Architectural Considerations for Scalable, Secure, Mobile Computing with Location Information

Mike Spreitzer Marvin Theimer
Xerox Palo Alto Research Center

Abstract

To take full advantage of the promise of mobile/ubiquitous computing requires the availability and use of location information about the various physical objects and persons in an environment. At the same time, indiscriminant use of location information for people can result in an invasion of privacy and provides the potential for abuse by third parties. In this article we describe a suite of useful location-based applications and discuss architectural considerations for supporting them in a scalable, secure fashion.

1 Introduction

There are a variety of reasons why location information is important to mobile computer users[14, 13, 12]:

- Wireless connectivity (including cellular phone service) requires knowledge of where you are in order to route communications to your current location.
- People frequently *want* to know their own location for a variety of everyday purposes, such as finding one's way through an unfamiliar city or building.
- People, in general, may want to know about the presence of "nearby" services, such as the nearest printer of a particular sort, or the nearby availability of certain kinds of stores or restaurants. Computing "nearness" functions requires knowledge of both the peoples' locations as well as the locations of the various services they might be interested in.
- People may wish to deal with a group of other people based on their location; for example, "all people within a given room".
- Finally, various facilities/services that are situated in the environment may need to be aware of what people and/or things are nearby.

While keeping track of and making available location information about "public" things - such as printers and restaurants - is both acceptable and desirable, the same does not necessarily apply to people. We consider unrestricted access to a person's location data to be an unacceptable invasion of privacy. We want to give people control over the revelation of their location (and other private information). Of course, the more one person restricts the revelation of personal information, the less other people can use that information.

In this paper we consider systems consisting of a suite of location-based applications and some supporting infrastructure. We explore the trade-offs between the privacy, security, and functionality that a system might provide, and the costs of implementing such systems. We discuss security in terms of two things: a set of security goals, and a security model. The security goals form a statement of what security guarantees are made to a user, and of what kinds of control a user has over disclosure of his private information. The security model is a technical statement about what an attacker can and cannot do.

Because one security model and set of goals do not fit all situations, we consider several combinations. We start with very simple security goals, and a very strong (i.e., restrictive on attackers) security model. We then strengthen the security goals, and weaken the security model, in a series of steps. The system architecture is responsible for implementing the application suite and security goals under the assumptions of the security model. Along the way we discuss how the semantics of some of our applications need to be reformulated to remain consistent with the security goals.

This paper consists of several "homogenous" designs, each appropriate for one security model and one set of security goals. However, in a large-scale mobile computing environment, the security model and goals appropriate will vary from user to user and time to time. What is ultimately needed is one "heterogenous" system, wherein individual users can vary their security model and goals from time to time. In a separate paper we present some ideas about how to construct such a heterogenous system, as well as our experiences with using a relatively homogenous prototype we have

*This research was supported in part by the Advanced Research Projects Agency under contract DABT63-91-C-0027.

built[9].

2 A Suite of Applications

The set of applications we will consider are the following:

Ubiquitous Message Delivery A message may be submitted for "ubiquitous" delivery to one or more recipients. Ubiquitous delivery can use any terminal near the recipient, if message delivery through that terminal to that recipient at that time is "appropriate". Appropriateness is a policy profile that is specified by each human user. It can depend on a variety of things, including the importance of a message, the identity of the sender, the contents of the message, and whether the message is labeled "private" or with some other important keyword. It can also depend on the number of other people currently near the user and what kinds of terminals are available for delivery (e.g. small, private pager versus large, electronic whiteboard). The submitter does not need to know where the recipient is; the system's knowledge of the recipient's location is used.

Scoreboard This application is an information-oriented "screen saver". When a display is not being used for anything else, it displays information of general interest (like some stadium scoreboards), but tailored to the interests of the people nearby.

Find This application finds the nearest resource or person matching a given specification, such as "is a UNIX wizard".

Note distribution Notes can be distributed to everybody in a meeting, as determined by their presence in the meeting room. If desired, only people actually at the meeting are allowed to receive the notes.

Activity-based Information Retrieval (AIR) A log is kept of everything the system knows that happens to you: where you are, who is there with you, message deliveries, and so on. Queries of this log can help you remember things by association with logged events[5, 4].

Locations This application displays the current locations of a designated list of people or the names of all people at a designated list of locations.

Visitor guidance A person is guided along an unfamiliar route.

Responsive environment A "smart building"'can optimize its energy usage by use of knowledge about which rooms are occupied. It can also control the environmental settings of each room according to the preferences of the people in them.

3 A Baseline Architecture

We begin our exploration with a baseline architecture for scalable, mobile computing with location information that implements no security goals whatsoever. Because there are no security goals to test achievement of, we don't need to specify a security model.

A baseline architecture that can support our applications suite has the following additional elements beyond those found in a traditional computing sytem:

Wireless communications network. This allows mobile devices to link into the rest of the system.

Sources of location information. Examples of kinds of sources of location information are: (1) a Global Positioning System[1], (2) infra-red-based active badges[13, 12], (3) wireless nano-cell communications activity[14], (4) device input activity from various computers[11], and (5) explicit indications from users.

A Location Service. This service receives location data about the locations and movements of users and devices. It answers queries about where entities are and what entities are at a given location. It supports historical versions of these queries as well.

Terminal Agents. Each I/O terminal has an associated agent, that provides access to it through a uniform interface and manages its resources. Terminal agents export a description of their device characteristics and support the notion of "background" display behavior, as needed by applications like *Scoreboard.*

User Agents. Each user has an associated agent, with a variety of responsibilities. It is the user's agent that collects the location information about the user from various sources and synthesizes it into one opinion about where the user is. This agent then reports that opinion to the Location Service, and also uses it directly to answer queries from interested parties. User agents represent well-known entities of whom various user-specific questions can be asked.

The remainder of this section describes how one can implement our applications on top of these elements:

Ubiquitous Message Delivery A sender submits a message for ubiquitous delivery to the User Agents for the recipients of the message. A user's agent keeps track of what terminals are near the user. Whenever there are sufficiently capable terminals near the

user and the user's delivery policy allows it, the User Agent asks the Agent of one of those terminals to present the message to the user. The User Agent considers the message delivered when a Terminal Agent reports that the user has read the message. If the user moves away from a terminal before that terminal's agent reports a decision from the user, the User Agent tells the old Terminal Agent to abort the presentation, and then continues attempting ubiquitous delivery at the new location.

Scoreboard Whenever a user is near a terminal displaying Scoreboard output, the User Agent tells the Scoreboard application the user's identity and Scoreboard preferences. When a user leaves the vicinity of a terminal, the User Agent tells the Scoreboard that the user has departed. A Scoreboard keeps track of a set of (user, preferences) tuples, and combines the preferences to prioritize the information to display whenever the terminal it is bound to is being used for nothing else.

Find Starting from a given location and working outward, the agents for the users at a given location are queried to see if they meet the required specification. The first one to respond in the positive is reported.

An alternative implementation, suitable for less commonly matched specifications, would be to query the Location Service for the locations of a specific set of users, reporting the one nearest the given location. This requires the seeker to know a-priori the set of users to seek, or have access to some kind of database that would reveal the set of users to seek.

Note Distribution A user asks for certain electronic materials to be distributed to the people in the same room. The Location Service is queried for that set of users, and their agents are given the materials.

AIR The Location Service and the User Agents cooperate to record the log. The Location Service is implemented in a distributed fashion; each server is responsible for recording information about locations in some region. This is consistent with queries being based on locations. Each User Agent records where its user has been and when, and other kinds of user-related events.

A User Agent does not need to store a history of other people being near its user. This information can be reconstructed, if and when needed, by joining the relation between the user and where he's been with the Location Service's relation between locations and people.

Locations The locations of people in a given region can be determined by querying the Location Service. The locations of a given set of people can be determined by asking their User Agents where they are.

Visitor Guidance The agent of the user to be guided is given - or can determine - the topology of locations near the most direct paths between the user's starting point and desired destination. The User Agent continuously presents guidance to the user, on fixed terminals near, and/or portables carried by, the user. The User Agent finds fixed terminal agents appropriate to use by querying the Location Service for them at the user's current location.

Responsive Environment The "smart building" determines occupancy of its areas of concern by continuously querying the Location Service. It determines what environmental settings to use in any given room by querying the User Agents of the people in that room.

4 Modifications to Support Privacy

In this section we will introduce a variety of modifications to our architecture in support of the provision of privacy. Our goal is to support the following notion of "privacy":

- Mobile computing can take place in a fashion that keeps location information inherently secret.
- Users have control over the revelation of information about their current and past locations.

That is, we define privacy to mean that information about a person's location remains known only to that person unless he explicitly hands it out to someone else.

Privacy is not easy to provide in an absolute sense. For instance, traffic analysis of even encrypted communications packets can be very revealing of location. Therefore the remainder of this paper will focus on a progression of security models. We will start with the assumption of a "friendly" environment that makes the provision of privacy easy to achieve and will progressively relax the assumptions made for the friendly environment and modify the location architecture to deal with these relaxations. The resulting security models and location architectures provide privacy in progressively more "hostile" environments, typically at the expense of more and more overhead and complexity.

Not all systems must operate in the same environment: some environments are inherently friendly and can be made physically secure against intrusion by hostile outsiders while others are not. By considering a spectrum of different architectures, we illustrate choices concerning how much "privacy enforcement"

must be paid for. The ultimate goals is creation of "heterogeneous" systems in which different domains provide different levels of privacy protection. Such hetergeneous systems offer the opportunity for "moded" operation, in which location-based applications may be freely usable in friendly environments but may have to be eschewed in others.

4.1 User Control Over Location Information

We shall start out with a more explicit statement of our privacy goal:

1. Mobile computing can take place in a fashion that keeps location information inherently secret.
2. Users can control the revelation of information about their identity and current and past locations as a function of the identity and location of the requestor of that information. The function may also depend on the value of state variables that are available to a user's user agent.
3. Requestors may choose to not reveal their own identity and location. In this case a user is free to refuse the request.
4. In all cases, identity information may optionally be replaced by a list of "group" identities. We define a group to be a publically known entity representing a set of persons. Knowledge of the membership of a group may or may not be publically known.
5. A user can reveal that *someone* is present at a given location without revealing *who*.

This more explicit statement of our goal allows us to specify the exact set of parameters that must be made available for controlling the revelation of information. It also explicitly introduces the notion that people may belong to various groups and might wish to reveal their membership in a group without necessarily revealing their individual identities. The last requirement is necessary to allow users to be counted by location, as is needed by applications like the responsive environment.

Our definition of a "friendly" environment is one that is safe from hostile intruders/monitors and where the (legitimate) members of the environment behave in an "honest" fashion. This yields the following security model:

1. Programs can be run "securely". That is, neither executable binaries nor running images can be read or modified.
2. Communications cannot be monitored in any way. Furthermore, recipients can't tell who sent a message and senders can provide a way for a recipient to reply without being able to discover the sender's identity[1].
3. People can write and run programs that use our architecture, but they will not lie.
4. People do not perform traffic analysis, such as counting the number of persons registered at every location in the Location Service and watching how those numbers change in order to infer people's identity and location.

An example of a friendly environment might be a physically secure office environment in which there is mutual trust between employees and management.

There are a number of implications that follow from our goal. One is that users may now operate in a "moded" fashion. That is, they may choose not to make themselves visible to other parties. As a consequence, the Location Service can no longer be assumed to contain complete information and applications must understand this. For example, the AIR application can no longer assume that it has been able to record every passing person. Perhaps more importantly, AIR queries must be made with the understanding that its version of history may seem inconsistent with the memories of the user - who may remember seeing someone who chose not to reveal their location electronically.

Another implication is that applications like *Locations* and *Find* must have the definition of their functionality modified. In particular, their function must be revised to state that only information about persons willing to reveal themselves in the appropriate fashion can be obtained.

There are two ways in which we can achieve our goal given our stated security model: add access controls to the Location Service or allow people to *not* register their current location in the Location Service and have only their user agents answer queries concerning their location.

In the former case we must provide a sufficiently powerful access control language to allow people to express the controls they desire for a registration. The latter approach requires that User agents be able to demand the identity and location of the person requesting location information.

The latter approach must also still support applications that wish to find persons by means of queries by location. This can be done with "anonymous" registrations in the Location Service. Anonymous registrations provide a communications handle back to a user

[1]Note that one can't limit revelation of private identity/location information if that information is publically visible in all one's communications.

agent without revealing the identity of the user agent. The registration allows applications to become aware of the presence of "someone" at a given location. However, in order to find out who that someone is, they must use the anonymous communications handle to ask the associated user agent for identity information.

One advantage of having user agents apply the access controls instead of the Location Service is that each user agent can choose to be arbitrarily simple or complex concerning its decisions to answer any given query without affecting anyone else. Also, if computation of access control depends on state that is specific to a user — such as the contents of an electronic calendar — then it will be more cumbersome to maintain a duplicate copy of that state in the Location Service.

4.2 Controlled Exchange of Information

The previous section's goals require that a requestor reveal information about themselves in order to make a request, whether or not they receive any information in return. We consider supporting the following modification of our basic goal:

> Allow revelation of identity/location information in a "neither or both" fashion. That is, allow a requestor of identity/location information to specify that their own identity and/or location should be revealed to the "owner" of the requested information if-and-only-if their query is answered.

This modification allows both parties of a location or identity query to maintain control over the information they reveal.

This goal is almost trivial to achieve in the Location Service-centric architecture. The only change needed is to allow the Location Service to notify the owner of a registration whenever that registration is returned as part of a query response.

The user agent-centric architecture is at a disadvantage: it must recreate the trusted, third party nature of the Location Service. An "escrow service" can compute user agents' access control functions for them and then decide to reveal information either to both parties or to neither party of a query. The specific steps to follow are:

1. The requesting client sends the user agent a message specifying the operation he is interested in and sends the escrow service his location and identity, as well as a specification of what information he is willing to reveal (location, identity, both, or neither).
2. The user agent sends a specification for the access control function he wishes to apply to the escrow server.
3. The escrow server executes the user agent's access control specification using the parameters received from the client.
4. If the access control function indicates that the client's request should be answered then the escrow server sends a message to the user agent containing the revealed information about the client and sends a message to client with the revealed location and/or identity of the user agent.

It might seem that we are merely duplicating the Location Service-centric architecture with this design. However, the ability to separate the functionality of the escrow service from that of the Location Service will be of use later on when we relax the assumption that all Location Services can be trusted.

Note that because the access control function that the User Agent sends to the escrow server is applied to a specific request, it need be parameterized only by the particulars of the request. The User Agent does not need to send the user's full access control function, but only the residual function that remains after partially evaluating the full function with respect to everything other than the particulars of the request (e.g., state variables private to the User Agent). This is another way in which an escrow service differs from the Location Service.

4.3 Secret Groups

We can get rid of the need for an escrow service if we allow ourselves a more restrictive privacy goal. In particular, assume that we replace requirements 2 and 3 of the privacy goal specified in Section 4.1 as follows:

- Assume that users are willing to control revelation of information about their identity and location based only on the set of group memberships a requestor is willing to admit membership of and on the value of state variables that are available to a user's user agent.
- Assume that requestors are willing to make known a selected list of their group memberships whether or not their query is answered.

A scenario fitting this privacy goal is one in which users may create "secret" groups, whose membership is known only to members. Users can control who can "recognize" them by handing out memberships in the groups they have created. User agents respond to queries with identity information that consists of the set of group memberships (secret and/or public) that

they are currently willing to admit to. Requestors of identity/location information include only those group memberships (again, secret and/or public) that they are currently willing to admit to.

4.4 Working in the Presence of Lies

So far we have discussed goal changes. Our first model change is to assume that people and their programs may lie.

The implications of this model change are the following:

- The identities claimed by people and the programs they run may be incorrect. Both people and programs may try to masquerade as others.
- Location information may be incorrect, both as returned from a user agent and as stored in a Location Service.

4.4.1 Traditional Authentication

To prevent people, programs, and services from lying about who they are one can use authentication[8]. We thus introduce into our location architecture the requirement that a standard authentication service, such as Kerberos[10], or a public key-based cryptography system[3], be available. Applications that wish to query a user agent for location or identity information will have to be prepared to present the appropriate authentication credentials as a parameter to each request. Similarly, when a query returns an identity, it will have to include appropriate credentials proving that it is authentic.

Note that if the Location Service is providing the information then authentication might occur at registration time, making the use of query results cheaper.

A somewhat more complicated form of authentication is needed to ensure that programs are dealing with trustworthy instances of various services. An example is when a User Agent doing Ubiquitous Message Delivery must decide whether to pass a message to a Terminal Agent. In a large-scale mobile setting, a User Agent will encounter servers of which the Agent has no prior knowledge (e.g., about how trustworthy that server is) — and yet the User Agent must decide whether (or how) to use such servers. A way to deal with such situations is to turn to third-party rating services (examples from the realm of human commercial activity are Better Business Bureaus and magazines like "Consumer Reports").

4.4.2 Authenticating Group Membership

If people and programs can lie then we must ensure that they can't falsely claim membership in groups. One approach is to employ traditional authentication techniques, with each group playing the role of a "principal": each member of the group knows the secret needed to be authenticated as "the principal" (i.e., a member of the group).

Another approach is to implement a claim of membership in a group by sending a "verifier" — something (such as a specified tuple of values encoded with a key specific to the group) that only a member of the group could construct and that the receiver can verify is properly constructed. Of course, the tuple of values needs to be specified in such a way that prevents replay attacks and minimizes the ability of an attacker to guess the group secret[8].

There may be significant differences in the message traffic required to implement these approaches, depending on the authentication technique, cryptography, and verifier construction rules used. For example, if the second approach is taken, using public-key cryptography to construct a verifier by encrypting a well-synchronized global time[6] with the group's private key, a request receiver who's got that group's public key cached can test the verifier with no additional message traffic. As a contrasting example, if the first approach is taken, using Kerberos authentication, additional message traffic with the Kerberos authentication server is required.

The two approaches outlined above involve the members of a group sharing a secret, which scales badly with group size. There are other approaches that avoid a shared secret, at the cost of extending other kinds of trust (e.g., using a group-membership authentication service).

4.4.3 Authenticating Location

In this section we assume that people and programs can lie about their location. However, we assume that the location infrastructure (sensors, location beacons, etc.) does not lie.

Authenticating that someone or something is at a given location in space and time requires the following:

- The object whose presence is to be authenticated must possess some distinguishing feature that only it has.
- There must be a sensor that can reliably distinguish the presence of the feature at a specified location.
- The sensor's output must be deemed trustworthy by the clients who use it and there must be a way

to reliably distinguish its output from anyone else's.

Unfortunately the observation systems available today may not be able to see the things we really care about. For example, a trusted sensor may detect the presence of wireless communications devices or of active badges. But such devices can be removed from a person's body; this implies that the sightings provided by the sensor system can only authenticate the locations of the devices and not of the persons associated with them.

Consequently we cannot guarantee that applications such as *Locations* will show correct information about the locations of various people. We also cannot guarantee that location-dependent access controls for queries will achieve their desired goals.

In fact, we can neither achieve our stated privacy goal nor implement all of our desired applications. We must modify our goals if we are to obtain a workable system design. In particular, we must retreat to assuming that users can only use the identity and not both the identity and location of a requestor when deciding whether or not to honor a request. We must also modify the specification of some of our applications:

- *Locations* may display incorrect information.
- *Find* must treat their results as hints rather than facts.
- AIR cannot assume that the information it is recording is correct except when it has other means of validating it. In particular, it may have to assume that data from anyone other than trusted, authenticated persons may be incorrect and should be treated with suspicion.
- The responsive environment must view the data it has on peoples' locations as hints and not truth.

Note that applications such as ubiquitous message delivery do not rely on authenticated location information. All the location information they need comes directly from the (still trusted) location infrastructure facilities.

An authenticated sighting of a device is still useful:

- The electronic equivalent of a key for applications such as visitor guidance can still be provided.
- One can prevent someone from monitoring many locations at once by claiming to be at those locations simultaneously (or in quick succession).
- Spoofing of things like a responsive environment system are made harder because an actual physical device must be present for the system to believe that someone is in a room.

Applications such as note distribution for a private meeting represent an interesting complication. In order to implement secure note distribution based on authenticated location information one needs to be able to detect "bugs": devices left hidden at various locations to mimic the presence of their owners.

One can imagine various techniques by which bugs could be flushed out to reveal their presence prematurely. Some involve having some external means of signalling to the "real" people in a room when they should perform an action that a bug won't know to mimic. A more automatic method involves a "door monitor": as people enter a room their identification devices obtain authenticated location values indicating that they were seen entering the room. (An authenticated location from an adjacent room would also suffice.) Any device whose door location was from a time before the start of the meeting would be considered to be a bug. This method works only for bugs planted on the room; bugs planted on the meeting participants (or clothing or equipment they carry) could get through.

4.5 Insecure Communications Facilities

We next consider relaxing the assumption that our communications facilities are secure. The implications of this change are that we must now worry about unwanted parties monitoring and altering the communications traffic.

In this section we will concern ourselves only with the question of what information can be deduced from the examination of individual communications packets. We will deal with traffic analysis questions - where one tries to deduce information from the *patterns* of multiple packets - in a later section.

We consider first the case of someone who can monitor our communications. This includes both the wired, traditional networks used and the wireless communication cells used.

We can use standard encryption techniques to prevent unwanted examination of the data being communicated[3, 7]. However, we cannot encrypt the addressing information. Consequently a network monitor (or even a legitimate receiver of a message) might be able to deduce location/identity information by examining the source and destination addresses involved: the source may identify a wireless communications cell and the destination may reveal the identity of a particular user (or vice-versa).

To prevent information from being "leaked" in this fashion, we effectively need an intermediate laundering agent to allow source and destination addresses to be decoupled from each other[2]. Thus, our location architecture must now include the notion of such a trusted laundering service, which we will define as

follows (other, similar definitions are also possible):

- Assume the existence of a trusted server that employs some form of cryptography, such as public key cryptography, to allow arbitrary clients to send it messages in a secure, encrypted fashion[7].
- Messages sent to the server must be of the following form: they contain a network address to which the remainder of the message will be forwarded.
- The server takes any messages sent to it, decrypts them, and forwards them on to the destinations specified inside them[2].

We also need to avoid using well-known addresses or identifiers to designate mobile devices. Otherwise, a monitor can simply look up the addresses he sees in order to determine their identity. The same argument applies to stationary entities. For example, the return address provided by a client querying a user agent must be sufficiently anonymous that the user agent (or a traffic monitor) can't deduce the client's identity from it.

Suppose now that communications data can also be injected or modified. Because we are already using encryption to combat monitoring attacks, we do not need to worry about modified data. While we *do* need to worry about communications being replayed (for example, to falsely register someone at a previously occupied location), there are well-known techniques for preventing this[8]. The worst thing an attacker can do is cause denials of service by disrupting the communications essential to the service.

4.6 Untrusted Location Services

In this section we will relax the assumption of our security model that programs can be run securely. More specifically, we will assume that not all components of the location infrastructure can be trusted. (Remember that we already allow user programs to lie, cheat, and steal.)

The most natural setting in which such a situation can arise is that of multiple administrative domains. One can assume, almost by definition, that the servers and services of a foreign administrative domain should be trusted less than those of one's own administrative domain. If we wish to provide a location infrastructure that can scale to the world then it will have to span multiple administrative domains, and hence will have to be able to deal with parts that are potentially untrustworthy.

We start by allowing corrupted Location Services[3]. The primary consequence of this change is that we can, in general, no longer rely on Location Services to be trusted repositories of private information.

Unfortunately this means that the Location Service-centric architecture we have been considering up until now should be discarded! Up until now the tradeoffs between our User Agent-centric and Location Service-centric architectures have been relatively unimportant. But when we make the assumption that not all Location Servers can be trusted, as might be the case in a system spanning multiple administrative domains, only the User Agent-centric architecture stays viable. Thus we believe that only a User Agent-centric architecture is practically scalable to large systems.

Suppose next that some sources of location data may be corrupt. This has two implications:

- User agents can't trust all location data they receive from their sources and hence must judge it accordingly.
- Noone can trust authenticated location information unless they have some external means of verifying the trustworthiness of the sensors involved.

One location infrastructure service whose corruption we do *not* have to worry about is the escrow service. This is because nothing precludes there being multiple escrow servers and users are free to use whichever one they wish. The key to this argument is the assumption that there will be enough trusted third party escrow services available that every pair of potential clients will be able to agree on an acceptable one to use. Otherwise, clients will experience a denial of service.

A similar argument holds for dealing with trusted laundering services. Note also that one can "compose" laundering services in order to obtain communications routes that pass through several laundering servers. As a consequence, no single server will know the complete route.

BBBs, like escrow and laundry services, can have multiple instantiations and clients are free to use whichever ones they trust.

If the authentication technique used involves an authentication service (such as Kerberos), wherein authentication servers know principals' secrets then the authentication service must still be trusted. As with escrow, this trust bottleneck can be widened by deploying multiple, alternate authentication services. In

[2]Note that we are not yet assuming that traffic analysis can occur, so no efforts need be made to hide correlation between inbound and outbound messages.

[3]Note that Location Services make good candidates for break-in attempts since the payoff for success is much higher than it would be for breaking into a single user agent.

contrast, if a public key-based authentication technology is used then the system service required is a publishing service for public keys; providing several independent instances of these would enable a client to rely on one it trusts.

4.7 Preventing Traffic Analysis

So far, our security model includes an assumption that an attacker won't analyze patterns of activity in the communications and higher levels. Even when communications packets don't contain source or destination addresses that immediately identify the communicating parties, an attacker that can observe all the communications in the system and think carefully about the timing of the transmissions could greatly reduce his uncertainty about some private information. The same is true for monitoring the contents of the Location Service, even if it contains only a count of the number of people at any given location. Exactly how much information an attacker can derive from traffic analysis depends on many factors and an exploration of this topic is beyond the scope of this paper.

Two special cases are worth noting. An extremum is the case where the security model allows attackers to observe all traffic and perform arbitrary computations over it, and the security goals allow a user to hide his location at a given time from everybody or from nobody. An implementation of this extremum involves having the user disable transmissions from his mobiles. Broadcast information can be received, which means that some applications (such as ubiquitous message delivery) can still be implemented (perhaps with some modifications). In general, the observable activities of the system must follow a set of rules that don't take private information into account (e.g., transmissions follow a fixed schedule, regardless of the need to communicate; each packet is encrypted so that only the intended recipient can tell whether it contains filler or an interesting message). Thus, the broadcasts for a given user should be made in a set of cells whose selection does not depend on where the user actually is.

A less extreme design point has a similar security model and goals, but stipulates that the user is willing to risk revelation of coarse-grain (in time and space) location information. This is achieved by directing broadcasts to a slowly varying set of cells that cover a large area; the user can control the spatial granularity by specifying the area, and the temporal granularity by exercising control over when the set of cells may change.

4.8 Foiling Analog Eavsdropping Techniques

So far our security model has stated that an attacker has access to only the intended, "digital" aspects of the communications. But, as someone once said, "one man's noise is another man's signal." Depending on how much capability the attacker might have, a user might be required to either turn off his mobiles' transmitters, turn off his mobiles, or not carry any mobiles, in order to hide his location.

4.9 Why Heterogenous Systems

The security model a user is willing to adopt says something about (among other things) how much trust the user is willing to place in various pieces of infrastructure. Some administrative domains will be more trustworthy than others, both in an absolute sense and in the opinion of a particular user. The choice of security model also involves an estimation of how hard an attacker would be willing to try to violate the security as well as whether the user thinks the model will be broken infrequently enough that the user is willing to suffer the attendant limited privacy loss. For these reasons, heterogenous systems, with moded operation and multiple providers of each service, are preferred. Heterogenous systems allow users to take advantage of this diversity, enabling them to enjoy applications with good knowledge of location, and fine control over revelation of their private information, where possible, and to employ less functional applications, with coarse control, where necessary.

5 Conclusions

Our overall conclusion from this and related work is that one can build useful location-based systems that can safeguard people's privacy in many situations. However, an absolute guarantee of privacy is not possible because of two kinds of problems: we cannot, in general, provide authenticated location information for people (as compared to the detachable electronic devices they possess) and we cannot easily protect against traffic analysis attacks.

Ignoring this absolute guarantee of privacy, we have focused on how to provide various more limited forms of privacy under several different system trust scenarios. One important distinction we have focused on is the choice between an inherently decentralized, User Agent-centric architecture and an inherently centralized, Location Service-centric one. Each has various minor advantages and the tradeoffs between them

seem relatively unimportant until we make the assumption that not all Location Servers can be trusted — as might be the case in a system spanning multiple administrative domains. In that case, only the User Agent-centric architecture can still provide any useful guarantee of privacy. Thus we believe that only a User Agent-centric architecture is practically scalable to large systems.

However, our real answer to these problems is to recommend the construction of heterogenous sytems — allowing people to operate in a moded fashion, where they are free to trade off reduced functionality for much greater guarantees of privacy. Moded operation allows us to provide "full-function" location-based systems in friendly environments and to people who don't care that their current location might be found out, while providing reduced, "passive listening" functionality and a high promise of privacy to people residing in potentially hostile environments. The work in this paper has focused on a variety of possible location architectures that might serve as the basis for particular modes in a heterogeneous architecture. A separate paper addresses the design trade-offs of moded operation in such a heterogeneous system[9].

We end by observing that technology can only provide part of the answer to concerns about peoples' privacy. While we have shown that technical means may provide people with various kinds of control over revelation of private information, that is not enough to assure a person of privacy—a person must also have the freedom (legal, economic, and social) to exercise that control. If your employer seems to only give good performance reviews to people who reveal their location indiscriminantly, if your friends berate you for every time they fail to find you, if attending a conference in a foreign country requires half a dozen uses of a credit card, if banks will only do business with people who reveal their social security number, you still have a privacy problem. Ensuring that people actually have the freedom to exercise their possible controls primarly involves non-technical issues. These social and legal regulation issues are at least as important as any technological issue since they are the only means of recourse to prevent "authorized" invasions of privacy.

Acknowledgements

We thank our collegues with whom we have discussed many of the ideas in this paper, especially Dan Greene and David Nichols, who provided valuable insights and feedback to us in the early phases of the design.

References

[1] N. Ackroyd and R. Lorimer. *Global Navigation: A GPS User's Guide.* Lloyd's of London Press, 1990.

[2] D.L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[3] D. Denning. *Cryptography and Data Security.* Addison-Wesley, 1983.

[4] M.A. Eldridge, M.G. Lamming, and M. Flynn. Can a video diary improve recall? In *Proc. of HCI-92.* York, 1992.

[5] M.G. Lamming and W.M. Newman. Activity-based information retrieval: Technology in support of personal memory. In *Proc. of IFIP-92.* Congress, 1992.

[6] K. Marzullo. Synchronized time service. PhD dissertation, ???

[7] J.L. Massey. An introduction to contemporary cryptology. *Proceedings of the IEEE*, pages 533–549, May 1988.

[8] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[9] M. Spreitzer and M.M. Theimer. Providing location information in a ubiquitous computing environment. In *Proc. of the 14th ACM Symposium on Operating Systems Principles*, December 1993.

[10] J.G. Steiner, C. Newman, and J.I. Schiller. Kerberos, an authentication service for open network systems. In *Proc. of Usenix Winter Conference, 1988*, pages 191–201. Usenix Assoc., 1988.

[11] ruser manual entry of the SUNOS UNIX manual.

[12] R. Want and A. Hopper. Active badges and personal interactive computing objects. *Transactions on Consumer Electronics*, 38(1), February 1992.

[13] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *Transactions on Information Systems*, 10(1), January 1992.

[14] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, September 1992.