Access Control to People Location Information

URS HENGARTNER
University of Waterloo
and
PETER STEENKISTE
Carnegie Mellon University

Ubiquitous computing uses a variety of information for which access needs to be controlled. For instance, a person's current location is a sensitive piece of information that only authorized entities should be able to learn. Several challenges arise in the specification and implementation of policies controlling access to location information. For example, there can be multiple sources of location information. The sources can be within different administrative domains, which might allow different entities to specify policies, and policies need to be flexible. We address these issues in our design of a distributed access control mechanism for a people location system. Our design encodes policies as digital certificates, which enables decentralized storage of policies. We also present an algorithm for the discovery of distributed certificates. Furthermore, we discuss several privacy issues and show how our design addresses them. To show feasibility of our design, we built an example implementation based on SPKI/SDSI certificates. Using measurements, we quantify the influence of access control on query processing time. We also discuss trade-offs between RSA-based and DSA-based signature schemes for digital certificates.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protections—*Access controls*; E.3 [**Data Encryption**]: Standards (e.g., DES, PGP, RSA); K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms: Security, Performance, Measurement

Additional Key Words and Phrases: Certificates, credential discovery, delegation, DSA, location, privacy, RSA, SPKI/SDSI, trust

This research was funded in part by DARPA under contract number N66001-99-2-8918 and by NSF under award number CCR-0205266.
This paper is an extended version of Hengartner and Steenkiste [2004].
Authors' addresses: U. Hengartner, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1; email: uhengart@cs.uwaterloo.ca; P. Steenkiste, Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213.

1. INTRODUCTION

Ubiquitous computing environments, such as Carnegie Mellon's Project Aura [Garlan et al. 2002], rely on the availability of people location information to provide location-specific services. Location is a sensitive piece of information; releasing it to random entities might pose security and privacy risks. For example, to limit the risk of being robbed, individuals want only a small set of trusted people to be able to locate them while walking home at night. A company might want to track the location of its repairmen to increase scheduling efficiency, but it would not want competitors to have this information, since this information can indirectly reveal confidential company data (identity of clients and schedules of repairmen). In short, only authorized entities should have access to people location information.

Whereas location information has received increased attention, its access control requirements have not been studied thoroughly. Location information is inherently different from information, such as files stored in a file system, whose access control requirements have been widely studied. Location information is different since there is no single point at which access can be controlled. Instead, a variety of sources (e.g., a personal calendar or a GPS device) can provide location information. Therefore, a system providing location information has to perform access control in a distributed way, considering different sources, potentially administrated by different organizations.
This paper makes the following contributions:

- We discuss challenges that arise when specifying policies controlling access to location information.
- We present the design of a distributed access control mechanism that is flexible enough to be deployed in an environment that has multiple sources of location information or whose services are administrated by different organizations. Our design exploits digital certificates.
- We introduce a discovery algorithm for certificates that are stored in a decentralized way.
- We describe an implementation of the proposed mechanism. Our implementation is based on SPKI/SDSI certificates [Ellison et al. 1999].
- We quantify the influence of the access control mechanism on the query-processing time of a people location system and discuss trade-offs between RSA- and DSA-based signature schemes for digital certificates.

The outline of the rest of this paper is as follows: We introduce the architecture of a location system in Section 2. In Section 3, we discuss several challenges that arise when specifying policies controlling access to people location information. We explain how we deal with multiple sources of location information in Section 4. In Section 5, we present the formal model of our access control mechanism. In Section 6, we demonstrate an architecture that implements this model. We examine the storage and retrieval of digital certificates in Section 7. In Section 8, we investigate several privacy concerns that need to be taken into account by the access control mechanism. We discuss our prototype implementation in Section 9 and evaluate it in Section 10. We

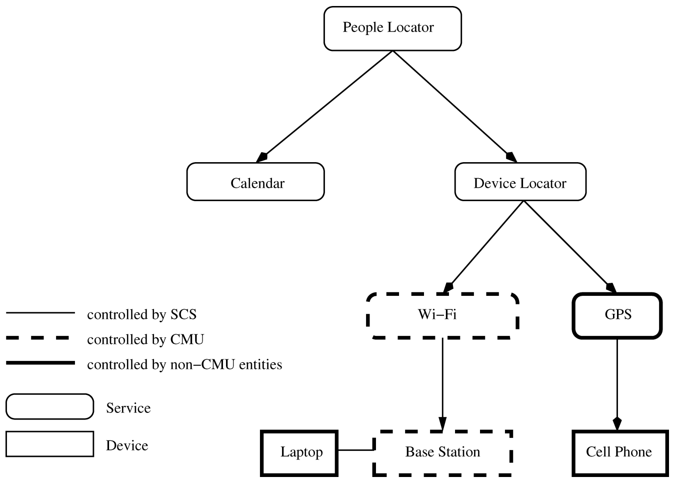


Fig. 1. Example people location system. The People Locator service forwards a query for the location of a person to the Calendar service and to the Device Locator service. The Device Locator service locates a person by locating her devices. It queries the Wi-Fi service for the location of the person's laptop and the GPS service for the location of the person's cell phone.

comment on related work in Section 11 and on our conclusions and future work in Section 12.

2. PEOPLE LOCATION SYSTEM

In this section, we introduce a people location system that exploits different sources of location information.

We assume that the location system has a hierarchical structure. The *location system* consists of multiple *location services*. Each location service either exploits a particular technology for gathering location information or processes location information received from other location services. Figure 1 shows an example of such a system, as it could be deployed in Carnegie Mellon's School of Computer Science (SCS). A client contacts the People Locator service at the root of the system. This service then contacts other services, which themselves may also contact other services. Location information flows in the reverse direction of a request (not shown in the figure). A location service can be implemented either on a single node or on multiple nodes to improve scalability and robustness.

There are two groups of location services. The first group consists of services that are aware of the location of people. The second group includes services that are aware of the location of devices. These services can locate a user indirectly by locating the device(s) that the user is carrying with her. The People Locator service, the Calendar service, and the Device Locator service belong to the first group. The People Locator service aggregates location information received from other services. The Calendar service looks at people's appointments to determine their current location. The Device Locator service maps a query for a person to potentially several queries for her devices and contacts services in the second group. In our example, this group of services consists of the Wi-Fi service and the GPS service. The Wi-Fi service keeps track of the location of wireless devices by identifying the base station(s) they are connecting to. The GPS service retrieves the location of GPS-enhanced mobile phones. We believe that our location system can easily incorporate other location services (e.g., Microsoft's Radar [Bahl and Padmanabhan 2000] or MIT's Cricket [Priyantha et al. 2000]).

A basic assumption in our work is that different organizations may administrate the various services. In our example, SCS Computing Facilities controls the Calendar service, Carnegie Mellon's Computing Services maintains the Wi-Fi service, and a phone company runs the GPS service.

Ubiquitous computing environments also provide personal information about people other than their location. We view the design of an access control mechanism for people location information as a first step in the design of a more general access control mechanism.

3. LOCATION POLICIES

To prevent location information from leaking to unauthorized entities, we employ location policies. An entity (e.g., an individual or a service requiring location information) can access a person's location information only if permitted by this person's location policy. In this section, we examine location policies and present requirements for the access control mechanism. We present a formal model for the establishment and validation of location policies based on the presented principles in Section 5.

3.1 Controllable Properties

In general, a location policy states who is allowed to get location information about someone. For example, Alice's location policy might specify that Bob is allowed to locate her. In addition, a location policy must support more specific access control. Namely, we believe that at least the following properties should be controllable:

1. *Granularity.* A policy can restrict the granularity of the returned location information. For example, a policy can state that the building in which a queried user is staying is returned instead of the actual room (e.g., "CMU Wean Hall" vs. "CMU Wean Hall 8220").
2. *Locations.* A policy can contain a set of locations (e.g., buildings or rooms). The location system will return location information only if the queried user is at one of the listed locations. For example, a policy can state that Bob is allowed to find out about Alice's location only if she is in her office.
3. *Time Intervals.* Location policies can limit time intervals during which access should be granted. For example, access can be restricted to working hours.

3.2 Individual vs. Institutional Policies

Depending on the environment, different entities specify location policies. For some environments, a central authority defines policies, whereas for others, individuals set them. In addition, some environments might give both individuals and a central authority the option to specify policies.

In general, governments and companies probably do not want the location of their employees or the people in their buildings to be known to outsiders, whereas this information can be delivered to (some) entities within the organization. In such cases, a central authority would establish the location policies such that no information is leaked. For other environments, such as a university or a shopping mall, the institution behind the environment cares less about where an individual is. For these cases, it should be up to an individual to specify her location policy. In this paper, we demonstrate the application of our approach in a university and a hospital environment.

In the rest of the paper, we call the entity that establishes location policies *policy maker*. Other access control models, such as role-based access control, refer to this entity with the term *administrator*. We refrain from using this term since an administrator is typically assumed to be a centralized entity that decides on all the people's location policies, which might not hold in our case.

3.3 Distributed Location Policies

For some environments, a policy maker should be able to establish a location policy in a distributed way. For example, in a university environment, Alice can grant Bob access to her location information. In addition, Alice should be able to state whether this access right is forwardable. A forwardable access right allows Bob to forward the granted access right to third parties. Therefore, Alice's location policy can be set by both Alice and Bob (and third parties that receive forwardable access rights from Bob). Similarly, Alice might decide to grant Carol's friends access to her location information. Here, it is Carol, not Alice, who decides on the identities of Carol's friends, that is, both Alice and Carol can influence the location policy. For other environments, we do not want this flexibility. In a hospital environment, a doctor who is granted access to a patient's location information should probably not be able to forward this access right. In such an environment, only the policy maker decides on the location policy. Here, the model is identical to traditional role-based access control, where an administrator decides on mappings both between resources and roles and between roles and people. Note that even though an entity might not be allowed to forward access rights, it could still issue queries on behalf of third parties.

4. SERVICE TRUST

As explained in Section 2, a location system can consist of multiple location services. Some of these services, such as the People Locator service shown in Figure 1, do not generate their own location information. Instead, they process location information received from other services. To avoid information leaks, the location system must ensure that only services that implement access control are given location information.

One way to implement this condition is to require that a service is granted access in the location policy of the queried user. Being granted access means that a service can actively issue requests for information. This option is appropriate for services that must be able to issue requests for location information. For example, the Device Locator service shown in Figure 1 has to generate location queries for devices upon receiving a location query for a user. However, for services such as the People Locator service, this option gives a service more privileges than it really needs to have. The People Locator service only forwards requests received from clients to other service that provide people location information. By granting it the right to issue requests, we increase exposure in case of a break-in. If an intruder issues requests, the location system will grant access to these requests.

Because of these reasons, we introduce the concept of *service trust*. If a service that is not granted access (i.e., it cannot actively issue requests) is trusted, it is given location information when forwarding an authorized request from someone else. We present a formal model for the definition of trusted services in Section 5.

The policy maker responsible for a user's location policy also identifies this user's trusted services. For example, in a university environment, each user can define her own set of trusted services. However, it might be difficult for a user to decide what services to trust. Therefore, we allow users to have third parties make this decision on their behalf. We elaborate on this mechanism in Section 5.4.2.

For a trusted service, the trust assumption is that the service implements access control in the following way:

1. The service ensures that the location policy for the requested information grants access to the entity that issued a request. If this check fails, access will be denied.
2. The service then checks whether the entity from which it received the request corresponds to the entity that issued the request. If it does, access will be granted. If it does not, the service must have received a forwarded request. Therefore, the service has to ensure that the policy maker trusts the entity from which the service received the request. If so, access will be granted or else access will be denied.

5. FORMAL MODEL

Based on our discussion in Sections 3 and 4, we now present a formal model for our access control mechanism for a people location system. We require that services respond to a location request only after performing a location policy check that verifies that the entity issuing the request has access. In addition, for forwarded requests, services need to ensure that the service from which they receive a request is trusted before returning an answer. To match the distributed nature of the location services, our location system runs access control in a distributed way. To eliminate redundant access control checks, services can delegate access control to other services. We now describe these concepts in more detail.

5.1 Policy and Trust Statements

Policy makers issue statements about a person's location policy or her trusted services. As mentioned in Section 3.3, if a policy is established in a distributed way, other entities will also make such statements. Our formalism for expressing these statements is a modified version of the formalism introduced by Clarke et al. [2001]. The formalism models the expressiveness of SPKI/SDSI certificates [Ellison et al. 1999], which we use in our architecture. SPKI/SDSI is a trust management system that supports decentralized specification and evaluation of security policies. This flexibility allows us to give different entities the right to define location policies, as discussed in Section 3.2, and to implement more complex access control strategies like vertical (e.g., in a hospital) or horizontal access control (e.g., in the military).

SPKI/SDSI is not founded on a formal semantics. However, there has been research into finding a logic-based semantics for SPKI/SDSI and understanding its delegation semantics [Halpern and van der Meyden 2001; Howell and Kotz 2000a; Li and Mitchell 2003]. These existing semantic models also apply to our access control mechanism.

In our formalism, an issuer, A, makes a statement concerning a subject, B, or

$$A \xrightarrow[scope]{type} B$$

The type of statement is described above the arrow ("*policy*" or "*trust*"). In decisions of type "*policy*," A grants B access to location information, assuming A itself has a forwardable access right to this information. In statements of type "*trust*," A specifies that A trusts service B. The scope of a statement can be limited by listing the scope below the arrow. In particular, for policy statements, the scope indicates the person whose location information the statement grants access to. For trust statements, the scope states the person whose trusted services the statement defines. The scope can also include a list of controllable properties, as discussed in Section 3.1. However, for readability reasons, we do not include them in our formalism.

An issuer making a statement should be able to indicate whether the subject of the statement is allowed to forward the access right to location information or the right to define a set of trusted services to third parties. We mark forwardable statements with the "$*$" character. For example, in the statement

$$A \xrightarrow[A]{policy*} B$$

A grants B access to A's location information (assuming that A itself has been granted a forwardable access right to this information) and allows B to forward the access right to third parties.

The subject, but not the issuer, of a policy or trust statement can be a group of people, such as $B.friend$. (There can be more than one identifier, such as $B.husband.friends$; we refer to Clarke et al. [2001] for details.) It is B who issues membership statements for $B.friend$. For example, B states that C is a friend, or

$$B.friend \longmapsto C$$

The entity on the right-hand side can also be a group of entities.

In our access control architecture, statements are implemented in two different ways. A service that provides a person's location issues a statement that grants a forwardable access right to the policy maker responsible for this person's location policy. The service stores this statement locally. For example, it can keep an access control list (ACL) for each person whose location information it provides. A policy maker that grants an entity access to location information will issue this statement in a SPKI/SDSI certificate and give it to the entity, which will present it to a service when requesting information. We use the same approach for trust statements.

When a client requests location information from a service, the service runs access control by combining locally stored and certificate-based policy statements (and potentially membership statements). The goal of this combination is to obtain a statement that grants the client access to the requested information. Clarke et al. [2001] examine the combination of statements in the case of SPKI/SDSI. We discuss this combination in the context of location policies in Section 5.2. A similar combination is required for trust statements, which is the focus of Section 5.3.

5.2 Location Policy Check

The issuer of a request will give a service a set of policy statements based on digital certificates. The service will validate the signatures of these certificates and combine the statements in the certificates with a locally stored policy statement. Ignoring membership statements, the combined policy statements will form a chain. The scope of each statement in the chain must either be unlimited or limited to the person whose location is requested. In addition, each statement, apart from the last one, must allow forwarding.

We now demonstrate this mechanism based on our formalism. To illustrate the flexibility of our formalism, we apply it to two environments with different requirements on the types of defined location policies. In particular, we examine a university and a hospital environment.

5.2.1 *University.* The administrator of the People Locator service, *PL*, chooses Alice as the policy maker for her location information and grants her a forwardable access right to this information in the corresponding ACL of the service:

$$\text{PL} \xrightarrow[Alice]{policy_*} \text{Alice} \tag{1}$$

Alice defines her location policy by issuing additional statements in the form of certificates. For example, she grants access to Bob:

$$\text{Alice} \xrightarrow[Alice]{policy} \text{Bob} \tag{2}$$

When Bob inquires about Alice's location, Bob will give this certificate to the People Locator service. The service will combine the statement in the certificate with Statement (1) and deduce

$$\text{PL} \xrightarrow[\textit{Alice}]{\textit{policy}} \text{Bob}$$

that is, Bob has access to Alice's location.

A location service does not have to be aware of all the users' identities. If Carol can present a valid certificate to a service and the service can form a chain, Carol will be granted access. This approach makes dealing with unknown users easy. Another benefit of digital certificates is that Alice can grant access rights to groups of entities (e.g., Bob's friends) in a single certificate:

$$\text{Alice} \xrightarrow[\textit{Alice}]{\textit{policy}} \text{Bob}.\textit{friend}$$

If Bob declares Carol to be his friend (i.e., Bob.*friend* $\longmapsto$ Carol), she will be granted access.

5.2.2 *Hospital.* In a hospital environment, a central authority, CA, manages everybody's location policies, that is, the People Locator service locally stores an unlimited statement granting a forwardable access right to the authority, or

$$\text{PL} \xrightarrow{\textit{policy}*} \text{CA}$$

The centralized entity then establishes location policies by issuing further statements. For example, it grants doctor Bob nonforwardable access to patient Alice's location:

$$\text{CA} \xrightarrow[\textit{Alice}]{\textit{policy}} \text{Bob}$$

In addition to this statement, the authority might also define a statement that grants Alice a (forwardable) access right to her location information.

5.3 Service Trust Check

When a service receives a forwarded request, it must also check whether the entity from which it got the request is trusted. Similar to the location policy check, this entity will present a set of trust statements based on digital certificates to the service. The service will combine these statements and a locally stored trust statement. We show how the Device Locator service handles trust decisions. Typically, this service is not directly contacted by individuals, but by other services (e.g., the People Locator service).

5.3.1 *University.* The administrator of the Device Locator service, DL, gives Alice the right to define her trusted services in the service's ACL. Alice then states that she trusts the People Locator service, PL. The issued statements appear as follows:

$$(a)\quad DL \xrightarrow[\textit{Alice}]{\textit{trust}*} \text{Alice} \qquad (b)\quad \text{Alice} \xrightarrow[\textit{Alice}]{\textit{trust}} \text{PL} \tag{3}$$

When receiving a forwarded request for Alice's location from the People Locator service, the Device Locator service deduces from these certificates that the People Locator service is trusted for the requested information.

5.3.2 *Hospital.* The administrator of the Device Locator service lets the central authority specify the trusted services for everybody's location information. The authority states that the People Locator service is trusted. The statements appear as follows:

$$(a)\quad \text{DL} \xrightarrow{trust*} \text{CA} \qquad (b)\quad \text{CA} \xrightarrow{trust} \text{PL}$$

5.4 Delegation

Chains of policy or trust statements exploit the concept of delegation, where an entity grants a second entity a particular right and also allows the second entity to grant the right to other entities. For example, Alice grants Bob access to her location information and lets him forward this access right to other entities. In this section, we elaborate on two other scenarios where delegation is useful.

5.4.1 *Delegating Access Control.* Each location service has to implement access control. However, to reduce overhead or in the case of low processing power, we do not require each service to combine the potentially many statements required for access control itself. Instead, it can delegate this task to some other service. For example, the Calendar service shown in Figure 1 is likely to delegate access control to the People Locator service since both services are run by the same organization and the People Locator service needs to combine statements for each request anyway. After this combination, the People Locator service issues a statement in a certificate that directly authorizes the querying entity. The service gives this certificate to the Calendar service, which does not have to combine the statements again. We describe this optimization in more detail in Section 6.2.

5.4.2 *Delegating Service Trust.* If there are lots of services available, it will be cumbersome for a user to issue a certificate for each service that she trusts. We assume that trust in a service is closely tied to the organization that administrates this service. For example, a user might trust all the services run by her company. Therefore, we give users the option to state in a certificate that they trust all the services in a particular organization. The organization then generates a certificate for each of the services that it runs. In this situation, a user effectively delegates the decision on which services she trusts to the organization and relies on the organization to do the right thing.

For the university environment, instead of issuing Statement (3b), Alice would issue the following statement: (ACME.*service* represents a service run by organization ACME.)

$$\text{Alice} \xrightarrow[Alice]{trust} \text{ACME}.service \tag{4}$$

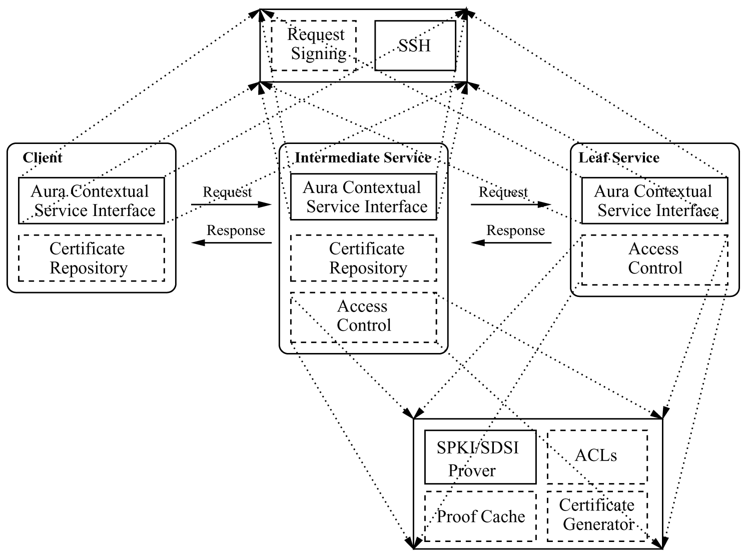


Fig. 2. Architecture of access control mechanism. The components with dashed lines are new; the components with solid lines are based on existing work.

For each service in the organization, the administrator of the organization issues a membership statement, such as,

$$\text{ACME}.\mathit{service} \longmapsto \text{PL} \tag{5}$$

For the example in Section 5.3, the Device Locator service would now deduce from Statements (3a), (4), and (5) that the People Locator service is trusted.

6. ARCHITECTURE

In this section, we present the architectural components required for deploying access control in a location system, as formalized in Section 5. We then discuss in an example scenario how these components interact to process a request.

In the rest of this paper, we concentrate on the university example discussed in Section 5. However, the presented architecture is flexible enough to be also applied to the hospital example.

6.1 Overview

Figure 2 gives an overview of the architecture. The figure shows the three possible types of entities involved in the processing of a request. In particular, there are (1) clients, which issue requests for a person's location, (2) intermediate services, which either forward received requests (e.g., People Locator service) or create requests for a device's location (e.g., Device Locator service), and (3) leaf services, which provide location information based on a particular technology (e.g., Calendar service). We now discuss each of the architectural components in detail.

Clients and services run within Carnegie Mellon's Aura environment [Garlan et al. 2002]. The Aura Contextual Service Interface [Judd and Steenkiste 2003] enables communication between entities. This RPC protocol runs over HTTP and exchanges XML-encoded messages. We extended the protocol to transmit digital certificates and to digitally sign issued requests. In addition, we modified the protocol to run over an SSH-based transport layer, which provides authentication of peers and confidentiality and integrity of the transmitted messages. The transport layer was implemented in previous work [Howell and Kotz 2000b] and uses RSA public/private key pairs. We optimized this implementation and extended it to alternatively use DSA public/private key pairs for authentication, together with a Diffie–Hellman session key exchange secured against man-in-the-middle attacks [Diffie et al. 1992].

The certificate repository contains certificates issued to an entity. In case of an entity that issues requests, these certificates express either policy or membership statements. In case of a service that forwards requests, these certificates express either trust or membership statements. Clients or services retrieve any required certificates from their repository when issuing or forwarding a request.

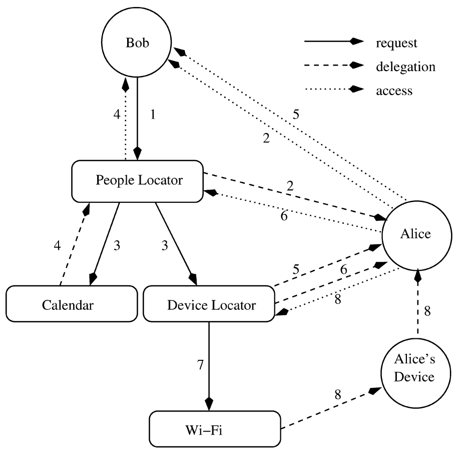
The access control component consists of four parts: ACLs contain a service's locally stored policy or trust statements. The SPKI/SDSI prover takes a bunch of statements, either locally stored ones or statements based on SPKI/SDSI certificates, validates the signatures of certificate-based statements, combines the statements, and returns the combined statement. The statements are either policy or trust statements. The prover also supports membership statements. The SPKI/SDSI prover was implemented in Java and is based on previous work [Howell and Kotz 2000b]. The proof cache caches combined statements returned by the SPKI/SDSI prover. Access control consults this cache before invoking the prover. A cache hit prevents invocation of the prover and thus revalidation of the digital signatures of certificates underlying the cached statement, which can be a computationally expensive operation. The certificate generator is used by a service that runs access control on behalf of some other service, as discussed in Section 5.4.1, in order to issue a certificate corresponding to the combined statement returned by the SPKI/SDSI prover.

6.2 Example Scenario

In this section, we present the step-by-step processing of a request by our architecture. The example is based on the university example discussed in Section 5. The system shown in Figure 3 processes a request from Bob for Alice's location. We differentiate between simply giving someone access to a resource ("access" arrows) and also allowing the recipient of the access right to forward this access right ("delegation" arrows).

In the initial step (not shown), entities in the system grant access rights to other entities and they delegate policy decisions:

- The administrators of the People Locator service and the Device Locator service each define a policy statement that delegates the right to decide on Alice's location policy to Alice [Statement (1) in case of the People Locator service]. The administrator of the Device Locator service also establishes a trust statement saying that it is up to Alice to specify which services she trusts [Statement (3a)]. The Calendar service is administered by the same



#	Required Statements
2	$\text{PL} \xrightarrow[Alice]{policy*} \text{Alice}, \quad \text{Alice} \xrightarrow[Alice]{policy} \text{Bob}$
4	$\text{Calendar} \xrightarrow{policy*} \text{PL}, \quad \text{PL} \xrightarrow[Alice]{policy} \text{Bob}$
5	$\text{DL} \xrightarrow[Alice]{policy*} \text{Alice}, \quad \text{Alice} \xrightarrow[Alice]{policy} \text{Bob}$
6	$\text{DL} \xrightarrow[Alice]{trust*} \text{Alice}, \quad \text{Alice} \xrightarrow[Alice]{trust} \text{PL}$
8	$\text{Wi-Fi} \xrightarrow[Alice's\ device]{policy*} \text{Alice's device},$ $\text{Alice's device} \xrightarrow[Alice's\ device]{policy*} \text{Alice},$ $\text{Alice} \xrightarrow[Alice's\ device]{policy} \text{DL}$

Fig. 3. Processing of a request, in which Bob inquires about the location of Alice. The location system processes the request in multiple steps. For each step, we show the chains of statements that are validated by the services.

organization as the People Locator service. Therefore, its administrator has the People Locator service run access control on its behalf and delegates the right to decide on people's location policy to the People Locator service, or

$$\text{Calendar} \xrightarrow{policy*} \text{PL}$$

These statements are all stored locally in the form of ACLs.

- The administrator of the Wi-Fi service defines an ACL that allows the owner of Alice's laptop to decide on the device's location policy, or

$$\text{Wi-Fi} \xrightarrow[Alice's\ device]{policy*} \text{Alice's device}$$

Alice, as the owner of her device, assigns this right to herself in a signed certificate, or

$$\text{Alice's device} \xrightarrow[Alice's\ device]{policy*} \text{Alice}$$

and stores the certificate in her repository.

- Alice grants Bob access in a policy certificate [Statement (2)]. She also certifies that she trusts the People Locator service [Statement (3b)]. In addition, since the Device Locator service is going to create a request for her device upon receiving a request for her location, she needs to give the Device Locator service access to her device's location information in a policy certificate, or

$$\text{Alice} \xrightarrow[Alice's\ device]{policy} \text{DL}$$

All these certificates are stored in the certificate repository of the entity on the right-hand side of the statement expressed by a certificate, that is, the entity that will need the certificate for a request. Alice also gives the certificate granting her access to her device's location information to the Device Locator service.

Given this setup, the system processes a request from Bob for the location of Alice as follows: (The numbers in Figure 3 correspond to the steps given below.)

1. Bob retrieves the certificate granting him access to Alice's location from his certificate repository and uses the Aura Contextual Service Interface to send a signed request and the certificate to the People Locator service.
2. The People Locator service checks Alice's location policy by giving the chain of statements consisting of its ACL for Alice's location information and of the access right received from Bob to the SPKI/SDSI prover.
3. Based on the combined statement returned by the prover, the People Locator concludes that Bob should be granted access and uses the certificate generator to issue a certificate that directly grants Bob access to the requested location information, or

   $$\text{PL} \xrightarrow[Alice]{policy} \text{Bob}$$

   The service then retrieves the trust certificate received from Alice from its certificate repository and forwards the certificate, together with the request from Bob, Bob's certificate from Alice, and the just-issued certificate, to the Calendar and Device Locator services.
4. The Calendar service uses its SPKI/SDSI prover to check Alice's location policy. The prover validates the chain of statements consisting of the service's ACL and of the certificate just issued by the People Locator service. The service then processes the request.
5. The Device Locator service checks Alice's location policy by validating a chain of statements in the same way as the People Locator service.
6. Since the Device Locator service received a forwarded request, it also needs to verify that the People Locator service is trusted by Alice. It has the SPKI/SDSI prover validate the chain of statements consisting of its ACL and of Alice's trust statement.
7. The Device Locator service determines Alice's wireless device. It retrieves the certificates granting Alice and the service access to the device's location from its repository and sends them, together with a new request, to the Wi-Fi service.
8. The Wi-Fi service uses the SPKI/SDSI prover to check the location policy of Alice's laptop, that is, it validates the chain of statements from its ACL to the Device Locator service. It then processes the request by contacting all the base stations. The base stations could run access control in a similar way. However, since they are typically resource limited, we expect them to be statically configured to return information only to the Wi-Fi service.

7. CERTIFICATE STORAGE AND RETRIEVAL

As discussed in Section 5, a client needs to present a set of policy statements in the form of digital certificates to a service. Similarly, services might have to show trust statements to other services. We now discuss where clients and services store certificates and how clients retrieve certificates required for a request.

7.1 Policy and Trust Certificates

In a simple solution, policy and trust certificates are kept in a centralized repository. However, this repository can become a bottleneck. We pursue a different approach: We store a certificate with the entity that is granted access to location information or that is given trusted status in the certificate. For example, when Alice grants Bob access to her location information, she gives the corresponding certificate to Bob, who stores it in his own repository. For a request, he retrieves the certificate from his repository and hands it over to the location service. If Bob did not want to run his own repository, he could store certificates in a repository run by a third party, from which he receives them for requests. Similar to policy certificates, a trust certificate is stored with the service that is given trusted status in the certificate.

Certificate storage and retrieval as discussed above are sufficient for the simple case where an entity is granted access or given trusted status in a single certificate. However, in more complicated scenarios involving forwarded access rights, the entity requires multiple certificates to demonstrate this property. In addition, the entity might require certificates demonstrating membership in a particular group or organization. We next discuss how we support these scenarios. We limit our discussion to policy certificates; we deal with trust certificates in the same way.

7.2 Forwarded Access Rights

We require an entity that forwards a received access right to third parties to hand over to these entities not only the certificate that forwards the access right, but also all certificates that grant the forwarded access right to the entity in the first place. For example, assume that Bob has access to Alice's location information. To also grant access to Carol, he issues a certificate and hands it over to Carol, together with the certificate(s) he received from Alice or from other people granting him access to Alice's location information.

7.3 Membership Certificates

The model introduced in the previous section assumes that an entity will issue a certificate that forwards an access right only after having actually acquired this access right. Whereas this is a realistic assumption for policy certificates, it does not necessarily hold for certificates expressing membership in a group. For example, Bob is free to issue a certificate saying that Carol is a friend of his before Alice issues a certificate granting Bob's friends access to her location information. Here, the (implicit) forwarding decision is made before the actual

$$\text{Alice} \xrightarrow[Alice]{policy} \text{Bob}.friend \qquad \text{Bob}.friend \longmapsto \text{Carol}.friend \qquad \text{Carol}.friend \longmapsto \text{Dave}$$

Fig. 4. Discovery of membership certificates. The search can start either at Bob or at Dave. For the former approach, certificates are stored with their issuer (i.e., Alice, Bob, and Carol). For the latter approach, certificates are stored with the entity on the right-hand side (i.e., Bob, Carol, and Dave).

access right is issued. Therefore, a client needs to be able to discover certificates that grant access to a group of which the client is a member.

To locate these certificates, a client could start its discovery at the entity that issues relevant membership certificates and explore all these certificates. Let us consider the scenario given in Figure 4. Dave wants to locate Alice and knows about Alice's policy certificate granting access to Bob's friends. Dave's search strategy is to enumerate all these friends, hoping that he is one of them. Namely, Dave acquires all membership certificates for Bob.*friend* from Bob and then all membership certificates for Carol.*friend* from Carol. This strategy is expensive and can fail if membership certificates issued by institutions or individuals are not publicly available. For example, most companies do not provide public listings of their employees.

The discovery of membership certificates can be simplified if, instead of an institution, members of the institution hold the certificate proving membership and members decide whether to reveal a certificate to third parties. We believe that having members hold membership certificates is practical. Many institutions already release membership cards to their members; a membership certificate could be encoded on this card. For individuals, social networks (e.g., [Orkut]) require explicit communication between entities for letting them become friends; the certificate handover could be part of this exchange. For the example in Figure 4, the discovery of membership certificates now proceeds in the other direction, that is, right to left instead of left to right. In particular, Dave acquires all memberships certificates from Carol that have Carol.*friend* become a member of another group. Since Bob.*friend* is such a group, Dave can stop his search. Note that Carol can decide to return a certificate to an entity only if the entity has legitimate interest in the certificate. In the example, an obvious strategy for Carol is to limit access to friends of hers. In this case, Dave would have to present his membership certificate to Carol to prove friendship.

If Dave failed to locate required certificates, as a last resort, he could directly ask Alice to issue a policy certificate to him. Alice could automate this process by, for example, employing a sophisticated rule engine that decides on Alice's behalf whether Dave should be granted access, based on credentials that Dave presents to the rule engine.

Thus for, we have assumed that a client knows the relevant policy certificates. However, in practice, this assumption might not hold. We now present a combined algorithm that discovers both policy and membership certificates. Assume that entity B wants to access location information about entity A and that A defines A's location policy. [A (sub)step is executed only if the preceding one fails.]

1. B tries to locate a policy statement stating

$$A \xrightarrow[A]{policy} B$$

in its collection of statements. (A statement is a single certificate or a combination of certificates.)

2. For each membership statement "$C.foo \longmapsto B$" in its collection, B asks C
   (a) for a policy statement showing

$$A \xrightarrow[A]{policy} C.foo$$

   (b) for membership statements of the form "$D.bar \longmapsto C.foo$". If successful, B combines the statement with "$C.foo \longmapsto B$" to "$D.bar \longmapsto B$" and adds this statement to the pool of statements over which step (2) is iterating.

When allowing membership statements of the form "$A.friend \longmapsto B.brother.friend$" (i.e., friends of B's brothers are also friends of A), we need to extend the algorithm slightly. In particular, when querying an entity about additional membership statements in step (2b), the entity also needs to return statements that cover the entity itself. For example, given the statement above, "$B.brother \longmapsto C$", and "$C.friend \longmapsto D$," D needs C to return "$B.brother \longmapsto C$" in order to be able to learn about B and to ask it for membership statements for $B.brother.friend$. Due to privacy reasons, C might refuse to hand out this knowledge. In this case, D could ask C to act as a proxy and to finish the search on its behalf. Eventually, this will still allow D to gather the denied knowledge. However, it will prevent D from learning about C's membership statements that are not required for D being granted access. To support membership statements that have more than two identifiers on their right-hand side, we can break them up into multiple membership statements having, at most, two identifiers on their right-hand side.

If an entity owns an access right because it is part of a group and decides to forward the access right, it will also have to forward all the membership certificates required for being granted access. Therefore, an entity needs to perform, at most, one discovery of certificates for a request. The lookup in step (1) of our algorithm can be implemented as an operation with $\mathcal{O}(1)$ worst-case time complexity. For step (2), if we allow only membership statements of the form "$A.friend \longmapsto B$" or "$A.friend \longmapsto B.brother$," we can think of the membership statements as being organized in a directed graph. The discovery algorithm traverses this graph by looking at each edge (i.e., statement), at most, once. Therefore, if there are n membership statements, the worst-case time complexity is $\mathcal{O}(n)$. If we allow membership statements with two identifiers on their right-hand side, the worst-case time complexity of the discovery algorithm becomes $\mathcal{O}(n^3)$. We refer to Li et al. [2003b] for a proof of this bound.

7.4 Caching, Expiration, and Renewal of Certificates

When discovering certificates, an entity can cache received and derived statements in its collection of statements so that it can reuse them for future requests.

Certificates can expire. A statement based on an expired certificate becomes invalid. We treat certificate renewals in the same way as new certificates; a certificate forwarding an access right that is granted in an expired and then renewed certificate needs to be sent out again, together with the renewed certificate. For certificate revocation, we rely on mechanisms proposed earlier, such as positive or negative on-line validation of certificates [Ellison et al. 1999].

8. PRIVACY CONSIDERATIONS

The discussion in Section 7 brought up some privacy issues that need to be considered during the retrieval of certificates. In this section, we discuss some additional privacy issues that a people location system should take into account.

8.1 Location Hiding

Limiting access to location information is a legitimate concept in a privacy-aware environment. However, not granting someone access to location information could have negative consequences for an individual. For example, an individual not granting her manager access to her location information could lead to the impression that the individual has something to hide. Alternatively, a service might refuse to offer its functionality to an individual if it failed to access the individual's location information, although the service does not require location information for providing the functionality. In these scenarios, instead of denying access to location information, it is possible to instruct a location service to return wrong location information, potentially in a very coarse-grained way (e.g., "off campus"). Returning wrong location information can also be useful when an individual is busy and does not want to be disturbed.

While it is possible to return wrong location information, this concept requires extreme care. (We do not support it in our sample implementation.) First, figuring out what wrong location information to return can be tricky. For example, it might be necessary to come up with realistic movement patterns. Second, returning wrong location information does not necessarily prevent an entity from learning that, in fact, its request for access was denied. In particular, the entity could try to and fail to validate the received information. Third, if the concept gets employed in a broad way, it will ultimately lead to entities mistrusting the location system and stop using it. As a trade-off, we suggest returning correct, but very coarse-grained location information. Although this will negatively influence an individual's privacy, the damage is limited.

For scenarios where individuals do not want to be disturbed, we suggest, based on a similar concept from instant messaging [Day et al. 2000], to make the notion of availability an explicit part of the location system. It should be possible to alert a client to the fact that an individual is busy. For example, the location system could provide location information enhanced with availability information. Alternatively, for busy individuals, it could return only availability information to explain the lack of location information. These solutions require that an individual provides her availability status to location services.

8.2 Confidentiality of Certificates

As mentioned in Section 7, due to privacy reasons, membership certificates might not be publicly available. Privacy also calls for the confidentiality of policy and trust certificates. By storing the certificates with entities to which they are issued, we avoid a centralized repository, where certificates are publicly available. Arguably, it could make sense to keep certificates away from the entity to which they are issued. For example, Alice might feel embarrassed when handing out a certificate that includes a constraint denying access when she is in the bathroom. However, ultimately, an entity being granted access by such a certificate will be able to figure out any constraints imposed on the access right by observing the location system.

Whereas shielding certificates from entities to which they are issued is of limited use, keeping certificates away from location services can be useful. Alice might want to use a location service offered by her cell phone provider for making her location information available. Since the service is run by an institution with which she has only a business relationship, she would like to avoid that, upon a request, the service learns personal information from the certificates presented to the service. For example, membership certificates reveal personal information, such as who is a friend of whom.

To address this scenario, policy makers can use a proxy host. To prevent certificate information from leaking to a location service, a policy maker has all requests go through the proxy host. This host runs access control to location information in the same way as the other services in Section 6.2, that is, there is a forwardable access right to the policy maker in the host's ACL and the host combines this and statements received from a client for the access control decision. If this decision is positive, the proxy host will issue a new request to the location service. This service lists the proxy host in its ACL for the requested information. For example, the People Locator service has an entry

$$\text{PL} \xrightarrow[\textit{Alice}]{\textit{policy}} \text{Proxy Host}$$

A variant of this method is to implement the People Locator service in a completely distributed way. That is, each policy maker runs a People Locator service on the policy maker's proxy host. However, running and administering a complete People Locator service is a potentially heavyweight operation.

If the policy maker is a central authority, the proxy host can be the authority itself. If the policy maker is an individual, the proxy host is a designated host. An individual has to make a trade-off between her privacy concerns and the convenience provided by her chosen solution. If the individual is willing to trust an organization, she can have the organization run a proxy host for her, or else she needs to set up her own proxy host.

8.3 Short-Term Use of Location Services

In general, we assume that an individual employs a fixed set of location services for providing her location information. She might have a business relationship with these services (e.g., services run by her employer or her cell phone provider), which implies that a service knows about the identity of the individual. However, the individual might also want to use location services with which she does not have such a relationship. For example, an amusement park could hand out badges to visitors, which allow the park to monitor its visitors for security reasons or parents to monitor their kids. In such scenarios, it should be possible for an individual to grant other entities access to this information, but the individual should not be required to make her personal information available to the location service in the park.

Our architecture easily supports such a scenario. Upon entering the park, an individual registers her badge with the Device Locator service. This service is run by an entity trusted by the individual and the individual grants it access to her badge's location. The Device Locator service then sends a query for the location of the badge to the location service in the park whenever there is a request for the location of the individual. Such a query will be accompanied with the certificate that grants access to the Device Locator service. Whereas some types of certificates (e.g., X.509) contain identity information about people, SPKI/SDSI certificates do not contain any such information. Instead, entities are identified solely with a public key. In this solution, the location service in the park learns the public key of the Device Locator service. If an individual uses the same Device Locator service during multiple visits to the park, the service in the park can recognize the individual (unless the Device Locator service is shared by many individuals). To avoid such information leaks, the individual can have the Device Locator service use a different public key for each visit. In addition, we can employ mix-based solutions [e.g., Chaum 1981; Syverson et al. 1997] to keep lower-level identification information (such as IP addresses) away from the service in the park. Alternatively, an individual can use a shared Device Locator service run by a third party for the duration of her visit.

9. PROTOTYPE IMPLEMENTATION

In this section, we illustrate how we encode the policy and trust statements introduced in Section 5 as SPKI/SDSI certificates [Ellison et al. 1999]. In addition, we present a prototype implementation of a people location system.

9.1 Application of SPKI/SDSI Certificates

Authentication and authorization based on SPKI/SDSI certificates [Ellison et al. 1999] do not rely on the existence of a global naming structure as, for example, the one introduced for X.509 certificates. Instead, the authentication and authorization step are merged, and a certificate gives access rights directly to a public key. Tools exist that evaluate chains of SPKI/SDSI certificates and decide whether requests should be granted access.

We demonstrate how Statements (1) and (2) in Section 5.2 can be implemented with SPKI/SDSI certificates. Keywords are printed in bold. The certificates have to be accompanied by signatures, which are not shown here.

In Statement (1), the People Locator service grants Alice (more specifically, her public key) the right to define her location policy. (`pub_key:foo` is replaced by the actual public key of foo in the real implementation.)

```
(cert
  (issuer (pub_key:people_locator))
  (subject (pub_key:alice))
  (propagate)
  (tag (policy alice)))
```

The keyword **propagate** states that forwarding is allowed. Alice can thus issue additional certificates that grant other people access to her location information. Corresponding to the notation introduced in Section 5.1, the entries following the keyword **tag** specify the type of the certificate (either `policy` or `trust`) and its scope (e.g., `alice`).

In Statement (2), Alice gives Bob access to her location information. As mentioned in Section 3.1, a policy maker might want to restrict access to information based on granularity, location, or time. For simplicity reasons, we refrained from adding these controllable properties to the formalism. However, our certificates do allow their specification. The following certificate corresponds to Statement (2), enhanced by some controllable properties.

```
(cert
  (issuer (pub_key:alice))
  (subject (pub_key:bob))
  (tag (policy alice
    (* set (* prefix world.cmu.wean) world.cmu.doherty.room1234)
    (* set (monday (* range numeric ge #8000# le #1200#))
      (tuesday (* range numeric ge #1300# le #1400#)))
    coarse-grained)))
```

Alice grants access to Bob's public key (`pub_key:bob`). Since Alice omits the keyword **propagate**, Bob is not allowed to issue further certificates for Alice's location information. Bob can locate Alice only if she is either in Wean Hall or in Room 1234 in Doherty Hall and on Monday between 8 AM and 12 PM and on Tuesday between 1 PM and 2 PM. Finally, Bob has only coarse-grained access to Alice's location information.

Trust certificates are implemented in a similar way. The information in the tag section needs to be modified accordingly. For example, **tag** `(trust alice)` identifies certificates that declare Alice's set of trusted services.

The tag mechanism provided by SPKI/SDSI certificates is powerful. It allows us to implement the properties outlined in Section 3.1 within the SPKI/SDSI framework and we do not have to resort to implementing a separate specification and verification solution. However, the flexibility of the tag mechanism does have its limits. It requires all the services in the system to have a common syntax, which may be difficult to enforce in heterogeneous environments. Moreover, it is difficult to support more general constraints, since the SPKI/SDSI prover shown in Figure 2 is currently not aware of the semantics of the entries in the tag section and applies only simple string comparison operations (extended by sets of strings and prefixes of strings). Therefore, supporting more sophisticated temporal constraints, such as the ones suggested by Bertino et al. [2001] or Bhatti [2003], requires changes to the prover.

9.2 Location Service

We have built a subset of the location system shown in Figure 1. In particular, we have implemented the People Locator service, the Device Locator service, the Wi-Fi service, and two Calendar services, which exploit calendar information from the Ical program [ICAL] and from the centralized Oracle CorporateTime calendar system. In addition, we have implemented a location service that uses login information to determine a person's current location.

We have built a web server-based frontend to the location system that allows users to specify their location policies and conduct searches for other users. The frontend makes dealing with certificates transparent to users. Public keys, password-encrypted private keys, and certificates are stored at the web server. The web server creates signed requests on behalf of clients and transmits the requests together with any required certificates to the People Locator service. Individual users are not given any certificates, therefore, revocation becomes easy; the web server merely deletes the revoked certificates.

When deploying our system in a different environment, we have to modify only this frontend such that the generated certificates implement the local security policy. The access control checking performed by the location system remains the same.

The users of the web server are supposed to trust the web server. Users are free to implement their own frontend if they do not trust the web server provided by an organization. The location system does not need to trust the web server, because the web server does not perform any access control decisions on behalf of the location system. Not having a centralized trusted component that stores location policies and makes access control decisions is a key advantage of our system.

10. EVALUATION

In this section, we quantify the influence of access control on query processing time. In addition, we examine the influence of delegation on query processing time.

10.1 Methodology

For our measurements, we use a subset of the implemented location system. Namely, we have a client query the People Locator service, which then contacts the Oracle CorporateTime-Based Calendar service. We measure the time it takes for a query to finish and report the mean, μ, and the standard deviation, σ, computed over 100 queries. In addition to identifying bottlenecks, we measure the duration of several individual steps in query processing. Ten nonmeasured queries precede each run of an experiment so that Java can load any required class files, compile them to native code, and optimize this code.

The client, the People Locator service, and the Calendar service run on an unloaded Pentium IV/2.5 GHz with 1.5 GB of memory, Linux 2.4.20, and Java

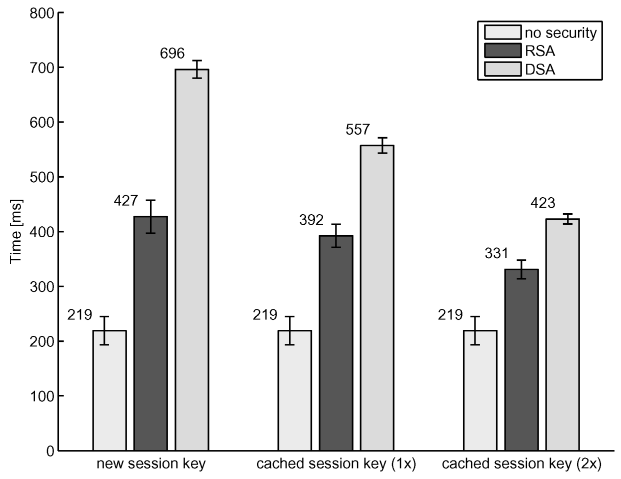


Fig. 5. Query processing time if a new session key is created for each connection, if the People Locator service and the Calendar service cache their session key, and if the client and the People Locator service also cache their cache session key.

1.4.2. The RSA and DSA keys have a size of 1024 bits. The Diffie-Hellman key exchange uses the same parameters as SSH2 [Ylonen 2003]. The policy certificates used for access control look similar to the first certificate shown in Section 9.1, that is, they grant full access to a user's location information and do not list any location- or time-based constraints. Similarly, the trust certificates do not have any such constraints. The client has all the required policy certificates and submits them with a request. Similarly, the People Locator service has all the required trust certificates and submits them with a forwarded request.

10.2 Cost of Access Control

We quantify the influence of access control on query processing time by examining the case where the client is directly authorized by the queried user. We assume that the Calendar service does not delegate access control to the People Locator service. The "new session" key bars in Figure 5 summarize our findings. The overhead introduced by access control is about 208 ms for RSA and 477 ms for DSA.

In Table I, we give a breakdown of the time consumed by various RSA and DSA operations. For comparison, we give also the time required by the Calendar service for retrieving the actual location information, which is the most expensive operation. In terms of access control, setting up a secure connection between two peers, which takes about 54 ms for RSA and about 171 ms for DSA, is the most expensive operation. This step is more expensive for DSA, since both peers also need to compute a session key based on Diffie–Hellman, which takes about 54 ms for a peer. There are two secure connections; first the client connects to the People Locator service, then the People Locator service connects to the Calendar service. Techniques like session key caching or persistent connections can significantly reduce the overhead of setting up these

Table I. Breakdown of Performance Impact for RSA and DSA [ms]

		RSA		DSA	
Entity	Operation	μ	(σ)	μ	(σ)
Client	Signing of request	17	(2)	10	(1)
	Connection setup	54	(2)	171	(3)
People Locator	Signature verification	2	(1)	1 9	(0)
	Policy proof	5	(1)	23	(2)
	Connection setup	55	(2)	173	(2)
Calendar	Signature verification	2	(0)	19	(1)
	Policy proof	5	(2)	22	(2)
	Trust proof	6	(2)	24	(2)
	Gather location	201	(7)	201	(6)
	Total	427	(30)	696	16

connections. In session key caching, a connection reuses a session key negotiated for an earlier connection. With persistent connections, a single connection is used for multiple requests. These techniques make most sense for connections between peers that often exchange location information. The connection between the People Locator service and the Calendar service is a good example since the two services need to establish a connection for every query. Clients such as a tracking service that connect often to the People Locator service can also apply the mentioned techniques. In Figure 5, we also present the average query time if only the session key between the People Locator service and the Calendar service is cached ["cached session key (1x)"] and for the case where both this and the session key between the client and the People Locator service are cached ["cached session key (2x)"]. For RSA, caching improves the performance by 8 and 22%. In the case of DSA, the improvement is 20 and 39%.

A secure connection serves two purposes: First, it keeps data exchanged between two peers confidential. Second, it authenticates the two peers to each other. We claim that the overhead caused by setting up a secure connection is not unique to our access control algorithm. All access control algorithms require data confidentiality and many require also client authentication in order to make an access control decision. Strictly speaking, access control does not require authentication of a service to a client. However, for practical scenarios, this authentication is highly recommended in order to avoid man-in-the-middle attacks. It is possible to avoid client authentication for some access control algorithms. In particular, for encryption-based algorithms, a service provides sensitive information to any client, but only in an encrypted form. Only clients authorized to access the information have the required decryption key. We examine this approach in related work [Hengartner and Steenkiste 2005].

Compared to the cost of setting up a secure connection, the costs more directly related to access control are small. RSA-based signature generation takes about 17 ms and is more expensive than its validation. For DSA, generating a signature is less expensive than its verification, which takes about 19 ms. The generation of DSA-based signatures is 41% less expensive than RSA-based signatures. For environments with resource-limited clients, this finding suggests using DSA instead of RSA for signing operations (together with session-key caching in order to avoid the expensive Diffie–Hellman key exchange for

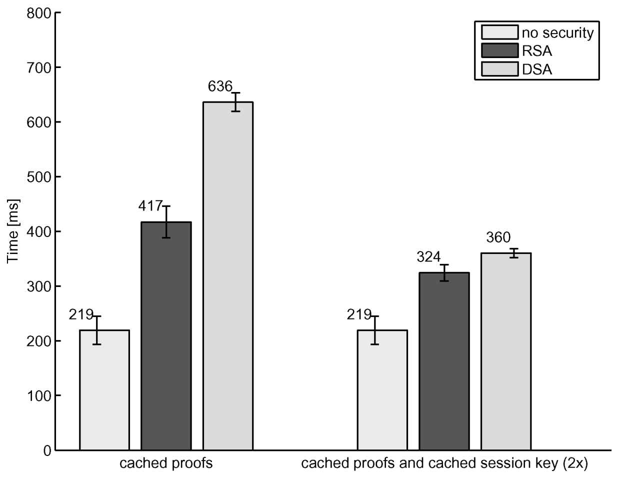


Fig. 6. Query processing time if the People Locator service and the Calendar service cache proofs and if they cache both proofs and session keys.

multiple requests). The drawback of using DSA for signing is that more load is pushed to the service. Namely, verifying a DSA signature is 9.5 times as expensive as validating an RSA signature.

Proving that an entity is granted access or that a service is trusted includes verifying the signatures of any certificates needed for the proof. Therefore, the cost for proving is similar to the cost for verifying a signature in the experiments. The cost for proving can be reduced by caching proofs. Caching is most beneficial when a client issues many queries for a queried person, as in the case of a tracking service. Figure 6 provides the average query times when both the People Locator service and the Calendar service cache proofs. The figure also presents the results for caching session keys in addition to proof caching. For DSA, performance improves by 22% as compared to the noncaching case. For RSA, the improvement is 43%. Compared to Figure 5, proof caching is of limited use for RSA, since its cost for proving is small in the first place.

10.3 Influence of Delegation

In the next set of experiments, we explore the influence of delegation on the query-processing time. Namely, we assume that the Calendar service delegates access control to the People Locator service. Upon granting access to a query by validating the certificate chain, the People Locator service issues a new certificate that directly grants access to the client at the end of the chain.

Shortening a certificate chain is useful only if there is at least one intermediate node between the client and the queried user. Without delegation, the Calendar service would have to check at least two certificates. (The decision of the Calendar service to authorize either the People Locator service or the queried user is stored locally at the Calendar service and does not require a certificate.) With the help of delegation, the Calendar service looks only at the certificate issued by the People Locator service. Figure 7 compares the cost of

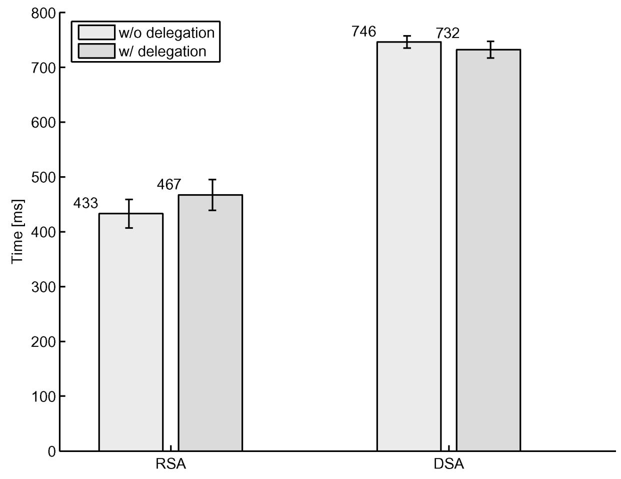


Fig. 7. Query processing time without and with delegation of the access control check by the Calendar service to the People Locator service.

query processing with and without delegation when there is exactly one entity in the certificate chain between the queried user and the client. Since there is an additional entity in the chain, the cost is higher than in Figure 5.

The figure shows that in the case of RSA, delegation has a detrimental effect on performance, whereas in the case of DSA, performance slightly improves. The reason for this behavior is the high cost of generating a signature for the new certificate by the People Locator service in the case of RSA. For DSA, the overall signature generation and checking cost decreases when using delegation, however, the decrease nearly disappears in the noise caused by the other operations.

From Figure 7, one might conclude that delegation does not help in terms of performance for the case of RSA and has only a minor influence in the case of DSA. However, this conclusion is wrong. First, our experiments assume that all the services run on hosts with similar hardware resources. However, the location services not at the root of the system could be deployed on resource-constrained machines or it might not be possible to change a proprietary location service to make it check a certificate chain. In cases like these, taking load away from leaf location services and delegating the policy check to the root location service can pay off. Second, the cost of running a policy check actually depends on the length of the certificate chain that is checked and on the depth of the tree of contacted services. We now elaborate on this trade-off for both RSA and DSA.

In the case of RSA, generating a signature takes about 17 ms and checking it about 2 ms. We first look at the influence of the length of the certificate chain on performance. We assume a service depth of size 2, as it is the case in our experiment. For a certificate chain of length l, the total cost for checking the signatures is $2 * l * 2$ ms if no delegation is used. If there is delegation, the service at the first level generates a new certificate. The services at the second level now need to check only one certificate. Thus, the total cost becomes l *

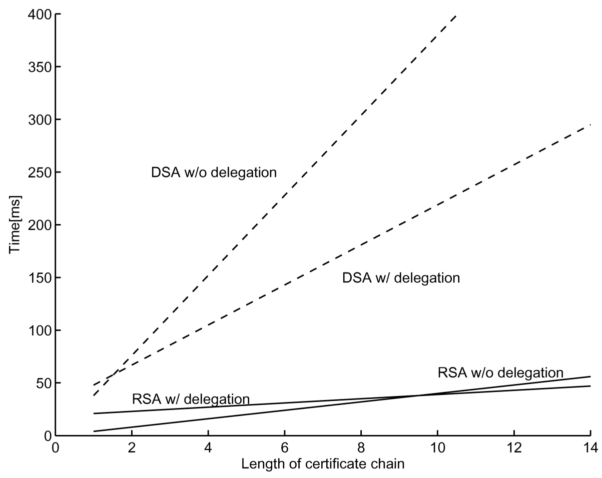


Fig. 8. Overall signature generation and verification cost endured by services, assuming a service depth of size 2, depending on length of certificate chain.

2 ms + 17 ms + 2 ms. Figure 8 presents the cost of the two cases, as l increases. Note that the shown cost does not include the client's cost for signing a request. We conclude that for certificate chain lengths of at least $l = 10$, delegation also pays off in the case of RSA.

Certificate chains of size 10 are probably going to be rare in real environments. However, the cost of security also depends on the depth of the tree of contacted services, k. When taking this value into account, we get a cost of $k * l * 2$ ms for the no delegation case and a cost of $l * 2$ ms + 17 ms + $(k - 1) * 2$ ms for the delegation case. This equation assumes that all the services delegate access control to the People Locator service. Therefore, if $l = 5$, delegation pays off if the tree of services has a depth of at least 4.

For DSA, the cost for verifying a certificate is greater than the cost for generating it. Therefore, as shown in Figure 8, delegation always pays off for $l > 1$ and becomes more beneficial the longer the certificate chain or the service tree depth. Creating a signature and checking it take about 10 and 19 ms, respectively. Modifying the given formulas for these parameters and setting k and l to the values where delegation starts to become beneficial for RSA ($k = 4$ and $l = 5$), we conclude that for DSA delegation improves performance by 218 ms.

Our calculations are on the conservative side, since delegation makes query processing faster than indicated. In our calculations, we do not include the cost caused by transferring certificates between nodes, which includes time for (de)marshalling and network transmission. If delegation is used, fewer certificates will be transferred between nodes and, again, performance improves. Finally, it is possible to amortize the cost of generating a certificate over multiple requests. For example, the People Locator service can give the certificate it generates to the client and the client can use this certificate for future requests.

Delegation and the concepts of session key caching and persistent connections, as introduced in Section 10.2, are orthogonal to each other in terms of query-processing time. However, there can be some interaction between delegation and proof caching. Namely, the cost for generating a certificate can be amortized only if services do not cache proofs.

10.4 Discussion

From our measurements, we conclude that the delay introduced by access control, in particular by setting up secure connections, is significant. This delay affects clients of our location system. There are two types of clients: people and services. The first type of clients use our service through a web browser. For them, delays in the range of 200–500 ms are noticeable, but acceptable, since they are used to similar processing delays when using other web applications. When services access our location system, these delays may be more important. However, by using the optimization techniques discussed before, delays can be reduced significantly.

11. RELATED WORK

Several location systems, all of them based on only one location technology, implemented only within one administrative entity and/or not addressing the various access control issues mentioned in this paper have been proposed [Bahl and Padmanabhan 2000; Harter and Hopper 1994; Priyantha et al. 2000; Ward et al. 1997]. We discuss three notable exceptions: In Myles et al.'s location system [2003], a middleware service runs access control on behalf of individual location services. Such a centralized solution has limited scaling properties. Our distributed solution allows us to offload some of the access-control load to clients. (Distributed access-control architectures have also been proposed for other applications [Bauer et al. 2002; Howell and Kotz 2000b].) In Spreitzer and Theimer's location system [1993], each user has her personal agent that gathers location information about her and that implements access control to this information. While avoiding a centralized bottleneck, this solution has the drawback that it puts the load of gathering any statements required for the access control decision on these agents. This task can be expensive, especially when access rights are forwarded. The system is designed to work in an environment with different administrative entities, although the actual implementation runs only within a single entity, and the authors do not mention how users specify services that they trust. Unlike our system, the location policy of a user is always specified by the user, and the system does not have the flexibility offered by digital certificates. Leonhardt and Magee's system [1998] also suffers from this weakness. It relies on policy matric which make dealing with unknown users and groups tedious.

Several frameworks for pervasive computing environments support access control to sensitive information [Al-Muhtadi et al. 2003; Chen et al. 2004; Covington et al. 2002; Gandon and Sadeh 2003]. They employ centralized rule engines with differing degrees of flexibility for running access control. This engine can become a bottleneck and needs to be fully trusted by all the entities. The authors do not examine trust in services that are run by different administrative entities.

Multiple specification languages have been used for expressing access rights to information in pervasive computing. For example, Myles et al. [2003] use an extended version of P3P [Cranor et al. 2002], which allows Web servers to express their privacy practices to control access to location information. Chen et al. [2004] exploit REI [Kagal et al. 2004], which is targeted at pervasive computing environments. XACML [Godik and Moses 2003] is an access control language for distributed systems. These languages are targeted at environments where access control is run by a single entity. As opposed to SPKI/SDSI, they have no built-in mechanisms for verifying the authenticity of a statement, which is essential when delegating access rights across multiple environment. Similar to SPKI/SDSI, KeyNote [Blaze et al. 1999] is a certificate-based framework for authorizing entities. KeyNote certificates do not support local names and access rights are always forwardable. The RT_0 trust management language [Li et al. 2003b] is similar to SDSI certificates, which we use for expressing membership certificates. We use SPKI/SDSI because it is standardized and because implementations are available for different platforms.

There has been some earlier work on authorizing intermediate services, for example, Howell's quoting gateways [2000b], Neuman's proxy-based authorization [1993], and Sollins' cascaded authentication [1988]. All this work focuses on intermediate services that create new requests upon receiving a request and thus need to be authorized to issue requests. However, in scenarios like our location system, where some services only forward requests, this model gives too many capabilities to intermediate services. It presents an unnecessary risk if the intermediate service is broken into. Using our model of trust, we avoid this risk and clearly define which services should be given location information and which services should be allowed to issue requests.

Covington et al. [2001] enhance role-based access control by "environment roles." Environment roles can describe any state of the system, such as locations or times and can be used to implement controllable properties, as outlined in Section 3.1. McDaniel [2003] presents a framework for the specification and instantiation of flexible controllable properties. A service running access control can contact a remote host for the evaluation of a property. In our system, we currently support only properties local to the service running access control.

In our work, we do not examine how users decide which services to trust. Acquiring trust has been addressed in related work. Shand et al. [2003] introduce a trust framework in which individuals compute their trust in information by combining their own trust assumptions with others' recommendations. Bertino et al. [2003] present a trust negotiation framework, which allows entities to establish mutual trust on first contact through an exchange of digital credentials.

Tamassia et al. [2004] propose a model of delegation in which entities delegate access rights by delegating the role membership that is required for being granted access. For example, if Alice is a professor and wants Bob to have access to some information accessible only to professors, she can delegate this role to Bob. The drawback of this model is that it does not allow for fine-grained delegation, where only a particular access right should be forwarded, but not all the other access rights granted to members of a role. Similar to our model, when delegating a role membership by issuing a certificate, an entity must also hand over all the certificates that prove that the entity is a member of this role. As explained in Section 7.3, this model breaks down when a membership certificate is created before an access right is being delegated. Tamassia et al. [2004] discuss the simple case where only one such membership certificate is necessary for being granted access. Kaminsky et al. [2003] present a distributed discovery algorithm for membership certificates that exploits discovering all the members of a particular role. As mentioned in Section 7.3, this approach requires that institutions provide public listings of their members, which might not hold in practice. Li et al. [2003b]'s distributed discovery algorithm exploits both discovering all the members of a particular role and discovering all the roles of a particular entity. In our distributed algorithm, we avoid the first kind of search because of the given problem. This restriction requires that a membership certificate is stored with the entity denoted on the right-hand side of the certificate. Li et al. argue that this assumption might not hold when a membership decision is not of interest to the entity on the right-hand side. However, we do not expect this to be an issue for the application scenarios of memberships certificates that we consider for our people location system. Li et al.'s algorithm operates in an environment consisting only of membership certificates, whereas our algorithm supports separate policy (trust) and membership certificates. In addition, Li et al. assume that certificates are publicly available, which, as discussed in Section 7.3, might not hold due to privacy reasons. Finally, Li et al. do not discuss certificate expiration and renewal.

12. CONCLUSIONS

In this paper, we have analyzed the access control requirements of a people location system and have presented the design of an access control mechanism. Our solution relies on several key concepts: services implementing location policy checks, service trust for dealing with services belonging to different administrative entities, and delegating various decisions to other entities in the system. Some advantages of our design are:

1. *Flexibility.* Depending on the environment, users themselves or a central authority can establish location policies of users.
2. *No bottleneck or trusted centralized node.* The services in the system run the location policy and trust checks by building chains of certificates. Certificates do not need to be kept at a centralized trusted node and bottlenecks are avoided.
3. *Unknown users.* The identity of entities issuing queries does not have to be known to the system. All the system requires is a digital certificate-granting access.
4. *Group access.* With a single certificate, an entire group of entities can be given access to location information.
5. *Delegation.* Access control can be delegated to other services

We have formulated all of our policy and trust decisions using a single data structure: SPKI/SDSI certificates. These certificates provide a high degree of flexibility. A ubiquitous computing environment poses new challenges on access control that cannot be easily satisfied by conventional mechanisms. We believe that, due to their flexibility, SPKI/SDSI certificates are a promising approach.

In future work, we will offer access to our location system to a bigger community of users, so that we can incorporate their feedback on usability into our system. In addition, we plan to investigate whether and how the ideas outlined in this paper can be applied to protect other kinds of information available in a ubiquitous computing environment. Finally, another area of future work is to perform a formal security analysis, as proposed by Li et al. [2003a], of our access control mechanisms.

REFERENCES

AL-MUHTADI, J., RANGANATHAN, A., CAMPBELL, R., AND MICKUNAS, M. D. 2003. Cerberus: A context-aware security scheme for smart spaces. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*. 489–496.

BAHL, P. AND PADMANABHAN, V. 2000. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of IEEE Infocom 2000*. 775–784.

BAUER, L., SCHNEIDER, M. A., AND FELTEN, E. W. 2002. A general and flexible access-control system for the Web. In *Proceedings of 11th Usenix Security Symposium*. 93–108.

BERTINO, E., BONATTI, P. A., AND FERRARI, E. 2001. TRBAC: A temporal role-based access control model. *ACM Trans. Informat. Syst. Secur. 4*, 3 (Aug.), 191–233.

BERTINO, E., FERRARI, E., AND SQUICCIARINI, A. C. 2003. Trust-χ: An XML framework for trust negotations. In *Proceedings of Communications and Multimedia Security 2003*. 146–157.

BHATTI, R. 2003. X-GTRBAC: An XML-based policy specification framework and architecture for enterprise-wide access control. Tech. Rep. 2003-27, CERIAS, Purdue University.

BLAZE, M., IOANNIDIS, J., AND KEROMYTIS, A. 1999. The KeyNote trust-management system version 2. RFC 2704.

CHAUM, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonym. *Communications of the ACM 24*, 2 (Feb.), 84–88.

CHEN, H., FININ, T., AND JOSHI, A. 2004. Semantic Web in the context Broker architecture. In *Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*. 277–286.

CLARKE, D., ELIEN, J.-E., FREDETTE, M., MORCOS, A., AND RIVEST, R. L. 2001. Certificate chain discovery in SPKI/SDSI. *J. Comput. Secur. 9*, 4, 285–322.

COVINGTON, M. J., FOGLA, P., ZHAN, Z., AND AHAMAD, M. 2002. A context-aware security architecture for emerging applications. In *Proceedings of 18th Annual Computer Security Applications Conference (ACSAC 2002)*.

COVINGTON, M. J., LONG, W., SRINIVASAN, S., DEY, A., AHAMAD, M., AND ABOWD, G. 2001. Securing context-aware applications using environment roles. In *Proceedings of 6th ACM Symposium on Access Control Models and Technologies (SACMAT '01)*. 10–20.

CRANOR, L., LANGHEINRICH, M., MARCHIORI, M., PRESLER-MARSHALL, M., AND REAGLE, J. 2002. The platform for privacy preferences 1.0 (P3P1.0) specification. W3C Recommendation.

DAY, M., AGGARWAL, S., MOHR, G., AND VINCENT, J. 2000. Instant messaging/presence protocol requirements. RFC 2779.

DIFFIE, W., VAN OORSCHOT, P., AND WIENER, M. 1992. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography 2*, 107–125.

ELLISON, C., FRANTZ, B., LAMPSON, B., RIVEST, R., THOMAS, B., AND YLONEN, T. 1999. SPKI certificate theory. RFC 2693.

GANDON, F. AND SADEH, N. 2003. A semantic eWallet to reconcile privacy and context awareness. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*.

GARLAN, D., SIEWIOREK, D., SMAILAGIC, A., AND STEENKISTE, P. 2002. Project Aura: Towards distraction-free pervasive computing. *IEEE Pervasive Computing 1*, 2 (Apr.–June), 22–31.

GODIK, S. AND MOSES, T. 2003. eXtensible access control markup language (XACML) version 1.0. OASIS Standard.

HALPERN, J. Y. AND VAN DER MEYDEN, R. 2001. A logical reconstruction of SPKI. In *Proceedings of 14th IEEE Computer Security Foundations Workshop (CSFW-14)*. 59–70.

HARTER, A. AND HOPPER, A. 1994. A distributed location system for the active office. *IEEE Network 8*, 1 (Jan.), 62–70.

HENGARTNER, U. AND STEENKISTE, P. 2004. Implementing access control to people location information. In *Proceedings of 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*. 11–20.

HENGARTNER, U. AND STEENKISTE, P. 2005. Exploiting hierarchical identity-based encryption for access control to pervasive computing information. In *Proceedings of First IEEE / CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks* (IEEE/CreateNet SecureComm 2005). 384–393.

HOWELL, J. AND KOTZ, D. 2000a. A formal semantics for SPKI. In *Proceedings of 6th European Symposium on Research in Computer Security (ESORICS 2000)*. 140–158.

HOWELL, J. AND KOTZ, D. 2000b. End-to-end authorization. In *Proceedings of 4th Symposium on Operating System Design & Implementation (OSDI 2000)*. 151–164.

ICAL. ftp://ftp.scriptics.com/pub/tcl/apps/ical/.

JUDD, G. AND STEENKISTE, P. 2003. Providing contextual information to ubiquitous computing applications. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*. 133–142.

KAGAL, L., FININ, T., AND JOSHI, A. 2004. A policy language for a pervasive computing environment. In *Proceedings of 4th International Workshop on Policies for Distributed Systems and Networks*. 63–76.

KAMINSKY, M., SAVVIDES, G., MAZIÈRES, D., AND KAASHOEK, M. F. 2003. Decentralized user authentication in a global file system. In *Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP'03)*. 60–73.

LEONHARDT, U. AND MAGEE, J. 1998. Security considerations for a distributed location service. *Journal Network Systems Management 6*, 1 (Mar.), 51–70.

LI, N. AND MITCHELL, J. C. 2003. Understanding SPKI/SDSI using first-order logic. In *Proceedings of 16th IEEE Computer Security Foundations Workshop (CSFW-16)*. 89–103.

LI, N., WINSBOROUGH, W. H., AND MITCHELL, J. C. 2003a. Beyond proof-of-compliance: Security analysis in trust management. In *Proceedings of 2003 IEEE Symposium on Security and Privacy*. 123–139.

LI, N., WINSBOROUGH, W. H., AND MITCHELL, J. C. 2003b. Distributed credential chain discovery in trust management. *J. Comput. Secur. 11*, 1 (Feb.), 35–86.

MCDANIEL, P. 2003. On context in authorization policy. In *Proceedings of 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*. 80–89.

MYLES, G., FRIDAY, A., AND DAVIES, N. 2003. Preserving privacy in environments with location-based applications. *Pervasive Computing 2*, 1 (Jan.-Mar.), 56–64.

NEUMAN, B. 1993. Proxy-based authorization and accounting for distributed systems. In *Proceedings of International Conference on Distributed Computing Systems*. 283–291.

ORKUT. http://www.orkut.com.

PRIYANTHA, N., CHAKRABORTY, A., AND BALAKRISHNAN, H. 2000. The cricket location-support system. In *Proceedings of 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*.

SHAND, B., DIMMOCK, N., AND BACON, J. 2003. Trust for ubiquitous, transparent collaboration. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*. 153–160.

SOLLINS, K. R. 1988. Cascaded authentication. In *Proceedings of IEEE Symposium on Security and Privacy*. 156–163.

SPREITZER, M. AND THEIMER, M. 1993. Providing location information in a ubiquitous computing environment. In *Proceedings of SIGOPS '93*. 270–283.

SYVERSON, P. F., GOLDSCHLAG, D. M., AND REED, M. G. 1997. Anonymous connections and onion routing. In *Proceedings of 1997 IEEE Symposium on Security and Privacy*. 44–54.

TAMASSIA, R., YAO, D., AND WINSBOROUGH, W. H. 2004. Role-based cascaded delegation. In *Proceedings of 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*. 146–155.

WARD, A., JONES, A., AND HOPPER, A. 1997. A new location technique for the active office. *IEEE Personal Communications 4*, 5 (Oct.), 42–47.

YLONEN, T. 2003. SSH transport layer protocol. Internet Draft.

Received September 2004; revised March 2005; accepted August 2005