Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pervasive and Mobile Computing 2 (2006) 344–367
pervasive and mobile computing
www.elsevier.com/locate/pmc

Exploiting information relationships for access control in pervasive computing

Urs Hengartner[a,*], Peter Steenkiste[b]

[a] *David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada*
[b] *Departments of Computer Science and Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh PA, USA*

Received 2 May 2005; received in revised form 27 April 2006; accepted 15 May 2006
Available online 7 July 2006

Abstract

Many information services in pervasive computing offer *rich* information, which is information that includes other types of information. For example, the information listed in a person's calendar entry can reveal information about the person's location or activity. To avoid rich information from leaking its included information, we must consider the semantics of the rich information when controlling access to this information. Other approaches that reason about the semantics of information (e.g., based on Semantic Web rule engines) are based on a centralized design, whose drawback is a single point of failure. In this paper, we exploit *information relationships* for capturing the semantics of information. We identify three types of information relationships that are common and important in pervasive computing and integrate support for them in a distributed, certificate-based access control architecture. In the architecture, individuals can either define their own information relationships or refer to relationships defined by a standardization organization. In our approach, access control is fully distributed while sophisticated rule engines can still be used to deal with more complex access control cases. To demonstrate the feasibility of our design, we give a complexity analysis of the architecture and a performance analysis of a prototype implementation.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Distributed access control; Information relationships; Semantics

* Corresponding author.
E-mail addresses: uhengart@cs.uwaterloo.ca (U. Hengartner), prs@cs.cmu.edu (P. Steenkiste).

1574-1192/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2006.05.001

1. Introduction

In pervasive computing, there are a multitude of information services, which provide potentially confidential information about an individual, such as her location, her personal files, her e-mail, her calendar, or her activity. Some of this information might be offered by multiple services. For example, there are multiple ways to locate a person (see Fig. 1) or to learn about her activity. In addition, a person might be a member of multiple environments over time. In order to be granted access to this confidential information, a client requires *access rights*. An individual should be able to issue access rights to her confidential information. However, having the individual issue access rights per client, per service, per environment, and per type of information is not scalable. Pervasive computing frameworks that support access control [1–6] address the first three axes by employing role-based access control, service-independent access rights, or sharing of policies across environments. In this paper, we concentrate on the fourth axis and examine ways to limit the number of types of information for which access rights need to be issued.

To achieve this goal, we exploit *relationships* between information for access control. Consider the case of Alice managing access rights to her personal information, such as her location or her activity information. In a naïve solution, whenever she wants to grant someone access to all her personal information, she has to issue a separate access right for each type of personal information. In a better solution, Alice can bundle these different types of information in a new type of information (e.g., "personal information") and grant access rights to this new type of information. When she wants to grant someone access to her personal information, she now has to issue only a single access right. By bundling information, Alice establishes information relationships (e.g., Alice's location information is related to her personal information). The access control mechanism exploits these relationships in order to derive individual access rights.

Another example demonstrating the usefulness of information relationships involves *rich* information, which is information that includes other types of information. Assume that the current entry in Carol's calendar says that she is having a meeting with Bob in her office, that is, the calendar entry reveals the location of Carol and Bob. Therefore, only people who are at least allowed to access Carol's and Bob's location should have access to the calendar entry. To implement this rule, Carol should issue an access right for the entry to someone only if he already has access to her and Bob's location information. However, this is tedious and might lead to consistency problems if Bob revoked an access right to his location information. Instead, access control should be aware of the semantics of information (e.g., calendar information contains location information) and take this semantics into account (e.g., granting access to calendar information only if there is access to location information). We can use information relationships, as introduced above, to capture the semantics of information (e.g., calendar information is related to location information).

There are several frameworks for pervasive computing that exploit knowledge representations developed for the Semantic Web and that use rule engines to reason about this knowledge [2,4]. Such an approach can exploit certain information relationships for access control, but it has the disadvantage that the rule engine is centralized. Therefore, the rule engine can become a performance bottleneck, and it is a single point of failure in case of an attack. As an alternative, there are distributed, certificate-based access control

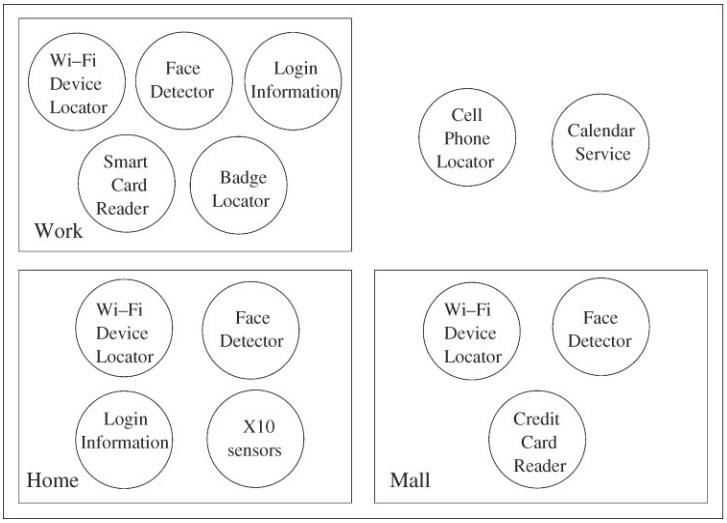

Fig. 1. Multitude of location services. There are multiple environments, each having its own set of location services.

architectures [7,8], where clients gather and reason about access rights, expressed as digital certificates, and services validate access rights received from clients. We propose making such a distributed architecture aware of information relationships that are common and important in pervasive computing. This way, we can run access control as often as possible in a fully distributed fashion. Only more complex information relationships need to be dealt with by a sophisticated rule engine.

The contributions of our work are the concept of information relationships as a first-class citizen in a distributed, certificate-based access control architecture and a formal model for incorporating relationships into access control. This paper expands on our PerCom 2005 paper [9] in that we more thoroughly discuss various information relationships (Section 2.2) and our information representation scheme (Section 3.1). We also introduce global information relationships, as defined by a standardization organization (Section 4).

We review the concept of distributed access control and introduce three information relationships that are important in pervasive computing (Section 2). With the help of a formal model of information relationships, we avoid ambiguities in access control (Section 3). While individuals can define their own information relationships, we also give them the option to exploit global relationships (Section 4). We present a distributed access control architecture where clients use information relationships (Section 5), a prototype deployment, and a measurement-based and an analytical evaluation (Section 6).

2. Access control architecture

In this section, we review a distributed, certificate-based access control architecture. We also introduce three types of information relationships that are important in pervasive computing.

2.1. Distributed access control

We want a distributed access control architecture, where access control can be run in a fully distributed way for many requests, without going through a centralized rule engine. Therefore, we have services that provide confidential information also make access decisions. Each service has an administrator who labels the provided information according to our representation scheme (see Section 3.1). To grant access to a client, a service requires that there is an access right authorizing this access. Locating these access rights can be an expensive task. To reduce load on services, we assign this task to clients. Namely, a client needs to assemble a *proof of access*, based on the client's access rights, and transmit this proof to a service, together with its request for information. Bauer et al. [10] show that even resource-constrained clients, such as cell-phones, can build proofs of access. Alternatively, it is possible for such a client to offload proof building to a third entity. A service receiving a proof of access validates the proof as part of its access decision. This decision is cheap (see Section 6.4) and feasible for resource-constrained services, such as a sensor. Proofs of access have been proposed in earlier work [7,8]. Our contribution is the combination of proofs of access and information relationships.

While validating a proof of access, a service must authenticate access rights and detect tampering attacks. Therefore, we represent access rights as digital certificates, signed with their issuer's private key. To avoid bottlenecks, we do not store a client's access rights in a centralized knowledge base. Instead, we store them directly with the client. An individual granting an access right to a client will hand over this right to the client for storage, together with any information relationships bound to the access right. A client then uses its collection of access rights and information relationships for building a proof of access. We elaborate on proof building in Section 5.

2.2. Information relationships

An information relationship states that a client should be granted access to an information item if the client already has access rights to information item(s) related to this item. We now describe a set of information relationships that are particularly relevant to pervasive computing.

Bundling-based relationships: Though there might be many different types of information about an individual, some of them have identical access requirements. The individual should be able to bundle such information and to issue only a single access right for the entire bundle. For example, assume that Alice wants to grant multiple people access to both her location and her activity information. Therefore, for each person, she needs to issue two access rights. Instead, Alice should be able to bundle both her location and her activity information in her personal information and to grant each person only a single access right to her personal information. Access control will then derive individual access rights for Alice's location and activity information from the bundle. This bundling of information establishes an information relationship, as defined above, between the location information and the personal information. Similarly, it establishes a relationship between the activity information and the personal information.

Bundling-based relationships reduce the number of access rights that Alice needs to establish and the possibility of mistakes and information leaks.

Only Alice should be able to bundle her location information in other information. If we allowed Bob to bundle Alice's location information, he could bundle this information in his own personal information. Since Bob has access to his personal information, he would also be granted access to Alice's location information.

Combination-based relationships: Rich information is information that includes other types of information. For example, a map shows the location of multiple people or Carol's calendar entry provides her location and the location of Bob, who is attending a meeting with Carol. Therefore, there is an information relationship, as defined above, between the rich information and the included types of information, and access rights to the rich information depend on the existence of access rights to the included types. For example, a client can access a map only if the client has access rights to all the people's location shown on the map. Similarly, a client can access Carol's calendar only if the client has access rights to Carol's and Bob's location information. As for bundling-based relationships, combination-based relationships reduce the number of access rights that Alice needs to establish and the possibility of mistakes and information leaks.

Only Carol should be able to define a combination-based relationship for her calendar entry. In particular, it is up to Carol to decide whether she wants to define such a relationship in the first place. If Bob agrees to a meeting with Carol, he will have to rely on Carol not to make this information publicly available. If Carol is malicious, she will not respect Bob's privacy and let anyone access the corresponding calendar entry (or she will exploit other channels for providing the information in this entry). Only laws can avoid this information leak. However, if Carol is well behaved, she will want to respect Bob's privacy, and she will want to take his access rights into account when granting people access to her calendar entry. Combination-based relationships make it easy to incorporate Bob's access rights, since Carol does not even need to know Bob's access rights. We discuss trade-offs between defining an access right to rich information and a relationship for the same information in the first author's Ph.D. thesis [11, Chapter 3].

Granularity-based relationships: Some information, such as location information, is available at different levels of granularity. There are information relationships, as defined above, between the different levels, namely coarse-grained information is related to more fine-grained information. In other words, access rights to coarse-grained information should be derivable from access rights to more fine-grained information. For example, if Alice had an access right to Carol's fine-grained location information, she automatically should also have an access right to Carol's coarse-grained location information. There should be no need for Carol to establish the second access right. Therefore, granularity-based relationships reduce the number of access rights that Carol needs to define.

With the exception of combination-based relationships, information relationships are static and require few updates by the individuals defining them. This property obviously holds for granularity-based relationships. For bundling-based relationships, we expect an

individual to organize her information once and to make only few changes after that. (If an individual did not want to define her own relationships, she could exploit global relationships; see Section 4.) Therefore, defining bundling-based information relationships will be a rare task.

We envision that services will automatically define combination-based relationships on behalf of individuals. For example, when Carol enters a calendar entry, the calendar application will automatically issue a relationship between her calendar information and her (and her meeting participant's) location information. Therefore, the potentially dynamic nature of combination-based relationships does not affect individuals.

Our solution has the benefit that it decreases the number of access rights that an individual needs to specify. This reduction becomes even more important when access rights are not established once and then left unchanged, but when they are defined in a dynamic way. For example, Alice might want to grant access to Bob only for a short period of time in order to enable Bob to fulfill a task. If access rights are dynamic, information relationships will decrease the number of access rights that Alice needs to establish on an ongoing basis.

An obvious question is whether the three information relationships discussed in this section are common and important in pervasive computing. We observe that, for each of our information relationships, there is a corresponding concept in role-based access control. In particular, bundling-based relationships correspond to assigning roles to people, combination-based relationships are similar to resources that require the presence of multiple roles, and granularity-based relationships correspond to hierarchical role schemes. There are additional concepts in role-based access control, such as separation of duty, which ensures that a person cannot be a member of conflicting roles. It looks possible to introduce corresponding information relationships in pervasive computing. However, after studying the various types of information available in an existing pervasive computing environment [12], we concluded that the three existing relationships are common and sufficient for controlling access to this information and that we have not identified scenarios that are frequent enough to justify the introduction of additional relationships as first-class citizens in our access control model. (It is always possible for a service to handle such scenarios on a case-by-case basis.) Another reason for keeping the types of information relationships simple and their number limited is that, in pervasive computing, access rights and information relationships will typically be managed by individuals. These individuals will have only limited understanding of access control.

3. Formalizing access control

Access control exploits access rights and information relationships. To avoid ambiguities, we require a formal definition of the conditions under which access should be granted. This formal model will also provide the basis for our implementation (Section 6.1). We introduce a representation scheme for information (Section 3.1). We then present a formal model for access control, based on previous work [13,14] (Section 3.2). Our contribution is the extension of this model to support information relationships (Section 3.3). We conclude by demonstrating its application in a complex scenario (Section 3.4).

3.1. Information representation

In order to formalize access control to information, we need a representation scheme for information. Access rights should be service and environment independent. For example, Alice should be able to grant access rights to her location information regardless of which services actually offer this information and which environment she currently is in. Therefore, the information representation scheme should not bind information to a service or environment that offers this information. For example, URLs, which are used for identifying information on the Web, do not fulfill this requirement since they bind information to a service. In addition, the scheme should include the *owner* of the information. The owner is the individual who is allowed to issue access rights to the information. Including this owner in the representation scheme will make it straightforward to differentiate between valid and invalid access rights during access control. These two requirements lead us to represent information in the following way:

(**owner**, **item**).**type**.

Let us elaborate on the various elements of this scheme:

Owner: As just mentioned, the owner is allowed to issue access rights to his information. For example, Bob issues access rights to his location information. We represent an owner with his public key. (The owner's private key is used by the owner to issue access rights.)

Item: Information is about a particular item (e.g., Bob, his car, a room in his house, or an event that he organizes).

Type: Information is of a particular type (e.g., location, activity, financial, temperature,...).

Each component of the representation should be associated with a formal description of the component. For example, the description could say that a public key representing an owner is an RSA public key of 1024 bits. Similarly, it could say that item Bob is a person or it could define the type "location". With the help of this description, a service offering information can ensure that its notion of the information corresponds to the owner's understanding, as expressed in an access right to this information. Similarly, a client that exploits an access right to information can ensure that its notion matches the owner's understanding.

For the formal description of information, we can exploit descriptions developed for the Semantic Web, based on OWL [15]. These *ontologies* make it possible to identify, for example, different types of information that have the same semantic meaning (e.g., "location" and "whereabouts") or that are expressed in different languages (e.g., "location" vs. "Standort"). Using this approach, the description of a component in the information representation scheme consists of a URL pointing to an ontology. For example, SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) [16] contains both an OWL representation of "person" (http://pervasive.semanticweb.org/ont/2004/06/person) and of "location" (http://pervasive.semanticweb.org/ont/2004/06/location).

With the help of ontologies, we can ensure that different individuals or services agree on the meaning of a component in the information representation scheme. For example, they agree that a particular item is a person, but not a car. However, different individuals or services could still use different schemes for naming the same owner or item. For example, a location service knows client Alice under the user-id "alice", whereas Alice uses her public key for identifying the owner of her location information in access rights granted by her. Related to this observation, we point out that the string denoting the item has meaning only in the name-space of the owner of the information. For example, the room that building manager Alice identifies as "Wean Hall 8220" might not correspond to the room that building manager Bob identifies as "Wean Hall 8220". Local name-spaces have been used in previous research [17,18]. They rely on the observation that the existence of a global name-space, which could avoid this confusion, is unrealistic.

To cope with different naming schemes for the same owner or item, a service must ensure that the owner and item of information that it provides really correspond to the owner and item of information in an access right that the service is going to use when controlling access to this information. This validation can occur manually or automatically. For example, the administrator of a service can perform manual validation when the owner of information registers with the service. For automatic validation, there could be ontologies that relate between different ownership representations and different local name-spaces. (Formally, we could also use bundling-based information relationships, as presented in Section 3.3.1, for automatic validation.)

Given the existence of such mappings between owners or items, we assume that all individuals and services agree on the representation and naming scheme for information. Using our representation scheme, we can describe Alice's location as "(⟨Alice's public key⟩, Alice).location". (We use ⟨⟩ as a placeholder for the actual public key.) Similarly, we can describe the temperature in Wean Hall 8220 as "(⟨Public key of building manager⟩, Wean Hall 8220).temperature".

To simplify the notation in the rest of this paper, we use the name of an owner instead of the public key representing the owner (e.g., "Alice" instead of "⟨ Alice's public key⟩"). In addition, we use the shortcut

entity.type

for representing information where the owner and the item are identical. Therefore, "(⟨Alice's public key⟩, Alice).location" simply becomes "Alice.location". Note that Alice's location information is not necessarily owned by Alice. For example, in an enterprise environment, this information could be owned by the enterprise. In such a scenario, we can represent Alice's location as "(Enterprise, Alice).location". In general, it is a service (i.e., its administrator) that decides about ownership of information provided by the service.

3.2. Basic access control

Our formal model is based on Howell and Kotz's "restricted speaks-for" predicate [13] between principals, which exploits Lampson et al.'s "speaks-for" predicate [14]. A principal is an entity that can make a statement, such as a client or an owner of information. For a client to be granted access to information provided by a service, the client needs to be

able to speak for the owner of the requested information in terms of this information. For example, the predicate Bob $\overset{\text{Alice.location}}{\Longrightarrow}$ Alice denotes that Bob can speak for Alice in terms of Alice's location information. Formally, this predicate allows the following conclusion: Bob **says** Alice.location $\supset$ Alice **says** Alice.location. (The $\supset$ connector denotes implication. In addition to the $\supset$ connector, there is also the $\wedge$ connector, which serves for conjunction, as shown below.) A service offering Alice's location information will grant Bob access to this information only if this predicate holds. The predicate is restricted, that is, Bob can speak for Alice only in terms of her location information, but not in terms of any other information (i.e., he will not be granted access to any other information). Note that speaks-for predicates are transitive.

We need to formalize the establishment of speaks-for predicates. The "handoff axiom" [14] says that such a predicate holds only if it is made by the principal on the right-hand side. Formally,

$$\vdash (A \textbf{ says } (B \overset{C.x}{\Longrightarrow} A)) \supset (B \overset{C.x}{\Longrightarrow} A). \tag{1}$$

In the above example, the predicate can be established only if Alice issues it, but not if Bob (or someone else) does. Therefore, A **says** $(B \overset{C.x}{\Longrightarrow} A)$ corresponds to our notion of an access right for information $C.x$ granted to B by A.

3.3. Relationship-aware access control

We extend the formal model to support information relationships. An information relationship states that if an entity B has access to information items $E_i.x_i$, B should also have access to a related item $D.x$. Formally, we use the $E_1.x_1 \otimes E_2.x_2 \otimes \cdots \longrightarrow D.x$ predicate for expressing this relationship,[1] and we want the axiom

$$\begin{aligned} &\vdash (E_1.x_1 \otimes E_2.x_2 \otimes \cdots \longrightarrow D.x \wedge B \overset{E_1.x_1}{\Longrightarrow} A_1 \wedge B \overset{E_2.x_2}{\Longrightarrow} A_2 \wedge \cdots) \\ &\quad \supset (B \overset{D.x}{\Longrightarrow} C) \end{aligned} \tag{2}$$

to hold for certain conditions on the principals E_i, their information $E_i.x_i$, and the principals A_i and C. In Sections 3.3.1–3.3.3, we show that instantiating Axiom (2) in different ways straightforwardly leads to the concepts of bundling-based, combination-based, and granularity-based information relationships, respectively. For each type of relationship, we also discuss plausible conditions on the various entities in the axiom and pick the most useful one.

In addition to formalizing the application of information relationships in access control, we also need to formalize their establishment. Since access to the information items on the left-hand side of a relationship also grants access to the item on the right-hand side, only a principal already speaking for the owner of the information on the right-hand side in terms of the information should be able to establish the relationship. (The owner can

[1] We use the $\otimes$ symbol instead of the $\wedge$ symbol, since the latter symbol already serves as a connector in the logic.

always speak for the owner.) Formally,

$$\vdash (F \textbf{ says } (E_1.x_1 \otimes E_2.x_2 \otimes \cdots \longrightarrow D.x)) \wedge (F \stackrel{D.x}{\Longrightarrow} D) \supset (E_1.x_1 \otimes E_2.x_2 \otimes \cdots \longrightarrow D.x). \quad (3)$$

For example, Alice, as the owner of her location information (and anyone she grants access to), can bundle this information in her personal (or some other) information. Similarly, Carol, as the owner of her calendar, can define a relationship stating that anyone who has access to her (and potentially other people's) location information can also access her calendar entry.

To prove soundness of our additions to Howell and Kotz's access control logic, we need to give our additions a semantics. We discuss soundness in more detail in the first author's Ph.D. thesis [11, Chapter 3]. We are now going to look at useful instances of Axiom (2).

3.3.1. Bundling-based relationships

Limiting the number of information items on the left-hand side of Axiom (2) to one item corresponds to the concept of bundling-based relationships. For example, the statement "Alice.personal $\longrightarrow$ Alice.location" denotes that information "Alice.location" is bundled in information "Alice.personal", that is, if someone has access to Alice's personal information, he should also have access to her location information.

There are two plausible conditions on the various entities in Axiom (2). The first one requires $A_1 = C$, or

$$\vdash (E_1.x_1 \longrightarrow D.x \wedge B \stackrel{E_1.x_1}{\Longrightarrow} A_1) \supset (B \stackrel{D.x}{\Longrightarrow} A_1). \quad (4)$$

The second one calls for $A_1 = E_1$ and $D = C$, or

$$\vdash (E_1.x_1 \longrightarrow D.x \wedge B \stackrel{E_1.x_1}{\Longrightarrow} E_1) \supset (B \stackrel{D.x}{\Longrightarrow} D).$$

We choose the first condition, since the second one is too limiting. For example, given $E_1.x_1 \longrightarrow D.x$, $B \stackrel{E_1.x_1}{\Longrightarrow} C$, and $C \stackrel{D.x}{\Longrightarrow} D$, it should be possible to conclude $B \stackrel{D.x}{\Longrightarrow} D$, which the second condition does not permit.

Our formal model allows individuals to bundle some of their information in someone else's information. For example, some people collaborating on a project can bundle information that is relevant to the project in information owned by the project manager. The project manager can then grant access to the information bundle.

Our discussion assumes that an individual establishes all her bundling-based relationships. However, many individuals might establish the same types of relationships. Therefore, it should be possible for a standardization organization to establish global bundling-based relationships, to which individuals can subscribe. We discuss this concept in Section 4.

3.3.2. Combination-based relationships

Not limiting the number of information items on the left-hand side of Axiom (2) corresponds to the concept of combination-based relationships. For example, the statement "Alice.location $\otimes$ Bob.location $\longrightarrow$ Map Service.map" denotes that the map offered by a

map service can be accessed by anyone who has access to both Alice's and Bob's location information. (The assumption is that only Alice's and Bob's location is shown on the map.)

Again, there are two plausible conditions on the various entities in Axiom (2). We can require $A_1 = E_1$, $A_2 = E_2, \ldots$, and $C = D$, or

$$\vdash (E_1.x_1 \otimes E_2.x_2 \otimes \cdots \longrightarrow D.x \wedge\ B \xRightarrow{E_1.x_1} E_1\ \wedge B \xRightarrow{E_2.x_2} E_2 \wedge \cdots)$$
$$\supset (B \xRightarrow{D.x} D).$$

The second option calls for $A_1 = A_2 = \cdots = C$, which is an extension of the condition for bundling-based relationships. This condition requires a single entity that has access to all of $E_1.x_1$, $E_2.x_2$,..., and $D.x$, through which B would then acquire its access rights. However, this requirement is unrealistic in practice and does not fit the intuitive model of combination-based relationships. Therefore, we pick the first condition.

3.3.3. Granularity-based relationships

Granularity-based information relationships are a special case of bundling-based relationships. For example, Alice could define two types of location information, "Alice.location_fine" and "Alice.location_coarse", and establish "Alice.location_fine $\longrightarrow$ Alice.location_coarse".

However, requiring individuals to introduce separate types of information for different levels of granularity is tedious and not intuitive. Instead, we observe that a granularity-based access right to information can be represented as an access right that has a constraint on the returned information. For example, if Bob had access to Alice's coarse-grained location information and asked for her location, the result would have to be coarse grained. This result (e.g., "(Building Manager, Wean Hall 8220)" using our representation scheme) is itself an item about which there is some information (such as its granularity).

Constraints are not a new concept in access control. For instance, an access right can be constrained to be valid only during office hours. By extending our formal model to support constraints, we can express that Bob has access to Alice's location at fine or coarse granularity as follows (for readability reasons, we put constraints under the arrow):

$$\text{Bob} \xRightarrow[\text{?result.granularity} \geq \text{fine}]{\text{Alice.location}} \text{Alice}.$$

3.4. Example

In this section, we demonstrate the application of our formal model in an example scenario. The scenario involves a location service that provides the identity of the people in a room: two users Alice and Bob managing their access rights, and two users Dave and Carol trying to access Alice's or Bob's confidential information.

Alice bundles fine-grained location information (and potentially other) information in her personal information (Statement (1) in Table 1). She grants Carol access to her personal information (2) and Dave access to her coarse-grained location information (3). Bob grants Carol access to his fine-grained location information (4). The information provided by the location service should be accessible only to clients having access rights to the

Table 1
Example statements

(1)	Alice **says** Alice.personal $\longrightarrow$ Alice.location [?result.granularity $\geq$ fine]
(2)	Alice **says** Carol $\overset{\text{Alice.personal}}{\Longrightarrow}$ Alice
(3)	Alice **says** Dave $\overset{\text{Alice.location}}{\underset{\text{?result.granularity=coarse}}{\Longrightarrow}}$ Alice
(4)	Bob **says** Carol $\overset{\text{Bob.location}}{\underset{\text{?result.granularity}\geq\text{fine}}{\Longrightarrow}}$ Bob
(5)	Location Service **says** Alice.location [?result.granularity = fine] $\otimes$Bob.location [?result.granularity = fine] $\longrightarrow$ (Location Service, Wean Hall 8220).people

Alice and Bob grant Carol and Dave access to their personal information using information relationships.

location information of all individuals in a room, that is, we require a combination-based relationship. Assuming that only Alice and Bob are in Wean Hall 8220, the service defines a corresponding relationship (5).

Carol queries for the people in Wean Hall 8220. She is granted access based on the information relationship of the location service (5), Alice's information relationship (1), Alice's access right (2), and Bob's access right (4). Dave also issues a query and is denied access, since the intersection of ?result.granularity = coarse (3) and ?result.granularity = fine (5) is empty.

As demonstrated in this section, our scheme makes it straightforward to run access control based on information relationships. Moreover, there is no need for the different types of information to be owned by the same entity.

4. Global information relationships

Many individuals might define the same kind of bundling-based relationships. We now introduce *global* information relationships, where standardization organizations specify bundling-based information relationships on behalf of individuals and let individuals subscribe to these relationships. This way, individuals no longer have to define their own relationships. We study two approaches, differing in their requirements for a global relationship to become valid. The first approach has the advantage that it does not require any changes to the existing formal model, but it gives a lot of power to a standardization organization. The second approach does not suffer from this drawback, but it requires an extension to the formal model.

4.1. Delegation-based approach

In the first approach, in order for a global relationship to become valid, an individual needs to grant the standardization organization access to her information. We call this approach "delegation-based", since the individual delegates all her capabilities to decide about the information's access rights and relationships to the organization. We illustrate the approach based on an example. Alice lets organization ACME decide about the

composition of her personal information. ACME decides that medical information is part of personal information and issues

$$\text{ACME } \textbf{says } (\text{Alice.personal} \longrightarrow \text{Alice.medical}). \tag{5}$$

In addition, Alice grants the organization access to her medical information, or

$$\text{Alice } \textbf{says } (\text{ACME} \xRightarrow{\text{Alice.medical}} \text{Alice}). \tag{6}$$

This statement defines a speaks-for predicate between ACME and Alice in terms of "Alice.medical", which we can exploit to establish "Alice.personal $\longrightarrow$ Alice.medical" from Statement (5), according to Axiom (3).

ACME bundles "Alice.medical" in "Alice.personal". This approach works well under the assumption that both ACME and Alice agree that the composition of "Alice.personal" is managed by ACME. However, it could break if there were no such agreement. For example, it is possible that ACME bundles "Alice.medical" not only in "Alice.personal" but also in other information owned by Alice (e.g., in "Alice.health"). Therefore, an access right to "Alice.health" would also grant access to "Alice.medical". Alice would not want this property if she was not aware of this additional relationship and assumed that only she managed the composition of "Alice.health". To avoid this conflict, ACME should not bundle information owned by Alice in other information owned by Alice. Instead, ACME should bundle the information in information owned by ACME. Formally,

$$\text{ACME } \textbf{says } ((\text{ACME, Alice}).\text{personal} \longrightarrow \text{Alice.medical}). \tag{7}$$

To enable Alice to issue access rights to her personal information, ACME needs to state

$$\text{ACME } \textbf{says } (\text{Alice} \xRightarrow{(\text{ACME, Alice}).\text{personal}} \text{ACME}). \tag{8}$$

Alice can exploit ACME's definition of her personal information by issuing an access right to "(ACME, Alice).personal". If she did not want to use ACME's definition, she could issue an access right to her own personal information, that is, "Alice.personal".

The approach requires that Alice grants ACME access to each type of information that is part of ACME's notion of personal information (e.g., Statement (6)). However, Alice might not know for what information she has to issue such a statement, because she does not know or does not want to know what information is part of ACME's notion of personal information. (The whole point of letting ACME decide about relationships is to take away these decisions from Alice.) Similarly, Statements (7) and (8) assume that ACME issues an information relationship and an access right for each individual that wants to use ACME's relationships. However, if ACME were a standardization organization, it might be oblivious to the individuals who are using its specifications. These two observations suggest that digital certificates implementing these three statements should allow for wildcards. For example, in a certificate expressing Statement (6), Alice should be able to list a wildcard (or a set of possible values) instead of "medical". Similarly, in certificates expressing Statements (7) and (8), ACME should be able to include a wildcard, instead of "Alice".

4.2. Naming-based approach

The first approach hurts the principle of least privilege, since it grants access rights to the standardization organization, but the organization is not actually going to access this information. Our second approach does not have this disadvantage. It gives the organization the capability to define a bundling-based relationship for information owned by an individual, but it does not grant access rights to the organization. Therefore, we call this approach "naming-based". Its drawback is that it requires the introduction of an additional axiom to the formal model.

In terms of the example introduced in Section 4.1, we use "ACME $\overset{\text{bundle Alice.medical}}{\Longrightarrow}$ Alice" to denote that ACME can establish relationships for "Alice.medical". Only Alice and principals speaking for Alice in terms of "bundle Alice.medical" can establish this statement. Formally,

$$\vdash \text{ACME} \overset{\text{bundle Alice.medical}}{\Longrightarrow} \text{Alice} \wedge \text{ ACME } \textbf{says } (\text{Alice.personal} \longrightarrow \text{Alice.medical}) \supset (\text{Alice.personal} \longrightarrow \text{Alice.medical}).$$

In this way, ACME can define a bundling-based relationship for "Alice.medical", without having an access right to this information. Note that the owners of the two types of information in the relationship must correspond to Alice. In particular, ACME must not be allowed to bundle "Alice.medical" in its own information. This relationship would grant ACME access to "Alice.medical", which we want to avoid. However, allowing ACME to bundle "Alice.medical" in information that is owned by Alice can result in the problem discussed in Section 4.1. There, we worked around the problem by having ACME bundle information owned by Alice in information owned by ACME. As mentioned above, this is not an option here. Instead, Alice and ACME can adopt a naming scheme for information that avoids disagreement between Alice and ACME about who is responsible for the composition of some of Alice's information. For example, ACME could prefix the type of information with its name, that is, instead of bundling "Alice.medical" in "Alice.personal", ACME could bundle "Alice.medical" in "Alice.ACME_personal".

Similar to the first approach, in a digital certificate expressing the permission to define relationships, a wildcard could be used instead of "medical". Furthermore, it is possible to limit the scope of a certificate. For example, it could allow bundling of information in "Alice.ACME_personal", but not in other information owned by Alice.

5. Proof building

In our distributed access control architecture, we have a client submit a proof of access to a service. This approach has been investigated in previous work [7,8]. Our implementation is based on Howell and Kotz's framework [8], to which we added support for proofs of access involving information relationships. Proofs of access are structured, the structure corresponds to the types of axioms required for validating the proof. Fig. 2 illustrates a structured proof. In Section 3.3, we saw that one of the differences between the three types of information relationships is the way in which they instantiate Axiom (2). Another difference is the way in which the relationships are used in proof building, which we now discuss.

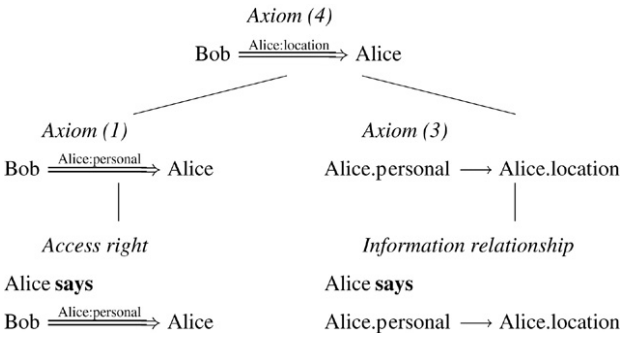

Fig. 2. Structured proof. The proof shows that Bob can speak for Alice regarding her location information, based on an access right to Alice's personal information and an information relationship between Alice's personal and her location information.

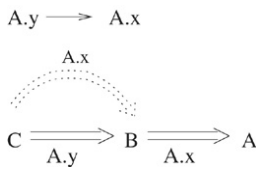

Fig. 3. Proof building. The goal is to prove "$C \overset{A.x}{\Longrightarrow} A$". The search starts at A. It locates "$B \overset{A.x}{\Longrightarrow} A$" and tries to prove "$C \overset{A.x}{\Longrightarrow} B$". The search locates "$C \overset{A.y}{\Longrightarrow} B$" and uses "$A.y \longrightarrow A.x$" to get the required information.

5.1. Bundling-based relationships

We first illustrate how proof building exploits bundling-based relationships. A client collects access rights and information relationships received from individuals. It stores speaks-for predicates derived from access rights or from other speaks-for predicates in a graph where nodes represent principals and edges represent information. For a request, the algorithm traverses the graph in a breadth-first way to find a proof of access, starting at the owner of the requested information and ending at the client that issues a request to a service. During this graph traversal, when the information in a candidate edge does not match the requested information, the algorithm looks at all bundling-based relationships that have the requested information on their right-hand side and checks whether the left-hand side of the relationship matches the information in the candidate edge. (If there is a hierarchy of bundling-based relationships, this step requires the exploration of multiple relationships.) If so, the algorithm accepts the candidate edge and the edges connected to this edge become new candidate edges. Fig. 3 illustrates this algorithm. In summary, the algorithm exploits backward chaining for searching access rights and forward chaining for searching information relationships.

The algorithm looks at each edge at most once. Therefore, if there are n speaks-for predicates and no information relationships, its worst-case complexity is $\mathcal{O}(n)$. If there are also m information relationships, the algorithm will look at an information relationship at most once for each candidate edge. The worst-case complexity becomes $\mathcal{O}(nm)$. Although

complexity is multiplicative, we expect proof building to be practical. First, the absolute value of n will be larger for the case without information relationships, since this case requires separate access rights for information with identical access requirements. Second, we expect principals to define information relationships in a way such that information is bundled only in a small number of information bundles and to keep the hierarchies of bundling low. We measure proof building time in Section 6.3.

Global relationships could increase the complexity of proof building to $\mathcal{O}(nm^2)$. In particular, each global relationship visited during proof building could require a separate proof building step, in which the additional speaks-for predicate required for the establishment of the global relationship is determined. For example, establishing the relationship in Statement (7) requires a predicate showing that ACME can speak for Alice in terms of "Alice.medical". However, it is possible to avoid this increase in complexity during proof building. Namely, a client should locate the access rights required for establishing a global information relationship early on, that is, when the client inserts the statement defining the relationship into its collection of access rights and relationships. This way, there are only validated information relationships in the collection upon proof building time.

5.2. Combination-based relationships

We do not expect clients to exploit combination-based relationships in proof building. Instead, these relationships are used by services. A combination-based relationship is tightly coupled to the information that a service provides (e.g., for a map service, the relationship consists of all the people shown on the map). Therefore, it is straightforward for a service to establish the corresponding relationship and to exploit it for access control. In particular, for each information item on the left-hand side of the relationship, the service has the client build a proof of access. The service then aggregates these individual proofs of access and its combination-based relationship into a summary proof.

Proof building by a client can be difficult in this scenario, since the client might not know what the individual proofs of access should look like and the service cannot inform the client of their nature without leaking information. For example, if a map service had the information relationship "Alice.location $\otimes$ Bob.location $\longrightarrow$ Map Service.map", a client would have to build proofs of access for "Alice.location" and "Bob.location". However, the client might not know who is on the map, and the service cannot tell the client leaking location information about Alice and Bob. We address this problem in related work [19].

5.3. Granularity-based relationships

For granularity-aware access control, the algorithm introduced in Section 5.1 must be extended to take constraints on access rights and information relationships into account. For example, "Carol $\xrightarrow[?result.granularity \geq fine]{Alice.location}$ Bob" and "Bob $\xrightarrow[?result.granularity=coarse]{Alice.location}$ Alice" can be concatenated to "Carol $\xrightarrow[?result.granularity=coarse]{Alice.location}$ Alice". In our approach, we encode granularity-based relationships directly in access rights and do not require separate statements to define these relationships. Implementing granularity-based relationships in

this way does not affect the algorithmic complexity of proof building (i.e., $\mathcal{O}(nm)$). However, this approach does have an influence on the absolute proof building cost. On the one hand, the approach increases the constant factor in the absolute cost, since, in addition to equality, the proof building algorithm now needs to be able to deal with inequalities. On the other hand, the approach reduces absolute cost, since the number of required information relationships, m, becomes smaller.

In summary, our proof building algorithm is simple, but it has proved sufficient for our application scenarios. We can also trade off computation vs. storage. Instead of computing a proof upon a request, a client can store the closure of its access rights and update this closure whenever it receives an access right or information relationship. Instead of employing a brute-force prover, a client can use more sophisticated theorem provers or Semantic Web rule engines. This approach would allow us to combine the benefits of both worlds (i.e., no single point of failure and more optimized proving).

6. Performance analysis

We analyze our proposed information relationships along three axes: their effect on the number of issued access rights and their influence on proof building time and on request processing time. We run our measurements on an unloaded Pentium IV/2.5 GHz with 1.5 GB of memory, Linux 2.4.20, and Java 1.4.2.

6.1. Deployment environment

We use the Contextual Service Interface (CSInt) [20] developed for the Aura pervasive computing project [12] at Carnegie Mellon as a testbed for the implementation and deployment of our access control mechanism. CSInt has a client/server architecture, is implemented in Java, and allows a client to retrieve information from a service using a simplified SQL query language. We store access rights and information relationships as extended SPKI/SDSI digital certificates [17]. We give two examples in Fig. 4. SSL provides peer authentication and confidentiality and integrity of transmitted messages. Proofs of access are implemented as Java classes. Each axiom in Section 3 has its corresponding class. For example, there is a class for the handoff axiom (Axiom (1)). Each instance of a class is initialized with the preconditions of an axiom. The initialization values can be instances of proof classes themselves. A class is able to serialize and deserialize its instance variables so that proofs of access can be submitted to a service. When deserializing such a proof, a service must instantiate only classes representing axioms from our formal model. After instantiating a class, a service calls the new instance's verification method, which ensures that the preconditions can be combined in the formal way defined by the axiom underlying the class.

6.2. Number of access rights

We examine how bundling-based relationships affect the number of access rights that an individual has to issue. In particular, we compare the number of statements to be made

```
(cert
   (version ``1'')
   (issuer (public_key:alice))
   (subject (public_key:bob))
   (permission (information (public_key:alice) alice location))
   (tag *))

(bundling-relationship
   (version ``1'')
   (issuer (information (public_key:alice) alice location))
   (subject(information (public_key:alice) alice personal)))
```

Fig. 4. Extended SPKI/SDSI certificates. Shown are an access right and an information relationship. `public_key:foo` stands for foo's public key. The **tag** section can list constraints. We omit digital signatures.

in a world with information relationships to the corresponding number in a world without information relationships.

Let us first consider the case with information relationships. Assume that there is a full, tree-based hierarchy consisting of l levels of bundling-based relationships, where $l = 1$ corresponds to the base case consisting of a root node and some leaf nodes. Each parent node has m children (i.e., m types of information are bundled in each parent node). There are k clients. Each of them is assigned to a single node in the hierarchy, meaning that the client is given an access right to the information covered by the node. There are different ways to distribute the clients in the hierarchy. In this scenario, regardless of the clients' distribution, the number of access rights to be issued is always k and the number of relationships is always $\sum_{i=1}^{l} m^i$. The first curve in Fig. 5 shows the overall number of access rights and relationships for different values of l. We choose $k = 50$ and $m = 3$.

We want to know the number of access rights that would have to be issued if there were no relationships, but the k clients should have access to the same information as in the case with relationships. This number depends on the distribution of these clients in the tree. We examine three different distributions. The first one is artificial and presents the best case in possible savings of issued access rights: all k clients are assigned to the root node. Without relationships, this scenario would require km^l access rights (since there are m^l leaf nodes), as shown by the second curve in Fig. 5. The second distribution is also artificial and presents the worst case in possible savings: all k clients are distributed randomly among the m^l leaf nodes. This scenario would require k access rights, as shown by the third curve. The third distribution is a more realistic one: we distribute the k clients evenly among the $l + 1$ layers of nodes in the hierarchy. This would require $\frac{k}{l+1} \sum_{i=0}^{l} m^i$ access rights, as shown by the fourth curve. As we can see in Fig. 5, with the exception of the worst case, where relationships become unnecessary, relationships can lead to a significant decrease in the number of issued access rights and information relationships.

We believe that it is intuitive for individuals to define relationships, since the underlying paradigm is well known and already used for organizing files in directories or digital pictures in galleries. If the individual did not want to define her own relationships, she could always exploit relationships defined by third entities.

Similar to bundling-based relationships, granularity-based relationships require individuals to issue fewer access rights, where the decrease is proportional to the levels of granularities. Combination-based relationships also reduce the number of access rights,

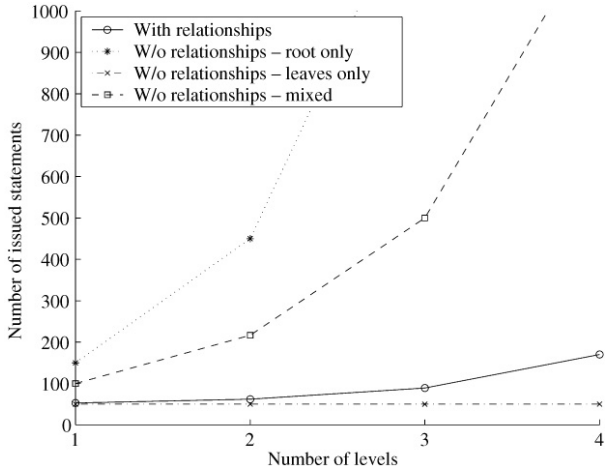

Fig. 5. Number of issued statements. We compare the number of issued access rights and relationships in a world with relationships to the number of issued access rights in a world without relationships, for different levels of bundling-based relationships and distributions of clients.

since they prevent an owner of information from having to issue separate access rights to rich information.

6.3. Proof building time

We examine the cost of proof building, as explained in Section 5. Our experiment consists of locating a set of speaks-for predicates in a pool of statements and assembling them in a proof of access. We consider up to five sequentially connectable (i.e., of the form "$C.x \longrightarrow B.x$", "$B.x \longrightarrow A.x$", ...) bundling-based relationships. Because of manageability reasons, we do not expect individuals to create more than five levels of bundling-based relationships in their information hierarchies. This expectation is based on observing people that organize their personal files in (sub) directories or their pictures in (sub) galleries. Note that the results presented in this section and our complexity analysis in Section 5.1 allow prediction of the cost even for a larger number of levels.

Our experimental setup covers a worst-case scenario: We pick five among 50 possible clients and create a sequential path of access rights between them (i.e., a client delegates its access right to the next client in the path). All experiments will locate this path, since there is considerable variation for different paths. (Whereas we present results only for this particular path, we did re-run our experiments for other paths, and the conclusions are identical.) We then put the access rights into a pool, which we expand by adding random access rights. The random access rights form a directed graph where each recipient of an access right is (indirectly) reachable from the issuer of the first access right in the predetermined path and where none of the recipients is on this path (i.e., we do not allow shortcuts). We randomly set the information in each access right to one of the possible information items covered by the bundling-based relationships. An experiment is characterized by a number of random access rights and a number of relationships and run

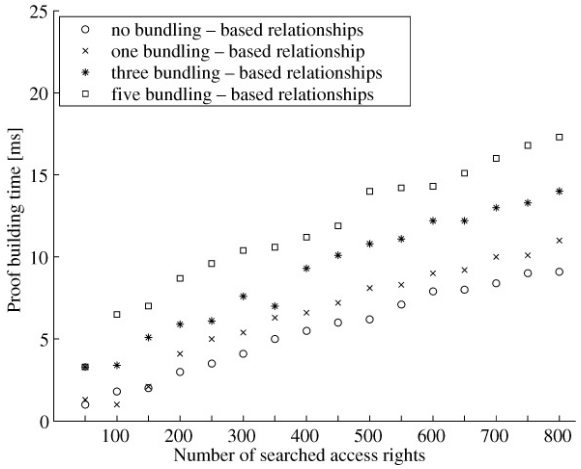

Fig. 6. Proof building time. For a fixed number of bundling-based relationships, mean proof building time increases linearly in the number of searched access rights.

ten times. Fig. 6 reports the mean proof building time. According to Section 5, the worst-case complexity is multiplicative in the number of speaks-for statements and information relationships. Our results confirm that the increase is linear for a particular number of relationships.

In practice, information relationships do not necessarily lead to increased proof building cost. In particular, if there are relationships, there typically will be fewer access rights in a client's pool, which makes proof building cheaper. Therefore, for a given value on the x-axis in Fig. 6, the various proof building times are not directly comparable. The actual cost strongly depends on the number and types of information relationships. Overall, as we will show in Section 6.4, the cost of building a proof is comparable to the cost of processing a request. We also observe that the numbers presented in Fig. 6 cover a worst-case scenario. First, we require four access rights for the proof of access; we expect this number to be smaller in practice (e.g., an access right in which the owner directly grants access to the client). Second, our experimental setup ensures that all access rights in a user's collection might be explored for proof building. In real collections, many of the access rights in a collection will not be explored. For example, a search for access rights to Alice's location information typically will not explore access rights to Bob's location information.

6.4. Client response time

We study the influence of bundling-based relationships on client response time in our prototype implementation. As mentioned in Section 5, the other types of relationships do not have a negative influence on complexity. Our asymmetric cryptographic operations required for setting up SSL connections and validating the signatures of certificates use 1024 bit RSA keys. An experiment is run 100 times. We report the mean and standard deviation (in parentheses). Since we are not interested in proof building time for these experiments, we assume that clients have pre-built proofs.

Table 2
Client response time

Entity	Step	μ	(σ)
Both	**SSL Socket creation**	53	(6)
Service	**Access decision**	3	(2)
Service	Gather location information	56	(7)
	Total	129	(13)

Mean and standard deviation of elapsed time for security operations (in bold) and for gathering information [ms].

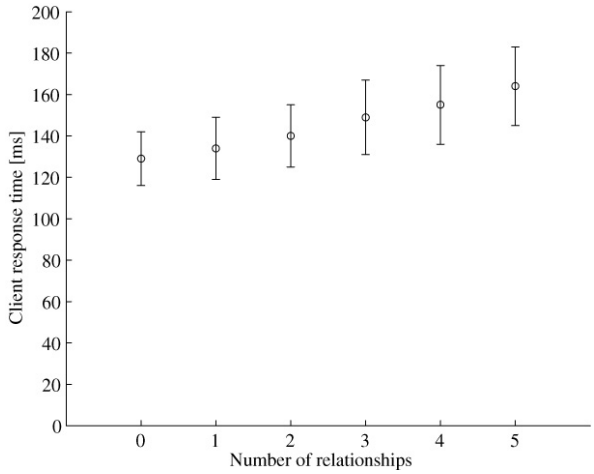

Fig. 7. Client response time. The elapsed time increases linearly with the number of relationships.

In the first experiment, Alice grants Bob access to her location information in an access right. Bob submits the corresponding proof to a location service, which validates it and locates Alice. The service then fingers Alice's Linux desktop computer and determines her location from her activity. We run the client and the location service on the same host. The mean response time is 129 ms (13 ms). More detailed results are in Table 2. Making an access decision takes only a few milliseconds, the main cost is validating the signature of the certificate. Gathering the location information is the most expensive step. Setting up an SSL connection requires two costly RSA decryption/signing operations for client and server authentication. Making an access decision and setting up SSL are CPU bound and will benefit from faster hardware.

In the second experiment, Alice grants Bob access to her location information via a variable number of sequentially connectable, bundling-based relationships. The results are in Fig. 7. Note that the scaling of the y-axis is different from the scaling in Fig. 6. The figure shows that client response time increases linearly by about 7 ms for each additional relationship. The main reasons for the increase are increased transmission cost and the validation of an additional signature.

7. Related work

There is a lot of research that deals with security and privacy issues for pervasive computing. Here, we limit ourselves to research that is closely relevant to the research presented, namely research devoted to access control frameworks, specification languages for access rights, and distributed access control. More research dealing with security and privacy issues for pervasive computing is discussed in the first author's Ph.D. thesis [11].

There are several frameworks for pervasive computing environments that support access control to confidential information [1–6]. Al-Muhtadi et al. [1], Chen et al. [2], Covington et al. [3], and Gandon and Sadeh [4] employ centralized rule engines with differing degrees of flexibility for running access control. It is possible to incorporate our proposed information relationships into these rule engines. However, the rule engine can become a performance bottleneck, represents a single point of failure, and needs to be fully trusted by all entities. In addition, these solutions store access rights in a centralized knowledge base, which is troublesome in terms of privacy; the knowledge base should not learn about all the access rights of an individual, it should know only about access rights that will be required for answering requests. A centralized approach seems to have the advantage of supporting negative access rights. However, it is unclear how big the benefit of negative access rights is; they might be too complex for individuals, not administrators, managing access rights. Also, since there can be multiple environments and thus multiple knowledge bases, consistency problems are still possible. Jiang and Landay [5] and Minami and Kotz [6] tag information with its access rights. The tag of derived information is derived from the tags of the source information. However, for many cases, automatic derivation is not possible and the tag needs to be manually specified, which is not scalable.

Multiple specification languages have been used for expressing access rights to information in pervasive computing. For example, Myles et al. [21] use an extended version of P3P [22], which allows Web servers to express their privacy practices. Chen et al. [2] exploit REI [23], which is targeted at pervasive computing environments. XACML [24] is an access control language for distributed systems. It is possible to add support for the expression of information relationships to these languages. However, these languages are targeted at environments where access control is run by a single entity. As opposed to the speaks-for predicate and SPKI/SDSI, they have no built-in mechanisms for verifying the authenticity of a statement, which is essential when running access control in a distributed way and delegating access rights across multiple environments.

Similar to Howell and Kotz, Bauer et al. [7] present an access control framework in which clients submit proofs of access to services. We exploit Howell and Kotz's framework because SPKI/SDSI is standardized [17] and it is based on the well-known speaks-for predicate.

8. Conclusions and future work

Access control in pervasive computing environments needs to take relationships between information into account. We proposed making such relationships first-class citizens in access control. This way, it becomes straightforward to control access to rich information. We identified three types of information relationships that are important in

pervasive computing environments, and we formalized their establishment and application in access control. Instead of defining their own relationships, individuals can exploit global relationships defined by a standardization organization. To avoid centralized access control, we integrated support for information relationships into a fully distributed access control architecture, where clients assemble proofs of access.

Our sample implementation and its deployment demonstrate the feasibility of our approach. Namely, compared to other costs, the cost of making an access decision is cheap, which makes it feasible for computationally weak devices to make access decisions. Furthermore, while bundling-based relationships increase the algorithmic complexity of proof building, the reduction in number of required access rights does not necessarily lead to higher absolute proof building cost.

We are deploying our access control infrastructure in additional Aura services. We will offer a bigger community of users access to these services in order to investigate what kind of access rights and relationships users define.

Acknowledgements

We thank the anonymous reviewers for their comments. This research was supported by the US Army Research Office through grant number DAAD19-02-1-0389 and by the US National Science Foundation under award number CNS-0411116.

References

[1] J. Al-Muhtadi, A. Ranganathan, R. Campbell, M.D. Mickunas, Cerberus: A Context-aware security scheme for smart spaces, in: Proceedings of IEEE International Conference on Pervasive Computing and Communications, PerCom 2003, 2003, pp. 489–496.

[2] H. Chen, T. Finin, A. Joshi, Semantic web in the context broker architecture, in: Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications, PerCom 2004, 2004, pp. 277–286.

[3] M.J. Covington, P. Fogla, Z. Zhan, M. Ahamad, A Context-aware security architecture for emerging applications, in: Proceedings of 18th Annual Computer Security Applications Conference, ACSAC 2002, 2002.

[4] F. Gandon, N. Sadeh, A semantic ewallet to reconcile privacy and context awareness, in: Proceedings of 2nd International Semantic Web Conference, ISWC2003, 2003.

[5] X. Jiang, J.A. Landay, Modeling privacy control in context-aware systems, IEEE Pervasive Computing 1 (3) (2002) 59–63.

[6] K. Minami, D. Kotz, Controlling access to pervasive information in the "Solar" system, Tech. Rep. TR2002-422, Dept. of Computer Science, Dartmouth College, February 2002.

[7] L. Bauer, M.A. Schneider, E.W. Felten, A general and flexible access-control system for the web, in: Proceedings of 11th Usenix Security Symposium, 2002, pp. 93–108.

[8] J. Howell, D. Kotz, End-to-end authorization, in: Proceedings of 4th Symposium on Operating System Design & Implementation, OSDI 2000, 2000, pp. 151–164.

[9] U. Hengartner, P. Steenkiste, Exploiting information relationships for access control, in: Proceedings of 3rd IEEE International Conference on Pervasive Computing and Communications, PerCom 2005, 2005, pp. 269–278.

[10] L. Bauer, S. Garriss, M.K. Reiter, Distributed proving in access-control systems, in: Proceedings of 2005 IEEE Symposium on Security and Privacy, 2005, pp. 81–95.

[11] U. Hengartner, Access control to information in pervasive computing environments, Ph.D. Thesis, Computer Science Department, Carnegie Mellon University, available as Technical Report CMU-CS-05-160, August 2005.

[12] D. Garlan, D. Siewiorek, A. Smailagic, P. Steenkiste, Project aura: Towards distraction-free pervasive computing, IEEE Pervasive Computing 1 (2) (2002) 22–31.
[13] J. Howell, D. Kotz, A formal semantics for SPKI, in: Proceedings of 6th European Symposium on Research in Computer Security, ESORICS 2000, 2000, pp. 140–158.
[14] B. Lampson, M. Abadi, M. Burrows, E. Wobber, Authentication in distributed systems: Theory and practice, ACM Transactions on Computer Systems 10 (4) (1992) 263–310.
[15] M. Dean, G. Schreiber, OWL Web Ontology Language Reference, W3C Recommendation, February 2004.
[16] H. Chen, F. Perich, T. Finin, A. Joshi, SOUPA: Standard ontology for ubiquitous and pervasive applications, in: Proceedings of First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous 2004, 2004.
[17] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen, SPKI certificate theory, RFC 2693, September 1999.
[18] N. Li, J.C. Mitchell, RT: A role-based trust-managment framework, in: Proceedings of The Third DARPA Information Survivability Conference and Exposition, DISCEX III, 2003, pp. 201–212.
[19] U. Hengartner, P. Steenkiste, 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, 2005, pp. 384–393.
[20] G. Judd, P. Steenkiste, Providing contextual information to ubiquitous computing applications, in: Proceedings of IEEE International Conference on Pervasive Computing and Communications, PerCom 2003, 2003, pp. 133–142.
[21] G. Myles, A. Friday, N. Davies, Preserving privacy in environments with location-based applications, Pervasive Computing 2 (1) (2003) 56–64.
[22] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, J. Reagle, The platform for privacy preferences 1.0 (P3P1.0) specification, W3C Recommendation, April 2002.
[23] L. Kagal, T. Finin, A. Joshi, A policy language for a pervasive computing environment, in: Proceedings of 4th International Workshop on Policies for Distributed Systems and Networks, 2004, pp. 63–76.
[24] S. Godik, T. Moses, eXtensible Access Control Markup Language (XACML) Version 1.0, OASIS Standard, February 2003.