Hiding Location Information from Location-Based Services

Urs Hengartner
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo ON, N2L 3G1, Canada
uhengart@cs.uwaterloo.ca

***Abstract*—In many existing location-based services, a service provider becomes aware of the location of its customers and can, maybe inadvertently, leak this information to unauthorized entities. To avoid this information leak, the provider should be able to offer its services such that the provider does not learn any information about its customers' location. We present an architecture that provides this property and show that the architecture is powerful enough to support existing location-based services. Our architecture exploits Trusted Computing and Private Information Retrieval. With the help of Trusted Computing, we ensure that a location-based service operates as expected by a customer and that information about the customer's location becomes inaccessible to a location-based service upon a compromise of the service. With the help of Private Information Retrieval, we avoid that a service provider learns a customer's location by observing which of its location-specific information is being accessed.**

I. Introduction

The ubiquity of cellphones has lead to the introduction of location-based services, such as services that provide information relevant to the current location of a cellphone user (e.g., directions to a target location or a list of interesting, nearby places). Another location-based service could track a cellphone user and raise an alarm when the user leaves a boundary area.

Location-based services raise privacy concerns. The provider of a location-based service could learn the current location of a cellphone user, which might reveal information about the user's activities or interests. While it is possible to provide more coarse-grained, less intrusive location information to a location-based service, some services, such as the tracking service mentioned above, require fine-grained location information. A user of such a service must trust the service provider not to misuse or leak her location information. Most providers probably have good intentions. Nonetheless, software bugs or computer break-ins can inadvertently leak location information. In this paper, we want to reduce the trusted computing base. In particular, we examine the question whether it is possible for a service provider to offer 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 where a cellphone user can keep her location hidden from a service provider while benefiting from location-based services. Our architecture exploits two concepts from cryptography and security research, namely, Private Information Retrieval (PIR) [1] and Trusted Computing [2]. With the help of PIR, a cellphone user can retrieve location-specific information from a provider without the provider being able to learn the location the requested information is about. 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 required by location-based services. Furthermore, with the help of Trusted Computing, we can ensure that the platform can access a user's location only when the platform is not compromised. Any changes to the software by an intruder will 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 operators of cellphone networks and providers of location-based services as a guide for designing an interface between the two parties.

The rest of this paper is organized as follows: We first present our system and threat models (Section II). We then introduce our architecture that maintains location privacy (Section III). Next, we demonstrate how to provide some location-based services in this architecture (Section IV).

II. System and Threat Models

In this section, we introduce a system model that underlies many existing location-based services. We also present our threat model.

A. System Model

Location-based services are becoming an important source of revenue for operators of cellphone networks. However, since the required technology and software are not necessarily part of the key expertise of a network operator, many operators outsource the provisioning of location-based services to third-party service providers. Therefore, in our system model, we assume that network operators and service providers are separate (business) entities and that a network operator implements an API that is used by a service provider to offer location-based services. For example, several network operators in the UK, such as Vodafone or Orange, provide their customers' location to various service providers, such as mapAmobile [3]

or World-Tracker.Com [4]. Sprint and Bell Canada use WaveMarket's Family Finder technology [5] to allow parents to track their children.

In our system model, a network operator always knows its customers' identity and location (unless a customer's cellphone is turned off). In many existing location-based services, a service provider also becomes aware of the customers' location and identity.

B. 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, leaks are still possible. For example, software bugs can enable an attacker to get unauthorized access to location information. Moreover, an intruder into a machine running a location-based service can passively monitor the service (and thus customers' location) or the attacker 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.

To learn a customer's location, 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 IV-B for an example). While we defend against these active 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 could detect the attack and would stop using the provider's services.

III. Architecture

Figure 1 illustrates our architecture for location-based services that does not reveal location information to a service provider. A customer sends a query to a network operator, which forwards the query to a service provider. The service provider generates a response and gives it to the network operator, which forwards it to the customer.

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

A. Network Operator

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

The Query/Response Forwarder module forwards a query from a 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 her query or let her network operator know of her choice beforehand. Data traffic exchanged between a network operator and a customer can exploit different means, such as SMS or MMS messages or GPRS. Data traffic between a network operator and a service provider uses TCP/IP. The response received by a network operator from a service provider is encrypted with the public key of the network operator. The Query/Response Forwarder module decrypts the response and checks whether the obtained plaintext corresponds to a dummy response. Dummy responses can be required to thwart traffic-analysis attacks (see Section IV-B) and are not forwarded to a customer. For non-dummy responses, there is a second layer of encryption; they are also encrypted with a customer's public key. This way, the network operator cannot learn any potentially confidential information returned by a service provider to the customer. The Query/Response Forwarder module forwards an (encrypted) non-dummy response 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 information (see below). The customer information and the public key are established when a customer signs up with the network operator.

The Locator module provides a customer's current location to a service provider, given the identity 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.

B. Service Provider

A service provider implements 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 tracking service stores the identity of people that are allowed to track a cellphone.

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

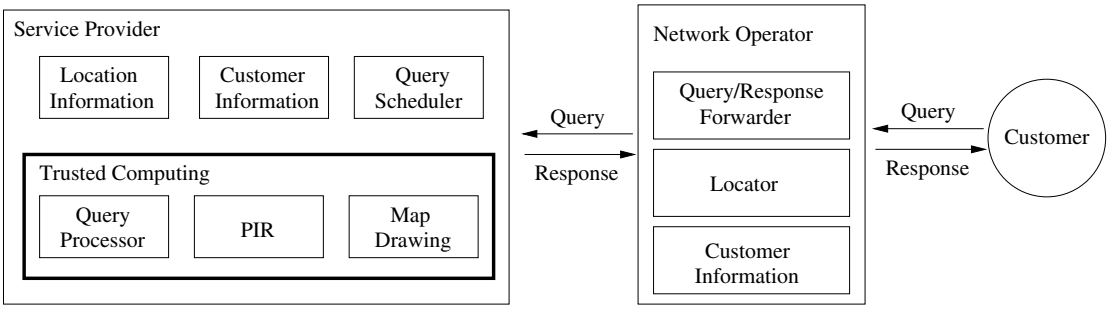

Fig. 1. Architecture for location-based services. The customer sends a query via the network operator to the service provider, which uses the Trusted Computing module for processing the query and generating a response.

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 module's software corresponds to a configuration approved by a 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.

We can use a Trusted Platform Module (TPM), as suggested by the Trusted Computing Group (TCG) [2], 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 III-A. 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 if 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 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 retrieve location information from this module, we must take several additional precautions. First, the software running on this module must never output location 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 runs 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 [6] makes it possible to limit the privileges of an administrator in this way. Fourth, a TPM, as suggested by the TCG, protects against software-based attacks, but not against (more expensive) hardware-based attacks. We can defend against this kind of attacks by implementing the Trusted Computing module on the XOM processor architecture [7] or in a secure coprocessor. However, the XOM architecture is not as widely distributed as TPMs and secure coprocessors tend to be expensive and to have limited computational power.

A customer (or a third-party auditor on her behalf) should review the software configuration of 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 [8], so that the customer can validate processing of the module retroactively. Secure logging ensures that log entries cannot be modified.

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 IV). The PIR component and the Map Drawing component each provide a common algorithm that is required by many location-based services. Namely, the PIR component implements a Private Information Retrieval (PIR) algorithm [1]. 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 for a particular area, 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.

A response generated by the Trusted Computing module might allow the service provider to learn the customer's location and the network operator (or a traffic sniffer) to learn other potentially sensitive information about the customer. To avoid this attack, the module encrypts its response with the customer's public key. Namely, the customer creates a second asymmetric key pair, in addition to the one used for encrypting the customer's location, and presents the public key to the Trusted Computing module after inspecting the module. The module generates a certificate that binds the public key to the customer's identity, using a private key kept in sealed storage, and stores the certificate in the Customer Information database. 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 sign responses with its private key.

We are currently implementing the proposed architecture. We discuss some implementation issues in the extended version of this paper [9].

IV. Sample Location-Based Services

Let us now discuss how we can exploit the architecture presented in Section III to implement a proximity service and a tracking service. We discuss the implementation of other services, such as a service to locate (nearby) friends, nearby people with similar interests, or a navigation service, in the extended version of this paper [9].

A. Proximity Service

In a proximity service, a customer informs the service provider of her current location, and the provider returns information about this location, such as places of interest, advertisements, or weather and traffic alerts. (All communication occurs indirectly via the network operator.)

We exploit the PIR component in the Trusted Computing module for retrieving information from the Location Information database. This way we can extract this information without the service provider becoming aware of the customer's location. In more detail, we implement the proximity 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.

B. Tracking Service

A tracking service allows a customer to track a third party. When the third party has left a boundary area, the customer is warned.

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. The 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 III-A.

The Trusted Computing module can also invoke the Map Drawing component, instead of returning only 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 II-B, this is not in a provider's interest.

V. Related Work

Access control has been used as a tool to limit the number of people or service providers that can access location information. In the distributed systems developed by Spreitzer and Theimer [10], Hong and Landay [11], and Tang et al. [12], a personal agent or device controls access to a user's location.

The drawback of a distributed architecture is that cyclic dependencies can make it difficult to implement some location-based services [13]. A centralized architecture, as proposed by Myles et al. [14], does not suffer from this drawback. To increase user privacy, Myles et al. suggest the usage of pseudonyms, which is also proposed by Beresford and Stajano [15]. A limitation of pseudonyms is that, as observed by Beresford and Stajano, some location-based services require a user's true identity, such as a service to locate (nearby) friends. Our architecture supports services that require a user's true identity [9]. In previous work [16], we studied the design of a centralized architecture that exploits multiple sources of location information.

Our approach avoids that a service provider becomes aware of a customer's location. Earlier work has explored privacy issues in architectures that do not have this property. Gruteser and Grunwald [17] introduce "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. Gedik and Liu [18] and Duckham and Kulik [19] also exploit cloaking. A drawback of cloaking is that it might decrease the quality of a response received from a location-based service. Cheng et al. [20] use probability to model this quality. They show that to get better quality, the degree of cloaking (and thus location privacy) needs to be reduced. Our approach guarantees perfect response quality without revealing any location information to a service provider. Mokbel et al. [21] also use spatial cloaking and have a location-based service return a superset of the information of interest to a customer. This approach is problematic for cellphone users because of bandwidth and processing constraints.

Ravi et al. [22] have a service provider migrate the code implementing a service to a network operator. The operator uses information flow control to ensure that the code does not leak a customer's location to the provider. This approach is targeted at services that exploit aggregate location information and does not support services that require precise location, such as a tracking service.

VI. 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. We are currently implementing our architecture. Our implementation must take special care to avoid traffic-analysis attacks. For example, properties of ciphertexts, such as their lengths, must not leak (implicit) location information to a service provider.

Acknowledgments

We thank the anonymous reviewers for their comments. This work is supported by the Natural Sciences and Engineering Research Council of Canada.

References

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

[2] Trusted Computing Group, https://www.trustedcomputinggroup.org, accessed February 2007.

[3] Cybit Ltd, "mapAmobile," http://www.mapamobile.com, accessed February 2007.

[4] World-Tracker.Com, http://www.world-tracker.com, accessed February 2007.

[5] WaveMarket, Inc, "Family Finder," http://www.wavemarket.com, accessed February 2007.

[6] 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.

[7] 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)*, November 2000, pp. 168–177.

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

[9] U. Hengartner, "Enhancing User Privacy in Location-based Services," Centre for Applied Cryptographic Research, University of Waterloo, Tech. Rep. CACR 2006-27, August 2006.

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

[11] 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)*, June 2004, pp. 177–189.

[12] K. Tang, J. Fogarty, P. Keyani, and J. 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)*, April 2006, pp. 93–102.

[13] A. Harter and A. Hopper, "A Distributed Location System for the Active Office," *IEEE Network*, vol. 8, no. 1, pp. 62–70, January 1994.

[14] G. Myles, A. Friday, and N. Davies, "Preserving Privacy in Environments with Location-Based Applications," *Pervasive Computing*, vol. 2, no. 1, pp. 56–64, January-March 2003.

[15] A. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, January-March 2003.

[16] U. Hengartner and P. Steenkiste, "Implementing Access Control to People Location Information," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 4, pp. 424–456, November 2005.

[17] 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.

[18] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *Proceedings of 25th International Conference on Distributed Computing Systems (ICDCS 2005)*, June 2005.

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

[20] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving User Location Privacy in Mobile Data Management Infrastructures," in *Proceedings of 6th Workshop on Privacy Enhancing Technologies (PET 2006)*, June 2006, pp. 40–58.

[21] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006)*, September 2006, pp. 763–774.

[22] N. Ravi, M. Gruteser, and L. Iftode, "Non-Inference: An Information Flow Control Model for Location-based Services," in *Proceedings of The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS 2006)*, July 2006.