other polygonal shapes as well.

Point-Pairwise Distance. Point-pairwise distance between CRs A and B measures the maximum distance between *any* point in A to *any* point in B. Formally,

$$d_{pp}(A,B) = \max_{p' \in A} \max_{p'' \in B} d(p', p'')$$

Figure 3 gives the pseudocode to compute d_{haus} and d_{pp}.

```
**HausdorffDistance(A,B)**
Input: CRs $A$ and $B$ with corners $a_{1..4}$, $b_{1..4}$ and sides $l^A_{1..4}$, $l^B_{1..4}$
Output: Hausdorff distance value
$dist_{AB} = dist_{BA} = 0$
for $i$ := 1 to 4
  $d' = \min_{j=1..4} d(a_i, b_j)$, $d'' = \min_{j=1..4} d(b_i, a_j)$
  for $k$ := 1 to 4
    let $q$ be the projection of $a_i$ on $l^B_k$, $q'$ the projection of $b_i$ on $l^A_k$
    /*projection is considered with respect to a line, not a segment*/
    if ($q \in l^B_k$) then $d' = \min\{d', d(a_i, q)\}$
    if ($q' \in l^A_k$) then $d'' = \min\{d'', d(b_i, q')\}$
  $dist_{AB} = \max\{dist_{AB}, d'\}$, $dist_{BA} = \max\{dist_{BA}, d''\}$
return $\max\{dist_{AB}, dist_{BA}\}$

**Point-PairwiseDistance(A,B)**
Input: CRs $A$ and $B$ identified by corners $a_{1..4}$, $b_{1..4}$
Output: Point-pairwise distance value
return $\max_{i,j=1..4} d(a_i, b_j)$
```

Fig. 3. Computing distances between cloaking regions

3.4. Attack Models and Privacy Requirements

We consider two distinct application settings, depending on whether or not the attacker has background knowledge about the sensitive locations on the map. Next, we define the attack models for both of these settings, and we formalize the privacy requirements. We also give sufficient *safety* conditions that must be met in order to ensure that reported CRs do not compromise privacy.

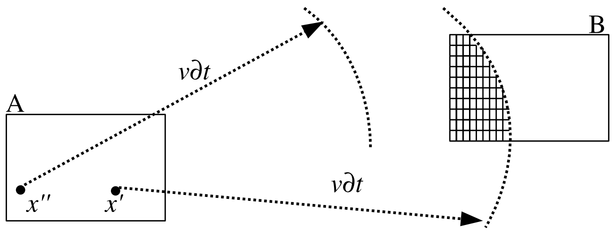


Fig. 4. Attack model without background knowledge

3.4.1. Attacker without Knowledge of Sensitive Locations. In this setting, the privacy objective is to prevent the disclosure of precise locations, which may result in physical threats to the user, e.g., stalking or assault [Fox News]. Consider, for instance, the well-established division of U.S. territory into zip-code areas. The map is partitioned into disjoint regions, each of them covering an area of a few square miles. Or, at a finer granularity level, a city can be sub-divided into block regions. As the user moves, his/her location can be mapped to a city block identifier, and only the block identifier is disclosed. The *privacy requirement* in this case is not to allow an attacker to pinpoint the user location within a sub-region of a reported CR.

Figure 4 shows an example of two CRs A and B which are reported by user u at timestamps t_A and t_B, respectively. Without loss of generality, let $t_A < t_B$. Denote by v the maximum user velocity, and let $\delta t = |t_B - t_A|$.

The attacker may try to prune parts of A and B to pinpoint u in two ways:

(i) determine if there is any location $x \in A$ from which the user cannot reach some location $y \in B$, even by traveling at maximum speed v. Formally, an attack is successful iff.

$$\exists x \in A \; s.t. \; \forall y \in B, \; d(x, y) > v\delta t \tag{1}$$

In Figure 4, a user traveling from point x' is able to reach a point in the hatched region of B within time δt. However, if the initial location of u were x'', reaching B would not have been possible. Therefore, an attacker can rule out a subset of A as possible positions for u, hence privacy is breached

(ii) determine if there is any location $y \in B$ which the user cannot reach from some initial location $x \in A$, even by traveling at maximum speed v. Formally,

$$\exists y \in B \; s.t. \; \forall x \in A, \; d(x, y) > v\delta t \tag{2}$$

To prevent privacy breaches, we need to ensure that none of Eq. (1) or (2) ever holds. Note that, according to the definition of Section 3.3, this is equivalent to stating that the Hausdorff distance $d_{haus}(A, B) \leq v\delta t$.

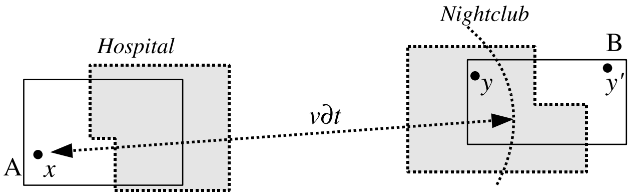


Fig. 5. Attack model with background knowledge

3.4.2. Attacker with Knowledge of Sensitive Locations. In practice, the attacker may have access to a map containing the placement of sensitive locations. We adopt the privacy profile proposed in [Damiani et al. 2010], where each object on the map is abstracted as a *feature*. Each feature has a type, (e.g., restaurant, park, etc). Some types are innocuous (e.g., shopping malls), whereas others are sensitive in nature (e.g., hospitals).

Denote by $\mathcal{FT} = \{ft_1, \ldots, ft_m\}$ the set of feature types. Users specify their profiles as an array $\mathcal{P} = \{thr_1, \ldots, thr_m\}$ of thresholds, where thr_j represents the maximum allowed probability of association between a user and a sensitive feature of type ft_j. The *privacy requirement* in this case dictates that the association probability between a user and a sensitive feature type must not exceed the user-specified threshold. Given CR A, the probability of association is equal to the area of the sub-region in A covered by sensitive features of type ft_j divided by the entire area of A. Formally, a CR satisfies privacy if

$$\forall \; i = 1..m, \frac{\sum_{\forall f \in ft_i} Area(f \cap CR)}{Area(CR)} \leq thr_i$$

Consider the example in Figure 5. There are two feature types (shown shaded), $Hospital$ and $Nightclub$, and the user has specified a threshold of 0.5 for both types. Each of the two CRs taken individually does satisfy the privacy requirement. However, if the attacker uses information about maximum user velocity, the disclosure of both A and B violates privacy. For instance, the presence of user u in an innocuous location $x \in A$ precludes u from being located at innocuous location $y' \in B$. Instead, u must be inside some sensitive area within B (e.g., at point y). The privacy breach can be formalized as

$$\begin{array}{l} \exists x \in A | (x\ is\ non-sensitive)\ s.t. \\ \nexists y \in B | (y\ is\ non-sensitive \wedge d(x,y) < v\delta t) \end{array}$$

Symmetrically, a breach occurs if

$$\begin{array}{l} \exists y \in B | (y\ is\ non-sensitive)\ s.t. \\ \nexists x \in A | (x\ is\ non-sensitive \wedge d(x,y) < v\delta t) \end{array}$$

To prevent privacy breaches, we must ensure that the user can be located outside sensitive areas at both timestamps. Since in the worst case the safe regions in two distinct CRs can be situated in their two opposite corners, a sufficient condition to ensure disclosure safety is $d_{pp}(A,B) \leq v\delta t$.

3.4.3. Location Disclosure Safety Condition

Definition 3.1. Two cloaked regions A and B separated by time interval δt are *safe to disclose* in the attack model without background knowledge if $d_{haus}(A,B) \leq v\delta t$. Similarly, in the attack model with background knowledge the two regions are *safe to disclose* if $d_{pp}(A,B) \leq v\delta t$.

3.4.4. Transitivity of the Safety Property. So far, we considered the safety property with respect to a pair of CRs. However, in our attack models, the entire set of reported CRs is available to an attacker. We show that the location disclosure safety property is transitive for both attack scenarios. This is an important result, since it means that at any time it is sufficient to check whether consecutive CRs are safe to disclose. Then, by induction, any pair of reported CRs are safe. However, in the CR generation algorithms, we only need to consider the most recent CR, which decreases considerably the computational overhead of the proposed solutions.

LEMMA 3.2. *Let A, B and C be three CRs disclosed at timestamps t_A, t_B, t_C, such that $t_A < t_B < t_C$. Then, if the pairs of CRs (A,B) and (B,C) are safe to disclose, so is the pair (A,C).*

Proof: We prove the transitivity property for both CR distance metrics.

Case 1 (d_{haus}):

The proof is by contradiction. Assume that the pair (A,C) is not safe to disclose. Therefore, $d_{haus}(A,C) > v(t_C - t_A)$. On the other hand, by hypothesis, the pairs (A,B) and (B,C) are safe to disclose, therefore

$$d_{haus}(A,B) \leq v(t_B - t_A)$$

and

$$d_{haus}(B,C) \leq v(t_C - t_B)$$

Adding the two inequalities term by term, we obtain that

$$d_{haus}(A,B) + d_{haus}(B,C) \leq v(t_C - t_A)$$

However, the Hausdorff distance satisfies the triangle inequality [Henrikson 1999]

$$d_{haus}(A,B) + d_{haus}(B,C) \geq d_{haus}(A,C)$$

therefore

$$d_{haus}(A,C) \leq v(t_C - t_A)$$

We obtain a contradiction, therefore the pair (A,C) must be safe to disclose.

Case 2 (d_{pp}):

Let $x \in A$, $y \in B$ be two point locations s.t. $d_{pp}(A,B) = d(x,y)$. It is guaranteed that such two points exist from the definition of d_{pp}. Similarly, let $y' \in B$, $z \in C$ s.t. $d_{pp}(B,C) = d(y',z)$ and let $x' \in A$, $z' \in C$ s.t. $d_{pp}(A, C) = d(x', z')$. Take any random point $p \in B$. From the triangle inequality, we have that

$$d(x', z') \leq d(x', p) + d(p, z')$$

On the other hand,

$$d(x', p) \leq d(x, y) \wedge d(p, z') \leq d(y', z)$$

It follows immediately that

$$d_{pp}(A, C) \leq d_{pp}(A, B) + d_{pp}(B, C)$$

and since, by hypothesis,

$$d_{pp}(A, B) \leq v(t_B - t_A) \wedge d_{pp}(B, C) \leq v(t_C - t_B)$$

it results that

$$d_{pp}(A, C) \leq v(t_C - t_A)$$

Therefore, disclosure safety is transitive with respect to d_{pp} as well. $\square$

3.5. Privacy Discussion

Our work focuses on attacks that an adversary may stage using information on mobile user velocity. In our view, such attacks are a very real threat in practice, and it is important to have protection techniques that can mitigate such threats. There are, however, other types of attacks which, although important, are outside the focus of our work. Solutions to such attacks are often orthogonal to our proposed techniques, and can be integrated with our approach without major modifications.

In our approach, we consider that the feature types are static, i.e., they do not evolve over time. In some practical scenarios, the characteristics of a certain feature type can change with time, possibly on a cyclical basis. For instance, a store that would have been safe to include in a CR at 3pm becomes unsafe at 9pm, as the adversary may know that the store is closed, and will be able to infer that the user must actually be elsewhere within the CR. Our methods can be easily extended to address this case, by dynamically updating the feature map. For instance, a system component orthogonal to our techniques may take as input opening times for each map feature, and mark stores as a non-accessible zone outside opening hours. As a result, when computing CR safety, such zones will not count, and a larger CR will be constructed that will include sufficient non-sensitive zones that are also accessible. The feature map is an input to our approach, so no changes to our techniques are necessary.

Another scenario of practical importance is the case when movement is restricted to a road network, as opposed to the free-space movement case which is currently the focus of our work. Note that, the principles of construction for our protection mechanisms remain valid for restricted movement, with the difference that rectangular CRs will become connected sets of graph vertices. The current solution for free space movement remains applicable to many scenarios, especially to city areas, where the road network is dense, and activities such as walking and cycling can be reasonably approximated as free space movement.

4. SYSTEM ARCHITECTURE

Figure 6 shows the proposed system architecture. Users feed their exact location information and privacy profile (Section 3.4) to a cloaking engine, which keeps track of previously-reported CRs and ensures that the disclosure safety condition is met (with respect to the attack scenario considered, i.e., with vs without background knowledge). The proposed architecture is flexible with respect to the deployment of the cloaking engine. Our privacy-preserving transformations are suitable both for a two-tier model, where the user's mobile device performs the cloaking, or a three-tier model, where the cloaking is delegated to a trusted third-party service. Note that, as opposed to spatial k-anonymity techniques [Kalnis et al. 2007] that *require* a trusted anonymizer service to pool large

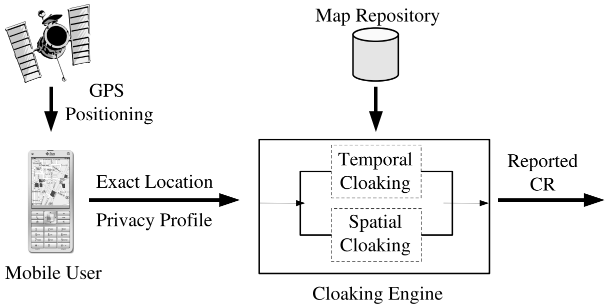


Fig. 6. System Architecture

number of users, our system architecture does not have such a conceptual constraint. However, we consider employing such a service as an alternative that improves performance, if the mobile device does not possess sufficient computational resources. Furthermore, in the case of attacks with background knowledge, the feature maps required to cloak locations may be too large to be stored (and updated) on a mobile device.

5. PRIVACY-PRESERVING ALGORITHMS

In this section, we present the proposed techniques to preserve the privacy of user requests. We consider two transformations[2]:

(1) *Temporal Cloaking.* In some applications, the dataspace (e.g., a city map) is partitioned into a fixed set of regions. We assume that all regions have rectangular shape, i.e., the map is partitioned into a set of tiles. This case fits scenarios where it is expensive to compute CRs on-line, or when the splitting is pre-defined (e.g., CRs represent zip-code areas). Consider the example in Figure 7(a), where the tiling is shown with dotted lines. The current time is t_1, and the user lies in CR B. Previously, at time t_0, CR A was disclosed. Assuming that the adversary has background knowledge about the map, CR B can only be disclosed if the distance $d_{pp}(A, B) \leq v\delta t$, where $\delta t = t_1 - t_0$. In the example, the condition does not hold, hence B cannot be issued at time t_1. Instead, the request is delayed. Temporal cloaking is presented in detail in Section 5.1.

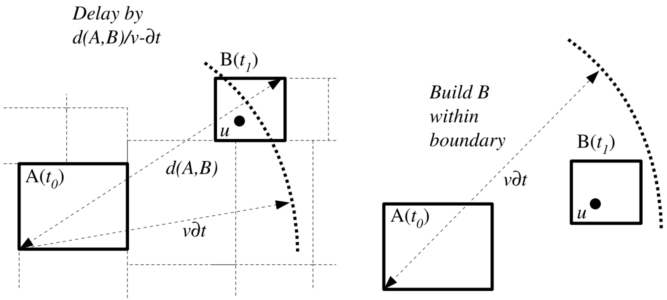
(2) *Spatial Cloaking.* In situations where enough resources exist to compute the CRs on-line, and no requirements for a fixed partitioning exist, the CRs can be constructed in such a manner that the safety property is met. Figure 7(b) shows a potential zone where the CR can be situated, within the $v\delta t$ boundary. A CR construction algorithm can take into consideration the boundary, and find a CR that is safe to disclose. We introduce spatial transformations in Section 5.2.

5.1. Temporal Cloaking

Temporal cloaking is suitable when the partition of the map into CRs is fixed in advance. Note that, since no CR computation is performed on-line, temporal cloaking is particularly suitable to be performed directly on the mobile device. As an additional benefit, performing cloaking on the device itself can make use of supplementary information about the user's trajectory. For instance, if a user is following the instructions of an in-car GPS navigation system, the future trajectory is already known to a considerable extent.

We identify two alternatives for achieving temporal cloaking: request *deferral* and *postdating*. We illustrate these two concepts in Figure 8. Consider user u who wants to issue a request at current time t_q. The location of u is enclosed by CR C. Previously at time t_A, u issued a request with CR

[2] We use the terms *temporal* and *spatial* cloaking to distinguish between the transformations that mainly target the time, respectively the space dimension. We emphasize that even for temporal transformations, the user location is still cloaked with the help of CRs.



(a) Temporal Cloaking (b) Spatial Cloaking

Fig. 7. Approach Overview

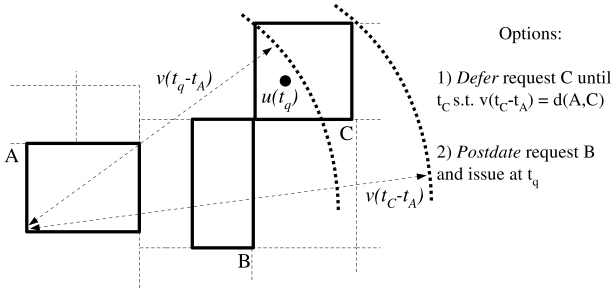


Fig. 8. Temporal Cloaking: *Deferral* vs *Postdating*

A. Prior to entering C, u was situated inside region B, but no request with associated CR B was issued. At current time t_q, C is not safe to disclose[3], as it is too far away from A.

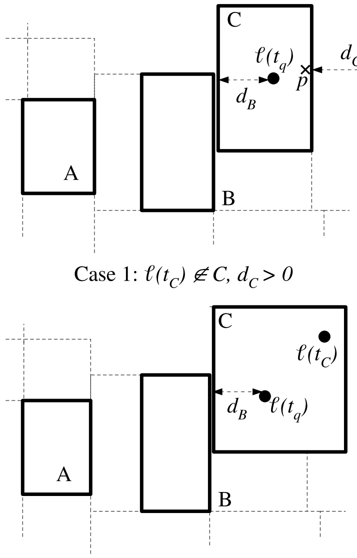
The first option is to *defer* the request until C becomes safe to disclose, i.e., until t_C s.t.

$$d(A, C) \leq v(t_C - t_A),\ t_C \geq t_A + d(A, C)/v,$$

where d can signify either the d_{haus} or d_{pp} distance. In this case, the request is delayed for a period of time equal to $t_C - t_q$. Note that, by that time it is possible that u will no longer be situated inside C, therefore a space error may be incurred. The second alternative is to issue the request immediately at t_q, but using CR B. Note that, since u is already outside B, the request will certainly incur some amount of space error. However, if the current position of u is not far away from B (e.g., u has only recently exited B), the error is likely to be low. We refer to this method as request *postdating*. Note that, it is not always the case that the tile/region visited by u just before its current tile is safe to disclose. Nevertheless, the same idea can be applied with respect to the last safe-to-disclose visited region. Keeping track of such a region is not computationally expensive, and can be done upon the receival of a GPS location update. Furthermore, the storage requirement is $O(1)$, since only one such region is maintained.

With the deferral and postdating primitives defined, we next devise a strategy that combines the two methods in order to maintain good QoS. We propose an heuristic that chooses the best of the two based on benefit estimation. Using the same convention as earlier, let A be the CR of the last issued request (which occurred at time t_A), let C be the CR currently enclosing the user, and let B be the last safe region visited before C. Note that, the existence of B is always guaranteed, as in the worst case we have $A = B$. We also assume that the user has the ability to predict (with

[3] In this example, we consider an attacker with background knowledge, and we measure safety with respect to d_{pp}. However, this scenario applies to d_{haus} as well.



Case 1: $\ell(t_C) \notin C$, $d_C > 0$

Case 2: $\ell(t_C) \in C$, $d_C = 0$

Fig. 9. Space error as a result of temporal cloaking

reasonable accuracy) its position at a future time. This prediction will be used in evaluating whether deferral or postdating is more beneficial. Since the proposed solution is an heuristic in the first place, predicting future locations with high accuracy is not a necessity. A low prediction accuracy will, however, affect the obtained QoS.

Let t_C be the time when C becomes safe to disclose, and denote by $\ell(t_C)$ the estimated position of u at time t_C. If the request is deferred, two distinct situations arise, as shown in Figure 9:

(1) $\ell(t_C) \notin C$. In this case, by the time C becomes safe, the user would have already exited C. Denote by d_C the minimum distance from $\ell(t_C)$ to any point in C, and let p be the point that minimizes that distance. Then, d_C represents an upper bound on the inaccuracy of the location reporting. For instance, if the request represents a nearest-neighbor query, and the NN of p is situated at distance r from p, then according to the triangle inequality, the NN of p will be at distance at most $r + d_C$ from $\ell(t_C)$. Therefore, the space error d_C is a good measure of the QoS deterioration due to privacy enforcement.

(2) $\ell(t_C) \in C$. In this case, the user is still inside C at time t_C, therefore no space error is incurred.

The time error is $t_C - t_q$ in both cases mentioned above. If the request is postdated (i.e., CR B is issued at time t_q), the time error is 0, and the space error is d_B.

Figure 10 details the proposed heuristic. The *TemporalCloaking* routine gets invoked at every timestamp t_q when a request is scheduled. The heuristic will determine (line 6) whether deferring the request exceeds the user-specified delay threshold $MaxDelay$ (defined in Section 3.2). If the threshold is exceeded, then the request is postdated (line 7). Otherwise, the distances d_B and d_C are compared (line 8), to determine whether it is more beneficial in terms of space error to issue B or C. The closest of the two CRs to the location of u is selected, and the request is postdated or deferred depending on the comparison outcome.

5.2. Spatial Cloaking

When the user's mobile device has sufficient resources, or when cloaking is performed by a trusted service, CRs can be dynamically computed at the time of the request. The advantage of such an

```
**TemporalCloaking**
Input: request timestamp $t_q$, location of requester $u$
Output: ($R$, $t_R$) where $R$ is the request CR and $t_R$ is the issuance time
$A$ = last issued CR at time $t_A$
$B$ = last visited CR safe to disclose
$C$ = CR enclosing $u$ at $t_q$
$t_C = t_A + \frac{dist(A,C)}{v}$ /*$dist$ is either $d_{haus}$ or $d_{pp}$ */
$d_B$ = min. distance from $u$ to $B$, $d_C$ = min. distance from $\ell(t_C)$ to $C$
if($t_C - t_q > MaxDelay$)
    return ($B$, $t_q$) /* postdate $B$ */
else if($d_B < d_C$)
    return ($B$, $t_q$) /* postdate $B$ */
else
    return ($C$, $t_C$) /* defer $C$ */
```

Fig. 10. Heuristic for Temporal Cloaking

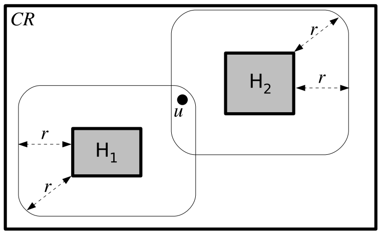


Fig. 11. Reverse-engineering attack when CR construction takes as seed the user location

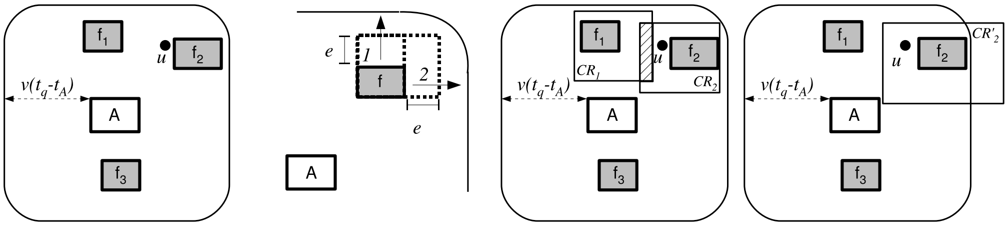
on-line approach is that the CR can be tailored for the user's privacy profile, and consequently the QoS can be improved. In this section, we focus on the more difficult setting of an attacker with background knowledge, i.e., the CRs must be constructed taking into consideration the sets of sensitive features and associated sensitivity thresholds introduced in Section 3.4. Attacks without background knowledge can be addressed as a special case where all sensitivity thresholds are set to ∞.

Assume that at some point along its trajectory, user u is situated inside a hospital H. Denote by $thr_H = 0.5$ the sensitivity threshold of u for feature type *hospital*. In this case, it is necessary to reduce the probability of association of u with H by creating a CR at least twice as large as the area of H. On the other hand, if the user is in a non-sensitive area, then the exact location could potentially be disclosed, since this is not a privacy violation[4].

Note that, computing and reporting a CR only when the user is inside the hospital is not an acceptable solution, since an adversary that has knowledge about the algorithm used for cloaking could immediately infer from the fact that a CR is generated that u must be inside the hospital. Therefore, the cloaking algorithm must take into account the sensitive features in the user's proximity even if the user is not currently situated within the perimeter of sensitive locations.

One naive solution could work as follows: given a distance r (chosen as a system parameter), initiate the CR construction whenever some sensitive feature is situated at distance less than r from u. This requirement is equivalent to an inclusion condition stating that every sensitive feature within distance r from u must be enclosed in the CR. This way, the fact that a CR including a hospital is generated does not imply that the user is necessarily inside a hospital. However, such an approach is vulnerable to reverse-engineering by an adversary, as shown next: Consider the example in Figure 11 with two sensitive features H_1 and H_2. An attacker that learns the disclosed CR including both

[4] Alternatively, if users do not wish to disclose exact locations, a random region with size above some minimum threshold can be trivially generated.



(a) Sensitive Feature Selection (b) Feature Enlargement (c) Privacy threat with known feature order (d) CR Deferral

Fig. 12. Spatial Cloaking

features may infer that the user is at distance at most r from both H_1 and H_2. By intersecting the Minkowski sum enlargements of the two features by radius r the adversary can narrow down the dataspace region where u is situated. In general, any CR construction algorithm that takes as seed the current user location may be susceptible to reverse-engineering attacks.

```
Spatial Cloaking
Input: request timestamp $t_q$, last disclosed CR $A$
Output: next CR $B$, issuance time $t_B$
$MS(A)$ = MinkowskiSum of $A$ with enlargement $v(t_q - t_A)$
$SF = \{f \in \mathcal{F} | f \cap MS(A) \neq \emptyset\}$
while $\neg empty(SF)$ do
    $f$ = random element in $SF$, $SF = SF \diagdown \{f\}$
    if $(u \in f)$ then Sensitive=True
    $R = Enlarge(f)$ //repeatedly considers all four directions
    if $R \neq$ null and $u \in R$ then
        $t_B = \max\{t_q, t_A + \frac{d_{pp}(A,B)}{v}\}$
        return $(R, t_B)$
if Sensitive=True then Drop Request
else return $(u, t_q)$
```

Fig. 13. Spatial Cloaking Pseudocode

To prevent reverse-engineering attacks, we propose a method that constructs CRs which are not directly dependent on the user location, but instead are built starting from the last reported CR, which is already known to the adversary (i.e., the adversary does not learn additional information even if s/he is capable of recovering the input used for spatial cloaking). This strategy is illustrated in Figure 12, which shows the user location and the last disclosed CR A. The numbered rectangles represent sensitive features. Denote by $MS(A)$ the Minkowski sum of A enlarged by $v(t_q - t_A)$: $MS(A)$ encloses all locations where user u could be situated at request time t_q. The CR construction consist of three steps:

Step1. **Filtering of features.** All sensitive features that intersect $MS(A)$ represent the set SF of candidates for inclusion in the CR. For instance, in Figure 12(a), the set of selected features is $SF = \{f_1, f_2, f_3\}$. Note that, if the set SF is empty, then no privacy threat exists with respect to the current location of u, and therefore the location of u can be directly disclosed.

Step 2. **Cloaking.** The cloaking step chooses a sensitive feature $f \in SF$ and progressively enlarges it to find a CR (denoted by CR_f) that satisfies the privacy requirement (i.e., the sensitive area within the CR represents a fraction of the total CR area no larger than the user-specified threshold). Initially, CR_f is set to be equal to f. As long as CR_f does not satisfy the privacy constraint, it is enlarged by a fixed amount e (Figure 12(b)) along one of its sides (in the order $\{top, right, bottom, left\}$). The process stops if CR_f satisfies the privacy constraint. If the resulting region encloses the location of u, Step 3 is executed for the obtained CR. Alternatively, if all features in SF are considered but no CR that satisfies both properties is found, there are two cases: (1) if the location of u is not enclosed by a sensitive feature, then the exact point location of u can be disclosed. Otherwise, (2) the user is inside a sensitive feature, and in this case, since no CR can be found, the request is dropped. The latter case may occur, for instance, in a scenario where the sum of areas of all sensitive features may represent a larger fraction of the dataspace than specified by the privacy threshold. However, as we show in the experimental evaluation, the request drop rate is low in practice, even for demanding privacy requirements.

Note that, if the order in which the sensitive features are processed is known, the attacker may be able to pinpoint the user location. Consider features f_1 and f_2 in Figure 12(c), with resulting cloaked regions CR_1 and CR_2, respectively. Assume that f_1 is processed before f_2. If CR_2 is issued, the attacker can infer that the user is not located inside CR_1 (otherwise CR_1 would have been issued), and can therefore learn that the user cannot be situated in the hatched region, possibly leading to a violation of the privacy threshold within CR_2. To prevent such inference, the features in SF are processed in random order.

Step 3. **Safety enforcement.**

In Figure 12(c), the obtained CR_2 is enclosed by $MS(A)$. However, this does not guarantee that $d_{pp}(A, CR_2) \leq v(t_q - t_A)$, therefore an additional delay may be necessary, similar to temporal cloaking. This situation is even more clearly illustrated in Figure 12(d), where the obtained CR'_2 extends beyond the boundaries of $MS(A)$. Given that the attacker already knows A, CR'_2 is not safe to disclose. To ensure safety, similar to the case of temporal cloaking, the CR is deferred for a time equal to $d_{pp}(A, CR'_2)/v - (t_q - t_A)$. Note that, since no intermediate CR is computed between A and CR'_2, post-dating is not possible in this case.

Figure 13 summarizes the spatial cloaking process. First, the algorithm determines the set SF of sensitive features enclosed by $MS(A)$ (line 1). Next, the features in SF are considered in random order, and enlarged until a CR is obtained such that it satisfies the privacy constraints and it encloses the user (line 7). If the obtained CR is not safe, it is deferred until t_B, computed in line 8. If no valid CR is found and u is inside a sensitive feature (line 10), the request is dropped.

6. PROTECTION MODELS AND ALGORITHMS FOR MOBILE INTERACTING USERS

So far, we focused on the case of protecting the location of an individual mobile user across multiple snapshots of position reporting. However, in today's computing landscape, mobile users constantly interact with each other through social media applications, and often in ways that directly involve geospatial information. For instance, location-based social networks (LBSNs) allow users to post geo-tagged content related to an event (e.g., photos, videos), in which multiple users are present. Tagging of users and their locations represents a potential privacy threat. In some cases, one can immediately detect and protect against direct breaches of privacy, e.g., when a photo is taken in a hospital, night club, etc., by simply blocking the tagging process. Most LBSNs require explicit permission from all entities that are geo-tagged before allowing the event to be posted. However, in practice, there are many cases where the privacy breach occurs in a more subtle fashion, for which requesting for direct user consent is not sufficient, due to geospatial inferences that an adversary may perform. Specifically, even though a particular geo-tagged post does not in itself pose a privacy threat (e.g., a group photo taken in a public park at noon), an adversary may use that information in conjunction with past and future location snapshot updates from users to infer private details regarding a user's whereabouts in the hours before and after the photo was taken. We provide a detailed description of this threat model and solutions to prevent it in Section 6.1.

In addition, there are other scenarios of interest outside the realm of location-based social media that require location protection with respect to the trajectories of multiple users. For instance, certain users may wish to keep secret the fact that they came in close spatial proximity to other individuals. As a motivating example, consider two business executives that have a secret meeting to plan a possible merger of their companies. Should an adversary find that such a meeting took place, this could have a negative impact on the outcome of the merger, and could cause the stocks of the

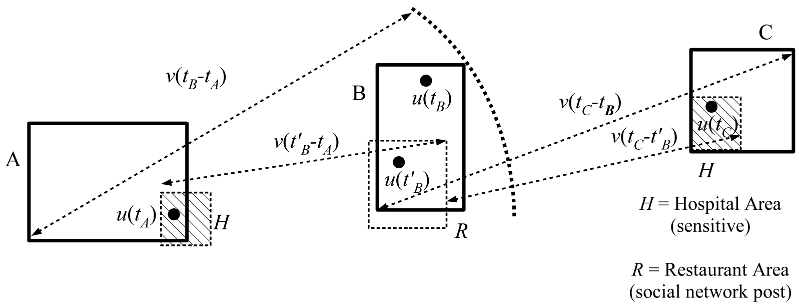


Fig. 14. Privacy Breach in the Presence of External Events

two companies to plummet. In another scenario, a journalist may wish to have a secret encounter with one of her sources to collect information about a sensitive political issue that may endanger the reputation, or even the life of the source, should an adversary learn about their encounter. We address this case in Section 6.2.

6.1. Protection in the Presence of External Disclosure Events

In this section, we investigate the privacy threat that arises when an external location disclosure event occurs outside the process of trajectory anonymization executed by each mobile user. Such situations often arise in practice, for instance in a social network application when a user is present in a photo with geospatial tags. We first overview the attack mechanism, and then propose a solution to protect against this threat. Also, we investigate the trade-off that occurs between protection strength and user satisfaction.

Consider the example in Figure 14 where user u updates her location with her service provider (e.g., LBSN) as she moves. To protect her location, the user's mobile device employs one of the mechanisms presented in Section 5. In the diagram, A, B and C (shown as rectangles with continuous lines) represent the reported CRs at timestamps t_A, t_B and t_C, respectively. CRs A, B and C are safe with respect to maximum velocity v.

Later on, after the user has released all CRs, and hence cannot adjust them further without revealing more details about her location, one of her social network friends wishes to post a photo that user u and her friends took inside restaurant R (shown with dotted line) at time $t'_B < t_B$. Note that, there is nothing sensitive about the fact that u was inside R, but due to velocity constraints, and having access to already released CRs A and B, an adversary can infer that u must have been in a smaller area than the reported CR A at time t_A. That smaller area may be associated with a hospital H, and hence the privacy of u is breached. A similar disclosure can occur for CR C, as the new maximum travel distance $v \times (t_C - t'_B)$ may pinpoint u's location inside another sensitive hospital area.

To address this privacy threat, one straightforward solution would be for user u to reject being tagged in the LBSN event, and thus disallow her identity to be included in the photo. However, this decreases user satisfaction, as u may want to be present as part of the event description of the encounter with her friends. Rejecting too many geo-tagged updates may determine her friends to remove u from their friend lists. Furthermore, even if u rejects the tag, her friends may still go ahead and post the photo without u's name tag. Doing so may still allow an adversary to either visually identify her from the photo, or run some automated face recognition program to assert that u is actually in the photo (even though she is not present in the event metadata). Hence, location disclosure may occur even if u blocks the update.

We propose an approach whereby users take into account the fact that future external events may create privacy breaches, and execute a modified version of the cloaking algorithms, in which each published CR is overprovisioned to protect against possible future additional disclosure. This way, even if a future geo-tagged event occurs, it is possible that the additional disclosure does not constitute a privacy breach. Of course, overprovisioning will cause the quality of service received by the user to decrease, by incurring higher space error and request failure ratios. Hence, a trade-off emerges between the benefit of overprovisioning in allowing integration of future external updates on the one hand, and its negative impact on quality of service on the other hand.

One effective approach to control the amount of overprovisioning is by adjusting the maximum velocity parameter v used when checking for CR safety. Decreasing the value of v to a smaller value $v_O = \frac{v}{\alpha}$ (denoting velocity with overprovisioning) has the effect of building more conservative CRs, which are closer to the previously published CRs than the case when a higher maximum velocity is used. Note that, the user velocity itself is not bounded. Users are still free to move with the speed that they would otherwise, it is just the speed bound for building CRs that changes. As an effect, the chances of a future external event disclosure being within the bounds of a safe CR with respect to the lower v increase. Another advantage is that decreasing the velocity bound has an isomorphic effect on CR placement in all directions, and thus increases the probability of allowing a future geo-tagged update for which the actual position is not known in advance. Finally, the amount of overprovisioning can be captured with a single parameter, namely α, which facilitates system tuning.

Each mobile user will choose CRs following an overprovisioning strategy with parameter α, where α is dynamically tuned to achieve a favorable trade-off between the percentage of blocked geo-tag events (which should be as low as possible) and the space error and request failure ratio, which should be minimized to preserve quality of service. In cases where no privacy breach results due to external events, the parameter α should decrease towards 1 (its minimum value), leading to the case where no overprovisioning is performed. Conversely, if a high rate of blocked events is recorded, α should be rapidly increased.

To obtain a good trade-off between blocking rate on the one hand, and space error and request failure ratio on the other, we employ a *linear decrease - exponential increase* approach to dynamically tune the value of parameter α. Specifically, each user starts its anonymization algorithm with an initial α_0 setting, which is a system-wide parameter. If there are no blocked events, then the value of α decreases linearly with time as

$$\alpha = \max\{\alpha_0 - a \times t, 1\}$$

On the other hand, if the rate of blocked updates reaches a threshold β, then the algorithm doubles the value of α at each time granule as long as the blocked rate in that time period exceeds β. This way, the system adjusts quickly to counter high event-blocking rates. The pseudocode in Figure 15 summarizes the proposed approach.

```
Overprovisioned Cloaking (executed at each time granule $\delta t$)
Input: initial overprovisioning parameter $\alpha_0$, minimum threshold $\alpha_{min}$, maximum blocking rate $\beta$
Output: next cloaking region $CR$, updated $\alpha$
$v_O = \frac{v}{\alpha}$
while (true) do
    blocking rate $br = \frac{approved\_tag\_requests}{received\_tag\_requests}$
    if ($br < \beta$) then
        $\alpha = \max\{\alpha - a \times \delta t, 1\}$
    else
        $\alpha = 2 \times \alpha$
    $v_O = \frac{v}{\alpha}$
    compute new CR with respect to $\alpha$ and previous CR
    $t = t + \delta t$
```

Fig. 15. Pseudocode for Adaptive External Event-Aware Location Cloaking

If an event is blocked, no further action is necessary. However, if an event is accepted, then it becomes yet another update in the movement history of the user, and it needs to be treated accordingly. The pseudocode in Figure 16 describes the algorithm for inserting an external event update in the user's past position history. As each event is basically a geo-tagged object, we refer to an event as a social tag. The algorithm takes as input a sequence S of timestamped CRs that have already been disclosed for a user, and an incoming social tag, having timestamp t_{tag} and spatial extent R_{tag}. Since the tag will be part of the history of the user after acceptance, we have to check if any past position became unsafe due to tag acceptance. In lines 1–2 the candidate position for insertion is found. In the case when t_{tag} is equal to some past CR timestamp, it is excluded by input pre-conditions, since in that case it is accepted only if it is a duplicate of a previously known CR (same timestamp and same region). Once the candidate position is found, the tag T is inserted in sequence seq (line 7) only if it is safe with respect to both previous (line 3) and following (line 5) disclosed positions.

```
Insert a social geo-tag
Input:   sequence of disclosed CRs: $seq = \{(t_1, CR_1), \ldots, (t_n, CR_n)\}$, $i < j \Rightarrow t_i < t_j$
         social geo-tag: $T = (t_{tag}, CR_{tag})$, $\forall i \; t_{tag} <> t_i$
Output:  original sequence (tag rejected)
         extended sequence (tag accepted)

// find the candidate position for T in $seq$
1. $prev =$ last $(t_i, CR_i)$ such that $t_{tag} > t_i$
2. $next =$ first $(t_i, CR_i)$ such that $t_{tag} < t_i$
3. if $prev <> \perp$ and not $isSafe(prev, (t_{tag}, CR_{tag}))$ then
4.     return $seq$          // reject: T it unsafe w.r.t. previous position
5. if  $next <> \perp$ and not $isSafe((t_{tag}, CR_{tag}), next)$ then
6.     return $seq$          // reject: T it unsafe w.r.t. following position
7. insert $(t_{tag}, CR_{tag})$ in $seq$ before $next$
8. return $seq$              // accept: T it safe
```

Fig. 16. Insert Tag Pseudocode

6.2. Protection of Relative User Proximity

In this section, we focus on scenarios where it is important to protect against inferring that two (or more) mobile users were co-located at a certain moment in time. We restrict our discussion at the two users case, as the general case with more than two users can be easily generalized.

The two users wish to provide to their service providers anonymized trajectories that hide their mutual proximity relationship. In addition to the methods proposed in Section 5, where each user generates CRs in such a way that association with any sensitive areas due to velocity bounds is prevented, Alice and Bob also need to ensure that there is always a certain threshold distance between their possible location. We called this threshold *separation distance* and we denote it by S.

Figure 17 illustrates an example of this scenario, and an overview of the process of enforcing separation. Users Alice (U_1) and Bob (U_2) move towards each other. At time t_A, both posted their location updates, namely A_1 and A_2, respectively. The two CRs were further apart from each other than separation distance S. At the next time stamp t_B, according to the independent anonymization algorithm executed by the users, their tentative CRs would be B_1 and B_2, shown with dotted line. While these two CRs abide the safety requirements stemming from the velocity constraint, they do not satisfy the separation threshold. In fact, they overlap, which may lead an adversary to infer with high probability that Alice and Bob might have met.

The objective of the relative user proximity protection algorithm is to generate individual CRs that are both within the safety bounds dictated by maximum velocity, but at the same time they

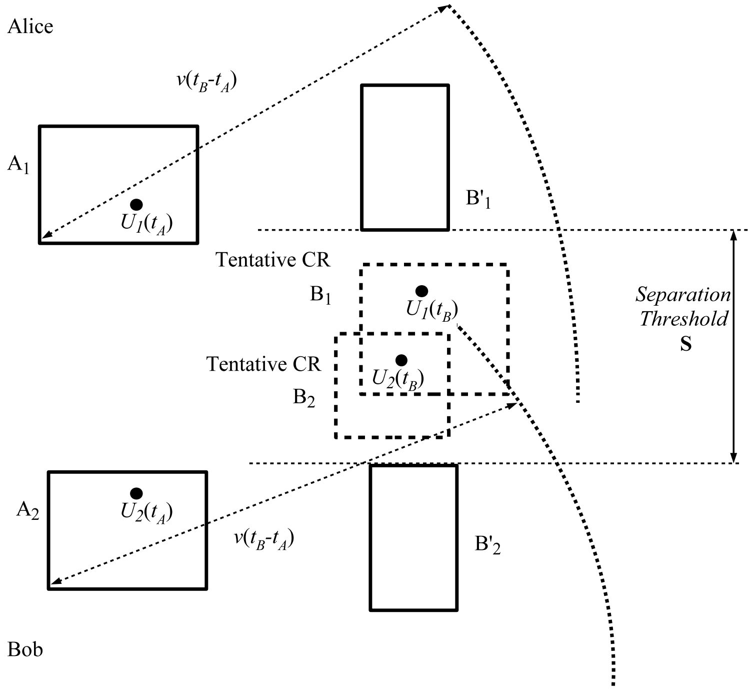


Fig. 17. Overview of Enforcing Threshold Distance Separation

are further apart from each other than separation threshold S. In the example of Figure 17, CRs B_1 and B_2 are suppressed, and instead the CRs B_1' and B_2' are released. Note that, it is possible (and in fact likely) that the two users are no longer enclosed within their reported CRs. As in the case of the techniques presented in Section 5, this situation creates a decrease in quality of service, as any location-based query asked with respect to B_1' or B_2' will return sub-optimal results for the respective users.

The objective of the protection mechanism is to enforce the separation threshold while at the same time minimizing the decrease in quality of service. A good measure of the latter is the distance between the actual user location and the closest point in its reported CR. If the CR encloses the user, then there is no penalty. As there are two users, the metric to minimize is the sum of the distances.

In addition, it may be important to maintain more information in the direction of movement of the users. In other words, users are less likely to be interested in locations that are behind them, than forward ones. Following that intuition, we construct the perpendicular bisector of the segment connecting the two users, and symmetrically place the two CRs at equal distance from the bisector. However, none of the users wishes to disclose his or her exact location, so only safe information should be used in the process.

Figure 18 illustrates the proposed solution, and the pseudocode in Figure 19 details the algorithm steps. The algorithm receives as input a separation threshold S and a pair of CRs B_1 and B_2, corresponding to the tentative CRs of users u_1 and u_2, respectively. In case the minimal distance of B_1 and B_2 satisfies threshold S, there is no need to change them (line 3). Otherwise we determine the direction of displacement $\overrightarrow{v}$ along the line passing through the center of mass of the two CRs for the initial user coordinates (line 5), or orthogonal to the bisector of the direction of movement of the two users for subsequent timestamps (line 10). In order to satisfy the separation constraint we translate the two CRs by a total of S plus the maximum diameter of the two CRs (line 12). Since

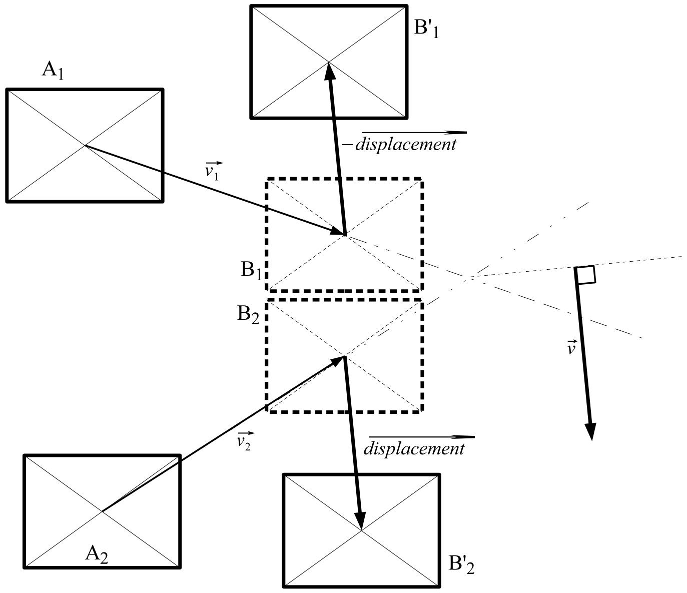


Fig. 18. Technical Details of Threshold Separation of CRs

the two users cooperate, each CR is translated according to one half of the displacement vector in the two opposite directions (lines 13-14).

We assume that the users who want to hide their proximity are able to communicate directly, and that position reporting is synchronous for all users. These restrictions are used to simplify the presentation, but can be removed in practice with minimal changes, thus not hindering the generality of the approach. Specifically, if updates are not synchronized, one simple solution is for the system to apply the deferral technique presented in Section 5.1 for the user with the earlier update. Then, when the update from the other user is received, the algorithm for synchronous updates is performed. The tradeoff of this method is accuracy, as the first user will experience a slightly higher space error.

7. EXPERIMENTS

We implemented a Java prototype of the proposed temporal and spatial transformation methods. The experimental testbed consists of a P4 2.0GHz machine with 1GB of RAM running Linux OS 2.6. We ran tests both on synthetic and on real datasets. The synthetic datasets allow us to vary continuously parameters such as maximum velocity, thus providing a clear illustration of the behavior trends of the proposed solutions for a broad range of parameters. On the other hand, evaluation on real data proves the practical applicability of our approach.

In the case of synthetic data, we consider a dataspace of $10,000 \times 10,000$ meters, corresponding to a medium-size city. In each experimental run, we randomly generate 100 trajectories consisting of 30 distinct timestamps each with an average delay of 30sec between consecutive requests. We consider maximum user velocities $MaxSpeed$ between $1m/s$ (walking speed) and $30m/s$ (highway driving speed). The actual user velocity is uniformly distributed in the range $MaxSpeed/2$ to $MaxSpeed$. All results are averaged over ten random seeds.

```
Proximity Protection
Input: CRs $B_1$ and $B_2$ of position reports from users $u_1$ and $u_2$, separation threshold $S$.
Output: $B'_1$ and $B'_2$ such that $dist(B'_1, B'_2) \geq S$

$d = dist(B_1, B_2)$
if ($d \geq S$)
    return $B_1$ and $B_2$
$A_1, A_2$ = most recently disclosed positions for $u_1$ and $u_2$
if ($A_1, A_2$ are first disclosed positions)
    $\overrightarrow{v}$ = unitary vector from the center of $B_1$ toward the center of $B_2$
else
    $\overrightarrow{v_1}$ = unitary vector from the center of $A_1$ toward the center of $B_1$
    $\overrightarrow{v_2}$ = unitary vector from the center of $A_2$ toward the center of $B_2$
    $\overrightarrow{v}_\perp$ = unitary angle bisector vector of $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$
    $\overrightarrow{v}$ = rotate $\overrightarrow{v}_\perp$ 90° toward the center of $B_2$
$\overrightarrow{displacement} = \overrightarrow{v}\,(S + max(diam(B_1), diam(B_2)))$
$B'_1$ = translate $B_1$ according to $-\frac{1}{2}\,\overrightarrow{displacement}$
$B'_2$ = translate $B_2$ according to $\frac{1}{2}\,\overrightarrow{displacement}$
return $B'_1$ and $B'_2$
```

Fig. 19. CR displacement for proximity protection

The real workload considered consists of the Rome taxi dataset from the CRAWDAD repository[5]. The data were collected from real GPS traces of Rome taxi drivers during the month of February 2014. Within the dataset, we group trajectories according to maximum recorded speed, so that we can obtain two different maximum speed settings, namely $Slow$ with speeds ranging within $[0, 5]$ m/s, and $Fast$ with speeds in the interval $[10, 18]$ m/s.

We present the results for temporal and spatial cloaking in Sections 7.1 and Section 7.2, respectively. Section 7.3 shows the results for protection in the case of external events, whereas Section 7.4 illustrates the case of hiding mutual proximity.

7.1. Temporal Cloaking

For the temporal cloaking experiments, we consider a fixed partitioning of the dataspace into rectangular regions (or tiles) of variable size. Tiles are randomly generated, and tile granularity is varied between 100m and 500m side length. The form factor (or skewness) of a tile T, measured as

$$\frac{\max(height(T), width(T))}{\min(height(T), width(T))},$$

is varied randomly between 1 and 2. Since the CRs are pre-determined by the space partitioning, we do not consider CR size for temporal cloaking: instead, we focus on space and time error.

First, we measure the fraction of dropped requests in the absence of postdating. This *"deferral-only"* method is similar to the solution proposed in [Cheng et al. 2006]. We consider two maximum delay settings: $MaxDelay = 5$sec, corresponding to an acceptable response time when asking queries, and 60sec, reasonable for location updates in a social networking application. Figure 20 shows that, if postdating is not allowed, the failure ratio grows as high as 60% of requests. The failure ratio is the highest in the scenario of location-based queries (i.e., low $MaxDelay$) and an adversary with background knowledge (i.e., d_{pp} distance). The high failure ratio motivates our choice for request postdating. In the rest of this section, we evaluate the proposed temporal cloaking method, including both request deferral and postdating. We emphasize that in all considered temporal cloaking scenarios, no request was dropped.

[5] Available online at http://crawdad.org/roma/taxi/20140717/

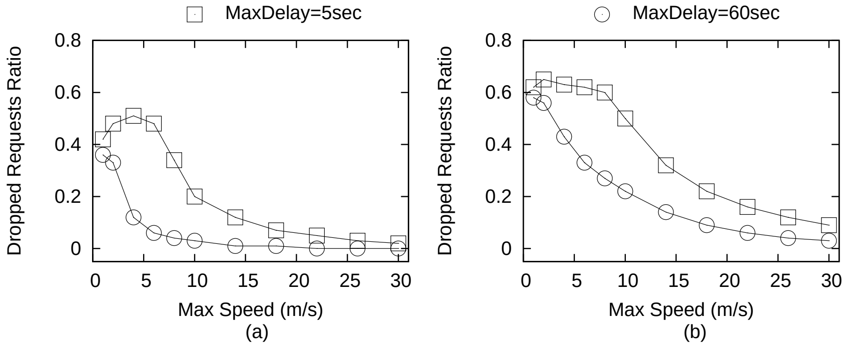


Fig. 20. Ratio of dropped requests in absence of postdating, (a) Hausdorff and (b) maxPP distance

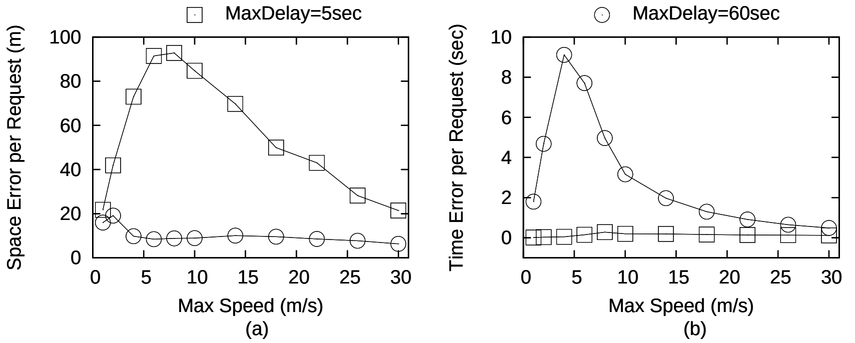


Fig. 21. Variable velocity, Hausdorff distance

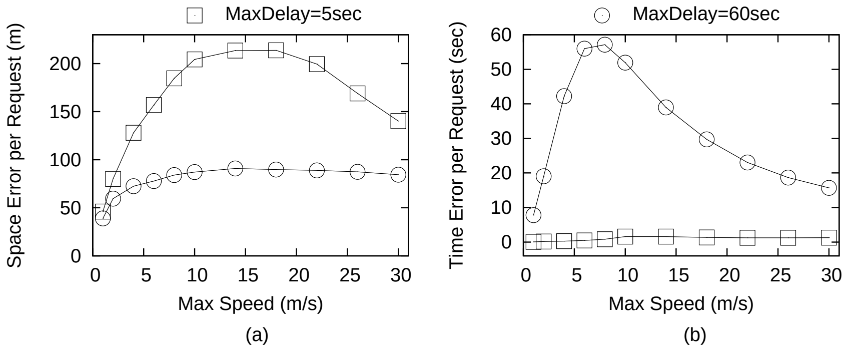


Fig. 22. Variable velocity, point-pairwise distance

Next, we measure the space and time error incurred by temporal cloaking when the maximum user velocity is varied, for a fixed partitioning granularity with average tile side 300m. Figure 21 shows the results for the Hausdorff distance, whereas Figure 22 considers point-pairwise distance. In both cases, the time and space errors exhibit a bell-shaped dependence, as a result of the relationship between velocity and tile side length. At low velocity, it is likely that several consecutive requests will fall within the same tile, hence the requests are safe to be issued. As velocity increases initially, consecutive requests fall within neighboring cells, and the queries need to be deferred/postdated. However, as velocity continues to increase, the distance traveled by the user between two consecutive requests A and B may span several adjacent tiles, but the difference between $d(A, B)$ (either

Hausdorff or point-pairwise) and $v(t_B - t_A)$ is small compared to the user velocity (i.e., at most the diameter of one tile). Therefore, the CR safety condition can be satisfied with only a short delay. As expected, the more demanding restrictions of point-pairwise distance determine larger absolute values of space and time error compared to Hausdorff distance.

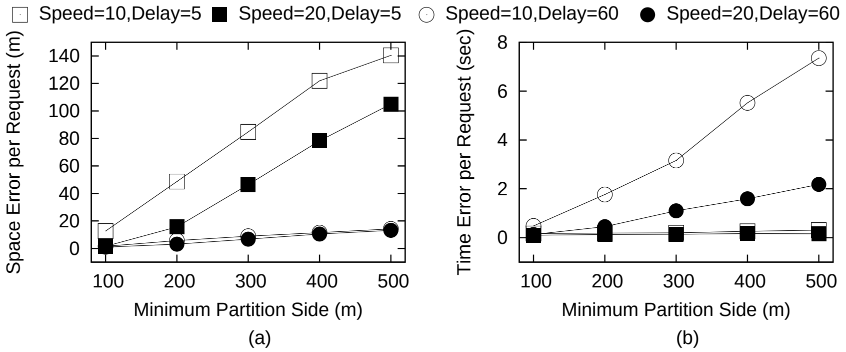


Fig. 23. Variable tile size, Hausdorff distance

In Figure 23, we measure the effect of varying the partitioning granularity for two distinct choices of $MaxSpeed$ and $MaxDelay$. For brevity, we only show the results for the Hausdorff distance. The diameter (i.e., diagonal) of the average tile grows with tile side, and consequently, the space and time errors also increase. This experiment also illustrates the clear trade-off between space and time error: for low acceptable delays (e.g., $MaxDelay = 5$sec, suitable for location-based queries) the space error grows larger (although, in absolute value, it never exceeds 1.5% of the dataspace side). On the other hand, if longer delays are acceptable, the space error is reduced below 20m.

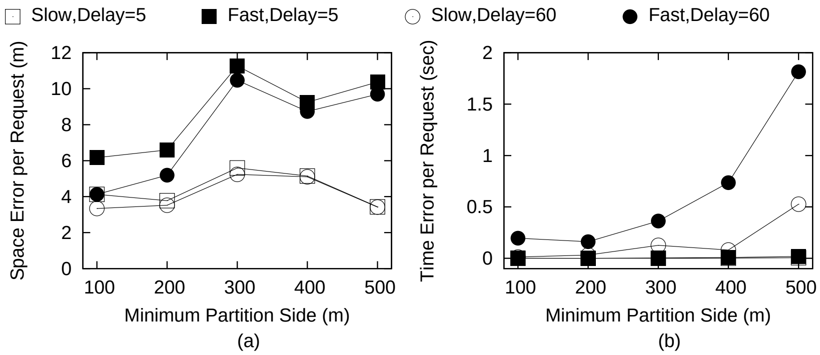
Figure 24 illustrates the same measurement on the real dataset[6]. We consider both $Slow$ and $Fast$ speed settings. Similar trends are observed as in the case of the synthetic data, except that there is more variability in the results, due to the coarser granularity in varying velocity values. We emphasize that, in absolute value, the results are much better than in the synthetic data case: specifically, space error was at most 12 meters, and the request delay at most 2 seconds.

7.2. Spatial Cloaking

For the spatial cloaking case, no pre-defined space partitioning is enforced, and users have the ability to construct CRs according to their privacy requirements. The construction of CRs when the attacker has no background knowledge is trivial (e.g., generate random CRs with a certain constraint on minimum size), so we only consider the case of an adversary with background knowledge, hence the point-pairwise distance is employed. Since spatial cloaking is not restricted to fixed CRs, we focus our evaluation on CR size, time error and failure ratio, and we omit space error, which only represents a significant factor for temporal cloaking. $MaxDelay$ is set to 10sec. In all spatial cloaking tests, we observed that the time required to construct a single CR is below 1sec. We also take into account the sensitive feature *coverage*, defined as the percentage of the dataspace area covered by sensitive features. Intuitively, the larger the sensitive coverage, the more difficult it is to find low-extent CRs that satisfy the privacy requirement.

Figure 25 shows the resulting CR area (expressed as a percentage of the dataspace area) and the time error when the user's sensitivity threshold is varied, for a fixed coverage of 5%. The CR area is always below 1% of the dataspace area, and it decreases as the sensitivity threshold increases (recall that, a higher sensitivity threshold corresponds to a less demanding privacy requirement). The time

[6] Since all experiments prior to this one make use of a granular increase in maximum velocity, which cannot be obtained using real traces, we restrict the evaluation on the real data to the variable tile size experiment.



(a)

(b)

Fig. 24. Variable tile size, Hausdorff distance, Real Dataset

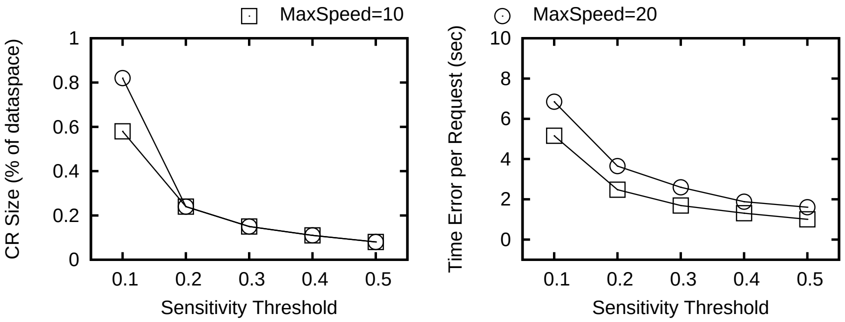


Fig. 25. Variable Sensitivity

error required by spatial cloaking exhibits a similar trend. A larger velocity results in an increase of CR size and time error because more sensitive features across a larger area are considered in the cloaking procedure (i.e., the set SF enclosed by the Minkowski sum of A, according to the notation in Section 5.2).

In Figure 26 we investigate the effect of sensitive feature coverage for a sensitivity threshold of 0.1. The CR area grows with coverage, since a larger fraction of the dataspace is sensitive, hence the CR must grow larger in order to include large enough non-sensitive areas. Note that, even for the most demanding privacy requirements considered (i.e., 10% coverage and 0.1 sensitivity) the CR area does not exceed 2% of the dataspace. The time error trend also shows an increase with coverage.

Finally, Table 27 shows that, in most cases, all requests are successfully cloaked. For the 0.1 sensitivity threshold, there are some requests that fail. However, the ratio of such requests is low, (5% in the worst case).

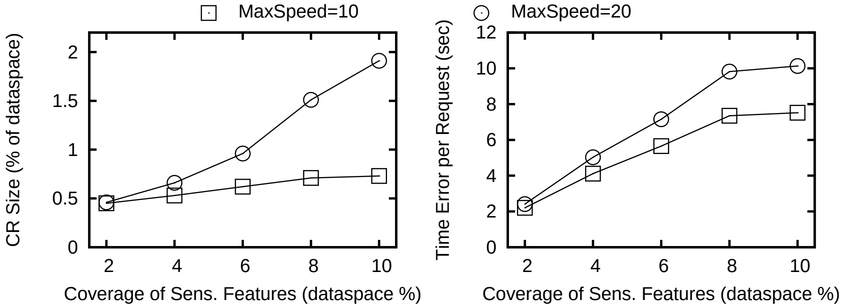


Fig. 26. Variable Sensitive Feature Coverage

Sens. Feature Coverage	Thr=0.1	Thr=0.2	Thr=0.3
2%	1%	0%	0%
4%	1%	0%	0%
6%	2%	0%	0%
8%	3%	0%	0%
10%	5%	0%	0%

Fig. 27. Failure Ratio

7.3. Protection in the Presence of External Events

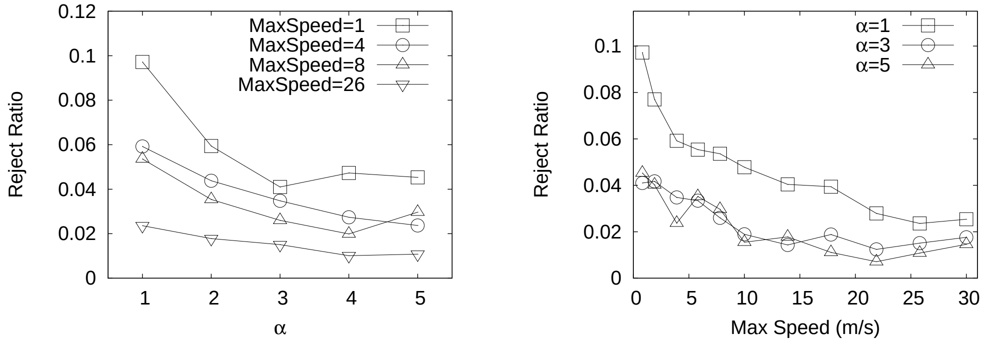
In this section, we evaluate the performance of the proposed technique for location protection in the presence of external events. In absence of such events, each user trajectory is represented by an ordered set of timestamped CRs. To model external events, such as social media posts, we consider that an additional tagged object with its own timestamp and geographical coordinates is inserted into the sequence of user updates. We refer to these additional entries in a user's trajectory as *tags*.

For each trajectory, we insert a number of additional tags selected uniformly at random between 1 and the number of snapshots in the user's trajectory. For each such tag, we randomly select the time within the user trajectory bounds, and determine the position by using linear interpolation between two consecutive user-reported positions. This is a realistic model, since the posts about a user will actually be situated along the user's trajectory. We average our results over $1,000$ randomly generated user trajectories.

Recall from Section 6.1 that the amount of overprovisioning that a user performs in order to increase the likelihood that a future event is approved for publication is modeled by parameter α, which measures the factor by which the maximum velocity is reduced when computing CRs. We vary α in the interval 1 to 5, and we determine the impact on *rejection rate* of tag insertions. Rejection rate represents the number of tagged events that the user blocks, and has ideal value 0 and maximum value 1. We also vary the actual maximum velocity of movement as $MaxSpeed$, as it also influences the rejection rate.

Figures 28 and 29 present the results for two distinct values of coverage, 1% and 10%, and several settings of actual maximum speed $MaxSpeed$. We observe that in the absence of overprovisioning, a relatively high proportion of tags are rejected, which may exceed 10% for low movement speed (e.g., walking users). In Figures 28(a) and 29(a), we can observe that as α increases, the rejection rate drops sharply, proving the effectiveness of the proposed technique. Furthermore, one does not need to increase considerably the value of α, as a value of 2 or 3 suffices to obtain low rejection rate. For further increases of α, the additional gain is not significant. In fact, in some cases, the rejection rate increases slightly, mostly due to random factors of movement and tag generation.

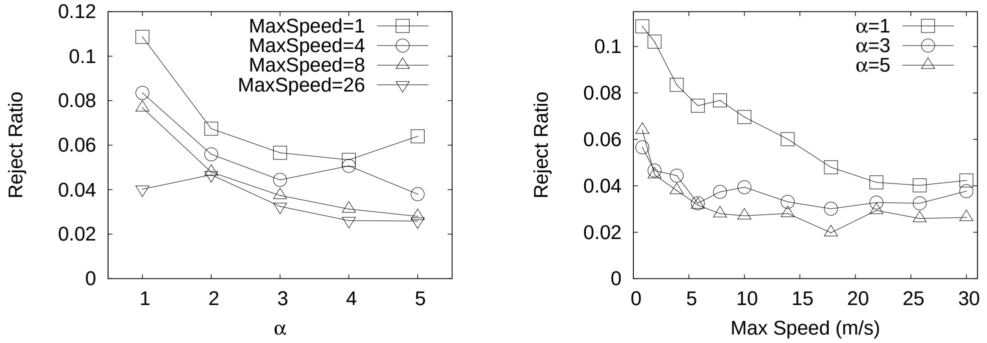
Figures 28(b) and 29(b) illustrate the behavior of the proposed technique when varying the maximum user velocity $MaxSpeed$. We consider a broad interval of velocities, ranging from $1m/s$ (walking speed), to $10m/s$ (cycling speed) and $30m/s$ (driving speed). Recall from Section 6.1



(a) Varying α

(b) Varying Maximum Velocity

Fig. 28. Event Reject Ratio, Coverage = 1%



(a) Varying α

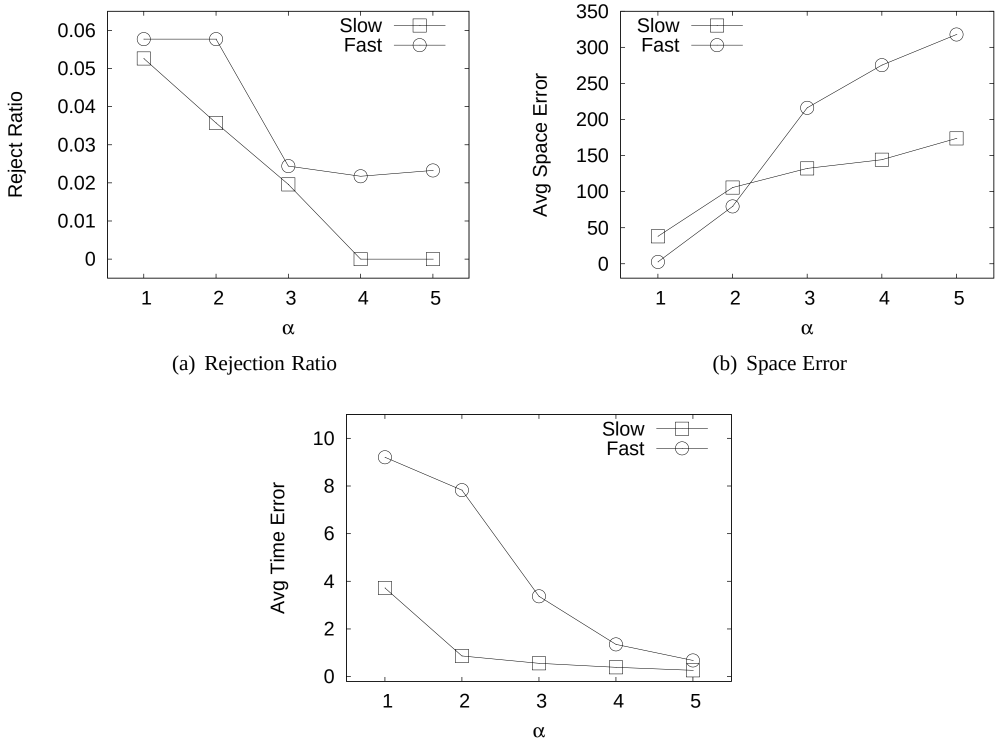
(b) Varying Maximum Velocity

Fig. 29. Event Reject Ratio, Coverage = 10%

that, for a given α value, the absolute reduction in velocity when computing CRs is proportional to $MaxSpeed$, hence we expect an increase in velocity to yield lower rejection rate. In addition, a higher $MaxSpeed$ also has a benefic effect on rejection rate as the actual user movement speed is fast, and therefore the user is able to travel relatively quickly between consecutive location updates. We observe that the walking speed range is most prone to rejected tags, whereas the rejection rate becomes negligible at higher speeds.

We also observe that an increase in coverage rate does affect negatively the rejection rate, but not significantly (to observe this increase, one can compare the corresponding graphs from Figures 28 and 29). This can be explained by the fact that, once the user CRs are selected according to the more strict coverage requirements, they are already large, so the effect of additional tags may not be that difficult to overcome. In other words, due to the already stringent requirements imposed by the user movement, the additional requirement imposed by overprovisioning for tag events is not significant.

Finally, we evaluate the performance of the protection algorithm in the presence of external events on a real dataset. Figures 30 shows the results. Due to the inability to control in a fine-grained manner the maximum velocity for real data, we show results only for variable α, and two different speed thresholds, $Slow$ and $Fast$. The results are very encouraging, as the rejection ratio is well below the one recorded for synthetic data. In the worst case, 6% of the update requests get rejected. The time error is also low, with 10 seconds as maximum value. As expected, the space error grows with α, and for larger values it may become significant, up to 300 meters. Nevertheless, the results show that, given the low rejection error and time error values, one can safely set the α value to be



(a) Rejection Ratio

(b) Space Error

(c) Time Error

Fig. 30. Protection in the presence of external events on real dataset

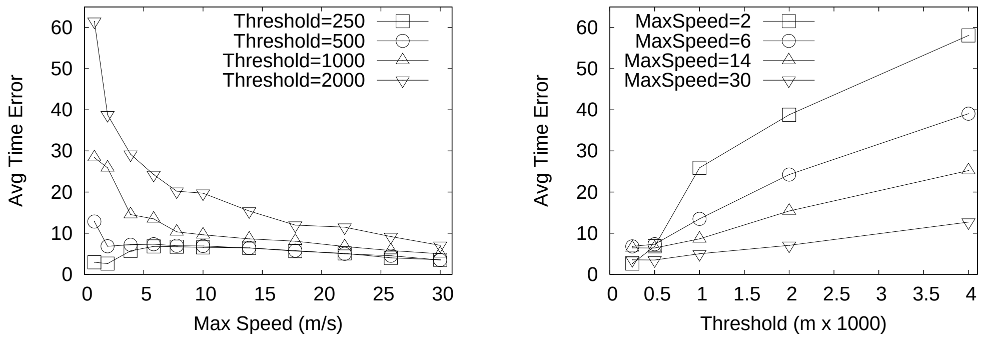
small (e.g., 1 or 2), and as a result the space error at this setting is also small, less than 100 meters. The results on real data confirm the practical applicability of the proposed technique.

7.4. Protecting User Proximity Relationship

We evaluate next the performance of the proposed technique to protect the mutual proximity distance relationship between a pair of users. We consider separation threshold values in the range between 250 meters and $4,000$ meters, which is a relatively broad range of requirements for a total considered movement space of 20×20 kilometers. We generate pairs of trajectories moving towards each other, and which intersect after an average number of 20 timestamps. To hide user proximity, the proposed technique will move CRs in opposite direction to each other, and as a result each user will experience delays in their reported locations, and consequently, in received service. We measure the amount of delay as a measure of loss in quality of service.

Figure 31(a) illustrates the effect of maximum velocity on the user-experienced delays, denoted as $AvgTimeError$. For very low-speed movement, the need to hide mutual proximity comes at a high cost, as it takes users a long time to arrive within the safe-to-disclose CRs (recall from Section 6.2 that in order to protect against mutual proximity detection, it may be necessary to report CRs that do not actually enclose the user). In some cases, the updates may need to be delayed for over 60 seconds. However, as the maximum velocity increases, the proposed technique is able to achieve protection at a much lower penalty in terms of quality of service decreases. For moderate movement speeds, and separation threshold values that to not exceed $1,000$ meters, the experienced delays are lower than 10 seconds.

Figure 31(b) presents the evolution of the time error as the separation threshold increases. As expected, a higher value of the separation threshold results in higher delays, as the CRs have to be moved further apart from each other. Nevertheless, for moderate values of maximum velocity, the obtained delays are always below 30 seconds. For high speeds, the delays are negligible for most of the range of separation threshold values. In practice, we believe that a separation threshold of $1,000$ is likely to be sufficient, and one can observe that for this setting, the delays are not significant.

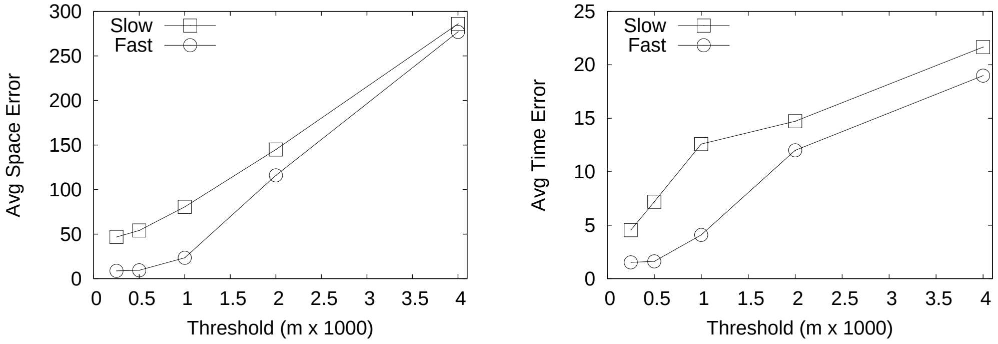


(a) Varying Maximum Velocity

(b) Varying Proximity Threshold

Fig. 31. Effect of Mutual Proximity Protection on Service Delays

In our final experiment, we evaluate the effectiveness of the mutual proximity protection technique on real data. Figure 32 shows the space and time errors obtained for two speed settings, $Slow$ and $Fast$, and several different proximity threshold settings. As expected, the space and time errors increase with the threshold. However, the actual time error values are smaller than in the case of synthetic data, with a worst-case delay of approximately 20 seconds. The space error is around 300 meters in the worst case, which is less than 10% of the proximity threshold setting. These results confirm the good performance of the proposed technique for real data.



(a) Space Error

(b) Time Error

Fig. 32. Effect of Mutual Proximity Protection for Real Datasets

8. USABILITY AND INTEGRATION WITH EXISTING LOCATION-BASED SERVICES

The objective of our work is to provide a balanced location protection approach which achieves good privacy and does not incur significant performance overhead. We have evaluated the above two objectives experimentally in Section 7. In addition, for our approach to gain widespread adoption, it is necessary to consider two additional factors: *(i)* usability aspects, and *(ii)* integration with existing location-based services (LBS). While a comprehensive study on these topics is outside the scope of this article, we provide in this section an informal discussion and outline future work directions towards achieving widespread adoption of the proposed techniques in real-life systems.

8.1. Usability

Making end-users aware of security and privacy threats, and devising controls that do not significantly affect usability are serious and ongoing concerns in the broader security and privacy research, not only in the location privacy area. In order to help widespread adoption of privacy-preserving services, it is important to reduce the burden on end users with respect to privacy-related system settings and parameters. Furthermore, users should be shielded from low-level technical details, and the privacy settings and parameter choices provided must be intuitive.

In the case of our proposed approach, the protection model is directly related to the ability of an adversary to locate the user within a certain geographical enclosure, e.g., a building, a neighborhood, or the location of an event. We believe that this approach tends to be more intuitive than other models which rely on more complex, statistical models of privacy that are difficult to understand by the end users. For instance, other approaches use protection definitions based on entropy, or probability of existence of a user within a dataset, which rely on understanding of advanced mathematical concepts. In contrast, our solution defines privacy with respect to the percentage of the reported area that lies within a certain sensitive region. In our view, a user is more likely to relate to a statement such as *“this privacy setting ensures that sensitive areas represent only 10% of the reported region”*.

The issue of setting system parameters is also an important one. A usable solution should have few parameters that need to be provided as input, and the choices of parameters should be simple and intuitive. Specifically, in our case, $MaxDelay$ is the maximum amount of time that the user is willing to wait for a query answer or service update. Response time is a parameter that a user can easily relate to. A simple graphical user interface (GUI) dialogue can be shown to the user the first time the system is set up, and prompt for a value for this parameter.

Furthermore, the sensitive threshold value is a direct representation of the association probability with sensitive features, which is the maximum percentage of a CR that can be covered by that sensitive feature type (e.g., 10% or 25%). In addition, for users who do not wish to provide numerical thresholds, one can design a Likert-scale GUI with several levels: e.g., *low sensitivity* (70-80%), *moderate* (20-50%), or *high* (below 20%). Similarly, for the maximum delay parameter, one can choose from several discrete options, labeled for instance as *low delay* (below 5 sec), *moderate delay* (5-30 sec), *long delay* (30-60 sec).

8.2. Integration with Existing LBS

Existing systems that process location updates are typically designed to accept as input individual locations, or groups of individual locations. However, in the case of our approach, the LBS must process updates that are submitted in the form of rectangular regions (CRs). This feature may require changes on behalf of the service providers to support processing of regions.

Our work assumes that a processing engine for regions exists at the LBS. In the spatial databases literature [de Berg et al. 2000] several approaches exist for processing regions. We emphasize that the adoption by the service providers of such techniques is not a requirement of location privacy solutions alone. Instead, many novel types of queries such as nearest-neighbor of groups of users or skylines require such operations. Hence, we believe that there will be strong incentive for LBS providers to adopt such processing techniques.

Another related concern is that of processing cost. In the case of processing regions, the computational overhead is typically higher than for points. However, as shown in previous work on location privacy with CRs [Kalnis et al. 2007], the performance obtained is quite good. To account for the additional overhead, the providers may impose an additional small processing fee, either in the form of a subscription, or indirectly through more advertisements, or requiring the users to complete a survey, etc. In addition, we expect that policy makers, who are becoming increasingly aware of the risks that privacy breaches pose to society, may mandate the use of such techniques, in a similar manner in which healthcare providers must implement controls to protect medical records.

9. CONCLUSIONS

In this work, we identified attacks on location privacy that occur when an adversary is able to use information such as maximum user velocity, geo-tagged social network posts, or mutual proximity between users. The proposed spatial and temporal transformations enforce privacy without significant deterioration of QoS. The techniques for overprovisioning to reduce the amount of blocked posts and for hiding mutual proximity are also effective in achieving protection without significant QoS deterioration.

In future work, we plan to investigate protection techniques against proximity-based and external event attacks when user movement is restricted to road networks. In this setting, the distance computation between consecutive user positions is more complex. In addition, the adversary has increased capabilities to prune “empty spaces” that do not belong to roads.

REFERENCES

Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: differential privacy for location-based systems. In *2013 ACM SIGSAC Conference on Computer and Communications Security*.

Mikhail J. Atallah. 1998. *Algorithms and Theory of Computation Handbook*. CRC Press.

Konstantinos Chatzikokolakis, Miguel E. Andrés, NicolsEmilio Bordenabe, and Catuscia Palamidessi. 2013. Broadening the scope of Differential Privacy using metrics. In *Symposium HotPets 2013. OnLine version:* http://freehaven.net/anonbib/papers/pets2013/paper_57.pdf.

Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Marco Stronati. 2014. A Predictive Differentially-Private Mechanism for Mobility Traces. In *Proc. of Privacy Enhancing Technologies - 14th International Symposium, PETS*.

Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. 2006. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *Proc. International Conference on Privacy Enhancing Technologies (PET)*.

Maria Luisa Damiani. 2014. Location privacy models in mobile applications: conceptual view and research directions. *GeoInformatica* 18, 4 (2014), 819–842.

Maria Luisa Damiani, Elisa Bertino, and Claudio Silvestri. 2010. The PROBE Framework for the Personalized Cloaking of Private Locations. *Transactions on Data Privacy* (3)2, 2001-145 (2010), 123–148.

Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. 2011. Fine-Grained Cloaking of Sensitive Positions in Location-Sharing Applications. IEEE Pervasive Computing. *IEEE Pervasive Computing* 10(4) (2011), 64–72.

Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. 2000. *Computational Geometry: Algorithms and Applications* (2nd ed.). Springer-Verlag.

Cynthia Dwork. 2006. Differential Privacy. In *ICALP (2)*. Springer, 1–12.

Fox News. *Man Accused of Stalking Ex-Girlfriend With GPS. http://www.foxnews.com/story/0,2933,131487,00.html. Sep 04, 2004*.

Dario Freni, Carmen Ruiz Vicente, Sergio Mascetti, Claudio Bettini, and Christian Jensen. 2010. Preserving Location and Absence Privacy in Geo-social Networks. In *Proc. of the 19th ACM International Conference on Information and Knowledge Management*.

Bugra Gedik and Ling Liu. 2005. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proc. ICDCS*.

Gabriel Ghinita. 2013. *Privacy for Location-Based Services*. Morgan & Claypool Publishers.

Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian Lee Tan. 2008. Private Queries in Location Based Services: Anonymizers are not Necessary. In *SIGMOD*.

Marco Gruteser and Dirk Grunwald. 2003. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of USENIX MobiSys*.

Marco Gruteser and Xuan Liu. 2004. Protecting Privacy in Continuous Location-Tracking Applications. *IEEE Security and Privacy* 2 (2004), 28–34. Issue 2.

Jeff Henrikson. 1999. Completeness and Total Boundedness of the Hausdorff Metric. *MIT Undergraduate Journal of Mathematics* 1 (1999), 69–80.

Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. 2007. Preserving Location-based Identity Inference in Anonymous Spatial Queries. *IEEE TKDE* 19, 12 (2007).

Ali Khoshgozaran and Cyrus Shahabi. 2007. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In *SSTD*.

Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. 2005. An anonymous communication technique using dummies for location-based services. In *International Conference on Pervasive Services (ICPS)*. 88–97.

John Krumm. 2009. A survey of computational location privacy. *Personal and Ubiquitous Computing* (13)6 (2009), 391–399.

Yanhui Li, Ye Yuan, Guoren Wang, Lei Chen, and Jiajia Li. 2016. Semantic-Aware Location Privacy Preservation on Road Networks. In *Proc. of International Conference on Database Systems for Advanced Applications (DASFAA)*.

Mohamed F. Mokbel, Chi Yin Chow, and Walid G. Aref. 2006. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proc. of VLDB*.

Alexandra-Mihaela Olteanu, Kvin Huguenin, Reza Shokri, and Jean-Pierre Hubaux. 2014. Quantifying the Effect of Co-location Information on Location Privacy. In *Proc. Privacy Enhancing Technologies*.

Carmen Ruiz-Vicente, Dario Freni, Claudio Bettini, and Christian Jensen. 2011. Location-Related Privacy in Geo-Social Networks. *IEEE Internet Computing* 15 (2011), 20–27.

Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying Location Privacy. In *Proc. IEEE Symposium on Security and Privacy*.

Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. 2012. Protecting location privacy: optimal strategy against localization attacks. In *In Proc. ACM Conference on Computer and Communications Security*.

Reza Shokri, Carmela Troncoso, Claudia Diaz, Julien Freudiger, and Jean-Pierre Hubaux. 2010. Unraveling an old cloak: k-anonymity for location privacy. In *Proc. ACM Workshop on Privacy in the Electronic Society*.

George Theodorakopoulos, Reza Shokri, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. 2014. Prolonging the Hide-and-Seek Game: Optimal Trajectory Privacy for Location-Based Services. In *Proc. Workshop on Privacy in the Electronic Society*.

Toby Xu and Ying Cai. 2009. Feeling-based location privacy protection for location-based services. In *Proceedings of the 16th ACM conference on Computer and communications security*.

Emre Yigitoglu, Maria Luisa Damiani, Osman Abul, and Claudio Silvestri. 2012. Privacy-preserving sharing of sensitive semantic locations under road-network constraints. In *IEEE MDM*.

Man Lung Yiu, Christian Jensen, Xuegang Huang, and Hua Lu. 2008. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In *International Conference on Data Engineering (ICDE)*. 366–375.