Privacy-Aware Proximity Based Services

Sergio Mascetti Claudio Bettini Dario Freni
DICo
Università di Milano

X. Sean Wang
Department of CS
University of Vermont

Sushil Jajodia
CSIS
George Mason University

Abstract

Proximity based services are location based services (LBS) in which the service adaptation depends on the comparison between a given threshold value and the distance between a user and other (possibly moving) entities. While privacy preservation in LBS has lately received much attention, very limited work has been done on privacy-aware proximity based services. This paper describes the main privacy threats that the usage of these services can lead to, and proposes original privacy preservation techniques offering different trade-offs between quality of service and privacy preservation. The properties of the proposed algorithms are formally proved, and an extensive experimental work illustrates the practicality of the approach.

1. Introduction

Location based services (LBS) are becoming popular thanks to the advances in positioning technologies and to the diffusion of mobile devices with data communication capabilities. *Proximity based services* are a special class of LBS in which the service adaptation depends on the comparison between a given threshold value and the distance between a user and other (possibly moving) entities. The so-called "friend-finder" services are an example: Alice would like to be alerted whenever her friend Bob is nearby, so that they could get in contact and possibly meet. The proximity service considered in this paper is a generalization of "friend-finder" in which each user is part of one or more possibly large and dynamically changing groups of users (called *buddies*) related to hobbies, sports, religious or cultural interests. Technically, a *proximity query* is a spatial range query on the database of moving buddies in which the range is defined by the circle centered at the issuer's location and having the proximity threshold as radius.

A major privacy concern with the use of LBS is the release to untrusted third parties of the user precise location information. This concern applies to proximity services as well: Alice would like to use the proximity service without necessarily releasing her exact position to the service provider (SP). In some cases, she may even wish not to provide the exact location to the buddies, although she may be willing to reveal whether she is in proximity. For example, she may agree to let Bob know that she is in a neighborhood near Bob's location, but keep the specific address hidden from Bob. In practice, this may avoid the situation in which buddies can directly walk to other buddies, as the goal of the service is usually to enable communication that may only eventually lead to meetings in person. A solution to the above privacy concern can be to allow each user to specify certain minimum location privacy requirements both with respect to the service provider and to the buddies. Note that these are minimum requirements, and a system should be designed with the goals to (1) guarantee the satisfaction of the minimum privacy requirements, and (2) reveal as little location information as possible. In this paper, we provide the design of several proximity service protocols toward these two goals. Among the different protocols achieving goal (1), we would like to identify the ones that are superior in terms of goal (2), the computation and communication cost, and the service precision. Here, service precision regards the approximate nature of the protocols that trade off location precision for privacy.

Several LBS privacy preserving techniques have been recently proposed [2]–[4], [7], [8]. Some techniques (e.g., [2], [4], [7]) consider the possible use of location information contained in anonymous LBS requests to discover the identity of the issuers and hence their connection with the private information such as the specific service being requested. Different spatial generalization functions are proposed to guarantee a given level of anonymity of the users. An anonymized request may contain a quite precise location, since a generalization is considered satisfactory whenever the region contains a sufficiently large number of potential issuers, independently from the actual size of the region. While effective in other scenarios, these techniques are less useful in the proximity services we are considering, since location is considered private information and we don't want to exclude that the identity of buddies may be discovered in other ways. Other techniques, including the one proposed in this paper, are more focused on the obfuscation of private information, and in particular user location. For example, SpaceTwist [8], which is specifically designed for K-NN queries, hides the location of the user by issuing a sequence of requests reporting fake locations. These requests are properly designed in order to obtain the same service as with a single request with the exact location.

Other interesting recent proposals exploit private information retrieval (PIR) techniques to encrypt all of the information exchanged with the service provider and to process the corresponding query in an encrypted form so that no location information is revealed to the SP. Among these proposals, the technique proposed in [3] is specifically designed for K-NN queries. The use of PIR techniques to compute range queries among static resources was proposed in [5]; however, it is not clear how these existing methods can be extended to computationally feasible solutions to handle range queries in which buddies, unlike fixed resources, are mostly moving around and possibly frequently leaving and entering a group. The research work closest to ours is [9] that presents a P2P solution based on public key encryption to solve the proximity problem. In our approach, since we consider dynamic groups, a preferable solution is one in which at least a first approximate proximity is computed by a privacy preserving protocol centralized at the SP.

The contributions of this paper are the following:

- To our knowledge, we propose the first privacy preserving protocols specialized for proximity services that takes advantage of a centralized protocol and that formally guarantees privacy with respect to SP and buddies;
- We illustrate three protocols providing different service precision, and we prove that each of them satisfies the user defined minimal privacy requirements;
- We report experimental results of the comparison of the protocols in terms of cost, performance and achieved privacy, highlighting the excellent properties of the protocol named *Hide&Crypt*.

The rest of the paper is organized as follows. In Section 2 we define the service, the privacy preferences, the threats, and the evaluation metrics. In Section 3 we illustrate the proposed protocols and their formal properties. In Section 4 we report the experiments, and in Section 5 we conclude with a short discussion.

2. Proximity services and privacy concerns

In this section we first formalize the proximity services and the related privacy requirements. We then describe how these requirements may be violated, and how different techniques avoiding these privacy threats could be compared.

2.1. The proximity service

The proximity service informally described in the introduction can be more precisely defined by considering a typical service provisioning session: a user A sends her own location information, acquired via GPS or other techniques, and requests to be alerted whenever a buddy[1] reports a location that is in proximity while it was not in proximity before, or vice versa is not anymore in proximity. Here, proximity is defined as being within a distance threshold given by A, denoted δ_A, i.e., A is interested in all the buddies B satisfying the following condition:

$$d(loc(A), loc(B)) \leq \delta_A \quad (1)$$

where $d(loc(A), loc(B))$ denotes the Euclidean distance[2] between the reported locations of A and B. When (1) is true, we say that B *is in the proximity of* A. Since δ_B may be different from δ_A, this proximity relation may not be symmetric.

In order to provide proximity service, it is convenient to have a service provider SP, especially when group sizes can be large and membership of the groups can dynamically change. Indeed, under these conditions, directly computing distances between A and every buddy can be extremely inefficient or even infeasible. Henceforth, we assume the existence of SP in providing proximity service.

With the presence of SP, and in the absence of privacy concerns, a simple protocol can be devised to implement the service: The SP receives from each user's automatic location updates and stores their last known positions, as well as the distance threshold δ_A for each user A. While in theory each user can define a different value δ_A for each buddy, in this paper, for the sake of simplicity, we consider the case in which each user defines a single value of δ_A. When the SP receives a location update, it may recompute the distance between A and each buddy (possibly with some filtering/indexing strategy for efficiency). If any proximity relation changes, A is notified. In a typical scenario, if B is in proximity, A may contact him directly or through SP; however, for the purpose of this paper, we do not concern ourselves as what A will do once notified.

2.2. User minimum privacy requirements

The privacy we are considering in this paper is *location privacy*, i.e. we assume that users are concerned about other persons obtaining information about their exact location at specific times. In the considered services, users may prefer the service provider to have as little information about their location as possible, and the buddies not to know her exact position, even when proximity is revealed.

In general, the level of location privacy can be represented by the uncertainty that an external entity has about the

1. Despite in principle a user may be part of several dynamical groups, without loss of generality in the following we consider each user belonging only to a single group. Hence, her buddies are the users of that specific group.

2. In this paper we consider the Euclidean distance only, but our approach can be easily extended to other distance functions, like, for example, shortest path on a road network.

position of the user, and this uncertainty can be formally represented as a geographic area in which no point can be ruled out as a possible position of the user. In principle each user could express her privacy preferences, by specifying for each other user (or class of users perceived as adversaries) a partition of the geographical space defining the *minimal uncertainty regions* that she wants to be guaranteed. For example, Alice specifies that Bob should never be able to find out the specific building where Alice is within the campus, i.e., the entire campus area is a minimal uncertainty region. The totality of these uncertainty regions can be formally captured with the notion of *spatial granularity*.

While there does not exist a formal definition of spatial granularity that is widely accepted by the research community, the idea behind spatial granularities is simple. Similarly to temporal granularity [1], a spatial granularity can be considered a subdivision of the spatial domain into a discrete number of non-overlapping regions, called *granules*. In this paper, for simplicity, we consider only granularities[3] that partition the spatial domain. In principle, granules of the same granularity can have any shape and don't need to have the same size or shape. Each granule of a granularity G is identified by a *index* (or a *label*). We denote with $G(i)$ the granule of the granularity G with index i.

Users specify their *minimum privacy requirements* via spatial granularities, with each granule being a minimum uncertain region. In the following of this paper we assume that each user specifies two granularities

$$G_A^{SP} \text{ and } G_A^U$$

defining the minimum location privacy requirements for SP and for any other user, respectively, as the two categories of potential adversaries.

The two extreme cases in which a user requires no privacy protection and maximum privacy protection, respectively, can be naturally modeled. For example, if a user A does not want her privacy to be protected with respect to other buddies (in this case A can tolerate other buddies to know her location at the maximum available precision) then A will set G_A^U to the *bottom granularity* $\perp$ (a granularity that contains a granule for each basic element, or pixel, of the spatial domain). Similarly, if A wants to impose the maximum privacy protection with respect to the SP, then A sets G_A^{SP} to the *top granularity* $\top$ (the granularity that has a single granule that covers the entire spatial domain). In Section 4 we show how changes of the minimum privacy requirements affect system performance.

2.3. Privacy threats

In order to formally identify the privacy threats, it is crucial to first specify the assumptions about the available external knowledge and about the behavior of considered adversaries. In this paper we consider both buddies and SP (as well as external entities that may have taken control of one of them) as potential adversaries, and we assume the following: (a) there is no external knowledge on the location of users other than the one exchanged during the protocol, and (b) buddies and SP do not collude. The formal proofs of our results also assume that the involved entities are not malicious, in the sense that they follow the protocols defined in the service.

Observe the simple protocol described in Section 2.1; it is easily seen that even under the above assumptions the location privacy of users is at risk. When the SP is considered as a potential adversary, an *SP-threat* can be identified: Since the exact location of user A is stored by the SP, if G_A^{SP} is not the bottom granularity, A's minimum privacy requirement is violated. When a buddy is considered as a potential adversary, a *buddy-attack* can be identified as follows. Suppose A is a buddy of B who sets a value of δ_B in a way such that the circular region of radius δ_B (centered at B's location) is properly contained in a granule of G_A^U. Then, if A happens to enter that circular region, the SP will notify B, and A's minimum privacy requirement would be violated.

2.4. Privacy protection performance

In the sequel, we present techniques to protect user privacy. The minimum goal of our techniques is to guarantee the satisfaction of users' minimum privacy requirements, which all of our protocols provide. However, there are three additional performance goals to be considered.

The first is based on the observation that more privacy is generally more desirable by users; we should strive to provide larger uncertainty region than the minimum ones given by the user. Hence, the first performance measure of the protection is the *size of the uncertainty region*. The larger the uncertainty region, the better.

The second performance goal is to minimize the *system costs*, including computation and communication. As we show later, there is a trade-off between the privacy level and the costs.

The third performance metrics is the *service precision*. Due to the user minimum privacy requirement, there may be uncertainty whether user B is actually in proximity of A. We take a conservative approach, namely when it's uncertain, we report to A that B is in proximity. The service precision is then defined as the percentage of times that B is indeed in A's proximity when alerted (based on the reported locations of A and B) among all the times A is alerted. Obviously, the higher precision, the better.

3. Here and in the following, when no confusion arises, we use the term "granularity" to mean "spatial granularity".

3. Privacy preserving techniques

A first solution to protect users' privacy is to encrypt the location information each user sends to the SP and devising a secure computation method for obtaining the distance between the users. However, in the applicative context we consider in this paper, this solution is not practical. Indeed, since the set of users in each group is potentially large and changes dynamically, maintaining a shared secret of the group may involve high costs. A solution that does not require a shared secret cannot be applied, either, since it would necessarily require the SP to contact every buddy each time any of the other buddies updates her location. This is clearly infeasible due to communication and computation costs both on the client and the SP sides.

In this paper, we consider a hybrid approach in which a secure computation is performed only after a filtering step based on obfuscated locations. More precisely, we present three different privacy preserving protocols, the first of which is called *SP-Filtering* and does not involve any communication between the buddies to evaluate their proximity. The other two, named *Hide&Seek* and *Hide&Crypt*, provide a more accurate estimation of the proximity by using *SP-Filtering* as the first step followed by a refinement step involving buddy-to-buddy communication. *Hide&Crypt* implements this last step with a secure computation. In this section we illustrate the protocols and their formal properties.

3.1. The *SP-Filtering* protocol

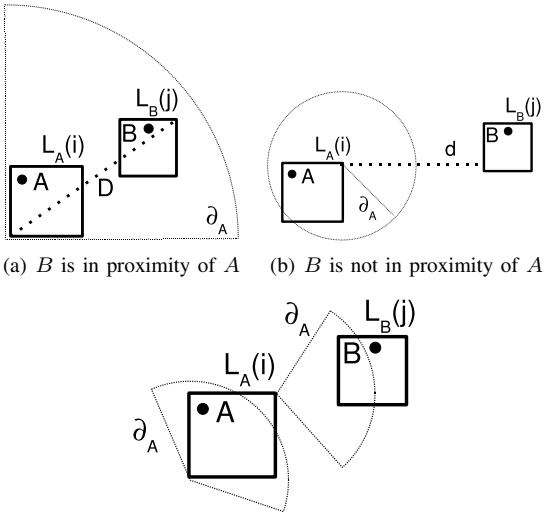
SP-Filtering is a three-party protocol that computes the proximity of B to A with a certain approximation, guaranteeing the satisfaction of the minimum location-privacy requirements of both A and B.

The idea of the algorithm is that when a user A performs a location update, instead of providing her exact location to the SP, she sends a generalized location that is computed as a function of G_A^U and the granule $G_A^{SP}(i)$ where A is located. More precisely, A sends to SP the location $L_A(i)$ that is computed as the union of the granules of G_A^U that intersects with $G_A^{SP}(i)$. Formally:

$$L_A(i) = \bigcup_{i' \in \mathbb{N} | G_A^U(i') \cap G_A^{SP}(i) \neq \emptyset} G_A^U(i')$$

Each buddy B does the same when location is updated with $L_B(j)$ similarly defined, where j is the index such that the location of B is in $G_B^{SP}(j)$. Then, the SP can compute, for each buddy B of A, the minimum and maximum distance between any two points of $L_A(i)$ and $L_B(j)$. We denote with d and D the minimum and maximum distance, respectively.

Given d and D, the SP can try to answer whether B is in the proximity of A or not. Indeed, if $D < \delta_A$, then B is in the proximity of A, independently from where exactly A and B are located within $L_A(i)$ and $L_B(j)$, respectively. Figure 1(a) shows an example of this situation. In this case, the SP sends the "B is in proximity" message to A. On the contrary, if $d > \delta_A$, then the SP can conclude that B is not in the proximity of A. Figure 1(b) graphically shows that this happens when, no point of L_B is in the proximity of A. In this case, the SP sends the "B is not in proximity" message to A (or stays silent depending on the service requested by A). Finally, if none of the two cases above happen (i.e., $d \leq \delta_A \leq D$), then the SP is not able to compute whether B is really in the proximity of A or not. See Figure 1(c) for an example: if A is located close to the top right corner of $L_A(i)$ and B is located close to the bottom left corner of $L_B(j)$, then B is in the proximity of A, otherwise he is not. However, since the SP does not know where A and B are located within the granules $L_A(i)$ and $L_B(j)$, respectively, it cannot precisely evaluate the proximity in terms of δ_A and sends the "B is possibly in proximity" message to A.



(a) B is in proximity of A (b) B is not in proximity of A

(c) B is *possibly* in proximity of A

Figure 1. Regions L_A and L_B

Proposition 1 formally states the location privacy provided by the *SP-Filtering* protocol.

The correctness of this formal result is based on an additional condition about the granules of G_A^{SP} and G_A^U that we require beyond the basic definitions of the spatial granularities. The condition is formally stated as

$$\forall g \in G_A^{SP}, h \in G_A^U \; g \subseteq h \text{ or } h \subseteq g \text{ or } h \cap g = \emptyset \qquad (2)$$

In other words, if a granule of G_A^{SP} and a granule of G_A^U have an intersection, then either the former totally contains the latter or vice versa. Since we assume a granularity "covers" the entire spatial domain, (i.e., the union of the granules of one granularity is the whole area), for each granule $G_A^U(j)$, there exists granule $G_A^{SP}(i)$ such that they intersect, and furthermore, if condition (2) is satisfied, it

must be the case that one of two granules totally contains the other. A sufficient condition to satisfy condition (2) is that either G_A^{SP} is a finer than G_A^U or vice versa[4].

When two users A and B run the *SP-Filtering* protocol, the SP learns the region $L_A(i)$ and $L_B(j)$ where the two users are located, but cannot exclude any location of these regions as possible location for A and B, respectively. When the *SP-Filtering* protocol is used to compute the proximity of B with respect to A, B does not receive any message and hence he does not acquire any information about the location of A. Vice versa, A can acquire some information about the possible location of B, depending on the message A receive from the SP. Proposition 1 formalizes the location knowledge that A acquires about B.

Proposition 1: Let A be located in $G_A^{SP}(i)$ and

$$S_{in} = \{j \in \mathbb{N} | MaxDist(L_A(i), L_B(j)) \leq \delta_A\}$$

$$S_{out} = \{j \in \mathbb{N} | MinDist(L_A(i), L_B(j)) \geq \delta_A\}$$

Whenever the *SP-Filtering* protocol is used to compute the proximity of B with respect to A:

1) if A receives the "B is in proximity" message from the SP, then A cannot exclude that B is located in any location of $Area_F^{in} = \bigcup_{j \in S_{in}} L_B(j)$;

2) if A receives the "B is not in proximity" message from the SP, then A cannot exclude that B is located in any location of $Area_F^{out} = \bigcup_{j \in S_{out}} L_B(j)$;

3) if A receives the "B is possibly in proximity" message from the SP, then A cannot exclude that B is located in any location of $Area_{NoFilter} = (Area_F^{in} \cup Area_F^{out})^C$.

In Theorem 1 we prove that the *SP-Filtering* protocol guarantees to protect the minimum location privacy requirements. The idea of the proof is that for each participating user C, $L_C(i)$ covers at least one granule of G_C^U and one of G_C^{SP}.

Theorem 1: Let C be a user participating in the *SP-Filtering* protocol. The minimum privacy requirements of C are guaranteed.

The computation complexity of the *SP-Filtering* protocol on the client and SP sides depends on the complexity of the operations applied on granularities (the computation of $L_C(i)$ on the client side for user C, and the computation of d and D on the SP side). In Section 4 we describe a class of spatial granularities for which these computations can be executed in constant time. In terms of communication cost, *SP-Filtering* requires each user to issue a message to the SP for each location update and the SP to issue a message to a user each time the proximity status with respect to any of her buddies needs to be updated.

4. The *finer-than* relationship was defined for temporal granularities (see, among others, [1]) and it can be easily extended for spatial granularities. Basically, G_A^{SP} is a finer than G_A^U if the former is a finer partitioning of the space than the latter is.

3.2. The *Hide&Seek* protocol

The main limitation of the *SP-Filtering* protocol is that, in order to protect the privacy of user C, granularity G_C^{SP} should be coarse. However, if G_C^{SP} is coarse, in many cases the SP in not able to compute whether a user is in the proximity, i.e., the case depicted in Figure 1(c). To address this problem, we now present the *Hide&Seek* protocol. The idea is that a user C can provide the SP with coarse location information and in case the SP is not able to determine the proximity with respect to another user, then the two users can run a two-party protocol to compute the proximity.

The two-party protocol is straightforward: A sends to B the values i' and δ_A where i' is the index of the granule of G_A^U where A is located. Since G_A^U is public, B can obtain it, for example from the SP. Then, B can compute d' as the minimum distance between any two points of $G_A^U(i')$ and $G_B^U(j')$ where $G_B^U(j')$ is the granule where B is located. If $d' > \delta_A$, then B sends to A the message "B is not in proximity". Otherwise, B can possibly be in proximity of A; In this case B sends to A the message "B is in proximity". Note here, the conclusion that "B is in proximity" is an approximate one as this can be wrong if δ_A distance is strictly judged. This is a necessary imprecision due to privacy protection, and as mentioned in Section 2.4, we take this imprecision as one performance measure of privacy protection techniques.

Protocol *Hide&Seek*
Prerequisites: A and B are running the *SP-Filtering* protocol. A is located in $G_A^U(i')$, B is located in $G_B^U(j')$.
Protocol:
1: A receives "B is possibly in proximity" from the SP
2: A sends to B "starting two-parties protocol $\langle i', \delta_A \rangle$"
3: B computes: $d' = MinDist(G_A^U(i'), G_B^U(j'))$
4: **if** $(d' \geq \delta_A)$ **then**
5: B sends to A "B is not in proximity"
6: **else**
7: B sends to A "B is in proximity"
8: **end if**

Note that the computation run by B during the *Hide&Seek* protocol is similar to the computation executed on the SP during the *SP-Filtering* protocol. The main difference is that, in this case, the location of A and B are generalized to the granularities G_A^U and G_B^U, respectively. In this case, B has more chances to be able to compute whether A is in the proximity, if granularities G_A^U and G_B^U are "finer-than" G_A^{SP} and G_B^{SP}, respectively. In this view, the *SP-Filtering*

protocol has the role of preventing A from starting the two-parties protocol with B, when not strictly necessary, hence reducing computation and communication overheads.

Whenever a user A runs the two-parties part of the *Hide&Seek* protocol to compute if B is in her proximity, the SP does not acquire any additional information about the locations of A and B. However, A acquires information about $B$and vice versa. Proposition 2 formalizes the location information A acquires about B and that B acquires about A.

Proposition 2: Let A be located in $G_A^U(i')$ and

$$S'_{out} = \{j' \in \mathbb{N} | MinDist(G_A^U(i'), G_B^U(j')) \geq \delta_A\}$$

Whenever the *Hide&Seek* protocol is used to compute the proximity of B with respect to A:

1) if A receives the "B is not in proximity" message from B, then A cannot exclude that B is located in any location of $Area_S^{out} = Area_{NoFilter} \cap \bigcup_{j' \in S'_{out}} G_B^U(j')$;

2) if A receives the "B is in proximity" message from B, then A cannot exclude that B is located in any location of $Area_S^{in} = Area_{NoFilter} \cap (Area_S^{out})^C$;

3) B cannot exclude that A is located in any location of $Area_S^{passive} = G_A^U(i')$.

Proposition 2 guarantees what user A can deduce on the position on B based on the messages A receives from B, and vice versa. Figure 2 may help in the understanding of the location privacy guarantees provided by the *Hide&Seek* protocol. User A receives the message "B is in proximity" or "B is not in proximity" from the SP when B is in close proximity ($Area_F^{in}$) or is far away ($Area_F^{out}$), respectively. If the SP is not able to compute whether B is in proximity of A, then A can infer that B is located in $Area_{NoFilter}$ ($Area^{in} \cup Area^{out}$ in Figure 2). More precisely, in this case, if A receives the message "B is not in proximity" or "B is in proximity" from B then A can deduce that B is located in $Area_S^{in}$ ($Area^{in}$, in Figure 2) or $Area_S^{out}$ ($Area^{out}$, in Figure 2), respectively.

Theorem 2 proves that the *Hide&Seek* protocol guarantees the minimum location privacy requirements.

Theorem 2: Let C be a user participating in the *Hide&Seek* protocol. The minimum privacy requirements of C are guaranteed.

The computational complexity of the *Hide&Seek* protocol is the same as the *SP-Filtering* protocol since the computation of d has the same complexity of the computation of d'. Hence, using the class of spatial granularities we present in Section 4, the computation can be executed in constant time. For what concerns the communication cost of the protocol, in addition to the messages required by the *SP-Filtering* protocol, *Hide&Seek* requires each user to exchange two messages of constant size with another user each time the two-parties computation is run. In Section 4 we evaluate the number of these messages in our experimental setting.

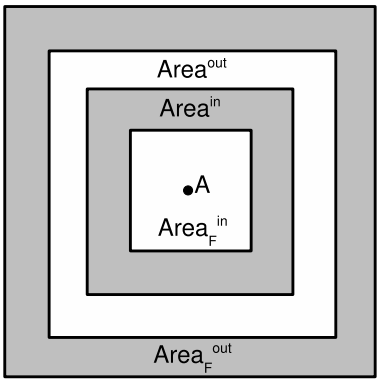


Figure 2. Possible locations of B

3.3. The *Hide&Crypt* protocol

The solution proposed in Section 3.2 to protect location privacy requires that exactly the maximum tolerable amount of location information is revealed each time a users initiates the two-parties computation. That is, the protocol does guarantee the minimum privacy requirements, but it does not do more in terms of privacy protection. As mentioned in the introduction, in addition to the minimum privacy requirements, a user would prefer to reveal as little information as possible about their location. In order to address this problem, we now present the *Hide&Crypt* protocol.

Similarly to the *Hide&Seek*, the *Hide&Crypt* is composed of two sub-protocols: the *SP-Filtering* protocol and a two-parties protocol between two users. The difference with respect to the *Hide&Seek* protocol is that the two-parties proximity problem is solved through a secure computation protocol. The main idea is that, when the *SP-Filtering* protocol cannot determine the proximity of B for A, A will compute the set S of granules of G_B^U such that, if B is contained in any of these granules, then B is possibly in the proximity of A. To test if B is indeed located in any granule of S, A can run the set-inclusion secure-computation protocol with B and hence to conclude if B is possibly in proximity (a solution to the secure two-parties set-inclusion problem was proposed in [6]). A technical issue of this protocol is that if B knows the the cardinality of S he can be able to infer some location information about A. For this reason, the protocol we propose is an extension of the secure two-parties set-inclusion problem in which the cardinality of the set S is kept secret too.

More precisely, *Hide&Crypt* works as follows. First, A computes the set S' of indexes of granules of G_B^U that intersects with the circle $\mathcal{C}$ centered in the location of A with

radius δ_A. Then, in order to hide to B the cardinality of this set, A creates a new set S by adding to S' some negative numbers. The aim of negative numbers is to increase the cardinality of S without affecting the result of the computation. The cardinality of S should be increased so that it is as large as the number S_{MAX} that represents the maximum number of granules of G_B^U that intersect with any circle with radius δ_A. Note that S_{MAX} can be computed off-line since its values depends only on G_B^U and δ_A. Then, A encrypts all the elements of S with a encryption function[5] E and a private key K_A and sends the result to B. User B encrypts again, using his private key K_B, each element in the set he receives and sends it back to A together with the encryption of the index j such that B is located in $G_B^U(j)$. Finally, A encrypts again $E_{K_B}(i)$ using the key K_A and checks if the result is contained in $E_{K_B}(E_{K_A}(S))$ [6]. Encryption function E is such that $E_{K_A}(E_{K_B}(i)) \in E_{K_B}(E_{K_A}(S))$ if and only if $j \in S$. Since negative numbers are not valid indexes, $j \geq 0$, and hence $j \in S$ if and only if $j \in S'$. Therefore A computes whether B is in her proximity or not.

Protocol *Hide&Crypt*

Prerequisites: A and B are running the *SP-Filtering* protocol. User A knows G_B^U, a private key K_A, the circle $\mathcal{C}$ centered in A's location with radius δ_A, and the value S_{MAX}. B knows a private key K_B, and the granule $G_B^U(j)$ where B is located.

Protocol:

1: A receives "B is possibly in proximity" from the SP.
2: A computes: $S' = \{j \in \mathbb{N} \text{ s.t. } G_B^U(j) \cap \mathcal{C} \neq \emptyset\}$
3: A computes: S'' as a set of $S_{MAX} - |S'|$ random negative numbers.
4: A computes: $S = S' \cup S''$
5: A sends "starting two-parties protocol $E_{K_A}(S)$" to B
6: B sends $\langle E_{K_B}(E_{K_A}(S)), E_{K_B}(j)\rangle$ to A
7: A computes: $E_{K_A}(E_{K_B}(j))$
8: **if** $(E_{K_A}(E_{K_B}(j)) \in E_{K_B}(E_{K_A}(S)))$ **then**
9: A computes that B is in proximity
10: **else**
11: A computes that B is not in proximity
12: **end if**

Similarly to the *Hide&Seek* protocol, also in *Hide&Crypt* the SP acquires not information about the location of A and B when the two-parties part is run. Proposition 3 formally states the information that A acquires about B and that B acquires about A when the *Hide&Seek* protocol is used to compute the proximity of B with respect to A.

5. Our results hold for any commutative encryption function such that, given two keys K_A and K_B and two values i and j, $E_{K_A}(E_{K_B}(i)) = E_{K_B}(E_{K_A}(j))$ if and only if $i = j$.

6. We denote with $E_K(S)$ the encryption of each element of the set S. Formally, $E_K(S) = \bigcup_{i \in S} E_K(i)$. Note that $E_K(S)$ is a set and, hence, its elements are not ordered.

Proposition 3: Let B be located in $G_B^{SP}(j)$, $\mathcal{C}$ be the circle centered in the location of A with radius δ_A and

$$S' = \{j' \in \mathbb{N} | G_B^U(j') \cap \mathcal{C} \neq \emptyset\}$$

$$S'_{sp} = \{i \in \mathbb{N} | MinDist(L_A(i), L_B(j)) <$$

$$< \delta_A < MaxDist(L_A(i), L_B(j))\}$$

Whenever the *Hide&Seek* protocol is used to compute the proximity of B with respect to A:

1) if A can compute that B is in proximity as the result of the secure computation protocol with B, then A cannot exclude that B is located in any location of $Area_C^{in} = Area_{NoFilter} \cap (\bigcup_{j' \in S'} G_B^U(j'))$;

2) if A can compute that B is not in proximity as the result of the secure computation protocol with B, then A cannot exclude that B is located in any location of $Area_C^{out} = Area_{NoFilter} \cap (Area_C^{in})^C$;

3) B cannot exclude that A is located in any location of $Area_C^{passive} = \bigcup_{i \in S'_{sp}} L_A(i)$.

The idea of the privacy protection guaranteed by *Hide&Crypt* is similar to the one of *Hide&Seek* and is graphically depicted in Figure 2. In this case, $Area^{in}$ and $Area^{out}$ correspond to $Area_C^{in}$ and $Area_C^{out}$, respectively. There are two main differences with respect to *Hide&Seek*. First, as we motivate and experimentally observe in Section 4, $Area_C^{in}$ is generally smaller than $Area_S^{in}$ and hence $Area_C^{out}$ is generally larger than $Area_S^{out}$ (we recall that $Area_S^{in} \cup Area_S^{out} = Area_{NoFilter}$ and $Area_C^{in} \cup Area_C^{out} = Area_{NoFilter}$). Second, while $Area_S^{passive}$ is the exact granule of G_A^U where A is located, $Area_C^{passive}$ is a coarser area that generally covers several granules of G_A^U.

In Section 4 we show, through our experimental results, that *Hide&Crypt* provides on average more privacy protection than *Hide&Seek*. With Theorem 3, we formally guarantee that *Hide&Crypt* provides the minimal location-privacy requirements.

Theorem 3: Let C be a user participating in the *Hide&Crypt* protocol. The minimum privacy requirements of C are guaranteed.

The computational complexity of the *Hide&Crypt* protocol is the same as the *SP-Filtering* protocol on the SP. On the client, the computational complexity is the same as in the *SP-Filtering* protocol plus the time required to encrypt the S_{MAX} integers. Assuming that the computation of the *SP-Filtering* can be performed in constant time, like in our experiments, and assuming a fixed size of the encryption key, the computational complexity is linear in the size of the data to encrypt i.e, is linear in the size of S_{MAX}. For what concerns the communication cost, *Hide&Crypt* requires the exchange the same number of messages as the *Hide&Seek*

protocol with the difference that each message has a length linear in S_{MAX}.

4. Experimental evaluation

The experimental evaluation of the proposed protocols was performed on a dataset of user movements obtained through the *MilanoByNight* simulation[7]. The dataset contains movements of $100,000$ users on the road network of Milan during a typical weekend night. The simulator is tuned to represent movements of potential users of our considered proximity service, which we believe are young people moving to entertainment places during the night. All the test results shown in this section are obtained as average values computed over $1,000$ users, each of them using the service during the 4 hours of the simulation. Locations are sampled every 2 minutes. The total size of the map is 324 km^2 and the average density is 308 users/km^2. We implemented the protocols using Java, and we performed our tests on a 64-bit Windows Server 2003 machine with $2,4$Ghz Intel Core 2 Quad processor and 4GB of shared RAM.

In our experiments we considered eleven levels of granularities with granules of level 0 (bottom) granularity forming a grid of 1024×1024 squares (each one with an edge of about 17 meters), covering the entire map. Level l granularity is obtained by grouping 2^l granules of the bottom granularity on each dimension. For example, granules of level 2 granularity are obtained by grouping 4×4 granules of the bottom granularity. Level 10 granularity contains only one granule covering the entire map. It can be easily seen that each pair of granularities satisfies Condition (2) in Section 3.1, and that the granule conversion operations required by our protocols can be performed in constant time.

Table 1. Parameter values

Parameter	Values
δ	125m, 250m, **500m**, 1000m
Level of G^{SP} granularity	0, 1, 2, 3, 4, 5, 6, **7**, 8, 9, 10
Level of G^U granularity	0, 1, 2, **3**, 4, 5, 6, 7, 8, 9, 10
Average number of buddies in a group	100, 200, **400**, 800, 1600

For the sake of simplicity, in our tests we assume that all the users share the same parameters. In particular, in each test we fix a single value of δ (the proximity threshold), G^U and G^{SP} for all the users. Table 1 shows the most relevant parameters of our experimental evaluation. Default values are denoted in bold. In this section, in order to evaluate the *SP-Filtering* protocol, as a stand-alone protocol, we modified it so that it always return a "B is in proximity message" also in the case when the SP can only conclude that B is possibly in proximity.

7. http://everywarelab.dico.unimi.it/lbs-datasim

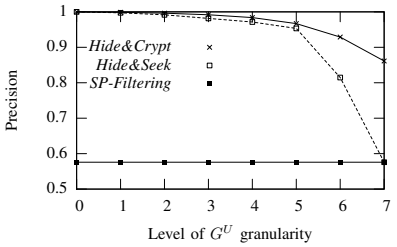
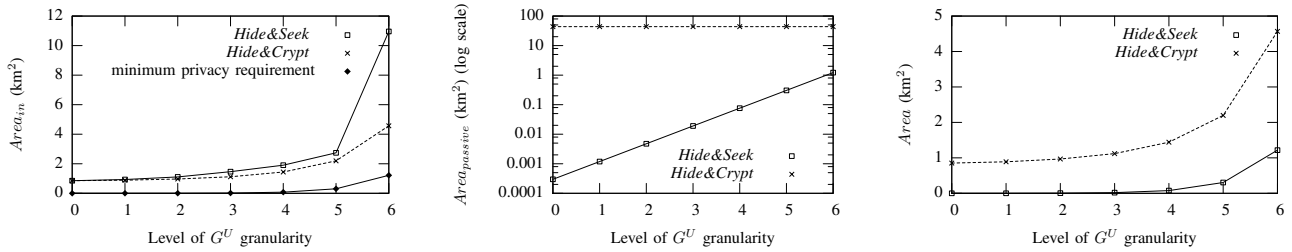


Figure 3. Percentage of correct answers

The first set of experiments is aimed at evaluating the *precision* of the proposed solutions i.e., the percentage of correct answers. Figure 3 shows the precision of the three protocols for different levels of the G^U parameter. It can be observed that for small values of G^U, *Hide&Seek* and *Hide&Crypt* have a precision value close to 1. This means that it is rare to obtain a false-positive result (we recall that no false-negative results are returned by the proposed protocols). However, for larger values of G^U the precision decreases due to the fact that, with both protocols, coarser granularities add uncertainty in the proximity computation. Figure 3 also shows that *Hide&Crypt* is more precise than *Hide&Seek*, especially when G^U defines granules whose edge is greater than δ. This is due to the fact that using *Hide&Crypt* the distance is computed between the actual location of the user A and the granules of G^U while using *Hide&Seek* the distance is computed between two granules of G^U_A and G^U_B hence adding more approximation to the computation. Note that the precision of *SP-Filtering* remains constant as $L_A(i)$ and $L_B(j)$ are not affected by the G^U level when it is below the default granularity level 7 for G^{SP}; the precision decreases for increasing levels of G^U when the G^U level exceeds that of G^{SP} (not shown in figure).

The aim of the second set of experiments is to evaluate the actual location privacy provided by the protocols. Our experimental analysis is focused on the uncertainty area obtained by buddy B about the position of A after running the *Hide&Seek* or *Hide&Crypt* protocols. As explained in Section 3.2, when two buddies engage in a two-parties protocol each user can deduce some informations about the position of the other user. In Figure 4(a) we first consider the case in which A receives the message *"B is in proximity"* from B. We can observe that, for small levels of G^U, the area of the region associated with user B is about 1 km^2 with both protocols, while the area of a granule of G^U (the minimum privacy requirement) is about 0.02 km^2 (for level 3 of G^U). Higher levels of G^U result in larger areas, which means that, as expected, with a coarser granularity there is more uncertainty about B's position. We also notice that,



(a) Uncertainty of A about B's location when A receives *"B is in proximity"* from B

(b) Uncertainty of B about A's location when A initiates the two-parties computation

(c) Uncertainty of a buddy about the location of C in the worst case

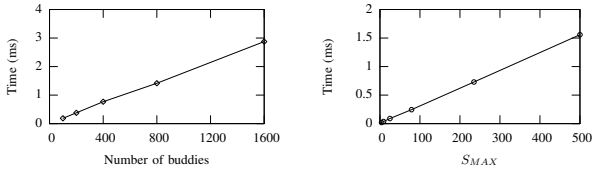
Figure 4. Location privacy.

in this case, regions deduced when using *Hide&Seek* are larger than the ones obtained with *Hide&Crypt*. The reason is analogous to the one already explained for Figure 3, as the granules which form these regions are chosen by computing their distance with the actual location of A when using *Hide&Crypt*, and the granule G_A^U when using *Hide&Seek*. In Figure 4(b) we consider the case in which A is checking the proximity of B, and we assume the two-parties protocol is started (note that a logarithmic scale is used for the area). In this case, the area determined by the *Hide&Crypt* protocol is very large, while using *Hide&Seek* it consists only of the granule G^U containing A, which is the minimum privacy requirement. This is due to the fact that user A sends to B this granule when starting the *Hide&Seek* two-parties protocol, while no explicit location information is sent to B when using *Hide&Crypt*. With this protocol, assuming that B knows the proximity threshold δ_A (this is not required by the protocol, but can be inferred from the value of S_{MAX} which B can compute), we observe that the area of the region associated to A using *Hide&Crypt* is always around 45 km^2 when G^U is finer than G^{SP}.

Figure 4(c) shows the location privacy obtained using both protocols in their respective worst-cases. Here, worst case means the location privacy afforded to a user when she both initiates the two-parties computation and replies to requests from buddies that initiates the protocol. That is, the curve is actually the smaller value from Figures 4(a) and 4(b) for each G^U granularity level. For *Hide&Seek*, the worst-case is equivalent to the minimum requirement of privacy expressed by G^U. Using *Hide&Crypt*, instead, we can observe that in the worst case the location privacy achieved is much higher than the minimum requirement. Even with a minimal requirement of privacy G^U (the area of a granule of G^U is about 300 m^2 when the level of G^U is 0), the uncertainty area determined by the *Hide&Crypt* protocol is at least 0.8 km^2.

In the third set of experiments we computed several measures regarding performance. Figure 5 shows the computation time both on the server and the client side. Since the two protocols share the *SP-Filtering* protocol, and this is the only server side computation, they have the same server side computational cost. Figure 5(a) shows that, while the server side computation time at each location update grows linearly with average number of buddies, the computation is very efficient also in the case in which a user has a relatively high number of buddies.

For what concerns the client side computation, as we observed in Section 3, the computational cost of *Hide&Seek* is negligible, since a single constant time distance computation is performed at each invocation. Therefore, we are only interested in evaluating the computational overheads of *Hide&Crypt*. In its implementation we used the RC4 algorithm, which meets our encryption requirement. Figure 5(b) shows that the client side computation time, for each invocation of *Hide&Crypt*, is linear in the S_{MAX} value. As defined in Section 3.3, S_{MAX} is the number of indexes that are encrypted and depends on the values δ and G^U; the larger is δ (or the finer is G^U), the larger is S_{MAX}. With our default parameters S_{MAX} is 25. The computation time is always in the order of a few milliseconds (less than 2ms) when simulating the protocol on a desktop computer like the one we used in our tests. We are confident that the cost would remain sustainable on high-end mobile devices, and this will be hopefully confirmed by the actual client implementations that are underway.



(a) Server side

(b) Client side *Hide&Crypt*

Figure 5. Computation time

Finally, Figure 6 shows the average number of two-parties computations executed by a user in a minute, with different levels of the G^{SP} granularity. This experiment gives an indication on the effect of the *SP-Filtering* protocol and on the communication cost of *Hide&Seek* and *Hide&Crypt* in terms of exchanged messages. As we can observe, the finer

G^{SP} granularity, the fewer two-parties computations; This is due to the fact that if G^{SP} is fine, the *SP-Filtering* protocol is more effective in the sense that it rarely returns the "B is possibly in proximity" message and therefore the two-parties computation is not required. For coarser G^{SP} levels, about 100 two-parties computations per minute are required (more than 1.5 per second), hence negatively affecting the system performance (e.g., the battery duration) and the communication cost. Note that with level of G^{SP} equals to 9 the entire map is divided in 4 granules only, hence $d = 0$. Consequently the *SP-Filtering* is not effective. For finer G^{SP} levels, the number of two-parties computations rapidly decreases. For example, if a user can tolerate the SP to know her location at a granularity of a few thousand square meters (e.g., 20,000 m^2 for level 3 of G^{SP}), then less than 2 two-parties computation per minute are required.

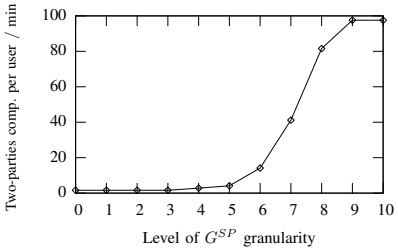


Figure 6. Communication cost

5. Conclusions and future work

In this paper we presented three privacy preserving protocols for proximity services, we proved their correctness with respect to privacy preferences and we showed the results of an extensive experimental evaluation. The *Hide&Crypt* algorithm, in particular, was shown to provide a high service precision, sustainable computational cost even for a large number of buddies, and to ensure in most cases much more privacy than the specified minimal requirements.

A relevant issue that could not be addressed in this paper for space limitations is the adoption of a safe and efficient strategy for the buddies to issue their location updates. Indeed, updates should not be issued exactly when crossing spatial granule borders, since this information may be used by an adversary to exclude the presence of the buddy in certain other parts of a granule, possibly violating the privacy preferences. In the full version of this paper we show how this *timing attack* can be easily avoided. Several other issues deserve further investigation, including the conditions under which a three-party secure computation involving the SP may be effective.

Acknowledgments

This work was partially supported by National Science Foundation under grants CT-0716567, CT-0716575, CT-0627493, IIS-0430165 and IIS-0430402, by Air Force Office of Scientific Research under grant FA9550-07-1-0527 and by Italian MIUR under grants PRIN-2007F9437X and InterLink II04C0EC1D.

References

[1] Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. *Time Granularities in Databases, Temporal Reasoning, and Data Mining.* Springer, 2000.

[2] Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2008.

[3] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *Proc. of SIGMOD*. ACM Press, 2008.

[4] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1719–1733, 2007.

[5] Ali Khoshgozaran, Houtan Shirani-Mehr, and Cyrus Shahabi. Spiral: a scalable private information retrieval approach to location privacy. In *Proc. of the 2th International Workshop on Privacy-Aware Location-based Mobile Services*. IEEE Computer Society, 2008.

[6] Shundong Li, Yiqi Dai, Daoshun Wang, and Ping Luo. Symmetric encryption solutions to millionaire's problem and its extension. In *Proc. of 1st International Conference on Digital Information Management*. IEEE Computer Society, 2006.

[7] Sergio Mascetti, Claudio Bettini, Dario Freni, and X. Sean Wang. Spatial generalization algorithms for LBS privacy preservation. *Journal of Location Based Services*, 2(1), 2008.

[8] Man Lung Yiu, Christian S. Jensen, Xuegang Huang, and Hua Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *Proc. of the 24th International Conference on Data Engineering*. IEEE Computer Society, 2008.

[9] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, Lester and Pierre: Three protocols for location privacy. In *Proc. of the 7th Privacy Enhancing Technologies Symposium*. IEEE Computer Society, 2007.