Implementing Access Control to People Location Information

Urs Hengartner[†] and Peter Steenkiste[†‡]
[†]Department of Computer Science
[‡]Department of Electrical and Computer Engineering
Carnegie Mellon University
{uhengart,prs}@cs.cmu.edu

ABSTRACT

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, which 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, different administrative domains might allow different entities to specify policies, and policies need to be flexible. We address these issues in our design of an access control mechanism for a people location system. Our design encodes policies as digital certificates. We present 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 Protection - *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

Keywords

Certificates, Delegation, DSA, Location, RSA, SPKI/SDSI, Trust

1. INTRODUCTION

Ubiquitous computing environments, such as CMU's Aura [9], 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 wish to keep their location secret when walking home at night. A company might want to track the location of its repairmen to increase efficiency, but it would not want competitors to have this information. 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 studied widely. 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 administered 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 an access control mechanism that is flexible enough to be deployed in an environment that has multiple sources of location information. Our design exploits digital certificates.
- We describe an implementation of the proposed mechanism. Our implementation is based on SPKI/SDSI certificates [8].
- 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-based 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 design of our access control mechanism. In Section 6, we apply our design to an example scenario. We discuss our prototype implementation in Section 7 and evaluate it in Section 8. We comment on related work in Section 9 and on our conclusions and future work in Section 10.

2. PEOPLE LOCATION SYSTEM

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

SACMAT'04, June 2–4, 2004, Yorktown Heights, New York, USA.

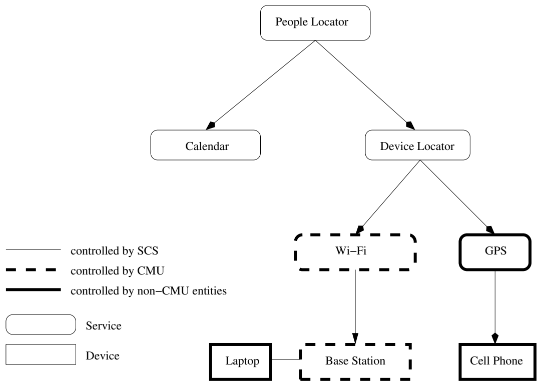


Figure 1: ***Example people location system. The People Locator service forwards a query for the location of a person to the Calendar service and to the Device Locator service. The Device Locator service locates a person by locating her devices. It queries the Wi-Fi service for the location of the person's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

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 CMU'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) 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 [2] or MIT's Cricket [19]).

A basic assumption in our work is that different organizations may administer the various services. In our example, SCS's computing facilities control the Calendar service, CMU's computing facilities maintain the Wi-Fi service, and a phone company runs the GPS service.

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 that person's location policy. In this section, we examine location policies and present requirements for the access control mechanism.

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:

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").

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.

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 concentrate on a university environment. We are also able to apply our model to environments with horizontal (e.g., military) and vertical (e.g., hospitals) access control requirements.

In the rest of the paper, we are going to call the entity that specifies location policies the *policy maker*.

3.3 Transitivity of Access Rights

If Bob is granted access to Alice's location information, should he be allowed to forward this access right to Carol? In short, should access rights to location information be transitive?

There is no simple answer to this question. Again the answer depends on the environment. For a military environment, access rights should not be transitive, whereas for a university environment, individuals defining their location policies should be able to specify whether they want access rights to be transitive. Therefore, the location system should let policy makers explicitly state whether they want access rights to be transitive. Note that even though an entity might not be allowed to forward its access rights to other entities, it could still issue queries on their behalf.

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 checks 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 the services more privileges than they really need to have. The People Locator service only forwards requests received from clients to other service providing 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.

Due to 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. 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.

The trust assumption is that the service implements access control in the following way:

1. In the first step, the service checks whether the policy maker has granted the entity that issued the request access to the requested information. Only if this check is successful, the service will proceed to the second step. Otherwise access is denied.

2. In the second step, the service 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 verify that the policy maker trusts the entity from which it received the request before access is granted. Otherwise access is denied.

How can we verify that a service fulfills its trust assumption? We require services to sign whatever location information they return to achieve non-repudiation. Therefore, an entity trusting a service can identify misbehaving services and revoke trust in them after the fact.

5. SYSTEM DESIGN

Based on our discussion in Sections 3 and 4, we now present the design of our access control mechanism for a people location system. We build on three main concepts. First, services respond to a location request only after performing a location policy check that verifies that the querying entity has access. Second, services verify that the service from which they receive a forwarded request is trusted before returning an answer. Third, services can delegate access control to other services; delegation can be used to eliminate redundant checks. In this section, we motivate and discuss these concepts. But first, we discuss digital certificates, which are a basic building block in our design.

5.1 Digital Certificates

We state policy and trust decisions in digital certificates. In the rest of this paper, we use the following notation for expressing a policy or trust decision in a certificate:

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

A is the entity making a decision concerning B. The type of decision is described above the arrow ("$policy$" or "$trust$"). In decisions of type "$policy$", A grants B access to some location information. In decisions of type "$trust$", A specifies that A trusts service B. The scope of the decision can be limited by stating the scope below the arrow (i.e., whose location policy or trusted set of services is specified). The scope can also include a list of controllable properties, as discussed in Section 3.1. However, for readability reasons, we refrain from showing them in this section.

A digital certificate is a signed data structure in which the signer states a decision concerning some other entity. Since certificates are signed, there is no need to store them in a centralized trusted database.

There are various kinds of digital certificates. Two examples are KeyNote [5] and SPKI/SDSI [8] certificates. Our implementation is based on SPKI/SDSI certificates. Four key features that we rely on are certificate chains, transitivity control, certificate reduction, and local names [1, 8]. In Appendix A, we explain these concepts in the context of a people location system. These features allow us to implement the various properties and requirements of location policies discussed in Section 3. (We present some example certificates in Section 7.1.) By exploiting the concepts of certificate chains and local names, we can give different entities the right to define location policies, as discussed in Section 3.2, and implement more complex access control strategies like vertical or horizontal access control. Transitivity control allows an entity to issue non-transitive access rights, as explained in Section 3.3. In summary, these features enable us to employ the same basic mechanisms in different environments, requiring different access control strategies.

5.2 Location Policy Check

When there is a request, a service must first check whether the location policy grants access to the issuer of the request. The service tries to build a chain of location policy certificates from itself to the issuer of the request. Each certificate in the chain needs to give the next entity further down the chain access to the requested location information. In addition, any constraints listed in the certificate (e.g., time or location based) must be fulfilled in order to have the policy check succeed.

We now present an example that illustrates policy checking in a university environment. To start with, the administrator of the People Locator service, PL, gives individual users access to their location information. For example, he issues the following certificate for user A:

$$PL \xrightarrow[A]{policy} A. \tag{1}$$

User A can define her location policy by issuing additional certificates. For example, she can grant user B access:

$$A \xrightarrow[A]{policy} B. \tag{2}$$

If B inquires about A's location, the People Locator service deduces that B has access to A's location, since it can combine certificates 1 and 2 in a chain of certificates and conclude

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

A location service does not have to be aware of the identity of the entities that have access rights. If Bob can prove that he has access (by presenting a digital certificate), he will be granted access. This approach makes dealing with unknown users easy. Solutions proposed earlier (e.g., by Leonhardt and Magee [15]) rely on policy matrices. In a policy matrix, the rows list all the entities that can query for location information, and the columns enumerate all the users whose location can be queried. The location policy for each combination is specified in the corresponding matrix entry. In such a solution, the location system needs to be aware of the identity of entities issuing queries.

Another benefit of digital certificates is that, with the help of local names, we can grant access rights both to single entities and to groups of entities. That is, with a single certificate, Alice can, for example, give all her friends access.

5.3 Service Trust Check

If a service receives a forwarded request, it must also check whether the service from which it got the request is trusted. Similar to the location policy check, the service tries to build a chain of trust certificates from itself to the forwarding service.

We now show how the Device Locator service handles trust decisions. Typically, this service is not directly contacted by users, but by other services (e.g., the People Locator service). The administrator of the Device Locator, DL, service gives user A the right to define her trusted services. User A then states that she trusts the People Locator service. The issued certificates look as follows:

$$(a) \quad DL \xrightarrow[A]{trust} A \qquad (b) \quad A \xrightarrow[A]{trust} PL. \tag{3}$$

When receiving a forwarded request from the People Locator service, the Device Locator service combines these certificates and concludes that the People Locator service is trusted.

5.4 Delegation

Certificate chains exploit the concept of delegation, where an entity grants some other 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 build the potentially long certificate chains required for access control itself. 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 build a certificate chain for each request anyway. After validating the chain, the People Locator service exploits the concept of certificate reduction and issues a new certificate that directly authorizes the querying entity. It gives this certificate to the Calendar service, which does not have to validate the entire chain again. We describe this optimization in more detail in Section 6.

5.4.2 *Delegating Service Trust*

If there are lots of services available, it is 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 administers this service. For example, a user might trust all the services run by her company. Therefore, we give users the possibility 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 about which services she trusts to the organization, and she relies on the organization to do the right thing.

For the university environment, instead of issuing certificate 3(b), user A would issue the following certificate: ($O.services$ represents the set of services run by the organization, it is based on the concept of local names)

$$A \xrightarrow[A]{trust} O.services. \tag{4}$$

For each service in the organization, the administrator of the organization issues a "membership certificate" (denoted by $\longmapsto$), for example,

$$O.services \longmapsto PL. \tag{5}$$

For the example in Section 5.3, the Device Locator service would now combine certificates 3(a), 4, and 5 to conclude that the People Locator service is trusted.

6. EXAMPLE SCENARIO

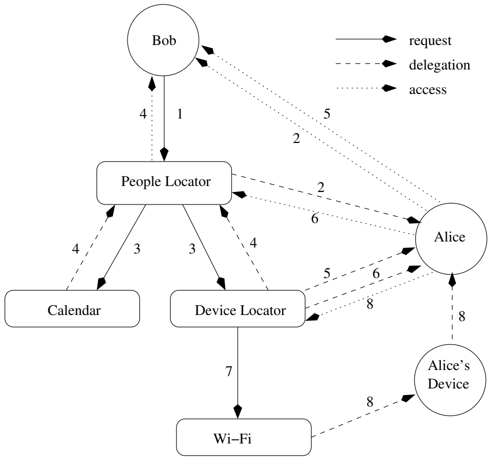
In this section, we present the step-by-step processing of a request by our location system and list the required certificates. In our example, Bob inquires about the location of Alice. The system shown in Figure 2 processes the request. We differentiate between simply giving someone access to a resource ("access" arrows) and also allowing the recipient of the access right to delegate 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 certificate that delegates the right to decide about Alice's location policy to Alice. The administrator of the Device Locator service also establishes a trust certificate saying that it is up to Alice to specify which services she trusts. The Calendar service is administrated by the same organization as the People Locator service. Therefore its administrator has the People Locator service run its access control and delegates the right to decide about Alice's location policy to the People Locator service.

- The administrator of the Wi-Fi service defines a policy certificate that lets the owner of Alice's laptop decide about the device's location policy. Alice, as the owner of her device, uses her device's private key to re-delegate this right to herself in a signed certificate.

- Alice grants Bob access in a policy certificate. She also certifies that she trusts the People Locator service. 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.



Step	Required Certificates
2	PL $\xrightarrow[Alice]{policy}$ Alice, Alice $\xrightarrow[Alice]{policy}$ Bob
4	Calendar $\xrightarrow[Alice]{policy}$ PL, PL $\xrightarrow[Alice]{policy}$ Bob
5	DL $\xrightarrow[Alice]{policy}$ Alice, Alice $\xrightarrow[Alice]{policy}$ Bob
6	DL $\xrightarrow[Alice]{trust}$ Alice, Alice $\xrightarrow[Alice]{trust}$ PL
8	Wi-Fi $\xrightarrow[Alice's\ device]{policy}$ Alice's device, Alice's device $\xrightarrow[Alice's\ device]{policy}$ Alice, Alice $\xrightarrow[Alice's\ device]{policy}$ DL

Figure 2: ***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 certificate chains that the services build and the required certificates for building the chains.***

Given this setup, the system processes a request from Bob for the location of Alice as follows: (The numbers in Figure 2 correspond to the steps given below.)

1) Bob sends a signed request to the People Locator service inquiring about Alice's location.

2) The People Locator service checks Alice's location policy by building a certificate chain. The first element in the chain is the policy certificate granting Alice access to her location information. The second element is the certificate issued by Alice to Bob.

3) The People Locator service applies certificate reduction and issues a certificate that directly grants Bob access to the requested location information. The service forwards the request to the Calendar service and to the Device Locator service.

4) The Calendar service relies on the People Locator service for running access control. It builds a certificate chain to Bob using the policy certificate issued to the People Locator service and the certificate just generated by the People Locator service. It then processes the request

5) The Device Locator service checks Alice's location policy by building a certificate chain in the same way as the People Locator service.

6) Since the Device Locator service received a forwarded request, it needs to verify that the People Locator service is trusted by Alice. Using its own and Alice's trust certificates, it builds a certificate chain to the People Locator service.

7) The Device Locator service determines Alice's wireless device. It issues a new request and sends it to the Wi-Fi service.

8) The Wi-Fi service checks the location policy of Alice's laptop and builds a certificate chain from its location policy certificate to the Device Locator. 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. PROTOTYPE IMPLEMENTATION

In this section, we illustrate how we encode the policy and trust decisions introduced in Section 5 as SPKI/SDSI certificates [8]. In addition, we present a prototype implementation of a people location system.

7.1 Application of SPKI/SDSI Certificates

Authentication and authorization based on SPKI/SDSI certificates [8] do not rely on the existence of a global naming structure as, for example, the one introduced for ISO's X.509 certificates. Instead, the authentication and authorization step are merged, and a certificate gives access rights directly to a public key. There are tools that evaluate chains of SPKI/SDSI certificates and decide whether requests should be granted access.

We demonstrate how certificates 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 the first certificate, 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 delegation 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 the second certificate, Alice gives Bob access to her location information.

```
(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. In this certificate, in addition to specifying the type and scope of the certificate, the tag section also includes a list of controllable properties. Bob can locate Alice only if she is either in Wean Hall or in Room 1234 in Doherty Hall and on Monday between 8am and 12pm and on Tuesday between 1pm and 2pm. We currently support only temporal constraints of the given form. If necessary, we can extend our scheme to support more sophisticated constraints, for example, the ones suggested by Bertino et al. [3]. 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. This might be difficult to enforce in heterogeneous environments. Similarly, extending the given fixed set of constraints to support very general constraints in location policies is difficult to achieve using the current tag mechanism.

7.2 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 [12] 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 achieve authentication of peers and confidentiality and integrity of transmitted information by using an SSH-like transport protocol for securing communication between entities. The protocol has been implemented in previous work [11] and uses RSA public/private key pairs for authentication and session key establishment. We have extended the protocol to alternatively use DSA public/private key pairs [18] for authentication, together with a Diffie-Hellman session key exchange secured against man-in-the-middle attacks [7].

Access control to services is based entirely on SPKI/SDSI certificates. Issuers of requests need to sign their requests so that the issuer can be identified. To avoid replay attacks, we have the signature of a request also cover a timestamp. To transmit requests for location information, actual location information, and SPKI/SDSI certificates between services, we employ the Aura Contextual Service Interface [13], which is a protocol running over HTTP and which exchanges messages encoded in XML. Our SPKI/SDSI implementation has been implemented in Java and is based on previous work [11]. The implementation also provides a tool for building and verifying chains of certificates.

We have built a web server-based frontend to the location system that lets users 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 just 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.

8. 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.

8.1 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 Ical-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 ten queries. In addition, to identify bottlenecks, we measure the duration of several individual steps in query processing. A non-measured query precedes each run of an experiment so that Java can load any required class files and compile them to native code.

For some of the measurements, the coefficients of variation are large ($> 100\%$). We have found that repeating the experiments with a larger number of runs does not have an influence on the amount of variation. Most of the observed variation is due to artifacts of Java, such as garbage collection, and thus not under our control. However, despite the observed variation, the numbers reported here allow us to reliably quantify the cost of access control.

The client, the People Locator service, and the Calendar service each run on an unloaded host. The hosts are Pentium III/733 with 256 MB of memory. They run Linux 2.4.17 and Sun's J2SE 1.4.0. The issued certificates grant full access to a user's location information and do not list any location- or time-based constraints. All the RSA and DSA keys have a size of 1024 bit.

8.2 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 3 summarize our findings. The overhead introduced by access control is about 1290ms for RSA and 1150ms for DSA.

In Table 1, we give a breakdown of the time consumed by various security operations for RSA and DSA, respectively. The most expensive operation is setting up a secure connection between two peers, which takes about 490ms for RSA and about 400ms for DSA. This operation is performed twice; 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 reduce the overhead of setting up these connections significantly. In session key caching, an SSH 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 connec-

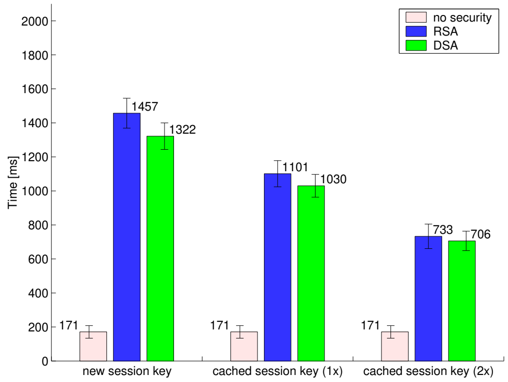


Figure 3: ***Query processing time if a new session key is created for each connection, if the People Locator service and the Calendar service cache their session key, and if the client and the People Locator service also cache their cache session key.***

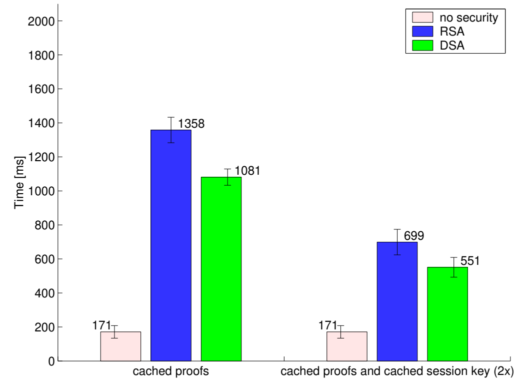


Figure 4: ***Query processing time if the People Locator service and the Calendar service cache proofs and if they cache both proofs and session keys.***

Entity	Operation	RSA		DSA	
		μ	(σ)	μ	(σ)
Client	Signing of request	180	(16)	24	(10)
	Connection setup	490	(32)	402	(27)
People Locator	Signature verification	16	(13)	43	(3)
	Policy proof	25	(10)	57	(9)
	Connection setup	501	(48)	398	(11)
Calendar	Signature verification	13	(15)	41	(1)
	Policy proof	20	(9)	52	(5)
	Trust proof	18	(5)	64	(33)

Table 1: ***Breakdown of performance impact for RSA and DSA [ms].***

tions between peers that exchange location information often. The connection between the People Locator service and the Calendar service is such a case 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 3, 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 24% and 50%. In the case of DSA, the improvement is 22% and 47%.

Another expensive operation is RSA-based signature generation, which takes about 180ms. For DSA, generating a signature is less expensive than its verification, which takes about 40ms. The generation of DSA-based signatures is 87% less expensive than RSA-based signatures. For environments with resource-limited clients, this finding suggests using DSA instead of RSA for signing operations. The drawback of using DSA for signing is that more load is pushed to the service. Namely, verifying a DSA signature is 2.5 times as expensive as validating an RSA signature. The expensive RSA signing operation is the main reason why queries using RSA key pairs tend to be slower than queries using DSA key pairs (as shown in Figure 3 'new session key'). When we omit the signing operation for setting up the secure connection and instead cache session keys, the performance of the queries becomes similar (as shown by the 'cached session key (2x)' bars).

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 significantly 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 4 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 58% as compared to the non-caching case. For RSA, the improvement is 52%.

8.3 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 5 compares the cost of 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 3.

The figure shows that in the case of RSA, delegation has a detrimental effect on performance, whereas in the case of DSA, performance stays about the same. 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 does decrease when using delegation, however, the decrease tends to disappear in the noise caused by the other operations.

From Figure 5, 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 lo-

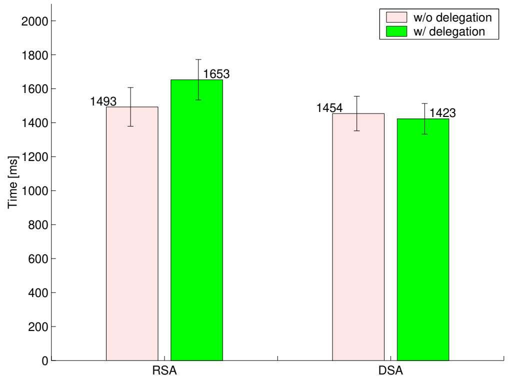


Figure 5: ***Query processing time without and with delegation of the access control check by the Calendar service to the People Locator service.***

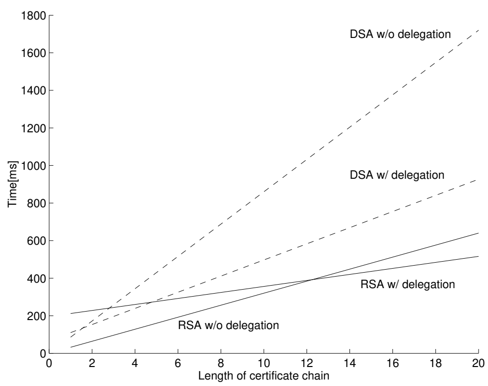


Figure 6: ***Overall signature generation and verification cost endured by services, assuming a service depth of size 2, depending on length of certificate chain.***

cation 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 is more likely to 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 180ms and checking it about 16ms. 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 * 16ms 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 * 16ms + 180ms + 16ms. Figure 6 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 $l = 13$, delegation also pays off in the case of RSA.

Certificate chains of size 13 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 * 16ms for the no delegation case and a cost of l * 16ms + 180ms + (k - 1) * 16ms 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 6, 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 24ms and 43ms, 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 480ms.

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 8.2, 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.

8.4 Discussion

From our measurements, we conclude that the delay introduced by access control is significant. We now elaborate on how 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, a delay in the range of 1s is noticeable, but acceptable, since they are used to similar processing delays when using other web applications. When services access our location system, these delays become more important. However, using the optimization techniques discussed before, delays can be reduced significantly.

Whereas the absolute cost for security tends to be high, it is important to consider its relative cost, that is, its cost compared to the cost required for a service to acquire location information. For the Calendar service based on Ical, this step is cheap and takes about 100ms. However, for other services, this step becomes much more expensive. For example, it takes our second Calendar service, which proxies to a centralized calendar system using the HTTP protocol, about 1700ms to acquire location information.

9. 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 [2, 10, 19, 23]. We discuss two notable exceptions:

Spreitzer and Theimer's location system [22] is based on multiple technologies. Each user has her personal agent that gathers location information about her and that implements access control to this information. 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 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 [15] also suffers from this weakness. It relies on policy matrices, which make dealing with unknown users and groups tedious.

KeyNote [5] is another certificate-based framework for authorizing entities. Similar to SPKI/SDSI certificates, KeyNote certificates directly authorize public keys. KeyNote certificates do not support local names, and access rights are always transitive.

There has been some earlier work on authorizing intermediate services, for example, Howell's quoting gateways [11], Neuman's proxy-based authorization [17], and Sollins' cascaded authentication [21]. 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.

Kagal et al. [14] propose an extended role-based access control model for ubiquitous computing. The model also supports delegation of access rights. For access control, the proposed system relies on a centralized trusted entity running a Prolog knowledge base. The authors do not examine trust in services that are run by different administrative entities.

Covington et al. [6] enhance role-based access control by "environment roles". Environment roles can describe any state of the system, such as locations or times. Environment roles can be used to implement controllable properties, as outlined in Section 3.1. The authors show that environment roles are constraint to similar issues as traditional subject roles in role-based access control, such as role activation and revocation, role hierarchies, and separation of duty considerations.

McDaniel [16] 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. [20] introduce a trust framework in which individuals compute their trust in information by combining their own trust assumptions with others' recommendations. Bertino et al. [4] present a trust negotiation framework, which allows entities to establish mutual trust on first contact through an exchange of digital credentials.

10. 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 delegation for delegating various decisions to other entities in the system. Some advantages of our design are:

Flexibility. Depending on the environment, users themselves or a central authority can establish location policies of users.

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.

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.

Group access. With a single certificate, an entire group of entities can be given access to location information.

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.

Acknowledgments

We thank Adrian Perrig and the anonymous reviewers for their comments. This research was funded in part by DARPA under contract number N66001-99-2-8918 and by NSF under award number CCR-0205266.

11. REFERENCES

[1] T. Aura and C. Ellison. Privacy and Accountability in Certificate Systems. Technical Report A61, Laboratory for Theoretical Computer Science, Helsinki University of Technology, April 2000.

[2] P. Bahl and V. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proceedings of IEEE Infocom 2000*, pages 775–784, March 2000.

[3] E. Bertino, P.A. Bonatti, and E. Ferrari. TRBAC: A Temporal Role-based Access Control Model. *ACM Transactions on Information and System Security*, 4(3):191–233, August 2001.

[4] E. Bertino, E. Ferrari, and A. C. Squicciarini. Trust-χ: An XML Framework for Trust Negotations. In *Proceedings of Communications and Multimedia Security 2003*, pages 146–157, 2003.

[5] M. Blaze, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System Version 2. RFC 2704, September 1999.

[6] M. J. Covington, W. Long, S. Srinivasan, A. Dey, M. Ahamad, and G. Abowd. Securing Context-Aware Applications Using Environment Roles. In *Proceedings of 6th ACM Symposium on Access Control Models and Technologies (SACMAT '01)*, pages 10–20, May 2001.

[7] W. Diffie, P.C. van Oorschot, and M.J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[8] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, September 1999.

[9] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2):22–31, April-June 2002.

[10] A. Harter and A. Hopper. A Distributed Location System for the Active Office. *IEEE Network*, 8(1):62–70, January 1994.
[11] J. Howell and D. Kotz. End-to-end authorization. In *Proceedings of 4th Symposium on Operating System Design & Implementation (OSDI 2000)*, pages 151–164, October 2000.
[12] ftp://ftp.scriptics.com/pub/tcl/apps/ical/.
[13] G. Judd and P. Steenkiste. Providing Contextual Information to Ubiquitous Computing Applications. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 133–142, March 2003.
[14] T. Kagal, L. Finin and A. Josh. Trust-Based Security in Pervasive Computing Environments. *IEEE Computer*, pages 154–157, December 2001.
[15] U. Leonhardt and J. Magee. Security Considerations for a Distributed Location Service. *Journal of Network and Systems Management*, 6(1):51–70, March 1998.
[16] P. McDaniel. On Context in Authorization Policy. In *Proceedings of 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, pages 80–89, June 2003.
[17] B.C. Neuman. Proxy-Based Authorization and Accounting for Distributed Systems. In *Proceedings of International Conference on Distributed Computing Systems*, pages 283–291, May 1993.
[18] National Institute of Standards and NIST FIPS PUB 186 Technology. Digital Signature Standard. U.S. Department of Commerce, May 1994.
[19] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, August 2000.
[20] B. Shand, N. Dimmock, and J. Bacon. Trust for Ubiquitous, Transparent Collaboration. In *Proccedings of IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 153–160, March 2003.
[21] K. R. Sollins. Cascaded Authentication. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 156–163, May 1988.
[22] M. Spreitzer and M. Theimer. Providing Location Information in a Ubiquitous Computing Environment. In *Proceedings of SIGOPS '93*, pages 270–283, Dec 1993.
[23] A. Ward, A. Jones, and A. Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

APPENDIX

A. SPKI/SDSI CONCEPTS

In this section, we explain the SPKI/SDSI concepts of certificate chains, transitivity control, certificate reduction, and local names [1, 8] in the context of a people location system.

A.1 Certificate Chains / Transitivity Control

Assume the People Locator service grants a person access to her location information by issuing a certificate to her. In the certificate, the People Locator service also states that the access right is transitive and that the person is allowed to issue additional certificates to other entities. In this way, the People Locator service effectively delegates its right to decide about the person's location policy to the person. In turn, the person can also delegate the decision to some other entity. What we end up with is a chain of certificates starting at the People Locator service. If the People Locator service receives a request for location information, it will try to build a certificate chain from itself to the entity making the request. Each certificate in the chain (with the exception of the last one) needs to delegate the access right for the requested information to the next entity in the chain. An example certificate chain for Alice's location information is given in Figure 7. When Dave wants to access Alice's location information, the People Locator service tries to build this certificate chain to Dave.

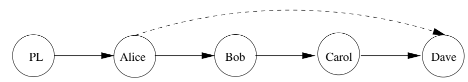


Figure 7: ***Certificate chain and certificate reduction. Certificates are combined to a certificate chain. After validating the chain between Alice and Dave, Alice can reduce it by issuing a certificate to Dave. ('PL' denotes the People Locator service.)***

A.2 Certificate Reduction

Sometimes, it is difficult for a service to build a certificate chain. For example, it may have scarce computing resources. A feature that comes in useful in such a situation is certificate reduction. In the example shown in Figure 7, Alice tries to build the chain from herself to Dave. If she succeeds, she will shorten the chain and issue a new certificate that directly authorizes Dave. The People Locator service then needs to check only the certificate chain People Locator service-Alice-Dave.

A.3 Local Names

The entity that is granted a right by a SPKI/SDSI certificate is typically a public key, but it can also be a name rooted at a public key. For example, assume Bob issues a digital certificate for each of his friends that binds his friend's public key to the name 'friend'. Bob thus creates a local namespace that is rooted at his public key. If Alice were to give all of Bob's friends access to her location information, she would issue a single certificate that grants access to the name 'friend' rooted at Bob's public key, instead of having to create a certificate for each of Bob's friends.