A Distributed k-Anonymity Protocol for Location Privacy

Ge Zhong and Urs Hengartner
Cheriton School of Computer Science, University of Waterloo
{gzhong,uhengart}@cs.uwaterloo.ca

September 29, 2008

Abstract

To benefit from a location-based service, a person must reveal her location to the service. However, knowing the person's location might allow the service to re-identify the person. Location privacy based on k-anonymity addresses this threat by cloaking the person's location such that there are at least $k-1$ other people within the cloaked area and by revealing only the cloaked area to a location-based service. Previous research has explored two ways of cloaking: First, have a central server that knows everybody's location determine the cloaked area. However, this server needs to be trusted by all users and is a single point of failure. Second, have users jointly determine the cloaked area. However, this approach requires that all users trust each other, which will likely not hold in practice. We propose a distributed approach that does not have these drawbacks. Our approach assumes that there are multiple servers, each deployed by a different organization. A user's location is known to only one of the servers (e.g., to her cellphone provider), and different users let different servers (cellphone providers) know of their location. With the help of cryptography, the servers and a user jointly determine whether the k-anonymity property holds for the user's area, without the servers learning any additional information, not even whether the property holds. A user learns whether the k-anonymity property is satisfied, but no other information. The evaluation of our sample implementation shows that our distributed k-anonymity protocol is sufficiently fast to be practical. Moreover, our protocol integrates well with existing infrastructures for location-based services, as opposed to the previous research.

1 Introduction

With the advance of location technologies, people can now determine their location in various ways, for instance, with GPS or based on nearby cellphone towers. These technologies have led to the introduction of location-based services, which allow people to get information relevant to their current location. Location privacy is of utmost concern for such location-based services, since knowing a person's location can reveal information about her activities or her interests.

In this paper, we focus on location-based services that need to know only a person's location, but not her identity. For example, these can be services that provide road maps, nearby places (e.g., restaurants or gas stations), or current traffic conditions. As it turns out, even if a service learns only a person's location, it might still be able to re-identify the person [19]. For example, the location could be associated with the person (e.g., her home), or the location corresponds to a place that is under physical surveillance by the location-based service. Once a service has re-identified a person, the service can connect the dots and build a detailed location profile for this person (assuming the person uses the service in a continuous way).

Location cloaking is a defence against re-identification. It is based on the idea of sending coarse-grained location information that covers multiple people to a service. In a naïve approach, a user simply determines an area (e.g., four city blocks) that contains her current location and sends the area's coordinates to the location-based service. The service will return information about the entire area, and the user will discard any irrelevant information. Unfortunately, this approach could still allow re-identification. For example, in a rural, less populated region, this kind of cloaking might well result in an area that includes only one user.

Location cloaking based on k-anonymity does not have this disadvantage. Here, a user's current location is cloaked such that there are at least $k - 1$ other users within the cloaked area. A location-based service learns only the cloaked area, which allows the user to remain anonymous within the set of k users. Applying k-anonymity to location cloaking has been studied extensively [1, 3, 5, 11, 15–17, 19, 21, 22, 25, 26, 30]. Traditionally, this approach has been implemented with the help of a central trusted server [1,3,5,15,19,21, 22,25,26,30]. Here, users register their current location with the trusted server. Whenever a user wants to access a location-based service, she has the trusted server compute a cloaked area that has the k-anonymity property. Then, the trusted server contacts the location-based service on the user's behalf. The drawback of this approach is that the trusted server knows everybody's location. Users must trust it not to leak their location information to unauthorized parties, maybe inadvertently. In short, the trusted server is a single point of failure. More recent research has proposed to get rid of the trusted server and to have (nearby) users jointly compute a cloaked area that has the k-anonymity property [11, 16, 17]. Then, the user (or, for increased privacy, another user on her behalf) contacts the location-based service. The drawback of this approach is that all previous solutions trust users to implement the proposed solution faithfully and not to leak location information learned during the computation. Whereas this requirement might hold in a closed environment, where users know each other, it will be difficult to satisfy in more open environments.

Another drawback of both the centralized and the distributed approach is that neither of them integrates well with existing infrastructures for location-based services. Namely, many existing location-based services are targeted at cellphone users, since the operator of a cellphone network knows the current location of *its* customers and can provide this information to a location-based service. However, there is no single entity that knows the location of *all* cellphone users across all cellphone networks, as required by the centralized approach. The distributed approach fails to take advantage of the already existing location information that an operator has about its customers.

We propose a solution that requires neither a single trusted server nor trust in all users of the system and that integrates well with existing infrastructures. Namely, we have multiple servers, each deployed by a different organization (e.g., an operator of a cellphone network) and each knowing the location of only a *subset* of users (e.g., the operator's customers), with the subsets being disjoint. When a user wants to access a location-based service, she cloaks her area and asks each server for the number of people in this area. In a naïve solution, the servers simply give her these numbers, she sums them up and, if the sum is at least k, she accesses the location-based service. However, this approach has the flaw that it might allow the user to track people. For example, if the user learns that there is only a single person in an area and nobody in the surrounding areas, the user can likely follow the path of the person when the person leaves the area and enters one of the surrounding areas. As soon as the person enters an area that is associated with her identity or that is under surveillance by the user, the user can re-identify the person. In general, sophisticated data-mining algorithms might allow the tracking or re-identification of a person even if there are multiple people in an area.

Our solution avoids this problem with the help of cryptography and ensures that a user cannot learn the number of people in an area reported by a server. The user can learn only whether the sum of these numbers is at least k. Our contributions are:

- First, we introduce a distributed k-anonymity protocol for location privacy in which a user collaborates with multiple servers and a third party to learn whether there are at least k people in her area. Nobody, not even the servers and the third party, can learn the total number of people in the area.

- Second, we present a protocol that prevents users from registering multiple times with different servers and hence from skewing the total number of users in area.

- Third, we present a sample implementation of our protocol. In its evaluation, we demonstrate that our protocol can be implemented efficiently.

A preliminary overview of our architecture appeared in a workshop paper [33]. The workshop paper misses several key components of the architecture and omits the evaluation and security analysis of the architecture.

In Section 2, we discuss related work in the area of k-anonymity and location privacy. In Section 3, we present our system and threat model. We introduce our distributed k-anonymity protocol for location privacy in Section 4. In Section 5, we present our defence against multiple registrations. We give a security analysis in Section 6 and evaluate our architecture in Section 7. Finally, we study some deployment issues in Section 8.

2 Related Work

Samarati and Sweeney [29] propose k-anonymity to enable the release of person-specific information from a database while maintaining individuals' privacy. Previous research has applied k-anonymity to the release of location information that occurs when a user queries a location-based service. We first discuss related work that is based on a central trusted server, then we review distributed approaches.

Gruteser and Grunwald [19] introduce location privacy based on k-anonymity. A trusted "location anonymizer" cloaks a user's location by subdividing space into quadrants until it finds a quadrant that contains the query issuer and fewer than $k - 1$ other users. The parent quadrant becomes the cloaked area. Gedik and Liu [15] let users have personalized values of k, and the cloaked area corresponds to the minimum bounding rectangle of k users. Mokbel et al. [26] observe that this approach can leak information about a user's location (e.g., some users will be on the boundary of the rectangle). They use a balanced quadtree that is traversed bottom-up for better performance until a quadrant with at least k users is found. In our approach, we choose the bottom-up strategy and allow users to personalize k.

Beresford [3] finds that, if a location-based service is familiar with the cloaking algorithm and knows the locations of all users within the cloaked area, the service could infer the identity of the query issuer from the shape of the cloaked area. Namely, this happens when the cloaked area generated for the query issuer is different from the cloaked areas that would have been generated for the other users in the cloaked area. Kalnis et al. [21] and Bettini et al. [5] later re-discover this finding. Kalnis et al. and Mascetti and Bettini [25] present (centralized) cloaking algorithms that are not susceptible to this attack. In our approach, we leave it up to a user to decide what kind of cloaking algorithm to use. She can use either an algorithm similar to Mokbel et al.'s algorithm that does not necessarily guarantee her privacy, but is easy to compute, or an algorithm similar to Mascetti and Bettini's that is robust in terms of privacy, but more expensive.

Chow et al. [11] propose the first distributed approach for location k-anonymity. A user who wants to access a location-based service broadcasts a message with Bluetooth or WiFi. Nearby users respond to this message with their current location. If the number of responses is smaller than $k - 1$, the user repeats the process, but has the nearby users forward the message, maybe iteratively. The user then computes her cloaked location and, for increased privacy, asks a nearby user to send her query for the cloaked location to the location-based service. Ghinita et al. [17] show that this approach often fails to achieve location privacy, since the query issuer tends to be in the center of the cloaked area. The same authors [16] later propose an approach based on a distributed hash table. Here, a user knows at least the positions of the two users that immediately follow and precede her in the hash table. Furthermore, for robustness reasons, a user also needs to know the positions of $log_2(n)$ other users, where n is the number of users. In summary, the

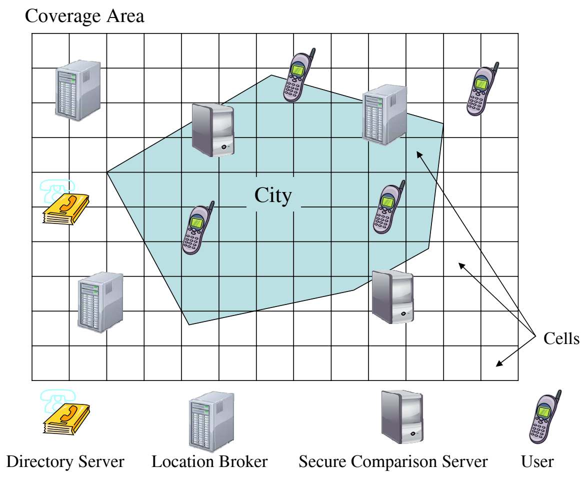

Figure 1: System model. A user registers her location (cell) with a location broker, whose contact information is provided by the directory server. The user can learn whether there are at least k registered users in her cell by contacting all location brokers and one of the secure comparison servers.

proposed distributed approaches for location k-anonymity have the drawback that nodes can learn location information about other nodes, so the nodes have to trust each other [16].

Kapadia et al. [22] propose “statistical k-anonymity”. They assume the global availability of statistical data about the number of people who are present in an area with high probability at a particular time of the day. When a user wants to access a location-based service, she independently decides based on this data whether her area is likely to be visited by at least k people. The drawbacks of this approach are that there remains a chance that fewer than k people are in the area and the requirement of extensive data collection (across different communication technologies and providers and during different times of the day, days of the week,...) to compute the provided statistical data, which raises privacy issues of its own. Our approach is always accurate and requires no such data collection.

k-anonymity is not the only approach that has been suggested for location privacy. Another option are pseudonyms, where a user assumes a pseudonym when contacting a location-based service. Previous work (e.g., by Beresford and Stajano [4] or by Jiang et al. [20]) has explored the challenges of pseudonym-based approaches, such as changing pseudonyms in an unlinkable way. k-anonymity-based and pseudonym-based approaches for location privacy complement each other; the former one is attractive for scenarios where a location is associated with a particular person, the other one for scenarios where locations are public and visited by many people.

The idea of replacing a single trusted infrastructure component with multiple components, each of them having only a limited view of the overall system and run by a different organization, has been exploited by other privacy-enhancing technologies, such as Tor [13]. Similar to Tor, our solution takes advantage of a directory server that advertises infrastructure components being part of the system to users of the system.

3 System and Threat Model

In this section, we present our system and threat model.

3.1 System Model

We present our system model in Figure 1. The figure omits actual location-based services, which a user would access once she learns that at least $k - 1$ other users are in her area, likely via a proxy or an anonymous communication network (e.g., Tor [13]) to hide her identity from the service. For scalability reasons, there are multiple *coverage areas*, where a coverage area corresponds to the area covered by a particular instantiation of our system (e.g., a city or a province). A coverage area is divided into a well-defined grid of equally sized, square cells. The width of a cell is chosen such that, for most cells, there is a realistic chance that multiple users can be located in the cell. For example, a cell could have a width of 100 meters. Moreover, there are four kinds of parties: location brokers, users, secure comparison servers, and a directory server. (Whereas there are solutions that do not require secure comparison servers, we show in Section 8.2 that they are much less efficient than the one proposed here.)

A location broker keeps track of the current location (i.e., the current cell) of a *subset* of the users in the coverage area. There are multiple location brokers, each keeping track of the location of a *different* subset of users, with the intersection of any two subsets being empty. Each broker is maintained by a different organization. For example, the operator of a cellphone network could maintain a location broker that keeps track of the location of the operator's customers in the coverage area. A location broker does not necessarily provide coverage for all cells in the coverage area. For example, whereas a broker maintained by a cellphone network operator would likely cover most cells, a broker operated by the provider of a WiFi network would provide coverage only for a subset of the cells.

Users carry a mobile device (e.g., a cellphone or a laptop) with them that can locate itself (e.g., using GPS or nearby WiFi base stations). A user registers her current location (i.e., her current cell) with exactly one of the location brokers, where she can choose with which one. Likely, if the provider of the communication service exploited by the user's device runs a location broker, the user will (maybe implicitly) register her location with this broker, since the provider already knows or at least has an estimate of the user's location. We assume that users always register their location with a broker. This assumption is also made in the earlier work. More registrations will lead to smaller cloaked areas, which in turn will increase the quality of service obtained from a location-based service and will give users an incentive to register. If a location broker is run by the provider of the user's communication service, registering her location continuously does not lead to additional loss of privacy for the user, since the provider already has this information.

A secure comparison server interacts with a user to let the user learn whether there are at least k users who have registered the user's current cell as their location across *all* location brokers. (See Section 4 for the detailed protocol.) Each secure comparison server is maintained by a different organization. An organization can maintain both a location broker and a secure comparison server. A secure comparison server provides coverage for the entire coverage area.

The directory server publishes contact information for the location brokers and for the secure comparison servers in the coverage area. Moreover, it publishes coverage information for location brokers, that is, which broker provides coverage for which cells in the coverage area. This way, users can choose a location broker to register with and a secure comparison server to interact with.

3.2 Threat Model

In our threat model, the location brokers and the secure comparison servers are honest-but-curious, that is, they honestly follow our protocol, but are curious about learning location information. We discuss malicious brokers and servers in Section 6.2.

A location broker can learn the location (i.e., cell) of users who register their location with this particular broker. Accordingly, a broker can learn the number of users in a cell that have registered this cell as their location with this particular broker. However, a location broker should not learn the *total* number of users in a cell that have registered this cell as their location across *all* location brokers. Knowing this number makes possible tracking attacks, similar to the one presented in Section 1. Similarly, a location broker should not learn the location of users who register their location with any other location broker. We assume that the organizations that run the location brokers do not collude with each other. Legal means (e.g., privacy laws or a contract between a user and a location broker) can enforce this assumption. Technical enforcement means make less sense here, since today's cellphone network operators know their customers' location and could potentially share this information with each other. For the same reason, we assume that location brokers do not collude with users.

A secure comparison server should learn neither a user's location nor the total number of registered users in a cell. This implies that the server should not learn the individual number for a location broker, either (except using back channels if a secure comparison server is run by the same organization as a location broker). A secure comparison server might collude with other secure comparison servers to learn additional information. Due to the same reason given above, we assume that secure comparison servers do not collude with location brokers (except in the implicit case where a broker and a server are run by the same organization, here it can learn at most the location and number of users registered with this broker).

A user should learn only her own location and whether the number of people in her cell (or superset of cells) is at least k, where k is a value of her choice. A user carries only one mobile device with her, and the device faithfully reports its location to a broker. A user cannot register multiple times with a single broker at the same time, since the broker authenticates the user's device. All these assumptions are also made in the earlier work. (We discuss weaker ones in Section 6.3.) A user might still try to register multiple times with *different* brokers at the same time. This could let other users erroneously conclude that k-anonymity holds. Finally, a user might collude with a secure comparison server to learn the number of people in her cell (or superset of cells).

The directory server should not learn any location information about users. The server might misbehave, for example, it might list a location broker multiple times as providing coverage for a single cell, it might fail to vet location brokers or secure comparison servers (see Section 4.4), or it might try to track clients by providing them different information.

4 Distributed k-Anonymity Protocol

In this section, we first give an overview of our distributed k-anonymity protocol and then present its key components.

4.1 Overview

The goal of a user is to learn whether there are at least k registered users (including herself) in the user's *query area*, where k is a value chosen by the user and where the query area initially corresponds to the user's current cell. If the user learns that there are fewer than k users in this cell, she can enlarge the query area to a superset of cells that contains the user's current cell and re-execute the protocol for the enlarged area. This process can be repeated multiple times. As mentioned in Section 2, a user can choose between different types of enlargement algorithms to determine the query area, which lets her trade off between privacy and cost. A user registers her current cell as her location with exactly one of the location brokers, but there is no need for the user to register additional cells when enlarging the query area.

To learn whether there are at least k users in her query area, a user first needs to identify the location brokers that provide coverage for (maybe parts of) the query area. The user must not ask the directory server for a list of brokers that provide this coverage, else the server could learn the user's location. Instead, the user should download the entire directory (or recent changes to it) from the server on a regular basis, such as once a day. The directory is signed, which allows retroactive detection of misbehaviour by the directory server. Another option is to have multiple directory servers, where users accept information only if it is signed by a threshold of the servers, similar to the directory servers in Tor.

The user then executes our distributed k-anonymity protocol with the relevant location brokers and one of the secure comparison servers, l. Our protocol uses the techniques of public-key cryptography, but we require the cryptosystem to have a special algebraic property: that it is *additive homomorphic*. Here, given only $\mathcal{E}_{\mathcal{A}}(m_1)$ and $\mathcal{E}_{\mathcal{A}}(m_2)$, where $\mathcal{E}_{\mathcal{A}}(m)$ is an encryption of message m under public key A, one can efficiently compute $\mathcal{E}_{\mathcal{A}}(m_1 + m_2)$. There are several cryptosystems with this property, such as the Paillier cryptosystem [27]. (See Appendix A.)

In our protocol, the user first asks each broker covering the query area for the number of users who have registered a cell in the query area as their current location with this particular broker. A broker gives this number to the user such that the user cannot learn it. Namely, if there are v_j users in the query area who have registered with broker j, broker j encrypts v_j with public key C_l of secure comparison server l, as published by the directory server, and sends $\mathcal{E}_{C_l}(v_j)$ to the user. Then, the user sums up the received numbers without being able to learn the sum. In particular, the user calculates $\mathcal{E}_{C_l}(r+\sum_i v_i)$ using the additive homomorphic property of the encryption scheme, where r is a random number generated by the user that will keep the total number of users hidden from secure comparison server l. Finally, with the help of secure comparison server l, the user determines whether this sum is at least k (see Section 4.2).

Our protocol gracefully deals with crashes of a location broker or of a secure comparison server. In the first case, the user contacts the remaining brokers, which might still report a sufficient number of registered users. To work around the second case, we can let the user and the broker choose a set of candidate secure comparison servers, instead of only a single one. Over time, the directory server will learn of the crash of a location broker or of a secure comparison server and will remove it from the directory.

4.2 Defence against Collusion

After computing the encrypted sum of users in her query area, the user, in cooperation with secure comparison server l, determines whether this sum is at least k. The user could simply send $\mathcal{E}_{C_l}(r + \sum_i v_i)$ and $r + k$ to server l, which would decrypt the first value, compare it to $r + k$, and inform the user of the result. Since both the sum and k are obscured with r, the server can learn neither of them. However, this solution is flawed, because it might reveal the total number of users to a secure comparison server and a location broker that are run by the same organization. Assume that the location broker is the only broker that covers the query area. Here, based on the knowledge of $\sum_i v_i$ (where the sum covers only one broker), the broker and the server can jointly determine r, which allows them to compute k. In turn, once they know a user's k, the server and the broker can infer the total number of registered people in any query area chosen by the user, as long as the user's choice of k is static and the query area is covered by the broker. The coverage condition guarantees that the broker will be contacted by the user and hence can learn the query area. Otherwise, the server and the broker could learn only the total number of people, but not for which query area. To avoid these information leaks, we need to hide the user's input to the comparison, $r + k$, from the secure comparison server. Moreover, we need to hide the result of the comparison from the server, else the server could still infer $r + k$ in case it is found to be equal to $r + \sum_i v_i$.

To execute the comparison in this way, we exploit the Greater Than - Strong Conditional Oblivious Transfer (GT-SCOT) protocol [6]. The protocol has two participants, a receiver and a sender. The receiver and sender have private inputs x and y, respectively. The sender has two secrets, $s_0 \in D_S$ and $s_1 \in D_S$, where D_S is a subset of $\mathbb{Z}_n$. The sender wants to send s_0 to the receiver if $x < y$ and s_1 if $x > y$, but is oblivious about which secret is sent. In short, the sender cannot learn whether $x < y$ or $x > y$. The protocol requires a semantically secure additive homomorphic encryption scheme with large message domains, such as the Paillier scheme. In the protocol, the receiver encrypts x bit by bit with the receiver's public key and sends the vector of ciphertexts to the sender. The sender encrypts y bit by bit with the receiver's public key and finds the most significant bit that is different in the two numbers without learning its position. The sender then obliviously assigns s_0 or s_1 to that bit and randomizes all other bits. Then, the sender permutes the vector of encrypted values to prevent the receiver from learning the position of that bit and sends the vector to the receiver. The receiver decrypts the elements of the vector and stops when a value in D_S is found. See Appendix B for details of the protocol.

If s_0 and s_1 are already known to the receiver, the GT-SCOT protocol simply allows the receiver to learn whether $x < y$, $x = y$, or $x > y$. We exploit this observation in our distributed k-anonymity protocol. Here, the user sends only $\mathcal{E}_{C_l}(r + \sum_i v_i)$ to secure comparison server l. Then, the user and the server run the GT-SCOT protocol. The server uses $r + \sum_i v_i$ as the sender's input, y, and the user uses $r + k$ as the receiver's input, x. The GT-SCOT protocol guarantees that the server will not learn $r + k$ and the result of the comparison. However, it allows the user to distinguish between three cases ($\sum_i v_i < k$, $\sum_i v_i = k$, and $\sum_i v_i > k$), whereas k-anonymity does not distinguish between the equality and the greater-than case. Also, telling a user that there are precisely k people in the query area enables tracking attacks, similar to the one outlined in Section 1. To avoid the equality case, we have the secure comparison server compute and compare bit-by-bit encryptions of $2 * (r + \sum_i v_i) + 1$ and $2 * (r + k)$.

4.3 Defence against Binary Search

A flaw of our protocol is that, using binary search, a user might still be able to learn the precise number of users in a cell. Namely, the user could present $\mathcal{E}_{C_l}(r + \sum_i v_i)$ multiple times to the secure comparison server, maybe with a different value of r each time. By adjusting the value of k in each run of the GT-SCOT protocol, the user can perform a binary search for the actual value of $\sum_i v_i$.

To prevent this attack, we use expiring tickets. Instead of sending $\mathcal{E}_{C_l}(v_j)$ to the user, a location broker sends $\mathcal{E}_{C_l}(v_j + r_j)$ and a ticket that contains $E_{C'_l}(r_j)$, where r_j is a random number changing with each request and $E_{C'_l}(\cdot)$ is the RSA encryption function using RSA public key C'_l of the secure comparison server. A location broker also includes an expiration date in the ticket and signs the ticket. A secure comparison server will decrypt all $E_{C'_l}(r_j)$ and subtract $\sum_i r_i$ from $r + \sum_i (v_i + r_i)$. The server also remembers tickets till their expiration date and refuses to re-use a ticket seen previously. This way, the attack mentioned above will fail, even if r is changed. Also, the user cannot use fresh tickets with a previously presented encrypted sum, since the r_i value will be different, meaning the secure comparison server cannot compute the correct input value for the GT-SCOT protocol and the user cannot learn any useful information from this operation. Similar to traditional k-anonymity approaches based on a central trusted server, a location broker can limit the query frequency of users.

Figure 2 illustrates our distributed k-anonymity protocol based on the GT-SCOT protocol and expiring tickets.

4.4 Choice of Secure Comparison Server

Having multiple secure comparison servers distributes load and avoids a single point of failure. Moreover, as we show below, it also limits the impact of collusion between a user and a secure comparison server. This is important because collusion might allow a user to learn the total number of registered users in her query area.

Our distributed k-anonymity protocol requires that we pick one of the secure comparison servers at the beginning of a protocol run. Because of possible collusion, we cannot let a user choose a server. Instead, we need to choose a server such that, over time, the risk of a user working together with a colluding server is limited by k/n, where n corresponds to the number of secure comparison servers, with k of them colluding with the user. Therefore, we cannot statically assign a secure comparison server to a user, since we might

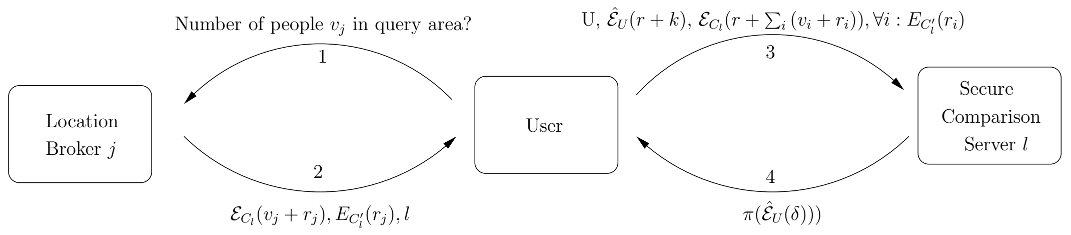
Number of people v_j in query area?
1
$\mathcal{E}_{C_l}(v_j + r_j), E_{C'_l}(r_j), l$
2
Location Broker j
User
U, $\hat{\mathcal{E}}_U(r + k)$, $\mathcal{E}_{C_l}(r + \sum_i (v_i + r_i)), \forall i : E_{C'_l}(r_i)$
3
Secure Comparison Server l
4
$\pi(\hat{\mathcal{E}}_U(\delta)))$

Figure 2: Distributed k-anonymity protocol. U and C_l are the Paillier public key of the user and secure comparison server l, respectively. C'_l is the RSA public key of secure comparison server l. $\mathcal{E}_{\mathcal{A}}(\cdot)$ denotes regular Paillier encryption with public key A. $\hat{\mathcal{E}}_A(\cdot)$ denotes *bit-by-bit* Paillier encryption with public key A. $E_A(\cdot)$ denotes RSA encryption with public key A. $\pi(\cdot)$ is a random permutation. δ is the result vector obliviously computed by secure comparison server l while executing the GT-SCOT protocol. The ciphertexts $E_{C'}(r_i)$ also include an expiration date and are signed by location broker i (not shown).

be unlucky and pick a colluding one. Moreover, a user might decide not to trust servers maintained by particular organizations, and she might refrain from using our system if we forced her to use such a server all the time.

Another strategy is to have each location broker randomly choose a secure comparison server for a query. However, this strategy has two flaws: First, our protocol requires that all brokers choose the same server, which will likely not be the case here. Second, if a user is assigned to a non-colluding server, she can repeat her query until a colluding server is chosen. To address these flaws, we need an assignment scheme that, within a particular time frame, has all brokers assign the same server to a particular user. The length of the time frame should be such that the impact of using a malicious server within the entire duration of the time frame is limited (e.g., the time frame should be shorter than a day) and such that if a user decides to perform an attack at a particular moment in time, her expected waiting time till she is being assigned a colluding server is so long that the attack environment (e.g., locations of users) will likely have significantly changed by then (e.g., the time frame should be longer than a minute).

We now present our algorithm for choosing a secure comparison server. There is a sign-up server that randomly assigns each user to one out of n groups when she signs up to our system, where n corresponds to the number of secure comparison servers. (The assignment can expire to deal with new secure comparison servers.) The sign-up server could be identical to the directory server. To determine the secure comparison server to be used for encrypting the response to a user's query, each location broker executes a well-known, deterministic function that maps the identity of the user's group to a secure comparison server. (Directly mapping the user's identity, instead of the identity of the user's group, is not an option because letting a broker know of a user's identity would make the user trackable and identifiable by a location broker.) This mapping function is bijective and depends on the current time. For example, the function can be expressed as $f = (ep + id) \bmod n$. Here, time is split into epochs, with ep indicating the current epoch ($ep \geq 0$). Our suggested duration of an epoch is one hour. id is the identity of a user's group ($0 \leq id < n$), as reported by the user when sending her query to the location broker. We can also design more complex mapping functions, such as a function that changes the order in which a group is mapped to a server over time.

To ensure that the group identity reported by a user is accurate, we take advantage of cryptographic group signatures [10]. In a group signature scheme, any member of a group can sign a message without a signature verifier being able to infer which member generated the signature. Only the group manager, that is, the entity generating and distributing signing keys to group members, can trace a signature to its issuer. In our scheme, the group manager corresponds to the sign-up server. When querying a location broker, a user creates a group signature to prove membership in her group to the broker. This allows the user to remain anonymous within her group while not being able to claim membership in other groups.

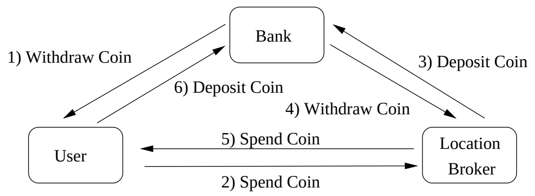

Figure 3: Defence against multiple registrations based on e-cash. By being given only one coin, a user can register only once. The user is returned her coin when de-registering.

In general, to minimize the risk of collusion, we do not let random people deploy a secure comparison server. Instead, the directory server should vet a server before listing it, similar to the limited vetting done by a directory server in Tor.

5 Location Registration

As mentioned in Section 3.2, consistent with earlier work, our threat model assumes that a location broker can detect attempts by a user to register with the broker multiple times in parallel. Having multiple location brokers, as it is the case in our solution, introduces a new vulnerability. Namely, a user could register multiple times, but each time with a *different* location broker. This way, other users might be wrongly told that their k-anonymity preference is satisfied. There are both technical and non-technical controls for this vulnerability. Charging money is an example of a non-technical control. Namely, if location brokers are maintained by operators of cellphone networks as a service to customers, a user would would have to buy multiple cellphones and plans to register in parallel with multiple brokers, which makes the attack expensive. In the remainder, we present a technical control that does not make any assumptions about the underlying communication technology. Since we control the vulnerability, our threat model for user behaviour can remain identical to the threat model in the earlier work.

There are two naïve approaches to prevent a user from registering with different location brokers concurrently. In the first one, a location broker contacts the other location brokers whenever a user registers and inquires whether the user has already registered with one of them. This approach raises privacy concerns and is expensive in terms of performance. The second approach has each broker keep records of its registered users. Periodically, the brokers compare records and try to detect misbehaving users. The main problems of this approach are the privacy concerns raised by the record keeping and by the comparison and that it is retroactive.

Our solution gets around these problems. It is based on e-cash [9] and is outlined in Figure 3. E-cash allows a player to withdraw a coin from the bank and to spend it with a second player. The second player deposits the received coin with the bank. In our solution, a user gets one (and only one) coin from the bank. The role of the bank can be assumed by the directory server. When a user registers with a location broker, she spends her coin at the broker. Since the user has only one coin, registering with another broker amounts to double spending of the coin. The other broker detects this double spending when depositing the coin, either immediately in case of an online e-cash scheme [9] or in a delayed way in case of an offline scheme [8]. In the former case, the broker will deny registration. In the later case, the bank will learn the user's identity and will ban her from the system (see Section 6.3). Whenever a location broker deposits a user's (valid) coin, the broker also withdraws a fresh coin from the bank. (Alternatively, since location brokers are not malicious, the bank can periodically give a set of coins to the broker for increased performance.) When the user wants to de-register, she asks the broker to spend this coin by giving it to the user. The user then deposits the coin and withdraws a fresh coin from the bank, which she can later spend at another broker.

The benefits of our solution are that it does not require all location brokers to be contacted for a registration and that location brokers do not need to keep records of registered users after their de-registration. Moreover, since e-cash is anonymous, the bank cannot learn which user registers with which location broker, and a location broker cannot learn a user's identity and where a user has registered previously.

In case a location broker crashes, a registered user will not be able to register with a new broker. Here, we have the user contact the bank with a proof of registration issued by the broker. The bank will then issue a new coin to the user. When the broker comes back up, it will re-synchronize with the bank.

6 Security Analysis

In this section, we first review how our protocol defends against the threats listed in Section 3.2, where we assume that location brokers and secure comparison servers are honest-but-curious. In the remainder, we discuss how our architecture can be extended to defend against malicious parties.

6.1 Threat Analysis

In our protocol, location brokers do not interact with each other, so they cannot learn the location of users in the query area who are registered with other brokers, not even their total number. Our architecture is based on e-cash and group signatures, which allows users to remain anonymous to a location broker. A user should not directly connect to a broker during a registration or query operation. Information about this connection (e.g., the user's IP address) might allow the broker to re-identify the user and to learn her location from her query area. Instead, the user should communicate through a trusted proxy or an anonymous communication network (e.g., Tor). In practice, the location broker that a user registers with might already know the user's identity and might forbid anonymous registrations. (E.g., an operator of a cellphone network often knows the identity of its customers.) Here, a user would still have to hand over a coin to the broker to detect multiple registrations. In this setup, the location broker that a user registers with can serve as the trusted proxy for contacting the other brokers during a query operation. Despite the communication between these brokers and a user being proxied, the brokers might be able to re-identify the user based on her query area if the area is associated with the user or under physical surveillance by a broker. To address this threat, the user should query the brokers only if she knows that there are at least $k - 1$ other people in the query area, that is, we end up with a chicken-egg problem. In our architecture, as stated in Section 3.1, we choose the width of a cell such that there is a realistic chance that multiple users can be located in the cell, which makes this attack hard.

A secure comparison server learns no useful information during a comparison operation, not even its outcome. A server also gains no benefit from colluding with other secure comparison servers. To remain anonymous to a location broker, a user should not directly connect to a secure comparison server, since the tickets issued by the broker allow a secure comparison server that is run by the same organization to link a query and a comparison operation.

A user learns only whether the total number of users in her query area is at least k. The expiring tickets prevent her from learning the actual number of users with a binary search.

The directory server cannot learn any location information, because users do not retrieve individual records for their current cell from the server. The published directory is signed, which prevents the directory server from misbehaving.

6.2 Malicious Servers or Brokers

Assume that a malicious secure comparison server fails to correctly execute some of the steps in the GT-SCOT protocol. While it is not possible for the server to learn the total number of users, due to the randomness added by the user, the server could misbehave with the intent to give the user the impression that there are at least k registered users in an area, even if this is not the case, or vice versa. We could address this concern by adding zero-knowledge proofs to each step of the protocol, proving that the step was executed faithfully. For example, Groth [18] proposes an efficient scheme for proving in zero-knowledge the correctness of a permutation of homomorphic encryptions. As it turns out, this scheme requires three additional rounds of interactions between the prover and the verifier, which makes it expensive for mobile devices. Therefore, in our scheme, we choose a retroactive approach. We have a secure comparison server log the random values used in its encryption and permutation operations. Furthermore, the server has to sign all its generated messages to achieve non-repudiation. If users suspect misbehaviour, they, likely in collaboration with the directory server, can force the secure comparison server to reveal the logged values and its private key and can validate the server's computations.

Similar to the secure comparison server, a malicious location broker can misbehave while executing our protocol. In particular, a broker can encrypt a value that is different from the actual number of users registered in an area. It is possible to ask a broker to keep a record of all its actions. However, this record would have to include location registrations of users, which is problematic in terms of privacy. We prefer a less invasive approach. If users suspect misbehaviour by a location broker (and misbehaviour by a secure comparison server can be excluded, based on the above mechanism), they report the set of location brokers from which they retrieved information to the directory server. Over time, this will allow the directory server to single out a particular location broker.

6.3 Malicious Users

Malicious users could report wrong locations to a location broker. As it turns out, a complete defence against this attack is likely impossible. A determined attacker can give her mobile device to another user or simply tamper with the location reporting mechanism on her mobile device. A user could also acquire multiple devices, maybe under different identities, and use them to register multiple times. As mentioned in our threat model (see Section 3.2), these threats are not new to our system; they also arise in previous schemes. Let us outline some mechanisms that make these attacks harder.

A location broker might be able to detect wrongly reported locations. For example, if a broker is controlled by the operator of a WiFi network, the operator can ensure that a reported location is close to the WiFi access point from which the registration request was sent. An operator of a cellphone network can verify whether the reporting device is close to a particular cellphone tower.

We can also exploit the sign-up process, as introduced in Section 5, to defend against malicious users. The sign-up server can require physical identification, which reduces the danger of a user signing up multiple times. However, this approach makes the system more difficult to use. An alternative is to ask the user for a credit card number, including her name and billing address. This option becomes especially attractive if the system charges its users in the first place. Billing for the usage of our system itself can become a mechanism for reducing misbehaviour, because an attacker might not have the necessary resources for a large-scale attack.

7 Evaluation

In this section, we evaluate our distributed k-anonymity protocol. We first examine the cost of contacting a location broker, followed by the cost of contacting a secure comparison server. In our evaluation, we focus on the cost of the homomorphic encryption operations and of the GT-SCOT protocol. To the best of our knowledge, no measurement-based evaluation of this protocol has been published, whereas there are such evaluations of the other two cryptographic protocols used in our architecture. For example, Belenkiy et

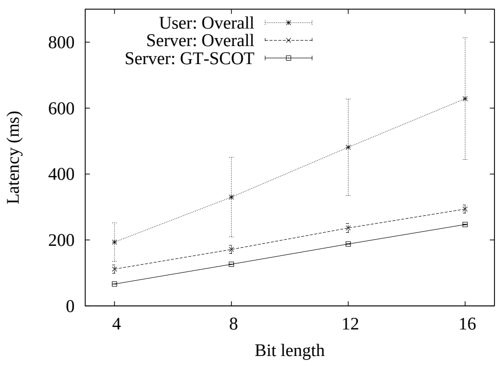

Figure 4: Latency experienced by the secure comparison server and the user in relationship to the bit length used in the GT-SCOT protocol. We show mean and standard deviation.

al. [2] evaluate e-cash and Cornelius et al. [12] evaluate group signatures.

We implemented our protocol using the OpenSSL and NTL [31] libraries. The key size for RSA and Paillier is 1024 bits. We deploy a location broker and a secure comparison server on a 2.4 GHz Intel Xeon Dual Core running Linux 2.6.24. The user has a slow laptop (a ThinkPad T43 with a 2 GHz Intel Pentium M running Linux 2.6.22) to approximate the capabilities of a modern smartphone. Communication runs over WiFi and is protected against eavesdroppers with TLS using AES128 in CBC mode with an ephemeral Diffie-Hellman key exchange for forward secrecy.

7.1 Location Broker

We examine the performance of querying a location broker for the number of people in the query area and of adding this number to an existing encrypted sum. In the experiment, when a user connects to a location broker, the location broker sends back a Paillier encrypted random value. The user then performs a homomorphic addition. We repeat the experiment ten times and report mean and standard deviation.

The overall delay experienced by the user is 39.9 ± 0.7 ms. It takes 32.5 ± 0.7 ms to set up a TLS connection, which includes client and server authentication. The server takes 7.4 ± 0.0 ms to Paillier encrypt a random value. The cost of the homomorphic addition operation by the user is negligible. In summary, setting up the TLS connection is about four times as expensive as the Paillier encryption operation. As mentioned in Section 6.1, to hide her identity, a user might not directly connect to a location broker. Here, the cost of encryption in relation to the cost of connection setup becomes even smaller.

In practice, the user will likely contact multiple location brokers. Apart from the addition operation, whose cost is negligible, the brokers can be contacted in parallel. If this is not feasible for the user's device, the overall delay will be linear in the number of location brokers. We envision that this number is small (5-10 brokers) in most scenarios. This number reflects the number of cellphone and WiFi network operators providing coverage for the query area, which tends to be small. In addition, there might be a small number of independently operated location brokers.

The user also needs to Paillier encrypt the random value that she will add to the encrypted sum of users reported by the location brokers. This encryption takes 67.2 ± 0.5 ms. As expected, the encryption operation is slower on the laptop than on the server. However, as opposed to the other operations, this encryption can occur offline. Moreover, the user can use an encrypted value multiple times for a secure comparison server.

7.2 Secure Comparison Server

We evaluate the performance of the GT-SCOT protocol for different bit lengths of the bit-by-bit homomorphic encryption operation. We vary the bit length between 4 and 16 and perform fifty runs for each configuration.

We present our results in Figure 4. The bottom graph shows the cost of the sender side of the GT-SCOT protocol, which varies between 66.3 ± 0.3 ms for a bit length of 4 and 247.9 ± 0.9 ms for a bit length of 16. The middle graph corresponds to the overall cost by the server. In addition to the sender side of the GT-SCOT protocol, it also includes the cost of setting up a TLS connection and Paillier decryption of the total number of users. Finally, the top graph shows the overall latency, as experienced by the user. It varies between 193.5 ± 58.4 ms for a bit length of 4 and 628.6 ± 184.7 ms for a bit length of 16. The overall latency corresponds to the overall cost by the server plus the cost of the receiver side of the GT-SCOT protocol, which takes 77.5 ± 47.7 ms for a bit length of 4 and 330.4 ± 179.6 ms for a bit length of 16. The standard deviation is large, because the user decrypts the permuted result vector received from the secure comparison server element by element and stops as soon as she finds the server's answer.

In our implementation, we let a user choose the bit length. In practice, we expect that bit lengths between 8 and 12 will be used mainly, depending on the number of location brokers covering the query area and the maximum number of reported users for the query area (which is different from the number of registered users in the query area). If there are a bits in total, we can support up to 2^c location brokers and up to $2^b - 1$ reported users per location broker per query area, where $a = b + c + 2$. (This implies $0 \leq k \leq 2^c * (2^b - 1)$.) A user informs a broker of her choice of b; if there are more than $2^b - 1$ registered users in the query area, the broker simply reports $2^b - 1$ users. (As stated in Section 4.1, each broker adds a random value to its reported number. We can ignore these values here, since they are subtracted by the secure comparison server before running the GT-SCOT protocol. This will also revert any wrap-arounds that might have occurred due to choosing a large random value.) The two remaining bits leave space for adding the random number r ($0 \leq r \leq 2^c * (2^b - 1)$)[1] chosen by the user to the total number of users, which requires at most $b + c + 1$ bits, and for allowing the secure comparison server to double the resulting sum to avoid the equality case. For example, for a bit length of 8, we can support up to 8 location brokers and 7 reported users per broker per query area. Here, the overall latency experienced by the user is 330.0 ± 120.7 ms. For a bit length of 12, we can support up to 16 location brokers and 63 reported users per broker per query area. Here, the overall latency experienced by the user is 481.2 ± 146.2 ms. In short, we expect the overall latency to be noticeable, but tolerable.

The user also needs to perform a bit-by-bit encryption of the sum of her privacy preference and of her chosen random value. The cost of this operation varies between 210.4 ± 0.8 ms for a bit length of 3 and 864.4 ± 115.0 ms for a bit length of 15. (The large variation is an artifact of using a bit length of 15. The variation is small for a bit length of 16, which has a larger mean. We are investigating this behaviour.) However, as opposed to the other operations, this encryption can be done offline. Moreover, the user can use an encrypted value multiple times for a secure comparison server.

8 Discussion

In this section, we give some guidelines that help a user choose among the set of location brokers during registration. Then, we discuss an alternative protocol for distributed k-anonymity and its drawbacks.

[1]If $r = 0$ and $\forall i : v_i = 0$, the secure comparison server can infer these values. Similar for the maximum case. To lower the probability for this scenario, the user can dynamically increase (decrease) her lower (upper) bound for r and keep her choice secret.

8.1 Choice of Location Broker

From a global point of view, having multiple location brokers provides more privacy than traditional approaches based on a central trusted server, because there is no longer a single party that knows all users' location. This is an inherent benefit of our scheme and lets it integrate well with existing infrastructures. For example, every operator of a cellphone network knows the location of its customers. However, there is no single entity that knows the location of all cellphone users.

From a single user's point of view, our approach does not necessarily provide more privacy. In particular, if a user always registers her location with the same broker, this broker will have complete information about the user's whereabouts. In some scenarios, it actually makes sense for a user to always register with the same broker, and registering with different brokers would reduce the user's privacy. Namely, many users carry a cellphone with them that is always on. In order for the operator of the cellphone network to be able to route phone calls to the cellphone, the operator needs to know the cellphone's current location. In environments with a dense deployment of cell towers, this information is very accurate. Here, assuming the operator provides a location broker, it makes most sense for the user to register with this broker. Registering with a location broker operated by somebody else just has another party know the user's location.

However, there are scenarios where it does make sense for a user to register her location with different location brokers over time to prevent a single broker from learning her location profile. These are scenarios where cellphone towers are sparse and a cellphone can determine its fine-grained location with the help of GPS, where a cellphone network operator does not provide a location broker, or where a user takes advantage of different communication providers over time (e.g., a company's WiFi during work hours and WiFi in a Starbucks over lunch). We now discuss some user strategies for choosing a location broker.

For many users, their movement patterns are regular (e.g., there is a commute to work in the morning at always roughly the same time and a reverse commute in the evening). A user could randomly choose a location broker whenever she switches cells. However, especially if the set of candidate brokers is small, this strategy could still result in a location broker ultimately obtaining a nearly complete picture of the user's daily commute pattern. Namely, a location broker can simply piece together location information gathered during different days for the same user, based on the time when a user enters or leaves a cell.

A better strategy for a user is to assign a single location broker to each cell that is part of her daily movement patterns and to always use the assigned broker when registering in a cell. In the example, a user registers with her company's location broker during work hours and with a third-party broker during lunch (assuming Starbucks does not provide a broker). This way, a broker will not be able to establish complete location profiles for a user. The drawback of this approach is that a user needs to remember which broker is assigned to which cell. However, her mobile device can assign and remember brokers on her behalf.

When choosing a location broker, especially among brokers that do not already have location information about a user, the user must decide which of the candidates she trusts not to leak her location information. To let a user establish this trust, the directory server should vet a broker before listing it. (As stated in Section 4.4, the directory server might also vet secure comparison servers.) Moreover, the directory server should keep the number of brokers per coverage area small since each broker needs to be contacted for a query. Obvious candidates for running a location broker in a coverage area are organizations that already have location information about people in this area, such as cellphone network operators or WiFi providers. Other candidates are organizations that have an interest in the well-being or privacy of people (e.g., a municipality or the Electronic Frontier Foundation). Unlike in the case of Tor, where any individual can contribute a relay, we do not expect individuals to run location brokers due to privacy concerns.

8.2 Perfect Threshold Set-Union

In our scheme, collusion between a secure comparison server and a user reveals the total number of people in an area. Our algorithm to choose a secure comparison server discussed in Section 4.4 makes collusion hard. We could avoid this threat completely by getting rid of the secure comparison server and by using the privacy-preserving Perfect Threshold Set-Union protocol proposed by Kissner and Song [23]. Here, if n users register a particular cell as their location with a location broker, the location broker includes the unique identifier of the cell n times in its private set. If a user wants to know whether there are at least k users in a cell, the location brokers jointly perform the Perfect Threshold Set-Union protocol with the threshold set to k. As a result, the servers learn which cell identifiers appear at least k times in the union of the private sets, that is, which cells contain at least k users. Unfortunately, the slow performance of the protocol makes it impractical for our usage scenario. The protocol needs to calculate the product of an unencrypted polynomial and an encrypted polynomial many times with large coefficients and high degrees using homomorphic encryption. In the case of five location brokers, ten registered users with each broker, and $k = 2$, the protocol takes several minutes to finish in our sample implementation.

Instead of distributing the task of the central server among multiple location brokers, another approach is to hide location information from the central server in the first place. Here, users register with and query the server without letting the server learn a user's location. During registration, a user sends a vector consisting of homomorphically encrypted values to the server. All entries in the vector are encryptions of zero, except the entry for the user's current location, which is an encryption of one. The server maintains the sum of all received vectors. (The user must also submit a vector upon leaving a cell.) For answering queries, the server and a user rely on an oblivious transfer scheme [28], similar to the one suggested by Kohlweiss et al. [24] for retrieving location-specific information from a location-based service, but augmented such that the user learns only whether the current number of users in a cell is at least k. To the best of our knowledge, no such protocol is yet known to exist. Furthermore, this approach can be computationally expensive, since the server needs to update each of the cells whenever a user (de-)registers with the server.

9 Conclusions and Future Work

We have presented a protocol for location privacy based on k-anonymity that needs neither a single trusted server nor users to trust each other. Our sample implementation and its evaluation have shown that the protocol is efficient.

In terms of future work, we are integrating our protocol into a platform for location-based services, which will allow us to gather more insights about the protocol's usability in practice.

References

[1] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid. In *Proceedings of 17th International World Wide Web Conference (WWW2008)*, pages 237–248, April 2008.

[2] M. Belenkiy, M. Chase, C. C. Erway, J. Janotti, A. Küpçü, A. Lysyanskaya, and E. Rachlin. Making P2P Accountable without Losing Privacy. In *Proceedings of 6th Workshop on Privacy in the Electronic Society (WPES 2007)*, pages 31–40, October 2007.

[3] A. R. Beresford. Location privacy in ubiquitous computing. Technical Report 612, Computer Laboratory, University of Cambridge, January 2005.

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

[5] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Anonymity in Location-based Services: Towards a General Framework. In *Proceedings of 8th International Conference on Mobile Data Management (MDM 2007)*, pages 67–79, May 2007.

[6] I. F. Blake and V. Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals. In *Proceedings of ASIACRYPT 2004*, pages 515–529, December 2004.

[7] F. Brandt. Efficient Cryptographic Protocol Design based on Distributed El Gamal Encryption. In *Proceedings of 8th International Conference on Information Security and Cryptology (ICISC)*, pages 32–47, December 2005.

[8] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In *Proceedings of EURO-CRYPT 2005*, pages 302–321, May 2005.

[9] D. Chaum. Blind Signatures for Untraceable Payments. In *Proceedings of CRYPTO '82*, pages 199–203, August 1982.

[10] D. Chaum and E. van Heyst. Group Signatures. In *Proceedings of EUROCRYPT '91*, pages 257–265, April 1991.

[11] C.-Y. Chow, M. F. Mokbel, and X. Liu. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services. In *Proceedings of 14th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS'06)*, pages 171–178, November 2006.

[12] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. AnonySense: Privacy-Aware People-Centric Sensing. In *Proceedings of 6th International Conference on Mobile Systems, Applications, and Services (MobiSys 2008)*, pages 211–224, June 2008.

[13] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of 13th USENIX Security Symposium*, pages 303–319, August 2004.

[14] M. Fischlin. A Cost-Effective Pay-Per-Multiplication Comparison Method for Millionaires. In *Proceedings of RSA Security 2001 Cryptographer's Track*, pages 457–471, April 2001.

[15] 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)*, pages 620–629, June 2005.

[16] G. Ghinita, P. Kalnis, and S. Skiadopoulos. MobiHide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries. In *Proceedings of Proceedings of 10th International Symposium on Spatial and Temporal Databases (SSTD 2007)*, pages 221–238, July 2007.

[17] G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVÉ: Anonymous Location-Based Queries in Distributed Mobile Systems. In *Proceedings of 16th International World Wide Web Conference (WWW2007)*, pages 371–380, May 2007.

[18] J. Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. In *Proceedings of 6th International Workshop on Practice and Theory in Public Key Cryptography*, pages 145–160, January 2003.

[19] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proceedings of 1st International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, pages 31–42, May 2003.

[20] T. Jiang, H. J. Wang, and Y.-C. Hu. Preserving Location Privacy in Wireless LANs. In *Proceedings of 5th International Conference on Mobile Systems, Applications, and Services (MobiSys 2007)*, pages 246–257, June 2007.

[21] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving Anonymity in Location Based Services. Technical Report TRB6/06, School of Computing, The National University of Singapore, 2006.

[22] A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, and D. Kotz. AnonySense: Opportunistic and Privacy-Preserving Context Collection. In *Proceedings of 6th International Conference on Pervasive Computing (Pervasive 2008)*, pages 280–297, May 2008.

[23] L. Kissner and D. Song. Privacy-Preserving Set Operations. In *Proceedings of CRYPTO 2005*, pages 241–257, August 2005.

[24] M. Kohlweiss, S. Faust, L. Fritsch, B. Gedrojc, and P. Preneel. Efficient Oblivious Augmented Maps: Location-Based Services with a Payment Broker. In *Proceedings of 7th Privacy Enhancing Technologies Symposium (PET 2007)*, pages 77–94, June 2007.

[25] S. Mascetti and C. Bettini. A Comparison of Spatial Generalization Algorithms for LBS Privacy Preservation. In *Proceedings of International Workshop on Privacy-Aware Location-based Mobile Services (PALMS)*, May 2007.

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

[27] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of EUROCRYPT '99*, pages 223–238, May 1999.

[28] M. O. Rabin. How to Exchange Secrets with Oblivious Transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[29] P. Samarati and L. Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Technical Report SRI-CSL-98-04, SRI International, 1998.

[30] H. Shin, V. Atluri, and J. Vaidya. A Profile Anonymization Model for Privacy in a Personalized Location Based Service Environment. In *Proceedings of 9th International Conference on Mobile Data Management (MDM 2008)*, pages 73–80, April 2008.

[31] Victor Shoup. NTL: A Library for doing Number Theory. `http://www.shoup.net/ntl`. Accessed September 2008.

[32] A. C. Yao. Protocols for Secure Computations. In *Proceedings of 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[33] G. Zhong and U. Hengartner. Toward a Distributed k-Anonymity Protocol for Location Privacy. In *Proceedings of 7th Workshop on Privacy in the Electronic Society (WPES 2008)*, October 2008.

A Paillier

In the Paillier cryptosystem [27], a user Alice selects random primes p and q and constructs $n = pq$; plaintext messages are elements of $\mathbb{Z}_n$, and ciphertexts are elements of $\mathbb{Z}_{n^2}$. Alice picks a random $g \in \mathbb{Z}^*_{n^2}$ and verifies that $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$ exists, where $\lambda = \text{lcm}(p-1, q-1)$ and $L(x) = (x-1)/n$. Alice's public key is then (n, g) and her private key is (λ, μ).

To encrypt a message m, another user Bob picks a random $r \in \mathbb{Z}^*_n$ and computes the ciphertext $c = \mathcal{E}(m) = g^m \cdot r^n \bmod n^2$. To decrypt this message, Alice computes $\mathcal{D}(c) = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$, which always equals m. Given $\mathcal{E}(m_1) = g^{m_1} \cdot r_1^n \bmod n^2$ and $\mathcal{E}(m_2) = g^{m_2} \cdot r_2^n \bmod n^2$, Bob can easily compute $\mathcal{E}(m_1 + m_2) = \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) \bmod n^2 = g^{m_1+m_2} \cdot (r_1 r_2)^n \bmod n^2$.

B GT-SCOT

Blake and Kolesnikov [6] propose the Greater Than - Strong Conditional Oblivious Transfer (GT-SCOT) protocol. The protocol has two participants, a receiver and a sender. The receiver and sender have private inputs x and y, respectively. The sender has two secrets, $s_0 \in D_S$ and $s_1 \in D_S$, where D_S is a subset of $\mathbb{Z}_n$. The sender wants to send s_0 to the receiver if $x < y$ and s_1 if $x > y$, but is oblivious about which secret is sent. In short, the sender cannot learn whether $x < y$ or $x > y$. The details of the protocol are as follows:

1. The receiver sets up the Paillier encryption scheme and chooses her public and private key. She randomly encrypts each bit x_i of x, where x_i denotes the i^{th} most significant bit in the n-bit binary representation of x, with her public key, R, and sends $(R, \mathcal{E}_R(x_1), ..., \mathcal{E}_R(x_n))$ to the sender.

2. The sender computes the following, where $i = 1..n$:

   (a) an encryption of the difference vector d, where $d_i = x_i - y_i$.

   (b) an encryption of the flag vector f, where $f_i = x_i \, XOR \, y_i = (x_i - y_i)^2 = x_i - 2x_i y_i + y_i$.

   (c) an encryption of vector γ, where $\gamma_0 = 0$ and $\gamma_i = 2\gamma_{i-1} + f_i$.

   (d) an encryption of vector δ, where $\delta_i = d_i + r_i(\gamma_i - 1)$, where $r_i \in_R \mathbb{Z}_n$.

   (e) a random encryption of vector μ, where $\mu_i = \frac{s_1 - s_0}{2}\delta_i + \frac{s_1 + s_0}{2}$.

   The sender sends a random permutation $\pi(\mathcal{E}_R(\mu_1), ..., \mathcal{E}_R(\mu_n))$ to the receiver.

3. The receiver obtains $\pi(\mathcal{E}_R(\mu_1), ..., \mathcal{E}_R(\mu_n))$, decrypts it, and determines the output as follows: if μ contains a single $v \in D_S$, output v, otherwise abort.

In a *random* encryption, we ensure $r \neq 1$ in the randomness part of the Paillier encryption scheme. Otherwise, we set $r = 1$, which makes the scheme much faster. In step 2 of the protocol, the sender introduces randomness only for the last computation, before sending the result to the receiver. Blake and Kolesnikov show that for properly chosen parameter values, value v obtained by the receiver equals the desired secret with high probability, that is, the probability of a false result or an abort is negligible.

In our problem setting, where the receiver simply wants to learn whether $x < y$ or $x > y$, we do not require the oblivious assignment in step 2(e). Instead, we observe that with high probability exactly one of the elements of vector δ computed in step 2(d) will equal 1 if $x > y$ and -1 if $x < y$. All elements will have random values if $x = y$. Therefore, we omit step 2(e) in our implementation, compute a *random* encryption of vector δ in step 2(d), send a permutation of it to the receiver, and modify the receiver accordingly.

There are other protocols that are applicable to our problem setting, for example, by Fischlin [14]. We choose Blake and Kolesnikov's protocol because it has both better communication complexity and better computation complexity in our application scenario. The problem that we study is different from Yao's millionaire problem [32], where two millionaires want to determine which of them is richer without revealing their net worth to each other. In the millionaire problem, both parties can learn the final result, which must be avoided in our application (see Section 4.1). Moreover, solutions for the millionaire problem require at least six communication steps [7], whereas Blake and Kolesnikov's protocol requires only three.