Integrating Identity, Location, and Absence Privacy in Context-aware Retrieval of Points of Interest

Daniele Riboni Linda Pareschi Claudio Bettini
Universita' degli Studi di Milano, D.I.Co., EveryWare Lab
Email: {riboni,pareschi,bettini}@dico.unimi.it

***Abstract*—The retrieval of close-by points of interest (POIs) is becoming a popular location-based service (LBS), often integrated with navigational services and geo-social networks. However, the access to POI services is prone to potentially serious privacy issues, since requests for POIs often include sensitive information like the user's location and her personal interests. Many techniques to enforce privacy in LBS have been proposed in the literature, in some cases focusing on anonymizing the requests and in others on obfuscating information in order to decrease its sensitivity. In many cases privacy protection comes at some cost in terms of service precision and performance. In this paper we propose a novel technique that combines the above cited approaches, overcomes some of their limitations in terms of assumptions on adversary knowledge, while still guaranteeing service precision. Our privacy solution has been integrated in an existing distributed system to share and retrieve POIs based not only on the user's current location but also on other (possibly sensitive) context data.**

I. Introduction

Services for retrieval of points of interest (POIs) are becoming increasingly popular due to the widespread diffusion of GPS-enabled mobile devices having access to fast wireless networks. However, the access to such services is prone to potentially serious privacy issues, since requests include sensitive information, and they are often handled by untrusted parties. Different privacy threats may arise, depending on the external knowledge available to an adversary, and on the kind of information that is considered sensitive by a user.

Example 1: Alice gets invited by her colleagues to have dinner together; however, she prefers to spend the night with her best friend. Since she is afraid that they would be disappointed by that, she tells them that she cannot go out that night because she is not feeling well, and that she will spend the night at home. Since some of her colleagues are part of Alice's contacts in a geo-referenced social network, she inserts her home as her (fake) location for the whole night to be consistent with the information she has given to the colleagues. Later, when she meets with her friend in the city center, she would like to use an LBS from her smartphone to look for nearby pubs, but would like to keep this request as private, even to the untrusted service provider.

The above example shows that different kinds of privacy issues may arise from the use of location-based services (LBS). Indeed, in addition to protect from *i)* the disclosure of an individual's exact location ("*Alice is at Macy's in the city center*"), in certain cases it is important also not to disclose *ii)* sensitive information included in an LBS request ("*Alice is looking for a pub*"), and *iii)* the fact that an individual is not in a given location at a given time ("*tonight Alice is not at home*"). In particular, location is often sensitive information that a user prefers not to be associated with her identity. In this case, we say that the user has a *location privacy* concern. In some cases, the user may not want to disclose the fact that she is not in a given location at a given time (*absence privacy* concern). In other cases, the interest for specific resources is considered sensitive, and the user needs an anonymous access to the service (*identity anonymity* concern). However, it is well known that the use of a pseudonym is not sufficient to enforce anonymity, since other data in the request (or in a sequence of requests), including location, may be used by an adversary to re-identify the issuer based on external information [1].

Different techniques have been proposed for protecting against privacy concerns in LBS. Most of these techniques rely on trusted third-party components to enforce identity anonymity (e.g., [2]), possibly coupled with obfuscation of location information [3]. Of course, relying on a third-party anonymizer may determine scalability and reliability issues.

A different approach has been proposed to overcome these problems by using a client-server architecture in which the only trusted entity is the user's device. The proposed techniques are based on cryptographic functions, location generalization, or on the use of fake locations (a survey of these approaches can be found in [4]). Among the proposed client-server approaches, an interesting one is based on cryptographic techniques inspired by private information retrieval (e.g., [5]). These techniques provide very strong guarantees in terms of privacy, but they determine a relevant overhead in network and power consumption as well as in service response time.

Alternative client-server techniques are based on a perturbation of the user's location to enforce location privacy. In obfuscation techniques based on generalization, the exact location is enlarged to a region; in other cases, a fake user's location is communicated instead of the real one. In particular, the latter approach is adopted by *SpaceTwist* [6] to enforce location privacy while guaranteeing that k-nearest neighbor (kNN) queries are correctly answered, at the cost of computation and communication overhead. More recently, a technique to enforce both location privacy and identity anonymity, named *AnonTwist* [7], has been proposed. One of the advantages of AnonTwist with respect to other solutions for identity anonymity in LBS is that it does not rely on a trusted third party that provides precise information about the actual

presence of individuals in a given region. Instead, it relies on a density map that provides statistical information about the density of persons in specific areas at specific times. Density maps are currently available for many metropolitan areas, and it is reasonable to expect that services providing presence statistics will become more and more available in a near future. However, both SpaceTwist and AnonTwist rely on the implicit assumption that the parameters of the function that generates the fake location are unknown to the adversary. As we will show in Section III, without this assumption, those techniques may fail to protect users' privacy. Moreover, absence privacy is not enforced by those techniques. A technique to enforce absence privacy has been proposed in [8]; however, that technique relies on a delay in the release of sensitive data, and it is not feasible to POI services, in which service requests must be immediately issued.

In this paper we propose a novel technique to protect against all of the above mentioned issues while retaining the exactness of responses to LBS requests for POIs. With respect to SpaceTwist and AnonTwist, our solution takes into account the stricter, yet realistic, assumption that the adversary may know not only the defense algorithm but also its parameters. Moreover, our technique enforces absence privacy, without any delay in service requests. Our privacy solution has been integrated in an existing tool [9] to share and retrieve POIs based not only on location but also on other (possibly sensitive) context data such as current activity and interests.

The rest of the paper is structured as follows. In Section II we formalize the adversary model and the resulting privacy threats, motivating the need to couple location obfuscation with identity anonymity. In Section III we illustrate the limits of existing techniques. In Section IV we present our defense. In Section V we describe the integration of our defense into the POIsafe system. Section VI concludes the paper.

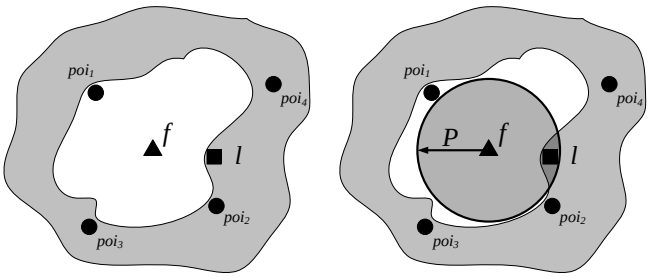
II. Adversary model and privacy threats

We assume that users issue context-aware requests to the service in order to find nearby POIs. Formally, a service request is a tuple $\langle STdata, SSdata \rangle$, where $STdata$ is spatio-temporal information about the location and time of the request, and $SSdata$ are service parameters. Before being issued, each request is modified on the user's client by the *defense algorithm*. Since privacy concerns depend on the individual's sensibility, as well as on her current context, we assume that every information in the request is potentially privacy-sensitive. We assume that the adversary may know:

- the defense algorithm, including its parameters;
- users' requests and service responses;
- a *density map* providing statistical information about the density of persons in specific areas;
- the current location of some individuals at certain times.

Given a user issuing requests for POIs, the adversary's goal is to associate the request issuer to this sensitive information:

- the precise user's location at the time of the request (STdata);
- sensitive information explicitly communicated within requests (SSdata);
- the fact that the user is not in a given location at the time of the request (if this is considered sensitive by the user).



(a) The adversary does not know P (b) The adversary knows P

Fig. 1. Adversarial inference against SpaceTwist and AnonTwist defenses. Symbol ■ represents the user's real location l; ▲ the fake location f; • the returned POIs. When the adversary ignores the maximum perturbation radius P (Figure 1(a)), the user's candidate area corresponds to the twisted space (gray area). When the adversary knows P (Figure 1(b)), the candidate area corresponds to the intersection between the obfuscation area (calculated based on P and f, and shown by the gray circle) and the twisted space.

The objective of our defense technique is to enforce location privacy, identity anonymity, and (when this is required by the user) absence privacy. To this aim, our technique applies a combination of location obfuscation and identity anonymity.

Why enforcing anonymity, in addition to obfuscation? Suppose that the whole sensitive information in a service request is obfuscated, so that it does not determine any privacy issue for its respondent. In this case, one may argue that anonymity is useless, since, even if an adversary would identify the request issuer, this would not threaten her privacy. However, in certain cases, the fact of accessing to a particular service may be a sensitive information by itself: several popular services, indeed, are targeted to specific categories of users, which are characterized by political opinions, religious beliefs, or sexual behaviors. In order to obfuscate the access to a specific service, the user should submit (fake) requests to many other services; however, that technique would obviously incur high overhead. Even when the access to a specific service is not sensitive information, the obfuscation of several service parameters would determine high overhead, since a large superset of the precise kNN set of POIs should be retrieved from the server, and then refined client-side, to include the precise kNN set.

Why enforcing obfuscation, in addition to anonymity? Theoretically, identity anonymity is sufficient to protect against privacy concerns when accessing a mobile service. Indeed, when anonymity is enforced, neither an adversary can associate sensitive information in the request to the identity of the issuer, nor he may understand that a user actually made a request to a specific service. However, in some cases, an adversary having access to additional external information than users' location may be able to re-identify the request issuer; for instance, the adversary may know that the actual issuer is the only habitual user of the service among the candidate ones, or because he may know that some information in the

requests is clearly referable to that individual. By enforcing obfuscation of STdata, we avoid the disclosure of sensitive location information, even when the adversary is able to violate identity anonymity. Moreover, by obfuscating STdata we can counteract some shortcomings of identity anonymity techniques; e.g., threats due to the homogeneity of sensitive information included in requests issued from the same area during the same time granule, and to the correlation of sensitive information in recurrent location-based queries [10].

III. Limitations of current approaches

In the following, we outline the SpaceTwist and AnonTwist algorithms, and we explain the adversarial inference that may lead to the disclosure of sensitive information.

A. SpaceTwist and AnonTwist algorithms

The goal of SpaceTwist is to enforce location privacy. According to SpaceTwist, a fake user's location is communicated, instead of the real one, in service requests. Then, in order to guarantee that kNN queries are correctly answered, the client issues a sequence of requests from the same fake location asking for more close-by POIs, until it is sure that those provided by the service include the kNN set corresponding to its real location. Based on the request-response sequence, an adversary can only identify an area (called *twisted space*) from which the requests may have been sent.

On the other hand, the goal of AnonTwist is to enforce both location privacy and identity anonymity, by a modification of the SpaceTwist algorithm. With AnonTwist, the client continues issuing requests until both the following conditions are satisfied: *i)* service responses include the kNN set corresponding to its real location, and *ii)* the twisted space contains at least N individuals. The latter condition has been introduced to enforce identity privacy. The practical advantage of AnonTwist with respect to other solutions for identity anonymity in LBS is that it does not rely on a trusted third party that provides precise information about the presence of individuals in a given region. Instead, it relies on statistical information about the density of persons in specific areas.

B. Limitations

Both SpaceTwist and AnonTwist rely on the implicit assumption that the adversary does not know the parameters of the function generating the fake location. Under that assumption, those algorithms guarantee that the *candidate area* (i.e., the one corresponding to the possible locations of the user) corresponds to the twisted space. However, without that assumption, the privacy guarantees of SpaceTwist and AnonTwist are no more valid. In particular, the knowledge of the maximum distance P among the fake and the real user's location is sufficient to break both location privacy and identity anonymity. Indeed, in both SpaceTwist and AnonTwist, the fake location f is randomly chosen inside a circle having radius P, and center in the real user's location l. Hence, when an adversary knows P and f, he can easily infer that l is within the circle having center f and radius P; we call this circle *obfuscation area*. After observing a sequence of service request/responses, the adversary can calculate the twisted space, which includes the user's location (Figure 1(a)); then, he can calculate the candidate area as the intersection between the twisted space and the obfuscation area (Figure 1(b)). Of course, since the candidate area may be a small subset of the twisted space, the privacy properties of SpaceTwist and AnonTwist are no more guaranteed when P is known.

Algorithm 1: PUPPET defense algorithm

Input: δ minimum dimension of the candidate area; α minimum anonymity level; k number of required NN POIs; l user's real location; t temporal characterization of the requests; $SSdata$ service parameters; P maximum obfuscation radius; p puppet location; τ time-to-live; D density map.
Output: H_k: k nearest POIs with respect to l.

PUPPET(δ, α, k, l, t, $SSdata$, P, p, τ, D) begin
  H_k := new MaxHeap(); /* *containing nearest POIs* */
  f := FakeLocation(l, P);
  O := ObfuscationArea(f, P);
  C_0 := O; /* *initial candidate area* */
  $\gamma := \infty$; /* *SpaceTwist parameters* */
  $\lambda := 0$; /* *SpaceTwist parameters* */
  $i := 0$;
  while $\Big((i < \tau) \ \wedge \ \Big(\big(\gamma + \text{dist}(l, f) > \lambda\big) \ \vee \ \big(\mathit{Dim}(C_i) < \delta\big) \ \vee \ \big(\mathit{Population}(C_i, t, D) < \alpha\big) \ \vee \ (p \notin C_i)\Big)\Big)$ **do**
    $i := i + 1$;
    poi_i := get-i^{th}-PoiFromServer(f,$SSdata$,i);
    λ := dist(f, poi_i);
    if $\big(\mathit{dist}(l, poi_i) < \gamma\big)$ **then**
      update(H_k, γ, poi_i);
    T_i^S := TwistedSpace(f, k, poi_i, poi_{i-1}, ..., poi_1);
    if $\Big(\mathit{Dim}(C_{i-1}) \geq \delta\Big) \ \wedge \ \Big(\mathit{Population}(C_{i-1}, t, D) \geq \alpha\Big) \ \wedge \ \Big(p \in C_{i-1}\Big)$ **then**
      $C_i := T_i^S \cap O$;
    else
      $C_i := C_{i-1} \cup T_i^S \cap O$;
  return H_k;
end

IV. Defense technique

The objective of our defense is to provide precise responses to LBS requests for POIs while enforcing identity anonymity and location privacy. Moreover, when the user wants to conceal the fact that she is not in a given location at a given time, our technique also enforces absence privacy.

A. The PUPPET algorithm

The specific goal of the *POIsafe Privacy Preserving Technique* (PUPPET) is that the *candidate area* (i.e., the one containing the actual user's location) that an adversary may

reconstruct after observing a sequence of requests/responses to the POI service has the following properties:

$p1$) it contains, with high probability, at least α persons at the time the first request has been issued;
$p2$) its dimension is greater than a threshold δ;
$p3$) (if it is required by the user) it contains the user's *puppet location* p; i.e., a location from which the user wants to conceal her absence at service request time.

The first property above is to enforce identity anonymity; the second one to enforce location privacy; the third one to enforce absence privacy. The PUPPET algorithm (shown in shown in Algorithm 1) is based on a significant modification of SpaceTwist and AnonTwist; it takes as input:

- the number k of required NN POIs;
- the time t at which the first request has been issued;
- the real user's location l at t;
- (optionally) the user's puppet location p;
- the $SSdata$ of the request;
- the radius P of the obfuscation area;
- the time-to-live (TTL) τ; i.e., the maximum number of requests to be issued to retrieve the exact set of kNN POIs;
- the density map D;
- the required minimum dimension δ of the candidate area;
- the required minimum number α of people in the candidate area at time t.

The algorithm returns the exact set of kNN POIs with respect to the user's real location.

At first (line 2), the algorithm initializes the max-heap H_k, which is used to store the k nearest POIs received by the server. Then (line 3), a fake location f is randomly chosen inside the circle having center l and radius P. At line 4, the obfuscation area is calculated, as the circle having center f and radius P. Note that the maximum distance among the user's real location l and the puppet location p must be $\frac{P}{2}$; otherwise, it cannot be guaranteed that the obfuscation area contains p[1]. The initial candidate area C_0 is set to the obfuscation area (line 5). At lines 6 and 7, the parameters γ and λ are initialized; those parameters are used to check whether the exact set of kNN objects has been received by the server, according to the SpaceTwist algorithm [6]. A counter i is set (at line 8) to keep track of the number of POIs already received by the server.

After initialization, the algorithm starts querying the server for POIs, until either the TTL has expired, or properties $p1$), $p2$), and $p3$) are satisfied. The termination conditions of the algorithm are inside the *while* clause at line 9. The algorithm continues asking for POIs while the TTL has not expired (first condition) and at least one of the following conditions holds:

- the set of POIs retrieved from the server does not include the exact kNN set (second condition); or,
- the dimension of the candidate area is smaller than δ (third condition); or,

[1]As it will be explained in Section V, when the distance between p and l is larger than $\frac{P}{2}$, the user may decide to give up absence privacy, while keeping location privacy and identity anonymity protection.

Algorithm 2: Adversary's inference algorithm

Input: δ minimum dimension of the candidate area; α minimum anonymity level; k number of required NN POIs; t temporal characterization of the requests; f fake location included in requests; P maximum obfuscation radius; (optional) p puppet location; τ time-to-live; D density map; poi_1, poi_2, ..., poi_i POIs received by the user.
Output: Candidate area C_i

1 **Attack**(δ, α, k, t, f, P, τ, D, poi_1, poi_2, ..., poi_i) **begin**
2 O := ObfuscationArea(f, P);
3 C_0 := O; /* *initial candidate area* */
4 **for** j := 1 *to* i **do**
5 T_j^S := TwistedSpace(f, k, poi_j, poi_{j-1}, ..., poi_1);
6 **if** $\left(Dim(C_{j-1}) \geq \delta\right) \wedge \left(Population(C_{j-1}, t, D) \geq \alpha\right) \wedge \left(p \in C_{j-1}\right)$ **then**
7 $C_j := T_j^S \cap O$;
8 **else**
9 $C_j := C_{j-1} \cup T_j^S \cap O$;
10 **return** C_i;
11 **end**

- according to D, at time t, the candidate area contains less than α persons (fourth condition); or,
- the candidate area does not include the puppet location p (fifth condition); this condition is evaluated only when the user requires to enforce absence privacy.

When the ith POI (poi_i) is received from the server (line 11), its distance from the user's fake and real location is calculated, in order to update the max-heap H_k (lines 12 to 14). Then, the twisted space T_i^S is calculated as explained in [6]. The next operation is to calculate the candidate area according to the adversary's reasoning. If, after receiving poi_{i-1}, the conditions for location privacy, identity anonymity, and absence privacy were satisfied (lines 16 and 17), this means that the exact kNN set was not yet retrieved after obtaining poi_{i-1} (otherwise, the algorithm would have stopped, and poi_i would not have been asked to the server). However, according to the adversary, the exact kNN set may have been retrieved after obtaining poi_i. Hence, following the adversary's reasoning, the candidate area C_i is set to the intersection between the obfuscation area O and the twisted space T_i^S. Otherwise (line 19), it is possible that the exact kNN set was retrieved either after obtaining poi_{i-1} or after obtaining poi_i: under the first assumption, the user's real location l belongs to the candidate area C_{i-1}; under the second assumption, l belongs to the intersection between the obfuscation area O and the twisted space T_i^S. Hence, the candidate area C_i is set to the union of C_{i-1} and T_i^S, intersected by O.

B. Defense effectiveness and performance

According to our assumptions, the adversary may know every input parameter of the defense algorithm, except the user's location l. Moreover, since the adversary may observe

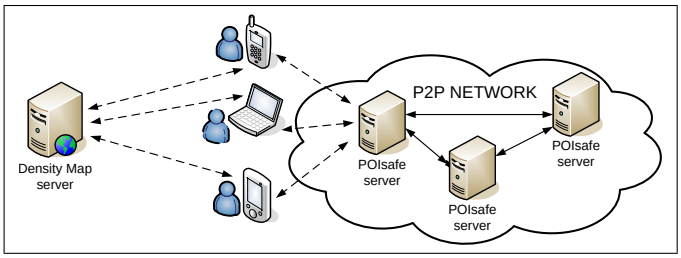


Fig. 2. The POIsafe network

service requests and responses, he may know the fake location included in requests, as well as the POIs received by the user in responses: his objective is to reason with this information to reconstruct the candidate area that includes the user's location.

Algorithm 2 illustrates the adversarial reasoning (privacy attack). At first (line 2), the adversary calculates the obfuscation area O, as the circle having center in the fake location f, and radius P. The candidate area is initialized to O (line 3). After initialization, the adversary starts to calculate the candidate area, considering the POIs iteratively received by the user (lines 4 to 9). Given the jth POI (poi_j), the adversary calculates the twisted space (line 5). The following reasoning (lines 6 to 9) is analogous to the one of the defender (lines 16 to 19 in Algorithm 1, and explained in Section IV-A). Finally, the algorithm returns the candidate area C_i.

After calculating the candidate area, the adversary should distinguish the case in which the user stopped sending requests *before* the TTL elapsed from the one in which she stopped *after* it elapsed. Below, we show that, in both cases, the PUPPET defense enforces identity anonymity, location privacy, and absence privacy.

- *The user stopped sending requests before the TTL elapsed.* In this case, according to the defense algorithm, the user received the exact kNN set, and stopped sending requests because the conditions for identity anonymity, location privacy, and absence privacy, were satisfied.
- *The user stopped sending requests after the TTL elapsed.* In this case, two situations may have occurred.
  - *The user has not received the exact kNN set.* In this case, the candidate area C_i does not include the user's real location.
  - *The user has received the exact kNN set.* In this case, C_i includes the user's real location.

  Note that the adversary cannot know whether the user actually received the exact kNN set, since he does not know the user's precise location. When the user stops sending requests after the TTL elapsed, it is possible that the candidate area C_i violates one or more of the conditions for identity anonymity, location privacy, and absence privacy. However, even in this case, the user's privacy is preserved: indeed, since the adversary cannot know whether the user actually received the exact kNN set, he cannot know whether the user's precise location actually belongs to the candidate area.

In terms of performance, the PUPPET algorithm is comparable with AnonTwist, while protecting against a larger class of privacy attacks. Indeed, the only differences that may impact performance are due to *a)* the different calculation of the candidate area (lines 16 to 19 in Algorithm 1), and *b)* the evaluation of the condition for absence privacy in the termination condition (line 9). While the different method for calculating the candidate area does not determine significant computational overhead, in order to enforce the additional requirement of absence privacy, it is possible that a larger number of POIs must be downloaded from the server. However, we recall that absence privacy is an optional feature of our privacy protection mechanism, and may be disabled by the user to improve the performance.

V. THE POISAFE SYSTEM

The PUPPET defense has been integrated into the POIsafe system for privacy-conscious sharing and retrieval of an extended form of POIs, called *POIsmarts* [9], which are the convergence between physical and virtual POIs; the latter being essentially Web resources related to physical spots.

A. Architecture

The POIsafe system (Figure 2) is based on a peer-to-peer network of POIsafe servers. Mobile client systems with different capabilities can be used to access the POIsafe network, to share and search for POIs. POIsafe clients query an external server to obtain the density map, and issue requests to the POIsafe network to retrieve POIs according to the PUPPET algorithm. For the sake of interoperability, all the entities in the POIsafe network communicate by means of Web services.

POIsafe servers are in charge of managing POIsmarts, and of searching them in the peer-to-peer network based on user's context (location, activity, ...) and explicit search keywords. The search mechanism is presented in detail in Section V-B. POIsafe servers would reside on department servers, or on personal servers in a cloud infrastructure.

Client systems provide an interface for the user to browse and reorganize her own POIsafe hierarchy, add new POIs, and search shared POIs in the network. As illustrated in Section V-C, client systems make use of user-friendly interfaces to assist the user in specifying her privacy preferences. Client systems are device-dependent. While the reference device is the smartphone, two basic communication models are supported: a) clients on devices that are not always online can use a local copy of the complete hierarchy, and synchronize with the POIsafe server when connected or upon user request; b) clients on devices that are always connected but that have limited resources store only part of the POIsafe hierarchy on the device, and any change server-side is immediately communicated to the client.

Density Map server: it is in charge of providing statistics about the number of people in a certain area at a certain time. When querying the density map server, client systems specify their current location at a very coarse granularity (e.g., "Downtown Los Angeles"); in order to avoid location

privacy issues. Moreover, density map information may be downloaded periodically and not necessarily at the time of the request; the density information is associated with validity time granules, and the selection of the time granule containing the instant of service request is performed client-side. While density map data is currently available for limited areas, we expect the diffusion of presence statistics services as part of the global offer of location services in a near future.

B. Context-aware POIs retrieval

POIsmarts are selected and ordered based on a *scoring function* that provides a relevance value for the POIs with respect to the current user's context. The scoring function is mapped to a multi-feature query that is locally executed by the peer that receives the request, and propagated into the POIsafe network (details can be found in [9]). Queries for searching POIsmarts include a set of keywords, the ordinal number of the POI, the fake user's location, and possibly other context data, such as the user's current activity and interests. The list of POIsmarts to be presented to the user is calculated on the client system by merging the POIsmarts obtained by the POIsafe network with the ones obtained by the local repository; POIsmarts are re-ordered on the client based on the user's real location.

C. Implementation

We have developed a prototype implementation of the whole POIsafe system. The POIsmart server has been developed in Java; multi-feature queries are executed by a dedicated software module exploiting the MySQL DBMS. The density map server exploits the spatial extensions of the postgreSQL DBMS, and covers the metropolitan area of Milano. At the time of writing, we have developed client systems for different mobile platforms, including Android, Java ME, and the .Net compact framework. The Android client (shown in Figure 3) takes advantage of the integration with Google Maps to graphically illustrate the returned POIs, and to provide directions.

A very important feature of client systems is the adoption of user-friendly interfaces to let users express their privacy preferences. Indeed, the average user has no acquaintance with concepts of anonymity and obfuscation; hence, she must be assisted in specifying the privacy parameters of our defense technique. To this aim, privacy parameters are automatically chosen by the client application based on high-level privacy preferences expressed by the user through graphical interfaces. For instance, in the Android client, a slider is used to let the user express her preferred tradeoff between fast response time and privacy. The user can also choose a puppet location of herself to protect absence privacy, by pointing to it on a map. As explained in Section IV, it is possible that the puppet location may be too distant from the user's exact location to enforce absence privacy without incurring excessive overhead. When this situation occurs, the user is notified by the client application, and she may choose to whether give up absence privacy, or avoid to submit the request.

Fig. 3. POIsafe client for the Android platform

VI. Conclusions and Future Work

In this paper we proposed a novel technique to protect the privacy of users accessing to mobile services for POIs retrieval. This technique overcomes serious limitations of existing approaches, while retaining the precision of service responses. Our solution has been integrated into a context-aware system for sharing and retrieval of POIs based on a wide notion of context. Future work includes a thorough experimental evaluation of our technique, in particular, to evaluate the impact of absence privacy on the system performance.

References

[1] C. Bettini, X. S. Wang, and S. Jajodia, "Protecting privacy against location-based personal identification," in *Secure Data Management*, LNCS, vol. 3674. Springer, 2005, pp. 185–199.

[2] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 12, pp. 1719–1733, 2007.

[3] C.-Y. Chow, M. F. Mokbel, and W. G. Aref, "Casper*: Query processing for location services without compromising privacy," *ACM Trans. Database Syst.*, vol. 34, no. 4, 2009.

[4] C. S. Jensen, H. Lu, and M. L. Yiu, "Location privacy techniques in client-server architectures," in *Privacy in Location-Based Applications*, LNCS, vol. 5599. Springer, 2009, pp. 31–58.

[5] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: anonymizers are not necessary," in *Proc. of ACM SIGMOD'08*. ACM, 2008, pp. 121–132.

[6] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu, "Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services," in *Proc. of ICDE'08*. IEEE Computer Society, 2008, pp. 366–375.

[7] S. Wang and X. Wang, "AnonTwist: Nearest Neighbor Querying with Both Location Privacy and K-anonymity for Mobile Users," in *Proc. of MDM'09 workshops*. IEEE Computer Society, 2009.

[8] D. Freni, C. R. Vicente, S. Mascetti, C. Bettini, and C. S. Jensen, "Preserving location and absence privacy in geo-social networks," in *Proc. of CIKM'10*. ACM, 2010, pp. 309–318.

[9] C. Bettini and D. Riboni, "Context-aware Web Services for Distributed Retrieval of Points of Interest," in *Proc. of ICIW'07*. IEEE Computer Society, 2007.

[10] D. Riboni, L. Pareschi, C. Bettini, and S. Jajodia, "Preserving Anonymity of Recurrent Location-based Queries," in *Proc. of TIME'09*. IEEE Computer Society, 2009.