Enhancing User Privacy in Location-Based Services

Urs Hengartner
David R. Cheriton School of Computer Science
University of Waterloo
uhengart@cs.uwaterloo.ca

August 16, 2006

Abstract

The ubiquity of cellphones has lead to the introduction of location-based services, which are services tailored to the current location of cellphone users. For example, a cellphone user can retrieve a list of interesting, nearby places, or parents are notified when a child (i.e., her cellphone) leaves a boundary area. Location is a sensitive piece of information, so location-based services raise privacy concerns. In this paper, we identify and address one such concern. Namely, a company providing location-based services will become aware of the location of cellphone users. Therefore, the company can, maybe inadvertently, leak this information to unauthorized entities. We study the question whether it is possible for the company to provide its services without learning the location of cellphone users. We present an architecture that provides this property and show that the architecture is powerful enough to provide various, existing location-based services. Our architecture exploits Trusted Computing and Private Information Retrieval algorithms for implementing location-based services. With the help of Trusted Computing, we ensure that a location-based service operates as expected by a cellphone user and that location information becomes inaccessible to a location-based service upon a compromise of the service. With the help of Private Information Retrieval, we avoid that a company providing a location-based service becomes indirectly aware of the location of cellphone users by observing which of its location-specific information is being accessed. We also discuss an implementation of the proposed approach and alternative design strategies.

1 Introduction

The ubiquity of cellphones has lead to the introduction of *location-based services*, which are services tailored to the current location of cellphone users. For example, there are services that allow cellphone users to retrieve information relevant to their current location, such as directions to a target location or a list of interesting, nearby places. Other services allow the tracking of cellphone users and raise an alarm when a cellphone leaves a boundary area. In this way, parents can track their children or employers their employees.

The latter class of services raises obvious privacy concerns and has received wide-spread attention (e.g., in a BBC article [3]). While these concerns are challenging, technological solutions are not sufficient for addressing them; they need to be addressed by society as a whole. However, there is another, less widely discussed privacy concern that affects location-based services and where technological solutions can help. In particular, when providing a location-based service, the company providing this service can become aware of the location of cellphone users. Therefore, a cellphone user must trust this company not to reveal her location to unauthorized entities. Many companies might have good intentions and do not leak information on purpose; for example, they allow cellphone users to identify authorized entities and provide and implement a company privacy policy. Nonetheless, bugs in the company's software or intrusions into its computers can inadvertently leak location information. In this paper, we want to reduce the trusted computing base, that is, we want to avoid that a cellphone user needs to trust a company to deal with her location properly. Namely, we examine the question whether it is possible for a company to provide location-based services *without* learning the location of cellphone users.

We answer this question in an affirmative way. Our main contribution is an architecture for location-based services with enhanced user privacy. In our architecture, a cellphone user can keep her location hidden from a company while benefiting from location-based services provided by this company. Our architecture exploits several concepts from cryptography and security research, such as Private Information Retrieval (PIR) algorithms [7] and Trusted Computing [9]. With the help of PIR algorithms, a cellphone user can retrieve location-specific information of interest from a company without the company being able to tell for which location the user has retrieved the information. We employ Trusted Computing to build a platform that is trusted by a cellphone user to properly implement both a PIR algorithm and some additional, simple algorithms that are required by location-based services. We strive to keep the software base running on this trusted platform small, so there is less chance for bugs in the software to occur, and it becomes possible for a cellphone user (or a third-party auditor) to verify whether the software deals with location information responsibly. Furthermore, with the help of Trusted Computing, we can ensure that the platform can access a user's location only when uncompromised. Any changes to the software by an intruder will immediately make a user's location inaccessible to the platform and hence to the intruder.

In another contribution, we underline the usefulness of our architecture by demonstrating that the architecture is powerful enough to support several, existing location-based services. Moreover, our architecture can serve as a guide for operators of cellphone networks and for companies providing location-based services in terms of the interfaces that are required between the two entities to enhance the privacy of their customers.

In the following section, we discuss some existing location-based services and their implementation in different architectures. We then choose one specific architecture, which is likely to be widely used for location-based services, and discuss the components required for enhancing user privacy in this architecture (Section 3). Next, we demonstrate how to provide the discussed location-based services in this architecture (Section 4). Furthermore, we consider alternative approaches and elaborate on an implementation of the architecture (Section 5).

2 System and Threat Model

In this section, we introduce various, existing location-based services and discuss possible architectures for providing them. We also present our threat model.

2.1 Overview of Location-Based Services

We first give an overview of different location-based services. We collected our list of services by surveying existing companies providing location-based services. We give more details about these providers in Section 2.2. We require that our architecture for privacy-enhanced location-based services can at least provide the services listed below.

Nearby-information service. This service provides information related to a cellphone user's current location, such as places of interest, advertisements, or weather and traffic alerts, to the cellphone user.

Locate-me service. This service informs a cellphone user of her current location, which is useful when being lost. An extension of this service is to let third parties know of a cellphone user's current location, which can be beneficial while the user is traveling.

Tracking service. This service warns a cellphone user when a third-party cellphone user leaves or enters a boundary area. Examples are parents tracking their children or employers tracking their employees. (The notified entity does not need to have a cellphone, we assume so for simplicity reasons.)

Locate-friends service. This service allows a cellphone user to learn the current location of her friends, assuming that they are also carrying a cellphone with them.

Nearby-friends service. This service notifies a cellphone user when some of her friends are nearby.

Similar-interests service. This service informs a cellphone user of nearby cellphone users with similar interests.

Personal-navigator service. This service provides directions from a cellphone user's current location to a target location.

Another service that is likely going to be popular is location-based games. We do not discuss such a service in detail, since it can likely exploit some of the services mentioned above. For example, the above services can easily answer queries like "Where are the other players in my team?", "Is there any information relevant for the game nearby?", or "Are any enemies nearby?".

2.2 Architectural Approaches for Location-Based Services

Let us review some existing architectural approaches for providing location-based services. We will use the following terminology: A *customer* denotes a cellphone and its owner, where the owner initiates actions taken by the cellphone (e.g., invoking a location-based service). A *network operator* is a company operating a cellphone network. The company learns the location of its customers by observing their proximity to cellphone towers and, optionally, by having GPS-equipped cellphones report their location to the company.[1] In terms of location-based services, the company's task is to provide location information to service providers. A *service provider* is a company offering location-based services.

In the first architecture, a network operator also becomes a service provider. For example, the network operators Sprint Nextel in the US and Bell Mobility in Canada offer locate-friends and tracking services, based on specialized software from WaveMarket [36]. T-Mobile in Germany offers a personal-navigator service. This architecture is of limited interest in this paper. Since the service provider and the network operator are tightly integrated, within the same company and maybe even business unit, hiding a customer's location from a service provider makes little sense. The network operator has this information anyway.

In a more open architecture, a network operator provides an API that can be used by (external) service providers to learn customers' location. For example, several network operators in the UK, such as Vodafone or Orange, provide their customers' location to various service providers, such as KidsOK [18], mapAmobile [22], or world-tracker.com [38], which offer locate-friends and tracking services. Skymo [31] acts as a proxy for various network operators and provides their customers' location to various service providers. We focus on this architecture in this paper since it is interesting from a privacy point of view. In particular, our solution enables customers of service providers to benefit from a provider's services without having to inform the provider of a customer's location.

Another possible architecture sidesteps the network operator and has the customer (i.e., the cellphone) determine her current location and submit it to a database run by a service provider. Companies like Wherify [37], Teen Arrive Alive [2], and uLocate [34] or projects like CellSpotting.com [6] and Mologogo [24]

[1]The latter approach allows a cellphone to report only coarse-grained locations, which might increase a cellphone user's privacy, but could reduce the quality of a location-based service. Reporting only coarse-grained locations is orthogonal to the privacy solution discussed in this paper.

follow this approach to offer locate-me, locate-friends, tracking, personal-navigator, or nearby-information services. From a privacy point of view, this architecture is similar to the first architecture. Namely, since the entity gathering a customer's location in a database and the entity providing a service based on the gathered information are identical, it makes little sense to prevent the location information from flowing to the service provider.

It is possible to decouple this architecture and to make a single database collecting location information available to multiple service providers. For example, Wherify mentions this feature for future versions of their software. This approach has several benefits: First, it avoids the scalability problem where each service provider requires its customers to run its own version of location-reporting software. Instead, there is only one version of this software. Second, it avoids the gathering of customers' location in many databases, which is highly troublesome in terms of privacy. From a privacy point of view, this fourth architecture is identical to the second architecture, that is, we want to enable a service provider to offer its services without learning a customer's location.

In another architecture, a customer keeps control over her location and does not automatically submit this information for collection in a database. Instead, the customer reveals the information only for very specific purposes. This architecture has been explored in various research projects, such as Hitchhiking [33], Confab [16], or Place Lab [29]. For example, a Hitchhiking customer sends information about her current location to a centralized database such that the information does not make the customer trackable or identifiable. This approach allows the implementation of some location-based services (e.g., amount of traffic at a particular location). However, it is currently unclear whether this approach is powerful enough to build all the services outlined in Section 2.1.

In this paper, we focus on the second (and hence on the fourth) architecture. This architecture is interesting from a privacy point of view, and it is currently being deployed by network operators. We envision that location-based services will become very important for network operators as an additional source of revenue. However, since the required technology and software are not necessarily part of a network operator's expertise, the operator is likely going to outsource the provisioning of location-based services to service providers. Here is where the privacy issues studied in this paper come into play.

In this paper, we focus on location-based services that are based on cellphones. However, our proposed solution is also applicable to location-based services based on WiFi devices exploiting a similar architecture.

2.3 Threat Model

The main threat that we address in this paper is a service provider becoming aware of a customer's location. A service provider is allowed to learn the identity of the customer while the customer is using the service, but the provider should never learn the customer's location.

It is important to address this threat because of the following reasons: A malicious service provider (or malicious employees) could exploit location information for purposes not sanctioned by a customer. For instance, the information could leak to criminals planning on robbing the customer or to stalkers. Even for non-malicious providers, which do not deliberately leak location information, such leaks are still possible. For example, bugs in the provider's software can enable an attacker to get unauthorized access to customers' location information. Moreover, an intruder into a machine running a location-based service can passively monitor the service (and thus customers' location) or he can actively query a network operator for location information. Finally, government authorities can exploit legal means to get access to the location information gathered by a service provider, which might not be in the interest of a customer. This concern is especially important in cases where a government cannot get access to a network operator, since the operator is outside of the country.

In order to learn a customer's location, we assume that a service provider can sniff traffic exchanged between itself and a network operator, perform traffic-analysis attacks on this traffic, and set up man-in-the middle attacks. There are also some active attacks that are easily detectable by a customer (see Section 4.3 for an example). While we defend against these attacks, they are not our main focus, since they are of limited interest for a service provider. Namely, if the provider executed such an attack, the customer would detect the attack and stop using the provider's services, which is not in the provider's interest.

In addition to not being able to learn the location of individual customers, a service provider should not be able to infer which of its customers are close to each other, either. If the provider happened to know the location of one of the nearby customers, it would also know the location of the other customers.

In cases where it is difficult to implement a location-based service without revealing a customer's location to the service provider, we want to ensure that at least the customer's identity remains hidden from the service (see Section 4.7).

We want to avoid that a network operator learns information (other than location information) provided by a service provider to a customer. For example, for a nearby-information service, a network operator with many customers querying the service from many different locations could potentially learn a big chunk of the information that is in possession of the service provider, which is not in the provider's interest.

Finally, traffic sniffers should not be able to learn or modify the contents of a service provider's response while this response is being sent to a customer.

3 Architecture of Privacy-Enhanced Location-Based Services

In this sections, we first discuss several assumptions that we make in the design of our architecture for privacy-enhanced location-based services. We then present the actual architecture.

3.1 Assumptions for Architecture

A service can be either pull based or push based. In a pull-based architecture, a customer submits a query to a service provider, which answers the query immediately. In a push-based architecture, a customer submits a query, but the provider returns information only when the customer arrives at a location having information asked for in the query (e.g., a location where friends are nearby) or when the provider becomes aware of new such information for the customer's location (e.g., bad weather is approaching). The provider can also continuously return information. Here, we concentrate on pull-based solutions first and discuss push-based solutions later.

In our architecture, for cost and efficiency reasons, we want to limit the number of messages received and sent by a customer. In particular, apart from the message containing the customer's query to the service provider and the message returning the provider's response, the customer should not have to receive or send any additional messages to get her desired service.

A network operator and a service provider have a unique, shared identifier for each customer, such as her SIM card number.

Unless explicitly stated, we use asymmetric, probabilistic encryption schemes, that is, if two identical plaintexts are encrypted with the same encryption scheme and key, chances that the two ciphertexts are identical are negligible. In this way, an observer will not be able to infer, for example, that a customer has not moved by observing two identical ciphertexts. Furthermore, an encryption scheme provides key privacy [4], where a ciphertext does not leak any information about the public key used for generating the ciphertext (see Section 3.2.1 for an application of this concept). Customers (i.e., their cellphones) are powerful enough to perform asymmetric cryptographic operations, such as decrypting ciphertexts or signing messages.

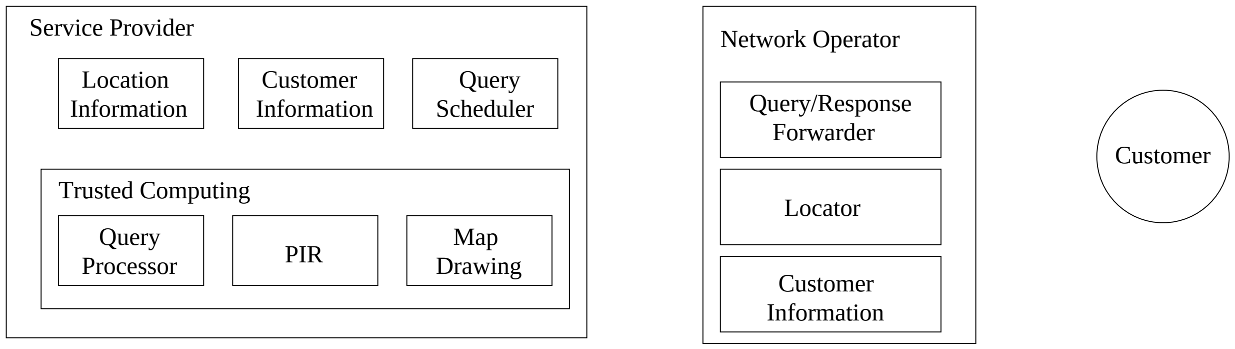

Figure 1: Privacy-enhanced architecture for location-based services. The customer sends a query via the network operator to the service provider, which can use the Trusted Computing module for processing the query and generating a response.

3.2 Design of Architecture

Figure 1 illustrates our privacy-enhanced architecture for location-based services. A customer sends a query to a network operator, which forwards it to a service provider. The service provider generates a response and gives it to the network operator, which forwards it to the customer.

The service provider encrypts its response to keep it hidden from the network operator and traffic sniffers. To achieve message integrity (and non-repudiation), the provider also signs the response.

Let us look at a network operator and at a service provider in more detail.

3.2.1 Network Operator

A network operator consists of the Query/Response Forwarder module, the Customer Information database, and the Locator module.

The Query/Response Forwarder module forwards a query from the customer to a service provider and forwards a response from the service provider to the customer. There can be multiple service providers. A customer can pick a provider in its query or let the network operator know of its choice beforehand. Data traffic between the network operator and the customer can exploit different means, such as SMS or MMS messages or GPRS. Data traffic between the network operator and the service provider flows across the Internet. The response received by the network operator from a service provider can be a dummy response. Dummy responses are encrypted with the operator's public key, whereas real responses are encrypted with the customer's public key. Key privacy guarantees that an observer cannot distinguish between dummy responses and real responses. Dummy responses can be required to thwart traffic-analysis attacks (see Section 4.3). The Query/Response Forwarder module filters dummy responses by decrypting received responses. If it can assure the integrity of the resulting plaintext, the response must be a dummy response and should not be forwarded to the customer. Otherwise, the response should be forwarded to the customer.

The Customer Information database contains information about a customer, such as billing information or her list of subscribed services. For each customer, there is also a public key, which will be used for encrypting her location (see below). This information is established when a customer signs up to the network operator and updated when necessary. We assume that the network provider bills a customer on behalf of a service provider. This way, for services where we keep a customer's identity (but not her location) hidden from a service provider (see Section 4.7), we avoid that a service provider learns a customer's identity during billing, while still making sure that only subscribed customers can access a service.

The Locator module provides a customer's current location to a service provider, given the identifier of the customer. The module always encrypts a customer's location with her public key, kept in the Customer Information database, before handing the information over to the service provider to avoid that the provider (and traffic sniffers) can learn the customer's location. To avoid tampering attacks, the module also signs a customer's location with its private key.

The Locator module optionally supports a closeness function, which lets us implement some services in a more efficient way. This function allows the service provider to query the module for a set of people who are nearby a particular customer and who have signed up to a particular service offered by the provider. There are two instances of this closeness function. The first instance takes as input a customer identifier, a maximum distance, and the type of service. It returns the set of people who are within the given maximum distance from the customer and who have signed up to the service in question. In particular, for each nearby person, the set contains her identifier and her location. The location is encrypted with the person's public key, as it is the case for any location information provided by the Locator module. The person's identifier is also encrypted. This way, we avoid that closeness information leaks to a service provider (see Section 4.6 for details). The second instance of the closeness function takes a set of identifiers as an additional parameter and determines whether the people in this set are nearby the customer. The function returns a boolean value for each identifier listed in the input. The value indicates whether the person in question is nearby. For each nearby person, her location is also returned. For people not nearby, a dummy location is returned. Both the boolean value and the location are encrypted with the public key of a person. The benefit of the second instance is that the length of its output is constant, given a particular input set of identifiers. Therefore, it does not leak any information about the number of nearby people to an observer, as opposed to the first instance.[2] We present scenarios that exploit the two instances in Sections 4.5 and 4.6.

3.2.2 Service Provider

A service provider consists of the Location Information database, the Customer Information database, the Query Scheduler module, and the Trusted Computing module.

The Location Information database stores service-specific information about locations, such as places of interests, weather or road conditions, road maps, or satellite pictures.

The Customer Information database keeps service-specific and customer-specific configuration information required for answering queries from a customer. For example, a nearby-friends service stores the identifiers of a customer's friends who have agreed to being located by the customer in their privacy preferences or a similar-interests service keeps a list of a customer's interests. Information specific to a query, such as the identifier of the party to be tracked or the location to which directions are requested, is handed over as part of the query.

The Query Scheduler module receives customer queries from a network operator and forwards them to the Trusted Computing module for processing, if necessary, or processes them itself. If required for this processing, the Query Scheduler module retrieves (encrypted) location information from the Locator module run by the network operator. When processing is finished, the Query Scheduler module returns the response generated by the Trusted Computing module to the network operator.

The Trusted Computing module is contacted by the Query Scheduler module and processes customer queries. The module has two main properties. First, it is possible for a customer to remotely ensure that the module can access the customer's location only if the software run on the module corresponds to a configuration approved by the customer (or a third-party auditor on the customer's behalf). Second, the service provider deploying the Trusted Computing module cannot learn location information that is being processed by the module.

[2]We could keep the output length of the first instance constant by limiting the number of identifiers that the instance returns and by returning dummy identifiers if necessary. This approach requires a trade-off between efficiency and flexibility.

We can use a Trusted Platform Module (TPM), as suggested by the Trusted Computing Group (TCG) [9], to implement this module. (We assume that the module is run on a dedicated machine.) In particular, we exploit the concepts of *remote attestation* and *sealed storage* to guarantee the first property mentioned above. Remote attestation lets an entity verify whether the software (including operating system and applications) running on a remote computer corresponds to an expected configuration. Sealed storage prevents certain encrypted information from being decrypted on a computer unless the software running on the computer corresponds to a given configuration. We apply these two concepts in the following way: Each customer creates an asymmetric key pair and gives the public key to her network operator, which stores the key in the Customer Information database, as mentioned in Section 3.2.1. The Locator module uses this public key for encrypting the customer's location when being queried by the service provider. The customer gives the corresponding private key to the Trusted Computing module only when the customer (or a third-party auditor on the customer's behalf) approves the software configuration of the module. This approval exploits remote attestation. To avoid that the private key leaks upon a compromise of the module, the module keeps the key in sealed storage. In this way, if the module gets compromised and its software configuration changed by the intruder (e.g., installation of a logging program), the private key becomes inaccessible and the module can no longer decrypt the customer's location.

To ensure that the service provider deploying the Trusted Computing module cannot learn customers' location from this module, we must take several additional precautions. First, the software running on this module must never output this information in plaintext. For example, the information should not be logged. Second, developers of the software should take special care to ensure that location information is immediately erased after its usage to decrease the risk of this information being swapped to disk. (Alternatively, since the Trusted Computing module is run on a dedicated machine, swapping could be disabled.) Third, the service provider's privileges for the machine on which the module is running must be limited so that the provider cannot inspect the memory of the module, even if the provider has administrator rights on the machine. In the case of Linux, SELinux [21] makes it possible to limit the privileges of an administrator accordingly. Fourth, a TPM, as suggested by the TCG, does not protect against attackers that have physical access to a machine. The requirement of physical access makes this attack more expensive to implement. We can completely avoid this attack by implementing the Trusted Computing module on the XOM processor architecture [20] or in a secure coprocessor. However, the XOM architecture is not as widely distributed as TPMs and secure coprocessors tend to have limited computational power [23].

A customer (or a third-party auditor on her behalf) should review the software configuration running in the Trusted Computing module. This software includes the operating system and algorithms that are required by location-based services. The customer can require that the operating systems corresponds to a reference configuration (e.g., Linux kernel 2.6.17.8). In our design, we strive to keep the algorithms simple, which makes them easier to review. For additional security, the module can employ secure logging [30], so that the customer can validate processing of the module retroactively. Secure logging ensures that log entries cannot be modified or removed from a log file.

Let us review the individual components of the Trusted Computing module. There is the Query Processor component, which runs service-specific algorithms, as required by a location-based service (see Section 4). The PIR component and the Map Drawing component each provide a common algorithm that is required by most location-based services. Namely, the PIR component implements a Private Information Retrieval (PIR) algorithm [7]. This algorithm allows the Trusted Computing module to retrieve an entry from the Location Information database without the administrator of the database (i.e., the service provider) becoming aware of which entry is being accessed. Without this component, the service provider could learn which database entries are retrieved by the Trusted Computing module, such as a road map, and hence learn a customer's location. The Map Drawing component is given a road map or a satellite image, as retrieved from the Location Information database by the PIR component, and draws additional information on the map, such as the location of a customer's friends.

To prevent the service provider from learning the customer's location, the network operator from learning information other than the customer's location from the response generated by the Trusted Computing module, and traffic sniffers from learning confidential information, the module encrypts its response with a customer's public key. We have a customer create a second asymmetric key pair, in addition to the one used for encrypting the customer's location, and present the public key to the Trusted Computing module after inspecting the module. The module can generate a certificate that binds the public key to the customer identifier, using a private key kept in sealed storage, and store the certificate in the Customer Information database of the service provider. Later queries from the customer should be signed with the customer's private key to avoid tampering attacks. Due to the same reason, the module should also signs a response with its private key.

4 Privacy-Enhanced Location-Based Services

Let us now discuss how we can exploit the architecture presented in Section 3 to implement the various location-based services outlined in Section 2.

4.1 Nearby-Information Service

In the nearby-information service, a customer informs the service provider of her current location, and the provider returns information about this location. (Remember that all communication occurs indirectly via the network operator.)

The service provider has a list of locations and information about each location. We want to implement the service such that the customer can retrieve information about her location from the provider without revealing her location to the provider. A simple solution for this problem is to have the service provider return its entire Location Information database to the customer and to have the customer extract the information about her location. Clearly, this approach is not practical in terms of required network and processing bandwidth. Furthermore, considering that this database contains information valuable to the service provider, the provider is hardly willing to send its entire database to each customer. More specifically, the provider wants to ensure that the customer can learn information only about her current location and about no other location. This is the symmetric instance of the PIR problem. We exploit the PIR component in the Trusted Computing module for retrieving information from the Location Information database. It can extract this information without the service provider becoming aware of which information. Note that the implementation of the PIR algorithm in the Trusted Computing module does not need to be symmetric. Since the provider deploys the module and hence can decide what software it uses (though this software needs to be approved by the customer), it is not a problem when the module learns more information during its processing than what should be given to the customer. The service provider just needs to ensure that the module does not actually return this additional information to the customer.

An alternative approach is to have the customer (i.e., her cellphone) run a symmetric PIR algorithm when accessing the Location Information database. However, PIR algorithms are expensive in terms of the number of messages exchanged and thus violate our requirement of minimizing this number, as stated in Section 3.1. We do not want to run the PIR algorithm with the network operator, either, because this approach would allow the network operator to learn information offered by the service provider, which we want to avoid, as stated in Section 2.3.

In more detail, we implement the nearby-information service in the following way: When receiving a customer's query from the Query/Response Forwarder module, the Query Scheduler module retrieves the customer's (encrypted) location from the Locator module. The Query Scheduler module forwards the query and the location to the Query Processor component in the Trusted Computing module, which decrypts the customer's location and invokes the PIR component to retrieve relevant information from the Location Information database. Next, the Query Processor component optionally has the Map Drawing component generate a map-based version of the information. Finally, the Query Processor component signs and encrypts the result and returns it to the Query Scheduler module, which forwards it to the customer via the Query/Response Forwarder module.

4.2 Locate-Me Service

The locate-me Service allows a customer to learn her current location. Upon receiving a customer's query, the Query Scheduler module retrieves the customer's (encrypted) location from the Locator module and hands over the query and the location to the Trusted Computing module. The Query Processor component decrypts the location and invokes the Map Drawing component to visualize the information. Next, the Query Processor component signs and encrypts the generated map and returns it to the Query Scheduler module.

Location-based services that allow a customer to release her location to third parties can be implemented in a similar way. The only change for the service provider is that the generated map should be encrypted with the public key of the third party, which can be contained in the (signed) query. The Query/Response Forwarder module must forward the response to the third party.

4.3 Tracking Service

When invoking the tracking service, a customer informs the service of the identifier of a third party. If the third party has left a boundary area, the customer is warned. We envision that most customers will use the push-based version of this service, where the third party is tracked for a continuous time and the customer is notified as soon as the third party leaves or enters the boundary area. We discuss push vs. pull in Section 5.1 and concentrate on the pull-based version here.

The Query Scheduler module needs to ensure that the third party has given consent to being tracked, as indicated in the party's privacy preferences stored in the Customer Information database. If there is consent, the module queries the Locator module for the location of the third party and hands over the customer query, the encrypted location, and the boundary area, as stored in the Customer Information database or in the customer's query, to the Trusted Computing module, whose Query Processor component verifies whether the third party is within the boundary area. As required by the definition of the service above, the Trusted Computing module needs to generate a response for the customer only if the third party has left the area. However, this approach is susceptible to traffic-analysis attacks by the service provider. Namely, whenever there is no result from the Trusted Computing module, the provider concludes that the third party is within the boundary area. Therefore, the Trusted Computing module should always generate a response, potentially a dummy response, as outlined in Section 3.2.1.

The Trusted Computing module can also invoke the Map Drawing component, instead of just returning a binary result to the customer.

Our scheme allows a malicious service provider to become a customer and to successfully issue queries that track a third party, even though the third party has not consented. The reasons are that consent checking is not part of the Trusted Computing module and that the integrity of a party's privacy preferences is not ensured. We can address this attack by moving consent checking into the Trusted Computing module and by having a party digitally sign its privacy preferences. However, this approach makes the Trusted Computing module more complex. We prefer a retroactive approach, where the Trusted Computing module employs secure logging to log all requests. This way, a third party can identify a malicious service provider and stop using the provider's services by revoking the public key used by the Locator module to encrypt the party's location. As stated in our threat model in Section 2.3, this is not in a provider's interest.

4.4 Locate-Friends Service

The locate-friends service allows a customer to locate her friends. When receiving a customer query, the Query Scheduler module ensures that the customer's friends have consented to this information exchange. Next, the module retrieves the friends' (encrypted) location from the Locator module. The Query Processor component in the Trusted Computing module decrypts the information and hands it over to the Map Drawing component, which generates one or several maps. The Query Processor component encrypts these maps and has the Query Scheduler module return them to the customer.

This service is subject to an attack where a service provider becomes a customer and locates "friends". This attack is identical to the one discussed for the tracking service in Section 4.3 and can also be addressed with secure logging.

In the locate-friends service, the network operator can learn the identifiers of a customer's friends by observing whose location the service provider is querying for. This observation violates one of our goals stated in Section 2.3. A PIR algorithm could solve this problem. For instance, the service provider could locate a superset of the customer's friends. However, we refrain from implementing this approach, since it is likely that just by tracking which cellphones are nearby in general, the network operator becomes aware of a customer's friends.

4.5 Nearby-Friends Service

The nearby-friends service is similar to the locate-friends service, but it locates only friends who are nearby. Ideally, the service provider cannot learn which of a customer's friends are nearby. (If the provider happened to know the location of one of the friends, it could learn the customer's location.)

The first step is identical to the locate-friends service, that is, the Query Scheduler module performs access control and retrieves the (encrypted) location of each friend and of the customer. Then, all the locations are given to the Query Processor component in the Trusted Computing module, which verifies whether the friends are nearby the customer. The component then asks the Map Drawing component to draw a map showing the location of the nearby friends. To avoid traffic-analysis attacks, the Trusted Computing module always has to return a result, potentially a dummy result, as outlined in Section 3.2.

If available, the service provider can exploit the closeness function offered by the Locator module. This approach has the benefit that only a single query needs to be sent to the network operator. Here, the Query Scheduler module submits the customer's list of friends to the Locator module, which then identifies the ones nearby and their location, while keeping this information secret from the service provider. The second instance of the closeness function introduced in Section 3.2 provides this functionality.

If we are willing to let the service provider infer which friends are nearby, we can have the network operator encrypt locations with a granularity-aware, deterministic encryption scheme. This way, the service provider can figure out which friends are nearby without learning their actual location. Here, a location is split into multiple levels, according to its granularity, and each level is encrypted separately. This way, ciphertexts of locations whose coarse-grained levels are identical, but whose fine-grained levels differ, will be identical for the coarse-grained levels and different for the fine-grained levels. If the service provider returns just nearby-status information (but no location information) to the customer, this approach does not require the Trusted Computing module, and a customer query can be entirely processed by the Query Scheduler module.

4.6 Similar-Interests Service

The similar-interests service reveals a list of nearby people with similar interests to the customer. Upon receiving a customer query, the Query Scheduler module first retrieves the (encrypted) location of all the people who have signed up to the similar-interests service from the network operator. Note that the module cannot ask only for the location of people with interests similar to the interests of the customer. Otherwise, by signing up dummy customers, the network operator would be able to learn about the interests of its customers. Querying for the location of all the customers who have signed up to this service is not efficient. Later, we discuss how the closeness function, as provided by the Locator module, can make querying more efficient.

For each person signed up to the similar-interests service, the network operator will return her identifier and her (encrypted) location. The Query Scheduler module hands over the identifier and location to the Trusted Computing module, which determines people nearby the customer issuing the query. Next, the module retrieves these people's interests using the PIR component from the Customer Information database. This way, the service provider cannot infer which people are nearby. Finally, the Trusted Computing module finds matching interests and invokes the Map Drawing component to generate the result to be returned to the customer.

We can optimize this service using the first instance of the closeness function introduced in Section 3.2. Here, the service provider retrieves only the set of people nearby the customer issuing the query and their (encrypted) locations from the Locator module. The drawback of this approach is that it leaks information about the number of people nearby the customer to the service provider. We cannot use the second version of the closeness function here, which does not have this disadvantage. The second version is useful only when we can provide an input set of people to the function that is likely significantly smaller than the number of people who have signed up to the service, which is not the case here.

To hide the identifiers of nearby customers from the service provider, the closeness function encrypts the identifiers with the public key of the Trusted Computing module. (We assume that the Trusted Computing module and the network operator have exchanged their public keys; the Trusted Computing module requires the public key of the operator for encrypting dummy responses.) A customer identifier should not be encrypted with the public key of the customer, as it is the case for her location. Otherwise, the Trusted Computing module would have to search all of its customers' private keys for a matching decryption key, which is not efficient.

4.7 Personal-Navigator Service

The personal-navigator service offers directions to a target location. Namely, a customer submits her target location to the service provider. Then, the service provider queries the network operator for the location of the customer, generates directions from this location to the target location, and sends them to the customer.

We could implement this service in a similar way as the other services and have the Trusted Computing module decrypt a user's current location, as received from the Locator module, and have it compute a path to the target location. However, this approach is difficult to implement in practice. In particular, the route-planning algorithm is likely to be part of the service provider's expertise. Therefore, the provider might not be willing to have a customer (or a third-party auditor on the customer's behalf) review this implementation, as required by our definition of the Trusted Computing module. In short, we need to pursue a different approach for this service.

The key observation that helps us to increase a customer's privacy for the personal-navigator service is that this service is independent of the identity of a customer. The service depends only on the target location and the customer's current location. Therefore, in our solution, we let a service provider (i.e., the Query Scheduler module) see a customer's location, but not her identifier. Namely, when receiving a customer query, the Query/Response Forwarder module assigns an ephemeral identifier to the customer and replaces her identifier in the query with the ephemeral identifier before forwarding the query to the Query Scheduler module. This module will cite this ephemeral identifier in its location queries sent to the Locator module and in the directions to be returned to the customer. The cited ephemeral identifier allows the Locator module and the Query/Response Forwarder module to identify and to locate the customer and to route the response generated by the Query Scheduler module to the customer. The customer should have the network operator discard the ephemeral identifier when she reaches the target location or the ephemeral identifier should expire so that the service provider cannot track the customer on a long-term base. Note that this service does not require the Trusted Computing module.

A privacy threat in this solution is that a customer can become identifiable by the service provider if the target location or the customer's location upon invoking the personal-navigator service correspond to a location that likely reveals the customer's identity (e.g., her house). Therefore, the customer should ensure that her initial location and her target location do not reveal any (or least only very limited) information about the customer. For example, the customer could first drive to the town square of her home town and get directions to her target location from there. It is possible for the network operator to assist the customer in this process. In particular, the operator can warn the customer when it realizes that she wants to invoke the personal-navigator service from a location typically occupied by only the same (small) set of people.

Our approach does not allow a service provider to identify a customer directly, but the provider can still track the customer's path to the target location and potentially identify the customer indirectly with the help of a physical observer. In particular, the provider could determine a spot that is typically occupied by only few people on the customer's path and send an observer to this spot, who watches out for the customer and tries to identify her. This attack requires physical observation, which makes its monetary cost high and decreases the provider's incentive to perform such an attack in general. However, the incentive depends on the target location. If knowledge of a customer being associated with a particular target location had some monetary value and made her susceptible to blackmailing, the customer should choose a nearby target location free from these limitations instead.

5 Discussion

5.1 Push vs. Pull

The approaches discussed in Section 4 assume a pull-based approach, where information flows back to the customer immediately after sending a query. Let us now discuss the implementation of a push-based approach, where a response is sent in a delayed way, whenever an event relevant for the service and query in question occurs. To implement this approach, upon receiving a customer query, the Query Scheduler module sets up state for this query and periodically re-executes the query. For example, the module periodically queries the Locator module for the customer's location and hands over this and any additional required information to the Trusted Computing module. Alternatively, if supported by the Locator module, the Query Scheduler module can register a callback function at the Locator module that notifies the Query Scheduler module whenever a customer of interest has moved more than a given threshold from her previous location.

5.2 Alternative Approaches

In this section, we discuss three alternative approaches that could be used for implementing some location-based services and explain why we did not apply these approaches.

5.2.1 Location k-Anonymity

We could implement the nearby-information service in the same way as the personal-navigator service and hide a customer's identifier, but not her location, from a service provider, since the former service is also independent of the identity of a customer. However, this approach would be subject to the same privacy threats as the ones that we discussed for the personal-navigator service in Section 4.7. Namely, the location about which a customer asks for information could leak her identity. Furthermore, a service provider might be able to track a customer and identify her using physical observation. Tracking is possible if the same customer issues multiple queries within a short time frame and from close, sparsely populated locations. Here, the service provider likely concludes that the two queries are issued by the same customer, even though the network operator assigns a fresh ephemeral identifier to each query. Again, the network operator can warn the customer when she issues queries from locations that might leak her identity or that allow her being tracked. Instead of warning the customer, the operator could also limit the granularity of the location revealed to the service provider or ensure that a sufficient number of people are concurrently issuing queries from the same area. (This approach is called location k-anonymity [10].) The drawbacks of this approach are that it reduces the quality of the returned information (since the information now covers a broader area) or increases response time (since the operator might have to wait for a sufficient number of people to show up). Due to these privacy threats, we decided not to reveal a customer's location to the service provider for the nearby-information service and to exploit a PIR algorithm instead. We do not consider this algorithm to be part of a provider's expertise and hence to be proprietary and not reviewable by a customer. If the algorithm were proprietary, the customer should not use this service provider, since proprietary cryptographic algorithms are dangerous.

5.2.2 Privacy-Preserving Set Intersection

For the nearby-friends service, the network operator knows the set of people nearby the customer issuing the query. The service provider knows the set of people who are the customer's friends. The challenge is to compute the intersection of these two sets without the network operator learning the set of friends and without the service provider learning the set of nearby people.

Similarly, for the similar-interests service, the network operator knows the set of people nearby the customer issuing the query. The service provider knows the set of people with similar interests as the customer. Here, the challenge is to compute the intersection of these sets without the network operator learning the set of people with similar interests and without the service provider learning the set of nearby people.

Privacy-preserving set operations [19] can solve these challenges. However, these operations assume that both involved parties can learn the intersection set. However, in our approach, we would like to avoid that a network operator learns which of the nearby people have similar interests. Similarly, there should be no need to let the service provider know which of the people with similar interests are nearby.

5.2.3 Encryption-Based Access Control

An approach that looks promising for implementing the nearby-information service is location-based encryption [1]. Here, a service provider encrypts its information with location-dependent encryption keys and makes the encrypted information publicly available in a distributed way (e.g., in a distributed hash table). The network operator provides decryption keys to customers based on their current location. This approach keeps customers' identity private based on the assumption that a provider is unlikely to track all requests to the nodes that keep the distributed database. However, this approach comes with the usual caveats associated with the deployment of encryption-based access control. First, the decryption keys (and thus potentially the encryption keys) should expire to avoid that obtained keys can be used after moving away from a location. Second, the encryption scheme should be aware of location hierarchies. For example, a decryption key allowing decryption of information covering a particular city block should also allow decryption of information covering the entire city. Third, the access-control scheme breaks when customers publicize decryption keys, which is difficult to avoid since these keys typically have no value for a customer. Fourth, the entity handing out the decryption keys has access to all the information.

5.3 Implementation

We are currently implementing a prototype of our privacy-enhanced architecture for location-based services. Since we do not have access to a network operator, we are building the fourth architecture outlined in Section 2.2 and have a cellphone send its location information to a centralized database, from which service providers access it in an encrypted form. We use software provided by the Place Lab project [29] for gathering location information. The Location Information database exploits Google Maps as a source of road maps. The Trusted Computing module is based on IBM's Integrity Measurement Architecture [27]. For the PIR component of the module, we can exploit previous research [17, 28].

6 Related Work

Privacy issues of location-based services have been researched for a long time. Let us review this research and compare it to our work.

One of the first systems for locating people, the Active Badge Location System [35], was targeted at small environments, such as a research laboratory. In this system, a badge owner could access the current location of any other badge. Clearly, this approach does not scale to larger environments, such as the one discussed in this paper. People caring about their privacy were given the option to leave their badge on their desk. However, peer pressure made it difficult for people to do so [11]. While it is possible to leave a badge behind, leaving behind a cellphone, as in our case, is much more difficult, since the cellphone is used for many tasks, not just for locating a person.

Another approach to implement privacy is to incorporate access control into the location system. This access control can be implemented in a distributed or in a centralized way. In the location system developed by Spreitzer and Theimer [32], there is an agent for each user, which controls access to the user's location information, potentially gathered from multiple sources. In Confab [16] and Hitchhiking [33], a user's device, such as a PDA, senses, stores, and controls access to the user's location information. The drawback of such a distributed architecture is that, as recognized by Harter and Hopper [12], cyclic dependencies can make it difficult to implement certain location-based services, such as a scenario where two devices each reveal their location to the other device only if the other device also reveals its location. A centralized architecture, where access control to a user's location information is performed by a centralized entity, does not suffer from this drawback. Myles et al. [25] describe a framework for implementing location-based services, where a centralized entity runs access control to users' location information and provides this information to individual service providers. To increase user privacy, the system exploits ephemeral identifiers, similar to the ones we use for the personal-navigator service (Section 4.7). The authors discuss only a very limited set of location-based services, and it is unclear whether the approach based on ephemeral identifiers is powerful enough to implement all the services described in this paper. In previous work [13], we examined the design of a centralized location system that exploits multiple sources for gathering location information. Our earlier work assumed that service providers and network operators were tightly coupled.

There are many techniques for locating people, see Hightower and Borriello [15] for an overview. Some of these techniques, such as Cricket [26] or Place Lab [29], have been developed with privacy in mind and allow a device to determine its location based on beacons sent out by an environment. In this way, the environment does not become aware of the device and its location. However, note that for many types of devices, the environment can still become aware of the device and its location. For example, a cellphone or a PDA can certainly exploit beacons to determine its location, but as soon as it communicates, which it is likely to do, the environment learns of the device's location. In this paper, we are oblivious to whether a device or its environment determines the location of the device, as outlined in Section 2.2.

In our work, we (largely) avoid that a service provider becomes aware of a customer's location. Earlier work has explored privacy issues in architectures where a service provider learns location information.

Gruteser and Grunwald [10] introduce the concept of location k-anonymity, where a customer's location is "cloaked" spatially or temporally such that at least k customers are at the same location or have visited the location within the same timeframe, respectively. Beresford and Stajano [5] and Duckham and Kulik [8] also exploit temporal and spatial cloaking, respectively. The drawback of cloaking is that it might decrease the quality of service received from a location-based service or the service's responsiveness. Also, it is unclear whether these approaches are powerful enough to implement all the location-based services outlined in this paper.

In earlier work [14], we studied what kind of privacy violations context-sensitive services, such as location-based services, can cause. We introduced several techniques to avoid these violations. Amongst them are *hidden constraints*, which make it possible to grant a customer access to information if a constraint, such as the customer being at a particular location, is satisfied without the service providing this information becoming aware of the nature of the constraint. Our architecture introduced in this paper is another possible implementation of hidden constraints.

7 Conclusions and Future Work

We have demonstrated that it is possible to build location-based services for which the provider of these services does not become aware of customers' location. Even though our solution does not let service providers become aware of location information, some privacy problems inherent with location-based services remain. For example, as mentioned in Section 1, some services, such as a tracking service, have implicit privacy concerns. Solutions like notifying customers when they are being tracked can at least inform a customer of her loss of privacy.

In terms of future work, we are currently implementing our architecture. Furthermore, we are investigating whether it is possible to implement privacy-aware location-based services in a completely distributed way, where mobile devices directly interact with other mobile devices and do not rely on an explicit service provider. Finally, we are applying our architecture to other scenarios where customers' privacy should be increased, not just in location-based services.

References

[1] J. Al-Muhtadi, R. Hill, R. Campbell, and D. Mickunas. Context and Location-Aware Encryption for Pervasive Computing Environments. In *Proceedings of Third IEEE International Workshop on Pervasive Computing and Communications Security (PerSec 2006)*, pages 283–288, March 2006.

[2] Teen Arrive Alive. `http://www.teenarrivealive.com`.

[3] BBC. `http://news.bbc.co.uk/2/hi/programmes/click_online/4747142.stm`.

[4] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Proceedings of 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT 2001)*, pages 566–582, December 2001.

[5] A. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, 2(1):46–55, January-March 2003.

[6] CellSpotting.com. `http://www.cellspotting.com`.

[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. In *Proceedings of 36th IEEE Symposium on Foundations of Computer Science*, pages 41–50, October 1995.

[8] M. Duckham and L. Kulik. A Formal Model of Obfuscation and Negotiation for Location Privacy. In *Proceedings of Third International Conference on Pervasive Computing*, pages 152–170, May 2005.

[9] Trusted Computing Group. `https://www.trustedcomputinggroup.org`.

[10] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proceedings of First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, May 2003.

[11] H. R. Harper. Looking at Ourselves: An Examination of the Social Organisation of Two Research Laboratories. In *Proceedings 4th Conference on Computer Supported Cooperative Work Sharing Perspectives (CSCW '92)*, pages 330–337, October/November 1992.

[12] A. Harter and A. Hopper. A Distributed Location System for the Active Office. *IEEE Network*, 8(1):62–70, January 1994.

[13] U. Hengartner and P. Steenkiste. Implementing Access Control to People Location Information. *ACM Transactions on Information and System Security (TISSEC)*, 8(4):424–456, November 2005.

[14] U. Hengartner and P. Steenkiste. Avoiding Privacy Violations Caused by Context-Sensitive Services. In *Proceedings of 4th IEEE International Conference on Pervasive Computing and Communications (PerCom 2006)*, pages 222–231, March 2006.

[15] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):57–66, August 2001.

[16] J. I. Hong and J. A. Landay. An Architecture for Privacy-Sensitive Ubiquitous Computing. In *Proceedings of Second International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)*, pages 177–189, June 2004.

[17] A. Iliev and S.W. Smith. Protecting Client Privacy with Trusted Computing at the Server. *IEEE Security and Privacy*, 3(2):20–28, March/April 2005.

[18] KidsOK. `http://www.kidsok.net`.

[19] L. Kissner and D. Song. Privacy-Preserving Set Operations. In *Proceedings of CRYPTO 2005*, August 2005.

[20] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz. Architectural Support for Copy and Tamper Resistant Software. In *Proceedings of 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, pages 168–177, November 2000.

[21] P. Loscocco and S. Smalley. Integrating Flexible Support for Security Policies into the Linux Operating System. In *Proceedings of FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX '01)*, June 2001.

[22] mapAmobile. `http://www.mapamobile.com`.

[23] J. Marchesini, S. Smith, O. Wild, and R. MacDonald. Experimenting with TCPA/TCG Hardware, Or: How I Learned to Stop Worrying and Love The Bear. Technical Report TR2003-476, Department of Computer Science/Dartmouth PKI Lab, Dartmouth College, December 2003.

[24] Mologogo. `http://mologogo.com`.

[25] G. Myles, A. Friday, and N. Davies. Preserving Privacy in Environments with Location-Based Applications. *Pervasive Computing*, 2(1):56–64, January-March 2003.

[26] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proceedings of 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, August 2000.

[27] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proceedings of 13th Usenix Security Symposium*, August 2004.

[28] F. Saint-Jean. A Java Implementation of a Single-Database Computationally Symmetric Private Information Retrieval (cSPIR) protocol. Technical Report YALEU/DCS/TR-1333, Yale University, July 2005.

[29] B. Schili, A. LaMarca, G. Borriello, W. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson. Challenge: Ubiquitous location-aware computing and the place lab initiative. In *Proceedings of First ACM International Workshop on Wireless Mobile Applications and Services on WLAN (WMASH 2003)*, pages 29–35, September 2003.

[30] B. Schneier and J. Kelsey. Cryptographic Support for Secure Logs on Untrusted Machines. In *Proceedings of 7th Usenix Security Symposium Proceedings*, pages 53–62, January 1998.

[31] Skymo. `http://www.skymo.com/Location`.

[32] M. Spreitzer and M. Theimer. Providing Location Information in a Ubiquitous Computing Environment. In *Proceedings of SIGOPS '93*, pages 270–283, Dec 1993.

[33] K.P. Tang, J. Fogarty, P. Keyani, and J.I. Hong. An Anonymous and Privacy Sensitive Approach to Collecting Sensed Data in Location-Based Applications. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI2006)*, pages 93–102, April 2006.

[34] uLocate. `http://www.ulocate.com`.

[35] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[36] WaveMarket. `http://www.wavemarket.com`.

[37] Wherify. `http://www.wherifywireless.com`.

[38] world tracker.com. `http://www.world-tracker.com`.