Università degli Studi di Milano

Dottorato di Ricerca in Informatica

XXIII Ciclo

Settore Scientifico Disciplinare INF/01 Informatica

Privacy Preservation in Location-Based Proximity Services

Tesi di Dottorato di Ricerca di:
Dario Freni

Relatore:
Prof. Claudio Bettini

Coordinatore del Dottorato:
Prof. Ernesto Damiani

Anno Accademico 2009/2010

Contents

1 Introduction 1

1.1 Problem description 1

1.2 Contribution 2

1.3 Outline 3

2 Privacy models for Location Based Services: an overview 5

2.1 A classification of attacks to LBS privacy 5

2.1.1 Attacks exploiting *quasi-identifiers* 7

2.1.2 Snapshot versus historical attacks 8

2.1.3 Single versus multiple-issuer attacks 9

2.1.4 Attacks exploiting knowledge of the defense 10

2.2 Defenses to LBS privacy threats 10

2.2.1 Anonymity based defenses 12

2.2.2 Defenses based on private information obfuscation . . 18

2.2.3 Encryption based defenses 22

2.3 Impact of realistic simulations on the evaluation of defense techniques 23

2.3.1 The *MilanoByNight* simulation 24

2.3.2 Experimental settings 28

2.3.3 Impact of the adversary model on the evaluation of the generalization algorithms 30

2.3.4 Impact of the deployment model on the evaluation of the generalization algorithms 31

3 Privacy preservation in Proximity Services 35

3.1 Related work 36

3.2 Problem definition 39

3.2.1 The proximity service 39

3.2.2 Privacy concerns and privacy requirements 41

3.2.3 Adversary model and privacy preservation 43

3.3 Privacy preserving techniques 45

3.3.1 *SP-Filtering*, *Hide&Seek* and *Hide&Crypt* 45

3.3.2 Analysis of *SP-Filtering*, *Hide&Seek* and *Hide&Crypt* protocols 50

3.3.3 *Longitude* 57

3.3.4 Analysis of the *Longitude* protocol 62

3.3.5 *C-Hide&Seek* and *C-Hide&Hash* 66

3.3.6 Analysis of *C-Hide&Seek* and *C-Hide&Hash* protocols 77

3.4 Comparison of the protocols 90

4 Evaluation of the proposed defenses 93

4.1 Empirical evaluation 93

4.1.1 The experimental setting 93

4.1.2 Evaluation of the quality of service 95

4.1.3 Evaluation of the system costs 100

4.1.4 Evaluation of the achieved privacy 105

4.2 Implementation of the service 106

4.2.1 Mobile client 109

4.2.2 Web client 110

4.2.3 Monitor application 111

5 Conclusions and future work 113

5.1 Summary of the contributions 113

5.2 Future work 114

A Proofs 117

A.1 Proof of Proposition 1 117

A.2 Proof of Theorem 1 119

A.3 Proof of Proposition 2 119

A.4 Proof of Theorem 2 121

A.5 Proof of Proposition 3 122

A.6 Proof of Theorem 3 125

A.7 Proof of Lemma 1 126

A.8 Proof of Lemma 2 126

A.9 Proof of Theorem 4 126

A.10 Proof of Theorem 5 128

A.11 Proof of Theorem 6 130

A.12 Proof of Theorem 7 130

B Notation 132

List of Figures

2.1 General privacy threat in LBS 6
2.2 Average anonymity. 30
2.3 Performance evaluation for different values of the total population. 32
2.4 Evaluation of the *Greedy* algorithm using AB and MRM data sets. $P_{id-in} = P_{id-out} = 0.1$ 33
2.5 Average privacy using AB and MRM data sets. $P_{id-out} = P_{id-in}/10$. 34

3.1 Regions L_A and L_B 47
3.2 Possible locations of B 54
3.3 Examples of modular translations of a point and of a cell. $E_{k_i}(c)$ represented in gray. 61
3.4 Examples of modular distance 62
3.5 Example of the certainty region CR_A 62
3.6 Two forms of approximation introduced by the *Longitude* Protocol. 66
3.7 Location update policy and generation of single-use keys. . . 68
3.8 Location update sub-protocol in *C-Hide&Seek*. 69
3.9 Proximity request sub-protocol in *C-Hide&Seek*. 70
3.10 Computation of granules of G_B^U considered in proximity by A 71
3.11 Proximity request sub-protocol in *C-Hide&Hash*. 75
3.12 Different cases of proximity between a point and a region . . 83

3.13 Approximation incurring with the *minimum-distance* semantics 85
3.14 Expected precision 85
3.15 Expected recall 86
3.16 Privacy with respect to a buddy 88

4.1 Examples of the *granularity approximation* 95
4.2 Evaluation of the impact of the approximations 97
4.3 Evaluation of the quality of service (considering both approximations) 98
4.4 Evaluation of the system costs 101
4.5 Size of the uncertainty region. 106
4.6 System architecture 107
4.7 Android application 109
4.8 Web application interface ("Map" tab) 110
4.9 Monitor application interface 111

Chapter 1

Introduction

1.1 Problem description

The increased availability of GPS-enabled mobile devices that provide access to Internet services has caused an incredible proliferation of new mobile services that offer personalized information to users depending on their location, called location-based services (LBS). In parallel, social network (SN) services have become very popular among Internet users. As SN services are almost entirely based on user-generated content, SN companies have focused on providing easy access to their services, as well as new ways for the users to generate information. We called geo-social networks (GeoSNs) those SN in which the geographical positions of participants and of relevant resources are used to enable new information services.

In most SNs, each user has a contact list of *friends*, also called *buddies*. A basic service in GeoSNs is the *proximity service* that alerts the user when any of her buddies is in the vicinity, possibly enacting other activities like visualizing the buddy's position on a map, or activating a communication session with the buddy. Such proximity services are already available as part of GeoSNs (e.g., *Brightkite*[1]), as part of a suite of map and navigation services (e.g., *Google Latitude*[2]), or as an independent service that can be

[1] http://brightkite.com

[2] http://www.google.com/latitude

integrated with social networks (e.g., *Loopt*[3]).

From a data management point of view, a proximity service involves the computation of a range query over a set of moving entities issued by a moving user, where the range is a distance threshold value decided by the user. All existing services are based on a centralized architecture in which location updates, issued from mobile devices, are acquired by the SP, and proximity is computed based on the acquired locations.

While proximity services are very attractive for many SN users, they also raise severe privacy concerns: a) the users may not fully trust the service provider that will handle their location data, b) the users would like to have better control on the precision of location data released to their buddies. For the purpose of alleviating these concerns, we address the problem of protecting users' location privacy in the context of proximity services, considering different possible *adversaries*, including the SP and, to a different extent, the buddies.

Existing proximity services do not offer any protection regarding point a) above other than legal privacy policy statements, and they offer a very limited control regarding point b); for example, some solutions allow the user to limit the location released to the buddies to the precision level of city.

1.2 Contribution

The main goal of this dissertation is the investigation of privacy issues in location-based proximity services and the proposal of effective defense techniques.

The solutions proposed in the research area of LBS privacy preservation (see Chapter 2 for a detailed survey) are not effective or directly applicable to proximity services. In this thesis we present a rigorous study of location privacy in proximity services, considering two categories of adversary: the

[3]http://www.loopt.com

service provider (SP) and the other buddies. We propose a flexible way to let users express privacy controls with respect to each category of adversaries. Indeed, each user is allowed to specify his privacy requirements in terms of regions of the geographical space: given a specific geographic position of the user, the region including the position defines the highest location precision exposed to the considered adversary.

We propose five protocols for enforcing users' privacy requirements. All the protocols have been formally studied and validated against a well defined adversary model. Each protocol differs from the others for its requirements, performance or privacy guarantees. A theoretical analysis of each protocol formally supports the privacy guarantees and evaluates the expected performance in terms of quality of service, privacy and costs. In addition, the same performance aspects are measured in an extensive experimental work. The practicality of our approach is illustrated by a complete implementation of the techniques in a fully-functional system, including client applications for web browsers and mobile phones.

Results presented in this dissertation are the outcome of research performed at the EveryWare Lab., at the D.I.Co. department of the Università degli Studi di Milano, and at the Center for Secure Information Systems at George Mason University. Contributions included in following chapters have been already partially published; in particular, the classification and contributions presented in Chapter 2 derive from investigations included in [5], and contributions presented in Chapter 3 are included in [16, 34, 36, 39].

1.3 Outline

The dissertation is structured as follows. In Chapter 2 we present the problem of privacy threat in LBS and a categorization of the attacks to the privacy and of the defense techniques that have been proposed in literature. In the same chapter, we present a study of the impact of the user movement dataset used during the experimental analysis on the evaluation of defense techniques, and we introduce a realistic simulation of user movement called

MilanoByNight. Chapter 3 provides a deep analysis of the privacy threats enabled by the use of proximity services, including a discussion of the existing techniques, a formalization of the privacy problem and the adversary model. In the same chapter we propose five privacy preserving protocols, and for each protocol we present a rigorous formal analysis of their properties. Chapter 4 presents an extensive experimental evaluation, and the implementation of these protocols in a fully-functional instant messaging system. In Chapter 5 we conclude the dissertation discussing the open problems and summarizing the contributions of this work.

Chapter 2

Privacy models for Location Based Services: an overview

This chapter presents an overview of the related work in the field of privacy in LBS. In Section 2.1 we introduce a categorization to the attacks to users' privacy in LBS, while in Section 2.2 we classify the main proposed defense techniques according to the threats they have been designed for, and according to other general features. Section 2.3 presents an analysis of the impact of the user movement dataset on the experimental evaluation of privacy preserving techniques.

2.1 A classification of attacks to LBS privacy

There is a privacy threat whenever an adversary is able to associate the identity of a user to information that the user considers private. In the case of LBS, this *sensitive association* can be possibly derived from location-based requests issued to service providers. More precisely, the identity and the private information of a single user can be derived from requests issued by a group of users as well as from available background knowledge. Figure 2.1 shows a graphical representation of this general privacy threat in LBS.

A *privacy attack* is a specific method used by an adversary to obtain the

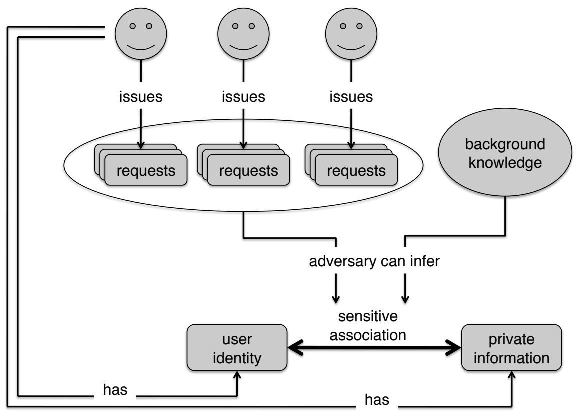


Figure 2.1: General privacy threat in LBS

sensitive association. Privacy attacks can be divided into categories mainly depending on several parameters that characterize the *adversary model*. An adversary model has three main components: a) the target private information, b) the ability to obtain the messages exchanged during service, and c) the *background knowledge* and the *inferencing abilities* available to the adversary. The target private information is the type of information that the adversary would like to associate with a specific individual, like e.g., her political orientation, or, more specifically, her location. Different classes of adversaries may also have different abilities to obtain the messages exchanged with the service provider, either by eavesdropping the communication channels or by accessing stored data at the endpoints of the communication. This determines, for example, the availability to the adversary of a single message or multiple messages, messages from a specific user or from multiple users, etc.. Finally, the adversary may have access to external knowledge, like e.g., phone directories, lists of members of certain groups, voters lists, and even presence information for certain locations, and may be able to perform inferences, like joining information from messages with external information as well as more involved reasoning. For example, even when a request does not explicitly contain the sensitive association (e.g., by using pseudo-identifiers to avoid identification of the issuer), the adversary may re-identify the issuer by joining location data in the request with presence data from external sources. Regarding background knowledge, two extreme cases can be considered. When no background knowledge is available, a privacy threat exists if the sensitive association can be obtained only from the messages in the service protocol. When "complete" background knowledge is available, the sensitive association is included and the privacy violation occurs independently from the service request.

Hence, privacy attacks should not only be categorized in terms of the target private information, and of the availability to the adversary of service protocol messages (the first two of the main components mentioned above), but also in terms of the available background knowledge and inferencing abilities. In the following, we list some categories of privacy attacks specifically enabled by background knowledge.

- Attacks exploiting *quasi-identifiers* in requests;
- Snapshot versus historical attacks;
- Single- versus multiple-issuer attacks;
- Attacks exploiting knowledge of the defense;

Each category is discussed in the rest of this section.

2.1.1 Attacks exploiting *quasi-identifiers*

Either part of the sensitive association can be discovered by joining information in a request with external information. When we discover the identity of the issuer (or even restrict the set of candidate issuers) we call the part of the request used in the join *quasi-identifier*. For example, when the location data in the request can be joined with publicly available presence data to identify an individual, we say that location data act as quasi-identifier. Similarly to privacy preserving database publication, the recognition of what can act as quasi-identifier in service request is essential to identify the possible attacks (as well as to design appropriate defenses).

2.1.2 Snapshot versus historical attacks

Many of the first approaches presented in the literature [6, 23, 27, 40] have proposed techniques to ensure a user's privacy in the case in which the adversary can acquire a single request issued by that user. More specifically, these approaches do not consider attacks based on the correlation of requests made at different time instants. An example are attacks exploiting the ability of the adversary to *link* a set of requests, i.e., to understand that the requests have been issued by the same (anonymous) user.

When historical correlation is ignored, we say that the corresponding threats are limited to the *snapshot case*. Intuitively, it is like the adversary can only obtain a snapshot of the messages being exchanged for the service at a given instant, while not having access to the complete history of messages.

In contrast with the snapshot case, in the *historical case* it is assumed that the adversary is able to *link* a set of requests. Researchers [3, 25] have considered such a possibility. Several techniques exist to *link* different requests to the same user, with the most trivial ones being the observation of the same identity or pseudo-identifier in the requests, and others being based on spatiotemporal correlations. We call *request trace* a set of requests that the adversary can correctly associate to a single user. We use Example 1 to show that defense techniques for the snapshot cases cannot straightforwardly be used in the historical case.

Example 1. *Suppose Alice requires* 3*-anonymity and issues a request* r*. An algorithm safe against attacks exploiting knowledge of the defense is used to generalize* r *into a request* r' *whose spatiotemporal region includes only Alice, Bob, and Carl. Afterwards, Alice issues a new request* r_1 *that is generalized into a request* r'_1 *whose spatiotemporal region includes only Alice, Ann, and John. Suppose the adversary is able to link requests* r' *and* r'_1*, i.e., he is able to understand that the two requests have been issued by the same user. The adversary can observe that neither Bob nor Carl can be the issuer of* r'_1*, because they are not in the spatiotemporal region of* r'_1*; Consequently, they cannot be the issuers of* r' *either. Analogously, considering the spatiotem-*

poral region in r', he can derive that Ann and John cannot be the issuers of the two request. Therefore, the adversary can identify Alice as the issuer of r' and r'_1.

In this example, in addition to adversary's ability of using location as quasi-identifier, the ability to link requests is crucial for the attack to be successful.

2.1.3 Single versus multiple-issuer attacks

When the adversary model limits the requests that can be obtained to those being issued by a single (anonymous) user, we say that all the attacks are *single-issuer attacks.* When the adversary model admits the possibility that multiple requests from multiple users are acquired, and the adversary is able to understand if two requests are issued by different users, we have a new important category of attacks, called *multiple-issuer attacks.* Note that this is an orthogonal classification with respect to snapshot and historical. Example 2 shows that, in the multiple-issuer case, an adversary can infer the sensitive association for a user even if the identity of that user is not revealed to the adversary.

Example 2. *Suppose Alice issues a request r and that the adversary can only understand that the issuer is one of the users in a set S of potential issuers. However, if all of the users in S issue requests from which the adversary can infer the same private information inferred from r, then the adversary can associate that private information to Alice as well.*

In the area of privacy in databases, this kind of attack is known as *homogeneity attack* [32]. In LBS, differently from the general case (depicted in Figure 2.1), in the snapshot, multiple-issuer case, a single request for each user in a group is considered. More involved and dangerous threats can occur in the historical, multiple-issuer case.

2.1.4 Attacks exploiting knowledge of the defense

In the security research area, it is frequently assumed that the adversary knows the algorithms used for protecting information, and indeed the algorithms are often released to the public. We have shown [35] that the first proposals for LBS privacy protection ignored this aspect leading to solutions subject to so called *inversion* attacks. As an example of these attacks, consider a defense technique that produces a LBS request with a generalized spatial region instead of the exact location (i.e., a *spatial cloaking* technique), and suppose that this request is observed by the adversary. Suppose also that he gets to know the identity of the four potential issuers of that request, since he knows who was in that region at the time of the request; Still he cannot identify who, among the four, is the actual issuer, since cloaking has been applied to ensure 4-anonymity. However, if he knows the cloaking algorithm, he can simulate its application to the specific location of each of the candidates, and exclude any candidate for which the resulting cloaked region is different from the one in the observed request. Some of the proposed algorithms are indeed subject to this attack. Kalnis et al. [27] show that each generalization function satisfying a property called *reciprocity* is not subject to the inversion attack. Recently, Deutsch et al. [14] observed that even if the reciprocity property is satisfied, if the generalization function produces generalized regions of different size or position for different users inside the same anonymity set, an adversary knowing the generalization could still break anonymity. In our chapter, depending on the assumption in the adversary model about the knowledge of the defense algorithm we distinguish *def-aware attacks* from *def-unaware attacks.*

2.2 Defenses to LBS privacy threats

Defense techniques can be categorized referring to the attacks' classification reported above, depending on which specific attacks they have been designed for. However, there are other important criteria to distinguish defense approaches:

1. Defense technique: Identity anonymity versus private information obfuscation versus encryption

2. Defense architecture: Centralized versus decentralized

3. Defense validation: Theoretical versus experimental.

The different defense techniques can be classified as *anonymity-based* if they aim at protecting the association between an individual and her private information by avoiding the re-identification of the individual through a request (or a sequence of requests). This is achieved by transforming the parts of the *original request* acting as quasi-identifiers to obtain a *generalized request*. On the contrary, techniques based on *private information obfuscation* aim to protect the same association by transforming the private information contained in the original request, often assuming that the identity of the individual can be obtained. Finally, *encryption-based* techniques use private information retrieval (PIR) methods that can potentially protect both the identity of the issuer and the private information in the request.

Centralized defense architectures assume the existence of one or more trusted entities acting as a proxy for service requests and responses between the users and the service providers. The main role of the proxy is to transform requests and possibly responses according to different techniques in order to preserve the privacy of the issuers. Decentralized architectures, on the contrary do not assume intermediate entities between users and service providers. Among the benefits of centralized architectures are a) the ability of the proxy to use information about a group of users (e.g., their location) in order to more effectively preserve their privacy, and b) the availability of more computational and communication resources than the users' devices. The main drawbacks are considered the overheads in updating on the proxy the information about the users, and the need for the user to trust these entities.

A third criterion to distinguish the defenses that have been proposed is the validation method that has been used. In some cases, formal results, based on some assumptions, have been provided so that a certain privacy is guaranteed in all scenarios in which the assumptions hold. In other cases, only an experimental evaluation, usually based on synthetic data, is provided. In Section 2.3 we discuss how this approach may be critical if the actual service deployment environment does not match the one used in the evaluation.

In this section we classify the main proposals appeared in the literature according to this categorization.

2.2.1 Anonymity based defenses

Most of the techniques proposed in the LBS literature to defend privacy through anonymity consider the location as a quasi-identifier. Indeed, it is implicitly or explicitly assumed that background knowledge can in some cases lead an adversary to infer the identity of the issuer given her location at a given time. Consequently, the target private information for the considered attacks is usually the specific service being requested, or the location of the issuer whenever that location cannot be used as quasi-identifier.[1]

When the location acts as a quasi-identifier, the defense technique transforms the location information in the original request into a *generalized location*. In the following we call *anonymity set* of a generalized request, the set of users that, considering location information as quasi-identifier, are not distinguishable from the issuer.

Centralized defenses against snapshot, single-issuer and def-unaware attacks.

Anonymity based defenses with centralized architectures assume the existence of a trusted proxy that is aware of the movements of a large number

[1]Indeed, location cannot be the target private information when it can be found explicitly associated with identities in background knowledge.

of users. We call this proxy Location-aware Trusted Server (LTS).

The first generalization algorithm that appeared in the literature is named *IntervalCloaking* [23]. The paper proposes to generalize the requests along the spatial and/or temporal dimension. For what concerns the spatial dimension, the idea of the algorithm is to iteratively divide the total region monitored by the LTS. At each iteration the current area q_{prev} is partitioned into quadrants of equal size. If less than k users are located in the quadrant q where the issuer of the request is located, then q_{prev} is returned. Otherwise, iteration continues considering q as the next area. For what concerns the temporal dimension, the idea is to first generalize the spatial location (with the above algorithm) at a resolution not finer than a given threshold. Then, the request is delayed until k users pass through the generalized spatial location. This defense algorithm has only been validated through experimental results.

An idea similar to the spatial generalization of *IntervalCloaking* is used by Mokbel et al. [40] that propose *Casper*, a framework for privacy protection that includes a generalization algorithm. The main difference with respect to *IntervalCloaking* is that, in addition to the anonymity parameter k, the user can specify the minimum size of the area that is sent to the SP. While it is not explicit in the paper, the idea seems to be that, in addition to k-anonymity, the algorithm also provides a form of location obfuscation. Similarly to *IntervalCloaking*, *Casper* has been validated through experimental results.

Centralized defenses against snapshot, single-issuer and def-aware attacks.

Many papers extend *IntervalCloaking* to provide defenses techniques that guarantee anonymity when more conservative assumptions are made for the adversary model. Kalnis et al. [27], propose the *Hilbert Cloak* algorithm that provides anonymity also in the case in which the adversary knows the generalization function. The idea of *Hilbert Cloak* is to exploit the Hilbert space filling curve to define a total order among users' locations. Then, *Hilbert Cloak* partitions the users into blocks of k: the first block from the user in position 0 to the user in position $k-1$ and so on (note that the last block can contain up to $2 \cdot k - 1$ users). The algorithm then returns the *minimum bounding rectangle* (MBR) computed considering the position of the users that are in the same block as the issuer. The correctness of the *Hilbert Cloak* algorithm is formally provided and the performance of the algorithm has been also experimentally evaluated.

A different algorithm, called *CliqueCloak* is proposed by Gedik et al. [18]. The main difference with respect to the *IntervalCloaking* algorithm is that *CliqueCloak* computes the generalization among the users that actually issue a request and not among the users that are potential issuers. Indeed, *CliqueCloak* collects original requests without forwarding them to the SP until it is possible to find a spatiotemporal generalization that includes at least k pending requests. Then, the requests are generalized and forwarded to the SP. The advantage of the proposed technique, whose correctness is formally proved, is that it allows the users to personalize the degree of anonymity as well as the maximum tolerable spatial and temporal generalizations. However, the algorithm has high computational costs and it can be efficiently executed only for small values of k.

In [35] Mascetti et al. present another three generalization algorithms that are proved to guarantee anonymity against snapshot, single-issuer and def-aware attacks. The aim is to provide anonymity while minimizing the size of the generalized location. The algorithm with the best performance with respect to this metric is called *Grid*. Intuitively, this algorithm partitions all users according to their position along one dimension. Then, it considers the users in the same block as the issuer and it partitions them according to their location along the other dimension. Finally, each block has at least cardinality k and the algorithm computes the generalized location as the minimum bounding rectangle (MBR) that covers the location of the users in the same block as the issuer.

Decentralized defenses against snapshot, single-issuer attacks.

Some papers propose defense techniques that do not require a centralized architecture. Chow et al. [13] propose a decentralized solution called *CloakP2P* in which it is assumed that users can communicate with each other using an ad-hoc network. Basically, before sending the request, a user looks for the $k-1$ closest users in the neighborhood through the ad-hoc network. The location information of the request is then generalized to the region containing these users and the request is issued to the server through one of these users that is randomly selected. This algorithm guarantees privacy only against def-unaware attacks and it is evaluated through experimental results only.

Privè is a distributed protocol based on the *Hilbert Cloak* algorithm ([22]). In this case, the data structure that contains the positions of the users on the Hilbert curve is a B$^+$-tree that is distributed among the users in the system. The generalization is a distributed algorithm that traverses the tree starting from the root and finds the set of users containing the issuer. The algorithm is proven to be correct and guarantees privacy also against def-aware attacks. However, this solution suffers from some scalability issues. To address these issues, Ghinita et al. [21] propose the *MobiHide* algorithm which improves the scalability but that does not guarantee anonymity if the generalization algorithm is known to the adversary. The algorithm is formally validated.

A different decentralized solution is proposed by Hu et al. [26]. The main characteristic of the proposed technique is that it does not require the users to disclose their locations during the anonymization process. Indeed, it is assumed that a user's devices is able to measure the closeness from its peers through its omnidirectional antenna (using WiFi signal, for example). When a request is generalized, the distance information is used to compute the anonymity set and the generalized location is obtained through a secure computation among the users in the anonymity set. The proposed approach is safe against def-aware attacks and its correctness is formally proved.

Centralized defenses against historical, single-issuer attacks.

Several papers further extend the ideas of *IntervalCloaking* to provide a defense in the historical case. The problem of anonymity in the historical, single-issuer case has been first investigated in [7]. In the paper it is shown that the defense technique for the snapshot case cannot be straightforwardly applied to provide protection against a historical attack. In addition, a centralized algorithm is proposed. A brief description of the formalization of the problem presented in the paper follows. To define the notion of historical anonymity, it is reasonable to assume that the LTS not only stores in its database the set of requests issued by each user, but also stores for each user the sequence of her location updates. This sequence is called *Personal History of Locations* (PHL). More formally, the PHL of user u is a sequence of 3D points $(\langle x_1, y_1, t_1\rangle, \dots, \langle x_m, y_m, t_m\rangle)$, where $\langle x_i, y_i\rangle$, for $i = 1, \dots, m$, represents the position of u (in two-dimensional space) at the time instant t_i. A PHL $(\langle x_1, y_1, t_1\rangle, \dots, \langle x_m, y_m, t_m\rangle)$ is defined to be *LT-consistent* with a set of requests $r_1, \dots, r_n$ issued to a SP if for each request r_i there exists an element $\langle x_j, y_j, t_j\rangle$ in the PHL such that the area of r_i contains the location identified by the point x_j, y_j and the time interval of r_i contains the instant t_j. Then, given the set $\bar{R}$ of all requests issued to a certain SP, a subset of requests $\bar{R}' = \{r_1, \dots, r_m\}$ issued by the same user u is said to satisfy *Historical k-Anonymity* if there exist $k-1$ PHLs $P_1, \dots, P_{k-1}$ for $k-1$ users different from u, such that each P_j, $j = 1, \dots, k-1$, is LT-consistent with R'.

Following the main ideas presented in [7] other anonymization techniques for the historical case have been proposed in [12, 50]. The work in [12] also aims at providing protection against a def-aware attack, however it is not clear if the proposed algorithm achieves this goal since it is only evaluated through experimental results. The work in [50] proposes two generalization algorithms, the first one, called *plainKAA*, exploits the same general idea presented in [7]. The second one is an optimization of the first, based on the idea that in the generalization of the requests the users that were not in the anonymity set of a previous request can contribute to anonymity protection. It is unclear if this optimization can preserve historical k-anonymity. Both algorithms are validated through experimental results only.

Mascetti et al. propose a formal model for the historical case [37] and experimentally show that, under certain conservative assumptions, it is not possible to guarantee anonymity without generalizing the user locations to large areas. Under these assumptions, considered in most of the related work on the snapshot case, the adversary knows the association between each user identity and the location of that user. The *ProvidentHider* algorithm is proposed to guarantee anonymity in the historical case under the relaxed assumptions that the adversary knows this association only when users are located in certain areas (e.g., workplaces). The correctness of the algorithm is formally proved and its applicability is experimentally evaluated.

The *Greedy* algorithm for historical k-anonymity We now present a simple centralized defense that belongs to this category. This algorithm enforces historical anonymity, and it will be used as reference algorithm in Section 2.3.

Our algorithm uses a snapshot anonymization algorithm, like already mentioned *Grid*, that is safe against snapshot, single-issuer and def-aware attacks. We modify this algorithm by adding the requirement that the perimeter of the MBR be always smaller than a user-given $maxP$ value. To achieve this, we basically shrink the obtained MBR from the snapshot algorithm by recursively removing users that are distant from the issuer until the perimeter of the MBR is smaller than $maxP$.

The idea of the *Greedy* algorithm was first proposed in [7] and a similar algorithm was also described in [50]. This algorithm computes the generalization of the first request r in a trace using an algorithm for the snapshot case. (In our implementation, we use *Grid* as the snapshot algorithm to compute the generalization of the first request.) When this first request is generalized, the set A of users located in the generalized location for the first request is stored. The generalized locations of each subsequent request r' that is linked with r is then taken as the MBR of the location of the users in A at the time of r'. As in the modification of the *Grid* algorithm, when the MBR is larger than $maxP$, we recursively shrink it until its perimeter is smaller than $maxP$. Algorithm 1 gives the pseudocode. This algorithm is called initially with the first request r and empty set $A = \emptyset$, and subsequently, it is called with the successive request and the A' returned from the previous execution.

Centralized defenses against multiple-issuer attacks.

Preliminary results on the privacy leaks determined by multiple-issuer attacks are reported in [4]. Defenses for this kind of attacks are based on accurately generalizing location (as a quasi-identifier) in order to obtain QI-groups of requests with a certain degree of *diversity* in private values. A defense against multiple-issuer attacks both in the snapshot and in a limited version of the historical case is proposed by Riboni et Al. [42] using a combination of identity anonymity and private information obfuscation techniques. Further research is needed along this line. For example, to understand under which conditions close values in private information can really be considered different (e.g., location areas).

2.2.2 Defenses based on private information obfuscation

As mentioned at the beginning of this section, these defenses aim at obfuscating private information released by users' requests as opposed to generalizing quasi-identifiers. To the best of our knowledge, all of the techniques in this category consider *location* as the private information to be protected, and implicitly or explicitly assume that user identity is known to the adversary or could be discovered. In the following of this chapter, we use *location obfuscation* to denote the general category of defenses aimed at obfuscating the exact location as private information of the (possibly identified) issuer.

Differently from the anonymity based defenses considering location as

Algorithm 1 *Greedy*

Input: a request r, an anonymity set A, anonymity level k, and a maximum perimeter $maxP$.

Output: a generalized request r' and an anonymity set A'.

Method:

```
1: find the MBR of all the current locations (at the time of request $r$) of
   users in $A$ (note that if $A = \emptyset$ then the MBR is empty).
2: if (the perimeter of the MBR is smaller than $maxP$) then
3:     if ($|A| > 1$) then
4:         replace the spatial information in $r$ with the MBR, obtaining $r'$
5:         let $A' = A$
6:     else
7:         call Grid algorithm* with $r$, $k$, and $maxP$, obtaining $r'$
8:         let $A'$ be the set of users currently in the spatial region of $r'$
9:     end if
10: else
11:     recursively shrink the MBR until its perimeter is smaller than $maxP$
12:     replace the spatial region in $r$ with the resulting MBR, obtaining $r'$
13:     let $A'$ be the set of users currently located in the resulting MBR
14:     if ($|A'| \leq 1$) then
15:         call Grid algorithm with $r$, $k$, and $maxP$, obtaining $r'$
16:         let $A'$ be the set of users currently in the spatial region of $r'$
17:     end if
18: end if
19: return $r'$ and $A'$
```

* *Instead of* Grid*, other snapshot algorithms can be used here.*

quasi-identifier, in this case it is less important to know the location of other users in order to provide privacy protection. For this reason, most of the location obfuscation techniques do not require a common location-aware trusted entity and, according to our categorization, they have a decentralized architecture. Sometimes these defenses are also claimed to provide a form of k-anonymity, leading to confusion with anonymity based defenses. The underlying idea is that due to the obfuscation, the location of the issuer (who is possibly not anonymous at all) cannot be distinguished among k possible locations. In order to avoid confusion this property should be called *location anonymity.*

The idea of protecting location privacy by obfuscating location information was first proposed by Gruteser et al. [24]. The technique is aimed at avoiding the association of a user with a *sensitive area* she is crossing or approaching. The proposed defense is based on appropriately suspending user requests, ensuring that the location of the user may be confused among at least other k areas. The proposed technique requires a centralized entity, but it should not be difficult to modify the proposed algorithm so that it could be run directly on the users' mobile device. This defense algorithm is only validated via experiments. It is also not clear which privacy guarantees are provided if the adversary knows the algorithm.

Duckham et al. propose a protocol that allows a user to obtain the result of 1-NN (Nearest Neighbor) queries among a set of points of interest without disclosing her exact location [15]. The protocol is iterative. At the first iteration the user sends her obfuscated location to the SP that replies with the pair $\langle q, C \rangle$ where q is the point of interest having the highest confidence C of being the closest to the user. At each following iteration, the user can decide whether to provide additional location information in order to obtain a result with higher confidence. It is not specified how the generalization of the user's location is computed.

A different approach, proposed by Kido et al. [30], consists in sending, together with the real request, a set of fake requests. Since the adversary cannot distinguish the real request from the fake ones, it cannot discover the real location of the issuer, among the locations of the fake requests. This decentralized solution is effective also in the case in which the adversary knows the defense function. However, this solution has the problem that, in order to effectively protect the location information, a high number of fake requests should be sent hence impacting on the communication costs. The technique is validated through experimental results only.

In [2], Ardagna et al. propose to use a combination of location obfuscation techniques and a metric to measure the obfuscation achieved. The difference with respect to other approaches is that the resulting obfuscation area may not contain the actual location of the issuer; moreover, the location measurement error introduced by sensing technologies is taken into account. It is not formally proved that the proposed defense protects against def-aware attacks. According to our categorization, the paper considers a centralized architecture, even if the proposed obfuscation techniques can be probably run on the client side.

Yiu et al. [51] proposed a different solution to obfuscate location information, specific for LBS requests that require K-NN queries. The idea of the algorithm, named SpaceTwist, is to issue each request as if it would originate from a location different from the real user location. The request may be repeated (from the same fake location) incrementally retrieving more nearest neighbor resources, until a satisfactory answer for the real location is obtained. This solution is particularly interesting since it does not require the existence of the centralized entity that provide privacy protection and involves no range NN queries on the server side. In the paper it is also formally shown how the adversary can compute the area where the user is possibly located under the assumptions that the adversary only knows the fake location, the number of requested resources, the replies from the server and the termination condition of the algorithm.

Solutions based on location obfuscation and/or encryption have also been proposed in the context of location-based proximity services, that we describe in detail in Chapter 3. A more detailed survey of the solutions proposed in literature for this category of services is in Section 3.1.

Referring to our categorization of attacks, the existing location obfuscation defenses focus on snapshot and single-issuer attacks. Example 3 shows that, in some cases, a historical attack can further restrict the possible locations of a user.

Example 3. *A request issued by Alice is obfuscated in such a way that an adversary only knows that Alice is located in an area* A_1 *at time* t_1*. After a short time, Alice issues a second request that is obfuscated in such a way that the adversary knows that Alice is located somewhere in area* A_2 *at time* t_2*. Now, assume that there is a subregion* A' *of* A_2 *such that, due to speed constraints, no matter where Alice were located in* A_1 *at time* t_1*, she has no way to get to* A' *at time* t_2*. Now the adversary knows that at time* t_2*, Alice cannot be located in* A' *and hence she must be in* $A_2 \setminus A'$*.*

This class of *dynamic* attacks based on the knowledge of speed constraints have been further investigated by Ghinita et al. [19] for generic location-based services. Recently, Freni et al. [17] proposed two techniques based on spatio-temporal obfuscation to protect privacy in the context of resource publishing services in geo-social networks, also considering an adversary aware of the maximum velocity of the users.

2.2.3 Encryption based defenses

We call encryption based, the defense proposals based on private information retrieval (PIR) techniques. The general objective of a PIR protocol is to allow a user to issue a query to a database without the database learning the query. In [20] this techniques is used to protect users' privacy in the LBS that computes 1-NN queries. The proposed solution is proved to solve the privacy problem under the most conservative assumptions about the adversary model as it does not reveal any information about the requests to the adversary. Nevertheless, some concerns arises about the applicability of the proposed technique. First, the proposed solution applies to 1-NN queries only and it is not clear how it could be extended to other kinds of queries like K-NN queries or range queries. Second, this technique has high computational and communication overhead. Indeed, the experimental results shown in the paper give evidence that, although using a small database of objects to be retrieved, the computation time on the server side is in the order of seconds, while the communication cost is in the order of megabytes. In particular, the amount of data that needs to be exchanged between the server and the client is larger than the size of the database itself. It is not clear for which kind of services this overhead could be tolerable.

Two more recent solutions propose PIR protocols that are aided by the presence of a *secure hardware* on the server [29, 41]. The secure hardware is a tamper-resistant additional CPU installed on the server that can run programs independently from the server itself and that can communicate directly with the clients over secure channel. In the solution proposed by Khoshgozaran et al. [29], upon receiving a K-NN query the secure hardware computes a set of cells that contains all the points of the query result. The client then retrieves the content of these cells using a PIR protocol. Papadopoulos et al. [41] observe that this approach could still be vulnerable to attacks based on the cardinality of the retrieved cells, as the SP can still observe the number of queries issued by one client. Therefore, their solution requires the server to compute a *query plan* to be distributed to the client before issuing a query. The query plan indicates the exact number of PIR retrievals that must be issued by a client during a transaction. Even if a particular query result could be retrieved with less retrievals, to ensure safety the client is required to issue *dummy* retrievals until the number indicated in the query plan is reached.

2.3 Impact of realistic simulations on the evaluation of defense techniques

As we motivated in the previous section, the correctness of an anonymity-preserving technique can be formally proved based on the specific assumptions made on the adversary model. However, in practice, different adversaries may have different background knowledge and inferencing abilities. Hence, one approach consists in stating conservative assumptions under which anonymity can be guaranteed against a broad range of potential adversaries. The drawback of this approach is clear from the conservative assumptions about location knowledge considered so far by anonymity based solutions: in order to protect from the occasional knowledge by the adversary about people present at a given location (unknown to the defender), it is (often implicitly) assumed the same knowledge for all locations. Such assumptions are not realistic and lead to overprotecting the users' anonymity, hence negatively impacting the quality of service. A different approach, taken by several researcher is experimental evaluation. Since large set of real, accurate data are very hard to obtain, in most cases experiments are based on synthetic data generated through simulators. In this section we focus on validating anonymity-based defense techniques, and we show that in order to obtain significant results, simulations must be very carefully designed. In addition to evaluating the *Greedy* algorithm as a representative of historical anonymity based defenses, we are interested in the following more general questions: a) *how much does the adversary model affect the privacy obtained by the defense according to the evaluation?*, and b) *how much does the specific service deployment model affect the results of the evaluation?*

It should be noted that question b) also applies to those experimental evaluations that aim to show the effectiveness and the impact of privacy-preserving techniques that not necessarily oriented at enforcing anonymity, like the obfuscation or encryption based defenses already discussed in the previous section.

2.3.1 The *MilanoByNight* simulation

In order to carefully design the simulation, we concentrate on a special category of proximity based services, called *friend-finders*, that can be accessed anonymously or by using a pseudonym. Commercial examples of such services are mobile dating services like Grindr or MeetMoi, in which users are looking for other people with similar sexual orientation. As these services can be accessed anonymously, anonymity-based techniques for LBS like the ones presented in this chapter are normally applicable. In Chapter 3 we will also employ this simulation during the experimental evaluation of the defense techniques based on private information obfuscation specifically designed for proximity services.

A first privacy threat for a user of this proximity-based service is the association of that user's identity with the service parameters and, in particular, with the group of target participants, since this can reveal the user's interests or other private information. Even if the user's identity is not explicit in a request, an adversary can obtain this association, by using the location information of a request as a quasi-identifier.

A second privacy threat is the association of the user's identity with the locations visited by that user. We recall that this association takes place independently from the service requests if the adversary's background location knowledge is "complete" (see Section 2.1). However, consider the case in which the background knowledge is "partial" i.e., it contains the association between user identity and location information only for some users in some locations at some time instants. Example 4 shows how, in this case, an adversary can exploit a set of friend-finder requests to derive location information that are not included in the background knowledge.

Example 4. *User A issues a friend-finder request r_1. An adversary obtains r_1 and discovers that A is the issuer by joining the location information in the request with his background knowledge (i.e., the location information of r_1 is used as quasi-identifier). Then, A moves to a different location and issues a request r_2. The adversary obtains r_2, but in this case his background knowledge does not contain sufficient information to identify the issuer of the request. However, if the adversary can understand that r_1 and r_2 are linked (i.e., issued from the same issuer), then he derives that A is also the issuer of r_2 and hence obtains new location information about A.*

We suppose that the friend-finder service is primarily used by people during entertainment hours, especially at night. Therefore, the ideal dataset for our experiments should represent movements of people on a typical Friday or Saturday night in a big city, when users tend to move to entertainment places. To our knowledge, currently there are no datasets like this publicly available, specially considering that we want to have large scale, individual, and precise location data (i.e., with the same approximation of current consumer GPS technology).

Relevant Simulation Parameters

For our experiments we want to artificially generate movements for $100,000$ users on the road network of Milan[2]. The total area of the map is 324 km^2, and the resulting average density is 308 users/km^2. The simulation includes a total of $30,000$ home buildings and $1,000$ entertainment places; the first value is strictly related to the considered number of users, while the second is based on real data from public sources which also provide the geographical distribution of the places. Our simulation starts at 7 pm and ends at 1 am. During these hours, each user moves from house to an entertainment place, spends some time in that place, and possibly moves to another entertainment place or goes back home.

All probabilities related to users' choices are modeled with probability distributions. In order to have a realistic model of these distributions, we prepared a survey to collect real users data. We are still collecting data, but the current parameters are based on interviews of more than 300 people in our target category.

Weaknesses of mostly random movement simulations

Many papers in the field of privacy preservation in LBS use artificial data generated by moving object simulators to evaluate their techniques. However, most of the simulators are usually not able to reproduce a realistic behavior of users. For example, objects generated by the Brinkhoff genera-

[2] $100,000$ is an estimation of the number of people participating in the service we consider.

tor [10] cannot be aggregated in certain places (e.g., entertainment places). Indeed, once an object is instantiated, the generator chooses a random destination point on the map; after reaching the destination, the object disappears from the dataset. For the same reason, it is not possible to reproduce simple movement patterns (e.g.: a user going out from her home to another place and then coming back home), nor to simulate that a user remains for a certain time in a place.

Despite these strong limitations, we made our best effort to use the Brinkhoff simulator to generate a set of user movements with characteristics as close as possible to those described above. For example, in order to simulate entertainment places, some random points on the map, among those points on the trajectories of users, were picked. The simulation has the main purpose of understanding if testing privacy preservation over random movement simulations gives significantly different results with respect to more realistic simulations.

Generation of user movements with a context simulator

In order to obtain a dataset consistent with the parameters specified above, we need a more sophisticated simulator. For our experiments, we have chosen to customize the Siafu context simulator [33]. With a context simulator it is possible to design models for agents, places and context. Therefore, it is possible to define particular places of aggregation and make users dynamically choose which place to reach and how long to stay in that place.

The most relevant parameters characterizing the agents' behavior are derived from our survey. For example, one parameter that characterizes the behavior of the agents is the average time spent in an entertainment place; This value was collected in our survey and resulted to have the following values: 9.17% of the users stays less than 1 hour, 34.20% stays between 1 and 2 hours, 32.92% stays between 2 and 3 hours, 16.04% stays between 3 and 4 hours, and 7.68% stays more than 4 hours. Details on the simulation can be found in [38].

2.3.2 Experimental settings

In our experiments we used two datasets of users movements. The dataset AB (Agent-Based) was generated with the customized Siafu simulator, while the dataset MRM (Mostly Random Movement) was created with the Brinkhoff simulator. In both cases, we simulate LBS requests for the friend-finder service by choosing random users in the simulation, we compute for each request the generalization according to a given algorithm, and finally we evaluate the anonymity of the resulting request as well as the Quality of Service (QoS).

Different metrics can be defined to measure QoS for different kind of services. For instance, for the friend-finder service we are considering, it would be possible to measure how many times the generalization leads the SP to return an incorrect result i.e., the issuer is not notified of a close-by friend or, vice versa, the issuer is notified for a friend that is not close-by. While this metric is useful for this specific application, we want to measure the QoS independently from the specific kind of service. For this reason, in this chapter we evaluate how QoS degrades in terms of the perimeter of the generalized location.

In addition to the dataset of user movements, we identified other two parameters characterizing the deployment model that significantly affect the experimental results: the *number of users* in the system, which remains almost constant at each time instant and the user-required degree of indistinguishability k. These two parameters, together with the most important others, are reported in Table 2.1, with the values in bold denoting default values.

We also identified three relevant parameters that characterize the adversary model. The parameter P_{id-in} indicates the probability that the adversary can identify a user when she is located in a entertainment place while P_{id-out} is the probability that the adversary identifies a user in any other location (e.g., while moving from home to a entertainment place). While we also perform experiments where the two probabilities are the same, our

Table 2.1: Parameter values

Parameter	**Values**
dataset	***AB***, *MRM*
number of users	10k, 20k, 30k, 40k, 50k, 60k, 70k, 80k, 90k, **100k**
k	**10**, 20, 30, 40, 50, 60
P_{id-in}	0.1, **0.2**, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
P_{id-out}	0.01, **0.02**, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1
P_{link}	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, **0.87**, 0.9, 1.0

scenario suggests as much more realistic a higher value for P_{id-in} (it is considered ten times higher than P_{id-out}). This is due to the fact that restaurants, pubs, movie theaters, and similar places are likely to have different ways to identify people (fidelity or membership cards, WiFi hotspots, cameras, credit card payments, etc.) and in several cases more than one place is owned by the same company that may have an interest in collecting data about its customers. Finally, P_{link} indicates the probability that two consecutive requests can be identified as issued by the same user.[3] While we perform our tests considering a full range of values, the specific default value reported in the table is due to a recent study on the ability of linking positions based on spatiotemporal correlation [49].

The experimental results we show in this section are obtained by running the simulation for 100 issuers and then computing the average values.

In our experiments we evaluated two generalization algorithms. One algorithm is *Greedy* which is described in Section 2.2 and is a representative of the historical generalization algorithm proposed so far [7, 12, 50]. The other algorithm is *Grid* which is briefly described in Section 2.2.1 is a representative of the snapshot generalization algorithms. In [35] *Grid* is shown to have better performance (in terms of the quality of service) when compared

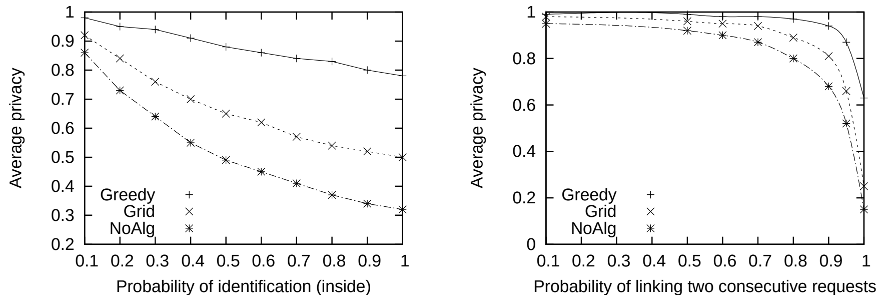
[3]The limitation to consecutive requests is because in our specific scenario we assume linking is performed mainly through spatiotemporal correlation.

to other snapshot generalization algorithms like, for example, *Hilbert Cloak*. We also evaluated the privacy threat when no privacy preserving algorithm is applied. The label *NoAlg* is used in the figures to identify results in this particular case.

2.3.3 Impact of the adversary model on the evaluation of the generalization algorithms

We now present a set of experiments aimed at evaluating the impact of the adversary model on the anonymity provided by the generalization algorithms.

Two main parameters characterizing the adversary model are P_{id-in} and P_{link}. In Figure 2.2(a) we show the average privacy, in terms of probability of re-identification, for different values of P_{id-in} when, in each test, P_{id-out} is set to $P_{id-in}/10$. As expected, considering a trace of requests, the higher the probability of identifying users in one or more of the regions from which the requests in the trace were performed, the smaller is the level of anonymity.



(a) Varying P_{id-in} ($P_{id-out} = P_{id-in}/10$). (b) Varying P_{link}.

Figure 2.2: Average anonymity.

Figure 2.2(b) shows the impact of P_{link} on the average privacy. As expected, high values of P_{link} lead to small values of privacy. Our results show that the relation between the P_{link} and privacy is not linear. Indeed, privacy depends almost linearly on the average length of the traces identified by the adversary. In turn, the average length of the traces grows almost exponentially with the value of P_{link}.

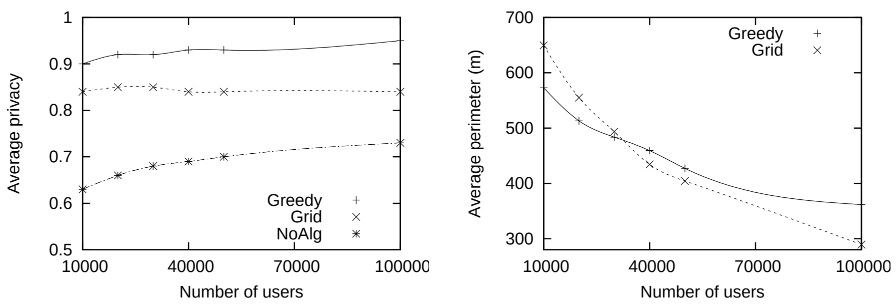
To summarize the first set of experiments, our findings show that the parameters that characterize the adversary model significantly affect the evaluation of the generalization algorithms. This implies that when a generalization algorithm is evaluated it is necessary to estimate realistic values for these parameters. Indeed, an error in the estimation may lead to misleading results.

2.3.4 Impact of the deployment model on the evaluation of the generalization algorithms

We now show a set of experimental results designed to evaluate the impact of the deployment model on the evaluation of the generalization algorithms.

Figure 2.3(a) shows that the average privacy obtained with *Greedy* and *Grid* is not significantly affected by the size of the total population. Indeed, both algorithms, independently from the total number of users, try to have generalized locations that cover the location of k users, so the privacy of the requests is not affected. However, when the density is high, the two algorithms can generalize to a small area, while when the density is low, a larger area is necessary to cover the location of k users (see Figure 2.3(b)). On the contrary, the privacy obtained when no generalization is performed is significantly affected by the total population. Indeed, a higher density increases the probability of different users to be in the same location and hence it increases privacy also if the requests are not generalized.

The set of tests reported in Figure 2.4 compares the privacy achieved by the *Greedy* algorithm on the two datasets for different values of k and for different values of QoS. The experiments on MRM were repeated trying also larger values for the QoS threshold ($maxP = 2000$ and $maxP = 4000$), so three different versions of MRM appear in the figures. In order to focus on these parameters only, in these tests the probability of identification was set to the same value for any place ($P_{id-in} = P_{id-out} = 0.1$), and for the MRM dataset the issuer of the requests was randomly chosen only among those



(a) Average privacy. (b) Average perimeter.

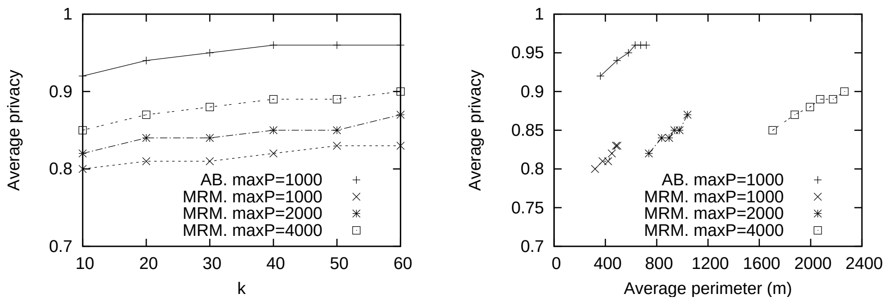
Figure 2.3: Performance evaluation for different values of the total population.

that stay in the simulation for 3 hours, ignoring the ones staying for much shorter time that inevitably are part of this dataset. This setting allowed us to compare the results on the two datasets using the same average length of traces identified by the adversary.

Figure 2.4(a) shows that the average privacy of the algorithm evaluated on the AB dataset is much higher than on the MRM dataset. This is mainly due to the fact that in AB users tend to concentrate in a few locations (the entertainment places) and this enhances privacy. This is also confirmed by a similar test performed without using any generalization of locations; we obtained values constantly higher for the AB dataset (the average privacy is 0.67 in AB and 0.55 in MRM).

In Figure 2.4(b) we show the QoS achieved by the algorithm in the two datasets with respect to the average privacy achieved. This result confirms that the level of privacy evaluated on the AB dataset using small values of k and $maxP$ for the algorithm cannot be observed on the MRM dataset even with much higher values for these parameters.

From the experiments shown in Figure 2.4 we can conclude that if the MRM dataset is used as a benchmark to estimate the values of k and $maxP$ that are necessary to provide a desired average level of privacy, then the results will suggest the use of values that are over-protective. As a consequence, it is possible that the service will exhibit a much lower QoS than the one that could be achieved with the same algorithm.



(a) Average privacy as a function of the level of indistinguishability k.

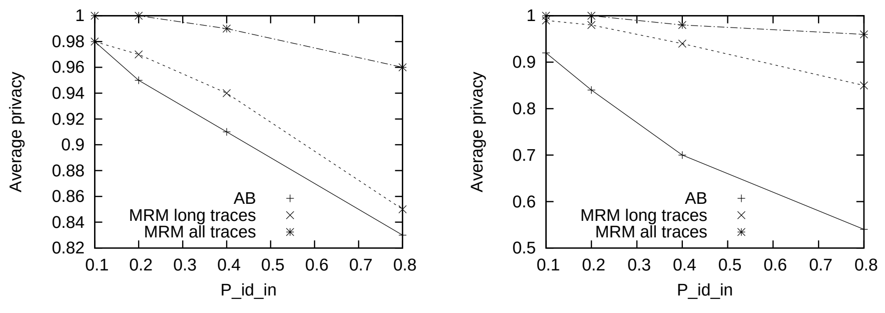
(b) Average privacy as a function of the average perimeter.

Figure 2.4: Evaluation of the *Greedy* algorithm using *AB* and *MRM* data sets. $P_{id-in} = P_{id-out} = 0.1$

The above results may still support the safety of using *MRM*, since according to what we have seen above a technique achieving a certain level of privacy may only do better in a real scenario. However, our second set of experiments shows that this is not the case.

In Figure 2.5 we show the results we obtained by varying the probability of identification. For this test, we considered two sets of issuers in the *MRM* data set. One set is composed by users that stay in the simulation for 3 hours, (*MRM long traces*, in Figure 2.5), while the other contains issuers randomly chosen in the entire set of users (*MRM all traces*, in Figure 2.5), hence including users staying in the simulation for a much shorter time.

In Figure 2.5(a) and 2.5(b) we can observe that the execution on the *MRM* dataset leads to evaluate a privacy level that is higher than the one obtained on the *AB* dataset. In particular, the evaluation of the *Grid* algorithm using the *MRM* dataset (Figure 2.5(b)), would suggest that the algorithm is able to provide a high privacy protection. However, when evaluating the same algorithm using the more realistic dataset *AB*, this conclusion seems to be incorrect. In this case, the evaluation on the *MRM* dataset may lead to underestimate the privacy risk, and hence to deploy services based on generalization algorithms that may not provide the minimum required level of privacy.



(a) *Greedy* algorithm. (b) *Grid* algorithm.

Figure 2.5: Average privacy using AB and MRM data sets. $P_{id-out} = P_{id-in}/10$.

Chapter 3

Privacy preservation in Proximity Services

In the previous chapter we presented the most relevant techniques to preserve privacy in generic LBSs, in which a user issues a location-based request to a SP, and does not want the SP to obtain sensitive information about her. In the context of GeoSNs, however, other users participating in the service may obtain the location information of a certain user, and this may pose additional threats to her privacy.

The focus of this chapter is the problem of protecting privacy in location-based proximity services, a particular category of service that is often offered by GeoSN providers to let their users discover which of their buddies are nearby. The existing techniques for LBS, presented in the previous chapter, are not directly applicable to this scenario, as it will be explained in Section 3.1. In the same section we present the works that have been already presented in literature. In Section 3.2 we formally define the privacy problem, the privacy threats and the considered adversary model. Section 3.3 describes some protocols that we propose to enforce users' privacy in this context, including a formal analysis of each of the proposed protocols. The implementation and experimental evaluation of the protocols will be presented in the next chapter.

3.1 Related work

Computing proximity involves the continuous evaluation of spatial range queries over a set of moving entities, with the radius range possibly changing [11, 46]. The literature on this problem is both from the database, and the mobile computing community; recent contributions are briefly surveyed in [1], where an efficient algorithm for proximity detection named *Strips* is presented. Another solution for efficient proximity detection has been recently proposed by Yiu et al. [52]. The goal of this and similar approaches is the efficiency in terms of computation and communication complexity, while privacy issues are mostly ignored.

The techniques presented in Section 2.2 cannot be directly applicable to the context of proximity based services. As already mentioned, anonymity-based defenses aim to enforce the anonymous access to a LBS by ensuring that a request cannot be reassociated to the original issuer. In the case of proximity based services, we assume that the identity of the user may be easily obtained or already known to the adversary, and we consider the location of the user as a sensitive information to be protected. Most of the defenses based on private information obfuscation that we mentioned in Section 2.2.2 have been applied for LBS performing k-NN spatial queries, and do not apply to proximity detection. Some encryption-based PIR approaches provide support for range and k-NN spatial queries, but they all require the index of the database to be built offline, and hence they are not still suitable for a database of moving entities like users of a proximity services, that requires frequent updates to the index structure. Khoshgozaran et al. [28] propose a system to maintain an encrypted index on the server side and efficiently update it, which makes it suitable for maintaining a database of moving buddies. The system supports encrypted range and k-NN spatial queries, hence it could be used to offer proximity based services. However, the system requires users to be organized in groups, with each group sharing a symmetric secret key, and all the users in a group must trust each other. Furthermore, the proposed techniques for retrieving the query results seem to be vulnerable to cardinality attacks, like the ones considered by Papadopoulos et al. [41] for PIR techniques, if the SP has an a-priori knowledge about the distribution of the users.

Ruppel et al. [43] propose a technique for privacy preserving proximity computation based on the application of a distance preserving transformation on the location of the users. However, the SP is able to obtain the exact distances between users, and this can lead to a privacy violation. For example, having this knowledge, it is possible to construct a weighted graph of all the users, assigning to each edge connecting two users their exact distance. It is easily seen that a "relative" distribution of the user locations can be extracted from this graph. If the SP has a-priori knowledge about the distribution of the users (as considered in our adversary model), it is possible to merge the distribution resulting from the graph with the a-priori one, thus revealing some location information about the individuals. In addition, there is no privacy guarantee with respect to the other users participating in the service.

Zhong et al. propose three different techniques for privacy preservation in proximity-based services called *Louis*, *Lester* and *Pierre* [53]. These techniques are decentralized secure computation protocols based on public-key cryptography. *Louis* is a three-parties secure computation protocol. By running this protocol, a user A gets to know whether another user B is in proximity without disclosing any other location information to B or to the third party T involved in the protocol. T only helps A and B compute their proximity, and it is assumed to follow the protocol and not to collude with A or B. However, T learns whether A and B are in proximity. Considering our adversary model, which will be explained in detail in Section 3.3, this third party cannot be the SP that may use proximity information to violate location privacy, and it is unlikely to be played by a third buddy since it would involve significant resources. The *Lester* protocol allows a user A to compute the exact distance from a user B only if the distance between the two users is under a certain threshold chosen by B. The main advantage of these two techniques is that they protect a user's privacy without introducing any approximation in the computation of the proximity. However, *Louis* incurs in significant communication overheads, and *Lester* in high computational costs. In addition, the only form of supported privacy protection with respect to the buddies is the possibility for a user to refuse to participate in the protocol initiated by a buddy if she considers the requested proximity threshold too small. The *Pierre* protocol partitions the plane where the service is provided into a grid, with each cell having edge equal to the requested distance threshold. The locations of the users are then generalized to the corresponding cell, and two users are considered in proximity if they are located in the same cell or in two adjacent cells. The achieved quality of service decreases as the requested proximity threshold grows. We will explain in more detail the actual impact on service precision in Section 4.1. Finally, it should be mentioned that Lester and Pierre protocols are based on a buddy-to-buddy communication, and although this can guarantee total privacy with respect to the SP (as no SP is involved in the computation), scalability issues may arise since each time a user moves she needs to communicate her new position to each of her buddies.

Another solution for privacy preserving computation of proximity, called `FriendLocator`, has been proposed by Šikšnys et al. [48]. Similarly to Pierre, two users are considered in proximity when they are located in the same cell or two adjacent cells of the grid constructed considering the proximity threshold shared by the users. An interesting aspect of the proposed solution is the location update strategy, which is designed to reduce the total number of location updates to be sent by the users, hence reducing communication costs. Two users share a hierarchy of grids, where each grid is identified by a *level*. The larger the value of the level is, the finer the grid. The highest level grid is the one in which the edge of a cell is equal to the proximity threshold. The detection of proximity is then incremental, i.e. if two users are in adjacent cells at the level n grid, then their respective cells in the grid of level $n+1$ are checked, until they are detected either not to be in proximity, or to be in proximity considering the highest level grid. With this solution, when two users are detected not to be in proximity at a certain level l, there is no need for them to check again the proximity until one of them moves to a different cell of the level l grid. As a consequence, less location updates are needed, and this is experimentally shown to significantly reduce the total number of messages exchanged. However, the `FriendLocator` protocol reveals some approximate information about the distance of users to the SP (e.g. the level in which the incremental proximity detection protocol terminates and whether the buddies are in proximity at that level). As already observed for the Louis protocol, in our adversary model this information can lead to a privacy violation. Furthermore, the impact on the quality of service of using a large proximity threshold is identical to the Pierre protocol discussed above.

A more recent solution by the same authors [47], called `VicinityLocator`, solves this problem by letting users specify their privacy preferences as spatial granularities (see Section 3.2.2) independently from the requested proximity threshold. A similar location update strategy is employed to minimize the communication costs. However, similarly to `FriendLocator`, the SP learns some information about the distance of the users, and this could lead to a privacy violation in our adversary model.

3.2 Problem definition

In this section we formally define the service we are considering, the users' privacy concerns and requirements, the adversary model, and the occurrence of a privacy violation.

3.2.1 The proximity service

By issuing a proximity request, user A is interested to know, for each of her buddies B, if the following condition is satisfied:

$$d(loc_A, loc_B) \leq \delta_A \tag{3.1}$$

where $d(loc_A, loc_B)$ denotes the Euclidean distance between the reported locations of A and B and δ_A is a threshold value given by A. When Equation 3.1 is true, we say that B *is in the proximity of* A. The proximity relation is not symmetric, since δ_B may be different from δ_A,

Two different subcategories of proximity-based services can be identified, based on how the buddies of a user are selected. If the buddies are explicitly added as "friends", like in most social networks and instant messaging applications, we call them "contact-list-based" services. On the contrary, if the buddies are retrieved through a particular query e.g. based on their interest, we call it a "query-driven" proximity service. Technically, the main difference is that in the "contact-list-based" service it is reasonable to assume that each user can share a secret with each of her buddies, as we do in our proposed techniques. On the contrary, in the case of "query-driven" services, the set of buddies may change dynamically, and the number of buddies can be potentially very large. In this situation, it may not be practical to share a secret with each buddy. This categorization is important when discussing the applicability of a certain technique to a real-world scenario, as in many solutions it is required that users share secret information.

With the presence of a service provider (SP), and in absence of privacy concerns, a simple protocol can be devised to implement the proximity service: The SP receives location updates from each user 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 different threshold values for different buddies, in this chapter, for simplicity, we consider the case in which each user A defines a single value δ_A for detecting the proximity of all of her buddies. When the SP receives a location update, it can recompute the distance between A and each buddy (possibly with some filtering/indexing strategy for efficiency) and communicate the result to A. In a typical scenario, if B is in proximity, A may contact him directly or through the SP; however, for our purposes, we do not concern ourselves as what A will do once notified. In the following of this chapter we refer to the above protocol as the *Naive* protocol.

3.2.2 Privacy concerns and privacy requirements

The privacy we are considering in the scenario of proximity-based services is *location privacy*: we assume that a user is concerned about the uncontrolled disclosure of her location information at specific times.

Considering the *Naive* protocol, it is easily seen that the SP obtains the exact location of a user each time she issues a location update. Furthermore, a user's location information is also disclosed to her buddies. If Alice is in the proximity of Bob (one of her buddies), then Bob discovers that Alice is located in the circle centered in his location with radius δ_{Bob}. Since δ_{Bob} is chosen by Bob and can be set arbitrarily without consent from Alice, Alice has no control on the location information disclosed to Bob.

Our definition of location privacy is based on the idea that the users should be able to control the location information to be disclosed. In the considered services, a user may prefer the service provider to have as little information about her location as possible, and the buddies not to know her exact position, even when the proximity is known to them. Moreover, the exchanged information should be protected from any eavesdropper.

In general, the level of location privacy can be represented by the uncertainty that an external entity has about the position of the user. This uncertainty is a geographic region, called *minimal uncertainty region* (MUR), and its intuitive semantics is the following: the user accepts that the adversary knows she is located in a MUR R, but no information should be disclosed about her position within R.

In the solutions presented in this dissertation, each user can express her privacy preferences by specifying a partition of the geographical space defining the MURs that she wants guaranteed. For example, Alice specifies that her buddies should never be able to find out the specific campus building where Alice currently is; in this case, the entire campus area is the minimal uncertainty region. The totality of these uncertainty regions for a user 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 this concept is simple. Similar to a temporal granularity [8], a spatial granularity can be considered a subdivision of the spatial domain into a discrete number of non-overlapping regions, called *granules.* For simplicity, we consider only granularities[1] that partition the spatial domain, i.e., the granules of a granularity do not intersect and the union of all the granules in a granularity yields exactly the whole spatial domain. Each granule of a granularity G is identified by an *index* (or a *label*). We denote with $G(i)$ the granule of the granularity G with index i.

Users specify their *privacy requirements* via spatial granularities, with each granule being a MUR. In our solutions we assume that a user A can specify 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. In one extreme case, for example, if a user A does not want her privacy to be protected with respect to her buddies then A sets her G_A^U privacy preference to the *bottom granularity* $\perp$ (a granularity that contains a granule for each basic element, or pixel, of the spatial domain). In the other extreme, if user A wants *complete* location privacy, for example with respect to the SP, then she sets her G_A^{SP} privacy preference to the *top granularity* $\top$, i.e., the granularity that has a single granule covering the entire spatial domain. In this case, A wants the entire spatial domain as MUR.

Our approach can be easily extended to model the case in which a user specifies different granularities for different buddies or for different groups

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

of buddies. However, for the sake of clarity, we will assume that a single G_A^U privacy is specified by each user.

3.2.3 Adversary model and privacy preservation

In the GeoSN setting, a typical user may have different privacy concerns with respect to her buddies or with respect to third-party entities, like the SP. For this reason, we choose to consider two separate adversary models, one for the SP and one for the buddies, respectively. Assuming the SP and the buddies as potential adversaries, also models other types of adversaries. Firstly, it models the case of an external entity taking control of the SP system or of a buddy's system. Secondly, it models the case of an external entity eavesdropping one or more communication channels between users and the SP. Note that, in the worst case, the eavesdropper can observe all the messages that are exchanged in the protocol. Since the same holds for the SP, the eavesdropper can learn at most what the SP learns.

The techniques we present in this dissertation not only guarantee each user's privacy requirement against these two adversary models, but also they are able to give some privacy guarantees in the case of a set of colluding buddies. This will be discussed in more detail in Section 3.3.6.

In both adversary models we assume that the adversary has exactly the following knowledge:

- the protocol,
- the spatial granularities adopted by each user, and
- an a-priori probabilistic distribution of the locations of the users.

The two models differ in the sets of messages received during a protocol run, and in their ability (defined by the protocol in terms of availability of cryptographic keys) to decrypt the content of the messages.

The a-priori knowledge of the location of a user A is given by a location random variable pri_A with the probability mass distribution denoted

$P(pri_A)$. In other words, as prior knowledge we assume that the location of a user A follows a known distribution given by the distribution of the random variable pri_A. Note that we assume the spatial domain is discrete, i.e., a countable set of "pixels".

Let M be the set of messages exchanged between the entities involved in the service. The adversary can compute the *a-posteriori* probability distribution of the location random variable $post_A$ as the distribution of the location of A under the given messages M and the prior knowledge pri_A:

$$P(post_A) = P(loc_A | M, pri_A)$$

Technically, we may view loc_A as a uniform random variable over the spatial domain, i.e., the possible location of A when no knowledge is available.

The condition for privacy preservation is formally captured by Definition 1.

Definition 1. *Given a user A with privacy requirement G_A*[2]*, and M the set of messages exchanged by the proximity service protocol in which A is participating, A's privacy requirement is said to be* satisfied *if*

$$P(loc_A | M, pri_A, loc_A \in g_A) = P(loc_A | pri_A, loc_A \in g_A)$$

for all a-priori knowledge pri_A and all granules g_A of G_A.

The above definition requires that the location distribution of user A does not change due to the messages M, given the a-priori knowledge and the fact that A is located in g_A. Hence, a privacy violation occurs when the adversary acquires, through the analysis of the protocol messages, *more* information about the location of A than allowed by her privacy requirements, i.e., when the probability distribution of the position of A within the region defined by granule g_A changes with respect to pri_A.

In the extreme case a user needs complete location privacy with respect to an adversary, the G_A requirement is the top granularity $\top$, and hence g_A

[2]Depending on the considered adversary, this can be either G_A^{SP} or G_A^U.

is the entire spatial domain in the above definition. In this case, the definition requires $P(loc_A|M, pri_A) = P(loc_A|pri_A)$, i.e., $P(post_A) = P(pri_A)$ or *no new location information* for each user A. In the case G_A is not $\top$, the definition requires that with the additional knowledge of A being in a granule, the adversary cannot derive anything more (e.g., where within the granule) from the messages exchanged.

3.3 Privacy preserving techniques

In this section we present our techniques to preserve privacy in proximity based services.

3.3.1 *SP-Filtering*, *Hide&Seek* and *Hide&Crypt*

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 are considering, maintaining a shared secret between each pair of users 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 the solutions we present in this section, 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. These solutions do not require users to share a secret with each other, and hence they could be applicable to both “contact-list based” and “query driven” category of proximity services. In this section we illustrate the protocols and their formal properties. Section 3.3.2 presents the analysis of the protocols presented in this section.

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.

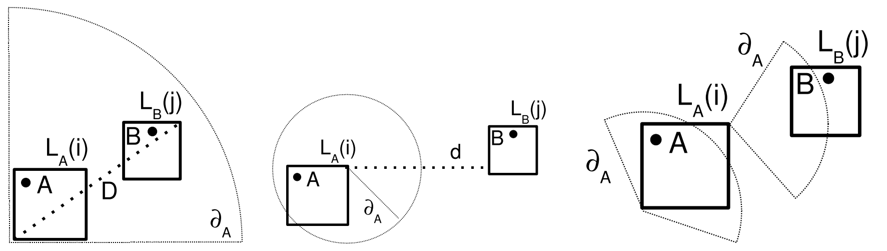
The idea of the algorithm is that when a user A performs a location update, instead of providing her exact location to the SP, she sends a generalized location that is computed as a function of G_A^U and the granule $G_A^{SP}(i)$ where A is located. More precisely, A sends to SP the location $L_A(i)$ 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 *mindist* and *maxdist* the minimum and maximum distance, respectively.

Given *mindist* and *maxdist*, the SP can try to answer whether B is in the proximity of A or not. Indeed, if *maxdist* $< \delta_A$, 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 3.1(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 *mindist* $> \delta_A$, then the SP can conclude that B is not in the proximity of A. Figure 3.1(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., $\textit{mindist} \leq \delta_A \leq \textit{maxdist}$), then the SP is not able to compute whether B is really in the proximity of A or not. See Figure 3.1(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 3.1: Regions L_A and L_B

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 3.1(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 $G^U_A(i')$ and $G^U_B(j')$ where $G^U_B(j')$ 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 we take this imprecision as one performance measure of privacy protection techniques, as it will be shown in Section 4.1.

Protocol 2 *Hide&Seek*

Prerequisites: A and B are running the *SP-Filtering* protocol. A is located in $G^U_A(i')$, B is located in $G^U_B(j')$.

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: $d' = mindist(G^U_A(i'), G^U_B(j'))$
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^U_A and G^U_B, respectively. In this case, B has more chances to be able to compute whether A is in the proximity, if granularities G^U_A and G^U_B are "finer-than" G^{SP}_A and G^{SP}_B, 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.

The *Hide&Crypt* protocol

The *Hide&Seek* solution 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 subprotocols: 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^U_B 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 [31]). A technical issue of this protocol is that if B knows the cardinality of S he can be able to infer some location information about A. For example, on a grid-based granularity, it can happen that the number of granules considered in proximity when the user is located at the center of a granule is different to the case in which the user is located close to the corner of a granule. 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^U_B 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 $sMax(G_B^U, \delta_A)$ that represents the maximum number of granules of G_B^U that intersect with any circle with radius δ_A. Note that $sMax(G_B^U, \delta_A)$ 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[3] 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 $E^*_{K_B}(j)$ using the key K_A and checks if the result is contained in $E^*_{K_B}(E^*_{K_A}(S))$ [4]. Encryption function E^* is such that $E^*_{K_A}(E^*_{K_B}(j)) \in E^*_{K_B}(E^*_{K_A}(S))$ 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.

3.3.2 Analysis of *SP-Filtering*, *Hide&Seek* and *Hide&Crypt* protocols

SP-Filtering

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

$$\forall g \in G_A^{SP}, h \in G_A^U \; g \subseteq h \text{ or } h \subseteq g \text{ or } h \cap g = \emptyset \tag{3.2}$$

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

[3]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 $i = j$.

[4] 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.

Protocol 3 *Hide&Crypt*

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

Protocol:

```
1:  $A$ receives "$B$ is possibly in proximity" from the SP.
2:  $A$ computes: $S' = \{j \in \mathbb{N}$ s.t. $G^U_B(j) \cap \mathcal{C} \neq \emptyset\}$
3:  $A$ computes: $S''$ as a set of $sMax(G^U_B, \delta_A) - |S'|$ 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
```

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 $G^U_A(j)$, there exists granule $G^{SP}_A(i)$ such that they intersect, and furthermore, if condition (3.2) is satisfied, it must be the case that one of two granules totally contains the other. A sufficient condition to satisfy condition (3.2) is that either G^{SP}_A is a finer than G^U_A or vice versa[5].

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.

[5]The *finer-than* relationship was defined for temporal granularities (see, among others, [8]) and it can be easily extended for spatial granularities. Basically, G^{SP}_A is a finer than G^U_A if the former is a finer partitioning of the space than the latter is.

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 receives 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 *mindist* and *maxdist* on the SP side). In Section 4.1 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.

Hide&Seek

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 $G^U_A(i')$ and*

$$S'_{out} = \{j' \in \mathbb{N} | mindist(G^U_A(i'), G^U_B(j')) \geq \delta_A\}$$

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 $Area^{out}_S = Area_{NoFilter} \cap \bigcup_{j' \in S'_{out}} G^U_B(j')$;*

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

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 3.2 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 3.2). 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 3.2) or $Area_S^{out}$ ($Area^{out}$, in Figure 3.2), respectively.

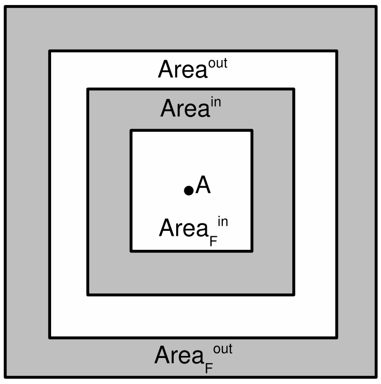


Figure 3.2: Possible locations of B

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 *mindist* has the same complexity of the computation of d'. Hence, using the class of spatial granularities we present in Section 4.1, 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 4.1 we evaluate the number of these messages in our experimental setting.

Hide&Crypt

Similarly to the *Hide&Seek* protocol, also in *Hide&Crypt* the SP acquires no 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.

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 $Area_{\mathcal{C}}^{in} = Area_{NoFilter} \cap (\bigcup_{j' \in S'} G_B^U(j'))$;*

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 $Area_{\mathcal{C}}^{out} = Area_{NoFilter} \cap (Area_{\mathcal{C}}^{in})^C$;*

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 3.2. 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 4.1, $Area_C^{in}$ 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 4.1 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 $sMax(G_B^U, \delta_A)$ 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 $sMax(G_B^U, \delta_A)$. 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 $sMax(G_B^U, \delta_A)$.

3.3.3 *Longitude*

One problem with *Hide&Crypt* is that it can lead to high communication and computation cost for the client when a user requires total privacy with respect to the SP. Indeed, in this case, the two-party subprotocol of *Hide&Seek* and *Hide&Crypt* has to run for every proximity request, as the SP would be totally excluded from the proximity computation. An empirical analysis of the costs of this drawback is in Section 4.1.3.

Aiming at reducing system costs, we developed another protocol called *Longitude*. This protocol, presented in this section, does not require users to do any two-party computation, and achieves total privacy with respect to the SP i.e. the G_A^{SP} is set to $\top$, under the assumption that the adversary has no a-priori knowledge about users' locations. While this assumption can be unrealistic when considering the entire world as spatial domain, it can be reasonable when the service is limited to a smaller area, like a city, and the adversary does not know the distribution of the users over this area. Differently from the other solutions, only grid-based granularities having cells of the same size can be chosen as privacy preference with respect to the other users. We then use the notation Gr_A^U for the grid selected as privacy preference by user A.

The protocol

The main steps of the *Longitude* protocol are the following: each time A wants to check whether B is in proximity, A runs the *encryptLocation* procedure to encrypt the cell c_A of Gr_A^U where A is located and sends it to the SP. Then, the SP sends a message to B requiring a location update. B runs the *encryptLocation* procedure to encrypt, using the same key as A, the cell c_B of Gr_B^U where B is located and sends the result to the SP. Since the SP does not know the key used to encrypt the two cells, it is not able to acquire any location information about A and B. However, the encryption function is designed in such a way that the SP, upon receiving a request from A and an answer from B with cells encrypted with the same key, can derive some information about the proximity relationship between A and B. This is obtained through the *computeProximity* procedure. The SP sends this proximity related information in the form of a boolean value to the requester A who can compute, through the procedure *getResult*, whether B is in proximity or not.

The encryption function is such that, if A sends her location cell to the SP using the same encryption key in different instants and while being in different cells, and the SP is aware of this, he can possibly learn some information about the movement of A, and hence about her location. For this reason, A changes the encryption key each time she communicates her location cell to the SP. The following is a simple protocol to achieve this, but many optimizations and different solutions can be devised without affecting the main results of *Longitude*. We assume A and B share a secret K; the actual key used to encrypt the location information is obtained through a pseudo-random number generator (PRNG) with seed K. To compute this shared secret A and B can, for example, employ a key agreement protocol, like Diffie-Hellman. An index i is locally stored by A, it is incremented at each proximity request, and it is used to select the i-th generated key for the current request. Its value is also included in the proximity request, since the locations of other buddies will need to be encrypted with a key selected according to i.

The *encryptLocation* procedure

The procedure is schematically illustrated as Procedure 4. It is used to issue requests for proximity as well as to send responses to location requests by the SP. The inputs are the location l of the user running the procedure, the grid Gr^U chosen to protect the privacy of the user, and the seed K, the parameter *lastIndex* that takes the value of i (i.e., the index of the last key generated by the PRNG by the user running the procedure), and the optional parameter *newIndex* that is only defined when the procedure is used to respond to a proximity request issued by another buddy; In this case, the value of *newIndex* is the index of the key used by the issuing buddy. If the procedure is used to issue a request for proximity, the index is incremented. If the procedure is used to send a response to a location request, it first checks if the index used by the buddy issuing the request has ever been used. If this is the case, using the same index again could compromise the user's privacy and the procedure simply terminates, hence ignoring the request incoming from the SP. Otherwise, the key with this index is generated with the PRNG.

The next steps consist in assigning to variable k_i the value of the i-th number generated by the PRNG with seed K and in computing the cell c of the grid Gr^U where the user running the procedure is currently located. This cell is then encrypted using the encryption function E, described in the following, and the key k_i. Finally, the result is sent to the SP together with the value of i and the value of i is stored so that it can be used in the next run of the procedure.

Procedure 4 *encryptLocation*

Input: a location l, a grid Gr^U, the seed K, the value *lastIndex*, the optional value *newIndex*.

Procedure:

```
1: if (issuing request for proximity) then
2:     $i = lastIndex + 1$
3: else {responding to a proximity request}
4:     if ($newIndex \leq lastIndex$) then return
5:     $i = newIndex$
6: end if
7: $k_i$ is the $i$-th number generated by the PRNG with seed $K$
8: $c$ is the cell of $Gr^U$ that contains the location $l$
9: $c' = E_{k_i}(c)$
10: send $\langle i, c' \rangle$ to the SP.
11: store $i$ {for the next execution}
```

Before describing the encryption function, we first introduce some notation. In our approach we assume that users are moving in a bi-dimensional space W which consists in a rectangular grid of $size_x \times size_y$ points. For each point $p \in W$, we denote with p_x and p_y the projection of p on the x and y axis, respectively.

The encryption function E we propose is based on a "modular translation". The idea is to apply, to each point of c, a translation followed by a modulus operation in such a way that no point is moved outside W. For example, if a point is moved by the translation right above the top boundary of W, the modulus operation moves it right above the bottom boundary of W and hence still within W (see Figure 3.3(a)).

The translation shift value is represented by $\alpha = \langle \alpha_x, \alpha_y \rangle$ which is computed from the key k_i as follows: $\alpha_x = k_i \mod size_x$, $\alpha_y = k_i \mod size_y$.

The encryption function E_{k_i} is then specified as:

$$E_{k_i}(c_A) = \bigcup_{p \in c_A} \langle (p_x + \alpha_x) \mod size_x, (p_y + \alpha_y) \mod size_y \rangle$$

In practice, $c'_A = E_{k_i}(c_A)$ is computed by applying a transformation to each point of c_A. On the x axis, the transformation consists in shifting the point by α_x and then in applying the module $size_x$. On the y axis the transformation is analogous. It is worth noting that, depending on α and c_A, $E_{k_i}(c_A)$ could be a set of contiguous points (see Figure 3.3(b)) as well as a set of non-contiguous points (see Figure 3.3(c))

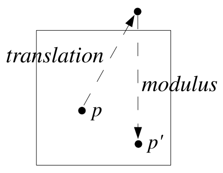
The *computeProximity* procedure

The *computeProximity* procedure (see Procedure 5) is run by the SP when it receives two locations encrypted with the same key.

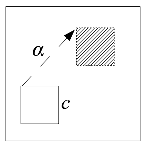
The first step of the procedure consists in computing the "minimum modular distance" between c'_A and c'_B as follows:

$$mmd(c'_A, c'_B) = \min_{p \in c'_A, p' \in c'_B} moddist(p, p')$$

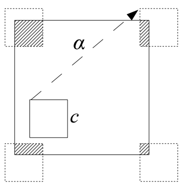
where *moddist* is the *modular distance* between p and p'. Intuitively, the modular distance is the Euclidean distance computed as if W were "circular"



(a) Translation of a point



(b) Contiguous points



(c) Non-contiguous points

Figure 3.3: Examples of modular translations of a point and of a cell. $E_{k_i}(c)$ represented in gray.

Procedure 5 *computeProximity*

Input: $\langle i, c'_A \rangle$ received from A, which issued a proximity request, and $\langle i, c'_B \rangle$ received from B, which is responding to the request.

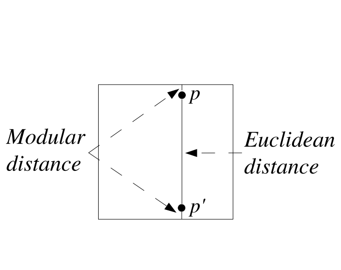
Procedure:

```
1: $dist = mmd(c'_A, c'_B)$ {minimum modular distance}
2: send $A$ the boolean value $(dist \leq \delta_A)$
```

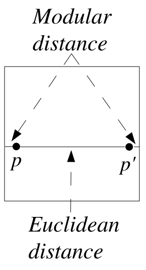
on both axes. For example, consider two points p and p' (see Figure 3.4(a)), with the same horizontal position such that p is close to the top boundary of W and p' is close to the bottom boundary. The Euclidean distance of the two points is about $size_y$ while the modular distance is close to zero. The same holds on the other axis (see Figure 3.4(b)) and also on the combination on the two axis (see Figure 3.4(c)). Formally, given two points p and p', $\Delta_x = |p_x - p'_x|$ and $\Delta_y = |p_y - p'_y|$, the modular distance is defined as:

$$moddist(p,p') = min(\sqrt{(\Delta_x)^2 + (\Delta_y)^2}, \sqrt{(size_x - \Delta_x)^2 + (\Delta_y)^2},$$
$$\sqrt{(\Delta_x)^2 + (size_y - \Delta_y)^2}\sqrt{(size_x - \Delta_x)^2 + (size_y - \Delta_y)^2}\ )$$

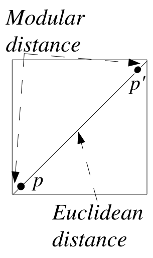
The final step of *computeProximity* consists in comparing the minimum modular distance between c'_A and c'_B with δ_A, the proximity threshold of A. The boolean value of this comparison is sent to A.



(a) On the vertical axis



(b) On the horizontal axis



(c) On both axis

Figure 3.4: Examples of modular distance

The *getResult* procedure

In the *getResult* procedure (see Procedure 6) user A, which is running the procedure, decides whether B is in proximity or not. This result is obtained considering the boolean value received from the SP and the relative position of the cell c_A, where A is located, with respect to a region called "certainty region" of A. This region, denoted by CR_A, is the set of points of W that are farther than δ_A from the boundaries of W (see Figure 3.5).

The correctness of the result computed by the *getResult*, as well as the approximation introduced by the protocol and its safety are discussed in Section 3.3.4.

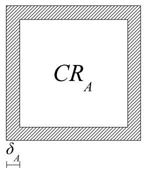


Figure 3.5: Example of the certainty region CR_A

3.3.4 Analysis of the *Longitude* protocol

In this section we first discuss the safety of the *Longitude* protocol with respect to privacy protection and then we analyze its correctness and the

Procedure 6 *getResult*

Input: The boolean value *res* received from the SP, the cell c where the user running the procedure is located, the certainty region CR of the user running the protocol, the user B which responded to the proximity request.

Procedure:

```
1: if (res = True AND $c \subseteq CR$) then
2:     $B$ is in proximity
3: else
4:     $B$ is not in proximity
5: end if
```

approximation it introduces. We first introduce a formal proposition that will be used in the protocol analysis.

Proposition 4. *Given two cells c_A and c_B and a key k_i, the encryption function E is such that:*

$$mmd(c_A, c_B) = mmd(E_{k_i}(c_A), E_{k_i}(c_B))$$

Proposition 4 intuitively states that the encryption function E presented in Section 3.3.3 does not alter the minimum modular distance between c_A and c_B.

Safety

We first analyze the privacy that the *Longitude* protocol provides to a user with respect to another buddy and with respect to the SP under the assumptions that the SP and the buddies do not collude. Then, we discuss the location information that is disclosed in case collusion occurs.

During the execution of the protocol the only message that A receives containing information related to the location of a buddy B is the boolean value received from the SP as a response to A's request for the proximity of B. When A receives **True** from the SP (i.e., $mmd(c'_A, c'_B) \leq \delta_A$), due to Proposition 4, A learns that B is located in a cell c_B of G_B such that

$mmd(c_A, c_B) \leq \delta_A$. Since A knows c_A and G_B, she can compute the set of cells where B is possibly located. Formally, A cannot exclude B is located in any cell c of G_B such that $mmd(c_A, c) \leq \delta_A$. Analogously, when A receives **False** from the SP A cannot exclude B is located in any cell c of G_B such that $mmd(c_A, c) > \delta_A$. Consequently, the minimum privacy requirement of B with respect to A are guaranteed.

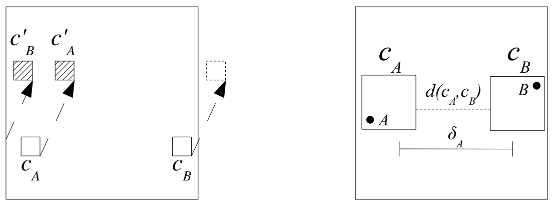
For what concerns the privacy protection with respect to the SP, it is easily seen from the protocol that the SP only learns the minimum modular distance between c'_A and c'_B and hence, due to Proposition 4, the minimum modular distance between c_A and c_B. This knowledge does not disclose any explicit location information about A and B. It should be noted, however, that the SP learn a relative information about the location of A and B, that is a value related to the distance among their cells. Hence, in case the adversary has an a-priori knowledge of users location, it may be able to exclude some portion of the spatial domain as a possible location of the users. Therefore, Definition 1 is not satisfied by *Longitude* with respect to the SP.

We now turn to consider collusion. If a user B considers all buddies as untrusted, he will probably use the same (coarse) grid for everybody. In this case, even if buddies collude, the minimum privacy requirements are guaranteed. However, if user B has different degrees of trust on her buddies (hence using different grids), and these buddies collude, the location of B could be discovered with high precision by intersecting the location information about B acquired by the colluding buddies. This can be easily avoided by imposing the following constraint on the relationship among the spatial grids used as privacy preferences: *cells from different grids never partially overlap*. In this case, the location of B is never disclosed with a precision higher than the finest grid among those defined for the colluding buddies. In other words, the minimum privacy requirement defined for the most trusted buddy among the colluding ones is guaranteed. Collusion with the SP is not likely in the service model we are considering, since the SP is considered untrusted, while a certain degree of trust is assumed among the participating buddies that indeed share a secret. In the worst case in which the trust model is broken by a buddy A of B colluding with the SP, the SP can obtain and share with A the cell c_B where B is located each time B sends this information encrypted with the secret seed K shared with A. Note that the minimum privacy requirement with respect to A is guaranteed, and that the SP can only obtain the same location information about B available to A.

Service Precision

We now discuss the correctness of *Longitude* in terms of the service precision it provides. If A receives **False** from the SP then, according to the *computeProximity* procedure, $mmd(c'_A, c'_B) > \delta_A$. Due to Proposition 4, this means that $mmd(c_A, c_B) > \delta_A$. Since $mmd(c_A, c_B)$ is a lower bound to the real distance between A and B, it is guaranteed that B is not in proximity of A. Vice versa, if A receives **True**, it is not possible for A to conclude that B is in proximity, since two forms of approximation are introduced. We now explain the reason for these approximations, and our choice for the conditions under which the protocol declares B's proximity.

One form of approximation, which we call the *modular-shift error* is due to the fact that the encryption function does not preserve the distance. Indeed, as shown in Figure 3.6(a), it can happen that, while c'_A is close to c'_B, c_A is far from c_B. This would imply that, when the SP sends **True** to A (i.e., $mmd(c'_A, c'_B) \leq \delta_A$) A does not actually know whether B is in proximity or not. However, it can be easily seen that when c_A is in the certainty region CR_A, $mmd(c_A, c_B) \leq \delta_A$ implies that $mindist(c_A, c_B) \leq \delta_A$. In this case A can exclude the *modular-shift error*. Consequently, A knows that $mindist(c_A, c_B) \leq \delta_A$ and considers B as in proximity whenever **True** is returned by the SP, and c_A is contained in CR_A (lines 1-2 of the *getResult* procedure). If **True** is returned but c_A is not contained in CR_A, then A cannot conclude that B is in proximity. As we shall see in our experimental results, this case is very rare and, as a practical and efficient solution, procedure *getResult* returns in this particular case B as not being in proximity. Clearly, this leads to some possible false negative responses. A technical solution to avoid this approximation at some extra cost is to apply a P2P protocol between A and B, whenever this case arises [36].



(a) *modular-shift* error (b) Cell approximation

Figure 3.6: Two forms of approximation introduced by the *Longitude* Protocol.

The second form of approximation, which we call *cell approximation*, is due to the fact that B may not be in proximity of A even if $mindist(c_A, c_B) \leq \delta_A$. Figure 3.6(b) shows an example of this situation. The consequence of *cell approximation* is that, even if A knows that $mindist(c_A, c_B) \leq \delta_A$, she cannot be sure whether $d(loc(A), loc(B)) \leq \delta_A$. Nevertheless, in this case A assumes B to be in proximity. This can lead to some false positive cases. In our experimental evaluation we show that for many practically useful grids G_A and G_B, *cell approximation* only minimally affects quality of service.

3.3.5 *C-Hide&Seek* and *C-Hide&Hash*

In this section we present two protocols to preserve location privacy in proximity-based services called *C-Hide&Seek* and *C-Hide&Hash*. Like *Longitude*, these solutions are completely centralized, in order to reduce computation and communication costs for the users, especially with respect to the *Hide&Crypt* when conservative privacy preferences with respect to the

SP are required i.e. the G_A^{SP} is set to $\top$. Differently from *Longitude*, these protocols can guarantee total privacy with respect to the SP even if the SP has a-priori knowledge about the users location, and the users can choose arbitrary granularity G_A^U as privacy preferences, instead of grids.

In order to ensure user's privacy, the two protocols adopt symmetric encryption techniques. In the following, we assume that each user A has a key K_A that is shared with all of her buddies and is kept secret to everybody else. Hence, each user A knows her own key K_A and one key K_B for each buddy B. Since we are considering a contact-list-based service, this key exchange is assumed to be performed with any secure method before running our protocols. An example of key exchanging protocol is provided in Section 4.2.

For the sake of presentation, we decompose each protocol into two parts: the *location update* sub-protocol is used by a user to provide her location information, while the *proximity request* sub-protocol is used by a user to compute the proximity of her buddies. The location update sub-protocol is almost the same in both of our proposed solutions. What really distinguishes *C-Hide&Seek* and *C-Hide&Hash* is the proximity request sub-protocol, and these sub-protocols are described later in this section. We conclude this section with a discussion about possible technical extensions.

The location update sub-protocol

The location update sub-protocol is run by a user to provide location information to the SP. In particular, it defines how a user A provides to the SP the encrypted index of the granule of G_A^U where she is located.

Before describing the sub-protocol, we first discuss when it should be run. Consider the following naive policy: a user A updates her location only when she crosses the boundary between two granules of G_A^U, reporting the index of the new granule. It is easily seen that, independently from how the location update is performed, each time this message is received, the adversary learns that A is very close to the border between two granules, excluding many other locations, and hence violating the privacy requirements. Intuitively, the problem of the above policy is that the probability that a location update is performed at a given time depends on the location from where the message is sent.

The solution we propose is the following: time is partitioned into *update intervals* and an approximate synchronization on these intervals among the participating nodes is assumed.[6] Each update interval has the same duration T and is identified by an index. Each user has a value t in $[0, T)$ and sends exactly one location update during each update interval after that time t elapses from the beginning of the interval (see Figure 3.7). It is easily seen that, by using this update policy, the location updates are issued independently from the location of the users.

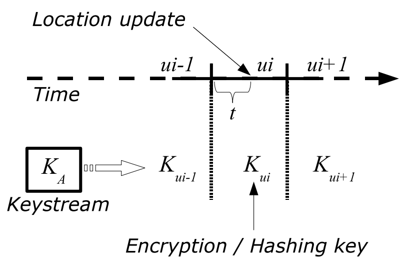


Figure 3.7: Location update policy and generation of single-use keys.

We now describe how the location update sub-protocol works. User A first computes the index i of the granule of G^U_A where she is located. Then, A encrypts i using a slightly different technique in the two proposed solutions. In the *C-Hide&Seek* protocol a symmetric encryption function E is applied, while in the *C-Hide&Hash* protocol a hashing function H is used. When applying the hashing function H, in order to prevent brute-force attacks, a secret key is used as a “salt”, i.e., a secret key is concatenated to i, and the resulting value is given as input to H. In the following, we refer to this salt

[6]In our current implementation, all the messages sent from the SP to the users contain the timestamp of the SP, allowing clients to synchronize their clocks using a Lamport-style algorithm. The overhead due to this solution is negligible. Other forms of global clock synchronization could also be used as, e.g., using GPS devices.

as the “key” used to hash i, and we denote with $H_K(i)$ the hashing of the value i with key K.

The safety of the protocols depends on the fact that the key used to encrypt or hash i is changed at every use. At the same time, we need the key to be shared by a user with all of her buddies. While other techniques can be adopted to achieve this result, our solution is the following: the key K_A that A shares with all of her buddies is used to initialize a keystream. When user A issues a location update, she computes the key K^{ui} as the ui-th value of this keystream, where ui is the index of the current update interval (see Figure 3.7). Since each user issues a single location update during each time interval, this solution ensures that every message is encrypted or hashed with a different key. Finally, A sends to the SP the message $\langle A, ui, E_{K^{ui}}(i)\rangle$ if running *C-Hide&Seek*, and $\langle A, ui, H_{K^{ui}}(i)\rangle$ if running *C-Hide&Hash*. The SP stores this information as the last known encrypted location for A. Figure 3.8 shows the message sent from A to the SP by the *C-Hide&Seek* protocol.

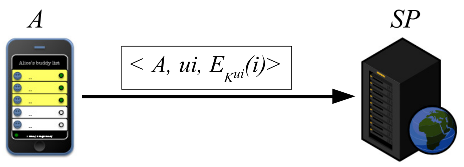


Figure 3.8: Location update sub-protocol in *C-Hide&Seek*.

Proximity request with *C-Hide&Seek*

The proximity request sub-protocol is run by a user that wants to discover which of her buddies are in proximity. In the *C-Hide&Seek* protocol, this sub-protocol works as follows: When A wants to discover which buddies are in proximity, she sends a request to the SP. The SP replies with a message containing the last known encrypted location of each buddy of A. That is, for each buddy B, A receives a tuple $\langle B, ui, E_{K^{ui}}(i)\rangle$. Since A knows K_B and the index ui is in the message, she can compute the value K^{ui} used by

B to encrypt his location, and hence she can decrypt $E_{K^{ui}}(i)$. Finally, since A also knows G^U_B, by using i, she obtains the granule $g_B = G^U_B(i)$ where B is located. A can then compute the distance between her exact location and g_B, and compare it with δ_A, finally determining the proximity. Figure 3.9 shows a graphical representation of the sub-protocol.

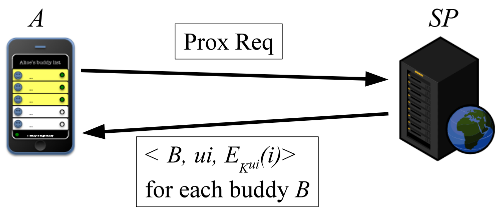


Figure 3.9: Proximity request sub-protocol in *C-Hide&Seek*.

Note that we are now considering the proximity between a point and a region. In this section, we consider that a point and a region are in proximity, with respect to a distance threshold, if the *minimum* distance between the two objects is less than the threshold. Since, in our protocol, the region represents the area where a user B is possibly located, this interpretation of proximity means that there is a possibility for users A and B to actually be in proximity. The same *minimum* distance interpretation has been used in related work on privacy-aware proximity computation. Alternative interpretations and their effects are discussed in Section 3.3.6.

The *C-Hide&Seek* protocol provides a simple and efficient solution that, as will be shown in Section 3.3.6, completely hides the location of the users to the SP, and that also guarantees the privacy requirements with respect to the buddies. However, it reveals exactly the maximum tolerable amount of location information (g_B for user B) to any buddy issuing a proximity request. Even if their privacy requirements are guaranteed, users would probably prefer to disclose as little information as possible about their location when not strictly needed. For example, is there an alternative solution that does not reveal to a user A the granule information of a buddy B if he is not in proximity?

In the next section we present the *C-Hide&Hash* protocol that provide such a solution and, in general, ensures a higher level of privacy. This is achieved at the cost of higher computation and communication costs, as explained in Section 3.3.6.

Proximity request in *C-Hide&Hash*

The *C-Hide&Hash* protocol has two main differences with respect to *C-Hide&Seek*. The first difference is that a hash function H is used during the location update, instead of the encryption function. This is due to the requirement in this protocol to avoid revealing the relationship between two plaintext values (the granule indexes) by observing the relationship among the corresponding encrypted values (see Section 3.3.6 for a more detailed explanation). Since in this protocol we do not need to decrypt the result of the function, but we only need to check for equality of encrypted values, hashing can be used. As specified in Section 3.3.5, each location update in *C-Hide&Hash* from user A to the SP is a message containing the tuple $\langle A, ui, H_{K^{ui}}(i)\rangle$.

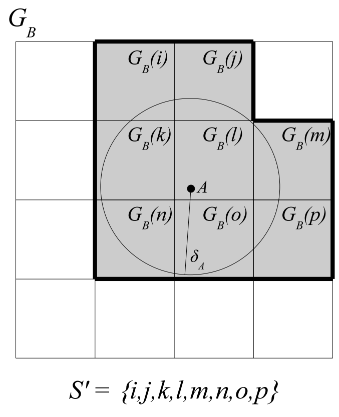


Figure 3.10: Computation of granules of G_B^U considered in proximity by A

The second and main difference with respect to *C-Hide&Seek* is the computation of the proximity request sub-protocol. The intuition is that when A issues a proximity request, she computes, for each of her buddies B, the set of indexes of granules of G^U_B such that, if B is located in any granule of the set, then B is in proximity (see Figure 3.10). Then, if B provides the granule in which he is located, it is possible to reduce the proximity problem to the set-inclusion problem, by checking if that granule is included in the set computed by A. We want to do this set inclusion without revealing to A which of the candidate granules actually matched the granule of B.

More precisely, the computation of a proximity request in the *C-Hide&Hash* protocol works as follows. When a user A issues a proximity request, she starts a two-party set inclusion protocol with the SP. The protocol is a secure computation, and consequently the SP does not learn whether A is in proximity with her buddies, and A only learns, for each of her buddies B, whether B is in proximity or not, without learning in which granule B is located. The secure computation exploits a commutative encryption function E^*. In addition to the keys used in the *C-Hide&Seek* protocol, at each proximity request, the requesting user and the SP each generates a random key that is not shared with anyone else. We denote these keys K_1 for user A and K_2 for the SP.

The proximity request sub-protocol is divided into three steps, whose pseudo-code is illustrated in Protocol 7. In Step (i), user A computes, for each buddy B, the set S' of indexes of granules of G^U_B such that, if B is located in one of these granules, then B is in proximity. More formally, A computes the set of indexes i such that the minimum distance *mindist* between the location of A and $G^U_B(i)$ is less than or equal to δ_A. Then, in order to hide the cardinality of S', A creates a new set S by adding to S' some non-valid randomly chosen indexes (e.g., negative numbers). This is done to increase the cardinality of S without affecting the result of the computation. The cardinality of S is increased so that it is as large as the number $sMax(G^U_B, \delta_A)$, already introduced in Section 3.3.1, that represents the maximum number of granules of G^U_B that intersect with any circle with radius δ_A. Note that $sMax(G^U_B, \delta_A)$ can be computed off-line since its values depend only on G^U_B and δ_A. In the following, when no confusion arises, we

Protocol 7 *C-Hide&Hash*: proximity request

Input: User A knows, the last completed update interval, and the proximity threshold δ_A. Also, for each of her buddy B, A knows the granularity G^U_B, the key K_B and the value of $sMax(G^U_B, \delta_A)$.

Protocol:

(i) Client request from A

```
1: $proxReq = \emptyset$
2: generate a random key $K_1$
3: for each buddy $B$ of $A$ do
4:     $S' = \{j \in \mathbb{N}$ s.t. $mindist(loc_A, G^U_B(j)) \leq \delta_A\}$
5:     $S''$ = a set of $sMax(G^U_B, \delta_A) - |S'|$ non-valid random indexes.
6:     $S = S' \cup S''$
7:     $K^{ui}$ is the $ui$-th value of the keystream initialized with $K_B$
8:     $ES = \bigcup_{i \in S} E^*_{K_1}(H_{K^{ui}}(i))$
9:     insert $\langle B, ui, ES \rangle$ in $proxReq$
10: end for
11: $A$ sends $proxReq$ to the SP
```

(ii) SP response

```
1: $proxResp = \emptyset$
2: generate a random key $K_2$
3: for each $\langle B, ui, ES \rangle$ in $proxReq$ do
4:     $ES' = \bigcup_{e \in ES} E^*_{K_2}(e)$
5:     retrieve $\langle B, ui, h_B \rangle$ updated by $B$ at update interval $ui$
6:     $h' = E^*_{K_2}(h_B)$
7:     insert $\langle B, ES', h' \rangle$ in $proxResp$
8: end for
9: SP sends $proxResp$ to $A$
```

(iii) Client result computation

```
1: for each $\langle B, ES', h' \rangle$ in $proxResp$ do
2:     $h'' = E^*_{K_1}(h')$
3:     if $h'' \in ES'$ then
4:         $A$ returns "$B$ is in proximity"
5:     else
6:         $A$ returns "$B$ is not in proximity"
7:     end if
8: end for
```

use $sMax$ as a short notation for $sMax(G^U_B, \delta_A)$. In Line 8, each element of S is first hashed using the key K^{ui}, which is obtained as the ui-th value generated by the keystream initialized with K_B. In this case ui is the index of the update interval preceding the current one. Then, the result is encrypted, using the commutative encryption function E^* and key K_1 that is randomly generated. The element composed by the set ES computed in Line 8, B, and ui is then added to the set *proxReq*.

Once the operations in Lines 4 to 9 are executed for each buddy B, the set *proxReq* is sent to the SP.

Upon receiving *proxReq*, the SP starts Step (ii). For each tuple $\langle B, ui, ES\rangle$ in *proxReq*, the SP encrypts with the E^* function each element of ES using key K_2, which is randomly generated. The result is the set ES'. Then, it retrieves the tuple $\langle B, ui, h_B\rangle$ updated by B at the update interval ui. In this tuple, h_B is the value of the index of the granule of G^U_B where B is located, hashed with the key K^{ui}. Since ui is the update interval preceding the current one, our location update policy assures that a location update with update interval ui has already been issued by every buddy B. Finally, the SP encrypts h_B with the commutative encryption function E^* using key K_2. The resulting value h' is added, together with B and ES', to the set $proxResp$.

Once the computations at Lines 4 to 7 are executed for each buddy B, the set *proxResp* is sent to A.

In Step (iii), given the message *proxResp* received from the SP, A computes the proximity of her buddies. For each tuple $\langle B, ES', h'\rangle$, A obtains h'' as the encryption of h' with E^* and the key K_1 and checks if the result is in ES'. If this is the case, then B is in proximity, otherwise he is not.

More formally, $h'' \in ES'$ if and only if the granule of G^U_B with index i containing B is in S', that is equivalent to B being in proximity. Indeed, for each buddy B, we recall that:

$$h'' = E^*_{K_1}(E^*_{K_2}(h_B))$$

and

$$ES' = \bigcup_{i \in S}(E^*_{K_2}(E^*_{K_1}(H_{K^{ui}}(i)))$$

Consequently, due to the commutative property of the encryption function,

$h'' \in ES'$ if and only if

$$h_B \in \bigcup_{i \in S} H_{K^{ui}}(i)$$

Since h_B and the elements of the set are hashed using the same key K^{ui}, h_B is in the set if and only if $i \in S$. Since $S = S' \cup S''$ and $i \notin S''$ (because S'' contains invalid integers only while i is a valid integer) then $i \in S$ if and only if $i \in S'$. By definition of S', this implies that B is in proximity.

Figure 3.11 shows the messages exchanged during the proximity request sub-protocol of *C-Hide&Hash*.

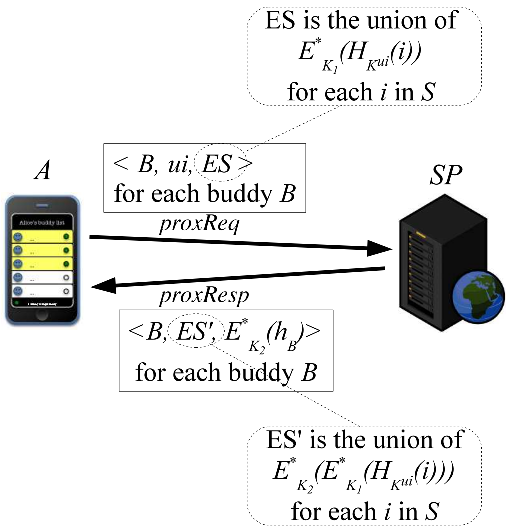


Figure 3.11: Proximity request sub-protocol in *C-Hide&Hash*.

Contrasting velocity attacks and other background knowledge

It is easily seen that our location update policy, based on fixed length update intervals, makes the probability that a location update is issued independent from the location from where it is issued. This is an important property used in Section 3.3.6, together with others, to prove the safety of our solutions under the adversary models we consider.

Clearly, if the adversary had arbitrary background knowledge, there would not be any technique that could guarantee privacy. However, it is interesting to consider some other forms of knowledge that the adversary could use. With respect to previous proposals, our defenses are resistant to an important type of background knowledge: a-priori distribution of the users' locations. There are, however, other types of knowledge that may be interesting to consider as, for example, the *time-dependent* a-priori location knowledge. This includes knowledge on the *relative* position of users at a certain time, as well as a-priori probability of user movements. With this kind of knowledge it is also possible to perform attacks based on the velocity of users. Consider Example 5.

Example 5. *User A sends two location updates in two consecutive update intervals i and j from granule g_1 and g_2, respectively. Her buddy B issues a proximity request in each update interval and discovers the granule where A is located. So far, no privacy violation occurred for A. However, if B knows that A moves at most with velocity v, then he can exclude that A is located in some locations l of g_2. Indeed, B knows that the temporal distance between the two location updates of A is equal to the length T of the update period. Now B can exclude that A is located in any location l of g_2 such that the time required to move from any point of g_1 to l with velocity v is larger than T. Hence B violates the privacy requirement of A.*

The problem in Example 5 arises when the adversary knows the maximum velocity of a user. Velocity-based attacks have been recently considered independently from proximity services [19], but the application of those solutions in our framework would lead to the release of some location information to the SP. In the following we show how to adapt our location update policy to provide protection preserving our privacy properties in the specific case in which the adversary knows the maximum velocity v of a user.

Let $tMax(g_1, g_2)$ be the maximum time required to move at velocity v from each point of granule g_1 to each point of granule g_2. The problem of Example 5 arises when the temporal distance between two location updates issued from two different granules g_1 and g_2 is less then $tMax(g_1, g_2)$. The problem can be solved by imposing that A, after entering g_2, randomly reports g_1 or g_2 as the granule where she is located until time $tMax(g_1, g_2)$ elapses from the last location update in g_1. This solution is a form of temporal generalization as it adds uncertainty to the adversary, about when the user crosses the border between g_1 and g_2. More specifically, the adversary is unable to identify the exact instant in which the user crossed the border in a time interval of length at least $tMax(g_1, g_2)$. Consequently, by definition of $tMax(g_1, g_2)$, the adversary cannot exclude that A moved from any point of g_1 to any point of g_2.

The extension of our defense techniques to other forms of background knowledge is one of the subjects for future work.

3.3.6 Analysis of *C-Hide&Seek* and *C-Hide&Hash* protocols

The main goal of our techniques is to guarantee the satisfaction of users' privacy requirements under the given adversary models. In the **Privacy** subsection, we prove that our two protocols have this property.

However, there are other important parameters to be considered in an evaluation and comparison among protocols that satisfy the privacy requirements. In general, the higher the privacy provided by the protocol, the better is for the users; since location privacy in our model is captured by the *size of the uncertainty region*, in the **Size of uncertainty region** subsection we consider this parameter.

A second parameter to be considered is *service precision.* The percentage of false positives and false negatives introduced by a specific protocol must be evaluated. This is considered in the **Service precision** subsection.

Last but not least, it is important to evaluate the overall *system cost*, including computation and communication, with a particular attention to client-side costs. This is considered in the **System costs** subsection.

The proofs of the formal results presented in this section are in Appendix A.

Privacy

We analyze the privacy provided by *C-Hide&Seek* and *C-Hide&Hash* considering the adversary models presented in Section 3.2 under the *no-collusion* assumption, i.e., assuming that the SP does not collude with the buddies and that the buddies do not collude among themselves. Then, we show the privacy guarantees provided by the two algorithms in the more general case of possibly colluding adversaries.

Satisfaction of privacy requirements We first analyze the *C-Hide&Seek* protocol. Since the private key K_A is only known to A and to the buddies of A, the SP is not able to decrypt the index of the granule where A is located. Analogously, the SP is not able to obtain location information about A's buddies and, in particular, does not obtain any information about the distance between A and her buddies.

We now state a formal property of the *C-Hide&Seek* that is used in the formal proof of the above observations.

Lemma 1. *The C-Hide&Seek protocol ensures that under any a-priori knowledge pri_A, the following two random variables are probabilistically independent: (1) The binary random variable $ur(A)$: an update/request is sent by user A, and (2) random variable loc_A, i.e., the location of A, of any distribution. Formally, we have*

$$P(ur(A)|loc_A, pri_A) = P(ur(A)|pri_A),$$

for any a-priori location knowledge pri_A and location random variable loc_A for user A.

Note that we are assuming discrete time and discrete location. A continuous case can be formalized and proved equally easily. Also, this lemma does not concern the type or content of a message sent by A, but just the fact that a message is sent by A.

Another property we use to prove our safety result is provided by the encryption algorithms, via the information theoretical notion of "perfect secrecy" [9]. Intuitively, perfect secrecy for an encryption algorithm means that given ciphertext c, each plaintext p has the same probability to be encrypted to c (posterior), with a randomly chosen key, as the probability of p to be used in the first place (prior). That is, $P(p|c) = P(p)$. Equivalently, given plaintext p, each ciphertext c has the same probability to be the encryption of p (posterior), with a randomly chosen key, as the probability of c to appear in the first place as ciphertext (prior). That is, $P(c|p) = P(c)$. Applied to our situation, when SP receives a message $\langle A, ui, E_{K^{ui}}(l)\rangle$, since K^{ui} is hidden from the SP and can be chosen arbitrarily, the probability that SP receives any other message of the form $\langle A, ui, E_{K^{ui}}(l')\rangle$ is the same.

Most of practical encryption algorithms do not have the theoretical perfect secrecy, but use computational hardness to achieve secrecy in the sense that it is computationally very hard (or impractical) to derive the plaintext from the ciphertext. Intuitively, $P(p|c) = P(p)$ holds because c does not yield any information about p. Therefore, we use the simplifying, practical assumption that the encryption methods we use do give us perfect secrecy.

The above perfect secrecy discussion applies to single messages. When dealing with multiple messages, correlation between plaintexts may reveal secrets when the same key is used. This is the classical scenario of repeated key use problem, and one solution to this problem is to use so-called one-use-pad or keystreams as we do in our proposed protocols. As each key is only used once, encrypted messages are independent to each other when perfect secrecy is assumed.

From the above discussion and assumptions, Lemma 2 follows. Since the lemma involves random variables on messages, we need to specify the *message space* for these variables. We consider the randomness of the messages to be on the encrypted part, while other parts are fixed. Formally, we call each sequence $\langle B, ui_1\rangle, \ldots, \langle B, ui_n\rangle$, where B is a user and ui_j is a time interval, a *(message set) type.* (Recall that a message is of the form $\langle B, ui, ES\rangle$.) The messages of the same type differ on the encrypted part of the messages and constitute a message space. When a generic message M

is mentioned, we assume it is a variable over all the messages with a specific type.

Lemma 2. *Given messages $M = M_1 \cup M_2$ issued in the C-Hide&Seek protocol, where $M_1 \cap M_2 = \emptyset$, we have*

$$P(M|loc_A, pri_A) = P(M_1|loc_A, pri_A) * P(M_2|loc_A, pri_A),$$

for all a-priori knowledge pri_A and location loc_A for user A.

With Lemma 1, perfect secrecy, and Lemma 2, we now show a main result, namely, the SP does not acquire any location information as a consequence of a location update or a proximity request using the *C-Hide&Seek* protocol. The following formal results implicitly refer to our adversary models that, in particular, assume that the SP has no background knowledge other than the protocol, the a-priori distribution, and the granularities.

Theorem 4. *Let A be a user issuing a sequence of location updates and proximity requests following the C-Hide&Seek protocol. Then, A's privacy requirement is satisfied with respect to the SP.*

We now turn to the location information acquired by the buddies. In the *C-Hide&Seek* protocol, a user A issuing a proximity request does not send any location information, hence her buddies, even if malicious, cannot violate her privacy requirements. When the same user runs the location update subprotocol in *C-Hide&Seek*, her buddies can only obtain the granule at the granularity G^U_A in which A is located. As a consequence, the privacy requirement of A is guaranteed. This is formally stated in Theorem 5.

Theorem 5. *Let A be a user issuing a sequence of location updates and proximity requests following the C-Hide&Seek protocol. Then, A's privacy requirement is satisfied with respect to each of A's buddies.*

We consider now the *C-Hide&Hash* protocol. Since K_A is only known to A and her buddies, the SP is not able to acquire the location information provided by A during a location update. This follows from Theorem 4.

The difference of the *C-Hide&Hash* from the *C-Hide&Seek* is that when A issues a proximity request in *C-Hide&Hash*, an encrypted message is sent to the SP. However, due to the property of the secure computation protocol in *C-Hide&Hash*, the only information that the SP acquires about the set provided by A is its cardinality. Actually, the cardinality of this set is always $sMax(G_B^U, \delta_A)$ that, by definition, depends only on δ_A and G_B^U, and not on the actual location of A or B. Consequently, the SP does not acquire any information about the location of A and B, including their distance. Theorem 6 formally states this property.

Theorem 6. *Let A be a user issuing a sequence of location updates and proximity requests following the C-Hide&Hash protocol. Then A's privacy requirement is satisfied with respect to the SP.*

Similarly to the *C-Hide&Seek* protocol, in *C-Hide&Hash* each buddy of A can only obtain location information derived from A's location update. It is worth noting that in the *C-Hide&Seek* protocol, each time B issues a proximity request, he obtains the granule of G_A^U where his buddy A is located. Differently, using the *C-Hide&Hash* protocol, B only gets to know whether the granule where A is located is one of those in S_A. This means that, if A is not in proximity, then B only learns that A is not in any of the granules of S_A. Otherwise, if A is in proximity, B learns that A is in one of the granules of S_A, without knowing exactly in which granule she is located. This is formally stated in Theorem 7.

Theorem 7. *Let A be a user issuing a sequence of location updates and proximity requests following the C-Hide&Hash protocol. Then, A's privacy requirement is satisfied with respect to each of A's buddies.*

In Section 4.1 we show that, on average, *C-Hide&Hash* provides more privacy with respect to the buddies than *C-Hide&Seek*, but at extra costs, making each protocol more adequate than the other based on user preferences and deployment modalities.

Privacy in case of possibly colluding adversaries We now consider the case in which our reference adversaries can collude, and we analyze the privacy guarantees of the *C-Hide&Hash* and *C-Hide&Seek* protocols in this scenario.

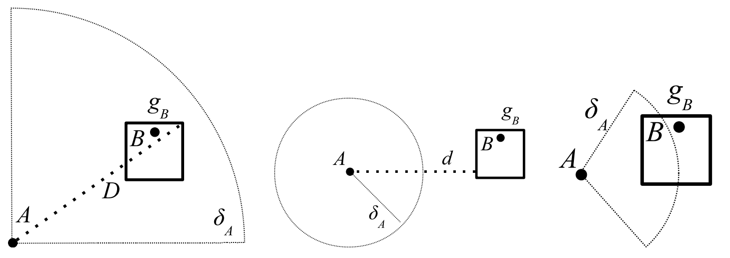
First, consider the case in which two buddies B and C collude to violate the privacy of a user A. The problem can be easily extended to consider more buddies. Let l_B be the set of possible locations of A obtained by B as a result of a proximity request. Let l_C be the analogous information acquired by C during the same update interval. Since B and C collude, they can derive that A is located in $l_B \cap l_C$. However, due to Theorem 7, given $G^U_A(i)$ the granule where A is located, it holds that $l_B \supseteq G^U_A(i)$ and $l_C \supseteq G^U_A(i)$ (recall that G^U_A is the privacy requirement of A with respect to the buddies). Consequently, $l_B \cap l_C \supseteq G^U_A(i)$ and hence the privacy requirement of A is guaranteed also in the case B and C collude.

Now, consider the case in which the SP colludes with one or more buddies. For example, if one of the buddies shares the secret key K_A with the SP, the SP can learn the granule where A is located. In this case, the privacy requirement of A with respect to the SP is not guaranteed. Nevertheless, even if the SP knows K_A, he cannot discover the location of A within the granule of $G^U_A(i)$ where A is located. This is because, by the definition of the two protocols, every message issued by A does not depend on the location of A within $G^U_A(i)$. Consequently, the privacy requirement with respect to the buddies is still guaranteed. This means that the lowest privacy requirement of the two colluding entities is preserved and this is the best that can be achieved in case of collusion.

Service precision

The techniques proposed in the literature as well as the techniques we propose in this dissertation, generalize the location of one of the two users to an area. When proximity is computed, the exact location of that user within the area is not known. Hence, proximity is evaluated as the distance between a point and a region.

Consider how it is possible to compute the proximity between a user A whose exact location is known and a user B whose location is only known to be in region. It is easily seen that if the maximum distance between the point and the region is less than the proximity threshold, then the two users are in proximity, independently from where B is located within the region. Figure 3.12(a) shows an example of this situation. On the contrary, if the minimum distance is larger than the distance threshold, then the two users are not in proximity. Figure 3.12(b) graphically shows that this happens when no point of the region containing B is in proximity of A. If none of the two cases above happen (i.e., the threshold distance is larger than the minimum distance and less than the maximum distance), we are in presence of an *uncertainty case*, in which it is not possible to compute whether the two users are in proximity without introducing some approximation in the result. For example, Figure 3.12(c) shows that if B is located close to the bottom left corner of the region then B is in the proximity of A, otherwise he is not.



(a) B is in proximity of A (b) B is not in proximity of A (c) B is *possibly* in proximity of A

Figure 3.12: Different cases of proximity between a point and a region

The choice we made in the presentation of our protocols is to consider two users as in proximity in the *uncertainty case*. The rational is that in this case it is not possible to exclude that the users are not in proximity. The other solutions presented in this chapter, as well as previous approaches

([53]) facing a similar issue have adopted the same semantics.

One drawback of this *minimum-distance* semantics is that it generates false positive results and this may be undesirable in some applications. Indeed, if user B is reported to be in proximity of A, then A may decide to contact B (e.g., through IM). This may be annoying for B, if he is not actually in proximity. Consider, for example, the case in which the location of B is reported at the granularity of a city: B is always reported as in proximity of A when A is in the same city, independently from the proximity threshold chosen by A.

An alternative semantics, that we name *maximum-distance* semantics, solves this problem. The idea is to consider two users as in proximity only when it is certain that they are actually in proximity. This happens when the maximum distance between their areas is less than the distance threshold. While this approach does not generate any false-positive, it does produce false-negatives. The two semantics above have a common drawback: in certain cases it happens that the probability of providing a false result is larger than the probability of providing a correct result. Consider the example depicted in Figure 3.13 in which the *minimum-distance* semantics is considered. User B is considered in proximity but the answer is wrong if B is located in the region colored in gray. Assuming a uniform distribution of B inside g_B, it is much more likely to have an incorrect result, rather than a correct one. An analogous problem can arise for the *maximum-distance* approach.

The percentage of false results can be minimized by considering user B as in proximity only when at least one half of the area is actually in proximity. The drawback of this *mostly-in-proximity* semantics is that it incurs in both false positive and false negative results.

Our protocols are designed so that it is very easy to change the current proximity semantics. Since this can be done client-side, without the need for changes server-side nor in the code other peers are running, the semantics can be potentially chosen through the user interface at any time.

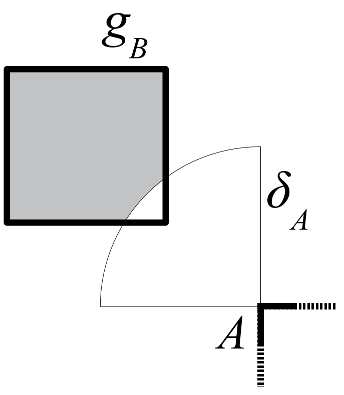


Figure 3.13: Approximation incurring with the *minimum-distance* semantics

We analytically measured the impact of the different semantics on the accuracy of our protocols by calculating the *expected precision* and the *expected recall*. The expected precision is defined as the probability that a buddy reported to be in proximity according to a given semantic is actually in proximity. Vice versa, the expected recall is defined as the probability that a buddy actually in proximity is reported to be in proximity according to a given semantic.

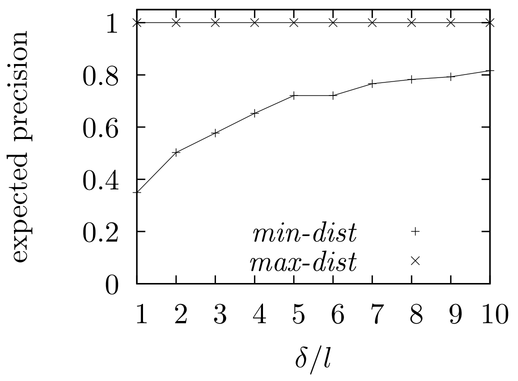


Figure 3.14: Expected precision

Figures 3.14 and 3.15 show the minimum expected precision and recall for the *minimum-distance* and the *maximum-distance* semantics. Both measures depend on the ratio between δ and the area of the granules in which

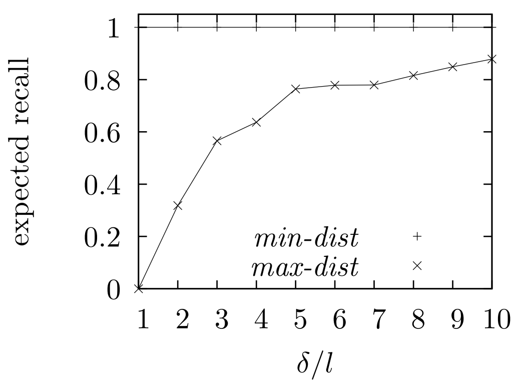


Figure 3.15: Expected recall

a user is considered in proximity. For this analysis we considered a grid-like granularity containing cells having edge of size l and we assume users are uniformly distributed. As can be observed in Figure 3.14, the *maximum-distance* semantic has always precision equal to 1. This is because all the buddies considered in proximity are always actually in proximity. The *minimum-distance* has precision of about 1/3 when the values of δ and l are equal, and this value grows logarithmically when δ is larger than l. The analysis of expected recall (Figure 3.15) shows that the *minimum-distance* has always recall equal to 1. This is because if a buddy is actually in proximity, it is always reported in proximity using this semantic. The *maximum-distance* semantic, on the contrary, has a minimum expected recall equal to 0 when δ and l are equal. This is because, with this parameters, it can happen that no cells of size l are fully contained in a circle having radius δ. However, the recall of the *maximum-distance* grows more rapidly than the precision of the *minimum-distance*.

Size of uncertainty regions

As already discussed in Section 3.3.6, our protocols are proven to always guarantee the privacy requirement with respect to the buddies. However, the main difference between our two protocols consists in the fact that *C*-

Hide&Hash can provide additional privacy with respect to one buddy. For example, if a user A issues a proximity request using *C-Hide&Hash*, and a buddy B is reported as being not in proximity, A only learns that B is not located in any of the granules considered in proximity (i.e., the ones included in S). The resulting uncertainty region of B, in this case, is equal to the entire space domain minus the region identified by S. When B is reported to be in proximity, A learns that B is located in one of the granules of S, but not exactly in which of those granules. Therefore, the uncertainty region in this case is given by the region identified by S. The size of this region depends on the value δ_A, on the area of the granules in G^U_B, and on the distance semantics chosen by A. In order to show how the size of the uncertainty region is affected by these parameters, we simplify the analysis by considering grid-like granularities, similarly to Section 3.3.6. Each granularity is a grid identified by the size l of the edge of its cells.

Figure 3.16 shows the additional privacy achieved by *C-Hide&Hash* for different values of δ/l. The additional privacy is measured as the lower bound of the number of granules in S. As can be observed, using both semantics, the additional privacy grows when δ is larger than l. This means, for example, that if δ is 5 times larger than l, then the actual size of the uncertainty region of B is 60 (or 88) times larger than the minimum privacy requirement if A is using the *maximum-distance* (or *minimum-distance*, resp.) semantics.

System costs

We separately evaluate the computation and communication costs involved in running the two proposed protocols. The analytical evaluation reported here is complemented with experimental results in Section 4.1.

C-Hide&Seek In order to perform a location update, a user needs to compute the index of the granule where she is located. The time complexity of this operation depends on the data structure used to represent granularities.

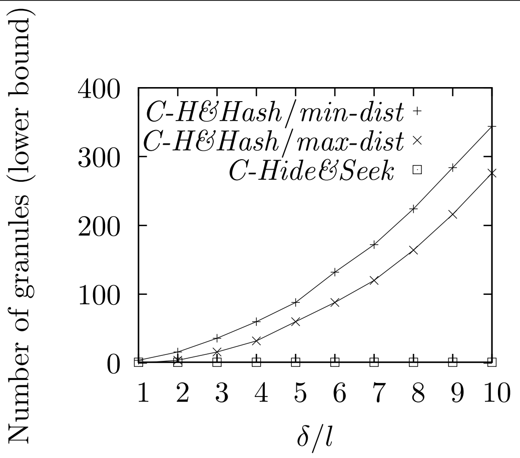


Figure 3.16: Privacy with respect to a buddy

As we shall show in Section 4.1, with our implementation of the granularities this operation can be performed in constant time. The complexity of the encryption operation depends on the encryption function and on the length of the encryption key. Considering a fixed key length, the encryption of the index of the granule can be performed in constant time. Since the SP only needs to store the received information, the expected computational complexity is constant. The communication cost is constant and consists in an encrypted integer value.

For what concerns the cost of a proximity request on the client side, for each buddy the issuing user needs to decrypt the index and to compute the distance of the granule with that index from her location. In our implementation these operations can be performed in constant time and hence the time complexity of the proximity request operation on the client side is linear in the number of buddies. On the SP side, the computational cost to retrieve the last known locations of the buddies is linear in the number of buddies. The communication consists in one request message of constant size from the user to the SP, and of one message from the SP to the user with size linear in the number of buddies.

C-Hide&Hash The cost of a location update operation on the client is similar to the cost of the same operation using *C-Hide&Seek*, since the only difference is that a hashing function, which can be computed in constant time, is applied instead of the encryption function. Like in *C-Hide&Seek*, the SP only needs to store the received information. Hence, computational costs of a location update are constant both for the client and for the SP. The communication cost is constant, as the only exchanged message consists in a hashed value.

On the client side, a proximity request from A requires, for each buddy B, the computation of the granules of G_B^U which are considered in proximity, the hashing, and the encryption of a number of granule indexes in the order of $sMax(G_B^U, \delta_A)$. The value of $sMax$ can be pre-computed for a given granularity. The computation of the granules considered in proximity can be performed in constant time in our implementation, using grids as granularities. The computation of the hashing and the encryption functions can also be performed in constant time, hence the time complexity of a proximity request is linear in the number of buddies times the maximum among the $sMax$ values for the involved granularities. When the client receives the response from the SP, the result computation performed by A for each buddy B requires the encryption of a number (the encrypted value sent by the SP), and the lookup of the encryption in a set of encrypted values with cardinality $sMax(G_B^U, \delta_A)$. As the lookup in the set of hashes requires at most $sMax$ operations, the time complexity is then linear in the number of buddies times the maximum value of $sMax$. Hence, this is also the overall complexity on the client side. On the SP side, the response to a proximity request from a user A requires, for each buddy B, a) the retrieval and the encryption of the hashed location of B, b) the encryption of the $sMax(G_B^U, \delta_A)$ hashed granule indexes sent by A. As the encryption runs in constant time, the time complexity is linear in the number of buddies times the maximum value of $sMax$.

Regarding the communication costs, both of the messages involved in the proximity request sub-protocol contain the encryption of a set of a number of hashed values linear in the number of buddies times the maximum value

Table 3.1: Parameter values

Protocol	Query driven buddies	Distance approximation	Privacy wrt SP	Privacy wrt buddies	System costs
SP-Filtering	✓	Region-Region	Minimum required	Same as SP	Low
Hide&Seek	✓	Region-Region	Minimum required	Minimum required	Average
Hide&Crypt	✓	Point-Region	Minimum required	More than required	High
Longitude		Region-Region	Total[1]	More than required[2]	Low
C-Hide&Seek		Point-Region	Total	Minimum required	Low
C-Hide&Hash		Point-Region	Total	More than required	Average

[1] assuming the adversary has no a-priori knowledge of the locations

[2] privacy requirement must be expressed as a grid with cells of the same size of $sMax$.

3.4 Comparison of the protocols

Table 3.1 shows a comparison of the characteristics of the protocols presented in this chapter.

As can be observed, the *SP-Filtering*, *Hide&Seek* and *Hide&Crypt* protocols are the only ones that can support services in which the buddies are not predetermined. However, the system costs of *Hide&Seek* and *Hide&Crypt* can significantly grow if the users have strict privacy requirements with respect to the SP, as this would require frequent buddy-to-buddy communications. Among these three protocols, *Hide&Crypt* achieves a better service precision due to a more precise distance approximation, and can provide more privacy than strictly required.

The main advantages of the *Longitude* protocol are the communication and computation costs, as it is fully centralized and employs a fast symmetric encryption function. The drawbacks of this protocol are the lack of privacy guarantee with respect to the SP, under the assumption that the a-priori distribution of the location of users is already known, and the constraint about the privacy preferences with respect to the users to be limited to grids, instead of arbitrary spatial granularities.

The *C-Hide&Seek* and *C-Hide&Hash* protocols can achieve a service precision comparable to *Hide&Crypt*, with a significant reduction of the system costs. Both protocols are formally proved to guarantee total privacy with respect to the SP. In addition, the *C-Hide&Hash* protocol can achieve more privacy than strictly required with respect to the buddies, at a relatively low additional cost in terms of computation and communication. The sustainability of these costs as well as the effective performance in terms of privacy and quality of service has been empirically verified, and results are reported in Chapter 4.

The selection of an appropriate protocol to be used depends on the applicative context and on the requirements of the service. The *C-Hide&Seek* and *C-Hide&Hash* protocols are the solutions that provide the best privacy guarantees, keeping the costs sustainable both on the client and the server side. The *C-Hide&Seek* protocol can also easily be modified to be used as a location tracking service (like Google Latitude), that is another popular category of services in the context of GeoSNs. It should be noted that the *C-Hide&Hash* protocol may require a large amount of computing resources when applied to a very large scale of users, compared to the other protocols, and the resulting cost may be a concern for a small service provider. If this is case, the *Longitude* protocol could be a lighter solution, but users should be aware of its weakness with respect to an adversary with a-priori knowledge of the locations of the users. Finally, if the applicative context requires "query-driven" buddies, only the *SP-Filtering*, *Hide&Seek* and *Hide&Crypt* solutions can be applicable.

Chapter 4

Evaluation of the proposed defenses

4.1 Empirical evaluation

We conducted experiments to measure the performance of our *C-Hide&Seek* and *C-Hide&Hash* protocols, which are our most recent and promising solutions, and to compare them with our *Hide&Seek* and *Hide&Crypt* protocols as well as the Pierre and `FriendLocator` protocols [48, 53]. We present the experimental setting in Section 4.1.1. Then, in Sections 4.1.2, 4.1.3 and 4.1.4 we evaluate the protocols according to three evaluation criteria: quality of service, privacy and system costs, respectively.

The experiments reported in this section do not include the *Longitude* solution, because it does not fully satisfy Definition 1 with respect to the SP as explained in Section 3.3.4. We point the reader to the relative publication for the experiments about this protocol [34].

4.1.1 The experimental setting

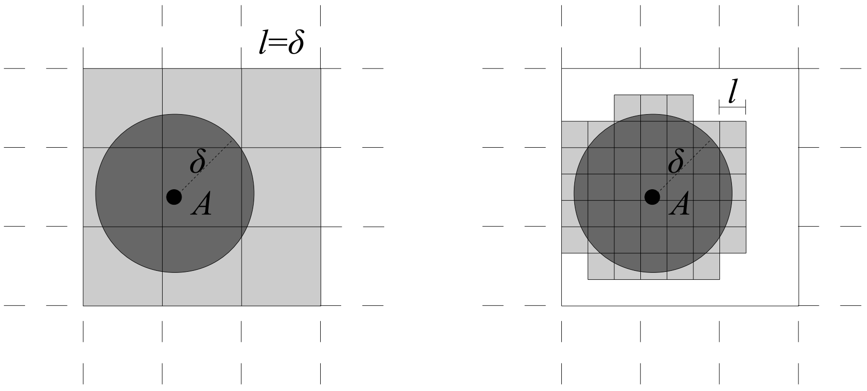
The experimental evaluation of the protocols presented in Section 3.3 was performed on a survey-driven synthetic dataset of user movements, which was obtained using the *MilanoByNight* simulation (see Section 2.3.1). We carefully tuned the simulator in order to reflect a typical deployment scenario of a proximity service for geo-social networks: $100,000$ potential users moving between their homes and one or more entertainment places in the city of Milan during a weekend night. The simulation also models the time spent at the entertainment places, i.e., when no movement occurs, following probability distributions extracted from user surveys. 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 215 km^2 and the average density is 465 users/km^2. All the components of the system are implemented in Java. Server-side test were performed on a 64-bit Windows Server 2003 machine with 2.4Ghz Intel Core 2 Quad processor and 4GB of shared RAM. Client-side tests were run on a HTC Magic mobile device, running Android as operating system. We implemented the symmetric encryption and the hashing functions using the RC4 and MD5 algorithms, respectively, while the RSA public key encryption algorithm was used for the key distribution.

In the experiments we used grid-based granularities. Each granularity is identified by the size of the edge of one cell of the grid. The location-to-granule conversion operations required by our protocol can be performed in constant time. For the sake of simplicity, in our tests we assume that all the users share the same parameters and that each user stays on-line during the entire simulation. Table 4.1 shows the parameters used in our experiments. Note that the "number of buddies" parameter refers to the number of *on-line* buddies that, for the considered type of application, is usually significantly smaller than the total number of buddies. The parameter G_A^{SP} is set to $\top$ for all protocols i.e. the users require total privacy with respect to the SP. This parameter could only vary for the *Hide&Seek* and *Hide&Crypt* protocols, and its fixed for the other protocols. For the impact of this parameter to *Hide&Seek* and *Hide&Crypt* we point the reader to the paper that presented these protocols [36].

Table 4.1: Parameter values

Parameter	**Values**
δ	200m, **400m**, 800m, 1600m
Edge of a cell of G^U_A	100m, **200m**, 400m, 800m
Number of buddies	10, 20, **40**, 80

4.1.2 Evaluation of the quality of service



(a) Pierre / `FriendLocator` (b) Our solutions

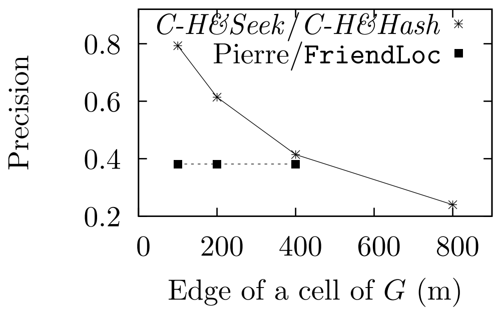
Figure 4.1: Examples of the *granularity approximation*

The first set of experiments evaluate the impact of the techniques on the *quality of service*, by measuring the exactness of the answers returned by each protocol. Indeed, two forms of approximation are introduced by our protocols. The *granularity approximation* is caused by the fact that, when computing the proximity between two users, the location of one of them is always generalized to the corresponding granule of her privacy requirement granularity. The other approximation, which we call the *time-dependent approximation*, is due to the fact that, when a user issues a proximity request with *C-Hide&Seek*, proximity is computed with respect to the last reported location of each buddy. The approximation is introduced because the buddies have possibly moved since their last location update. Similarly, during the computation of a proximity request with *C-Hide&Hash*, the location transmitted by each buddy during the previous update interval is used.

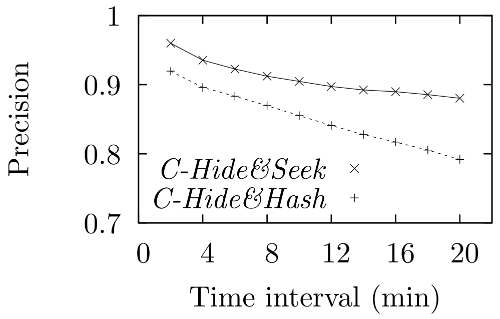
For what concerns the *granularity approximation*, a similar problem occurs with the Pierre and `FriendLocator` protocols too. Indeed, both protocols, in order to detect proximity between buddies, partition the domain space into a grid, with each cell having edge l equal to the distance threshold δ, that must be shared by the users. Then, a buddy B is considered in proximity of A whether B is located in the same cell as A or in one of the 8 adjacent cells. The approximation introduced by these techniques depends entirely on the chosen value of δ. Differently, in our solutions, each user can choose her privacy requirements independently from the value of δ. For example, consider Figure 4.1. The black dot is the actual location of user A. The dark gray circle with radius δ is the area where the buddies of A are actually in proximity of A. The light gray area is the region in which buddies are erroneously reported to be in proximity[1]. Considering Figure 4.1(a), as l is always equal to δ when using Pierre or `FriendLocator`, the total area of the 9 cells considered in proximity is $9\delta^2$, while the area of the circle is $\pi\delta^2$, which is almost 3 times smaller. This means that, assuming a uniform distribution of the users, using Pierre or `FriendLocator` the probability that a buddy reported as in proximity is actually in proximity is about 1/3. On the contrary, in the protocols presented in this paper the size of the granules is independent from the chosen δ. In our example, this means that when the value l is smaller than δ, the region in which users are erroneously reported in proximity becomes smaller (Figure 4.1(b)).

Figure 4.2(a) shows how the *granularity approximation* impacts on the service precision for different values of the edge of granularity cells. The metric we use for the measurement is the information retrieval notion of *precision*: the ratio between the number of correct "in proximity" answers

[1]Here and in the following, we assume users of our protocols are choosing the *minimum-distance* semantics



(a) *Granularity approximation* only



(b) *Time-dependent approximation* only

Figure 4.2: Evaluation of the impact of the approximations

over the total number of "in proximity" answers. Intuitively, the precision measures the probability that a buddy reported "in proximity" is actually in proximity. Note that the analysis would be incomplete without considering the notion of *recall*: the ratio between the number of correct "in proximity" answers over the sum of correct "in proximity" and incorrect "not in proximity" answers. Intuitively, the recall measures the probability that a buddy actually in proximity is reported "in proximity". In this case, since we are considering the *minimum-distance* semantics (see Section 3.3.6), the *granularity approximation* does not produce any incorrect "not in proximity" answer, and hence the recall is equal to 1. When conducting this experiment, in order to exclude from the evaluation the effects of the *time-dependent approximation*, for each buddy we used his current location as the last reported location. Since Pierre and `FriendLocator` do not consider

G_A^U, their precision is constant in the chart and, as expected, is below 0.4. On the contrary, *C-Hide&Seek* and *C-Hide&Hash* have a significantly better precision when the edge of the cells of G_A^U is small. Intuitively, this is because the area where a buddy is erroneously reported as in proximity is smaller than δ (see Figure 4.1(b)). Figure 4.2(a) also shows the precision when the edge of a cell of G_A^U is larger than δ; The values are not reported for Pierre and `FriendLocator` since in this case they do not guarantee the privacy requirements.

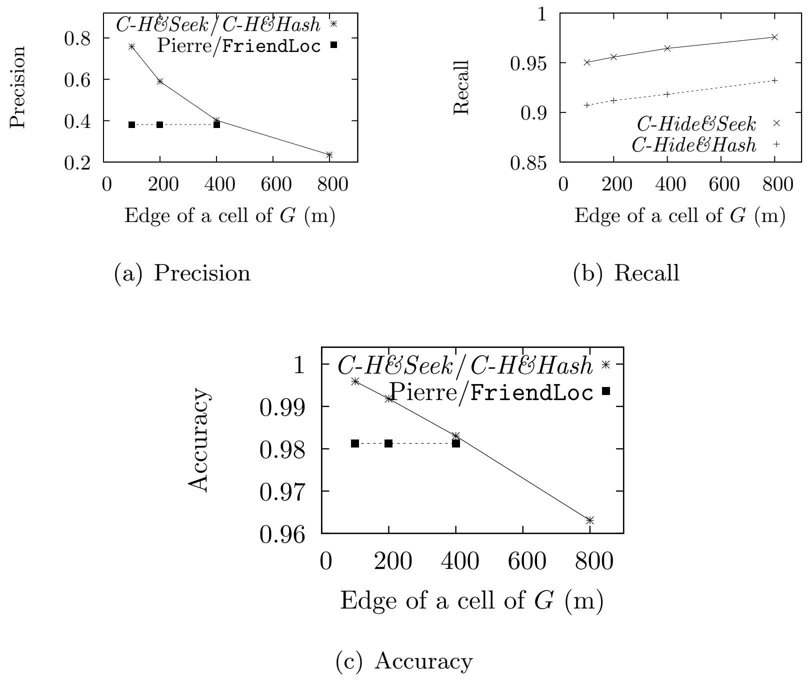


Figure 4.3: Evaluation of the quality of service (considering both approximations)

Figure 4.2(b) shows the impact of the *time-dependent approximation*. The chart shows the results for our protocols only, as the other protocols proposed in the literature are not exposed to this kind of approximation. In order to exclude from this evaluation the effects of the *granularity approximation*, we performed these tests with the exact locations of the users, instead of the generalized ones. The chart shows, on the x axis, different lengths of the update interval and, on the y axis, the precision of the *C-Hide&Seek* and *C-Hide&Hash* protocols. It can be observed that *C-Hide&Seek* has better precision. This is due to the fact that *C-Hide&Hash* always uses the location reported during the previous update interval, while *Hide&Seek* uses the last location, that can be the one reported during the current update interval or during the previous one. Since the *time-dependent approximation* also introduces incorrect "not in proximity" answers, we also measured the recall. The corresponding chart is omitted as it is almost identical to the one in Figure 4.2(b). For example, using *C-Hide&Hash* and an update interval of 4 minutes, the value of the precision is 0.89 and the recall is 0.88.

The computation of the precision and recall under the *time-dependent approximation* confirms the intuition that using long update intervals negatively impacts on the quality of service. The choice of a value for the update interval should consider, in addition to this approximation, the cost of performing a location update. In general, the optimal value can be identified based on specific deployment scenarios. Considering our movement data, we chose 4 minutes as a trade off value since it guarantees precision higher than 0.9 and sustainable system costs as detailed in Section 4.1.3. Our choice is consistent with similar proximity services like, for example, Google Latitude that currently requires location updates every 5 minutes.

Figure 4.3 shows the analysis of the quality of service considering both the *granularity* and *time-dependent* approximations. Figure 4.3(a) shows the precision of *C-Hide&Hash* and *C-Hide&Hash* protocols compared with the precision of Pierre and `FriendLocator`. We represent the precision of *C-Hide&Seek* and *C-Hide&Hash* with a single curve because the two protocols behave similarly. For example, when the edge of a cell of G_A^U is 200m, the precision of *C-Hide&Seek* and *C-Hide&Hash* is 0.59 and 0.57, respectively, while it is 0.61 for both protocols when the *time-dependent approximation* is not considered. This shows that this second type of approximation does not have a significant impact.

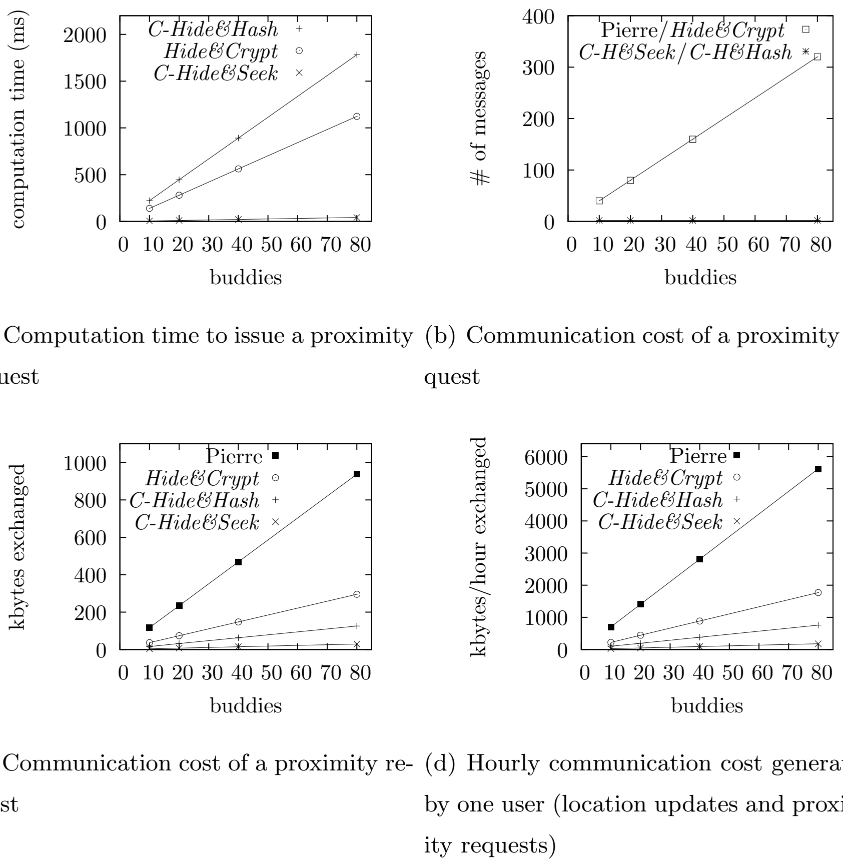
Figure 4.3(b) shows the recall of our protocols. Note that Pierre and

`FriendLocator` do not lead to incorrect "not in proximity" answers, and hence their recall is equal to 1. On the contrary, our protocols can generate incorrect "not in proximity" answers due to the *time-dependent approximation*. This chart shows that the recall of *C-Hide&Seek* and *C-Hide&Hash* is always above 0.95 and 0.9, respectively. From Figure 4.3(b) we can also observe that the recall increases for coarser granularities. This is due to the fact that less incorrect "not in proximity" answers are returned if a coarser granularity is used. While this may appear unreasonable, the explanation is straightforward: there is an incorrect "not in proximity" answer only when a buddy is currently in proximity (considering Figure 4.1(b), his location is in the dark gray area) while the location used in the computation of the proximity is outside the light gray area. If a granularity is coarse, then the light gray area is large and hence incorrect "not in proximity" are less frequent.

Figure 4.3(c) shows the *accuracy* for each considered protocol, i.e., the percentage of correct answers. Also in this case, the accuracy of *C-Hide&Seek* and *C-Hide&Hash* is represented with a single curve, as the two protocols behave similarly. Comparing this figure with Figure 4.3(a), we can observe that the accuracy achieved by all the protocols is much higher than the precision. This is due to the fact that this metric also considers the correct "not in proximity" answers that are usually the most frequent answers, since the proximity query area determined by the distance threshold is usually much smaller than the entire space. Figure 4.3(c) shows that our protocols achieve better accuracy than Pierre and `FriendLocator` when the value of the edge of the granularity cells is smaller than δ. In particular, for our default values, the accuracy of both *C-Hide&Seek* and *C-Hide&Hash* is higher than 0.99.

4.1.3 Evaluation of the system costs

The second set of experiments evaluates the computation and communication costs of the different protocols. For the analysis of the Pierre protocol,



(a) Computation time to issue a proximity request

(b) Communication cost of a proximity request

(c) Communication cost of a proximity request

(d) Hourly communication cost generated by one user (location updates and proximity requests)

Figure 4.4: Evaluation of the system costs

we used the NearbyFriend[2] application, developed by the same authors, which integrates the Pierre protocol in a desktop IM application.

First, we consider the costs related to the location update sub-protocol. This analysis does not apply to existing solutions as location updates are only required by our centralized solutions. As analyzed in Section 3.3.6, the temporal complexity of computing a location update is constant in the number of buddies. In our implementation, the computation of each location update requires, on the client side, about half of a millisecond for both the *C-Hide&Seek* and the *C-Hide&Hash* protocols. Similarly, the communication cost is independent from the number of buddies and the payload of each

[2]http://crysp.uwaterloo.ca/software/nearbyfriend/

location update message consists in few bytes. Considering the overhead caused by the XML encapsulation, the dimension of each location update is in the order of a few hundred bytes.

The computation time needed to run a proximity request on the clients is shown in Figure 4.4(a). Note that the values reported in this figure only consider the computation time required by the issuing user. Indeed, all the protocols require the SP (in case of centralized services) or the other buddies (in case of distributed services) to participate in the protocol, and hence to perform some computation. For example, in the case of *Hide&Crypt* and Pierre, the total computation time of a user's buddies to answer a proximity request issued by that user is about the same as the computation time required to issue the request. As observed in Section 3.3.6, the computation time of a proximity request is linear in the number of buddies. Figure 4.4(a) shows that *C-Hide&Hash* requires significantly more time with respect to *C-Hide&Seek*, especially when the number of buddies is large. For example, the time needed to issue a proximity request for 40 buddies is about 20 ms for *C-Hide&Seek*, while about 900 ms using *C-Hide&Hash*. The figure also shows that the computation times of *C-Hide&Hash* and *Hide&Crypt* are similar, with *Hide&Crypt* performing slightly better. This is due to the fact that in *Hide&Crypt* each of the $sMax$ indexes only needs to be encrypted, while in *C-Hide&Hash* it also needs to be hashed.

For what concerns other existing solutions, we did not implement the Pierre protocol on our mobile device platform. However, considering the experimental results presented by the authors (see [53]), the computation time of a single proximity request with a single buddy is more than 350ms[3]. Since, for *C-Hide&Hash*, the computation time on a mobile device of a proximity request with a single buddy is about 22ms, according to the data we have, our solution is at least one order of magnitude more efficient than the Pierre solution.

Regarding the computation costs on the server side, the complexity of

[3]It is unclear whether this result is obtained on a mobile device.

a proximity request using *C-Hide&Hash* on the server side is similar to the one on the client side. However, in our experiments we observed that our high-end desktop machine is about 500 times faster than the mobile client to execute these operations. As a consequence, the computation for a single user having 40 buddies requires less than 2ms. While we did not run scalability tests on our server, this result suggests that, from the computational point of view, even a single desktop machine can provide the service to a large number of users.

Figures 4.4(b) and 4.4(c) show the system communication cost of a proximity request issued by a user. In Figure 4.4(b) we measure the number of messages exchanged by the system for each proximity request. It is easily seen that using a centralized protocol (i.e., *C-Hide&Seek* and *C-Hide&Hash*), only two messages need to be exchanged (one for the request and one for the response) independently from the number of buddies the issuer has. On the contrary, the decentralized protocols requires at least two messages for each buddy. Moreover, in our implementation of the *Hide&Crypt* protocol, each communication between two users needs to transit through the SP. The same applies to the Pierre protocol, using the NearbyFriend implementation. Consequently, at each location update, for each buddy, four messages transit in the system: two between the issuer and the SP and two between the SP and the buddy.

Figure 4.4(c) shows a comparison of the total amount of data exchanged in the system for each proximity request. Consistently with our analysis, the communication cost grows linearly with the number of buddies for both of our centralized protocols. It is easily seen that this also applies to the other protocols. The chart shows that NearbyFriend incurs in high communication costs. The reason is that, each time a proximity request is issued, a message of almost 3KB is sent from the user to each of her buddies and a message having a similar size is sent back in the reply. We believe that this overhead is mostly given by the fact that NearbyFriend needs all the communications between two users to be encapsulated in a secure channel.

This is required because the Pierre protocol itself does not guarantee that any third party acquiring the messages cannot derive location information about the users. Since each message between two users transits through the server, the communication cost is almost 12KB for each buddy. The other decentralized solution we compare with, *Hide&Crypt*, has better communication costs. Indeed, each message is less than 1KB, and hence the cost is about 1/4 if compared to Pierre.

Our centralized solutions are even more efficient. This is due to the fact that only two messages need to be exchanged between the user and the SP for each proximity request. In case of *C-Hide&Hash*, each message has the same dimension than in *Hide&Crypt*, and hence, in this case, the communication cost is one half with respect to *Hide&Crypt*, and about one order of magnitude less with respect to Pierre. Finally, *C-Hide&Seek*, in addition to being a centralized solution, also benefits from the fact that each message contains only a few hundred of bytes. Consequently, this protocol is about 4 times more efficient than *C-Hide&Hash*.

In Figure 4.4(d) we evaluate the communication cost of the continuous use of a proximity service with our protocols. As mentioned in Section 4.1.2, we consider that location updates are issued every 4 minutes. Considering the results of our user survey, we use 10 minutes as the average frequency of proximity requests. The main difference of this figure with respect to Figure 4.4(c) is that it also considers the communication costs derived by the location updates. However, since each location update costs less than 300 bytes, and 15 location updates need to be issued in one hour, the total hourly cost for this sub-protocol is about 4KB, which is negligible with respect to the communication cost of the proximity requests. The figure also shows that the centralized protocols require significantly less communication than the decentralized ones. In particular, *C-Hide&Seek* for one hour requires less than 100KB when the user has 40 online buddies. *C-Hide&Hash*, on the other side, requires 400KB per hour for the same number of buddies. We believe that this cost is largely sustainable on a wireless broadband network (e.g., 3G), and that, given the additional privacy with respect to curious buddies achieved using *C-Hide&Hash*, privacy concerned users may find this trade-off attractive.

Our experimental evaluation also included the measurement of the cost to distribute the private key (see Section 4.2). Both the computation and communication costs are linear in the number of buddies that need to receive the new key. For a single buddy, the computation time is about 7ms, measured on the mobile device, while the communication cost is less than 200 bytes. An experiment of key distribution to 40 buddies, resulted in a computation time of 275 ms, and a communication cost of 7KB.

4.1.4 Evaluation of the achieved privacy

In Section 3.3.6 we proved that both of our protocols guarantee the users' privacy requirements. We also observed that that *C-Hide&Hash* provides more privacy than what would be strictly necessary to guarantee the requirements. In this last set of experiments we evaluate how much additional privacy is provided by *C-Hide&Hash* in terms of the size of the uncertainty region. We recall that this is the area where a user A is possibly located as it can be computed by one of A's buddies after issuing a proximity request that returns A as in proximity.

Figure 4.5 shows that the privacy provided by *C-Hide&Hash* is always significantly larger than the privacy requirement, and it grows for coarser granularities G_A^U. Intuitively, with *C-Hide&Hash*, the uncertainty region corresponds to the union of the light and dark gray areas represented in Figure 4.1(b). Consequently, as the size of the cells of G_A^U decreases, the size of the light gray area tends to zero, and the uncertainty region becomes closer and closer to the dark gray area only. This means that the privacy provided by *C-Hide&Hash* is at least $\pi\delta^2$ even when the user requires her location to be obfuscated in a smaller area.

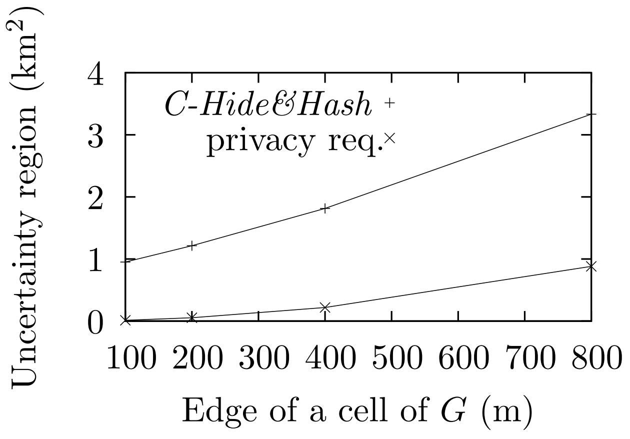


Figure 4.5: Size of the uncertainty region.

4.2 Implementation of the service

We implemented the techniques presented in Section 3.3 and the Pierre and `FriendLocator` solution in a system, called `Pcube`, that provides proximity notification coupled with typical instant messaging (IM) functionalities. In addition to providing this service, the system allows a "live" comparison of the performance of the different protocols in terms of service precision and privacy achieved.

The system is built as an extension of XMPP (Extensible Messaging and Presence Protocol), an open standard protocol often used in commercial applications as a message oriented middleware [44]. The choice of extending XMPP is driven by the following considerations. First, the XMPP protocol can be easily extended to support custom services and messages, like the proximity service, in our case. In particular, by extending XMPP messages, we designed a proper XML protocol for each of our techniques. The system architecture is shown in Figure 4.6. The SP providing the proximity services is implemented as a XMPP component (called Proximity component) i.e., a pluggable entity that extends the default XMPP functionalities. Hence, we developed a distinct XMPP component for each technique that requires algorithms to run on the SP side. A second advantage is that the XMPP protocol already includes standard sub-protocols for client-to-client commu-

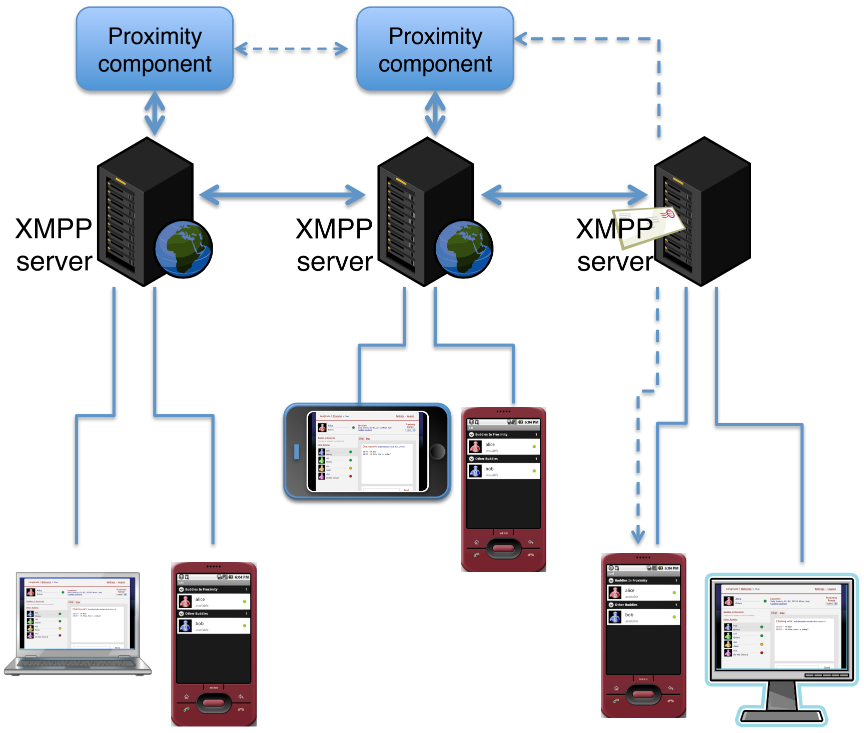


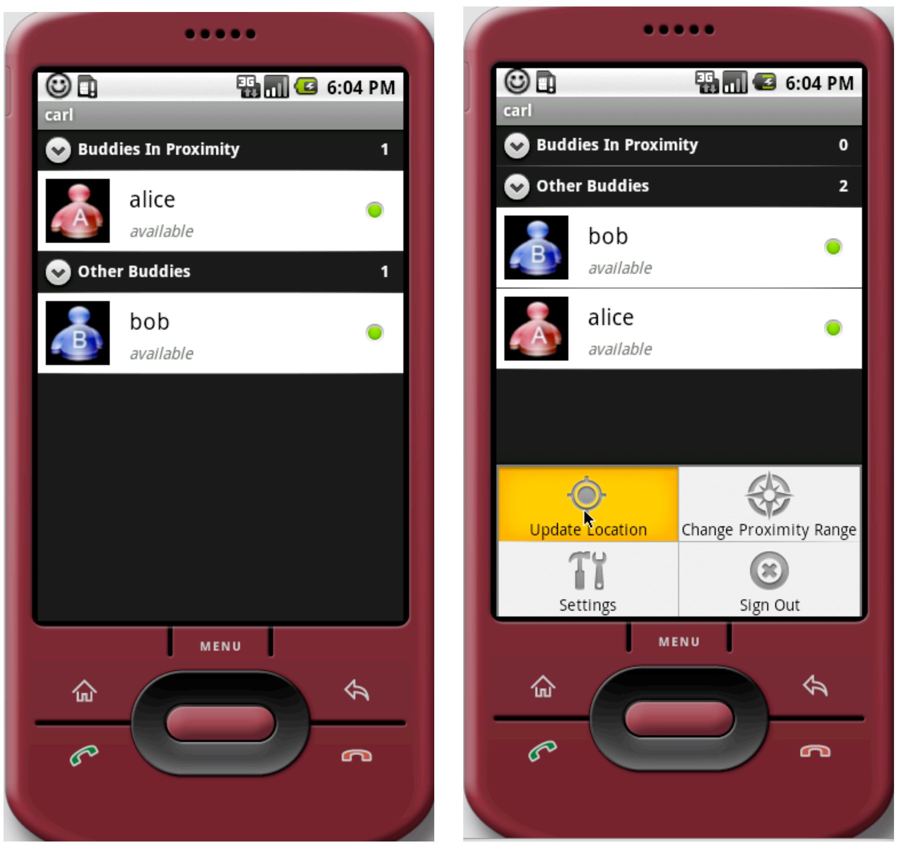
Figure 4.6: System architecture

nication and for managing the list of buddies. We used these sub-protocols as primitives in our implementation. Since the XMPP architecture is decentralized, clients running on different servers can communicate with each other. In our case, since a component acts as a special type of client, this means that our proximity service is accessible to a user registered to an existing XMPP service, including popular IM services like Google Talk or Jabber. This makes it possible to use, in the proximity service, the same list of buddies used in those IM services. Clearly, proximity can be computed only for those buddies that are participating in the same proximity service.

For what concerns the client, we developed a multi-platform web application and an other application specifically designed for Android based smartphones. In addition to the typical functionalities of an IM application, the clients implement all the proximity protocols presented in Section 3.3 and provide the typical functionalities of a full-fledged proximity service, including the detection of the client user's location, the notification of any buddies in proximity, and the graphical visualization of the location uncertainty region for each buddy. We also developed a monitoring system that emulates different users using multiple protocols at the same time, and provides a web interface to control the users and analyze the performances of the different protocols. A more detailed description of the clients is in Sections 4.2.1, 4.2.2 and 4.2.3.

One of the issues emerged during the implementation of the *C-Hide&Hash* and *C-Hide&Seek* protocols concerns key management. Indeed, both protocols require that each user A has a key K_A that is shared with all of her buddies, and it is kept secret to everybody else. A first problem is how A can share her key with one buddy B in a secure manner. This operation is required, for example, when the user accesses the proximity service for the first time or a new buddy is added to the buddy list. To address this problem, we employ standard public key cryptography techniques to encrypt, for each buddy of a user A, the key K_A; After being encrypted, the key can be safely transmitted over an insecure channel. The second problem is how to revoke a secret key. For example, this is necessary when a buddy is removed from the buddy list, or when the key is compromised. In our implementation, in order to revoke a key, it is sufficient to generate a new secret key and to send it to the authorized buddies.

The cost of sending a key to all the buddies is clearly linear in the number of buddies. In Section 4.1 we show that the costs to perform this operation on a mobile device are sustainable. In addition, it should be observed that the distribution of the key to all the buddies is only needed when a user first subscribes to the proximity service or when a buddy is removed from the buddy list. These are very sporadic events during a typical IM service provisioning.



(a) Buddy list (b) Menu buttons

Figure 4.7: Android application

4.2.1 Mobile client

The application running on Android uses the platform's API to acquire the location and has the main functionalities of the web application. For example, Figure 4.7(a) shows the main interface of the application, containing the two lists of buddies with Alice being in proximity. When the physical "menu" button is pressed (Figure 4.7(b)), the application shows the options to update the location, to set the proximity range, to change the settings and to sign out.

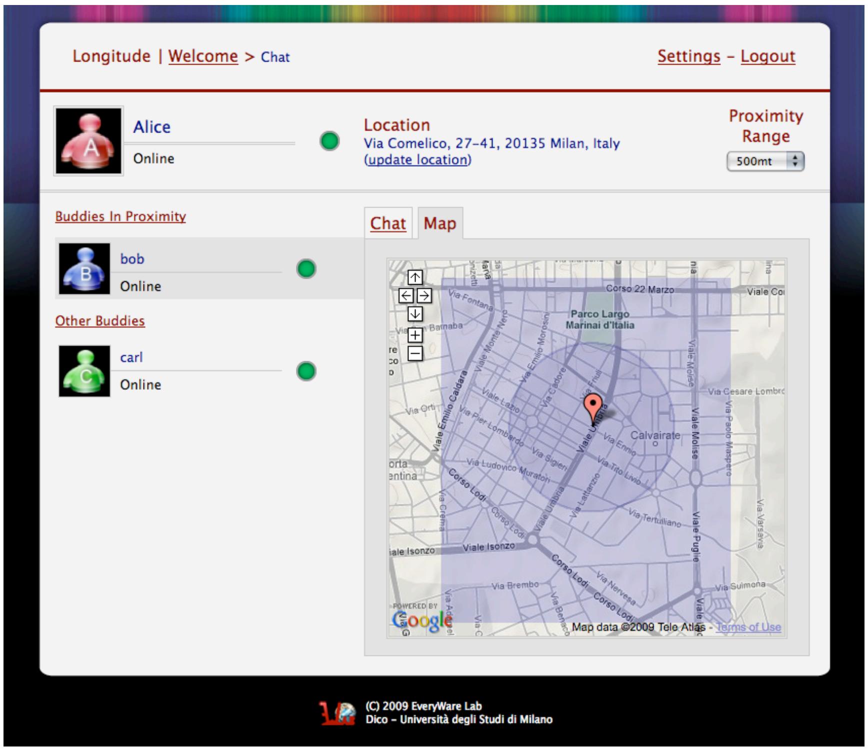


Figure 4.8: Web application interface ("Map" tab)

4.2.2 Web client

Figure 4.8 shows a screenshot of the web application. All the typical functionalities of an IM application are provided (e.g.: setting a nickname, choosing a picture, changing the availability status). The user can see the address from where the last location update was sent to the SP and can require to update her location (using the "update location" link). The application uses Geolocation API[4] to obtain a user's location directly from the browser. Also, the user can see and change the distance threshold below which buddies are considered in proximity (the so called "Proximity Range"). The "Settings" link brings to an interface that allows a user to change his preferences for the location updates, the protocols to be used and the privacy preferences for each protocol, where applicable. In the center part of the interface, there

[4]http://www.w3.org/TR/geolocation-API/

are two tabs. The “Chat” tab allows users to communicate each other via instant messaging. The “Map” tab, which is the one highlighted in this screenshot, shows on a map a) a red “location-mark” situated where Alice (the user, in the example) issued the last location update, b) a circle, centered in Alice’s location, representing the proximity query (the “proximity range” is 500m in the example) and c) a shadowed rectangle representing the uncertainty region where the selected buddy (Bob, in the example) is located.

4.2.3 Monitor application

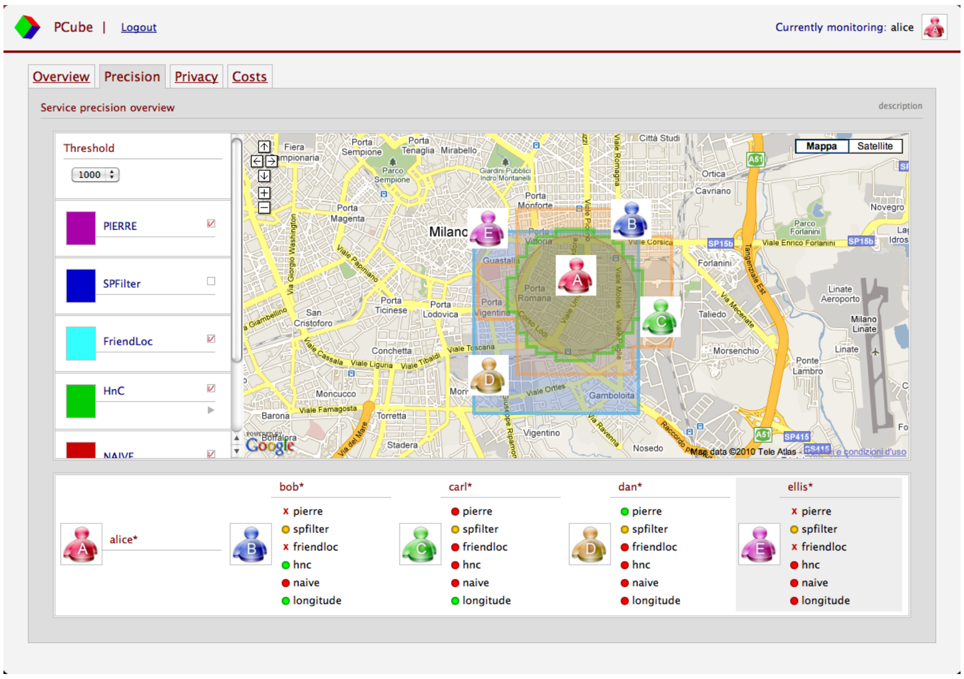


Figure 4.9: Monitor application interface

We developed a set of tools to visually evaluate the performances of the different techniques. To perform the evaluation, we simulate multiple users accessing the service, that can be arbitrarily moved on a map by using a web-based control application, shown in Figure 4.9. The simulated users are semi-automatic XMPP clients that accept movement and setting "commands" from the monitor, run the proximity protocols and report back statistics to the monitor. The statistics are then processed and graphically shown on the screen.

The "Precision" tab shows, for each protocol, the region in which the buddies of a user are reported in proximity, and the actual proximity region centered in that user's location. It is then possible to see interactively how these regions change with the different parameters, and what would be their precision in a real-word scenario. The "Privacy" tab reports the effective uncertainty region achieved by each protocol, letting the evaluator observe how the regions compare with different parameters. Finally, the "Costs" tab reports live graphs about the effective traffic generated by the clients for each different protocol.

Chapter 5

Conclusions and future work

In this chapter we summarize main contributions of this dissertation and we address problems arising from data privacy preservation in proximity-based services that still need to be deeply investigated.

5.1 Summary of the contributions

The main goal of this thesis was the analysis of the privacy issues in the context of location-based proximity services and the proposal of defense techniques to provide controllable disclosure of location information to the users.

To perform this analysis, a deep study of the existing privacy preserving techniques for LBS has been performed, in order to evaluate their applicability to this particular context. We presented a formalization of the problem of location privacy in proximity services, considering both the SP and the other participants to the service as potential adversary for a user. We proposed a flexible way to express users' privacy requirements, by letting them define geographical region in which they do not want to be exactly localized by a considered adversary.

We proposed five different protocols that are formally proved to enforce users' privacy preferences against a well defined adversary model with each protocol having different characteristics in terms of requirements, performance or privacy guarantees. All the protocols have been theoretical analyzed and the result of the analysis has been confirmed by an extensive experimental evaluation. The protocols perform better than other solutions already presented in literature under every performance measure we considered.

An implementation of our solution was included in a fully functional instant messaging system that can extend the functionality of existing commercial products with proximity services. The developed tools also allowed an empirical and visual comparison of the different solutions we proposed, as well as other solutions presented in literature.

5.2 Future work

Proximity services

An interesting extension of our protocols is to allow users to specify different privacy preferences with respect to different groups of buddies. This is not difficult, but it exposes the users to dangerous collusion attacks if further constraints are not imposed. The presented protocols are not subject to buddies' collusion attacks since each user defines the same granularity as privacy preference with respect to all of her buddies. If this is not the case, a user A, by assigning two different granularities with respect to buddies B and C to reflect her different level of trust, would expect that if B and C collude the lowest privacy requirement among the two is preserved. However, an adversary could actually intersect the uncertainty regions and potentially violate both privacy requirements. In order for our protocols to defend against such a collusion, some relationships need to be imposed on the granularities used in the system. While details are out of the scope of this paper, intuitively, granules from different granularities should never partially overlap. For example, using hierarchical grids as granularities would be a sufficient condition.

Another direction we plan to investigate is to extend the adversary models we considered in this paper to include not only (atemporal) a-priori location knowledge, but also *time-dependent* location knowledge. This would model not only a-priori knowledge about velocity, that our solutions can already deal with, but also a-priori probabilistic *proximity* information. It is still unclear if the proposed protocols, with appropriate location update strategies need to be modified in order to be proved privacy-preserving according to our definitions.

The design of an entirely decentralized proximity service is another challenging task. As we have observed, current decentralized solutions can lead to high computation and communication costs, which is the main reason why, in our solutions, we involved the SP in the computation of the proximity. However, we do not exclude that a properly designed decentralized protocol could provide a location privacy-aware proximity service while keeping the system costs sustainable.

Privacy in GeoSN

Our current research effort is dedicated to the study of privacy threats for GeoSN services different from proximity services. In fact, proximity services only one of the services available to GeoSN users. More precisely they are a particular subcategory of the friend tracking services, that typically allow a user to see the location of their buddies on a map regardless of the distance between them.

Another popular category of GeoSN services is the publication of resources tagged with geo-location. Such resources can be, for example, status messages, photos, or "check-ins" and they are tagged with the location in which they were generated. Further, resources may reference other users: for example, this occurs when a user tags a photo with the people in the photo. Most popular services exploiting GeoSN resources are Facebook Places, Foursquare, Google Picasa, Flickr, Brightkite, Google Buzz, Google Latitude, Gowalla, Loopt, Twitter, and Whrrl.

In a recent work [17], which we describe briefly, we analyzed some of the privacy threats for users of this category of services and proposed two techniques to preserve privacy while maintaining some utility of the published data.

The privacy threats we described in Section 3.2.2 for proximity-based services also apply to the context of resource publishing services. In addition, we considered the temporal component of the location information as potentially private. A typical example is a user who elects to not let people know that he attended a religious ceremony or a political meeting. Another privacy concern we addressed is *absence privacy*, that is the uncontrolled disclosure of the absence of a user from a geographic position at specific times can occur. This concern is conceptually different from location privacy and requires different protection techniques. A typical example is a user not wishing to let people know that he will not be at home for an extended period of time, as this information could be used to plan a burglary. Note that these concerns are amplified in those GeoSNs that allow user-tagging of geo-localized resources, because the users do not have control about being tagged in a resource. We proposed a model to let users express their location and absence privacy preferences, and designed an enforcing mechanism that employs specific spatial or temporal cloaking techniques, and publication delay. For further details we point the reader to the relative paper.

As a future work in this field, it would be interesting to investigate a more flexible way of expressing users privacy preferences, possibly differentiating their preference for different (groups of) users. Another interesting problem that deserves a deeper study are the *co-location* privacy violation [45] i.e. an adversary obtains to know that two users met in a certain place, which can be considered a privacy violation if those users do not want to disclose the fact that they meet.

Appendix A

Proofs

A.1 Proof of Proposition 1

In order to prove Proposition 1, we first prove Lemma 3.

Lemma 3. *Let i, S_{in}, S_{out}, $Area_F^{in}$, $Area_F^{out}$ and $Area_{NoFilter}$ as defined in Proposition 1. Whenever the* SP-Filtering *protocol is used to compute the proximity of B with respect to A:*

1. *A receives the message "B is in proximity" from the SP, if and only if B is located in $Area_F^{in}$;*
2. *A receives the message "B is not in proximity" from the SP, if and only if B is located in $Area_F^{out}$;*
3. *A receives the message "B is possibly in proximity" from the SP, if and only if B is located in $Area_{NoFilter}$;*

Proof. We prove thesis 1. The proof for thesis 2. is analogous. Given thesis 1. and 2., thesis 3. follows trivially.

$\Rightarrow$) **If $loc(B) \in Area_F^{in}$ then A receives the message "B is in proximity" from the SP.**

Let j be such that $loc(B) \in G_B^{SP}(j)$. We now prove that $j \in S_{in}$. Thesis follows since, by definition of S_{in}, $maxdist(L_A(i), L_B(j)) \leq \delta_A$ and hence, by definition of the *SP-Filtering* protocol, the SP sends the message "B is in proximity" to A.

By definition of $Area^{in}_F$, since $loc(B) \in Area^{in}_F$, then $loc(B) \in \bigcup_{k \in S_{in}} L_B(k)$. By definition of L_B, this implies $loc(B) \in \bigcup_{k' \in S'_{in}} G^U_B(k')$ where $S'_{in} = \bigcup_{k \in S_{in}} \{k' \in \mathbb{N} | G^U_B(k') \cap G^{SP}_B(k) \neq \emptyset\}$.

Now, let j' be the index such that $loc(B) \in G^U_B(j')$. Since $loc(B) \in \bigcup_{k' \in S'_{in}} G^U_B(k')$ and $loc(B) \in G^U_B(j')$ and since two granules of the same granularity do not intersect, $j' \in S'_{in}$.

By Condition 3.2, either $G^U_B(j') \subseteq G^{SP}_B(j)$ or $G^{SP}_B(j) \subset G^U_B(j')$.

If $G^U_B(j') \subseteq G^{SP}_B(j)$ then it does not exist any other granule of G^{SP}_B except $G^{SP}_B(j)$ that intersects with $G^U_B(j')$. Hence, by contradiction, $j \in S_{in}$. Indeed, if $j \notin S_{in}$ then, by definition of S'_{in} $j' \notin S'_{in}$, which is absurd.

If $G^{SP}_B(j) \subset G^U_B(j')$ then, by Condition 3.2 and by the fact that each granularity is a partitioning of the entire spatial domain, there exists a set S such that $G^U_B(j') = \bigcup_{k \in S} G^{SP}_B(k)$. Since there exists no other granule of G^{SP}_B except those in S that intersects with $G^U_B(j')$, by definition of S'_{in}, there exists at least one $k \in S$ which is also in S_{in}, otherwise j' would not be in S'_{in}, By definition of L_B, for each $m, n \in S$, $L_B(m) = L_B(n)$. In particular, $L_B(k) = L_B(i)$. Hence, by definition of S_{in}, since $k \in S_{in}$, $i \in S_{in}$.

$\Leftarrow$) **If A receives the message "B is in proximity" from the SP, then $loc(B) \in Area^{in}_F$.**

Let j be such that $loc(B) \in G^{SP}_B(j)$. By definition of the *SP-Filtering* protocol, the SP returns "B is in proximity" to A if $maxdist(L_A(i), L_B(j)) \leq \delta_A$. Hence, by definition of S_{in}, $j \in S_{in}$. Finally, by definition of L_B, $loc(B) \in G^{SP}_B(j) \subseteq L_B(j)$ and hence, by definition of $Area^{in}_F$, $loc(B) \in Area^{in}_F$ □

We can now prove Proposition 1.

Proof. The thesis follows since, by definition of the *SP-Filtering* protocol, if A receives any of the three messages, then A does not receive any other message. Hence, the thesis follows by Lemma 3. □

A.2 Proof of Theorem 1

Proof. We first prove that (a) there exists at least one granule of G_C^{SP} such that SP is not able to exclude any location of that granule as possible location of C. Then we prove that (b) there exists at least one granule of G_C^U such that any other buddy D is not able to exclude any location of that granule as possible location of C.

(a) By assumption, the SP has no information about the location of C other than the one exchanged during the protocol. Hence, the SP only knows that C is located in $L_C(i)$ where i is the granule of G_C^{SP} where C is located. By definition of L_C, $G_C^{SP}(i) \subseteq L_C(i)$, hence the SP cannot exclude any location of $G_C^{SP}(i)$ as possible location of C.

(b) By assumption, D has no information about the location of C other than the one exchanged during the protocol. Hence, by Proposition 1, D cannot infer any location information about C other than C is located in $Area_F^{in}$ or in $Area_F^{out}$ or in $Area_{NoFilter}$. We now prove, for each of the three areas, that if C is located in the area, than that area covers at least the granule $G_C^U(i')$ where C is located.

$Area_F^{in}$ and $Area_F^{out}$ are defined as the union of regions L_C which, in turn are obtained as the union of granules of G_C^U. Hence, $Area_F^{in}$ and $Area_F^{out}$ are defined as union of granules of G_C^U. Consequently, by definition, also $Area_{NoFilter}$ is the union of granules of G_C^U.

Since, by definition of granularity, two granules of the same granularity do not overlap, if $loc(C) \in G_C^U(i')$, then it does not exists any other granule $i'' \neq i'$ such that $loc(C) \in G_C^U(i'')$. Hence, if $loc(C) \in Area_F^{in}$, then, since $Area_F^{in}$ is the union of granules of G_C^U, then $G_C^U(i') \subseteq Area_F^{in}$.

The proof is analogous for $Area_F^{out}$ and $Area_{NoFilter}$. □

A.3 Proof of Proposition 2

In order to prove the thesis, we prove that:

(a) A receives the message "B is not in proximity" from B if and only if B

is located in $Area_S^{out}$;
(b) A receives the message "B is in proximity" from B, if and only if B is located in $Area_S^{in}$;
(c) B receives the message "starting two-parties protocol $\langle i', \delta_A \rangle$" if and only if A is located in $G_A^U(i')$.

Once (a), (b) and (c) are proved, the thesis follows. Indeed, in case A receives the message "B is not in proximity" from B, by definition of the *Hide&Seek* protocol, the only other message that A receives is "B is possibly in proximity" from the SP. From Lemma 3 the SP sends the "B is possibly in proximity" message to A if and only if B is located in $Area_{NoFilter}$. Hence, if A receives the message "B is not in proximity" from B, then A cannot exclude that B is located in any location of $Area_S^{out} \cap Area_{NoFilter}$. Since $Area_{NoFilter} \subseteq Area_S^{out}$, then $Area_S^{out} \cap Area_{NoFilter} = Area_S^{out}$ hence the thesis. The proof is analogous in the case A receives from B the message "B is in proximity" or in the case B receives from A the message "starting two-parties protocol $\langle i', \delta_A \rangle$".

Case (a) $\Rightarrow$) **If A receives the message "B is not in proximity" from B then B is located in $Area_S^{out}$.**

We show that $loc(B) \in Area_{NoFilter}$ and that $loc(B) \in \bigcup_{k \in S'_{out}} G_B^U(k)$. Thesis follows by definition of $Area_S^{out}$.

By Lemma 3, the *SP-Filtering* protocol returns the "B is possibly in proximity" message to A if and only if $loc(B) \in Area_{NoFilter}$.

By definition of the *Hide&Seek* protocol, given j' such that $loc(B) \in G_B^U(j')$, B sends the message "B is not in proximity" to A only if $mindist(G_A^U(i'), G_B^U(j')) \geq \delta_B$. Then, by definition of S'_{out}, $j' \in S'_{out}$ and hence $loc(B) \in G_B^U(j') \subseteq \bigcup_{k \in S'_{out}} G_B^U(k)$.

Case (a) $\Leftarrow$) **If B is located in $Area_S^{out}$, then A receives the message "B is not in proximity" from B.**

Since, by assumption, $loc(B) \in Area_S^{out}$, then $loc(B) \in \bigcup_{k \in S'_{out}} G_B^U(k)$. By definition of S'_{out}, $mindist(G_A^U(i'), G_B^U(j')) \geq \delta_A$. Hence, by definition of the *Hide&Seek* protocol, the protocol returns the "B is not in proximity" message.

Case (b) $\Rightarrow$) **If A receives the message "B is in proximity" from B then B is located in $Area_S^{in}$.**

By definition of the *Hide&Seek* protocol, A receives the the message "B is in proximity" from B if and only if the *SP-Filtering* protocol returns "B is possibly in proximity" and if B does not return the "B is not in proximity" message. From Lemma 3, it follows that the SP returns "B is possibly in proximity" if and only if $loc(B) \in Area_{NoFilter}$. From point (a) above, it follows that B does not return the "B is not in proximity" message if and only if $loc(B) \in (Area_S^{out})^C$. Hence B returns the "B is in proximity" message if and only if

$$loc(B) \in Area_{NoFilter} \cap ((Area_S^{out})^C = Area_S^{in}$$

Case (b) $\Leftarrow$) **If B is located in $Area_S^{in}$, then A receives the message "B is in proximity" from B.**

Since, by assumption, $loc(B) \in Area_S^{in}$, by definition of $Area_S^{in}$, $loc(B) \notin \bigcup_{k \in S'_{out}} G_B^U(k)$. By definition of S'_{out}, given j' such that $loc(B) \in G_B^U(j')$, $mindist(G_A^U(i'), G_B^U(j')) > \delta_A$. Hence, by definition of the *Hide&Seek* protocol, the protocol returns the "B is in proximity" message.

Case (c) directly follows from the definition of the *Hide&Seek* protocol.

A.4 Proof of Theorem 2

Proof. During each execution of the *Hide&Seek* protocol the SP does not receive any additional information with respect to the execution of the *SP-Filtering* protocol. Hence, from Theorem 1, it follows that there exists at least one granule of G_C^{SP} such that the SP is not able to exclude any location of that granule as possible location of C. We now prove that, for each buddy D, there exists at least one granule of G_C^U such that D is not able to exclude any location of that granule as possible location of C.

By assumption, D has no information about the location of C other than the one exchanged during the protocol. While executing the protocol, D can receive 6 messages:

(a) "C is in proximity" from the SP;

(b) "C is not in proximity" from the SP;

(c) "C is possibly in proximity" from the SP;

(d) "C is not in proximity" from C;

(e) "C is in proximity" from C;

(f) "starting two-parties protocol $\langle i', \delta_C \rangle$" from C.

Given i' such that $loc(C) \in G^U_C(i')$, we prove that, for each of these messages, D is not able to exclude any location of $G^U_C(i')$ as possible location of C.

For messages (a), (b) and (c), thesis follows from Theorem 1. For message (f), thesis follows by the definition of the *Hide&Seek* protocol.

For message (d), by definition, $Area^{out}_S$ is the union of granules of G^U_A. Since, by definition of granularity, two granules of the same granularity do not overlap, if $loc(C) \in G^U_C(i')$, then it does not exists any other granule $i'' \neq i'$ such that $loc(C) \in G^U_C(i'')$. Hence, since $loc(C) \in Area^{out}_S$, and $Area^{in}_F$ is the union of granules of G^U_C, then $G^U_C(i') \subseteq Area^{in}_F$.

The proof is analogous for case (e). □

A.5 Proof of Proposition 3

We first prove, in Lemma 4, that the secure computation of the *Hide&Crypt* protocol solves the set-inclusion problem.

Lemma 4. *Let S' be a set of positive integers. Let S'' be a set of negative integers. Let j be a positive integer. Let E^* be a commutative encryption function such that, for each pair of values i and i' and each pair of key K_A, K_B,*

$$E^*_{K_A}(E^*_{K_B}(i)) = E^*_{K_B}(E^*_{K_A}(i')) \Leftrightarrow i = i'$$

Let K_A, K_B be two keys. Then,

$$E^*_{K_A}(E^*_{K_B}(j)) \in E^*_{K_B}(E^*_{K_A}(S' \cup S'')) \Leftrightarrow j \in S'$$

Proof. By defintion,

$$E^*_{K_A}(E^*_{K_B}(j)) \in E^*_{K_B}(E^*_{K_A}(S' \cup S''))$$

$$\Updownarrow$$

$$E^*_{K_A}(E^*_{K_B}(j)) \in \bigcup_{j' \in (S' \cup S'')} E^*_{K_B}(E^*_{K_A}(j'))$$

By assumption about E^*, if j is in $S' \cup S''$, then $E^*_{K_B}(E^*_{K_A}(j))$ is in $\bigcup_{j' \in (S' \cup S'')} E^*_{K_B}(E^*_{K_A}(j'))$. Vice versa, if $E^*_{K_B}(E^*_{K_A}(j))$ is in $\bigcup_{j' \in (S' \cup S'')} E^*_{K_B}(E^*_{K_A}(j'))$, then j is in $S' \cup S''$. Hence

$$j \in (S' \cup S'') \Leftrightarrow E^*_{K_B}(E^*_{K_A}(j)) \in \bigcup_{j' \in (S' \cup S'')} E^*_{K_B}(E^*_{K_A}(j'))$$

Since, by assumption, j is positive and S'' contains negative elements only, $j \notin S''$,

$$j \in S' \Leftrightarrow j \in (S' \cup S'')$$

Hence the thesis.

$\square$

We can now prove Proposition 3.

Proof. Thesis 3. follows from the fact that, by the definition of the *Hide&Crypt* protocol, A sends a "starting the secure two parties protocol" only when the SP cannot compute whether B is in the proximity of A. The SP cannot compute whether B is in the proximity of A if and only if A is located in $Area_C^{passive}$ (the proof is analogous to the proof of Lemma 3).

In order to prove thesis 1. and 2., we show that:

(a) A can compute that B is in proximity as the result of the secure computation protocol with B if and only if B is located in $Area_C^{in}$;

(b) A can compute that B is not in proximity as the result of the secure computation protocol with B if and only if B is located in $Area^{out}_{C}$ The thesis follows since, A initiates the secure two parties computation only if the SP is not able to compute whether B is in proximity of A. From Lemma 3 the SP is not able to compute whether B is in proximity of A only when $loc(B) \in Area_{NoFilter}$. Hence, if A can compute that B is in proximity as the result of the secure computation protocol with B, then from result (a) above A cannot exclude that B is located in any location of $Area_{NoFilter} \cap Area^{in}_{C}$. The thesis follows since, by definition of $Area^{in}_{C}$, $Area_{NoFilter} \subseteq Area^{in}_{C}$.

The proof is analogous in case A can compute that B is not in proximity as the result of the secure computation protocol with B.

We prove case (a). Case (b) follows since if A starts the two-parties protocol, then B must be in $Area_{NoFilter}$ and $Area^{out}_{C} = Area_{NoFilter} \cap (Area^{in}_{C})^{C}$.

$\Rightarrow$) **If A can compute that B is in proximity as the result of the secure computation protocol with B then $loc(B) \in Area^{in}_{C}$.** We show that $loc(B) \in Area_{NoFilter}$ and $loc(B) \in \bigcup_{k \in S'} G^{U}_{B}(k)$.

By Lemma 3, A initiates the two-parties protocol only if $loc(B) \in Area_{NoFilter}$.

Given j' such that $loc(B) \in G^{U}_{B}(j')$ since A can compute that B is in proximity, then $j' \in S'$. Hence $loc(B) \in G^{U}_{B}(j') \subseteq \bigcup_{k \in S'} G^{U}_{B}(k)$.

$\Leftarrow$) **If $loc(B) \in Area^{in}_{C}$ then A can compute that B is in proximity as the result of the two-parties computation**

Since, $loc(B) \in Area^{in}_{C} \subseteq Area_{NoFilter}$, from Lemma 3, it follows that A initiates the secure computation with B. If $loc(B) \in Area^{in}_{C}$, then $loc(B) \in \bigcup_{k' \in S'} G^{U}_{B}(k')$. Let j' be such that $loc(B) \in G^{U}_{B}(j')$. Since, by definition of granularity, granules of the same granularity do not overlap, from $loc(B) \in \bigcup_{k' \in S'} G^{U}_{B}(k')$ and $loc(B) \in G^{U}_{B}(j')$ it follows that $j' \in S'$. Since, by Lemma 4, the two-parties protocol computes the set inclusion between j' and S', A computes that B is in proximity if and only if $j' \in S'$. Hence, since $j' \in S'$, then A can compute that B is in proximity as the result of the two-parties protocol. $\square$

A.6 Proof of Theorem 3

Proof. During each execution of the *Hide&Crypt* protocol the SP does not receive any additional information with respect to the execution of the *SP-Filtering* protocol. Hence, from Theorem 1, it follows that there exists at least one granule of G_C^{SP} such that the SP is not able to exclude any location of that granule as possible location of C. We now prove that, for each buddy D, there exists at least one granule of G_C^U such that D is not able to exclude any location of that granule as possible location of C.

By assumption, D has no information about the location of C other than the one exchanged during the protocol. From Proposition 1 and Proposition 3, it follows that, while executing the *Hide&Crypt* protocol, D can restrict the location of C to:

(a) $Area_F^{in}$, when D receives the message"C is in proximity" from the SP;

(b) $Area_F^{out}$ when D receives the message "C is not in proximity" from the SP;

(c) $Area_{NoFilter}$ when D receives the message "C is possibly in proximity" from the SP;

(d) $Area_C^{in}$ when D can compute that C is in proximity as the result of the two-parties computation;

(e) $Area_C^{out}$ when D can compute that C is not in proximity as the result of the two-parties computation;

(f) $Area_C^{Passive}$ when D receives the message "starting two-parties protocol" from C.

Given i' such that $loc(C) \in G_C^U(i')$, we prove that, for each of these cases, D is not able to exclude any location of $G_C^U(i')$ as possible location of C.

For cases (a), (b) and (c), thesis follows from Proposition 1. For case (d), (e) and (f), the proof is analogous to the one of Theorem 2. □

A.7 Proof of Lemma 1

Proof. The sought after independence intuitively means that whether an update/request is sent to SP by a user A is not related to where the user is located. Formally, by the definition of conditional probability, we have

$$P(ur(A)|loc_A, pri_A)$$

$$\begin{aligned} &= P(ur(A), loc_A|pri_A)/P(loc_A|pri_A) \\ &= (P(ur(A)|pri_A) * P(loc_A|pri_A))/P(loc_A|pri_A) \\ &= P(ur(A)|pri_A). \end{aligned}$$

The second equality is due to the protocol, in which an update/request is sent at fixed time intervals for each user *independent* of the user's location. Hence, the lemma follows. □

A.8 Proof of Lemma 2

Proof. All we need is

$$P(M_1|M_2, loc_A, pri_A) = P(M_1|loc_A, pri_A),$$

i.e., the knowledge of the messages in M_2 does not have any impact on the probability of messages in M_1. But this follows the perfect secrecy assumption and the use of keystreams in our protocol. □

A.9 Proof of Theorem 4

Proof. We prove the theorem by showing that for each set M of messages exchanged during the protocol, we have $P(post_A) = P(pri_A)$. That is, the messages M do not change the SP's knowledge of A's location. By assumption of the theorem, $P(post_A) = P(loc_A|M, pri_A)$ as the only knowledge is M and pri_A. The knowledge that $loc_A \in g_A$ is useless as we assume in this case that g_A is the whole spatial domain. By the definition of conditional probability, we have

$P(loc_A|M, pri_A) =$

$P(M|loc_A, pri_A) * P(loc_A|pri_A)/P(M|pri_A).$

It now suffices to show

$$P(M|loc_A, pri_A) = P(M|pri_A). \tag{A.1}$$

Intuitively, Equation A.1 says that the messages M are independent of the location of A. This follows from two observations: the first is that the issuance of messages does not depend on the location of A by Lemma 1 and the second is that the (encrypted) messages are independent of the content of the messages by Lemma 2. More formally, assume

$$M = m_1, \dots, m_n.$$

Let $ur(M)$ be the messages of the form

$$ur(m_1), \dots, ur(m_n),$$

where $ur(m_i)$ is "an update/request is sent by user B_i". That is, $ur(m_i)$ disregards the encrypted part of the message but only says that a message is sent and by whom. By perfect secrecy assumption, the probability of a particular (single) message is the same as any other (single) message that differs only in the encrypted part, and hence the same as the probability of $ur(m_i)$. Consider the case of two messages in M, i.e., $n = 2$. Now we have:

$$\begin{aligned}
P & \quad (M|loc_A, pri_A) \\
= & \quad P(m_1, m_2|loc_A, pri_A) \\
= & \quad P(m_1|m_2, loc_A, pri_A) * P(m_2|loc_A, pri_A) \\
= & \quad P(m_1|loc_A, pri_A) * P(m_2|loc_A, pri_A) \text{ by Lemma 2} \\
= & \quad P(ur(m_1)|loc_A, pri_A) * P(ur(m_2)|loc_A, pri_A) \\
& \qquad \text{by the above discussion} \\
= & \quad P(ur(m_1), ur(m_2)|loc_A, pri_A) \text{ by Lemma 2} \\
= & \quad P(ur(M)|loc_A, pri_A)
\end{aligned}$$

The above can be extended to n messages in M and also to show the equation $P(M|pri_A) = P(ur(M)|pri_A)$. Hence,

$$\begin{aligned} P & \quad (M|loc_A, pri_A) \\ = & \quad P(ur(M)|loc_A, pri_A) \\ = & \quad P(ur(M)|pri_A) \text{ by Lemma 1} \\ = & \quad P(M|pri_A) \end{aligned}$$

and the thesis is established. □

A.10 Proof of Theorem 5

Proof. Given a buddy B, we prove the theorem by showing that for each set M of messages exchanged during the protocol, we have

$$P(loc_A|M, pri_A, loc_A \in g_A) = P(loc_A|pri_A, loc_A \in g_A),$$

where A is another user, and g_A is the location information that is encrypted in the messages of A with the key shared between A and B. In other words, we want to show that B will not acquire more location information about A through the messages other than what B already knows. Intuitively, this is true since the location information revealed by A is only at the granule level, but not *where* within the granule.

The formal proof is the same as for Theorem 4 but with the following two changes: (1) $ur(m)$ represents that request was sent from the granule included in the message if the message is intended to B; otherwise, it is the same as before. (2) $loc_A \in g_A$ is included in pri_A, or equivalently we replace each occurrence of pri_A with "$loc_A \in g_A, pri_A$". Let us now examine the steps in the proof of Theorem 4.

Lemma 1 still holds since updates/requests are sent regardless of locations if the user who sent the message is $C \neq A$. If $C = A$, then the $ur(A)$ gives the location (the granule) where the message is sent. In this case, the location is totally dependent on the given information of loc_A, $loc_A \in g_A$

and pri_A. Note that l is an index of a granule, any information contained in loc_A and pri_A below the granule level is not relevant to the probability of a message.

For Lemma 2, the content in M_2 still does not have any impact on the content in M_1 even when B can decrypt the messages intended to him as there is no information (from pri_A, loc_A, and $loc_A \in g_A$) that restricts any possible content in M_1, so the conditional probability of M_1 does not change regardless the existence of M_2.

For the discussion regarding the probability of m_i and $ur(m_i)$, with the addition of $loc_A \in g_A$, we still have that the conditional probability of m_i being the same as that of $ur(m_i)$. Indeed, assume

$$m_i = \langle C, ui, E_{K^{ui}}(l)\rangle.$$

If $C \neq A$, then all messages of the type have the same probability with or without knowing A's location since C's location information is not assumed in the conditional probability. This case is exactly the same as for the SP and the conditional probability of m_i is the same as that of $ur(m_i)$. If $C = A$, since B can decrypt the message, hence knowing the location l in the message, this location l (an index value of a granule in G^U_A) needs to be consistent with the location knowledge in loc_A and pri_A: if it is not consistent, then the probability of the message is zero; otherwise, the probability is totally dependent on the probability of A being in $G^U_A(l)$ given loc_A, $loc_A \in g_A$, and pri_A. But the same can be said about $ur(m_i)$ (which says that a message was sent at the given location), i.e., the probability of $ur(m_i)$ depends totally on loc_A, $loc_A \in g_A$, and pri_A. Therefore, m_i and $ur(m_i)$ have the same conditional probability. By the same reasoning as in the proof of Theorem 4, $ur(M)$ has the same conditional probability as M.

With all the above discussions, the theorem is established. $\square$

A.11 Proof of Theorem 6

Proof. The proof follows the same style of that for Theorem 4. That is, we show $P(M|loc_A, pri_A) = P(M|pri_A)$, i.e., the location of A does not change the probability of messages M conditioned on pri_A. Like for Theorem 5, we examine the proof steps of Theorem 4 for the purpose of the current thesis. Lemmas 1 and 2 both hold due to the use of hashing function that displays stronger secrecy than encryption. The important difference is the discussion of the conditional probabilities of m and $ur(m)$. If m is an update, then the same applies as in the proof of Theorem 4. The difference is when m is a proximity request. In this case, the message contains multiple components. The critical step is to show that all such messages have the same conditional probability (to the SP) and hence the same as the conditional probability of $ur(m)$. This is not difficult since the location information in the condition is opaque to the SP. This opaqueness is given by two facts. The first is that the number of components in the message is the same regardless of the location information. The second is that the indexes of the granules and the "padding" (S'' in the protocol) in the message components are hashed and hence to the SP all possible granule indexes are equally possible in the encrypted (by K_1 in the protocol) message. (Here, hashing before encryption with K_1 is important as the adversary cannot attack using known pattern of the plaintext.) The above observations lead to the thesis of this theorem. □

A.12 Proof of Theorem 7

Proof. Intuitively, to the buddies, the *C-Hide&Hash* is much stronger than *C-Hide&Seek* since buddies only share a hashing function and the buddies location information is encrypted by a random key (generated by the SP) before sending to the requesting user B. Formally, the proof follows the same style as that for Theorem 5. The only difference is what it means when a message is "consistent" with the location knowledge. In this case, from B's perspective, we need to define $ur(m)$ to be the binary random variable that "the user is in one of the requesting granules or not" for the message sent back from the SP (as the reply to a proximity request from B). After B requesting proximity, B will receive a message from the SP with the encrypted hash value of A's location (in addition to the "kick back" from the SP in the form of encrypted values that B sent to the SP). Even though B and A shares the hash function, B does not know the encryption key which is randomly generated by the SP (K_2 in the protocol). Therefore, this value is probabilistically independent of the location of A. In this case, based on the protocol, the only information B obtains is whether A is in a granule among the ones given by B. This needs to be consistent with the location information contained in loc_A and pri_A. If not, then the probability of this message is zero, and otherwise the probability is totally dependent on loc_A and pri_A as no other information is available. The thesis follows the above discussions in the same style as the proof of Theorem 5. $\square$

Appendix B

Notation

A summary of the notation used in this dissertation follows:

- loc_A – the original (exact) location of a user A
- δ_A – the proximity threshold requested by A i.e. the minimum distance required to consider another user being in proximity
- R – the minimum uncertainty region (MUR) in which a user does not want an adversary to exclude any of the points as possible location
- M – the set of exchanged messages
- G_A^{SP} – the spatial granularity selected by a user A as privacy preference, with respect to the service provider
- G_A^U – the spatial granularity selected by a user A as privacy preference, with respect to the other users
- Gr_A^U – the grid-based spatial granularity selected by a user A as privacy preference, with respect to the other users
- $L_A(i)$ – given the granule of G_A^{SP} with index i, the region given by the union of granules of G_A^U intersecting $G_A^{SP}(i)$
- E_{K_A} – E is a symmetric encryption function, using K_A as encryption key.

- $E^*_{K_A}$ – E^* is a commutative encryption function, using K_A as encryption key.
- H_{K_A} – H is an hashing function, using K_A as salt.
- $d(loc_A, loc_B)$ – the Euclidean distance between the locations of users A and B
- pri_A – random variable denoting the distribution of the location of a user A. This distribution is known a-priori by the adversary
- $post_A$ – random variable denoting the distribution of the location of a user A. This distribution is computed a-posteriori by the adversary i.e. after considering the set M
- *mindist* – the minimum distance between two regions
- *maxdist* – the maximum distance between two regions
- *moddist* – the modular distance between two points i.e. the Euclidean distance computed as if the considered finite spatial domain were “circular” on both axis
- *mmd* – the minimum modular distance between two regions

Bibliography

[1] Arnon Amir, Alon Efrat, Jussi Myllymaki, Lingeshwaran Palaniappan, and Kevin Wampler. Buddy tracking - efficient proximity detection among mobile friends. *Pervasive and Mobile Computing*, 3(5):489–511, 2007.

[2] Claudio A. Ardagna, Marco Cremonini, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Location privacy protection through obfuscation-based techniques. In *Proc. of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, volume 4602 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2007.

[3] Alastair R. Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *Proc. of the 2nd Annual Conference on Pervasive Computing and Communications*, pages 127–131. IEEE Computer Society, 2004.

[4] Claudio Bettini, Sushil Jajodia, and Linda Pareschi. Anonymity and diversity in LBS: a preliminary investigation. In *Proc. of the 5th International Conference on Pervasive Computing and Communications*, pages 577–580. IEEE Computer Society, 2007.

[5] Claudio Bettini, Sergio Mascetti, X. Sean Wang, Dario Freni, and Sushil Jajodia. Anonymity and historical-anonymity in location-based services. In *Privacy in Location-Based Applications*, volume 5599 of *Lecture Notes in Computer Science*. Springer, 2009.

[6] Claudio Bettini, Sergio Mascetti, X. Sean Wang, and Sushil Jajodia. Anonymity in location-based services: towards a general framework. In *Proc. of the 8th International Conference on Mobile Data Management*, pages 69–76. IEEE Computer Society, 2007.

[7] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *Proc. of the 2nd VLDB workshop on Secure Data Management*, volume 3674 of *LNCS*, pages 185–199. Springer, 2005.

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

[9] Matt Bishop. *Computer Security: Art and Science*, chapter 32. Addison-Wesley, 2003.

[10] Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[11] Hae Don Chon, Divyakant Agrawal, and Amr El Abbadi. Range and knn query processing for moving objects in grid model. *Mobile Networks and Applications*, 8(4):401–412, 2003.

[12] Chi-Yin Chow and Mohamed Mokbel. Enabling private continuous queries for revealed user locations. In *Proc. of the 10th International Symposium on Spatial and Temporal Databases*, pages 258–275. Springer, 2007.

[13] Chi-Yin Chow, Mohamed F. Mokbel, and Xuan Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proc. of the 14th International Symposium on Geographic Information Systems*, pages 171–178. ACM, 2006.

[14] Alin Deutsch, Richard Hull, Avinash Vyas, and Kevin Keliang Zhao. Policy-aware sender anonymity in location based services. In *Proceed-*

ings of the 26th IEEE International Conference on Data Engineering (ICDE 2010), 2010.

[15] Matt Duckham and Lars Kulik. A formal model of obfuscation and negotiation for location privacy. In *Proc. of the 3rd International Conference, on Pervasive Computing*, pages 152–170. Springer, 2005.

[16] Dario Freni, Sergio Mascetti, Claudio Bettini, and Marco Cozzi. `Pcube`: A system to evaluate and test privacy-preserving proximity services. In *Proc. of the 11th International Conference on Mobile Data Management*, 2010.

[17] Dario Freni, Carmen Ruiz Vicente, Sergio Mascetti, Claudio Bettini, and Christian S. Jensen. Preserving location and absence privacy in geo-social networks. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010)*, 2010.

[18] 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.

[19] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *Proc. of ACM International Symposium on Advances in Geographic Information Systems*, pages 246–255. ACM Press, 2009.

[20] 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*, pages 121–132. ACM Press, 2008.

[21] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Mobihide: A mobile peer-to-peer system for anonymous location-based queries. In *Proc. of the 10th International Symposium of Advances in Spatial and Temporal Databases*, pages 221–238. Springer, 2007.

[22] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Prive: anonymous location-based queries in distributed mobile systems. In *Proc. of the 16th international conference on World Wide Web*, pages 371–380. ACM Press, 2007.

[23] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of the 1st International Conference on Mobile Systems, Applications and Services*, pages 31–42. The USENIX Association, 2003.

[24] Marco Gruteser and Xuan Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security & Privacy*, 2(2):28–34, 2004.

[25] Baik Hoh and Marco Gruteser. Protecting location privacy through path confusion. In *Proc. of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 194–205. IEEE Computer Society, 2005.

[26] Haibo Hu and Jianliang Xu. Non-exposure location anonymity. In *Proc. of the 25th International Conference on Data Engineering*, pages 1120–1131. IEEE Computer Society, 2009.

[27] 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.

[28] Ali Khoshgozaran and Cyrus Shahabi. Private buddy search: Enabling private spatial queries in social networks. In *Symposium on Social Intelligence and Networking (SIN 2009)*, 2009.

[29] Ali Khoshgozaran, Cyrus Shahabi, and Houtan Shirani-Mehr. Location privacy: going beyond k-anonymity, cloaking and anonymizers. *Knowledge and Information Systems*, 2010.

[30] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. Protection of location privacy using dummies for location-based services. In *Proc. of the 21st International Conference on Data Engineering Workshops*, page 1248. IEEE Computer Society, 2005.

[31] 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.

[32] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, page 24. IEEE Computer Society, 2006.

[33] Miquel Martin and Petteri Nurmi. A generic large scale simulator for ubiquitous computing. In *Proc. of the 3rd Conference on Mobile and Ubiquitous Systems: Networks and Services.* IEEE Computer Society, 2006.

[34] Sergio Mascetti, Claudio Bettini, and Dario Freni. Longitude: Centralized privacy-preserving computation of users' proximity. In *Proc. of 6th VLDB workshop on Secure Data Management*, Lecture Notes in Computer Science. Springer, 2009.

[35] Sergio Mascetti, Claudio Bettini, Dario Freni, and X. Sean Wang. Spatial generalization algorithms for LBS privacy preservation. *Journal of Location Based Services*, 1(3):179–207, 2007.

[36] Sergio Mascetti, Claudio Bettini, Dario Freni, X. Sean Wang, and Sushil Jajodia. Privacy-aware proximity based services. In *Proc. of the 10th International Conference on Mobile Data Management*, pages 31–40. IEEE Computer Society, 2009.

[37] Sergio Mascetti, Claudio Bettini, X. Sean Wang, Dario Freni, and Sushil Jajodia. *ProvidentHider*: an algorithm to preserve historical k-

anonymity in lbs. In *Proc. of the 10th International Conference on Mobile Data Management*, pages 172–181. IEEE Computer Society, 2009.

[38] Sergio Mascetti, Dario Freni, Claudio Bettini, X. Sean Wang, and Sushil Jajodia. On the impact of user movement simulations in the evaluation of LBS privacy-preserving techniques. In *Proc. of the International Workshop on Privacy in Location-Based Applications*, volume CEUR-WS Vol-397, pages 61–80, 2008.

[39] Sergio Mascetti, Dario Freni, Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *The VLDB Journal*, 2011.

[40] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: query processing for location services without compromising privacy. In *Proc. of the 32nd International Conference on Very Large Data Bases*, pages 763–774. VLDB Endowment, 2006.

[41] Stavros Papadopoulos, Spiridon Bakiras, and Dimitris Papadias. Nearest neighbor search with strong location privacy. In *Proceedings of the 36th International Conference on Very Large Data Bases (VLDB 2010)*, 2010.

[42] Daniele Riboni, Linda Pareschi, Claudio Bettini, and Sushil Jajodia. Preserving anonymity of recurrent location-based queries. In *Proc. of 16th International Symposium on Temporal Representation and Reasoning*. IEEE Computer Society, 2009.

[43] Peter Ruppel, Georg Treu, Axel Küpper, and Claudia Linnhoff-Popien. Anonymous user tracking for location-based community services. In *Proc. of the Second International Workshop on Location- and Context-Awareness*, volume LNCS 3987, pages 116–133. Springer, 2006.

[44] Peter Saint-Andre. Extensible messaging and presence protocol (XMPP): core. RFC 3920, IETF, 2004.

[45] Carmen Ruiz Vicente, Dario Freni, Claudio Bettini, and Christian S. Jensen. Location-related privacy in geo-social networks. *IEEE Internet Computing*, 2011, to appear.

[46] Simonas Šaltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. *SIGMOD Rec.*, 29(2):331–342, 2000.

[47] Laurynas Šikšnys, Jeppe R. Thomsen, Simonas Šaltenis, and Man Lung Yiu. Private and flexible proximity detection in mobile social networks. In *Proc. of the 11th International Conference on Mobile Data Management*, pages 75–84, 2010.

[48] Laurynas Šikšnys, Jeppe R. Thomsen, Simonas Šaltenis, Man Lung Yiu, and Ove Andersen. A location privacy aware friend locator. In *Proc. of the 11th International Symposium on Spatial and Temporal Databases*, volume 5644 of *Lecture Notes in Computer Science*, pages 405–410. Springer, 2009.

[49] Nikolay Vyahhi, Spiridon Bakiras, Panos Kalnis, and Gabriel Ghinita. Tracking moving objects in anonymized trajectories. In *Proc. of 19th International Conference on Database and Expert Systems Applications*, pages 158–171. Springer, 2008.

[50] Toby Xu and Ying Cai. Location anonymity in continuous location-based services. In *Proc. of ACM International Symposium on Advances in Geographic Information Systems*, page 39. ACM Press, 2007.

[51] 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*, pages 366–375. IEEE Computer Society, 2008.

[52] Man Lung Yiu, Leong Hou U, Simonas Šaltenis, and Kostas Tzoumas. Efficient proximity detection among mobile users via self-tuning poli-

cies. In *Proceedings of the 36th International Conference on Very Large Data Bases (VLDB 2010)*, 2010.

[53] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, Lester and Pierre: Three protocols for location privacy. In *Privacy Enhancing Technologies*, volume LNCS 4776, pages 62–76. Springer, 2007.