Securing Context-Aware Applications Using Environment Roles[*]

Michael J. Covington[†], Wende Long, Srividhya Srinivasan
Anind K. Dey, Mustaque Ahamad, Gregory D. Abowd
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332 USA

ABSTRACT

In the future, a largely invisible and ubiquitous computing infrastructure will assist people with a variety of activities in the home and at work. The applications that will be deployed in such systems will create and manipulate private information and will provide access to a variety of other resources. Securing such applications is challenging for a number of reasons. Unlike traditional systems where access control has been explored, access decisions may depend on the context in which requests are made. We show how the well-developed notion of roles can be used to capture security-relevant context of the environment in which access requests are made. By introducing environment roles, we create a uniform access control framework that can be used to secure context-aware applications. We also present a security architecture that supports security policies that make use of environment roles to control access to resources.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access*; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; K.4.2 [**Computers and Society**]: Social Issues—*Abuse and crime involving computers*; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Design, Security, Standardization, Theory

Keywords

Role-Based Access Control, Context-Aware Computing

*This work was supported in part by National Science Foundation grant CCR-9988212 and the members of the Industrial Advisory Board of the Georgia Tech Broadband Institute.

†Contact Author: Michael.Covington@cc.gatech.edu

1. INTRODUCTION

As computers become pervasive in the home and community, new applications will emerge that will make daily life easier for people. Such applications, which will be enabled by a ubiquitous computing and communication infrastructure, will provide unobtrusive access to important information, resources and services. Clearly, successful deployment of such applications will depend on our ability to secure them. In particular, we will have to ensure that access to information and services is granted only to authorized users, without requiring them to deal with complex security policies or burdensome access control mechanisms.

The Aware Home initiative at the Georgia Institute of Technology is exploring several applications [12] in the home environment that have novel security needs. We have built a home with a rich computation and communication infrastructure. Sensors in the home can capture, process and store a variety of information about the home's residents and their activities. Such information could be sensitive and the residents certainly would want to control who has access to it. The applications could also allow users to control and manage resources in the home from a variety of locations. Access control will be important in environments like the Aware Home to protect the privacy concerns of the home's residents as well as to prevent unauthorized access to resources in the home. The fact that attacks on an always connected home can be mounted from anywhere and at anytime further motivates the need for secure access to resources in the home.

Many of the applications in the Aware Home are context-aware [15] and their behavior can be customized based on the environment in which an access request is made. For example, access to certain appliances may only be granted when the request is made from a certain location or at a certain time. In a smart intercom application that is being explored in the Aware Home [14], permission to talk to a resident in another room may depend on the activity in which the resident is currently involved. Access requests can also be triggered when the request is not explicitly made by a resident. One of the applications being explored aims to allow elderly residents to remain in their homes, instead of moving to assisted living facilities. If such a resident falls and injures himself, the Aware Home could detect the emergency and respond by requesting medical assistance. This access request would be automatically generated and approved based on the context of the situation. Other environmental conditions such as temperature in the home, the time of day, or location from which a request is made could also affect whether an access request is granted or denied.

In this paper, we explore an access control model for securing context-aware applications. Such a model should be flexible enough to support policies that make use of security-relevant environmental data to control access to resources or information manipulated by the applications. We show how a generalization of the role-based access control (RBAC) model [19, 7] provides an elegant means of capturing and using a user's context in access control. In particular, similar to subject roles of RBAC, we define *environment roles*, which can be used to capture security-relevant aspects of the environment in which an application executes. By using the uniform notion of a role to capture both user and environmental attributes, we develop an access control model that is easy to use and understand.

We precisely define environment roles and show that they share many properties with subject roles that have been explored in great detail. In particular, there could be a hierarchical structure between environment roles, and their activation and revocation leads to interesting problems. We also present an architecture based on the Context Toolkit that has been developed at Georgia Tech [5]. This toolkit provides abstractions for assessing environmental state which could be used to manage environment roles. This architecture addresses issues such as role activation and authorization based on environment roles. We also illustrate how context-aware applications can be secured when environment roles are used.

The remainder of this paper proceeds as follows: section 2 motivates why environmental state is important in defining access control for applications in the Aware Home. Section 3 presents details of environment roles, and is followed by a more precise description of an access model using such roles in section 4. The security architecture based on environment roles is discussed in section 5. Then, we present several applications and show how they can be secured in section 6. We discuss the benefits of environment roles in section 7 and compare them with related work in section 8. The paper is concluded in section 9.

2. SECURITY CHALLENGES IN THE AWARE HOME

The Aware Home applications alluded to in the previous section present new and interesting security challenges. Given the sensitivity of information that is generated and stored in such an environment, as well as the many complex interactions that will take place both within and outside of the Aware Home, security policies can potentially be quite complex. A policy can restrict access to information or resources based on several factors, including attributes about the subject, the resource or the environment. For example, subjects can be classified as "resident" or "guest", "adult" or "child," or even as "pet." Access rights then can depend on the subject's classification (e.g., "resident"), as well as on his or her identity. Access also may be restricted based on the subject's location, or based on environmental factors such as the temperature or the time of day. For example, a policy might say that a repairman has access to the refrigerator only while he is inside the home on January 17, 2000, between 8:00 a.m. and 1:00 p.m.

While time and location are natural examples of environmental state that could be used in access control, richer contextual information could also impact the result of an access request. For example, consider a smart intercom application that is configured to permit a child in one location of the house to request an intercom connection with the mother who is in the kitchen. The request for this may only be granted if the mother is not busy at the time (e.g., not involved in another conversation or activity). Also, unlike traditional access control models where requests are made explicitly by subjects, requests in the Aware Home may be generated based solely on the environmental conditions. For example, if the Aware Home detects that a resident has fallen and injured himself, a request for medical help can be generated and should be granted based on this context of the resident.

Although we have used the Aware Home to motivate how environmental state can be used in authorization, there are many other real-world instances in which an access control decision depends on the state of the environment at the time of the request. For example, many organizations restrict access to their facilities during nights and weekends. In the military, secure computer systems are often restricted only to personnel in designated physical areas, such as a highly secure computer room. In the home, parents might restrict their children's access to the television, allowing the kids to watch TV only after they have done their homework, and only until 9:00 p.m. In each of these instances, the access control policy depends on information from the *environment*. Any security-relevant information in the environment that can be accurately captured by the system can be used to restrict access to system resources.

An access control language can be developed that allows environmental state to be considered when access decisions are made. However, this could be complex because it must address what state is security-relevant and how it should be captured and used. This could also impact the ease with which security policies that used environmental state in access control can be defined and understood. We take another approach which makes use of the well-known notion of roles to capture security-relevant state. In particular, we define environment roles based on the context or state of the environment. Although other mechanisms, some of which are discussed in section 8, can be used to control access based on environmental conditions, we chose to use roles for this purpose and justify our decision throughout the remainder of this document.

3. ENVIRONMENT ROLES

Traditional RBAC offers an elegant solution to the problem of managing complex access control rule sets. The basis of RBAC is the concept of a *role*. Fundamentally, a role is a grouping mechanism that is used to categorize subjects based on various properties. Such properties include job title, user functions or responsibilities. Much of the RBAC model is based on the mathematics of set theory; thus many of the constructs of the RBAC model are based on the notion of set membership. Individual users in an RBAC system are called *subjects*. Each subject has an *authorized role set*, which consists of all the roles that the subject has been authorized to use.

Although RBAC is very useful for modeling access control in a variety of applications, its roles are inherently subject-centric. Thus, it cannot be used to capture security-relevant context from the environment which could have an impact on access decisions. We have proposed a generalization of the basic RBAC model that allows policy designers to specify such environmental context through a new type of role that we call the *environment role* [4, 16]. In this paper, we focus on environment roles and explore how they can be used and implemented to enable context-aware applications.

We have chosen to use the notion of a role to capture environmental conditions that are relevant to access control because of several reasons. Although environment roles differ from traditional roles in some ways, the two types of roles do share many important properties. Thus, by generalizing the notion of roles to capture environmental state, we maintain uniformity and are able to use familiar properties such as role activation, role hierarchies and role separation to manage complex policies that depend on environmental state.

In [18], Sandhu distinguishes roles from groups by stating that roles possess permission. We show that this can apply to environment roles as well. However, in general, our model assigns permissions to sets of roles, where a set may include both subject and environment roles.

Environment roles share many characteristics with subject roles. For example, user Alice must provide some authentication information for the system to ascertain her identity which is then used to allow her to assume one or more roles. For environment roles, role activation is based on conditions in the environment where a request is made. These could include time, location or other contextual information that is relevant to access control. The state of the environmental conditions must be captured via sensors that are embedded in the environment. For example, currently the Aware Home makes use of active badges to track user locations. Clearly, the context information must be collected securely, in a manner similar to credential collection in user or subject role authentication.

RBAC also addresses many other issues such as role activation, revocation, role hierarchies and separation of duty constraints. These issues apply to environment roles as well and are discussed below.

3.1 Environment Role Activation

Environment roles generalize traditional RBAC roles by allowing the concept of a role to be applied to system states. A request in RBAC comes from a certain user or subject S who has a set of roles associated with her. This association is achieved via a two stage procedure. First, the security administrator in the system must define what roles S is allowed to take on based on the responsibilities and functions of S. Second, the user must provide evidence to prove her identity. Once this occurs, the set of roles that were defined by the security administrator are transferred to S and can subsequently be used during access requests. This is called *role activation* in RBAC.

A similar role activation problem exists for environment roles. First, the system administrator must define environment roles. For each role, she must define the associated environmental variables and conditions that must hold on the values of the variables. Unlike a user whose functions in an organization, and hence her roles, are well understood and relatively static, it may not be clear what environmental roles are active in the processing of an access request. In fact, there may be a very large number of environment roles defined in the system; at access time, the system must determine which of those roles are "active." For example, suppose an access request is made at 3:30 p.m. on Monday, January 1, 2001, under a CPU load of 74% and a network load of 31%. To mediate the access request, the system must gather information about which environment roles are currently active. There may be an environment role called *"high CPU load (over 70%)"*, as well as roles for *"Monday afternoons"*, *"weekdays"* and *"business hours*." All of these roles are active at the time of the request; however, it is likely that not all of them are relevant to the access control decision that must be made. Testing every environment role on every access control mediation would be prohibitively expensive, so the system should employ an efficient means of role entry testing for environment roles. We explore such methods further in section 5.

3.2 Environment Role Revocation

Roles in RBAC can be revoked (e.g., no longer be assumed by the subject) either when the subject's duties no longer require the privileges associated with the role's functions or when the role may conflict with some other roles that are to be activated (see "Separation of Duty" in section 3.4).

Role revocation for environment roles differs fundamentally because the conditions that lead to their activation can change dynamically and rapidly. Clearly, time and location of a mobile user are two conditions that change constantly. Other context of a resident in the Aware Home (e.g., if they are busy) could also change unpredictably. Thus, an environment role can be activated based on some system conditions at the time of a user's request and the request may be granted. However, at the time of the next request from the user, the system conditions may change and the environment role may no longer be active. In other words, an environment role is not necessarily active for an entire session.

There are several options to consider when revoking environment roles. In one extreme case, such roles may be activated only when a request needs to be authorized and can be implicitly revoked after the request is processed. However, the overhead of environment role activation must be incurred on each access. A better solution may be to associate a lifetime with an activated environment role. The role will have to be reactivated after the lifetime expires. For example, if a "business hours" role is activated at 1 p.m., it can be given a lifetime of 4 hours (assuming business hours end at 5 p.m.). In other cases, it may not be easy to associate a lifetime with an activated role. For example, a certain user may be expected to be at home during certain times. If she leaves the home unexpectedly, the system must detect the change in location condition and the role should be revoked, as with subject roles in RBAC, even if its lifetime is still not expired. We explore several of these options in the implementation section.

3.3 Role Hierarchies

One useful construct provided by RBAC is the *role hierarchy*. Role hierarchies can help manage role complexity through structure to exploit commonality not only among subjects but among roles as well. For example, in an organization all managers may have a certain set of core "management privileges" even though they all work in different departments. This commonality can be exploited through a role hierarchy that makes each department manager role a sub-role of a generic "managers" role. Role hierarchies allow a policy implementor to write generic access rules just once, rather than for every role to which the rules apply. Hierarchies also can serve as a tool for cleaner policy design, thereby eliminating some cases in which role precedence conflicts might otherwise have occurred. As a result, our model for environment roles incorporates support for role hierarchies.

To illustrate the power that hierarchical environment roles add to an access control mechanism, we begin by creating a subject role hierarchy[1], such as the one displayed in figure 1. This role hierarchy presents a graphical view of the sample household that we will consider in the following scenario. Specifically, it shows the relationships that exist between the various users and the roles that are present in the system. The figure shows that users *Mom* and *Dad* can assume the *Parent* role. The children in the home, *Alice* and *Bobby*, are allowed to assume the *Child* role. Also, all household members can assume the *Family* role.

In addition, we define a simple environment role hierarchy in figure 2. This role hierarchy presents a view of some basic environment roles that could be found in a home environment. In figure 2, we are concerned with time-related environment roles. Other environment roles could also define such a hierarchy. For example, a location environment could be inside or outside of a home. Inside the home, one could have upstairs or downstairs locations.

Assume that *Mom* and *Dad* have granted the children permission to use the smart intercom service [14] – a context-aware intercom

[1]In RBAC, by convention, senior roles appear at the top of the tree and junior roles are at the bottom. If we chose that representation, the tree shown in the figure would be inverted. Instead, we provide the shown representation to enhance clarity in this example.

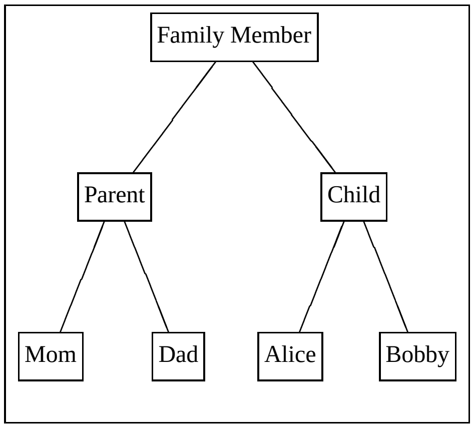

Figure 1: An Example Subject Role Hierarchy for the Home

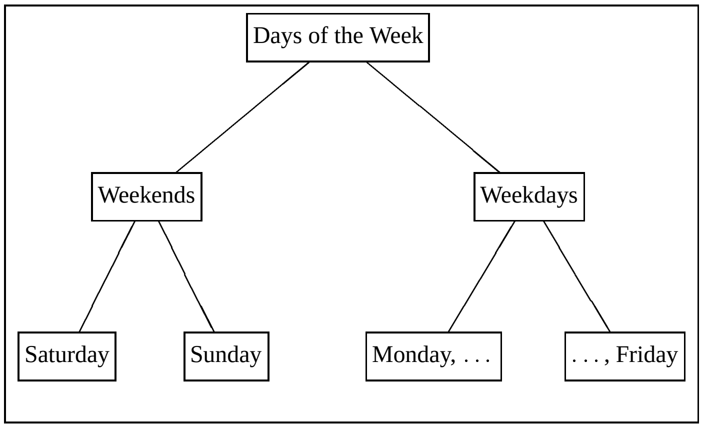

Figure 2: An Example Environment Role Hierarchy

application that supports audio connections between residents – on weekdays, but only during their free time after dinner, before going to bed. To enforce this policy, the system must be configured to identify the various entities in the system and classify them into the particular roles that are relevant to the access decision being processed. In addition, the security services must be able to identify the various environmental states (e.g, weekdays, “dinner time” and “bed time”) that are important components of this access rule.

In this particular example, we use hierarchies for both the traditional subject roles of RBAC, as well as for environment roles. First, the users must be classified so that a specific user identity can be mapped to a role, such as *Parent* or *Child*. By mapping a set of users to roles, the home administrator can specify an access control policy for a *group* of users, rather than for each individual user. For example, once a user is identified as the father in the family, the “Family Member,” “Parent” and “Dad” subject roles can all be activated for the user. In addition to subject roles, the system in this example uses an environment role named *weekdays*. *Weekdays* are defined by the system as the time from 12:01 a.m. on Monday to 11:59 p.m. on Friday. Also, since dinner usually is over by 7:00 p.m., and since the children have a bed time of 10:00 p.m., the environment role *free time* is defined to be from 7:00 p.m. to 10:00 p.m.

After defining all the necessary roles, the administrator needs to establish just one rule to specify the policy. The rule in this case is “any *child* can use the intercom on *weekdays* during *free time*.” This example illustrates how environment roles can significantly enhance an access control system, making it easy to take a fairly complex access policy and state it cleanly and efficiently. In addition, an access control policy using environment roles offers significant flexibility over one that simply sets access control based on resource and subject identity.

3.4 Separation of Duty

An attractive feature offered by RBAC is its ability to prevent a user from assuming conflicting roles. This mechanism can be used to enforce separation of duty requirements that ensure that a single user cannot acquire too much authority. For example, a single individual may be able to assume both “instructor” and “student” roles at different times but not both simultaneously. Similar separation of duty requirements are also useful for environment roles. For example, if non-employees are not allowed access to a building outside of working hours, the “inside-building” and “non-working-hours” roles should not both be active at the same time.

There is an important distinction between user roles and environment roles when it comes to separation of duty. Although we may want to prevent the activation of two environment roles at the same time, role activation is driven by the system state. This should be compared to user roles where the system determines what roles are activated for the user. If the system state is such that it implies that two conflicting environment roles are both active at the same time, the system is not in a safe state. In this case, separation of duty constraints help the system determine when it may be in a potentially unsafe state so it can attempt to resolve the conflict.

3.5 Example Environment Roles

An environment role can be based on any system state that the system can accurately collect. For example, we can define a role corresponding to each day of the week, or each month of the year. A policy rule such as "managers may edit salary data for their employees only on the first Monday of each month" is easy to implement using environment roles. Similarly, environment roles may be used to describe rules that relate access permissions to the locations of subjects. In the home, we can define location roles such as "upstairs," "downstairs," "master bedroom," etc. We can then use these roles in policy rules; for example: "children may only use the intercom while they are in the kitchen." In many homes and organizations, access control is dependent on the pattern of user activity within the organization. In other words, users must be granted access privileges only for the time periods in which they are expected to request access to a resource. As an example of such periodic access control, consider a part-time babysitter who should be authorized to have access to resources within the home only each *working day* during the hours that she is scheduled to work, between 3 p.m. and 6:30 p.m.

In addition to environment roles that incorporate time and location, we can define roles that capture any relevant environmental state. For instance, the Smart Intercom application in the Aware Home is capable of using sensed ambient noise levels and activity monitors to determine whether an individual is capable of accepting a call. Such environmental conditions can make access control rules extremely flexible and capable of adjusting to demands made by future applications.

4. ENVIRONMENT ROLE-BASED ACCESS CONTROL MODEL

Based on the formalization of the RBAC model in [18], we present a precise description of an access model that includes environment roles. As discussed in the previous section, both role hierarchies and separation of duty are meaningful in the context of environment roles, though they are omitted here in our description. Thus, we only consider flat user and environment roles. This formalization can be extended to hierarchies and constraints similar to the RBAC_1 and RBAC_2 models of [18].

Our model has the following components:

- From RBAC_0, we keep U, R, P and S. These capture users, roles, permissions and sessions respectively.
- We add ER and EC, where ER refers to *Environment Roles* and EC captures the *Environment Conditions* that are used to define such roles. To some degree, EC is analogous to U because the credentials associated with a user allow her to assume roles in R. Similarly, values of variables in EC allow certain roles in ER to be activated.

We have three relations UA, PA and EA that define the associations between subject roles, users, permission assignments and environment roles. These relations are as follows:

- $UA \subseteq U \times R$
  This comes from RBAC and defines what roles in R a user from U is allowed to assume.
- $PA \subseteq P \times R \times 2^{\mathrm{ER}}$
  This captures permissions that are assigned to a user role when a given set of environment roles is active. Thus, PA not only associates a permission with a subject role but makes it conditional on a set of active environment roles. Clearly, permissions may change for a single subject role accessing a resource if the environmental conditions vary between requests.

The following functions define what user or environment roles can be activated:

- User: $S \longrightarrow 2^{\mathrm{R}}$
  In a given session S, a set of roles can be activated for a user.
- Request: $EC \longrightarrow 2^{\mathrm{ER}}$
  Although some environment roles can be activated for the duration of a session, changing conditions will require other roles to be evaluated every time. Thus, based on the environmental conditions, a set of environment roles are activated at the time of a request.

In our system, a request that requires permission *p* can be granted if (1) <*p, r, e-set*> $\in PA$, (2) the subject role *r* is in the active role set of the user making the request, and (3) the environment roles active in the current environmental conditions *EC* contain the roles in *e-set*.

5. IMPLEMENTING ENVIRONMENT ROLES

This section presents our architectural design and the current implementation of a security infrastructure to support environment roles in the Aware Home. As noted earlier, the system administrator is responsible for defining environment roles. For each role, the administrator must define a set of environmental variables that are to be monitored and the conditions, *EC*, that must hold for activation. While the specification of environmental conditions is trivial, the secure and accurate capture of variable values from the environment is not an easy task. For example, in traditional RBAC, a user must provide some authentication information to prove her identity which is then used to allow her to assume one or more roles. Similarly, sensors capturing security-relevant environment context must provide some authentication information; also, the integrity of the information provided to the security subsystem must be guaranteed.

Below, we discuss issues concerning the collection of contextual information from the environment and the relation of this context to environment roles.

5.1 Collecting Environment State

To facilitate the collection of environment variables and their associated values, we make use of the Context Toolkit [5, 6]. The Context Toolkit is a software infrastructure that provides useful abstractions for collecting and organizing environmental state information; it allows for the seamless incorporation of sensed context into "aware" applications. The overall organization of the software is shown in figure 3.

Context widgets represent abstractions over sensors that hide details of how sensing and interpretation of the environment occurs. As an example, the intercom application presented in [14] provides two types of widgets – location and speech recognition. These widgets are essentially wrappers around an underlying local positioning system and speech recognition software; they provide interfaces that automatically deliver information to interested components or services in the system.

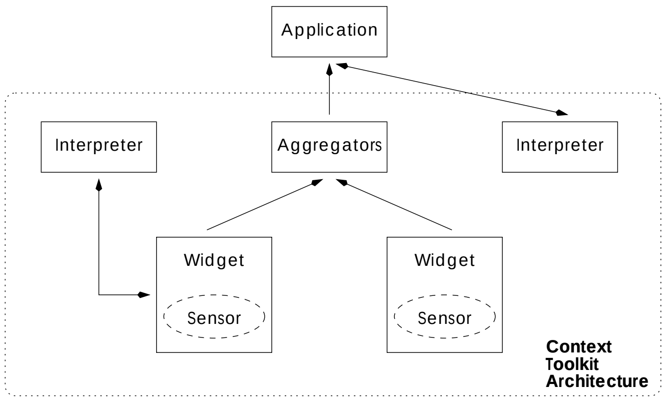

Figure 3: The Context Toolkit

Aggregators collect information for relevant entities of an application. In the home, there could be aggregators for rooms in the house (Room Aggregators) and residents of the household (Person Aggregators). For example, the Living Room Aggregator may know that both Dad and Bobby are in the living room. A Room Aggregator can also maintain additional information about the room, such as appliance status, ambient noise level, or even an interpretation or prediction of a high-level activity (e.g., party preparation or medical emergency). Person Aggregators currently hold information about a person's whereabouts in the house and may also store information about a current activity.

Interpreters are responsible for abstracting low-level context to higher-level information. This has traditionally been performed by applications, however, it has been separated to allow reuse of interpreters by multiple applications. An interpreter can convert state information to another format or meaning. For example, an interpreter can convert a room location into a building location (e.g., Room 123 maps to Building A). A more complex example is an interpreter that takes location, identity and sound information and determines that a meeting is underway. Context interpreters can be as simple or as complex as the designers want.

In the Context Toolkit, every software component described above can be shared simultaneously by multiple context-aware applications. Application components subscribe to aggregators and are notified when interesting events take place. In the smart intercom example, the application itself is responsible for managing subscriptions and responding properly to events in the system.

5.2 Secure Acquisition of Environment State

The Context Toolkit, as described above, was designed to support applications in a trusted research environment. As this toolkit is deployed and used to support security-relevant services, it is critical that mechanisms be provided to secure both the internal exchange of information (e.g., between the components listed above) and the external communication that takes place with applications.

We have built a "trusted" version of the Context Toolkit that will allow us to collect environment information in a manner that is both secure and reliable. The first stage in securing the toolkit involved a minor redesign of the internal components and the underlying communication mechanisms. A fundamental concern in building a secure networked system is authentication of both local and remote entities. Once obtained, authentication information provides the foundation for controlling access and enforcing policy in the network. Our redesign involved the distribution of authentication information (e.g., keys and certificates) to all components in the toolkit. Such tokens are necessary to guarantee the authenticity and privacy of information exchanged. We also have enabled all toolkit components to perform data encryption and to use this feature to protect the confidentiality of contextual information that is generated within the home.

The Context Toolkit is, by design, a distributed system. It is reasonable for us to assume that the individual components of the toolkit are secure as stand-alone services. For example, we assume that sensors and widgets are securely bound together in such a way that information from a sensor (e.g., an RF transmitter) can be securely transmitted to its associated widget. Also, we assume that all software components are "secure" as individual entities on the network; in other words, they are designed to properly implement interfaces. Moreover, malicious parties should not be able to subvert any access control mechanisms that may restrict access to the component.

Therefore, our goal was to provide an authentication framework for the Context Toolkit that would enable the various "principals" to accurately identify one another and communicate with confidence. The principals in this model include sensors, widgets, aggregators and interpreters. Authentication schemes ultimately enable each principal to obtain or possess some information identifying the other [3]. In the Context Toolkit, this information is difficult to bootstrap (e.g., aggregators may be dynamically generated to handle a particular component of the environment). In order to facilitate the free exchange of information within the toolkit, pairwise shared secrets are not an option for us. Instead, we decided to use public key encryption and established a certificate authority to aid with the distribution of keys between components.

While we begin with an assumption that the individual components of the toolkit are secure, we make no such assumption regarding the network over which they communicate. Interactions between secure components in the toolkit pass through a "chain" of intermediaries. First, the sensors must be trusted to accurately transmit data to the widgets. Second, the widget must be trusted to either properly execute a series of commands (e.g., a filter) or to securely forward information to an Aggregator. In some scenarios, interaction with an interpreter may be required. The chain of components and services that is constructed during a distributed transaction must be secured so communication channels between the involved systems can be trusted. This trust provides assurance that the commands and responses are safe from alteration, forgery and disclosure [8].

Rivest and Lampson [17] have devised a method of authentication support for distributed systems that do not contain a global hierarchy. Their egalitarian model allows for each principal to make (signed) statements and requests on the same basis as any other principal. Essentially, this allows each principal in the system to act as a certificate authority (CA). The policies and procedures adhered to by a principal are self-determined, making the model extremely flexible and non-limiting. Although their proposal allowed for some principals to act as "special roots", they did not require or rely upon the presence of a global name space. This work has since been incorporated into the Simple Public Key Infrastructure (SPKI), a technology that we make use of in our implementation. We view the egalitarian design of SPKI as an ideal environment for key management in the Aware Home – it allows home owners to localize their control of certificates and does not require the involvement of a "trusted third-party" to obtain signed certificates for their networks.

5.3 Environment Role Activation Service

The activation of environment roles involves several different issues that previously have been mentioned. First, environment roles need to be defined based on environmental conditions that are relevant to access control. Second, the appropriate set of roles must be active when a request is processed. This is done by an environment role activation service that we discuss in this section. We also discuss an authorization service which determines if a request should be granted or denied based on the active roles and the access rules that are defined to control access to resources.

5.3.1 Defining Environment Roles

In traditional RBAC, subject roles and their associated hierarchies are maintained by a security administrator who is familiar with the security requirements and objectives of the organization. As with subject roles, environment roles require similar administrative care in order to ensure that appropriate sets of environment roles exist for policy definition and enforcement.

We have chosen to define our environment roles using a prolog-style logical language for expressing policies. Our policy definition language consists of *statements*, each terminated by a period. As with the Generalized Policy Definition Language (GPDL) presented in [16], statements are used to define roles, sub-role relationships, transactions, and policy rules. The syntax for role definition is described below:

```
erole(role_name).
```

In the above role definition, `role_name` denotes the name of the role. The following examples show how to use this definition. The meanings of the roles should be obvious from the role names.

```
erole(weekends).
erole(business_hours).
```

5.3.2 Role Relationship Definitions

Environment roles are useless without a precise description of how they are activated and of the conditions that must be met. We accomplish this via the role relationship definitions. These definitions could be entry conditions that have to be satisfied for a particular role to be active, or it could be some logical combination of conditions that have to be satisfied in order to enter that role.

```
role_rel(erole_name, entry_condition).
role_rel(parent_role, child_role).
```

In the above definition, `erole_name` denotes an environment role, and the `entry_condition` represents a boolean statement about the conditions that the current environmental state must satisfy before it can enter that role. The specific syntax of these entry conditions depends on the type of environmental state that is being tested. For example, the policy administrator could use a statement such as `08:00 < time_of_day < 17:00` and use it as an entry condition for the `business_hours` environment role. The second statement is used to define higher-order relationships in a role hierarchy. It specifies a parent-child relationship where `child_role` is a child of `parent_role`.

Below, we present examples of the syntactical forms described above.

```
role_rel(business_hours,
08:00 < time_of_day < 17:00).
role_rel(sunday, day_of_week=SUNDAY).
```

The first statement says that the business hours are from 8:00 a.m. to 5:00 p.m. The second statement says that the environment role `sunday` may be entered when the system variable `day_of_week` is equal to "SUNDAY". The policy administrator would have to define both the `day_of_week` variable and the constants corresponding to each day. The activation of these roles is done by the environment role activation service by collecting information about the environment states from Aggregators and Interpreters in the Context Toolkit.

In order to keep track of errors due to conflicting definitions in the rule base, it is necessary to have error rules.

```
error(erole1, erole2).
```

This above rule states that given two environmental rules `erole1` and `erole2`, the system cannot simultaneously activate both of the rules. For example, it would be erroneous to have the roles weekend and weekday active at the same time. This is a mechanism to preserve the integrity of the rule base.

5.3.3 Activation of Environment Roles

The activation of environment roles depends on the environmental conditions. These conditions change dynamically and hence the active role set also changes. In our system, we plan to implement an environment role activation service that keeps track of all the active roles at a given time. This can be done easily with the facilities that are available. For example, such a service can read the current sensor state from widgets and/or aggregators. It can also place callbacks at the aggregators which will result in a notification if any of the values provided to the activation service change. Thus, the service can have a consistent view of the environmental conditions. Based on these, it can maintain the list of currently active roles, updating the list when any condition change notifications are received. When a request is made, the requester can request the set of needed active roles before the request is sent to the authorization service. The authorization service can also "pull" the needed active roles when it handles a request. Our design of the environment role activation service permits these different approaches for dealing with the activation of environment roles.

5.4 Authorization Service

Our model for providing security in the Aware Home separates out the function of access control and makes this a distributed core service, which performs authorization on behalf of the resources in the system. A client or subject desiring service from a resource must first contact an authorization server to obtain the credentials necessary to access the resource. Ubiquitous computing environments such as the Aware Home consist of many devices and services which will be centrally administered. The authorization service ensures that access rules are consistent across all resources and allows for all resources – regardless of processing capabilities – to enforce security policies. This section will provide a detailed description of how a centralized policy is defined using environment roles.

Both subject roles and environment roles provide powerful tools for capturing and organizing security-relevant information about various users and system states. In traditional RBAC, *transactions* are used to mediate access control. A transaction specifies a particular action to be performed in the system. Specifically, a transaction is a tuple in the form:

<*srole, object, erole-set, op*>,

where *srole* specifies a subject role, *object* specifies the object or resource for which access has been requested, *erole-set* specifies environment roles that must be active for the request to be granted, and *op* specifies the operation (e.g., read, write, execute, etc.) to be performed in the transaction. Semantically, the tuple represents an operation in which a subject acting in subject role *srole* performs operation *op* on a resource *object* under environmental conditions specified by the environment roles in *erole-set*. A policy database would consist of a transaction listing, paired with a *permission bit* for each transaction. The permission bit indicates whether the associated transaction is allowed or prohibited. Each <*transaction, permission bit*> is called a *policy rule*.

In cases where one or more components of the policy rule are not required (e.g., a rule that applies to all subjects), we maintain constructs that apply to all roles in the particular "class". Consider an example that forbids access to resource abc during working hours. The policy rule corresponding to this requirement would be represented as follows:

< *all-subjects*, abc, *working-hours, all-ops* >

By defining general roles for subjects and environment conditions, a policy administrator can create broad policy statements that remain in effect for a variety of active roles. In the example provided above, one policy rule replaces a handful of rules that were applied for each individual subject and environment role.

6. SECURE APPLICATION SCENARIO

Environment roles are a powerful and elegant concept for specifying access control rules in a computationally rich environment. This section shows how environment roles can be applied in practice to the home environment. It also illustrates some of the additional security benefits that such roles can provide in a system.

6.1 Securing the Smart Intercom

To illustrate the power and elegance of environment roles, we begin by creating a simple environment role hierarchy such as the one provided in figure 2. This role hierarchy presents a graphical view of some basic time-related environment roles. Specifically, it shows the relationships that exist between the various roles that are defined in the system. The figure shows that environment roles *Monday, Tuesday, . . . , Friday* inherit traits (e.g., permissions) from the role *Weekdays*, which in turn inherits traits from the role *Days of the Week*.

Using our policy definition language, we can generate the following statements to define roles and sub-role relationships:

```
erole(monday).
⋮
erole(friday).

erole(weekdays).
erole(weekends).
erole(days_of_the_week).

role_rel(monday, day_of_week=MONDAY).
role_rel(weekdays, monday).
⋮
role_rel(weekdays, friday).
```

In addition to defining a set of roles and their relationships, we have described the conditions that must be met in order for a role to be activated. In this example, the environment role `monday` may be entered when the system variable `day_of_week` is equal to "MONDAY".

Assume we want to create an access policy that states: "children may only use the intercom during weekdays, while they are in the kitchen." This rule is defined as specified in section 5.4. Specifically, our rule would appear in the form:

< *child, intercom, (weekdays, in kitchen),* activate:page, allow >

To illustrate the access request from beginning to end, we refer to figure 4.

Suppose that *Alice*, classified as a *child* by the home policy administrator, wants to use the intercom service on a Wednesday afternoon. Whether done implicitly via sensors or explicitly, Alice presents credentials to the system and is provided with a set of active subject roles. Ultimately, these subject roles will help determine the resources she is allowed to access. These rules are fundamentally consistent with those found in traditional RBAC.

With her set of active roles, Alice is now able to request access to a particular resource in the home. Knowing the policy limitations in advance, Alice proceeds to the kitchen and turns on the intercom service. Her request is forwarded to the centralized authorization service where the current home security policy is defined. As indicated above, a policy exists to grant access to Alice under certain conditions. In order to verify those (environmental) conditions, the authorization service contacts the environment role activation service. The environment role activation service, which interacts securely with the context toolkit, has already received notification from the Kitchen Aggregator that Alice is in the room. It also knows that it is currently Wednesday (also a weekday). This set of active roles is returned to the authorization service. The environment's active role set, along with the subject role and resource request, provides a match to the rule specified in the security policy. Access rights are therefore granted to Alice and her intercom session is established.

6.2 Enhancing a System with Environment Roles

The scenario above presents some sample environment roles and illustrates how role relationships can be used to establish security policies for the home. As previously stressed, ease of security policy definition and implementation is a key requirement for applications in this domain, since the typical homeowner cannot be expected to understand information security. In addition, we have stated that the system and related security mechanisms must be non-intrusive and easy to use. In this section, we briefly explore how environment roles can be used to enhance a system and also fulfill these requirements.

As already mentioned, a system should make access decisions without placing any undue burden on the users. One example of this would be a system in which an access request is triggered by a change in an environment role or condition. That is, the access request could be generated without explicit input from the user.

Consider a specific application example from the Aware Home. One research group is exploring how the Aware Home concept can

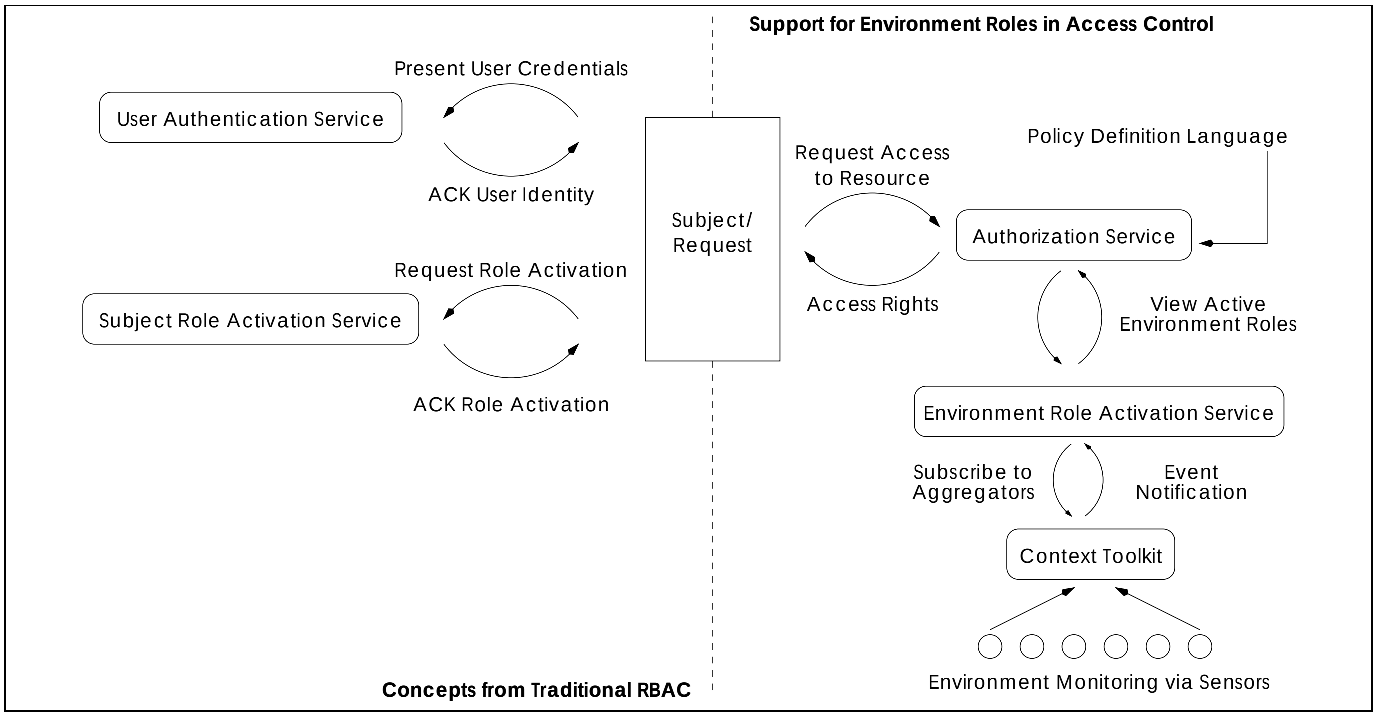
Support for Environment Roles in Access Control
Present User Credentials
User Authentication Service
ACK User Identity
Request Role Activation
Subject Role Activation Service
ACK Role Activation
Subject/ Request
Request Access to Resource
Access Rights
Policy Definition Language
Authorization Service
View Active Environment Roles
Environment Role Activation Service
Subscribe to Aggregators
Event Notification
Context Toolkit
Environment Monitoring via Sensors
Concepts from Traditional RBAC

Figure 4: Transactions with Environment Roles

help elderly residents remain in their homes as opposed to moving into assisted living communities. This application uses the home's sensors to enable important interactions with relatives outside of the home and with care specialists, effectively providing the same level of care and supervision that today can be found only in nursing homes and hospitals. It is important to note that access control policies are especially important in this example, as it involves a home that must be "opened" to allow regular access by many outside entities. For example, doctors, nurses, family members, lawyers and care groups may each require separate access rights in the home, depending on their primary function or responsibilities.

Assume we want to create a policy that states: "dial emergency contact if resident falls and injures himself." This rule could easily be defined as follows:

$<$ none, *dial emergency, (resident, injured),* call, allow $>$

In this example, the Context Toolkit would create a Person Aggregator that maintains an activity trait for the resident. Should the resident be injured or incapacitated, the toolkit would recognize the change in activity status via a callback. The *injured* role would then be activated for the resident and propagated to the appropriate services. The request to dial the emergency contact would be immediately granted. This is done without the presence of a specific subject role in the transaction.

7. DISCUSSION

We have introduced environment roles and describe why we believe they can be useful for securing "aware" applications in a ubiquitous computing environment. In addition, we have provided both a model and implementation details for an access control mechanism that makes use of environment roles in policy definition. In the following, we discuss some noteworthy aspects of environment roles that did not receive sufficient review in previous sections.

7.1 Environment Roles and Sessions

In [18], a *session* is created in order to enforce the principle of least privilege – a user should be allowed to login to the system with only those roles appropriate for a given occasion. Unlike subject roles, however, environment roles are dynamic in nature and it may not be possible to assign a set of active environment roles to a session. With traditional RBAC, a session allows constraints to be established and enforced that limit user-controlled role activation.

It is important to realize that many environment roles may be active at the time of a request; however, it is likely that not all of them are relevant to the access control decision being processed. Testing every environment role on every access control mediation would be very expensive, so the system should employ an efficient means of role entry testing for environment roles.

There are several strategies that may be employed to properly manage environment role testing. In our framework, we use the environment role activation service to automatically activate and deactivate roles when appropriate. By maintaining an internal data structure of all active roles, the environment role activation service can interact efficiently with the authorization service to aid in making prompt access control decisions.

Regardless of how environment role activation is implemented, sessions are simply inappropriate for this type of role. Additional research is necessary to determine more efficient strategies for environment role activation in context-aware environments.

7.2 Manipulation of Environment State

One subtle, though potentially dangerous security issue that may arise with the use of environment state in access control decisions is the possibility that a user can affect access rights through his activities in the system. For example, if a user knows that the system prohibits users from playing MPEG video files when the CPU load is high, he may intentionally run several CPU-intensive processes to mount a selective denial-of-service attack against other users who are accessing MPEG files. This type of vulnerability is very hard to eliminate; perhaps the best way to minimize exposure to it is to keep the access control policy secret. Of course, any user who is willing to experiment with the system can gather significant information about the current access control policy, such as what objects he can access, under what conditions he can and cannot access them, etc. It is unclear whether such vulnerabilities present a significant risk to the system; however, these vulnerabilities are common to all current "real-life" computer systems. Certain environmental data, such as CPU load, are inherently user-related; in contrast, environmental data such as the current time are not affected by any (legitimate) user activity. To solve the problem described above, we could simply ensure that all "important" access control rules are *not* dependent on any environmental data that malicious users can affect.

7.3 Policy Definition

The policy definition language described in section 5.4 is sufficient for defining policies, but in practice could be frustrating and clumsy for a policy administrator to manage, especially when editing large, complex policy files. Roles are inherently visual, so it would be useful to have a graphical policy editor that displays available roles, their relationships, and policy rules in an easy-to-understand manner. We have built a prototype graphical editor and are currently exploring how it can help to define and understand complex security policies. Such an interface is necessary as our access model is deployed in the Aware Home.

8. RELATED WORK

In this section, we briefly highlight several existing access models that have influenced our work with environment roles and context-aware access control models. We discuss traditional RBAC, time-based authorization, system-load-based authorization, and several other notable models.

We have discussed traditional RBAC [7, 19] and acknowledge the tremendous influence it has had on our research efforts. Our work expands the RBAC model by providing a more versatile and more expressive framework that incorporates the use of environment roles. By using the uniform notion of a role to capture both user and environmental attributes, our model allows for the definition of context-aware security policies. In addition, roles make it easy to define and understand complex security policies; adding environment roles to the model was necessary to support the advanced access control requirements that we are faced with in pervasive computing environments such as the Aware Home.

Environment roles are really one component in a Generalized Role-Based Access Control Model (GRBAC) [4, 16]. GRBAC is highly expressive, and easy-to-use access control model and was designed with two major goals in mind: flexibility and simplicity. GRBAC is flexible because it gives a policy administrator the freedom to structure an access control policy around subjects, objects, environmental conditions, or even a combination of all three. In addition, GRBAC is a very simple model; it achieves its goal of flexibility in policy design by using only one general grouping primitive: the *role*. In GRBAC, access policies are defined with subject roles, environment roles and object roles.

Bertino et al. [1, 2] have investigated support for temporal authorizations in database systems. They have examined both periodic and non-periodic authorizations. Their access control model is discretionary, whereas RBAC is mandatory. But in principle, their notion of temporal authorization is similar to our notion of time-based environment roles. We believe our model is better in terms of its usability and flexibility. Through environment roles, access policies can be simplified by defining temporal access rules that are assigned human-understandable names to various periods of time (e.g., "Monday," "Weekends," or even "Weekday mornings in July"). In contrast, their authorization language is very technical, which inherently limits its usefulness to the small set of people who have the background necessary to understand it.

The conditions on environmental state variables can be captured by constraints; this method has been explored in past RBAC work. Giuri and Iglio [9] have proposed a role-based access control model that provides special mechanisms for the definition of content-based access control policies. By extending the notion of *permission*, they have allowed for the specification of security policies in which the permissions to an object may depend on the content of the object itself. This approach, if applied to environment roles, would allow for the creation of *role templates* specifying constraints that must be bound to a role. In other words, subject roles would be parameterized based on various environmental conditions. Our model, however, allows for security policies to be specified using both subject roles and environment roles. We argue that our approach is more flexible as it separates the conditions that define constraints from the subject roles. These conditions are captured by environment roles which can be activated when necessary.

Similarly, in their Generalized Access Control Language (GACL), Woo and Lam [21] use the notion of *system load* as a determining factor in certain access control scenarios, so that, for example, certain programs only can be executed when there is enough system capacity available to handle them adequately. Given appropriate support for monitoring and reporting changes in system state, our model can also support such state-based authorization decisions using environment roles. In fact, the scope of environment roles is limited only by the level of support that the system provides for accurately reporting environmental information. As illustrated in section 5, we are implementing a toolkit that will allow for the accurate and secure capture of contextual information from the environment.

Access control languages that allow role-based security policies to be defined have been proposed recently. For example, a role definition language (RDL) is defined in [10] which allows roles to be activated (via role certificates) by services; credentials are supplied by the user making the access request and the RDL program manages access to individual service resources. A novel aspect of this scheme is that change in security-relevant state results in the revocation of such role certificates. A trust policy language (TPL) is presented in [11] which allows role activation to be based on certificates that are available to a requester. The FAM/CAM language presented in [13] provides support for both negative and positive access rights. It seeks to separate access policy from access mechanism by providing the policy designer with a language that is provably capable of expressing any access policy.

There are several other access control models that are worth noting due to their influence on our work with environment roles; we briefly mention them here. The first related model was proposed by Jajodia et al. [13]. It seeks to separate access policy from access mechanism by providing the policy designer with a language that is provably capable of expressing any access policy. We also note the work of Shen and Dewan [20]. They have developed a flexible, powerful role-based model for access control in collaborative environments, where policies must account for concurrent operations on shared objects and other complex access issues.

9. CONCLUSION

In this paper we have extended traditional role-based access control to include the notion of an environment role. We are focused on solving the problem of securing context-aware applications in a ubiquitous computing environment. Our work shows how the well developed concept of a role can be used to capture security-relevant context of the environment in which access requests are made. The resulting access control framework is highly versatile, yet the underlying constructs (roles) remain consistent with traditional RBAC. We have presented our extended access control model as well as an initial implementation. This work is currently being used to build an authorization framework to secure applications in the home and community, such as those being explored by the Aware Home Initiative at Georgia Tech.

10. REFERENCES

[1] Elisa Bertino, Claudio Bettini, Elena Ferrari, and Pierangela Samarati. Supporting periodic authorizations and temporal reasoning in database access control. In *22nd VLDB Conference*, 1996.

[2] Elisa Bertino, Claudio Bettini, Elena Ferrari, and Pierangela Samarati. A temporal access control mechanism for database systems. In *IEEE Transactions on Knowledge and Data Engineering*, volume 8, 1996.

[3] Andrew D. Birrell, Butler W. Lampson, Roger M. Needham, and Michael D. Schroeder. A global authentication service without global trust. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 223–230, 1986.

[4] Michael J. Covington, Matthew J. Moyer, and Mustaque Ahamad. Generalized role-based access control for securing future applications. In *Proceedings of the National Information Systems Security Conference (NISSC)*, October 2000. Also appeared as technical report GIT-CC-00-02, available from the Georgia Tech College of Computing.

[5] Anind K. Dey and Gregory D. Abowd. The context toolkit: Aiding the development of context-aware applications. In *Workshop on Software Engineering for Wearable and Pervasive Computing*, June 2000.

[6] Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A context-based infrastructure for smart environments. In *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), Dublin, Ireland*, pages 114–128, December 1999.

[7] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A role based access control model and reference implementation within a corporate intranet. In *ACM Transactions on Information Systems Security*, volume 1, February 1999.

[8] Morrie Gasser, Andy Goldstein, Charlie Kaufman, and Butler Lampson. The digital distributed system security architecture. In *Proceedings of the National Computer Security Conference*, pages 305–319. NIST/NCSC, October 1989.

[9] Luigi Giuri and Pietro Iglio. Role templates for content-based access control. In *Proceedings of the Second ACM Workshop on Role Based Access Control*, pages 153–159, Fairfax, Virginia, USA, November 1997.

[10] R. J. Hayton, J. M. Bacon, and K. Moody. Access control in an open distributed environment. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 3–14, 1998.

[11] Amir Herzberg, Yosi Mass, and Joris Mihaeli. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 2–14, 2000.

[12] Georgia Tech Broadband Institute. The Aware Home Research Initiative. Research Initiative Web Page, 2000. http://www.cc.gatech.edu/fce/ahri/.

[13] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Elisa Bertino. A unified framework for enforcing multiple access control policies. In *Proc. of the 1997 ACM International SIGMOD Conference on Management of Data*, May 1997.

[14] Cory D. Kidd, Thomas O'Connell, Kris Nagel, Sameer Patil, and Gregory D. Abowd. Building a better intercom: Context-mediated communication within the home. Technical Report GIT-GVU-00-27. GVU Center, Georgia Institute of Technology. June 2000.

[15] D. Moore, I. Essa, and M. Hayes. Exploiting human actions and object context for recognition tasks. In *IEEE International Conference on Computer Vision*, 1999.

[16] Matthew J. Moyer and Mustaque Ahamad. Generalized role based access control. Technical Report GIT-CC-00-16, College of Computing, Georgia Institute of Technology, September 2000.

[17] Ronald L. Rivest and Butler Lampson. SDSI - a simple distributed security infrastructure. SDSI Version 1.1, October 1996.

[18] Ravi S. Sandhu. Role based access control. In *Advances in Computers*, volume 46. Academic Press, 1998.

[19] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role based access control models. In *IEEE Computer*, volume 2, February 1996.

[20] Honghai Shen and Prasun Dewan. Access control for collaborative environments. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 51–58, November 1992.

[21] Thomas Y. C. Woo and Simon S. Lam. Designing a distributed authorization service. In *Proceedings of IEEE INFOCOM*, March 1998.