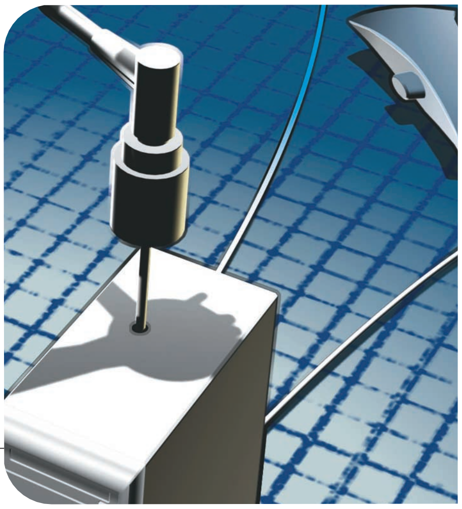
Protecting Privacy in Continuous Location-Tracking Applications

Although some users might willingly subscribe to location-tracking services, few would be comfortable having their location known in all situations. The authors investigate disclosure-control algorithms that hide users' positions in sensitive areas and withhold path information that indicates which areas they have visited.



Marco Gruteser
University of Colorado

Xuan Liu
IBM T.J. Watson Research Center

Recent technological advances in wireless location tracking (such as that found in cell phones[1] and radio frequency identification (RFID) chips, among others) present unprecedented opportunities for monitoring individuals' movements.[2] While such technology can support useful location-based services (LBS), which tailor their functionality to a user's current location, privacy concerns might seriously hamper user acceptance.[3]

LBSs can be classified into three types: position awareness, sporadic queries, and location tracking. *Position awareness* refers to devices that monitor an individual's position, such as in-car navigation systems or GPS-enabled PDAs, but that only use such information internally. *Sporadic queries* apply to services in which an individual initiates the transfer of position information to an outside service provider. These queries contain only the user's current position—as in point-of-interest queries to find the nearest hotel, for example. Finally, *location-tracking* services receive frequent updates of an individual's position—for example, experimental automotive telematics applications, which seek to improve transportation through the use of information technology, use such updates to estimate highway gridlock and reroute drivers around traffic jams.

Continuous location-tracking applications exacerbate privacy problems because individuals' desired levels of privacy can be situation-dependent. Even if users consent to having certain second parties track their locations, we believe that few would be comfortable revealing their position in every situation. The tracked device, for example a cell phone, could ask the user to approve every location-tracking request. However, if these requests arrive too frequently, manually approving every request or switching the tracking on and off becomes too cumbersome. Context-aware privacy policies[4,5] enable automatic privacy decisions, but such policies tend to get complex and difficult to define. Furthermore, suppressing location updates in a few *sensitive areas*, whose visits an individual prefers not to reveal, is insufficient. In continuous location-tracking applications, the receiver of the updates accumulates prior movement data that might help infer where the individual has visited.

In this article, we analyze algorithms that suppress location updates and thus hide visits to sensitive areas. We introduce the location inference problem—an adversary can infer supposedly hidden locations from prior or future location updates—and present algorithms to address this problem. A synthetic urban mobility model helps us analyze their effectiveness.

Privacy challenges in location tracking

LBSs usually involve at least three parties: the user, a provider of positioning technology, and at least one service provider. The organization supplying the positioning technology uses a location broker to deliver the user's location to service providers. For example, cell-phone providers, which deploy positioning technology for their cell phones, could implement such a broker on a server in their network infrastructure to deliver cell-phone position information to external service providers, which might offer hotel-finders or navigation applications.

Users establish a trusted relationship with the organization operating the positioning technology and location broker as part of their service contract. This is typically a longer-term business relationship, in which users have time to study the service contracts and learn about the company, whereas they might only sporadically use service providers. Thus, the main objective is to protect the user's privacy from these external service providers.

Therefore, a location broker typically offers privacy-enhancing technologies that govern the circumstances under which external subscribers may access an individual's location information. These technologies would likely ask the user for approval when a new service provider first requested the user's location, and might repeatedly ask the user to renew this approval. They might also let users specify their preferences: they could automatically grant access to certain classes of service providers or temporarily deny access for approved service providers—for example, via time constraints or when the user visits certain defined locations. We also assume that a location broker implements appropriate security measures, such as authenticating service providers.

Problems in protecting privacy

Users usually request external services and approve their location-tracking requests because they find the applications useful, for help in finding hotels or routing around traffic jams, for example. Therefore, users do not wish to withhold all information. However, in a few situations—when users enter certain sensitive areas, for example—they will want their locations to remain private. The service provider offers users methods to define these sensitive areas by proposing default policies that vary in their degree of restrictiveness but that users can customize according to their needs. For example, one default policy, implemented in the location broker using appropriate maps, could enable location tracking on public streets but prevent tracking in buildings or on private property. Other policies could further distinguish between building types (such as residential or commercial).

We phrase the privacy problem with regard to an adversary, which can be any party that has gained access to the location records that were transmitted to a service provider. The adversary seeks to track a user's location inside sensitive areas, or simply to determine which sensitive areas a user has visited, but does not have access to the location broker.

Our challenge, then, is to provide a higher degree of privacy for such sensitive areas. To this end, we must distinguish between weak and strong privacy protection for a sensitive area. With weak privacy protection, location updates are not released from that area; an adversary cannot say with certainty that a user visited this area. Strong privacy protection offers more—it also guards against *location inference*, when an adversary can probabilistically infer from prior or future location updates that the user has visited a specific sensitive area. We restrict this to inferences based on location updates; we don't take into account adversaries who might have additional information about locations or users, such as users' preferences for a certain type of location.

However, strong privacy protection should not unnecessarily decrease availability of location information in *insensitive areas*, which users don't mind others knowing they have visited, because it would likely degrade the quality of the service. As stated previously, because users sign up for the tracking applications, they must perceive them as useful, and are not interested in blocking all information. Thus, we can summarize the problem as follows: Classifying areas as either sensitive or insensitive, the privacy-protection algorithm should minimize position inaccuracy for third-party applications when an individual is located in an insensitive area and maximize position inaccuracy, especially the uncertainty about which building an individual has entered when the individual is in a sensitive area.

Disclosure-control algorithms

We developed three disclosure-control algorithms that protect individuals' sensitive locations by seeking to maximize the adversary's uncertainty about the visited areas. Algorithms should effectively protect individuals' location privacy when they enter sensitive areas and ensure that no one can infer their locations from previously released (or future) information. However, the algorithms should not overly restrict the location information available to applications in insensitive areas.

Architecture

Figure 1 illustrates how mechanisms that control the release of location updates can be embedded in a location broker. The location broker contains two major software components: a *notification manager* that keeps track of location requests from external service providers and a *privacy manager* that stores users' privacy preferences and executes the disclosure control algorithms to determine whether locations can be revealed to the requesting service provider.

Specifically, the components interact as follows. The user's device periodically updates its location to the location broker. For each location update from users, a notification manager identifies the service providers that subscribe to these updates. Before a location update is forwarded to a service provider, the notification manager requests permission from a privacy manager, which checks the user's privacy policy for any constraints described in prior work. For example, the policy could restrict access to certain classes of service providers, certain times of day, or dependent on the user's current location.[4] Such spatial constraints are represented in the location-sensitivity map, which classifies locations as either insensitive or sensitive according to the users' policy and de-

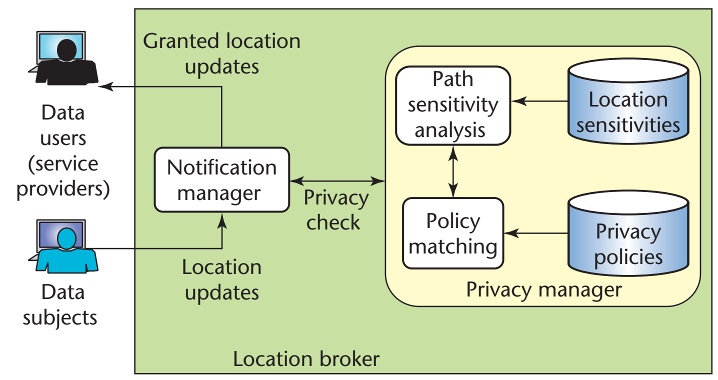


Figure 1. Architectural context. The location broker uses a notification manager and privacy manager component to deliver user location updates to service providers. The notification manager determines which service provider is interested in a location update, and the privacy manager checks the users' policies and executes disclosure-control algorithms to determine whether each service provider is allowed to receive the location update.

```
visitedSensitiveArea = false
For each location update {
 If the new location belongs to a new zone{
  If not visitedSensitiveArea {
   Disclose path
  }
  Delete path
  visitedSensitiveArea = false
  Disclose the new zone
 }
 Add current location to path
 If current location inside sensitive area {
  visitedSensitiveArea = true
 }
}
```

Figure 2. The *k*-area algorithm. This algorithm suppresses location updates in a region around the entered sensitive area, so that the area remains indistinguishable from at least *k*–1 other sensitive areas. It assumes a partition of the map wherein each partition contains at least *k* sensitive areas.

fault settings. If these constraints allow the data release, the privacy manager then conducts the sensitivity analysis (using the algorithms explained in the next section) to reach a final decision. This process repeats for each service provider.

Algorithms

The key challenge in protecting privacy with these applications is reducing the possibility of correctly inferring sensitive locations from user-revealed path information. To this end, the algorithms reduce location information not just in but also around sensitive areas that an individual visits. In dense areas such as cities, however, the user is typically surrounded by such sensitive areas. So, determining whether a subject is about to enter that area or just passing by is difficult. We compare three algorithms—base, bounded-rate, and *k*-area—to characterize this problem's magnitude and analyze possible solutions. The base and bounded-rate algorithms serve as a baseline and illustrate simple approaches to the problem. The more sophisticated algorithm, *k*-area, takes into account knowledge about sensitive areas' locations.

Our experiment with these algorithms assumes a *sensitivity map* that classifies locations as either insensitive or sensitive. This map could be generated for each user based on the sensitive areas specified in their individual privacy policy, for example. The specifics of this process are outside this article's scope, however, because the algorithms are independent from the way in which the sensitivity map is generated.

Base. The base algorithm releases only location updates in areas classified as insensitive in the sensitivity map. The algorithm checks the sensitivity map independently for each location update the client sends to the broker.

Bounded-rate. The bounded-rate algorithm ensures that updates are not sent with a frequency higher than a predefined threshold—in addition to the privacy controls of the base algorithm. Thus, for lower frequency thresholds, this algorithm reduces the amount of information released in insensitive areas to make it more difficult for an adversary to infer a user's visits to sensitive areas. For example, lowering the frequency means that the user will travel further between each location update; thus it becomes less likely that a location update will be released just before an individual enters a sensitive building.

k-area. The *k*-area algorithm restricts location updates only when an individual enters a sensitive area. Furthermore, location updates should be released only when they do not give away which of at least *k* sensitive areas the user visited.

To this end, we partition a sensitivity map in zones that include at least *k* distinct sensitive areas. A *distinct sensitive area* is an area that can be reached from at least one public area and from which no other sensitive areas can be reached without traveling through a public area. For example, each building on a city block would typically comprise a distinct sensitive area because it has an entrance from a public road and its walls separate it from other distinct areas. Each zone must include all entrances to the contained sensitive areas.

We expect that an approximate partition could be computed from maps, aerial imagery, or possibly from collected movement traces by applying clustering algorithms. Using this partition, we define the *k*-area algorithm as described in Figure 2. All location updates in one zone are stored and not released until the individual crosses a zone boundary. If the individual has visited a sensitive area in the previous zone, the algorithm suppresses the location updates from that zone; otherwise, it releases the updates to third-party applications. This process repeats for every visited zone.

Evaluating the algorithms

We experimentally explored the effectiveness of these three algorithms, asking the following questions:

- How well do the algorithms protect location privacy when individuals enter sensitive areas?
- How is location accuracy affected in public areas?

To this end, we obtained movement patterns from City Simulator,[6] a tool originally developed to generate mobility scenarios for evaluating spatial database indexing schemes. The simulation models a Manhattan-style city segment that comprises approximately two by three blocks with six intersections, a park, and 71 buildings up to 15 floors high. Figure 3 depicts a map of this model. It also shows an example partition of the map into 12 zones, a 12-area partition, for evaluating the *k*-area algorithm. The following rules govern people's movements in our model city: in a house or park, they walk randomly; when coming close to a building's door, a random process decides whether they enter or exit the building based on predefined enter and exit probabilities. Similarly, within a building, up and down probabilities rule how many people ascend to higher floors. On streets, people move according to a fluid model. The mean velocity is 0.1 m/s, corresponding to slow, mostly in-building, walking patterns.

Sensitivity map generation

For our experiment, we generated a sensitivity map that classifies well-frequented areas (such as streets) as insensitive and less frequented areas (buildings) as sensitive. We believe this is a reasonable—but not the only or "best"—policy choice because it matches the requirements of many automotive telematics applications and corresponds to the legal concept of a *reasonable expectation of privacy*.[7] In this concept, individuals can claim a right to privacy only when they have a reasonable expectation of privacy—that is, the information is not easily visible to the public.

Specifically, we generated a sensitivity map by measuring the population density in each 10m × 10m segment of our evaluation area. For a given area *a*, we define the population density *pd*, which measures the fraction of a population that has visited the area in a given time interval *t*:

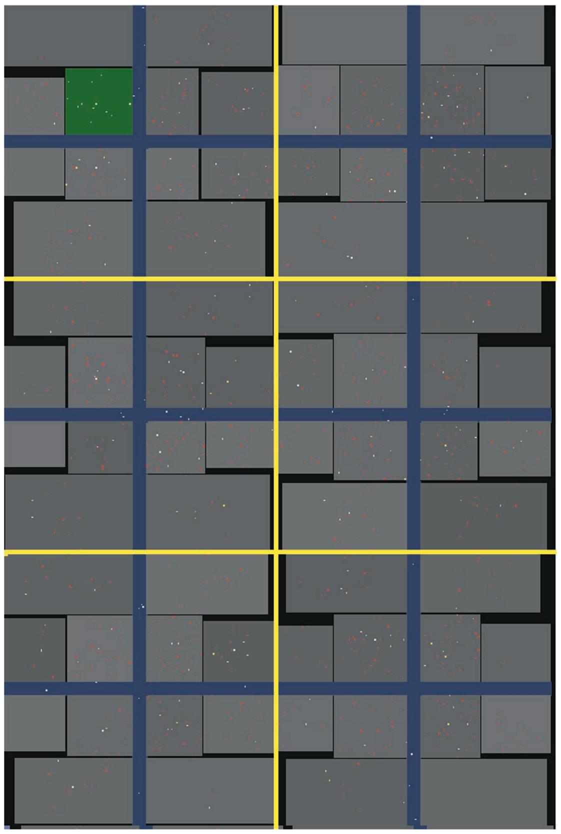


Figure 3. A map of the city model used to generate movement traces for the experimental evaluation. The rectangles show the locations of multistory buildings, dark lines indicate streets, and the yellow lines show the partitions we used to evaluate the *k*-area algorithm.

$$pd_t(a) = \frac{v_{a,t}}{p}$$

where $v_{a,t}$ is the number of visitors to area *a* in time interval *t* out of the total population *p*.

Comparing Figure 4 to the map in Figure 2 shows that the algorithm can clearly distinguish public streets from building areas. Although we initially expected the algorithm to also identify the park area, the result is an artifact of the mobility model. The City Simulator tool assigns the same entry and exit probabilities to the park as to buildings.

Adversary model

From the simulation data, we extracted movement traces of 73 individuals who entered and left sensitive areas. The traces contain position updates for each individual in 30-second intervals. To measure location privacy, we devised an automated adversary that guesses an individual's position and the building (sensitive area) he or she entered, even when such movements are hidden. We assume that the adversary has

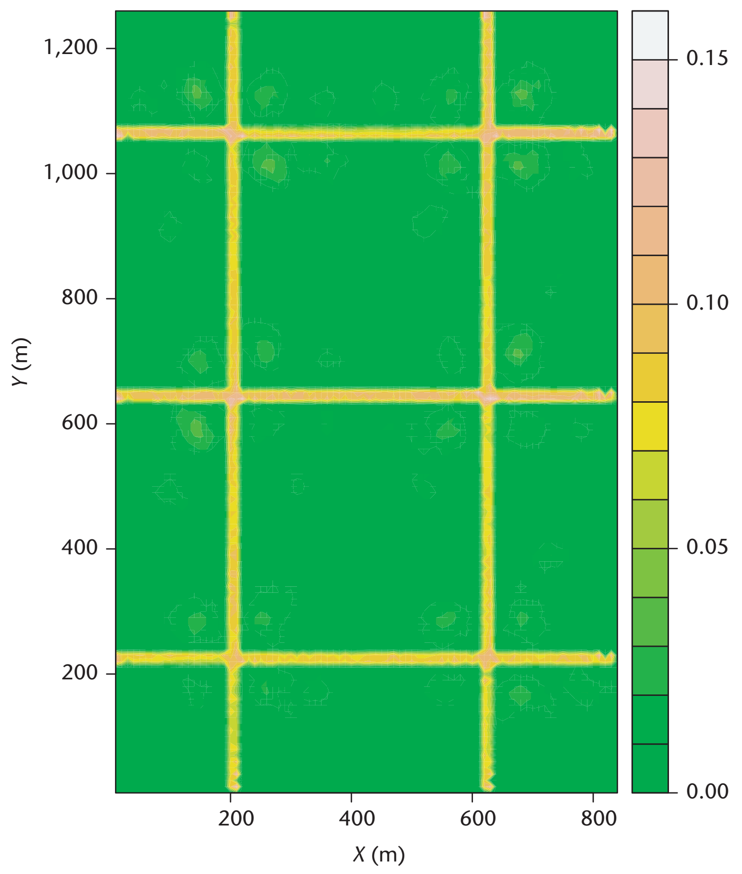


Figure 4. The sensitivity map. It classifies location into sensitive and insensitive. This particular map shows the population density at different locations, where values less than the threshold 0.05 are considered sensitive in our experiment.

access to the location updates received by a service provider. Thus, the adversary can watch a user's movements in insensitive areas by keeping track of the location updates. However, the disclosure-control algorithms can block location updates, causing the user to sometimes disappear from the adversary's view. In this case, the adversary estimates the user's position by linearly interpolating the first location after an individual appears and the last point before their disappearance. The adversary also estimates which sensitive area the user visited by determining the sensitive area closest to the midpoint of the interpolation. While we could devise much more sophisticated adversaries, we believe this one provides a reasonable first approximation to evaluate this problem, assuming that the adversary has no additional information about the monitored subject (such as preferences for certain locations).

To measure the degree of privacy in sensitive areas, we calculated the percentage of sensitive areas in which the adversary could correctly place a hidden individual.

Experiment results

Figure 5 shows the sensitivity map overlaid with movement traces. The black clusters show individuals' paths in sensitive areas. The red lines depict the adversary's estimate, using the base algorithm, during these invisible phases. A visual test reveals that the adversary's estimates are in close proximity to the actual paths. Thus, an adversary could, in most cases, guess which sensitive area (in our case, which building) an individual entered.

Figure 6 quantifies this observation and compares privacy protection with location availability in insensitive areas for the different algorithms. We configured the bounded-rate algorithm with one-minute, five-minute, and 15-minute intervals and selected example partitions for the *k*-area algorithm with at least four and 12 houses. (We chose the sizes for experimental convenience.) The blue bars show the percentage of sensitive areas in which the adversary correctly placed an individual. The base algorithm and one-minute rate offer virtually no protection. The other algorithms yielded comparable results between 20 and 45 percent, but they differ dramatically in the number of location updates revealed in insensitive areas (purple bars). The *k*-area algorithms only blocked less than 15 percent of location updates in public areas, whereas the bounded-rate algorithm withheld between 50 and 75 percent.

The *k*-area algorithm can concentrate the blocked location updates in areas where individuals actually visited sensitive places. Our first experiment's results show that the *k*-area algorithm is the most effective in terms of protecting sensitive locations while not diminishing location accuracy in insensitive areas; the bounded-rate algorithm reduces overall accuracy because it suppresses more location updates in sensitive areas. The trade-off, however, is that the *k*-area algorithm delays location update transmission in sensitive areas until the user leaves the zone.

Discussion

In our experiment, we assumed that an adversary had no significant a priori knowledge about an individual's relationship to the buildings in a visited area. In other words, from an adversary's perspective, a user is equally likely to enter each building. If it is common knowledge that an individual regularly visits a specific building and is highly unlikely to go into the adjacent ones, then the algorithms offer little protection. This vulnerability is most apparent for locations such as an individual's home and work, which can often be easily determined.

However, assuming no a priori knowledge, our automated adversary cannot distinguish between a correctly identified area and an incorrect guess. The adversary might be able to estimate the overall identification rate; however, for each guess, it remains unknown whether this guess was correct. For example, the 12 area results indicate that the adversary can claim only that an individual visited the identified area with 20-percent probability.

The percentage of vulnerable areas is higher than one-quarter or one-twelfth for the four-area and 12-area algorithms, respectively. This indicates that, in our mobility model, a positive correlation exists between the visited building and the entrance and exit streets the individual used. We expect the *k*-area algorithm to offer a relatively constant degree of privacy protection across scenarios with different densities of sensitive areas. In less dense areas, the partition's size would expand to accommodate the same number of sensitive areas. The bounded-rate algorithm, however, does not take topological information into account, which—given a fixed rate—would yield more privacy breaches in scenarios with fewer sensitive areas.

Similarly, changes in an individual's walking or driving speed should not influence the *k*-area algorithm's degree of privacy protection. Service providers can benefit from users with higher speeds because the intervals between crossing partition boundaries will be shorter, so path information will arrive more quickly. When using the bounded-rate algorithm, protection increases with movement speed. A faster user can, in the same time interval, leave the sensitive location further behind. Location accuracy in public areas, however, is negatively affected by higher speed.

Finally, the current algorithms seek to protect only the identity of the visited location. An adversary can still derive other information, such as the duration of stay in an area or the frequency of visits to a larger area. By developing privacy-controlling algorithms and experimenting on their effectiveness, we have shown that protecting an individual's location in sensitive areas is difficult—whether these areas are established via policies or defaults—because an adversary could infer approximate positions from prior and future location updates and trajectories.

In future work, we hope to evaluate such algorithms in real world prototypes that collect outdoor movement traces through GPS, and perhaps indoor traces through wireless LAN positioning mechanisms. This will also require developing and validating tools for automatically determining distinct areas, such as buildings, in a larger space, and for partitioning these areas as necessary for the *k*-area algorithm. Eventually, we hope that this work will lead to effective mechanisms that support useful LBSs, while maintaining privacy. ☐

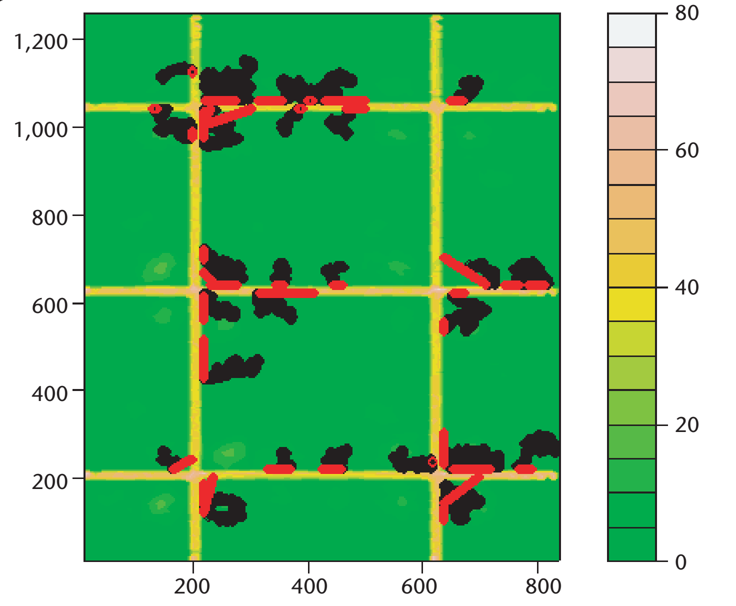


Figure 5. A plot of sensitive movements (black) and the adversary's estimates (red) based on the revealed data. This illustrates that, with only weak privacy protection, the adversary's estimates are generally close enough to the real movements to infer the sensitive area that a user entered.

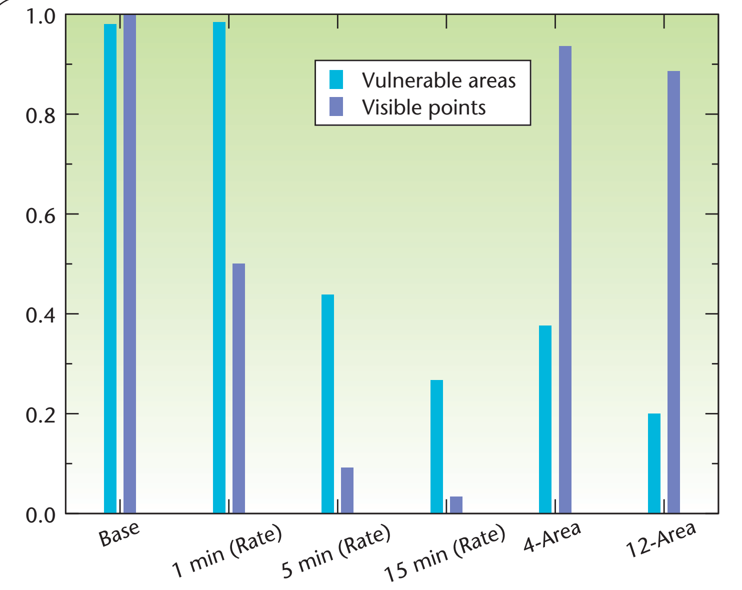


Figure 6. Privacy protection with location availability. The percentage of sensitive areas that a simple adversary could infer (vulnerable areas) and the percentage of visible points in insensitive areas (at a mean velocity of 0.1 m/s).

References

1. J. Reed et al., "An Overview of the Challenges and Progress in Meeting the E-911 Requirement for Location Service," *IEEE Personal Comm.*, vol. 5, no. 3, 1998, pp. 30–37.
2. J. Warrior, E. McHenry, and K. McGee, "They Know Where You Are," *IEEE Spectrum*, vol. 40 no. 7, 2003, pp. 20–25.

Related work in location tracking

Privacy policies serve as a common approach to addressing privacy challenges. Typically, a service provider describes data-handling practices through such policies; users can then decide to what extent they trust the service provider. Advanced systems let users specify a set of rules or preferences that can automatically decide whether to release data.[1,2] Although such systems let users specify location constraints—exclusion zones, for instance—it is unclear how users can author such constraints and whether they can easily comprehend more complex constraints.

Similarly, the IETF Geolocation & Privacy Working Group[3] is addressing privacy and security issues regarding the transfer of high-resolution cell phone location information to external services and its subsequent storage at location servers. It concentrates on the design of protocols and interfaces that enable devices to communicate their location in a confidential and integrity-preserving manner to a location server. The location server can reduce the data's resolution or transform it to different data formats, which external services can then access if the user's privacy policy permits.

Strictly anonymous location information provides a solution for sporadic point queries when very high accuracy is not required, as Marco Gruteser and Dirk Grunwald have shown.[4] In this earlier work, a system automatically coarsens location granularity to ensure that correlation of position information from different sources cannot identify individuals. For continuous queries, Alastair Beresford and Frank Stajano introduced the concept of mix zones.[5] At least in office environments, however, a sufficient user population is not always available. Pierangela Samarati and Latanya Sweeney[6,7] have developed generalization and suppression techniques for values of database tables that safeguard individuals' anonymity. While our *k*-area algorithm uses a similar principle—*k* indistinguishable entities—our goal and algorithm differ in that we aim to hide the visited location rather than its identity.

Location privacy has also been studied in position-sensor systems. The Cricket system[8] places location sensors on the mobile device, as opposed to the building infrastructure. Thus, location information is not disclosed during the position-determination process; the data subject can choose the parties to which the information should be transmitted. Asim Smailagic and David Kogan describe a similar approach for a wireless LAN-based location system.[9] However, these solutions do not address how algorithms can automatically decide which positions to reveal to location-tracking services.

References

1. G. Myles, A. Friday, and N. Davies, "Preserving Privacy in Environments with Location-Based Applications," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003, pp. 56–64.
2. E. Snekkenes, "Concepts for Personal Location Privacy Policies," *Proc. 3rd ACM Conf. Electronic Commerce*, ACM Press, 2001, pp. 48–57.
3. J. Cuellar, J. Morris, and D. Mulligan, "IETF Geopriv Requirements," 2002, www.ietf.org/html.charters/geoprivcharter.html.
4. M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," *Proc. 1st Int'l Conf. Mobile Systems, Applications, and Services*, Usenix Press, 2003, pp. 31–42.
5. A. Beresford and F. Stajano, "Location Privacy In Pervasive Computing," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003, pp. 46–55.
6. P. Samarati and L. Sweeney, *Protecting Privacy when Disclosing Information: k-anonymity and its Enforcement Through Generalization and Suppression*, tech. report SRI-CSL-98-04, SRI Int'l Computer Science Laboratory, 1998.
7. L. Sweeney, "Achieving k-Anonymity Privacy Protection Using Generalization and Suppression," *Int'l J. Uncertainty, Fuzziness, and Knowledge-Based Systems*, vol. 10, no. 5, 2002, pp. 571–588.
8. N.B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," *Proc. 6th Int'l Conf. Mobile Computing and Networking*, ACM Press, 2000, pp. 32–43.
9. A. Smailagic and D. Kogan, "Location Sensing and Privacy in a Context-Aware Computing Environment," *IEEE Wireless Comm.*, vol. 9, Oct. 2002, pp. 10–17.

3. L. Barkhuus and A. Dey, "Location-based Services for Mobile Telephony: A Study of Users' Privacy Concerns," *Proc. 9th Int'l Conf. Human-Computer Interaction* (INTERACT), ACM Press, 2003, pp. 709-712.
4. G. Myles, A. Friday, and N. Davies, "Preserving Privacy in Environments with Location-Based Applications," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003, pp. 56–64.
5. K. Bohrer et al., "Individualized Privacy Policy Based Access Control," *Proc. 6th Int'l Conf. Electronic Commerce Research* (ICECR-6), 2003, to appear.
6. J. Kaufman, J. Myllymaki, and J. Jackson, "City Simulator: A Spatial Data Generator," July 2003, www.alphaworks.ibm.com/tech/citysimulator/.
7 E. Alderman and C. Kennedy, *The Right to Privacy*, Diane Publishing, 1995.

***Marco Gruteser** is currently a PhD candidate in computer science, advised by Dirk Grunwald at the University of Colorado at Boulder. His research interests include privacy-enhancing technologies, context-aware applications, and wireless networks. He received an MS in Computer Science from the University of Colorado at Boulder and completed a Vordiplom at the Technical University Darmstadt, Germany. He is a student member of the ACM. Contact him at gruteser@cs.colorado.edu.*

***Xuan Liu** is a research staff member at IBM T.J. Watson Research Center. She received her PhD in computer science from University of Minnesota, Minneapolis. Her current research interests include privacy technologies, XML technologies, location-based services, data mining, and spatial databases. Her research work has been published in many prestigious journals and conferences, and has led to five patent-pending inventions. She has served as a program committee member for many international conferences. She is a member of the IEEE, IEEE Computer Society, and the ACM. Contact her at xuanliu@us.ibm.com.*