Exploiting Information Relationships For Access Control

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

Abstract

Pervasive computing environments offer a multitude of information services that provide potentially complex types of information. Therefore, when running access control for sensitive information, these environments need to take relationships between information into account. Other approaches to relationship-aware access control (e.g., based on Semantic Web rule engines) are often expensive and based on a centralized design. In this paper, we identify three types of information relationships (bundling-based, combination-based, and granularity-based) that are common and important in pervasive computing, and we integrate support for them in a distributed, certificate-based access control architecture. 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.

1. Introduction

In pervasive computing environments, there are a multitude of services that provide potentially sensitive information about an individual, such as her location, her personal files, her email, 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 Figure 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 sensitive information, a client requires *access rights*. An individual should be able to issue access rights for her sensitive information. However, having the individual issue access rights per client, per service, per environment, and per type of information is not scalable. To address this problem, pervasive computing frameworks that support access control [1, 5, 6, 10, 16, 20] address the first

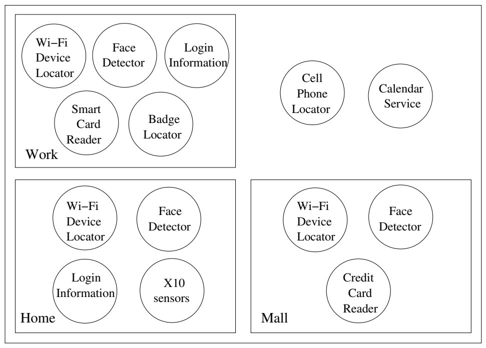

Figure 1. Multitude of location services. There are multiple environments, each having its own set of location services.

three axes. They employ 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 information about her context, such as her location or her activity information. In a naïve solution, whenever she wants to grant someone access to all her context-related information, she has to issue a separate access right for each type of context-related information. In a better solution, Alice can bundle these types of information and grant access rights to information bundles. When she wants to grant someone access, she now has to issue only a single access right. By bundling information, Alice establishes information relationships. The access control mechanism exploits these relationships in order to derive individual access rights.

Another example demonstrating the usefulness of information relationships involves complex information. Assume that the current entry in Alice's calendar says that she is having a meeting with Bob, that is, the calendar entry reveals the location of Alice and Bob. Therefore, only people who are at least allowed to access Alice's and Bob's location should have access to the calendar entry. To implement this rule, Alice should issue an access right for this 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. Instead, access control should be aware that there is a relationship between Alice's and Bob's location information and her calendar entry and directly take the relationship into account.

There are several frameworks for pervasive computing that exploit knowledge representations developed for the Semantic Web [3] and that use rule engines to reason about this knowledge [5, 10]. Such an approach can exploit certain information relationships for access control, but it has the disadvantage that the rule engine is centralized and can become a performance bottleneck and an attractive target for attackers. As an alternative, there are distributed, certificate-based access control architectures [2, 15], where clients gather and reason about access rights and services validate access rights received from clients. Weak clients can offload this reasoning to a proxy. 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 the centralized rule engine.

The contributions of our work are the concept of information relationships as a first-class citizen for access control and a distributed access control architecture that exploits information relationships (Section 2). We also present a formalism for incorporating these relationships into access control (Section 3) and examine how clients reason about them (Section 4). We discuss a prototype deployment and analyze its complexity and performance (Section 5).

2. Access Control Architecture

In this section, we discuss our distributed access control architecture and information relationships in more detail.

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. In our architecture, we have a client assemble a *proof of access* based on the client's access rights and transmit this proof to a service, which validates the proof. For validation, the service must be able to authenticate access rights. Therefore, we represent access rights as digital certificates, signed by their issuer. Digital certificates can be stored anywhere. 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 4.

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 some information item(s) related to the requested 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 information and to issue only a single access right for the entire bundle. For example, assume that Alice bundles her medical and her location information in her private information and grants Bob access to her private information. Bob's access rights for her medical and her location information can then be derived from this access right.

Combination-based relationships: Complex information combines other types of information to form some new information. For example, assume that a map shows the location of multiple people or that a calendar entry provides the location of people attending a meeting. Based on this relationship between the complex and the individual pieces of information, it should be possible to derive access rights for the complex information from access rights for the pieces. For example, if a client had access rights to all the people's location shown on the map, the client should also have access to the map.

Granularity-based relationships: Some information, such as location information, has different levels of granularity. There is an information relationship between the different levels, meaning that access rights to coarse-grained information should be derivable from access rights to fine-grained information.

With the exception of combination-based relationships, information relationships are static and require few updates by the individuals defining them. Combination-based relationships will mainly be defined on the fly by services providing information and not by individuals managing their access rights, so their dynamic nature does not affect these individuals. An obvious question is whether the three information relationships discussed in this section are common and important. We consider our approach of running access control by relating information as a dual to running access control by relating people (i.e., role-based access control). For each type of information relationship, there exists a corresponding, major concept in role-based access control. In particular, bundling corresponds to assigning roles to people, combination to separation of duty, and granularity to hierarchical role schemes. This observation suggests that the three relationships are common and sufficient, and so far this has held up in our examples.

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 formalism will also provide the basis for our implementation (Section 5.1). The formalism is based on previous work [14, 19] (Section 3.1). Our contribution is its extension to support information relationships (Section 3.2). We conclude by demonstrating its application in a more involved scenario (Section 3.3).

3.1. Basic Access Control

In our formalism, we represent information by a tuple $\langle$`owner, entity, type, description`$\rangle$. Information has an `owner` who is responsible for issuing access rights for it. Since we use information relationships for deriving access rights, the owner of the information is also responsible for defining relationships for this information. Information is of a particular `type` (e.g., location) and is about some `entity` (e.g., Alice). In addition, there needs to be a formal `description` of the type (e.g., defining the notion of "location"). For this description, we can exploit descriptions developed for the Semantic Web, based on OWL [8]. In our formalism, the information ⟨Alice, Alice, location, URL_to_description⟩ refers to Alice's location information. The information ⟨Facilities_Manager, Wean_Hall_8220, temperature, URL_to_description⟩ refers to the temperature in Wean Hall 8220. To keep the formalism concise, we shorten this tuple and also refrain from including the description in the notation. We use $(A, I).x$ for denoting a piece of information about entity I of type x controlled by owner A. If the owner and the entity are identical, we will use the shortcut $A.x$.

Our formalism is based on Howell and Kotz's "restricted speaks-for" notation [14], which exploits Lampson et al.'s "speaks-for" notation [19]. For a client to be granted access to some information by a service, the client needs to speak for the owner of the requested information in terms of this information. For example, the statement Bob $\xRightarrow{\text{Alice.location}}$ Alice denotes that Bob speaks for Alice in terms of Alice's location information. Therefore, a service will grant Bob access to Alice's location information.

We need to formalize the establishment of speaks-for statements. The "handoff axiom" [19] says that such a statement holds only if it is made by the principal on the right-hand side. Formally, ($\supset$ denotes implication.)

$$(1) \quad \vdash (A \textbf{ says } (B \xRightarrow{C.x} A)) \supset (B \xRightarrow{C.x} A).$$

In the above example, the statement holds only if Alice makes it, but not if Bob (or someone else) makes it.[1] Therefore, A **says** $(B \xRightarrow{C.x} A)$ corresponds to our notion of an access right for information $C.x$ granted to B by A.

3.2. Relationship-aware Access Control

We extend the formalism to support information relationships. An information relationship implies 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 $E_1.x_1 \otimes E_2.x_2 \otimes \ldots \longrightarrow D.x$ for expressing this relationship, and we want the axiom

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

to hold for certain conditions on the entities E_i, their information $E_i.x_i$, and the entities A_i and C. In Sections 3.2.1, 3.2.2, and 3.2.3, 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 the principal owning the information on the right-hand side should be able to establish a relationship[2], or

$$(3) \quad \begin{aligned} &\vdash (D \textbf{ says } (E_1.x_1 \otimes E_2.x_2 \otimes \ldots \longrightarrow D.x)) \\ &\supset (E_1.x_1 \otimes E_2.x_2 \otimes \ldots \longrightarrow D.x). \end{aligned}$$

We are now going to look at useful instances of Axiom 2.

[1] Lampson et al. show that principals speaking for Alice can also establish such a statement.

[2] Similar to the speaks-for case, it is possible to show that the information relationship can also be established by principals speaking for the principal owning the information.

3.2.1 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.private $\longrightarrow$ Alice.location denotes that information Alice.location is bundled in information Alice.private, that is, if someone has access to Alice's private 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

$$(4)\ \vdash (E_1.x_1 \longrightarrow D.x \land B \xRightarrow{E_1.x_1} A_1) \supset (B \xRightarrow{D.x} A_1).$$

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

$$\vdash (E_1.x_1 \longrightarrow D.x \land B \xRightarrow{E_1.x_1} E_1) \supset (B \xRightarrow{D.x} D).$$

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

Our formalism 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 some 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 (e.g., OWL-based), to which individuals can subscribe. We have extended our formalism accordingly. However, due to space reasons, we refrain from discussing this extension in detail.

3.2.2 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 $A_1 = E_1$, $A_2 = E_2$,..., and $C = D$, or

$$\begin{aligned}&\vdash (E_1.x_1 \otimes E_2.x_2 \otimes \ldots \longrightarrow D.x\\ &\land B \xRightarrow{E_1.x_1} E_1 \ \land B \xRightarrow{E_2.x_2} E_2 \land \ldots)\\ &\supset (B \xRightarrow{D.x} D).\end{aligned}$$

The second option is $A_1 = A_2 = \ldots = C$, 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.

3.2.3 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, such as 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 granularities is tedious and not intuitive. Instead, we observe that granularity-based access rights to information can be represented as 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., (Facilities_Manager, Wean_Hall_8220)) is itself an entity about which there is some information (such as its granularity). Therefore, it should be possible to associate information relationships with constraints. (Constraints are not a new concept in access control. For example, an access right can be constrained to be valid only during office hours.) Expressing granularity as a constraint, we can now reformulate the information relationship above as follows: (?result serves as a placeholder for the returned information.)

Alice.location [?result.granularity = fine]
$\longrightarrow$ Alice.location [?result.granularity = coarse]

This approach still requires Alice to establish a relationship. Realizing that the information items on the left-hand side and on the right-hand side are identical (i.e., $E_1.x_1 = D.x$ for Axiom (4)), we do not need an explicit relationship. Instead, we incorporate it directly into an access right. For example, the following statement says that Bob has access to Alice's location at fine or coarse granularity: (For readability reasons, we put constraints under the arrow, the statement remains a speaks-for operator.)

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

3.3. Example

In this section, we demonstrate the application of our formalism 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 sensitive information.

Alice bundles fine-grained location information (and potentially other) information in her private information (Statement (1) in Table 1). She grants Carol access to her private 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 if a client had access rights to the 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. Proof Building

In our distributed access control architecture, we have a client ship a proof of access to a service. This approach has been investigated in previous work [2, 15]. Our implementation is based on Howell and Kotz's framework [15], 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. Figure 2 illustrates a structured proof.

As shown in Section 3.2, for each type of information relationship, Axiom (2) is instantiated in a different way. Another characteristics that differentiates the relationships from each other is how they are used in proof building. We now discuss proof building for each relationship.

4.1. 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 statements derived from access rights or from other speaks-for statements in a graph where nodes represent principals and edges represent some information. For a request, the algorithm traverses the graph in a breadth-first

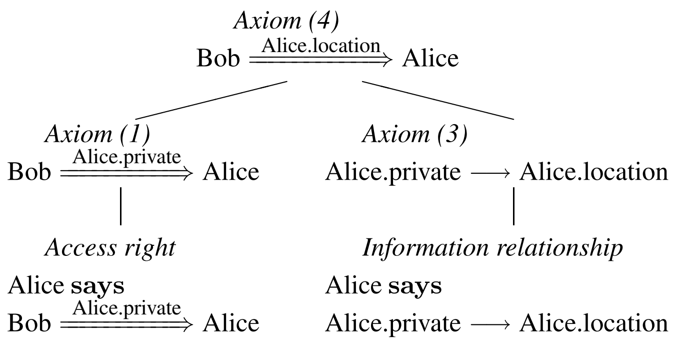
Axiom (4)
Bob $\xRightarrow{\text{Alice.location}}$ Alice
Axiom (1)
Bob $\xRightarrow{\text{Alice.private}}$ Alice
Axiom (3)
Alice.private $\longrightarrow$ Alice.location
Access right
Alice **says**
Bob $\xRightarrow{\text{Alice.private}}$ Alice
Information relationship
Alice **says**
Alice.private $\longrightarrow$ Alice.location

Figure 2. Structured proof. The proof shows that Bob can speak for Alice in terms of her location information, based on an access right for Alice's private information and an information relationship between Alice's private and her location information.

way to find a proof of access, starting at the owner of the requested information and ending at the channel (e.g., an SSL connection) that issues a request to a service on behalf of the client. 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 exploration of multiple relationships.) If so, the algorithm accepts the candidate edge and the edges connected to this edge become new candidate edges. Figure 3 illustrates this algorithm.

The algorithm looks at each edge at most once. Therefore, if there are n speaks-for statements 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 5.3.

Table 1. Example statements.

(1)	Alice **says** Alice.private $\longrightarrow$ Alice.location [?result.granularity $\geq$ fine]
(2)	Alice **says** Carol $\xRightarrow{\text{Alice.private}}$ Alice
(3)	Alice **says** Dave $\xRightarrow[\text{?result.granularity = coarse}]{\text{Alice.location}}$ Alice
(4)	Bob **says** Carol $\xRightarrow[\text{?result.granularity} \geq \text{fine}]{\text{Bob.location}}$ Bob
(5)	Location_Service **says** Alice.location [?result.granularity = fine] $\otimes$ Bob.location [?result,granularity = fine] $\longrightarrow$ (Location_Service, Wean_Hall_8220).people

$$\begin{tikzcd} A.y \arrow[r] & A.x \end{tikzcd}$$

$$\begin{tikzcd} C \arrow[r, Rightarrow, "A.y"'] \arrow[r, dotted, bend left=60, "A.x"] & B \arrow[r, Rightarrow, "A.x"'] & A \end{tikzcd}$$

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

4.2. 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 how 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 it without leaking location information about Alice and Bob. We present a solution for this problem in related work [13].

4.3. Granularity-based Relationships

For granularity-aware access control, the algorithm introduced in Section 4.1 must be extended to take constraints on access rights and information relationships into account. For example, Carol $\xRightarrow[\text{?result.granularity} \geq \text{fine}]{\text{Alice.location}}$ Bob and Bob $\xRightarrow[\text{?result.granularity = coarse}]{\text{Alice.location}}$ Alice can be concatenated to Carol $\xRightarrow[\text{?result.granularity = coarse}]{\text{Alice.location}}$ Alice. Since granularity-based relationships can be directly encoded in access rights instead of requiring separate information relationships, there will be no extra information relationships, which reduces the complexity of proof building.

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, we can use more sophisticated theorem provers. For example, Bauer et al. [2] employ the Twelf logical framework [22].

5. 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.

5.1. Deployment Environment

We use the Contextual Service Interface [17] developed for the Aura pervasive computing project [11] as a testbed for the implementation and deployment of our proposed access control mechanism. The framework is implemented in

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

(bundling-relationship
    (version ``1'')
    (issuer
        (information (pub_key:alice) alice location))
    (subject
        (information (pub_key:alice) alice private)))
```

Figure 4. Extended SPKI/SDSI certificates. Shown are an access right and an information relationship. `pub_key:foo` stands for foo's public key. The tag section can list constraints. (Digital signatures are omitted.)

Java. It stores access rights and information relationships as extended SPKI/SDSI digital certificates [9]. We give two examples in Figure 4. Proofs of access are implemented as Java classes. Each axiom in Section 3 has its corresponding class. A class is able to serialize and deserialize its contents. A service accepts only types of proofs it knows how to validate, that is, the methods of such proof classes are not transmitted. SSL [23] provides peer authentication and confidentiality and integrity of transmitted messages.

5.2. 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 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 Figure 5 shows the overall number of access rights and relationships for different values of l ($k = 50$ and $m = 3$).

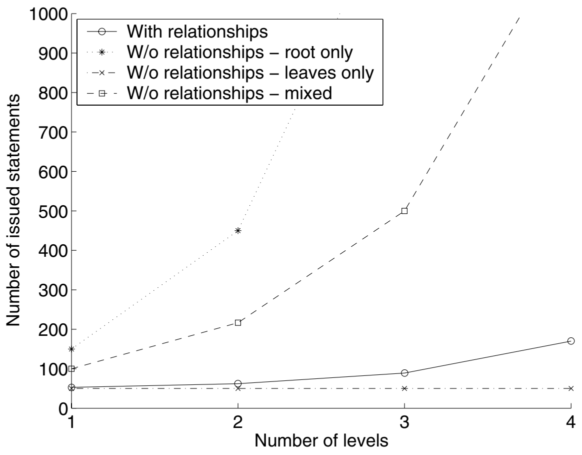

Figure 5. Number of issued statements. We compare the number of issued access rights and relationships in a world with relationships to the number of issued access rights in a world without relationships, for different levels of bundling-based relationships and distributions of clients in this hierarchy.

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 Figure 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 Figure 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 a filesystem or digital pictures in a photo album. 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 available levels of granularities. Combination-based relationships also reduce the number of access rights since they prevent a service from having to issue separate access rights for complex information.

5.3. Proof Building Time

We examine the cost of proof building, as explained in Section 4. Our experiment consists of locating a set of speaks-for statements in a pool of statements and to assemble them in a proof of access. We consider up to five sequentially connected, 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.

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. The results that we present are consistent with results for other paths. 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 ten times. Figure 6 reports the mean proof building time. According to Section 4, 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 Figure 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 5.4, the cost of building a proof is comparable to the cost of processing a request. We also observe that the numbers presented in Figure 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

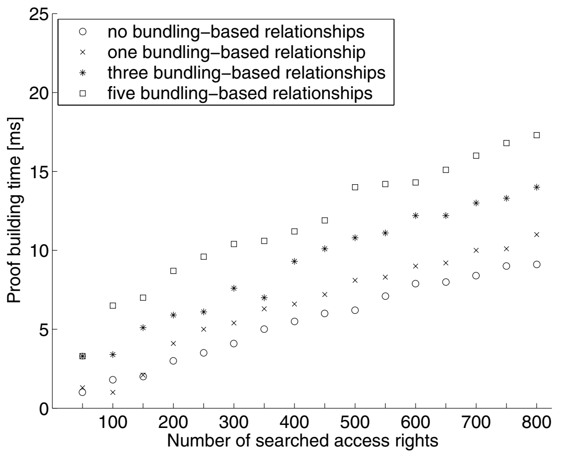

Figure 6. Proof building time. The graph shows the mean proof building time for different numbers of bundling-based relationships. For a fixed number, there is a linear increase in the number of searched access rights.

(e.g., an access right and a statement declaring that a channel speaks for a 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.

5.4. Client Response Time

We study the influence of bundling-based relationships on client response time in our prototype implementation. As mentioned in Section 4, the other types of relationships do not have a negative influence on complexity. Our asymmetric crypto operations required for setting up SSL connections and validating the signatures of certificates employ 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.

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. Access control takes only a few milliseconds, the main cost is validating the signature

Table 2. Client response time. Mean and standard deviation of elapsed time for security operations (in bold) and for gathering information [ms].

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

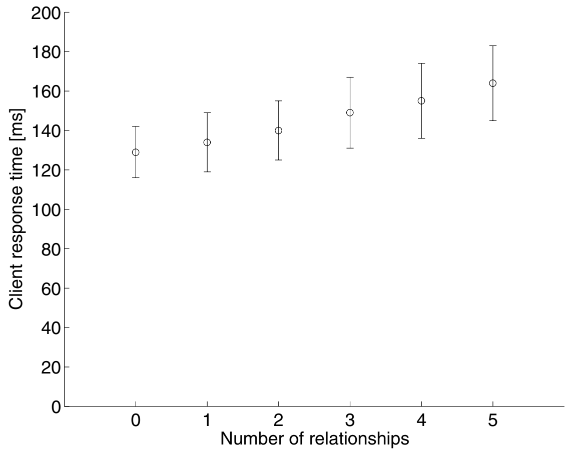

Figure 7. Client response time. The elapsed time increases linearly with the number of bundling-based relationships.

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. Access control 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 connected, bundling-based relationships. The results are in Figure 7. Note that the scaling of the y-axis is different from the scaling in Figure 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.

6. Related Work

There are several frameworks for pervasive computing environments that support access control to sensitive information [1, 5, 6, 10, 16, 20]. Al-Muhtadi et al. [1], Chen et al. [5], Covington et al. [6], and Gandon and Sadeh [10] 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 bottleneck and needs to be fully trusted by all entities. In addition, these solutions store access rights in a centralized knowledge base, which is an attractive target for attackers. This approach is also 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 [16] and Minami and Kotz [20] 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 [7], which allows Web servers to express their privacy practices. Chen et al. [5] exploit REI [18], which is targeted at pervasive computing environments. XACML [12] 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 SPKI/SDSI, they have no builtin mechanisms for verifying the authenticity of a statement, which is essential when delegating access rights across multiple environments. (For languages developed in the context of the Semantic Web, such as REI, Named Graphs [4] are a possible solution.)

Similar to Howell and Kotz, Bauer et al. [2] 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 [9].

7. Conclusions and Future Work

Access control in pervasive computing environments needs to take relationships between information into account. We proposed to make such relationships first-class citizens in access control. This way, it becomes straightforward to control access to complex 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.

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. Its performance is competitive. Based on the complexity analysis of proof building and its measurement-based validation, we anticipate small cost for building proofs in future pervasive computing environments.

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.

Acknowledgments

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

References

[1] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and 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)*, pages 489–496, March 2003.

[2] L. Bauer, M. A. Schneider, and E. Felten. A General and Flexible Access-Control System for the Web. In *Proceedings of 11th Usenix Security Symposium*, pages 93–108, August 2002.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2002.

[4] J. J. Carroll, C. Bizer, and P. Stickler. Named Graphs, Provenance and Trust. Technical Report HPL-2004-57, HP Labs, 2004.

[5] H. Chen, T. Finin, and A. Joshi. Semantic Web in the Context Broker Architecture. In *Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*, pages 277–286, March 2004.

[6] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad. A Context-Aware Security Architecture for Emerging Applications. In *Proceedings of 18th Annual Computer Security Applications Conference (ACSAC 2002)*, December 2002.

[7] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. W3C Recommendation, April 2002.

[8] M. Dean and G. Schreiber. OWL Web Ontology Language Reference. W3C Recommendation, February 2004.

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

[10] F. Gandon and N. Sadeh. A Semantic eWallet to Reconcile Privacy and Context Awareness. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, October 2003.

[11] 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.

[12] S. Godik and T. Moses. eXtensible Access Control Markup Language (XACML) Version 1.0. OASIS Standard, February 2003.

[13] U. Hengartner and P. Steenkiste. Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information. Technical Report CMU-CS-04-172, Computer Science Department, Carnegie Mellon University, October 2004.

[14] J. Howell and D. Kotz. A Formal Semantics for SPKI. In *Proceedings of 6th European Symposium on Research in Computer Security (ESORICS 2000)*, pages 140–158, October 2000.

[15] 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.

[16] X. Jiang and J. A. Landay. Modeling Privacy Control in Context-Aware Systems. *IEEE Pervasive Computing*, 1(3):59–63, July-September 2002.

[17] 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.

[18] L. Kagal, T. Finin, and A. Joshi. A Policy Language for a Pervasive Computing Environment. In *Proceedings of 4th International Workshop on Policies for Distributed Systems and Networks*, pages 63–76, June 2004.

[19] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in Distributed Systems: Theory and Practice. *ACM Transactions on Computer Systems*, 10(4):263–310, November 1992.

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

[21] G. Myles, A. Friday, and N. Davies. Preserving Privacy in Environments with Location-Based Applications. *Pervasive Computing*, 2(1):56–64, January-March 2003.

[22] F. Pfenning and C. Schürmann. System Description: Twelf - A Meta-Logical Framework for Deductive Systems. In *Proceedings of 16th International Conference on Automated Deduction (CADE-16-99)*, pages 202–206, July 1999.

[23] C. Systems. PureTLS. `http://www.rtfm.com/puretls/`.