Access Control to Information in Pervasive Computing Environments

Urs Hengartner

July 15, 2005
NSH 3305 at 3pm (EDT)

Committee
Peter Steenkiste (Chair)
Adrian Perrig
Michael K. Reiter
Edward W. Felten, Princeton
Abstract

Pervasive computing envisions a world in which our environment is full of embedded devices that gather and share vast amounts of information about people, such as their location, activity, or even their feelings. Some of this information is confidential and should not be released to just anyone. In this thesis, I show how existing solutions for controlling access to information are not sufficient for pervasive computing because of four challenges: First, there will be many information services, potentially offering the same information, run by different organizations, even in a single social environment. Second, there will be complex types of information, such as a person’s calendar entry, which reveal other kinds of information, such as the person’s current location. Third, there will be services that derive specific information, such as a person’s activity, from raw information, such as a videostream, and that become attractive targets for intruders. Fourth, an individual’s ability to access information could be constrained based on confidential information about the individual’s context.

This thesis presents a distributed access-control architecture for pervasive computing that supports complex and derived information and confidential context-sensitive constraints. In particular, the thesis makes the following contributions: First, I introduce a distributed access-control architecture, in which a client proves to a service that the client is authorized to access requested information. Second, I show how to incorporate the semantics of complex information as a first-class citizen into this architecture, based on information relationships. Third, I propose derivation-constrained access control, which reduces the influence of intruders by making a service prove that the service is accessing information on behalf of an authorized client. Fourth, I study the kinds of information leaks that context-sensitive constraints can cause. I introduce access-rights graphs and hidden constraints for avoiding these leaks. Fifth, I show how pervasive computing makes it difficult for a client to prove that the client is authorized to access complex confidential information. I propose a cryptographic solution based on an extension of hierarchical identity-based encryption. Sixth, as an alternative approach, I introduce an encryption-based access-control architecture for pervasive computing, in which a service gives information to any client, but only in an encrypted form.

I present a formal model for my contributions based on Lampson et al.’s theory of authentication. All of my contributions have been implemented in an actual pervasive computing environment. A performance analysis of my implementation demonstrates the feasibility of my design.
1 Introduction

Pervasive computing envisions a world in which there are hundreds of wireless computing devices available to a person. These devices will be embedded in the environment so that people will use them without thinking about them [21]. This infrastructure enables the continuous gathering and sharing of information about people. For example, there will be sensors for monitoring people’s location, activity, health status, or even feelings.

The vast amounts of personal information gathered in pervasive computing makes privacy a major concern [14, 15]. Most personal information is confidential, that is, not just anyone should be able to access it. Therefore, a pervasive computing environment must control access to confidential personal information.

1.1 Information Privacy in Pervasive Computing

To get a better understanding of the challenges for access control to confidential personal information in pervasive computing, let us look at an example scenario. The scenario illustrates how personal information is gathered, provided, and used in pervasive computing. Figure 1 presents the providers of information that participate in the scenario. We use the term service for such a provider. In the scenario, there is an individual, Carol, who works for company ACME. She uses a centralized calendar application, maintained by her employer, for managing her calendar. For example, she schedules a meeting with Bob that will take place in her office. Carol lets Alice access her current calendar entry. However, Carol limits this access such that Alice will be granted access only if Carol is not busy and if Carol is currently on the company premises, but not if Carol is,
for example, at home or at Starbucks. Therefore, whenever Alice wants to access Carol’s current calendar entry, the access control mechanism needs to retrieve Carol’s current location and activity information from services offering this information and validate the restrictions imposed by Carol. Multiple services, exploiting different technologies, provide Carol’s location information. For example, ACME deploys cameras in the company’s meeting rooms and provides a wireless network. Therefore, ACME can locate Carol by detecting her face in a camera’s videostream or by finding the access point to which Carol’s Wi-Fi laptop is connecting. Furthermore, Carol’s cellphone company, Nextel, makes her location information available. Finally, while Alice is at home or at Starbucks, her and Starbucks’ wireless network, respectively, also offer her location information. Carol’s activity information is also provided by multiple services. For example, one of the services derives this information from information gathered by body sensors worn by Carol. Our scenario reveals four interesting properties of information in pervasive computing:

- Information will be offered by multiple services, potentially run by different organizations. For example, while Carol is at work, both Carol’s cellphone company and her employer provide her location information. Also, different social environments might exploit different services. While Carol is at work, her employer’s location service will provide her location information. While she is at home, her own location service will offer this information. In both environments, the location service of her cellphone company could provide additional location information.

- Certain types of information will be complex. By complex information, we mean information that contains multiple types of information. For example, Carol’s calendar entry about her meeting with Bob in her office reveals information about Carol’s current location and activity. In addition, the entry also gives away Bob’s current location and activity.

- Information will be processed and will change its nature while flowing through multiple services to a client asking for information. For example, one of the location services retrieves a videostream from a camera, which is also a service, detects Carol’s face in the stream, and determines Carol’s current location. Another location service derives Carol’s location from her calendar entry, provided by the calendar application.

- Information will be retrieved from a service not only for handing it over to a client asking for this information (or for information derived from this information), but also for constraining access to information that the client asked for. For example, Carol’s location information is used for constraining Alice’s access to Carol’s calendar information.

Access control to information has been well investigated in the context of information stored in distributed and remote filesystems (such as AFS with Kerberos, SFS [16], or Microsoft Windows 2000 Server) or in relational databases (such as System R [7], Oracle Database, or databases exploiting role-based access control). However, the properties of information that we just discussed make access control in pervasive computing challenging and applying existing access-control techniques could easily result in information leaks [8]. Let us take a closer look at the challenges and how they have been traditionally addressed for filesystems or databases. We will summarize each challenge in a lesson for access control to information in pervasive computing.
1.1.1 Diversity in Service Administration

Services that provide information within a social environment can be administrated by different organizations. For example, while Carol is at work, her cellphone company offers her location information and the calendar application run by her employer provides her calendar information. Since services are run by different entities, we must ensure that no personal information leaks from one entity to another entity. For example, assume that the calendar application contacts the location service run by Carol’s cellphone company in order to ensure that Carol is currently on the company premises, as required by Carol when she granted Alice access to her current calendar entry. Here, the cellphone company must validate that the calendar application (i.e., the organization running this service) has access to Carol’s location information. In contrast, within a specific social environment, a distributed filesystem or a database is typically deployed by a single organization on a set of trusted machines communicating over secure channels. In this setting, access control is required only at the interface between the trusted machines and entities accessing information provided by the filesystem or database. (Such an entity is typically a machine or an application that accesses information on a user’s behalf.) There is no need to run access control (other than authentication) when file servers exchange information or while information is processed in a distributed database.

Not only can there be services run by different organizations in a single social environment, there can also be multiple social environments. If people transition between multiple environments, this transitioning should be seamless. For instance, it should not be necessary for an individual to decide whom to grant access to her location information for each environment. Instead, the person should be able to make a single decision that applies to any environment in which she takes part. In the case of filesystems or databases, these decisions are typically coupled to the organization that provides information in an environment and sharing the same decision across environments is not possible. For example, in a remote filesystem, such as AFS, an individual needs to make separate decisions for each set of directories or files bound to a particular organization that is part of the filesystem. An individual can have a home directory both at Princeton and at Carnegie Mellon, but she cannot make a single decision that applies to her private files or directories both at Princeton and at Carnegie Mellon.

The lesson from this challenge is that, in pervasive computing, access control needs to be able to deal with services run by different entities, while making it easy for individuals to manage access to their personal information provided by these services.

1.1.2 Complex Information

Complex information lets someone observing this information derive multiple types of information about one or multiple individuals. For example, a calendar entry reveals location and activity information. Similarly, a videostream delivered by a camera can reveal the location and activity of the people shown in the videostream. If a client accessing this complex information is not authorized to access the derivable information, there will be an information leak and a person’s privacy will be invaded.

Information leaks are also possible in filesystems or databases. Avoiding such information leaks in a filesystem is hard since (current) filesystems are not aware of the contents of a file; access control takes only the name of a file into account. A coarse-grained way for an individual to protect a file that contains complex information (e.g., a schedule) is to keep the file private and to not put anyone else on the access control list (“ACL”) of the file. Obviously, this solution does not
work for pervasive computing, where information needs to be accessible in order to serve people. Alternatively, the individual could carefully establish the ACL such that there are no information leaks. That is, a client is put on the ACL only if the client has access to all the information revealed by the complex information in the file. However, this process is tedious and error prone and could lead to consistency problems. For example, consistency problems arise if the revocation of a client’s access to information revealed by the complex information stored in the file does not result in the removal of the client from the ACL of the file.

Similarly, a database can provide complex information in the rows of a table (or of a view derived from tables). Database access control is based on the name of a table (or view) and can be row based [19]. To avoid information leaks, the entity that grants access to a row must be aware of the clients that have access to the information stored in the row and must carefully grant them access to the row. However, this approach suffers from the same problems as the ones mentioned in the filesystem scenario.

The lesson from this challenge is that, in pervasive computing, access control itself needs to be aware of the contents of complex information and treat this content as a first-class citizen when making an access decision.

1.1.3 Derivation of Information

In pervasive computing, there will be services that derive information. In particular, these services will take input information and derive output information. Input information can be raw, such as a videostream, audiostream, or a person’s heart rate, and potentially reveal information in addition to the output information derived by a service. For example, while a service can derive location information from a videostream, it might also be possible to derive activity or health information from the stream. If a service that derives information has access to raw information, it will become an attractive target for attackers. Namely, an intruder into the service will be able to access the raw information.

In a filesystem, there is no processing of information. In a database, information can be processed (e.g., a view can contain processed information). However, as mentioned in Section 1.1.1, the processing typically takes place within the trusted environment. As a result, there is no need for securing this processing since an intruder into the trusted environment will have access to the input information anyway.

The lesson from this challenge is that, in pervasive computing, a service’s access to input information should be limited such that the service can still perform its derivation functionality, but without giving an intruder into the service complete access to the input information.

1.1.4 Confidential Context-Sensitive Constraints

The availability of an individual’s context-sensitive information (such as her current location or the current time) makes it possible to exploit this information for context-sensitive access control. For example, in our example scenario, Carol grants access to Alice only if Carol is not busy and if Carol is at a particular location. However, exploiting confidential context-sensitive information in a constraint can result in information leaks. In particular, by observing the fate of a request

---

1 We ignore column-based access control, where the values in some columns are masked, since we are interested in access control to complex information as a whole.
for information that is accessible only under a set of context-sensitive constraints, an observer that knows about the nature of these constraints will learn whether they are satisfied. If any of these constraints involve confidential information to which the observer does not have access, there will be an information leak. For example, if Alice is granted access to Carol’s calendar entry, she will learn that Carol is on the company premises.

Current filesystems or databases are limited in the types of definable constraints or do not exploit confidential context-sensitive constraints. For example, filesystems or databases typically limit constraints to membership in a group or role, respectively. For some databases [19], it is possible to constrain access based on context-sensitive information, such as the current time. However, this information is not confidential and knowing it does not result in an information leak. While it is possible to incorporate more flexible context-sensitive constraints into database access control, this option has received limited interest because of the missing availability of context-sensitive information (such as location information). However, pervasive computing is going to make such information much more widely available.

The lesson from this challenge is that, in pervasive computing, access control needs to support context-sensitive constraints, but without leaking the potentially confidential information listed in such a constraint.

1.2 Related Work

In this thesis, we present an access-control architecture that solves the challenges discussed in the previous section. Before we outline our approach, let us review the solutions chosen by other research projects in pervasive computing. We will re-visit each of the challenges and will see that none of them has been addressed sufficiently in previous work.

1.2.1 Diversity in Service Administration

Many pervasive computing projects [1, 4, 5, 18, 22, 20, 13] rely on the existence of a centralized entity (e.g., “inference engine” [1], “specialized server entity”[4], “security agent” [13], or “middleware service”[18]) that knows about all the services in an environment and that hides the fact that multiple services can provide the same type of information from individuals. The centralized entity is defined as being trusted by all the services to run access control on their behalf and as being able to retrieve information from any service.

The advantage of this approach is that it prevents individuals from having to be aware of each service. In particular, individuals do not have to decide for each service whom to grant access to the information provided by the service.

However, this approach has several drawbacks. The existing projects typically assume that all services within an environment are run by the organization associated with the environment. However, this assumption does not necessarily hold in practice. For example, in our example scenario, Carol might want to use the location service provided by her cellphone company regardless of the environment that she is participating in. In addition, many research projects have been deployed only in the context of a single environment; it is not clear how they would be deployed in multiple environments. Furthermore, while the approach prevents an individual from making service-dependent decisions about whom to grant access, the individual still has to decide for each type of information whom to grant access. This approach will not scale if there are lots of different types of information.
There are a few projects that are not centralized [17, 11, 6]. However, these projects assume that an individual controls all the services that provide information about her [6, 11], which is not the case in practice, or they do not address the scalability problem [17], where multiple services offer the same information.

1.2.2 Complex Information

Many existing pervasive computing projects initially neglected access control to information and focused on “Smart Rooms” [13] and on controlling access to physical resources available in such a room, such as a projector or a printer [1, 5, 13]. However, the upcoming of the Semantic Web [3] has also triggered interest in access control to information. The goal of the Semantic Web is to develop languages that are adequate for representing and reasoning about the semantics of information on the Web. Several projects [4, 6] have exploited techniques from the Semantic Web for representing and reasoning about individuals’ privacy preferences and about resources in a pervasive computing environment. However, while it has been recognized that complex information can cause information leaks [4], the problem has not been addressed in a deployed architecture.

1.2.3 Derivation of Information

As discussed in Section 1.2.1, many existing pervasive computing projects rely on a centralized approach for running access control. Therefore, in terms of dealing with derived information, they suffer from the same problem as traditional solutions for access control in a filesystem or database, as explained in Section 1.1.3. In particular, the derivation takes place within the trusted environment. There is no sense in securing this processing since an intruder into the trusted environment will have access to all the information anyway. Projects that are not centralized and that address information derivation [17] do not suffer from this drawback. However, it is still possible for an intruder into a service to get access to any information that the service is authorized to access. Ideally, we can limit the amount of information that an intruder can access.

1.2.4 Confidential Context-Sensitive Constraints

In existing pervasive computing projects, it is typically possible to constrain an individual’s access to a resource based on the individual’s context, such as her location or the current time. However, most projects ignore information leaks caused by such context-sensitive constraints. While there has been some research in this area [17], the existing research considers only a small subset of the information leaks that constraints could cause in pervasive computing.

1.3 Thesis Goal

The challenges and the related work discussed in Sections 1.1 and 1.2, respectively, raise the following central question:

\[
\text{Is it possible to run access control to information in pervasive computing, where this information can be complex or derived and where access decisions might be constrained based on confidential information, without relying on a centralized entity?}
\]

In this thesis, we answer this question affirmatively. We build on the following key techniques:
**Client-based access control.** Since services in pervasive computing environments will be administered by different organizations, we cannot expect these services to trust a single centralized entity to run access control on their behalf. Instead, we adopt a distributed approach, where services themselves control access to their provided information, assisted by clients. In particular, a client needs to prove to a service that the client is authorized to access the requested information, based on the possession of one or multiple digital certificates. Such *client-based* access-control architectures are not a new concept. However, existing implementations [2, 12] of the concept are not targeted at pervasive computing and do not deal with the challenges mentioned in Section 1.1. In addition, for complex information, they can leak information when a service informs a client of the required proof of access. Our architecture is based on a formal model for access control in a distributed system.

**Flexible information representation scheme.** Since there will be many services offering information and many different types of information, we need to minimize the number of decisions that an individual needs to make when deciding about whom to grant access to her personal information. We introduce an information representation scheme that allows an individual to grant service-independent and environment-independent access to her personal information and to re-use the same decision for multiple types of information.

**Semantics of information.** Since complex information can reveal multiple types of information, access control must be aware of the semantics of information. We formalize the semantics of information and incorporate these semantics as a first-class citizen into a formal model of access control. In particular, our formal model supports access control to complex information and to derived information. Based on this model, we make a client-based access-control architecture for pervasive computing aware of the semantics of the information to which access is controlled.

### 1.4 Contributions

All the contributions of this thesis have been implemented in an actual pervasive computing environment and evaluated with measurements or analytically. In particular, this thesis makes the following key contributions:

- We present a client-based access-control architecture for pervasive computing that is powerful enough to deal with complex or derived information and to make context-sensitive access decisions.

- We introduce the concept of *information relationships*, which capture the semantics of complex information, and incorporate information relationships into a client-based access-control architecture. Information relationships have the additional benefit that they can prevent an individual from having to decide for each type of information whom to grant access.

- We present the concept of *derivation-constrained access control*, which enables a client-based access-control architecture to take derivation properties of information into account for access control. In this way, the architecture can limit the influence of intruders into a service.
• We present the concepts of access-rights graphs and hidden constraints, which support the identification of potential information leaks in context-sensitive access control and which prevent context-sensitive information from leaking to services, respectively. We also incorporate the concepts into a client-based access-control architecture.

• We extend client-based access control by obscured proof-of-access descriptions. Such obscured descriptions avoid information leaks that are possible when a service that provides complex information informs a client of the required proof of access.

• As an alternative option to distributed access control based on proofs of access, we present an encryption-based access-control architecture for pervasive computing. In this architecture, a service grants any client access to its information, but the service encrypts the information before giving the information to a client.

2 Information Relationships

In pervasive computing, there will be a multitude of services that provide potentially confidential 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. In addition, a person might be a member of multiple environments over time. Therefore, having an individual issue access rights per client to be granted access, per service, per environment, and per type of information is not scalable. In this section, 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. Information relationships are also useful for controlling access to complex information, such as a calendar entry that reveals location and activity information, since they can capture the semantics of this information [10].

Information relationships reduce the number of access rights that an individual needs to define. 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 and grant access rights to information bundles. 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. 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. If the current entry in Carol’s calendar says that she is having a meeting with Bob, only people who are at least allowed to access Carol’s and Bob’s location and activity information should have access to the calendar entry. To implement this rule, Carol should grant Alice an access right to this entry only if Alice already has access rights to Carol’s and Bob’s location and activity information. However, this is tedious and might lead to consistency problems if Bob revoked an access right. Instead, access control should be aware of the semantics of information and know that calendar information contains location and activity information. We can use information relationships to capture the semantics. In the example, calendar information is related to location and activity information and access control should be able take this relationship into account.
In this work, we make a client-based access-control 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. For more complex information relationships, it is possible to employ a centralized rule engine to reason about information. In particular, our contributions include

- the concept of information relationships as a first-class citizen in a client-based access-control architecture,
- a formal model for incorporating relationships into access control,
- a prototype implementation based on this model, and an evaluation of this implementation.

3 Derivation-Constrained Access Control

Access control ensures that only authorized clients get access to confidential information provided by a service. Often, when a client issues a request for information provided a service, this service acts as a gateway and issues further requests for information to another service, acting as an endpoint. For example, a people location service can issue a request for the location of a device when receiving a request for the location of a person. A service providing health information might issue queries for a person’s medical information (heart rate, blood pressure,..), as delivered by wearable sensors, when receiving a query for a person’s health. There could also be multiple gateways between the client and the endpoint(s).

The level of indirection makes access control difficult. On the one hand, a gateway needs access rights in order to retrieve information from other services. On the other hand, granting access rights to a gateway makes the gateway vulnerable in case of an intrusion. Namely, an intruder into the gateway has access to any information that the gateway is authorized to get, and the intruder can actively issue requests for information to other services. Services will answer these requests if the gateway is authorized to access the requested information.

The conventional solution to this problem exploits short-lived access rights. In particular, when a client issues a request for information to a gateway, the gateway is granted short-lived access rights, which allow the gateway to temporarily retrieve the information required for answering the client’s request from a service. A drawback of this solution is its lack of flexibility. The solution is easy to implement if the entity granting access to the gateway and the client asking for information are identical, but it becomes more difficult if they are not. One of the difficulties is that the entity granting access might be offline upon a request and thus not be able to issue access rights.

In this work, we propose derivation-constrained access control, where an endpoint, before returning information to a gateway, requires the gateway to prove that the gateway is asking for this information in order to answer an authorized request for derived information sent to the gateway by a client. If there is no such request, an intruder into the gateway will not be able to prove that the gateway is asking for the information on a client’s behalf. Therefore, the endpoint will not grant the gateway (and the intruder) access to the requested information.

Let us illustrate derivation-constrained access control with an example application. There is a service that locates people and a service that locates laptops (see Figure 2). The latter service locates people indirectly by locating their laptops, that is, the people location service acts as a gateway and contacts the laptop location service upon a client’s request for a person’s location. Assume that
Bob asks the people location service for Alice’s location information. The people location service then asks the laptop location service for the location of Alice’s laptop. Using derivation-constrained access control, the laptop location service now makes the people location service prove that it is issuing this request in order to answer Bob’s authorized request for Alice’s location and that the location of Alice’s laptop should be used for deriving Alice’s location.

Derivation-constrained access control does not suffer from the drawback we observed for a solution based on short-lived access rights. Our approach requires the entity that grants access to the information provided by an endpoint to specify what information is derived from this information. However, there is no need to issue these specifications upon a request; they need to be issued only when there is a change in the derivation properties of information. Therefore, the entity does not need to be online when a request is made. Our contributions are

- the concept of exploiting derivation properties of information as a first-class citizen in a client-based access-control architecture,
- a formal model for incorporating derivation properties into access control,
- a prototype implementation based on this model, and an evaluation of this implementation.

4 Confidential Context-Sensitive Constraints

Access control ensures that only people that have an access right to confidential information can access this information. Access rights can be constrained, that is, they are valid only if certain conditions are met. For example, an access right can state that access is granted only during office hours. Pervasive computing makes it possible to constrain access rights based on a person’s context information. For example, an access right can declare that access should be granted to an individual only if she is at a particular location. However, context-sensitive constraints can lead to information leaks. In this work, we study the information leaks that context-sensitive constraints can cause and examine how to support context-sensitive constraints without leaking confidential information.

Let us look at three examples to illustrate potential information leaks caused by context-sensitive constraints. In our first example, information leaks to a service. Assume that Carol lets people standing in front of her office see her current calendar entry; a cellphone service provides people’s location information, and a centralized calendar system offers Carol’s calendar information. Given this setup, when Bob asks the calendar service for Carol’s calendar entry, the calendar service could learn Bob’s location while running access control, either by querying the location service directly or by being told that the constraint in Carol’s access right is fulfilled. Therefore, Bob’s location information could leak to the calendar service (i.e., to the organization running this service).

In the second example, information leaks to a person who asks for information. Assume that Carol allows people to access her calendar entry if she is in her office. Therefore, if somebody is
able to retrieve this entry, he can infer that Carol is in her office. Somebody planning on breaking into Carol’s house would happily take advantage of this information leak.

In our third example, information leaks to a person who hands out access rights. Assume that Carol grants herself an access right to her calendar entry under the condition that Alice is at a particular location. When the calendar system grants Carol access to her entry, she will learn Alice’s location, which could be an information leak.

Related work has largely ignored information leaks caused by constraints on access rights to information. In this work, we examine how constraints on access rights can cause information leaks and how to avoid these leaks. In particular, our contributions include

- a systematic investigation of different access-control approaches for information leaks caused by constraints,
- the organization of access rights and of constraints on them in a directed graph, which simplifies the detection of conflicting constraints and of information leaks and the resolution of constraints,
- the concept of hidden constraints, which makes it possible to keep constraints secret and thus reduces the danger of information leaks, and
- a formal model that supports constraints on access rights in distributed access control, a prototype implementation based on this model, and an evaluation of this implementation.

5 Obscured Proof-of-Access Descriptions

Traditional client-based access control assumes that either a client knows the nature of the required proof of access or that a service can inform the client of this nature. However, for complex information, this assumption no longer holds. If Carol’s calendar entry states that she is meeting with Bob, Alice needs to present a proof of access consisting of access rights to Carol’s location and activity information and to Bob’s location and activity information. However, Alice does not know that Carol is meeting with Bob, so she does not know the nature of the required proof of access. Similarly, the service cannot inform Alice that she has to present access rights to Bob’s information, since otherwise Alice could deduce that Carol is currently meeting with Bob, which could be an information leak. In this work, we present a solution for this problem in the context of pervasive computing, based on cryptography.

A naïve solution for a client that wants to access complex information, such as a calendar entry, is to have the client transmit a proof of access listing all potentially relevant access rights, such as all the client’s access rights to location and activity information. This solution has privacy and bandwidth issues: a service can learn a lot about a client, and a client might have to transmit a lot of data.

Our proposed solution has a service return a description of the required proof of access to a client, but the service obscures the description such that the client can interpret the description only if the client has some secret knowledge [9]. The client has this secret knowledge only if it has access to the information listed in the proof description. In particular, our contributions include

- a novel application of hierarchical identity-based encryption in the context of client-based access control,
• an extension of an existing scheme for hierarchical identity-based encryption to support multiple (hierarchical) constraints on access rights,
• a novel way for dealing with expiration in identity-based encryption, and
• the first implementation of a hierarchical identity-based encryption scheme, to the best of our knowledge.

6 Encryption-Based Access Control

So far, we have concentrated on client-based access control, where a client needs to prove to a service that the client is authorized to access the requested information. An alternative option is encryption-based access control. In such a scheme, a service provides confidential information to any client, but the service encrypts the information before handing it over to a client. Only clients authorized to access the information have the required decryption key. This approach is attractive in scenarios where there are lots of similar queries to a service since it shields the service from having to run client-specific access control. Instead, the service can encrypt an information item once and use the ciphertext for answering multiple queries for this item. In this work, we present an encryption-based access-control architecture for pervasive computing.

In Section 1.1, we outlined four challenges for access control in pervasive computing. Let us discuss how we can address these challenges in an encryption-based access-control architecture. First, to keep key management simple, we have a client use only one decryption key for each type of information, even if the same type of information is offered by multiple services, potentially run by different organizations. Second, for complex information, such as a calendar entry, a service offering this information encrypts the information such that a client requires multiple decryption keys, one for each type of revealed information. Third, in order to limit the influence of intruders into a gateway, a gateway can potentially perform its derivation functionality on encrypted information. In this way, a gateway does not need to be given a decryption key. Fourth, when information is accessible only if a constraint is satisfied, we have each value of the constraint require a separate decryption key. Constraints can make key management difficult.

In this work, we present an encryption-based access-control architecture for pervasive computing that supports constraints and multiple services offering the same information and that keeps key management simple [9]. In particular, our contributions include

• a novel application of hierarchical identity-based encryption in the context of encryption-based access control,
• a prototype implementation and an evaluation of this implementation.

7 Conclusions

This dissertation set out to address challenges in access control to information in pervasive computing environments. We presented a distributed access-control architecture that supports complex and derived information and confidential context-sensitive constraints. It is our hope that the ideas presented in this thesis will serve as a foundation for future work in the study of privacy in pervasive computing.
References


