Privacy challenges of open APIs: case LBS

Alberto Vila Tena
Helsinki University of Technology
alberto.vilatena@tkk.fi

Abstract

Many of the most popular Location-Based Services(LBSs) have recently released open APIs for developers in order to extend their services to third party applications. With this strategy, LBSs try to exploit to the maximum the value of location information by helping the fast development of specific context-aware applications involving location. However, protecting location privacy has always been the main challenge for LBSs and despite all the research dedicated to this aspect, it has been yet impossible for them to define standardized ways to protect users' location privacy. In this context in which ideas about protecting location data are still unclear, open APIs for LBSs appear offering new ways for location data to be distributed. Therefore, now the challenge of protecting location privacy is even bigger. This paper reviews how different existing LBSs aim to protect location privacy and different techniques of protecting location data, and discusses which directions LBSs and their open APIs could take in order to respond to the privacy challenges that they have to face.

1 Introduction

Open APIs (Application Programming Interfaces) are currently boosting the interaction within different websites and the Internet. Every day more and more web applications are using and offering open APIs because of the big advantages they offer. By publishing an open API, a website converts its services into public available resources, which are open to any developer or any other site in the Internet. This has as a consequence that sites and developers are able to easily improve and complement the services that these open APIs offer, creating this way more advanced and complex services, and, accordingly, a better user experience. Location Based Services (LBSs) are following the same way. Initially, we used different LBSs for concrete and different goals, such as exploring our city or sharing punctually our location with friends. But lately many of these LBSs have released their open APIs, and new opportunities due to the interaction of LBSs with other LBSs or different web services have appeared[15][5]. Synchronizing a single location update with all the LBSs that we are using so we can receive real time services from all of them, or sharing our location with our contacts in our favourite social network are for example some of the features that we can enjoy thanks to the implementation of open APIs in LBSs. Another powerful reason to favour the interaction of LBSs with other services is the wide range of contexts in which LBSs could be used. Even though leisure is the context of the most popular LBSs, they could also offer interesting features for domains such as health, transport, work, etc.

However, since the concept of LBSs started to appear, there have been great privacy concerns regarding them. LBSs deal with location data, which is, by nature, a very sensitive and complex information item. While a single piece of location data about a user may be irrelevant and not compromise her privacy, a big collection of them certainly reveals many clues about a user's life, being able to reveal information such as where she lives, where she works, in which areas of the city does she go out, which are her favourite stores, which buildings does she visit and how often, etc. In other words, a big collection of location data from a single user is so connected with her daily life and routines that it makes possible to infer many details of her identity, habits and lifestyle. Users typically consider some of this information private, and that is why protecting location data is a priority and a big challenge for LBSs, and the recent release of open APIs in LBSs just makes the challenge more difficult..

This paper studies some of the recently appeared location open APIs to determine which challenges should they face in order to preserve their users' location privacy and presents a collection of ideas to respond to these challenges. The paper is structured as follows: The next section reviews some of the current open APIs in LBSs. Section 3 identifies privacy challenges that current location open APIs present. Solutions to these privacy challenges are presented in Section 4. Section 5 discusses how to apply the presented solutions in order to achieve more secure LBSs and location Open APIs and, finally, Section 6 concludes the paper.

2 Current Location Open APIs

This section reviews some of the currently existing location APIs. The following five examples show different ways in which APIs deal with location data.

2.1 Telecom APIs

Mobile operators were behind the first generation of LBSs, a bit before the emergence of Web 2.0. Basically, operators made use of cell-id positioning using triangulation techniques to provide their clients with really simple services[3]. These services did not succeed, and soon GPS (Global Positioning System) and Web 2.0 went completely over them, causing the appearance of a new generation of LBSs, created by companies and developers independently of mobile operators. However, some mobile operators have recently released APIs offering their services to developers in order to open ways for new business models, and location is within these services. Two examples of these APIs are GSMA OneApi[1] and Innovation World Developer[2]. The first one is a global initiative and its first pilot is taking place in Canada involving the three main operators there: Bell, Rogers and Telus. Innovation World Developer is, instead, a contest organized by the Finnish operator TeliaSonera in which they provided a set of REST (Representational State Transfer) APIs to developers in order to facilitate the creation of new web applications and mashups.

To use these APIs, a developer should register to them providing a user name, a password and a mobile phone number. Then, the API provider checks that the given mobile phone belongs to the developer by sending a validation code to her phone and, after this step, the registration is complete and the developer is able to use the APIs. Once registered, the developer is given a user key and a service key, which are the credentials she should use in order to invoke the REST APIs. Both keys are random character strings. The user key identifies the developer and is periodically renewed after a certain time. To get her current user key the developer should call a login API with her user name and password. A service key, instead, identifies an application in which the user invoking the APIs. A developer is able to create as many service keys as she wishes and these keys are permanent and accessible from the developer's profile. In the case of the location API, the developer should authorize her device to be tracked by the operator first by sending an SMS, and after this step, she is able to execute the API normally by sending requests using her user key and one of her service keys. To every request the API replies with a XML or JSON message containing the location data corresponding to the requestor's mobile phone. Finally, location tracking of a mobile device could be deactivated in the same way as it is activated, with an SMS.

2.2 Yahoo! Fire Eagle

Yahoo! Fire Eagle[3] was the first location broker. This definition comes from its function. Fire Eagle collects the user's location data from different sources and forwards it to third party applications, which are the ones which provide different services regarding the given location. Then, Fire Eagle assumes that a regular user may use different LBSs and, for this reason, its main purpose is to offer the users a central point to store their location information and from which to update their location in all the LBSs that they are using with a single operation.

Fire Eagle exchanges location information with third party applications through its REST API. This API allows to a third party application to update a user's location to Fire Eagle or to extract the location of a single user or of a set of users from Fire Eagle. In order to perform any of these operations over a user, a third party application must be authorized by the user to do it. Fire Eagle uses the OAuth protocol for authentication and authorization. Following this protocol, when a user decides to authorize a third party application to update or extract her location information, she provides the application with an authorization token, and with this token the application is able to make requests to Fire Eagle's API on behalf of the user. In other words, the user authorizes the third party application to update or to extract her location information in Fire Eagle with the token. When a user authorizes a third party application to use her location data in Fire Eagle, she is able to specify the accuracy with which she wants her location to be transmitted. Fig. 1 shows a sample of the different levels of accuracy in Fire Eagle, which are postal code, city, region, state and country. Finally, it is also remarkable the fact that a user is able to remove at any moment all the location data about her that is stored in Fire Eagle.

2.3 Foursquare

Foursquare[4] combines social networking, location and gaming. Users get points by checking-in at venues and, by accumulating certain amounts of points, they are able to unlock different badges which distinguish them with different levels in the application according to their activity, such as "newbie", "explorer", "superstar", etc. Also, users are able to become the "mayors" of certain venues if they check-in frequently at them. Also, Foursquare works as a social network, because users are able to share their check-ins and achievements in the game with their friends and also with other users of the application which are not their friends. Concretely, users are able to show their check-ins just to friends, but, if they wish to participate in the game, they should comply in showing their achievements in public. Also, Foursquare is popular for its integrated interaction with the two most well-known social networks, Twitter and Facebook.

Foursquare's API for developers offers a wide range of operations related with the users' check-ins, profiles and check-in histories. Also it offers a set of operations related to venues, allowing categorizing them, giving tips related to them, etc. To allow third party applications to request the users' authorization to execute these API operations, Foursquare recommends OAuth, even though these operations are also accessible through other protocols such as XAuth or simple authentication. All the location data that could be extracted from Foursquare API has no levels of granularity regarding its accuracy, so it is always shared with the best accuracy possible.

2.4 Google Latitude

Google Latitude[5] started as a location sharing social network integrated with Google Maps[6]. Google Latitude allows live tracking both for mobile phones and laptops, and uses this option by default, so when a user has Latitude activated, her friends are able to see where she is in a map, unless she blocks them or turns the live tracking feature off. If the user does not feel comfortable with being permanently tracked and decides to turn this feature off, she is able to update her

[1]http://oneapi.aepona.com/
[2]http://developer.medialab.sonera.fi
[3]http://fireeagle.yahoo.net/
[4]http://foursquare.com/
[5]www.google.com/latitude
[6]http://maps.google.com/

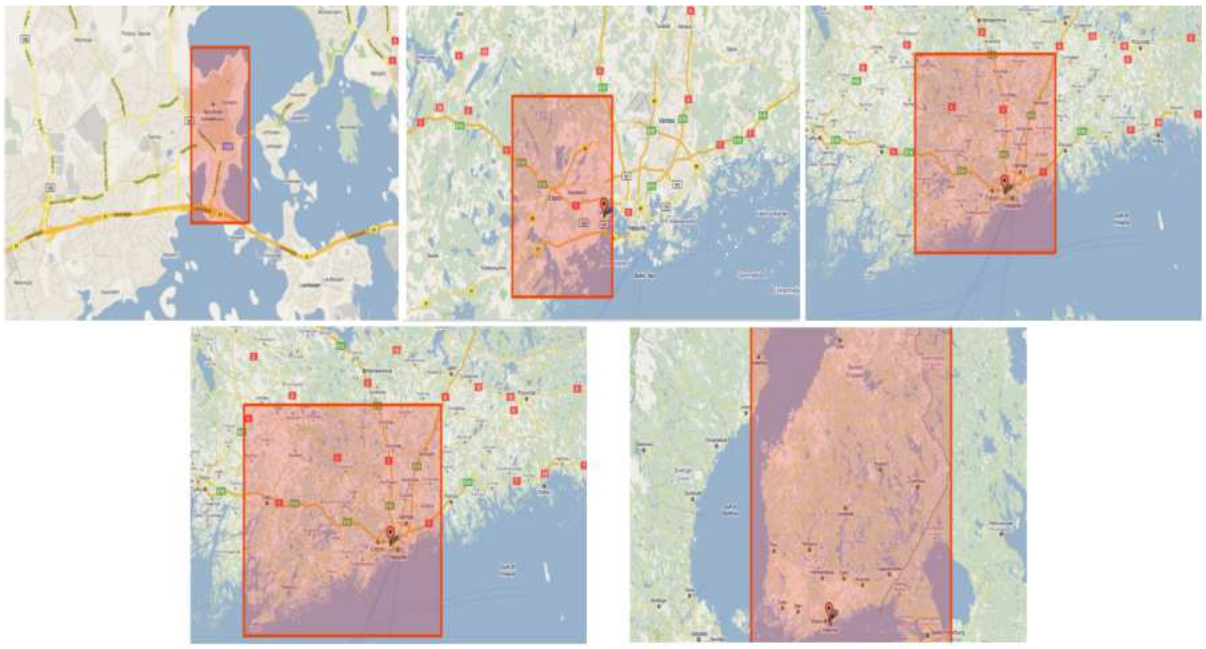

Figure 1: The 5 levels of accuracy in FireEagle

location in latitude by manually typing in a web form the address where she is. Later, Google Latitude features were complemented by a location history that records the different locations of the user in time. With the location history, Latitude intends to offer the user stats and information about her movements and locations, and also an intelligent system of alerts, through which the user receives an e-mail or SMS notification about the proximity of friends on a circumstance in which she may like to meet friends around.

In May 2010 Google Latitude released its API, giving developers access to its location features and data. The result is an API which is similar to the Fire Eagle's one. OAuth is the protocol through which a third party application obtains the authorization of a user to update or extract her location data in Google Latitude. Also, the API also works with the location history, and with an OAuth token a user is able to authorize a third party application to insert, list or extract location data from the history. When a user authorizes a third party application the access to her location data, she is able to choose within two levels of granularity, city and best, for the accuracy of the accessed location.

2.5 Simple Geo

SimpleGeo[7] is a project in development that aims to provide a scalable, cloud-based interface for storing, managing and querying location data. SimpleGeo offers, as a main feature, a cloud storage service adapted for location data. Location data is represented in this storage system by records. A record is a data type that specifies the location of a place, a person or related to a multimedia item. Every stored record should belong to a layer, hence, a layer is equivalent to a folder (or a bucket in Amazon S3 terminology) that contains a collection of records. When developers have signed up in Simple Geo, they are able to create their own layers to store the location data of their applications. Finally, SimpleGeo also offers an interface giving developers the option of selling their layers to other developers, and also, the possibility of buying other developers' location data collections. Currently, and despite SimpleGeo is still in a beta development phase, services such as Flickr are already offering their location data in SimpleGeo.

In addition to the storage service, SimpleGeo offers also a REST API to developers to interact and make queries to it. These queries allow actions such as inserting a new record, updating a record, get a record's location history or make nearby queries to check records located near a given geographical point or area. Besides, other services are available such as reverse geocoding, identifying the administrative boundaries of a point or getting the population density of a certain point through SpotRank. Every REST query should be authenticated using OAuth tokens.

3 Privacy Challenges

Almost all current LBSs could be divided into two categories[8]. The first ones are the called reactive LBSs, which wait for an update from the users to offer them context-aware services. The others are the proactive ones, which receive live tracking location from the users and send them alerts about the proximity of interesting places or friends. Each of these categories has so different privacy implications. In reactive services, the user is the one who voluntarily discloses her location with a concrete finality. This means that the user is confortable with sharing her location, on the condition that this location reaches its intended recipients. Then, the only privacy measurement needed is to avoid the location reaching an undesired recipient. On the other hand, proactive LBSs require less work from the users, as

[7]www.simplegeo.com

they do not need to update their location every time they want to receive a service, but also create bigger privacy problems. In them, users get permanently tracked, so the probability of disclosing a location about which they do not feel comfortable gets increased. In addition to this, live tracking allows to infer much more about users' lives and habits through location than occasional check-ins, a fact that may increase privacy concerns in users. Also, it is remarkable that LBSs involving live tracking usually run in background and most of the times, the user does not notice this. This fact reduces the control that the user has over the application, because with time, the user may forget that she is tracking her position permanently. Also, this opens the door for attackers to stalk their victims in some cases. For example, a jealous husband has many chances to reach her wife's phone for a while and activate Google Latitude or the location tracking feature of an operator's API; or a boss could give company phones to their employees with a live location tracking service already activated. Finally, many users may also consider undesired alerts intrusive. Therefore proactive LBSs not only need advanced privacy controls but also a really accurate system of alerts.

Independently of their nature, LBSs should address also a set of common challenges.

3.1 Effective Privacy Policies

A big challenge regarding LBSs is the implementation of effective privacy policies. Even though there are many different ways to protect location privacy, as surveys such as the one by Scipioni and Langhenrich[9] indicate, user-defined privacy policies have become by far the method chosen in practice by the vast majority of the current LBSs. However, user-defined privacy policies present for LBSs the same problems than they do for social networks. So, if they are so simple, they will have big chances to disclose undesired information. On the other hand, if they are complicated, they will take a big amount of work from the users, in addition to their more than possible need of being updated with the time. Also, the appearance of Open APIs and the interaction of LBSs with other services, especially, social networks adds a new difficulty for user-defined privacy policies. In this case the users not only need to configure their privacy policies in their original LBS, but also they should not forget about their privacy policies in the social network where they have decided to share their location. This accumulation of different user-defined privacy policies in different services favours data leaks.

3.2 Features vs. Privacy

Also, API providers, in their emphasis of providing their services to developers, forget sometimes about their users' privacy. An illustrating example of this is in Foursquare's API. Foursquare, in their API documentation[1] make the following recommendation for authentication "OAuth is the method we strongly encourage you to use so that clients do not have to hang on usernames and passwords but can initiate requests on a user's behalf via a special token". However, there are API methods, such as "checkins" and "user" that, when requested, reveal respectively the requestor's friends' most recent check-ins, and the requestor's user profile, including her friends and their details, and within these details it is possible to find contact information such as e-mails and phone numbers. Therefore, if a user authorizes a third party application to make API requests to these methods on her behalf, the third party application is gaining access to the recent locations and contact information of the user's friends, who have not given any authorization for this. This example shows that avoiding oversharing without losing attractive is a challenge for API providers.

3.3 Authorization and Authentication

Authorization and authentication are also a critical factor for keeping the users' privacy in Open API because they prevent against identity theft. Currently, as we have seen before, most part of the open APIs make use of OAuth tokens for authentication and authorization. With OAuth, users are able to authorize external applications to make certain requests to an API on their behalf without revealing their credentials at all. However, in many cases, OAuth tokens have validity for long periods of time. As a consequence, if the external application stored the access token when it was authorized by the user, it will be able to make API requests on behalf of the user for a long period of time. So these long-lasting tokens may be an inconvenience for the occasional user that requests the third-party application for a single and punctual operation from time to time. Despite this inconvenience, there are other ways of authorization and authentication that suppose a much bigger risk, as for example the way Foursquare makes use of XAuth[2] or the way in which the user key is obtained in TeliaSonera's Innovation World Developer. In both cases, the user should introduce her user name and password in the third party application to produce an access token or a user key and, even though the third party application should dismiss these credentials and store the token or the key instead, there is no technical impediment for it to store the user's credentials too.

3.4 Data Ownership

Data ownership, as the past shows[11], represents also some of the privacy concerns of Internet users. It is true that all the applications we have reviewed in this paper try to give control to their users regarding their stored location data. However, the open APIs include a new party in the game, the third party applications. These third party applications receive the results to their API requests in XML or JSON responses, being perfectly able to store the locations or other information in these responses in their databases. In these cases, users lose the control they initially had over their information in the API providers' original services and third party applications are not obliged at all to offer them a similar deal.

3.5 Other Risks

Finally, very powerful solutions for processing and data analysis of big amounts of location data, as for example it was SimpleGeo, are appearing all around the World Wide Web.

Even though these solutions are able to offer significant improvements and new features to current LBS, they could also become good allies for attackers. Friedland and Sommer show good examples of this on their research paper Cybercasing the Joint: On the Privacy Implications of Geo-Tagging[6].

4 Privacy Preserving Solutions

This section shows possible solutions for the given privacy problems.

4.1 Geo-fences

Geo-fences[10](Fig. 2) consist in virtual areas defined over real world geographic areas. New ideas are considering geo-fences as the key elements of solutions that try to be a middle point between manual location updates and live-tracking. In these solutions, users define various geo-fences in which they wish their location to be updated to the LBS. This way, the probabilities of updating a compromising location to a LBS get significantly reduced compared to live-tracking solution, and, on the other hand, the user is able to receive alerts and services in LBSs in those contexts in which he is more eager to do it. Also, geo-fences enable a new feature that may be interesting in some cases, as they are not only able to determine where a user is located, but also if the user is entering the area or leaving it. A new service involving this use of geo-fences is Neer[8].

4.2 Implicit Authorization

Implicit Authorization[13] is a technique oriented to Location Sharing Social Networks that has reciprocity as its central element. This reciprocity is based in the following principle: An inquirer A is granted access to the location information of target B, only if B in turn has previously attempted to access the location information of A. This means that location information is only shared if there is trust and mutual interest between the two users involved in the communication. The whole implicit authorization process works as follows(Fig. 3):

1. An inquirer A requests a target B's location.

2. As B has not required A's location first, A receives an error. However, B is automatically given permission to access A's location for an established time.

3. B executes her acquired permission and requests A's location.

4. B receives A's location and, automatically, A receives permission to access B's location for an established time.

5. A and B are able to share location information freely till one of them does not access the other's location during the time established by the permission.

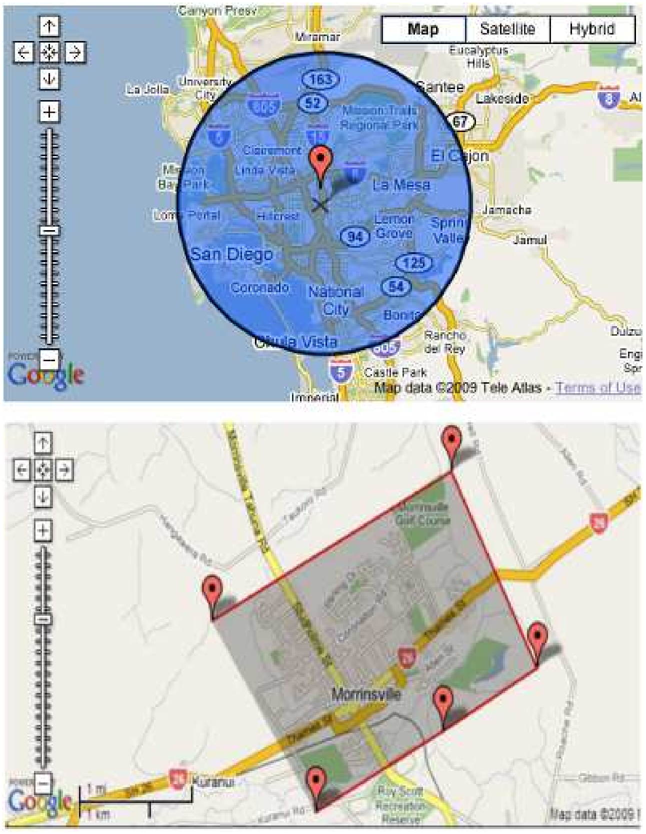

Figure 2: Examples of geo-fences

In conclusion, implicit authorization is a solution aimed to satisfy those users that expect reciprocity in their use of location sharing social networks, and its biggest attractive is that it helps to reduce possible abuses of the system, such as stalking, without demanding any effort from the users. As a consequence, implicit authorization is an interesting solution for Location Sharing Social Networks involving live-tracking location.

4.3 Improving Privacy Policies

The vast majority of the current LBSs rely on user-defined privacy policies to protect their users' privacy. Concretely, users are able to decide with whom they decide to share their location and with which accuracy. However, these configurations do not help to avoid completely the risk of disclosing a compromising location to an undesired recipient. Different studies by the Carnegie Mellon University[4][14] have determined that adding other factors to privacy policies such as time and nearby locations to privacy policies significantly improves the feeling of security in the users. All these studies got reflected in Locaccino[12][9]. However, this approach has also its inconveniences, as the elaboration of privacy policies become more complex for the users. This increasing complexity of privacy policies is also a problem that research is trying to face, for example, the same research group in Carnegie Mellon University proposes a solution based in incremental privacy policies generated by machine learning

[8]http://www.neerlife.com/

[9]www.locaccino.org

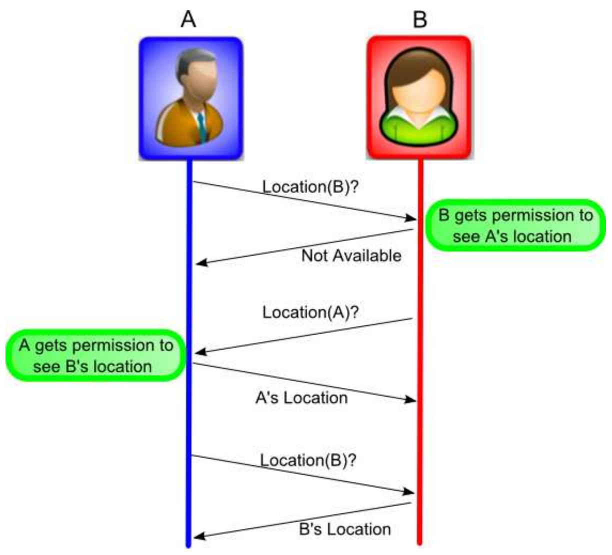

Figure 3: Implicit authorization schema

from the users' feedback[7].

4.4 Other Possible Solutions

There are also other different solutions that would help to solve the privacy problems described in Section 3. First, enabling visual alerts is a simple solution to allow users to know that a live-tracking location service is running in background in their mobile devices, which was one of the problems of Google Latitude. Identically, periodic SMS alerts from operators would allow users to remember that they have their phones being tracked.

Also, it is necessary for API authorization procedures to use protocols such as OAuth, which do not involve the users' authentication credentials in the process. Regarding OAuth, enabling access tokens for short periods of time is a solution that enforces the privacy of those users that may request punctual actions from a third party application, as they translate the punctuality of the operation also to its authorization.

Prevent the users about the risks of oversharing is also a simple solution that may solve many privacy problems. Currently, LBSs have usually really intuitive tutorials for the users, and to include alerts and advices about oversharing in them or even in other parts of the application may make the users more aware about the problems of oversharing.

Finally, a problem that is difficult to share in open APIs associated to LBSs is the data ownership one. API providers could include terms of use prohibiting third party applications to store the location information that they are getting from them, or to acquire the compromise of letting the users manage their stored location data. Even though, solutions like these may also discourage possible developers from using the APIs.

5 How to effectively preserve privacy?

It is easy to come to a curious conclusion after reviewing different LBSs and their open APIs, which is that, regarding their user-defined privacy policies, they give the same treatment to people that to third party applications. Users define in their privacy policies who they want to share their location with and with which accuracy, and identically, in those LBSs with open APIs, users are able to decide with which associated third party applications they want to share their location and with which accuracy. However, sharing location data with another people has far away different implications than doing it with an application.

When we share our location with people, we usually do it with someone with whom we have some kind of social binding, and it is around this social binding where all the privacy concerns arise. Thoughts like 'how this person would react if she knows that I have been in this place?' or similar are usually behind our privacy concerns and our defined policies. Also, this social binding usually demands a certain reciprocity, which means that, usually, if we trust something about us to someone, we usually expect to receive the same from the person in which we have trusted. What means when a user decides to share her location with a friend, she expects mainly her friend to do the same. Finally, people are unpredictable and we cannot control when our friends are going to need our location. On the contrary, when dealing with an application, we do not have any social binding with it and the reciprocity we expect is completely different, as we just expect to receive a concrete service regarding our location. In this case, privacy policies are more a tool to tailor a service to our needs taking into account the reasonable concerns that sharing our location with a completely unknown party involves.

Usually, third party applications that make use of LBSs' open APIs have really concrete finalities and that, with the appropriate privacy controls probably helps for the elaboration of effective, long-lasting privacy policies. Enabling the use of geo-fences for the implementation of user-defined privacy policies could be a very interesting feature, because it gives us the chance of associating a third party application with the area in which it would potentially offer us the best service. We can think in a bar recommendation system as an example. For this case, we could trace geo-fences around the different bar areas in our city, so every time we enter one of these areas, we allow our location to be sent to the service, and we receive the bar recommendations in our phone.

However, in what concerns to location sharing social networks, it gets more difficult to define effective and long lasting privacy policies, as the contexts and the reasons why we get our location requested are much more diverse. That is why new approaches involving reciprocity, such as Implicit Authorization, look very interesting for this sort of services, because they represent the social nature that motivates the exchange of information in these services better than a long list of user-defined privacy policies. This does not mean that user-defined privacy policies should be completely ignored in location sharing social networks, but I think that they should carry a complementary role for techniques such as Implicit Authorization more than the main role that they have right now in the privacy mechanisms of location sharing social networks.

In conclusion, I consider that LBSs should take different approaches when sharing their user's location data accordingly to the destination of this information. If the recipients are other people, one of the best ways LBSs have to protect privacy is to ensure reciprocity in the shared information. However, if the recipient is a third party application providing a certain service, LBSs should facilitate the necessary tools for the users to describe the circumstances in which they will be potentially using this service.

6 Conclusions

This paper provides an overview of several open APIs recently released by different LBSs. Examining the different mechanisms in these APIs for protecting location privacy reveals that providing user-defined privacy policies regarding the third party application is the solution chosen by most API providers, together with enabling authorized API calls on behalf of the users through protocols such as OAuth. However, these mechanisms are still far from perfect in their purpose of avoiding undesired disclosures of location data. Privacy policies are frequently difficult to define for users, and now that users should not only define them in regard to their friends, but also in regard of these applications associated to an API that they want to use, they get more difficult to implement and also to memorize. On the other hand, OAuth succeeds in protecting users' credentials, but it is also sensitive to possible abuses mainly derived from too generous authorizations to third party applications. The paper finally considers new approaches such as geo-fences and implicit authorization interesting, because they present features that could help for the future deployment of more accurate and, at the same time, simpler privacy policies.

References

[1] Foursquare api documentation. `http://www.google.com/url?url=http://groups.google.com/group/foursquare-api/web/api-documentation`.

[2] Foursquare oauth howto. `http://groups.google.com/group/foursquare-api/web/oauth`.

[3] P. Bellavista, A. Küpper, and S. Helal. Location-based services: Back to the future. *IEEE Pervasive Computing*, 7(2):85–89, 2008.

[4] M. Benisch, P. Gage, K. N. Sadeh, and L. F. Cranor. Capturing location-privacy preferences: Quantifying accuracy and user-burden tradeoffs, 2010.

[5] A. DuVander. Foursquare api fuels third-party app that may be a better foursquare, March 2010. `http://blog.programmableweb.com/2010/03/30/foursquare-api-fuels-third-party-app-that-may-be-a-better-foursquare/`.

[6] G. Friedl and R. Sommer. Cybercasing the joint: On the privacy implications of geo-tagging, 2010.

[7] P. G. Kelley, P. Hankes Drielsma, N. Sadeh, and L. F. Cranor. User-controllable learning of security and privacy policies. In *AISec '08: Proceedings of the 1st ACM workshop on Workshop on AISec*, pages 11–18, New York, NY, USA, 2008. ACM.

[8] A. Kupper, G. Treu, and C. Linnhoff-Popien. Trax: A device-centric middleware framework for location-based services, September 2006.

[9] M. P. Scipioni and M. Langheinrich. IŠm here! privacy challenges in mobile location sharing, 2010.

[10] A. Sheth, S. Seshan, and D. Wetherall. Geo-fencing: Confining wi-fi coverage to physical boundaries. In *Proceedings of International Conference on Pervasive Computing*, pages 274–290, Berlin, Heidelberg, May 2009. Springer-Verlag.

[11] B. Thompson. Facing the future facebook style. January 2008. `http://news.bbc.co.uk/2/hi/technology/7178954.stm`.

[12] E. Toch, J. Cranshaw, P. Hankes-Drielsma, J. Springfield, P. G. Kelley, L. Cranor, J. Hong, and N. Sadeh. Locaccino: a privacy-centric location sharing application. In *Ubicomp '10: Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, pages 381–382, New York, NY, USA, 2010. ACM.

[13] G. Treu, F. Fuchs, and C. Dargatz. Implicit authorization for accessing location data in a social context. *Availability, Reliability and Security, International Conference on*, 0:263–272, 2007.

[14] J. Y. Tsai, P. G. Kelley, L. F. Cranor, and N. Sadeh. Location-sharing technologies: Privacy risks and controls. In *In Research Conference on Communication, Information and Internet Policy (TPRC*, 2009.

[15] A. Ulin. With the new google latitude api, build latitude and location into your app, May 2010. `http://googlecode.blogspot.com/2010/05/with-new-google-latitude-api-build.html`.