A Flexible Security Architecture for Pervasive Computing Environments

A Thesis
Presented to
The Academic Faculty
by
Michael J. Covington

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
April 2004

Copyright © 2004 by Michael J. Covington
A Flexible Security Architecture for Pervasive Computing Environments

Approved by:

Dr. Mustaque Ahamad, Advisor

Dr. Hariharan Venkateswaran

Dr. Irfan Essa

Dr. Wenke Lee

Dr. Blaine W. Burnham

Date Approved: 4/12/2004
To my family:

Thank you for all of your love, support and encouragement.
ACKNOWLEDGEMENTS

If there is one thing that I have learned from 24 years of schooling, it is this – I would not be here today were it not for the support and encouragement that I have received from a lot of people.

First, I would like to thank my advisor, Mustaque Ahamad. When we started working together, Mustaque took a risk and agreed to change his research focus so that I could work in an area that was of interest to me. His willingness and ability to change have provided me with opportunities that far exceeded my expectations for graduate work. Mustaque has provided support, friendship and guidance, and I am thankful for everything that he has taught me.

I would also like to thank the rest of my thesis committee for their support. Hariharan Venkateswaran, Irfan Essa, Wenke Lee and Blaine Burnham have provided me with invaluable advice and feedback on both my academic pursuits and my future research directions. I also cannot forget to thank Ken Calvert, who served as my interim academic advisor during my first year in the Ph.D. program at Georgia Tech. Ken introduced me to academic research, sparked my interest in the Aware Home, and provided growth opportunities that were unavailable to many first-year students. His enthusiasm for research and his encouragement kept me focused and helped to make my transition to graduate school an enjoyable learning experience.

I have also been very fortunate to have a great group of friends at Georgia Tech. This includes my officemates, the regulars from the Systems group Pizza e Birra outings, and the many other students and faculty, too numerous to name, with whom I’ve interacted over the years. These people have helped to keep me grounded; we’ve worked together and had fun together, and the Georgia Tech experience would not have been complete without them.
I would also like to express my thanks to the many people who have encouraged and fostered my love of computing over the years: David Steinour, Lisa Yeo, Fred Portier, Jim Harle, Suzanne and David Blackburn, Will Mumford, and Gerald Ursitti. You have been my mentors, my friends, my colleagues, and a source of never-ending moral support. Your encouragement and guidance led me to pursue a career in computer security research, and for that I am grateful.

This work would not have been possible without the support of my close friends outside of the Georgia Tech community: Amy, Vijay, Tomko, Reece, Tucker ... and all of my close friends who have provided moral support and routine escapes from the world of computing. Whether it was a walk to the park, a weekend rock climbing adventure, or a trip to the beach; the times we have spent together have helped me to remember that there’s more to life than work. Thanks for keeping me balanced and aware of the world around me.

Finally, I would like to dedicate this dissertation to my family: Gloria, David, Jennifer, Melissa, Martha, George, Nancy, Mike, and Cole. Without your support, encouragement, guidance, and love, I never would have made it through the whole Ph.D. process. You have given me the support to pursue my dreams, the courage to stand up for my beliefs, and the confidence that I need to succeed. Thank you.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................ iv  
LIST OF TABLES ............................................. x  
LIST OF FIGURES ........................................... xi  
SUMMARY ...................................................... xii  

## CHAPTER 1  INTRODUCTION AND MOTIVATION .................. 1

1.1 Pervasive Computing ........................................ 2  
1.1.1 Defining Context and Context-Awareness ............... 2  
1.1.2 The Aware Home: An Application Scenario ........... 4  
1.2 Security Challenges in Pervasive Computing Environments .......... 5  
1.2.1 Context-Aware Authorization ................................ 6  
1.2.2 Non-Intrusive User Authentication ..................... 7  
1.2.3 Secure Context Collection ............................... 8  
1.3 Dissertation Overview ..................................... 8  
1.3.1 Thesis .................................................. 9  
1.3.2 Thesis Contributions ................................... 9  
1.3.3 Thesis Outline ........................................ 10  

## CHAPTER 2  BACKGROUND AND RELATED WORK ............. 12

2.1 Access Control Models ..................................... 12  
2.2 Authentication ........................................... 14  
2.2.1 Distributed Authentication .............................. 15  
2.2.2 User Recognition and Modeling ....................... 15  
2.3 Sensor Matching and Fusion ................................ 17  
2.4 Trust and Reputation Management .......................... 19  
2.5 Security Architectures .................................... 20  

## CHAPTER 3  GENERALIZED ROLE-BASED ACCESS CONTROL .... 23

3.1 Overview ................................................ 23
LIST OF TABLES

Table 1 Generating Access Requests in CASA over HTTP (in milliseconds) . . . 110
Table 2 Generating Access Requests in CASA over HTTPS (in milliseconds) . . 111
LIST OF FIGURES

Figure 1  Basic RBAC Definitions and Rules ........................................... 25
Figure 2  An Example Subject Role Hierarchy for the Home .......................... 37
Figure 3  An Example Environment Role Hierarchy .................................... 38
Figure 4  The Context Toolkit .................................................................. 43
Figure 5  Transactions with Environment Roles ....................................... 51
Figure 6  Using the Generalized Policy Definition Language to Define Roles, Sub-
roles, and Role Relationships ................................................................. 52
Figure 7  Overview of Authentication Model ............................................ 61
Figure 8  10 Positive Experiences, Varying Trust ...................................... 79
Figure 9  10 Negative Experiences, Varying Trust ..................................... 80
Figure 10  10 Alternating Positive/Negative Experiences ............................ 81
Figure 11  Consensus of Opinions ............................................................. 82
Figure 12  10 Random Sensors forming Consensus ...................................... 84
Figure 13  Logical Framework for Securing Context-Aware Applications ....... 92
Figure 14  Example GRBAC Policy Specification in XML .......................... 93
Figure 15  Environment Role Definition .................................................... 96
Figure 16  Example Environment Role Definition: Party ............................ 98
Figure 17  Original AudioIM Application with Login Prompt ..................... 105
Figure 18  Screenshot of Graphical Policy Manager ................................... 107
Figure 19  Example XML encoding of Role Definitions and Policy Statements . . 109
Figure 20  Enhanced AudioIM With Parameterized Authentication Support ...... 112
Figure 21  Using AudioIM without Authentication ..................................... 113
Figure 22  Using AudioIM with Enhanced Authentication and Access Control . 113
SUMMARY

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 resources and services. Clearly, the 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 authentication procedures.

We address the problem of securing applications that will access and control information resources in context-aware environments. Specifically, we have designed security services that are adaptable, in the sense that they make use of contextual information to provide flexible system access and policy enforcement. Our contributions include a flexible access control model that makes significant use of context in policy definition, a technique that provides enhanced authentication services and an architecture for secure context-aware computing.
CHAPTER 1

INTRODUCTION AND MOTIVATION

As computers become more pervasive and their functionality is more transparently integrated into homes and communities, new applications will emerge to make everyday living easier for people. Such applications, which will be enabled by a pervasive computing and communication infrastructure, will provide unobtrusive access to important information, resources and services. Clearly, the 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 authentication procedures.

Security requirements have traditionally been defined in terms of static policies and procedures. For instance, the majority of access control decisions are based on the evaluation of rules that do not consider the “context” of an access request or account for changing conditions in the environment. In addition, secure transactions require explicit interaction with the subject of an access request; the process of authentication requires the user to provide proof of his identity. Unfortunately, this rigid approach to security often leads to problems with the protection of resources and the granting of legitimate access rights. By ignoring context and environmental conditions, policies must be expressed in a language that is strict and inflexible. Likewise, authentication can be inconvenient, demanding and intrusive.

As computing technology becomes more tightly integrated into the fabric of everyday life, it is imperative that security mechanisms become more flexible and less intrusive. To address these concerns, our research is focused on providing security services for pervasive computing environments that can adapt to changing conditions as requests are made.
In the remainder of this chapter, we discuss some of the properties that define pervasive computing environments and context-aware applications. We motivate our research by exploring the security challenges in this application space and present our thesis. The chapter is concluded with a summary of the contributions of this dissertation and an outline of the following chapters.

1.1 Pervasive Computing

The concept of pervasive or ubiquitous computing was first introduced by Mark Weiser [101] in the early 1990s. He detailed a vision in which computers were seamlessly integrated into the world, and through which information flowed freely. In this vision, technologies disappeared into the background, allowing people to use them unconsciously to accomplish everyday tasks.

Recent advances in processing, communications, sensors and other related technologies have brought the vision of pervasive computing closer to reality. But the one component of this vision that is perhaps the most difficult to achieve involves altering the interface between humans and computers. In order for technologies to “disappear,” the interfaces they provide must be more natural. In addition, the systems must be capable of accomplishing tasks without being provided with explicit instructions that detail what must be done and how to do it. Once computers are capable of collecting, interpreting and acting on context, we will be that much closer to achieving the vision of pervasive computing that was defined over a decade ago.

1.1.1 Defining Context and Context-Awareness

When humans communicate, a number of factors contribute to the success of an exchange and the efficient flow of information, including a rich, shared language, and a common understanding of everyday situations. Furthermore, the implicit use of information that is apparent from a given situation, also known as context, can often enrich communication.
between humans and increase conversational bandwidth.

Unfortunately, this ability to communicate and interact does not naturally transfer to human-computer exchanges. Computers do not share a common language or set of understandings with humans. As a result, information must be explicitly provided to computers, along with a significant amount of detail that would otherwise be implicitly communicated to humans. Because computers are not currently enabled to utilize context in the same way as humans do, computers are limited in the tasks they can perform and the input they can process. By improving the computer's access to contextual information, it is possible to increase the richness of the communication in human-computer interactions.

There is an entire body of research dedicated to improving interactions between humans and computers. One particular branch of study is focused on addressing the deficiency that computers have in processing input. Some approaches aim to improve the "language" that is used between humans and computers, while others are attempting to increase the amount of situational information that is made available to computers.

This situational information, or context, would allow the computer to better understand information that is provided explicitly as input. Context-aware computing strives to enable developers to provide context to computers, so they can make applications aware and responsive to the full context available in human-computer and human-environment interactions. By doing so, context-awareness makes it easier for users to interact with computers and the environment, by allowing them to focus less on the computer and more on the information being exchanged.

As computers become more pervasive, users will begin to expect transparent access to information and resources, whenever they want and wherever they are located. With computers being deployed in a wide-variety of settings, and contextual-awareness being incorporated into applications, interesting new problems arise and the need for enhanced security services becomes clear: users, many of whom may not be authenticated, are trying to obtain access to different information from the same services in different situations.
Context can be used to help determine what information or services are available, who has privileges to access them, and whether access requests should be generated based on a series of events or situational changes.

We have explored the need for context, both in improving human-computer interactions and in settings where access scenarios change rapidly. In the next section, we will provide a detailed look at an application scenario that makes significant use of context.

1.1.2 The Aware Home: An Application Scenario

In our research, we target an “aware” or smart home setting, such as the Aware Home [38], in which a pervasive computing environment is used to provide augmented services to the residents and their guests in the home. The Aware Home research initiative at Georgia Tech brings together an interdisciplinary team of researchers to build an information-rich “home of the future” from the ground up. This interactive home has a rich computation and communication infrastructure, which includes a variety of sensors and cameras that will enable the home to become “aware” of its residents and their activities [63]. Various applications are being explored to exploit this “awareness” to make daily living easier for the home’s residents. The applications in the Aware Home will span the domains of education, entertainment, physical security, inventory management (e.g., for the contents of the pantry or refrigerator), and utility management (e.g., for gas and electricity), as well as applications that permit rich interactions with people and institutions outside the home.

Consider some specific examples of Aware Home applications. One research group is exploring how the Aware Home concept can help elderly residents remain in their homes longer, rather than having to move into assisted living communities. This application [66] 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. Another class of applications will allow residents to manage inventories in the home from any location, inside or outside
the home. For example, the Cyberfridge [59] application collects information about food items in a refrigerator and makes the data accessible from anywhere. Cyberfridge can interface with a local food delivery service to automatically reorder food items such as milk or eggs when necessary. A third example is an application that automatically manages home resources such as hot water and heat, based on the residents’ preferences and daily living habits. It can, for example, choose to heat the house only when it knows there are residents inside, and it can choose to produce hot water only at times when residents usually take showers or wash clothing. Such an application can even negotiate the best possible electricity rates with utility services, based on the needs and preferences of the home’s residents.

All of the applications described above share a common thread: they are possible as a result of the Aware Home’s ability to gather, store and transmit useful information about the state of its resources and occupants. Clearly, this information should be available only to legitimate users and applications. Financial loss, public embarrassment and even physical harm are just a few of the many potential negative consequences of a breach in the security of any of these applications. Therefore, from a security standpoint, the Aware Home project presents a unique opportunity to explore the support that is necessary to secure the home of the future, before the technology—and the risk that accompanies it—becomes widely available. The following section discusses the challenges that arise when attempting to secure pervasive computing environments, such as the Aware Home.

1.2 Security Challenges in Pervasive Computing Environments

The pervasive computing environments and context-aware applications alluded to in this chapter present new and interesting security challenges. Although considerable work has been done in securing military and commercial information systems, few projects have specifically addressed the needs of a residential computing infrastructure. Current research
that specifically targets home environments attempts to move traditional security mechanisms into the residential space. In contrast, we are developing security techniques for the Aware Home that are natural, intuitive and non-intrusive for connected homes and community environments. Our approach is also relevant in other pervasive and distributed computing environments where authorization policies are rich with context and authentication services are distributed and non-intrusive.

The transparent nature of ubiquitous computing motivates the need for a security architecture that can operate effectively in these dynamic computing environments. For instance, pervasive computing strives to minimize the amount of explicit interaction between a user and the system by making information more readily available to users via more natural interfaces. In order to maintain these properties, it will be necessary for the system to transparently determine the source of a request. In addition, the contextual information collected and managed by the system will provide insight and knowledge that can be used to specify more fine-grained security policies. Thus, we must look at new access control models and authentication techniques that can accommodate the demands of these information-rich computing environments.

The main challenges in providing security for pervasive computing environments are discussed in the following sections.

1.2.1 Context-Aware Authorization

Security policies in an information-rich environment like the Aware Home can potentially be quite complex. A policy can restrict access to information or resources based on several factors, including attributes pertaining to the subject, the resource or the environment. For example, subjects can be classified into roles such as "resident" or "guest." Access rights can depend on the subject’s classification (e.g., "resident"), as well as on his or her actual 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.
While time and location are natural examples of environmental states 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 [68] 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 or if the request is an emergency. Unlike traditional access control models where requests are made explicitly by subjects, requests in the Aware Home may be triggered by changing environmental conditions. For instance, if an emergency situation is detected, a request for professional assistance can be generated and should be granted based on the context of the situation. Likewise, the context of a resource itself may determine whether access can be granted. The attributes of a resource, such as its access history, its current state, or its location, may determine whether access can be granted or denied.

The secure collection of this information, in addition to the specification of policy using context, is a challenge that we will discuss in detail later in this dissertation.

1.2.2 Non-Intrusive User Authentication

Another challenge presented by a pervasive computing environment like the Aware Home involves relieving the user from the “burden” of authentication. Ideally, information available from sensors in the home should be used to automatically infer a subject’s security-relevant attributes (e.g., identity, role, location, etc.). Although it is possible for a resident to use a physical authentication token, it is undesirable to expect the user to carry such an object around at all times. Previous work with physical identification tokens such as the Active Badge system [99] have yielded useful results but are less practical for home use and unreliable for authentication (users can assume a different identity by simply carrying another person’s badge).

Several sensor-based technologies such as voice and face recognition can be deployed in the Aware Home to non-intrusively identify humans and track their movements. Many
such techniques can establish the identity of a subject with only a partial level of certainty. Such "partial authentication" has important implications for access control models. In particular, some identification mechanisms are known to be more reliable than others.

1.2.3 Secure Context Collection

Traditionally, interactive applications are limited to using input that is provided by users explicitly interacting with the system. As users move away from the traditional desktop computing model and move toward ubiquitous computing environments like the Aware Home, there is a greater need for applications to leverage implicit information, or context. These pervasive computing environments are rich in context, with users and devices moving around and interacting, computational services becoming available or disappearing over time, and a seamless flow of information that is available “wherever, whenever.” This contextual information is usually not available to applications but can be useful in adapting the way in which they perform services and in changing the services they make available. In addition to making this information available to context-enabled applications, we envision it being used to enable context-aware security services.

Clearly, our ability to secure applications using context is limited by our ability to securely collect context. If we use contextual information that is provided by a malicious entity, the security of our system is compromised. In addition to managing role complexity, any context-aware security architecture must also provide support for secure context collection.

1.3 Dissertation Overview

In this section, we clearly define our thesis and highlight the contributions presented in this dissertation.
1.3.1 Thesis

Our thesis is that it is possible to provide enhanced security services, appropriate for pervasive computing environments and context-aware applications, that permit flexibility in both policy definition and user authentication, and for which there exists an architecture that can adequately meet the demands of these services in a timely fashion.

1.3.2 Thesis Contributions

The contributions of this dissertation that provide evidence to support our thesis are in the following four areas.

**Generalized Role-based Access Control Model**  An access control model provides the mechanisms that are necessary to implement security policies. We have identified an intuitive and flexible access control model that allow a variety of security policies to be implemented. In particular, we present a generalized role-based access control model that allows a variety of policies to be specified using contextual information. In addition to subject roles that have been explored in previous models, we have used the notion of a role to capture security relevant state of the system or environment, and attributes of resources for which access must be controlled. We describe our generalized role-based access control model, the various components, and the language that is used to intuitively specify policies in the model.

**Parameterized Authentication**  In an environment like the home where the computing infrastructure should be largely invisible, it is desirable that security be provided to users transparently. We have explored techniques that use sensor-generated information with varying accuracy and trust to authenticate users. In particular, we have explored an approach to sensor-based authentication that can adapt to accommodate incomplete, unreliable, or inaccurate input provided to the system. *Parameterized Authentication* moves
beyond the traditional approach to security by acknowledging that identity verification cannot always produce perfect results. Our model addresses such inherent imperfections by introducing a metric, the Authentication Parameter, that captures the overall “quality” of authentication.

**Context-Aware Security Architecture** The authorization and authentication mechanisms will provide a foundation for building a core of security services that will be used to secure applications. Through our context-aware security services, our system architecture offers enhanced authentication services, more flexible access control and a security subsystem that can adapt itself based on current conditions in the environment.

**System Implementation and Evaluation** In addition to the design of our security architecture mentioned above, we discuss an implementation and show how it can be used to secure several sample applications.

1.3.3 Thesis Outline

The remainder of this dissertation is organized as follows:

**Chapter 2** reviews the related research for this work. This includes an in-depth discussion of existing access control models. It also demonstrates why existing security mechanisms are not sufficient for pervasive computing environments and context-aware applications.

**Chapter 3** introduces the Generalized Role-based Access Control Model (GRBAC) that provides enhanced support for using context in the definition of security policy. It explains the concept of a role and details how it can be used to describe subjects, objects and environmental context – the three primary components in a secure context-aware transaction.

**Chapter 4** describes an approach to sensor-based authentication that can adapt to accommodate incomplete, unreliable, or inaccurate input provided to the system. Using a
concept called the Authentication Parameter, we are able to enforce and enhance the prin-
ciple of least privilege by ensuring that the authentication process provides credentials that
are sufficient but are not stronger than the access level required by the requested opera-
tion. This approach is particularly well-suited to meet the demands of a context-aware and
pervasive computing environment in which authentication may be performed using passive
and non-intrusive techniques.

Chapter 5 describes a security architecture that can be used to secure context-aware
applications.

Chapter 6 builds upon our description of the security architecture in chapter 5 through
an implementation of the various security services and an illustration of how they can be
used to secure example applications.

Finally, chapter 7 contains a summary and conclusion, with suggestions for future
work.
CHAPTER 2

BACKGROUND AND RELATED WORK

The design of new, flexible security services for pervasive computing environments and context-aware applications will naturally build upon the large collection of work that has preceded it. In this chapter, we introduce relevant background material necessary for understanding the research contributions described in the remainder of this dissertation. This chapter will present related work that includes access control modeling, authentication, sensor fusion, and trust management. In addition, we discuss general system architectures for providing security services, and some competing approaches for providing security in intelligent environments.

2.1 Access Control Models

Secure computer systems enforce security policies via access control models that are supported by the system. Such models must define who the users or subjects are, and what resources or objects should be protected. Access control rules define when requests from subjects should be granted or denied [54]. Many access control models have been developed. For example, in the Unix discretionary access control model, individual users or groups of users can be granted read, write or execute access to files in the system. A mandatory access control model [8] makes use of subject and object labels to govern read and write access to objects. Role based access control (RBAC) is a form of mandatory access control (e.g., centrally administrated) in which access to resources is granted based on a subject’s position or job title in an organization [32, 83].

Security policies can be defined at a high level in a natural language, or they can be refined to a form that is suitable for implementation in a system. They may also be formally
characterized using mathematical models [1].

Another related access control model, proposed by Jajodia et al. [46], has explored flexible access control models. Their approach seeks to separate access policy from access mechanism by providing the policy designer with a language that is provably capable of expressing rich access policies. Among other features, the Flexible Authorization Manager (FAM) allows both positive and negative access rights and hierarchical grouping of subjects, providing a rich language for defining and structuring policy.

Bertino et al. [9, 10] have investigated support for temporal authorizations in database systems. They have examined both periodic and non-periodic authorizations. Similarly, in their Generalized Access Control Language (GACL), Woo and Lam [102] 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.

There are several other access control models that are worth noting due to their influence on the design of our model; we briefly mention them here. The first related model is multilevel access control [8], which traditionally has been used in military computer systems for highly sensitive data. Its basic premise is to allow information to flow up the chain of security levels, but never down.

The use of XML in security policy definition has been studied in numerous contexts, though work in this area has focused primarily on the development of access control models and methods to address encryption in the language. Some of the more relevant work in XML security includes [22] in which Damiani et al. present an access control model that uses an XML-based approach to define and enforce access restrictions directly on the structure and content of Web documents. In [11], Bertino et al. present an XML-compliant formalism for specifying security-related information for Web document protection. In addition, Herzberg et al. present a Trust Policy Language (TPL) [43] that defines policies
using a well-formed XML document. Finally, Netegrity [69] has presented S2ML, a Security Services Markup Language that provides a mechanism for describing existing security models using XML syntax.

Finally, access control for collaborative applications, where several users may access shared documents at the same time, has been explored by Shen and Dewan [24, 90]. They have developed a flexible role-based model for access control in collaborative environments, where policies must account for concurrent operations on shared objects and other complex access issues.

2.2 Authentication

Authentication of subjects is essential in secure systems to determine if a certain request should be granted. Authentication is generally based on the following: what the subject knows (e.g., passwords), what the subject has (e.g., possession of smartcard), or who the user is, which makes use of biometric information about the user. These, and several new methods being proposed for automatic authentication [44] do not completely meet the goal of making security invisible. In particular, they require some degree of participation from a subject. For example, even with biometric authentication, a finger print or retina scan must be done.

Another issue with authentication involves the need to improve the security in the design or implementation of some approaches. For example, it has been shown that passwords are a fairly weak mechanism for authenticating users. Studies have shown that users tend to select passwords that are easily broken. In work from Monrose, et al. [61], the authors propose a technique for improving the security of password-based authentication by incorporating biometric information into the password. Specifically, their approach generates a "hardened password" based on both the password characters and the user's typing patterns when typing the password.

Another similar strategy involves improving individual authentication by employing
multiple "factors" or approaches to the process. Multi-factor authentication consists of verifying and validating the authenticity of an identity using more than one validation mechanism. Examples include a PIN plus a hardware token (something you know and something you have) or a PIN and a biometric (something you know and something you are). This approach is reasonable, but it is not foolproof. It is based on the assumption that the failure modes for different factors are largely independent. In many cases, multi-factor authentication can improve the security of authentication. However, as with any security solution, a proper analysis of the threat model is required.

2.2.1 Distributed Authentication

In sensor-based authentication, identification information is collected from multiple sensors that may be trusted to different degrees. In distributed system environments that span multiple trust domains, authentication may rely on certificates that are issued by certification authorities (CAs) that also have different levels of trust associated with them. Several researchers have explored such models where an authentication measure based on the trust level of the CAs is derived. Beth et al. [12] present a model where trust can be combined from multiple CAs. Reiter and Stubbing [77] explore the design principles that must be followed by such a model. Maurer [60], Jassang [47] and others have explored additional techniques to compute an authentication metric based on paths or chains of CAs that are used for authentication. These techniques primarily focus on trust of the relevant CAs and do not address accuracy of identification information.

2.2.2 User Recognition and Modeling

Our approach to authentication will use information recorded by environmental sensors to implicitly infer user identity or characteristics. There is much work in tracking and recognizing people and activities from video [28]. A survey of computer vision methods for recognizing and identification of people by face recognition is presented in [18, 81, 52]. Some well-known methods for tracking movements of people include [23, 75, 14, 87, 104].
These methods measure pixel-level information to track people while some researchers rely on a near-complete reconstruction of human form from image sequences for recognition and understanding of what people are doing [7, 37, 48, 76, 79]. The importance of time in the analysis and recognition of hand and body movements has resulted in the use of hidden Markov models for recognition after training on views of the model [16, 15, 71, 92, 97, 105].

A significant aspect of our research involves the integration of scalable and distributed security services with sensing technologies for identification and authentication of users, both actively and passively. Earlier work in this area has concentrated heavily on developing algorithms, methodologies, and systems for identifying and tracking users and interpreting what is happening in a given scene [28]. This research effort is directly relevant to surveillance and monitoring applications.

We also consider the areas of signal processing and the understanding of video images for tracking, location and recognizing people [93]. In this approach, multiple cameras are used to track users in a space. Additional cameras are used to lock on the face for tracking and recognition of the individuals and recognition of their expressions [30, 31] and for detecting where a person is looking [29, 86]. Recently many face recognition systems have become commercial, however these system require high-resolution frontal images of faces [52]. Other active tracking system which uses a pan-tilt-zoom camera, provide very high-resolution images that yield good recognition accuracy using such commercial systems [96]. These enhanced systems have also improved on the tracking methods in complex scenes through changing lighting conditions to provide robust tracking [95].

Other research efforts involve the incorporation of commercially available speaker identification and speech recognition systems with person tracking system to accurately locate a speaker and identify the speaker and what they are saying [34]. This form of data/sensor fusion yields very reliable metric in case of uncertainty from a single source. In addition, these efforts also provide confidence and believability measures so as to allow for reliability
and robustness. These measures provide cues that certain sensing information is becoming unreliable, which in turn instantiates a different process with higher expectation to allow for more accuracy. Finally, methodologies have been presented that allow for modeling of occlusions by sensors to provide tracking and recognition even when the person is occluded by something in a scene [17].

In addition to tracking and recognition of people, some research efforts have also developed methodologies for recognition of what people are doing in a scene. This is useful in the context of implicit identity based on what is happening. This can then be connected to the above framework of tracking and provide stronger cues for recognition of a person in a very transparent fashion over an extended time frame. In [62, 64], a framework is presented that uses hidden Markov modeling (HMMs) with Bayesian inference mechanisms for recognizing action and also recognizing the objects a person interacts with (or is close to). This framework allows for determining the context in a scene and the relationship of a person, the objects and in the scene itself over time. This form of information is crucial in monitoring the users and determining what they are intending to accomplish in a certain situation.

2.3 Sensor Matching and Fusion

In interactive intelligent environments such as the home, biometric technologies are often used to obtain user identification with minimal explicit input. These devices are able to associate an individual with a previously recorded identity based on their physiology. Because many physiological or behavioral characteristics are distinctive to each person, some claim that biometric identifiers are inherently more reliable and more capable than knowledge-based and token-based techniques in differentiating between an authorized person and a fraudulent impostor [45].

Unfortunately, many of the biometric devices widely available today cannot guarantee very high quality identification. This is typically a result of noise that interferes with sensor
readings, limitations of the processing methods or the variability in both the biometric characteristic as well as its presentation [73]. Such weaknesses in biometric technology create opportunities for an impostor to "mimic" the actions of a legitimate user and, in essence, trick the system into believing that they are someone else.

The accuracy of biometric devices can be measured in two ways. For identification-based systems, a biometric signature of an unknown source is compared against a database of biometric signatures belonging to known individuals. The primary performance measure would be based on the system’s ability to properly identify a biometric signature owner. On the other hand, in verification-based systems, a user presents a biometric signature and a claim that a particular identity belongs to the biometric signature. Performance measures of this system type are traditionally characterized by the false-reject and false-alarm rates that result from the comparison of the signature and claim [74].

The algorithms that are used to determine the accuracy of biometric devices are also used to return a confidence measurement of the readings validity. For instance, the Smart Floor [72] has been demonstrated to identify users with a 93% accuracy rating. Despite the availability of such measurements, most security systems do not correlate a user’s access rights with the level or quality of their identification.

Sensor fusion refers to the combining of multiple identification “inputs” in order to produce a single identification metric. For example, one research group has recently incorporated speaker identification and speech recognition systems with a person tracking system to accurately locate a speaker and identify the speaker and what they are saying [35]. This form of sensor fusion yields a more reliable metric in case of uncertainty from a single source.

In addition to combined input, sensor fusion allows the system to reason about the fidelity of the composite information. This measure can be used to enhance the strength of the authentication service. By authenticating a user into a role that is based on the strength of her identification, we ensure that a user is never allowed to have more access than the
evidence provided by her for authentication.

Our model for Parameterized Authentication uses a similar approach to reason about discrepancies in multiple sensor input and forms consensus that reflects all inputs. Our approach tags all incoming data with two measurements. The first, an accuracy rating, provides a measure of how well the principal’s identity can be matched when compared to stored attributes. The second measurement captures the level of trust held in the sensor that provided the input. The authentication process takes this input from one or more external sensors. The number of sensors from which input is combined can be determined based on the level of authentication required by the system.

2.4 Trust and Reputation Management

In application areas such as e-commerce, knowledge sharing, and the Internet in general, people and services interact with one another, often times not personally knowing the party with whom they communicate. Because these parties are autonomous and often controlled by different administrative and legal domains, it is important that participants be able to identify trustworthy parties with whom they can interact and untrustworthy parties with whom they should avoid interaction.

Trustworthiness is often viewed as the expectation of cooperative behavior and can be based on previous experiences with the same party. However, it is often necessary to evaluate the trustworthiness of an entity without having any prior direct interactions. In such situations, a participant can place trust based on the latter’s “reputation” among others in the system. This approach bases reputation on the collection of evidence that supports certain claims of good or bad behavior. In parameterized authentication, reputation or sensor trustworthiness can be based on whether a sensor’s input led to correct authentication or a breach of security.

eBay and other similar Internet communities are practical examples of reputation management systems. On eBay’s site, for example, sellers receive feedback (+1, 0, −1) for their
reliability in an online auction. Reliability is computed using the feedback values that are collected over a period of several months.

Yu and Singh [106] developed an approach for social reputation management and examine the key properties of trust. Specifically, they look at how agents place trust in other agents and explore ways for principals to convey trustworthiness to one another.

In similar work, Kamvar et al. [49] describe a trust-based algorithm that identifies malicious entities in a peer-to-peer environment and isolates them from the network. Their reputation system, called EigenTrust, identifies inauthentic files on a network and even handles conditions where malicious peers cooperate in an attempt to deliberately compromise the system.

The mathematical foundations for reputation management are firmly rooted in probability and statistics. Our work draws heavily from Bayesian statistics in which a mechanism for combining evidence is presented. We have also reviewed the theories presented by Dempster and subsequently extended by Shafer [88] in which they present a calculus that can accommodate uncertainties in evidence collection.

2.5 Security Architectures

Despite considerable interest and research in pervasive computing [101, 80], security concerns in such environments have received little attention. In this section, we briefly highlight several existing projects and technologies that have influenced our work with providing security services in context-aware environments.

There are a number of well-known architectures that have been used to build secure systems in the past. The Kerberos system [51] implements a protocol that can be used to efficiently authenticate entities in a distributed system. Satyanarayan [84] discusses the architectural issues surrounding the security services that are incorporated into the Andrew distributed computing environment. In addition, significant work has been focused on the modeling, building and analysis of secure Public Key Infrastructures (PKIs). Other issues
related to building secure distributed systems, including subject role authentication and delegation are addressed by Lampson et al. [55].

In addition to these architectures, other security infrastructures have been presented to specifically address the needs of authorization logic. In [6], an architecture is proposed for securing distributed document management system. The authors address the need for an access control logic in a complex distributed system. In particular, they discuss a logic that supports linked local name spaces and the management of a large system spread across administrative domains.

Gopal and Manber [41] discuss how to integrate content-based access mechanisms into traditional file systems. Their work is based on Gifford’s Semantic File System [39]. Specifically, they focus on the systems issues involved in efficiently integrating hierarchical file systems with database-like query functionality.

Al-Muhtadi et al. [2] present an approach to adapt traditional security solutions—specifically the Kerberos extension SESAME—into “smart spaces” such as the home. Their work focuses on the integration issues that must be addressed when placing computing-intensive security mechanisms into devices with limited resources. While a GRBAC-like component could conceivably be added to their solution, our architecture is appropriate in the context-aware environments that we target. The architecture that we propose can support partial authentication, context-aware access control and dynamic policy generation based on roles — elements not found in traditional computing environments.

Our research will lead to a security architecture that will include authorization and authentication services for securing applications. This architecture will exploit currently existing technologies such as Java access control model [40] and the security framework defined for distributed object systems such as CORBA [70]. Several other results from previous research are also relevant to our proposed work. Since the application scenarios described by us may download mobile code at participant sites, security issues related to mobile code are also of interest [98, 40]. Finally, our research on services for securing
applications is complementary to network level security techniques such as firewalls and secure communication [50, 19].
CHAPTER 3

GENERALIZED ROLE-BASED ACCESS CONTROL

3.1 Overview

We are addressing the problem of securing applications that will access and control information resources in the home of the future. Specifically, we are designing a security system based on a paradigm called Generalized Role-Based Access Control (GRBAC). GRBAC is an extension of traditional Role-Based Access Control (RBAC). It enhances traditional RBAC by incorporating the notion of object roles and environment roles, with the traditional notion of subject roles. These new types of roles allow one to define rich, easy-to-understand security policies without having significant technical knowledge of the underlying computer systems that implement those policies. In this chapter, we motivate the need for GRBAC, provide a high-level description of it and demonstrate its usefulness and flexibility via several example applications.

3.2 Traditional RBAC

Traditional Role Based Access Control (RBAC) [32, 83] is a form of mandatory (i.e., centrally administered) access control. It is based on the premise that most real-world access control decisions are determined by a person’s position or job title within an organization. Accordingly, the purpose of RBAC is to encourage the design of security policies that closely mirror the structure of organizations. In this section, we highlight the most important features of the traditional RBAC model.
3.2.1 Basic Features and Rules

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. 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. We use the term role possession to denote that a role is in the authorized role set of a subject. In other words, we say that subject $S$ possesses role $R$ if $S$ has been authorized to use $R$.

The other two fundamental concepts in RBAC are the object and the transaction. An object is any resource in a system. Example resources in the home include appliances such as a dishwasher or stereo, media objects such as movies, and sensitive digital information such as medical records or income tax returns. A transaction is a series of one or more accesses to a set of one or more objects. A transaction in the home could be as simple as reading file foo on the family computer. In contrast, a transaction in a military setting can be as complex as aiming and firing a missile from a Navy destroyer. All policy rules in RBAC are linked to roles, rather than to individual subjects. Formally, each role $R$ is associated with an authorized transactions set; this is the set of transactions that a subject may perform using role $R$. Therefore, for a subject $S$ to gain access to transaction $T$, $S$ must demonstrate possession of a role $R$, for which $T$ is in the authorized transactions set of $R$. Figure 1 summarizes the basic RBAC features.

3.2.2 Some Problems and Solutions

At its core, RBAC is quite simple; however, in practice, RBAC policies can become very complex and unwieldy. In this section, we describe some of the problems that RBAC systems face, as well as several advanced RBAC features that have been used to solve the
problems. We first examine two problems that stem from the complexity of policies: separation of duty and role precedence. Then we discuss role activation and role hierarchies, two constructs that can help mitigate these problems.

Separation of Duty It is implicit from the previous section that a subject can possess multiple roles simultaneously. Typically, there are no problems associated with multiple role possession; however, there are some cases in which the set of access privileges granted by multiple role possession can constitute a conflict of interest. For example, in a financial institution, two possible roles are teller and account holder. An account holder authorizes certain actions (such as withdrawals and deposits) on his account, and a teller executes those actions. If a person is a bank employee and also owns a checking account at the bank, there exists the potential for that person to act as both an account holder and a teller at the same time. With the privileges of both account holder and teller, an employee may be able to perform illegal actions, such as making fraudulent deposits, on his account.

Such scenarios occur often in RBAC systems; the circumstance described above is known as a separation of duty problem. There are two varieties of separation of duty: static and dynamic. Dynamic separation of duty occurs when two roles present a conflict of interest if a subject uses them both at the same time. The conflict of interest described above
is an instance of dynamic separation of duty. Note that there is no conflict of interest if the
employee acts as a teller during one time interval and an account holder during another
interval, since only when he assumes both roles simultaneously is it possible for him to
abuse the system. In contrast, static separation of duty occurs when two roles present a
conflict of interest that cannot be resolved by simply preventing the roles from being used
simultaneously. In these cases, the two roles may never be used by the same subject. If
roles R1 and R2 exhibit static separation of duty, and subject S has acted in role R1, he may
never act in role R2.

Role Precedence  Another problem that relates to multiple role possession is role prece-
dence. Role precedence stems from inconsistent access rules between two roles that a
subject possesses. For example, in the home environment, suppose that user Bobby is
authorized to possess the roles of child and family member. Suppose also that the family
member role is authorized to read family medical records, but the child role is not. If Bobby
tries to read the family’s medical records, the system must decide how to resolve the inco-
sistency in the access policy. To solve the problem, the system must decide which access
rule takes precedence over the other. There are many ways to make this decision. The sim-
plest way would be to always give precedence to the role that denies access. Similarly, the
system could always give precedence to the role that allows access. Or there could be some
other predefined rule or algorithm established to decide role precedence. One approach to
solving this problem, as we discuss below, is the use of role activation. Role precedence is
a problem that every RBAC system must solve.

Role Activation  We discussed above the concept of an authorized role set: the set of roles
that a subject is allowed to use. The problems of separation of duty and role precedence
both are related to an authorized role set, because as the size of an authorized role set grows,
separation of duty and role precedence become more difficult to manage. One of the most
common and effective ways to handle this problem is to restrict a subject’s role usage to a
subset of his authorized role set at all times, so that only those roles that are necessary to perform his current duties are active. This is called role activation. When role activation is used, a subject must declare which roles he intends to use at all times. The roles that have been declared active constitute the subject’s active role set. Only roles in the active role set can be used to execute transactions. This mechanism allows the system to more easily enforce dynamic separation of duty constraints: the system simply disallows any two roles with dynamic separation of duty constraints from being active at the same time. Role activation also provides a natural mechanism for resolving role precedence: in case of a conflict between roles, active roles take precedence over inactive roles.

Role Hierarchies Another useful RBAC construct 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 once for every role to which the rules apply. This kind of structuring tool can help avoid policy “bugs”: cases in which the policy implementor has incorrectly written the policy. Hierarchies also can serve as a tool for cleaner policy design, thereby eliminating some cases in which role precedence conflicts might otherwise have occurred.

3.3 The GRBAC Model

Traditional RBAC is very useful, but it suffers from subject-centric limitations that restrict the policy designer to a subject-oriented viewpoint. Generalized Role Based Access Control (GRBAC) is an extension of RBAC that removes the subject-centric limitation, allowing a policy designer to write the policy from a subject-centric, object-centric, or
environment-centric viewpoint, or whatever combination of these is most appropriate for the circumstances. GRBAC removes the limitations of RBAC by using the basic concept of a role to organize all entities in a system. It exploits the organizational power of roles for grouping environment states and objects, in addition to subjects. This section introduces our GRBAC model.

3.3.1 Subject Roles

A subject role in GRBAC is analogous to a traditional RBAC role. Each subject is authorized to assume a set of subject roles. Subject roles may be hierarchical or “flat” (single-level) in nature. The system may also use subject role activation. The only difference between GRBAC subject roles and traditional RBAC roles is the way that they are used to make access decisions. In traditional RBAC, an access decision is based entirely on the permissions associated with the set of roles that the subject possesses. In GRBAC, an access decision depends not only on subject roles, but also on environment roles and object roles. We describe each of these roles below.

3.3.2 Environment Roles

There are many real-world instances in which access control depends not only on the person making the access and the object being accessed, but also on the state of the environment during the access. 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. The two most basic types of environmental information are time and location, but any security-relevant information in the environment that can be accurately captured by the system also can be used to control access to system resources.
The GRBAC model allows policy designers to specify system state through 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 videophone while they are in the kitchen.”

Relating to environment roles, there clearly are many tangential issues that must be addressed before environment roles can be used in real access control systems. First and foremost, the system must be able to securely and accurately collect enough system data (e.g., an accurate estimate of the current time, or the location of a subject in the home) to determine whether a given environment role is active. One effective approach to this problem would be to use a trusted event system that is capable of generating events based on various system state changes. Second, the system must provide a means for policy implementors to define roles. Some basic environment interface must exist, so that policy writers can associate their environment role definitions with actual system states. Both of these issues are the subject of ongoing research and are beyond the scope of this paper.

3.3.3 Object Roles

Subject roles and environment roles allow a policy implementor to structure a policy based on either the properties of the subjects in the system, or the system state itself. But what if the policy implementor wants to structure the policy according to the properties of the resources in the system? To accommodate this scenario, the GRBAC model also includes object roles. Object roles allow us to capture various commonalities among the objects in a system, and use these commonalities to classify the objects into roles. Object roles can be
based on any classifiable property of an object, including its date of creation, object type (image, source code, streaming video, etc.), sensitivity level (secret, top secret, etc.), or information about the contents of the object (for example, we could classify objects based on whether they contain any content related to Microsoft Corporation). After classifying the objects, we can make access control decisions based on the classification scheme that we created.

3.3.4 Making Access Decisions with GRBAC

In Figure 1, we outline the basic algorithm for mediating access to objects in the traditional RBAC model. In RBAC, if subject $S$ wants to access object $O$, $S$ must possess a role $R$ that is authorized to execute transaction $T$, such that $T$ can access $O$. In GRBAC, the access mediation algorithm is similar, but slightly more complex. Subject $S$ possesses a set of subject roles, and object $O$ possesses a set of object roles. In addition, the system keeps track of a set of environment roles. For $S$ to access $O$, $S$ must possess some subject role $R_S$, such that:

1. $\exists$ some object role $R_O$, possessed by $O$;
2. $\exists$ some environment role $R_E$ that is currently active;
3. $\exists$ some transaction $T$ that allows $R_S$ to access objects in role $R_O$ when $R_E$ is active.

Clearly, this access mediation rule is more complex than the corresponding rule for traditional RBAC.

We briefly discussed separation of duty and role precedence above, two of the potential problems that can arise in an RBAC system. These two problems are not confined to traditional RBAC; they also can cause difficulty in the GRBAC model. In fact, GRBAC’s generality makes it even more susceptible to various types of policy conflicts and ambiguities. Our purpose in this paper is not to outline all of these potential problems, but simply to introduce the reader to the fundamental GRBAC concepts of subject roles, object roles and
environment roles. We do not discuss the GRBAC model in any more detail here; however, we encourage interested readers to refer to [65] for a more thorough review of the model, its usage, and the problems that can arise from it.

3.4 Environmental Conditions in Access Control

Many of the applications in the Aware Home are context-aware [63] 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 [68], 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.

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 [25]. 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.
3.4.1 Context through the Environment

The Aware Home applications previously alluded to 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 is 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.

3.4.2 Defining Environment Roles

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 [21, 65]. 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 [82], 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.

3.4.2.1 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 a later section.

3.4.2.2 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.

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.

3.4.2.3 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, such as the one displayed in figure 2. 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.

![Family Member]

![Parent

![Child

![Mom

![Dad

![Alice

![Bobby

Figure 2: An Example Subject Role Hierarchy for the Home

In addition, we define a simple environment role hierarchy in figure 3. This role hierarchy presents a view of some basic environment roles that could be found in a home environment. In figure 3, 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

---

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.
downstairs locations.

Figure 3: An Example Environment Role Hierarchy

Assume that *Mom* and *Dad* have granted the children permission to use the smart intercom service [68] – a context-aware intercom 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 bedtime 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 clearly 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.

3.4.2.4 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.
3.4.2.5 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.

3.4.3 An Environment Role-Based Access Control Model

Based on the formalization of the RBAC model in [82], 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 [82].

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.

- \(PA \subseteq P \times R \times 2^{ER}\)
  
  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: \(S \to 2^R\)
  
  In a given session \(S\), a set of roles can be activated for a user.
• Request: $EC \rightarrow 2^{ER}$

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 $e-set$.

3.4.4 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.

3.4.4.1 Collecting Environment State

To facilitate the collection of environment variables and their associated values, we make use of the Context Toolkit [26, 27]. 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 4.

![Diagram](image)

**Figure 4: The Context Toolkit**

*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 [68] 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.

*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.

3.4.4.2 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 [13]. 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 [36].

Rivest and Lampson [78] 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.

3.4.4.3 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.

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 [65], statements are used to define roles, sub-role relationships, transactions, and policy rules. The syntax for role definition is described below:

\[ \text{erole}(\text{role\_name}). \]

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.

\[ \text{erole(weekends)}. \]
\[ \text{erole(business\_hours)}. \]

**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.

\[ \text{role\_rel(\text{erole\_name}, \text{entry\_condition})}. \]
\[ \text{role\_rel(\text{parent\_role}, \text{child\_role})}. \]

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(ero1, ero2).

This above rule states that given two environmental rules ero1 and ero2, 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.

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.

3.4.4.4 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:

<role, object, erole-set, op>.
where *role* specifies a *subject role*, *object* specifies the object or resource for which access has been requested, *role-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 *role* performs operation *op* on a resource *object* under environmental conditions specified by the environment roles in *role-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.

3.4.5 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.

3.4.5.1 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 3. This role hierarchy
Figure 5: Transactions with Environment Roles

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.

Figure 6 illustrates how our policy definition language can be used to define roles, sub-roles, and role relationships.

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 using the form:

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

To illustrate the access request from beginning to end, we refer to figure 5.

51
Figure 6: Using the Generalized Policy Definition Language to Define Roles, Sub-roles, and Role Relationships

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.
3.4.5.2 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 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.

### 3.4.6 Discussion of Environmental Roles

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.

#### 3.4.6.1 Environment Roles and Sessions

In [82], 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.

3.4.6.2 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.

3.4.6.3 Policy Definition

The policy definition language described earlier 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.

3.5 Summary

In this chapter we have introduced a new access control model, Generalized Role-Based Access Control (GRBAC), and described why we believe it will be useful for securing applications in the highly-connected homes of tomorrow, as well as in other environments. The major benefit of GRBAC over current access control models is its combination of usability and expressiveness. GRBAC is easy to use because it is based on one main concept: the role; however, the uniformity and flexibility with which roles are applied to subjects, objects and environment states also makes the model very powerful and expressive. It is important to note that GRBAC is not a complete security solution in itself. It is only an access control model; to be useful in the real world, it must be integrated carefully into a trusted computer system. In the near future, we intend to explore these integration issues and build a prototype system based on GRBAC.
CHAPTER 4

PARAMETERIZED AUTHENTICATION

We describe an approach to sensor-based authentication that can adapt to accommodate incomplete, unreliable, or inaccurate input provided to the system. Parameterized Authentication moves beyond the traditional approach to security by acknowledging that identity verification cannot always produce perfect results. Our model addresses such inherent imperfections by introducing a metric, the Authentication Parameter, that captures the overall "quality" of authentication. We define authentication "quality" in terms of sensor trustworthiness and the accuracy of sensor measurements. Using the Authentication Parameter, we are able to enforce and enhance the principle of least privilege by ensuring that the authentication process provides credentials that are sufficient but not stronger than the access level required by the requested operation. This approach is particularly well-suited to meet the demands of a context-aware and pervasive computing environment in which authentication may be performed using passive and non-intrusive techniques. Our model supports the transparent capture of authentication-relevant information from the environment and provides a foundation for generating dynamic credentials for sources of requests. We present our model, discuss its contributions, and illustrate how it can be used to support rich access control policies.

4.1 Overview

Authentication is a fundamental building block in any system that enforces a security policy; it enables "principals" to identify themselves to the system and provides a foundation for access control. For the purposes of this paper, the principals we consider are users,
though their attributes such as location, role, or history may also be relevant in authoriza-
tion decision-making. All authentication schemes follow the same basic approach: known
identification information about a principal is compared with information received from
the source claiming to be that principal. Authentication is successful if both pieces of in-
formation match; however, authentication failure will result if a match cannot be produced.

The conceptual simplicity of the authentication process is deceptive as it conceals many
important aspects of its design and implementation. For example, most systems make the
assumption that authentication information is free from error. In reality, however, data per-
taining to user authentication could be erroneous – it could come from a malfunctioning or
malicious source, a forged or stolen credential, or a faulty transmission. Clearly, the trust-
worthiness of a source providing authentication information and how closely the provided
information matches stored data are important issues that must be addressed.

Pervasive computing environments strive to provide transparent access to resources
and services. For example, the Aware Home [38], a prototype “home of the future” that
has been built at Georgia Tech, is exploring a variety of emerging applications that range
from remote appliance management (e.g., Cyberfridge [59]) to “awareness” and the active-
monitoring of each resident’s activities and needs. One project explores how a “smart”
home can help elderly residents by monitoring their medical condition [67]. The prototype
home has a rich computation and communication infrastructure and will eventually be con-
nected to other homes and institutions in the community. A variety of sensors are used to
infer the activities of the home’s residents, and various applications use this information to
help improve the quality of life for residents.

If access to such services is restricted or controlled, authentication information must be
captured in a manner that is consistent with the theme of pervasive computing; it must be
minimally intrusive and readily available. With access controls potentially included on ev-
ery embedded device, authentication should be non-intrusive to the user, perhaps performed
through embedded sensors, and efficient so as to avoid service delays and interference with
the transparent nature of the pervasive computing model. This approach reduces the need for explicit interaction between the system and the user. However, this approach could lead to the corruption of authentication data: malicious entities could tamper with sensors, sensors could provide erroneous data, or sensors may provide data that simply does not match information stored by the system. Such information may be insufficient for the system to establish an absolute identification for the source of a request. An authentication service in this type of environment must be capable of adapting to doubt or uncertainty with regard to user identification.

Clearly, the assumption that a principal’s identity can be verified with absolute certainty is impractical in real world scenarios, even when explicit interaction is required. Cases of identity theft, unauthorized access to secure facilities and circumvented authentication mechanisms are regularly reported in news headlines. A recent article in The Register [58] discusses biometric sensors that can be easily defeated, including a fingerprint sensor that can misinterpret imprints in a Gummi Bear candy as a valid fingertip scan and a vision system that can be defeated by a photograph of an authorized user.

Authentication services that rely on sensors to provide relevant user attributes must be designed to identify and/or process input that is less-than-perfect. In addition, the output produced by the authentication service should indicate the quality of the result – if the input does not produce a match that is 100% certain, the output should reflect this discrepancy, thus allowing access privileges to more accurately reflect the level of authentication achieved by the user. Non-binary authentication could be used to limit the damage that may result from an erroneous authentication; access to closely guarded resources would only be granted when stronger credentials are presented to verify the claimed identity.

We have designed a model that can be used to produce an authentication measure from incomplete, unreliable, or inaccurate identification information that is provided by a set of sensors. We accomplish this by providing a quality measure for authentication. We define two metrics that allow us to measure trust and accuracy values associated with input to an
authentication process. In addition, we provide a method for computing an authentication value by combining inputs from multiple sources; by reinforcing authentication and forming a consensus, our authentication framework is more robust and fault-tolerant than those that rely on a single source for authentication. Our model provides an adaptive approach to authentication because additional sensors can be polled if a stronger level of authentication is required based on the processing done at the authorization service. We refer to this approach as parameterized authentication.

The remainder of this chapter proceeds as follows: Section 4.2, we discuss the various logical components that comprise our model and introduce the Authentication Parameter. We also identify a set of design principles to serve as guidelines in the development of our model. Sect. 4.3 details our approach to managing trust and accuracy in the system and illustrates how these measures are used to produce the Authentication Parameter. We provide details on computing accuracy and discuss how trust evolves over time. We revisit our design principles in section 4.4 to discuss how well our model meets these principles. We discuss several outstanding issues and related research contributions in section 4.5. Finally, the paper is concluded in section 4.6.

4.2 System Model

Our model for parameterized authentication is based on information obtained from a distributed network of sensors. The model is composed of the following logical components: users, sensors, a user-attribute database, an attribute-matching service, a trust analysis engine, an audit service and the authentication service. A high-level overview of our adaptive authentication architecture is given in figure 7. In the following sections, we detail the functionality of these components and describe how they interact with one another.
4.2.1 Users

Our approach to user authentication assumes an open-world model in which there exist two classes of Users – those that are “known” (identifiable) by the system, and those that are not. A user is defined by a collection of traits, or properties, that are either non-intrusively captured by sensors or explicitly provided as input to the system. While some users may have similar traits, it is the collection of properties that define a user. By definition, no two collections are equal. Our model identifies four fundamental trait types: physiological, knowledge-based, object-based and historical.

Physiological traits physically identify a user. To perform an identity verification using physiological traits, a physical property is measured and compared with stored data that identifies the user. For example, various biometric techniques can be used, including fingerprint, retina scan, speech recognition, etc. Ideally, the measured property would be
distinguishing and unique among all known principals.

Knowledge-based traits represent information that is held specifically by the user. To perform an identity verification using knowledge-based traits, the user requesting authentication must demonstrate knowledge of secret information that is known or produced only by the claimed identity. Password-based authentication is an example of a knowledge-based system. Knowledge-based traits could also be produced through computed responses to challenges presented by the system.

Object-based traits are typically held in the user’s possession. Examples of object-based traits include an ID badge or key-card. To perform an identity verification using object-based traits, the user must produce an item that can only be held by a principal with the claimed identity. Ideally, the objects used would be unforgeable and closely guarded to prevent unauthorized usage.

Historical traits consist of collections of historical data regarding the user, her activity, possessions, etc. Historical data can often be used to determine user identity without using sensor resources to retrieve "fresh" data. However, historical data is also subject to misuse and misinterpretation. Our model presents an approach for the use of historical data that supports efficient resource management and maintains an effective level of security in the system.

By providing support for multiple trait types, we are able to define a robust user model that considers multiple independent sources of information when determining user identity. This approach allows for the handling of irregularities that occur when one trait type does not produce the same user match as another trait type. For instance, our model could produce meaningful output in a system that receives multiple identity matches for the same user. This approach also allows the system to identify and, at times, correct malfunctioning or compromised sensors.
4.2.2 Sensors

Our system model consists of distributed sensors that collect identity information that is ultimately provided to an authentication service. Sensors are mechanisms designed to observe and capture user-specific traits from the environment. Some sensors require explicit user participation to complete the capture, while others are less intrusive and can record information without the user’s knowledge or active participation. These sensors observe user traits and forward relevant values to the authentication service where the information is interpreted and used to authenticate users. For example, in figure 7, sensor $S_v$ provides value $val_{x,v}$, which is the measured value of trait $x_v$.

It is important that data from each individual sensor be treated with respect to its past performance. Our authentication framework gathers information from a variety of sensors to infer user characteristics. For example, one sensor may capture biometric information such as user height and weight while another can record images that are later used to approximate facial features. Such sensors can be used to non-intrusively gather information from the environment and produce output that, when combined, provides a stronger, more reliable identification measure than a single source could alone provide. Our model for Parameterized Authentication takes into account two important metrics of sensor functionality—the overall accuracy of captured data and the trustworthiness of a sensor.

4.2.3 User-Attribute Database

We have defined a user-attribute database that maintains a collection of traits to define users in the system. A particular user definition consists of the collection of traits that define that user. We can think of these features for user $u$ as a feature vector $x_u$, where $x_u$ is the $r$-dimensional vector

$$x_u = (x_{u1}, x_{u2}, \ldots, x_{ur})$$

and $x_{ui}$ is the stored value for trait $i$ associated with user $u$. In our model, it is assumed that user attribute-values are sent through a secure channel to the user-attribute database.
before they are permitted to interact with the system. This enables the sensors to identify legitimate users the moment their attribute-values are observed.

In our model, accuracy describes the quality of a “snapshot,” or specific instance, of user identification. Instead of being assigned to a particular sensor, accuracy is used to determine how well an observed trait matches the value stored in the user-attribute database.

4.2.4 Attribute-Matching Service

The Attribute-Matching Service is responsible for collecting sensor output, or observed user attributes, and comparing it with data stored in the user-attribute database. This service computes accuracy by comparing the stored definition for a user with the observed traits that are obtained from sensors. If a collection of sensors assemble a feature vector \( y \), it can be compared directly to the associated values stored in \( x \) to determine how well the observed features match those stored in the system. Accuracy measures the distance between these vectors. We discuss this further in section 4.3 and present a working example to better illustrate how accuracy is computed.

4.2.5 Trust Analysis Engine

We define trust as a measure of the system’s confidence in a particular sensor; it reflects the quality of the information provided by a given sensor in the system. Clearly, this measure of capability can only be refined over time, through a series of experiences that are recorded with the sensor. For instance, when sensors identify or observe traits in the environment, the quality of the experience should be reflected in the trust value that is assigned to the agent. A sensor that provides a series of positive experiences should be rewarded with a high trust value, while a sensor that provides negative experiences is penalized with a low trust value. It is this collection of positive and negative experiences that is used to compute the trust value for each sensor.

We have identified three possible outcomes that can result from an authentication process. Positive user identification captures instances in which the correct user is identified or
in which an unauthorized user is prevented from obtaining credentials from the system. A denial of service results when a legitimate user is prevented from being authenticated due to malicious tampering with sensors or authentication data. Similarly, a compromise has taken place when a user obtains credentials belonging to another user. Any interaction that leads to a denial of service or a system compromise is considered to be a negative experience and the trust measures that contribute to a negative experience are subsequently degraded. Likewise, the system attempts to reinforce positive experiences by assigning higher trust values to sensors that consistently provide data to the authentication service which leads to correct authentication.

Trust is a dynamic metric that evolves over time; one static value assigned to a sensor or agent would be an inappropriate use of this value, as it is intended to allow the system to quantify its belief in the overall capabilities of each individual sensor based on observed behavior of the sensor. While each individual interaction is important, it is the trust value that enables the system to determine how much trust can be placed in each sensor reading.

4.2.6 Audit Service

On-line scrutiny of all authentication decisions may not be possible for a number of reasons. Therefore, selective authentication decisions may be logged to enable subsequent off-line examination. For example, if a security violation is detected, a log can be used to reconstruct the sequence of interactions that led up to the violation. This would allow the system to revise the experience values associated with the sensors that contributed to the incorrect authentication; instead of contributing to a positive experience, the sensors would receive a negative rating for the interaction.

In addition, the audit service would be responsible for processing log information to produce feedback that is sent to the Trust Analysis Engine. This feedback is used to maintain evolving trust values for each of the sensors. Historical data can also be stored and accessed from the audit service.
4.2.7 Authentication Service

The authentication service receives input from the Trust Analysis Engine and the Attribute-Matching Service that is combined to derive the Authentication Parameter. The output from this service is based on input supplied by one or more sensors. We identify a set of useful design principles that guide the derivation of an Authentication Parameter from the information received. We will revisit these principles after presenting our model in detail to see how well it meets the intended design and what limitations, if any, exist.

**Principle 1: Accuracy.** Accuracy measures the similarity between a stored trait value and the value of a trait observed by a sensor. There are many ways to compute similarity and the accuracy measure should address the need for normalized results so comparisons can be made. In addition, weighted features should be used to indicate importance, and perform computations involving the combination of multiple traits.

**Principle 2: Evolution of Trust through experience.** Trust is a measure of confidence held by the authentication service in a particular sensor; it represents a reputation that is established through consistent behavior and observed over a series of experiences, both positive and negative. Trust should increase slowly to allow reputation to build between the authentication service and sensor. Likewise, once negative experiences are detected, trust should decrease quickly to defeat malicious or compromised sensors that attempt to improve their reputation through a short-run of positive performance.

**Principle 3: Combining Trust and Accuracy.** When trying to determine a user’s authenticity, the authentication service analyzes input provided by each relevant sensor. The authentication service forms an opinion by taking sensor input and adjusting it to reflect the current level of trust held by the sensor. This opinion, generated for each sensor, should reflect both certainties and uncertainties with regard to user identity, giving both trust and accuracy important roles in generating the opinion.

**Principle 4: Consensus of Opinions.** Once a set of individual opinions have been collected, they must be combined to generate the authentication parameter. Since an opinion is
based on input from one sensor, the consensus of opinions reflects the system’s belief and confidence in a particular authentication decision. When sensor opinions are in agreement, certainty should increase. Conflicts in opinion should be indicated through increased lack of confidence in the authentication decision.

Principle 5: Evaluation order independence. The derived conclusions of the model, namely the Authentication Parameter, should be independent of the order in which sensor input is considered. The Authentication Parameter is ultimately based on a consensus from multiple input sources; the order in which the inputs are considered should not alter the final output. This principle allows additional sensor inputs to be factored into an Authentication Parameter that has already been computed.

Principle 6: Property dependence and feedback. The value of the model’s trust and accuracy parameters should impact the feedback cycle in which trust is recomputed and returned to the system. In other words, an untrusted sensor that provides a negative experience will be punished less than a highly trusted sensor that provides a negative experience. This principle protects the model from being significantly harmed by a trusted sensor that is malfunctioning; it also allows misbehaving sensors to correct their behavior and return to a trusted status over time.

Principle 7: Scalability. It should be possible to collect data from a large number of sensors, to easily update the parameters when new sensors are included in a computation, and to implement computations of significant complexity that do not result in a denial-of-service (e.g., situations in which many sensors form a consensus, complex attribute-matching algorithms, or trust analysis that tests boundary conditions).

Principle 8: Robustness. The authentication parameter should be designed to be resilient to manipulations of its model by misbehaving entities, and its sensitivity to various forms of misbehavior should be made explicit.
4.3 Deriving the AP

The authentication service builds an authentication opinion by collecting input from one or more sensors with information relevant to the current user or request. An opinion is formed for each independent source of input and has two measures that impact its outcome. The first measure, accuracy, measures the similarity between a user trait that is observed by a sensor and the value of the same trait that is stored in a user signature. The second measure, trust, represents the reputation a sensor has with the authentication service; it measures consistency between past results and the output of an individual sensor. In this section, we further describe accuracy and trust and show how they are derived independent of one another. We then detail the process of combining trust and accuracy to produce the Authentication Parameter and discuss one approach to using the AP with existing systems.

4.3.1 Accuracy

In our model, accuracy is defined as a similarity measure between a stored trait value and an observed trait value. When a sensor provides input to the system, that input data is compared with stored identity data to determine if a match exists. A matched trait indicates that the value of an observed trait is similar to the value that identifies one or more known users. In order to account for variations between stored and observed data, a perfect match is not always required by the system. The closeness or quality of a match, therefore, is reflected in the accuracy value.

In order to assess how well an observed trait $x'$ matches a stored trait $x$, we consider a distance measure $d(x', x)$ such that

$$d(x', x) = \begin{cases} 
\text{large} & \text{when } x', x' \Rightarrow \text{mismatched traits} \\
\text{small} & \text{when } x', x' \Rightarrow \text{similar traits}
\end{cases}$$

The most obvious measure of similarity (or dissimilarity) between two measurements is
the distance that exists between them. Using a simple threshold-based clustering procedure, we can easily compute distances between all pairs of samples and

\[
\text{assign } x^e, x^s \text{ to } \begin{cases} 
\text{the same cluster} & \text{if } d(x^e, x^s) \leq d_0 \\
\text{different clusters} & \text{if } d(x^e, x^s) > d_0
\end{cases}
\]

Using this approach, a larger distance measure implies greater dissimilarity. The distance between two traits in the same cluster will be significantly less than the distance between two traits in a different cluster. Using \( d_0 \) as a distance threshold, observed traits that are placed within a cluster are sufficiently close in value to match a stored trait \( x^s \) and, therefore, serve as potential candidates for users who may have made the request.

Suppose that we assume that an observed trait \( x^e \) is clustered with a stored trait \( x^s \) if the Euclidean distance between them is less than some threshold \( d_0 \); clearly, the selection of \( d_0 \) is very important. If \( d_0 \) is very small, each trait will form an isolated cluster. Likewise, if \( d_0 \) is very large, all observed trait values will be assigned to the same cluster and, therefore, appear to represent a single user.

Similarity and distance measures have been explored in a variety of domains and can be used to compare one or more identifying features. Some comparison studies exist among similarity measures and indicate that different similarity measures perform best when coupled with the appropriate set of attributes. Clearly, the method for computing accuracy is implementation-specific.

Some similarity measures will weigh attributes, assigning more significance to one over another. Consider the following example in which the observed and stored traits being compared are constructed as vectors in a \( T \)-dimensional Euclidean space. And the weighted Euclidean distance between two traits (or trait vectors) is defined as

\[
d_{w}(x^e, x^s) = \sqrt{\sum_{t=1}^{T} w_t \cdot (x^e_t - x^s_t)^2}
\]
where $T$ is the number of attributes and $\vec{w} = (w_1, w_2, \ldots, w_T)$ is the weight vector. $d_S(\vec{x}, \vec{z})$ is defined as the accuracy of a measurement that is provided by sensor $S$ when a certain user is to be authenticated.

Although there are other approaches for computing distance, the example included above illustrates one method for obtaining an accuracy measure. Some instances will require that the attribute-matching service produce a distance measure between only two points. Other instances will require that a collection of identifiers or traits be used in the computation. For instance, a vision-based sensor will attempt to collect a series of traits that are combined to produce a single user identifier. This collection forms a vector that is compared with one stored by the system. Other sensors will produce more primitive output that will require less intensive comparisons.

4.3.2 Trust

In our model, the system consists of sensors that observe user features and report their findings to an authentication service. The authentication service must use these findings to identify the source of an access request. When taking input from multiple sensors, the authentication service may be presented with conflicting information that could either represent a compromised sensor, a failed sensor or a malicious agent trying to access the system.

When faced with seemingly incomplete or conflicting input, the system must decide whether it can trust each sensor and the associated output. Trust, as used in our model, is defined as “trust not to betray”. This definition allows us to focus on event independence. For example, sensors $a$ and $b$ are independent and provide unique perspectives and identification information on users in the environment; a conflict between the two can signify a number of things, including an attempted system compromise or an unauthorized user.

An untrusted sensor can claim to provide accurate input yet it could lead to incorrect results. Thus, accuracy captures how closely observed traits are to stored values, and trust
captures the quality of the input provided by a sensor.

We make use of techniques from Bayesian statistics to derive and maintain the trust metrics for individual sensors. Bayes’ theorem is based on the subjective definition of probability as “degrees of belief.” This approach assigns probabilities to hypotheses, allowing for the combination of a priori judgments and experimental information. Trust, as defined in our model, is not a static value; it involves a set of uncertainties that are refined over time through experiences and interactions. This makes Bayesian logic a natural mechanism for evolving trust values in our model.

Bayesian analysis begins with the declaration of a hypothesis. In our model, the hypothesis $T$ is straightforward: sensor $S_i$ is trustworthy. A probability, $P(T)$, is assigned to the hypothesis and specifies the current level of trust associated with sensor $S_i$. The initial value for $P(T)$ can be based on an organization-wide security policy (e.g., “All devices are to be assigned an initial trust value of $X$”), a determination made by the device manufacturer, or through other means such as independent testing and analysis. Regardless of how it is derived, all sensors are assigned initial values for $P(T)$ that will serve as the starting value for the trust metric. By using Bayesian statistics, $P(T)$ assumes dynamic properties and is allowed to evolve over time as the system obtains additional information related to the hypothesis; the model also allows for the handling of uncertainties and any errors that may occur.

After defining the initial hypothesis and the associated probability, Bayes’ approach attempts to refine $P(T)$ by calculating the effect of a correlated event on the original hypothesis. Our model focuses on an event $E$ that captures the outcome of an authentication experience. We define $E$ to be a binary event that is either positive or negative and reflects the overall trust in a single experience. We then compute the following probability using Bayes’ theorem, where $T$ is the hypothesis or current trust value, and $E$ is the value of the experience:

71
\[ P(T|E) = \frac{P(E|T) \cdot P(T)}{P(E|T) \cdot P(T) + P(E|\neg T) \cdot P(\neg T)} \]

In order to calculate \( P(T|E) \) we must provide additional details and measurements. First, the value of \( P(E|T) \) will vary depending on the outcome of the experience. If the event is positive, \( P(+E|T) = a \), where \( a \) is the accuracy associated with the reading. This is due to the definition we have provided for an experience – the probability of an event being positive, if the device is trustworthy, is directly tied to the accuracy of the result. Likewise, a negative experience produces \( P(-E|T) = (1 - a) \).

In addition, we define \( P(+E|\neg T) = a \). This value is the probability that a positive event will occur even when the sensor is not trustworthy (e.g., when the hypothesis is invalid). The value for \( a \) is predetermined and indicates a lower-bound threshold for untrusted activity. Our model assumes that a compromised sensor will provide some positive experiences as it attempts to evade detection and exclusion. A high value for \( a \) implies that when a sensor is compromised, its malicious behavior will be caught quickly and it will no longer be used in authentication decisions. Thus, if a compromised sensor wants to damage the system over a longer period of time, it must limit incorrect results that it provides and behave more like a good, trustworthy sensor.

This results in two formulas for computing trust when an event occurs. If the event is positive, we compute

\[ P(T + E) = \frac{a \cdot t}{a \cdot t + a \cdot (1 - t)} \]

where \( a \) is the accuracy of the input, \( t \) is the current trustworthiness of the sensor providing the input, and \( 0 < a < 1 \).

Likewise, if the event is negative, we compute
\[ P(T|E) = \frac{(1-a) \cdot t}{(1-a) \cdot t + (1-a) \cdot (1-t)} \]

To demonstrate the effectiveness of this formula, consider an example where sensor trust is \( t = 0.9 \) and the accuracy of the input has been determined to be \( a = 0.8 \). The security administrator has predefined \( a = 0.7 \). If the current experience is deemed positive, we evolve the trust value as follows

\[ P(T|+E) = \frac{0.8 \cdot 0.9}{0.8 \cdot 0.9 + 0.7 \cdot (1-0.9)} = 0.91 \]

Similarly, if the current experience is determined to be negative, we evolve trust in the opposite direction

\[ P(T|-E) = \frac{(1-0.8) \cdot 0.9}{(1-0.8) \cdot 0.9 + (1-0.7) \cdot (1-0.9)} = 0.85 \]

In this example, trust is increased when the experience is positive and decreased with the experience is negative. The degree to which a positive or negative experience changes the trust value depends largely on the accuracy of the input. As defined in the design principles, a highly trusted sensor that produces incorrect input that was originally flagged as less-than-accurate is punished less than a highly trusted sensor that produces incorrect input but flagged it as highly accurate.

Once \( P(T|E) \) is computed, it provides a revised trust value for a given sensor. This updated measure replaces the current trust value, \( P(T) \), and is used as the foundation for future refinements under the Bayes approach. We previously defined Design Principle 6 in which we detailed a “feedback cycle” for recomputing trust and returning it to the system. Our approach to evolving trust by computing \( P(T) = P(T|E) \) provides this essential feedback cycle and allows us to react accordingly when sensors malfunction or misbehave.
As previously discussed, accuracy measurements have a significant impact on the evolution of trust values. Highly trusted sensors that provide incorrect input that is tagged with a low accuracy value will be punished less than similar sensors that provide incorrect input that is tagged, for one reason or another, with a high accuracy value. For example, consider a negative experience with a sensor that has a trust value \( P(T) = 0.9 \) and \( \alpha = 0.6 \). An accuracy rating of \( a = 0.9 \) would produce the following:

\[
P(T) - E = \frac{(1-a)(1-\alpha)}{(1-a)(1-\alpha)+(1-\alpha)}
\]

\[
= \frac{0.1 \times 0.9}{0.1 \times 0.9 + 0.1 \times 0.1}
\]

\[
= \frac{0.09}{0.1}
\]

\[
= 0.09
\]

While an accuracy rating of \( a = 0.6 \) would produce the following:

\[
P(T) - E = \frac{(1-a)(1-\alpha)}{(1-a)(1-\alpha)+(1-\alpha)}
\]

\[
= \frac{0.4 \times 0.9}{0.4 \times 0.9 + 0.4 \times 0.1}
\]

\[
= \frac{0.36}{0.4}
\]

\[
= 0.9
\]

In the second scenario, an accuracy rating of \( a = 0.6 \) causes the trust value to stay at \( P(T) = 0.9 \) despite the negative experience. This is because the accuracy was such that it did not warrant a decrease in overall trust for the device. This example illustrates the dependencies that exist between parameters. Trust is a measure that evolves over time through experiences. Although the experiences are recorded as either positive or negative, the overall impact of those experiences is determined by the accuracy value associated with each experience. A higher accuracy rating can make significant difference in whether a principal is properly authenticated. Therefore, our model provides a mechanism that
aims to reward sensors that contribute to more accurate and trustworthy authentication parameters, while punishing those that degrade the quality of the output.

Finally, we can compute trust in a more efficient manner, allowing it to be computed quickly and on-demand, without the need to determine \( P(T|E) \) for each experience. Using our function for trust evolution, the trust after \( n \) experiences can be written recursively as follows, with \( k \) positive and \( m \) negative experiences.

We begin by defining the base trust value, \( t_0 \), which is assigned to each individual sensor by the security administrator. It can be easily shown that after \( n \) experiences, out of which \( k \) are positive and \( m \) are negative,

\[
t_n = \frac{\left( \prod_{i=1}^{k} a_i \right) \cdot \left( \prod_{j=1}^{m} (1 - a_j) \right) \cdot t_0}{\left( \prod_{i=1}^{k} a_i \right) \cdot \left( \prod_{j=1}^{m} (1 - a_j) \right) \cdot t_0 + \alpha^k \cdot (1 - \alpha)^m \cdot (1 - t_0)}
\]

We will discuss this computation further when we provide validation of the model.

4.3.3 Authentication Parameter

The authentication service builds an initial opinion regarding the user’s authenticity by collecting sensor observations that are relevant to the current user or request. As previously discussed, there are two measures associated with each sensor that affect the value associated with the observation. The first is a measure of trust that exists between the authentication service and each sensor. The second measure is one of accuracy.

With trust, accuracy and multiple input sources influencing the authentication process, it is clear that authentication can no longer produce output that is binary in nature. The lack of “perfect” knowledge in the authentication process yields uncertainty. Therefore, our model must account for this uncertainty by forming an opinion that best reflects the input received, taking into account trust and accuracy values associated with that input. This opinion, which translates into degrees of belief or disbelief, will be used as the authentication parameter and indicates the quality of the authentication process.

In [47], Jatsang defines a framework for artificial reasoning called Subjective Logic that
consists of a belief model called opinion. Subjective Logic was developed to mathematically describe and manipulate subjective beliefs; it is an extension of standard logic that uses continuous uncertainty and belief parameters instead of only discrete truth values. We have extended the Subject Logic framework to be used with Parameterized Authentication as it provides a foundation for the handling of uncertainties and the forming of conclusions based on insufficient evidence.

Similar to Jøsang’s approach, we assume that knowledge about the world (obtained through a sensor) is never perfect and it may be impossible to verify a user’s identity with absolute certainty. Given this imperfect knowledge, it is impossible to know authoritatively whether a user has been properly identified, so a sensor can only have an opinion about the observation. For a single opinion about a user’s authentication, we assume that

\[ b + d + u = 1, \quad [b, d, u] \in [0, 1] \]

where \( b, d, \) and \( u \) represent belief, disbelief and uncertainty respectively. A situation in which there is zero uncertainty is equivalent to the traditional probability model.

Our method for assigning values to the \([b, d, u]\)-tuple differs from that proposed by Jøsang. His approach involves mapping the opinion space to an evidence space that consists of a probability certainty density function. Our evidence space, however, consists of trust and accuracy measures obtained from sensors. Therefore, we let \( \omega_x = [b_x, d_x, u_x] \) be a single sensor’s opinion about the authentication of user \( x \). We now define \( \omega_x \) as a function of trust and accuracy measures that have been obtained from the sensor:

\[
\omega_x = \left\{ \begin{array}{l}
  b_x = t \cdot a \\
  d_x = t \cdot (1 - a) \\
  u_x = (1 - t)
\end{array} \right.
\]

Here, \( t \) is a measure of trust for the sensor that is providing the authentication data and \( a \) is the accuracy of the observed trait. Our definition for \( \omega_x \) makes it clear that authentication based on sensor output is subject to uncertainties. When dealing with a potentially
untrusted or inaccurate device, it is important to include uncertainty to allow for more correct results than what is possible with a traditional probabilistic approach.

Subjective Logic contains several different operations, with the most relevant to our work being consensus. A consensus opinion consists of combining two or more independent opinions about the same proposition (e.g., “The user’s identity as Bob can be verified”) into a single opinion. In our notation, superscripts indicate ownership (e.g., sensors) and subscripts indicate the proposition to which the opinion apply. For example \( \pi^{s_1}_p \) is an opinion held by sensor \( S_1 \) about the truth of proposition \( p \).

To demonstrate the consensus operation, let \( \pi^{s_1}_p \) represent the opinion held by the authentication service regarding the identity provided by sensor \( S_1 \) of the user making an access request. Let \( \pi^{s_2}_p \) represent the opinion held by the authentication service regarding the identity of the same user based on claims made by sensor \( S_2 \). The resulting consensus opinion,

\[
\pi^{s_1,s_2}_p = \{ b^{s_1,s_2}_p, d^{s_1,s_2}_p, u^{s_1,s_2}_p \}
\]

such that

\[
b^{s_1,s_2}_p = \left( b^{s_1}_p u^{s_2}_p + b^{s_2}_p u^{s_1}_p \right) / k
\]

\[
d^{s_1,s_2}_p = \left( d^{s_1}_p u^{s_2}_p + d^{s_2}_p u^{s_1}_p \right) / k
\]

\[
u^{s_1,s_2}_p = \left( u^{s_1}_p u^{s_2}_p \right) / k
\]

where \( k = u^{s_1}_p + u^{s_2}_p - u^{s_1}_p u^{s_2}_p \) such that \( k \neq 0 \). We follow Jassang’s notation by using the symbol \( \oplus \) to designate this operation, yielding

\[
\pi^{s_1,s_2}_p = \pi^{s_1}_p \oplus \pi^{s_2}_p
\]

Our use of the consensus operator is only for the combination of input from sensors.

The Authentication Parameter is found in the final \( \omega \), after consensus has been computed for all relevant opinions. The value of interest is the \( b \), that reflects the overall belief.
in the user’s authenticity based on input from multiple sensors. However, this value alone is insufficient to describe the quality of authentication. The entire \( (b, d, u) \)-tuple is retained for a variety of reasons, including the addition of supplemental input through the consensus operation and the comparison of results.

Parameterized Authentication can be implemented using a role-based approach to achieve a high performance and flexible authentication service. Fundamentally, a role is a grouping mechanism that is used to categorize subjects based on various properties. We propose that roles be used in our authentication service because they provide the most generalized way of describing subjects. For example, a user identification that maintains a high degree of belief may be authenticated to a different role than an identification with a low belief value. This approach would also allow the authentication parameter to co-exist with popular role-based access control models such as \([32, 83, 21]\).

4.4 Validation

In order to properly validate our model for parameterized authentication, we revisit the design principles presented in 4.2.7. We begin by running a series of simulations and use the results to demonstrate that the results of our model are consistent with the design principles that were stated earlier.

Design Principle 1: Accuracy The method presented in section 4.3.1 to compute authentication accuracy meets the guidelines presented in our model’s design principles. The weighted accuracy function provides an approach that measures similarity and takes into account relative importance of the measurement of a trait value by a sensor. The assigned weight vector is provided by a security administrator to declare the maximum level of accuracy that is allowable based on each sensor’s inherent capabilities.

Design Principle 2: Evolution of Trust through Experience The following figures demonstrate how trust evolves for various sensor “types” as experiences are encountered
over a period of time. In figure 8, a sensor is defined with an initial trust value of $t = 0.9$, a sensor reading rated with accuracy $a = 0.9$, and an administrative $\alpha$-value set to $\alpha = 0.6$. The solid line in the plot shows how trust changes when ten positive experiences are encountered with the sensor. Likewise, the dotted line shows a similar evolution process for trust except for a lower setting of the initial trust value than found in the previous example, with $t = 0.6$.

![Figure 8: 10 Positive Experiences, Varying Trust](image)

In figure 9, a sensor is defined with an initial trust value of $t = 0.9$, a sensor reading rated with accuracy $a = 0.9$, and an administrative $\alpha$ set to $\alpha = 0.6$. The solid line in the plot shows how trust changes when ten negative experiences are encountered with the sensor. Additionally, the dotted line shows a similar evolution process for trust except for a lower setting of the initial trust value than found in the previous example, with $t = 0.6$.

As illustrated in both figures, trust is slow to build and quick to degrade. These results follow in line with the expectations set forth in the second design principle. These properties allow for trust reputation to build over time, while quickly punishing any malicious or compromised components.

Figure 10 provides a look at trust evolution when both positive and negative experiences
are encountered in a single sensor. The solid line illustrates how trust is increased when a series of 5 consecutive positive experiences are encountered and, subsequently, how trust is quickly degraded when it is followed by 5 consecutive negative experiences. The dotted line shows how trust is decreased during the time period when a random collection of experiences occur (e.g., an equal number of alternating positive and negative experiences).

As above, these properties remain consistent with our design principles: trust is slow to build, quick to fall, and our model always errs on the side of caution – random experiences reflect an inconsistency with the sensor and cause trust to degrade.

**Design Principle 3: Combining Trust and Accuracy** By utilizing the Subjective Logic framework, our model is capable of describing and manipulating subjective beliefs, or sensor opinions. Our evidence space is comprised of the trust values maintained for each sensor and the accuracy of the each sensor output. As previously discussed, the “opinion” is a function of trust and accuracy, such that
The consensus operation described in section 4.3.3 allows for the combination of two independent opinions. Clearly, an opinion consists of more than just trust and accuracy. By generating belief, disbelief and uncertainty values, the resulting tuple reflects a multitude of information that could not be conveyed with a single value. Combining trust and accuracy in this manner reflects both certainties and uncertainties, without placing any undue weight or emphasis on a particular component of the model. These properties are aligned with the design principles that govern the combination of trust and accuracy.

**Design Principle 4: Consensus of Opinions** Figure 11 provides a table defining two sensor types and shows how an Authentication Parameter is affected when more sensors are used to form a consensus. A "good" sensor is one with a high trust, or belief, value. A "bad" sensor is one that has low trust and high distrust. With both sensor types, uncertainty results from the lack of perfect trust in sensors and perfect accuracy from their readings.
<table>
<thead>
<tr>
<th>Sensor Desc.</th>
<th>b</th>
<th>d</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Detail</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good</td>
<td>0.85000</td>
<td>0.05000</td>
<td>0.10000</td>
</tr>
<tr>
<td>Bad</td>
<td>0.20000</td>
<td>0.70000</td>
<td>0.10000</td>
</tr>
<tr>
<td>2 good</td>
<td>0.89473</td>
<td>0.05263</td>
<td>0.05263</td>
</tr>
<tr>
<td>3 good</td>
<td>0.91071</td>
<td>0.05357</td>
<td>0.05357</td>
</tr>
<tr>
<td>4 good</td>
<td>0.91891</td>
<td>0.05405</td>
<td>0.02702</td>
</tr>
<tr>
<td>5 good</td>
<td>0.92391</td>
<td>0.05434</td>
<td>0.02173</td>
</tr>
<tr>
<td>6 good</td>
<td>0.92727</td>
<td>0.05454</td>
<td>0.01812</td>
</tr>
<tr>
<td>7 good</td>
<td>0.92968</td>
<td>0.05468</td>
<td>0.01562</td>
</tr>
<tr>
<td>8 good</td>
<td>0.93150</td>
<td>0.05479</td>
<td>0.01359</td>
</tr>
<tr>
<td>9 good</td>
<td>0.93297</td>
<td>0.05487</td>
<td>0.01219</td>
</tr>
<tr>
<td>10 good</td>
<td>0.93406</td>
<td>0.05494</td>
<td>0.01098</td>
</tr>
<tr>
<td>1 good, 1 bad</td>
<td>0.55263</td>
<td>0.39473</td>
<td>0.05263</td>
</tr>
<tr>
<td>1 good, 2 bad</td>
<td>0.44642</td>
<td>0.51785</td>
<td>0.03571</td>
</tr>
<tr>
<td>1 good, 3 bad</td>
<td>0.39189</td>
<td>0.58108</td>
<td>0.02702</td>
</tr>
<tr>
<td>1 good, 4 bad</td>
<td>0.35896</td>
<td>0.61956</td>
<td>0.02173</td>
</tr>
<tr>
<td>1 good, 5 bad</td>
<td>0.33636</td>
<td>0.64545</td>
<td>0.01812</td>
</tr>
<tr>
<td>1 good, 6 bad</td>
<td>0.32031</td>
<td>0.66406</td>
<td>0.01562</td>
</tr>
<tr>
<td>1 good, 7 bad</td>
<td>0.30821</td>
<td>0.67808</td>
<td>0.01369</td>
</tr>
<tr>
<td>1 good, 8 bad</td>
<td>0.29878</td>
<td>0.68902</td>
<td>0.01219</td>
</tr>
<tr>
<td>1 good, 9 bad</td>
<td>0.29120</td>
<td>0.69780</td>
<td>0.01098</td>
</tr>
<tr>
<td>1 good, 10 bad</td>
<td>0.28300</td>
<td>0.70500</td>
<td>0.01000</td>
</tr>
<tr>
<td>2 bad</td>
<td>0.21052</td>
<td>0.73682</td>
<td>0.05263</td>
</tr>
<tr>
<td>3 bad</td>
<td>0.21426</td>
<td>0.75090</td>
<td>0.03571</td>
</tr>
<tr>
<td>4 bad</td>
<td>0.21621</td>
<td>0.75675</td>
<td>0.02702</td>
</tr>
<tr>
<td>5 bad</td>
<td>0.21791</td>
<td>0.76070</td>
<td>0.02173</td>
</tr>
<tr>
<td>6 bad</td>
<td>0.21818</td>
<td>0.76363</td>
<td>0.01812</td>
</tr>
<tr>
<td>7 bad</td>
<td>0.21875</td>
<td>0.76526</td>
<td>0.01562</td>
</tr>
<tr>
<td>8 bad</td>
<td>0.21917</td>
<td>0.76712</td>
<td>0.01369</td>
</tr>
<tr>
<td>9 bad</td>
<td>0.21951</td>
<td>0.76829</td>
<td>0.01219</td>
</tr>
<tr>
<td>10 bad</td>
<td>0.21978</td>
<td>0.76923</td>
<td>0.01098</td>
</tr>
</tbody>
</table>

**Figure 11:** Consensus of Opinions

The results in figure 11’s table are consistent with design principle 4 in that certainty only goes up when the majority of opinions are in agreement. Any conflicts or increased disbelief cause a decline in confidence and, as a result, a lowering of the belief value in the authentication parameter.

82
Design Principle 5: Evaluation order independence. The fifth design principle can be validated by showing that order does not matter in evaluating trust. Earlier, we demonstrated that the trust after n experiences can be expressed as a function, with k positive and m negative experiences. This approach enables us to produce a trust value that is based on multiple experiences and computed in any order. Since trust is based on the collection of experiences, regardless of order, this design principle applies to our model and ensures that trust can be computed efficiently.

Design Principle 6: Property dependence and feedback. Our approach for managing trust using Bayesian statistics takes into account current trust values and accuracy readings. In addition, the model provides a feedback loop that evaluates the quality of an experience and incorporates this into an evolving trust value. By allowing trust parameters to evolve and accuracy readings to impact feedback cycles, our model is protected from misbehaving and malfunctioning sensors. Property dependence and feedback are crucial components of our model for deriving a dynamic, robust authentication parameter.

Design Principle 7: Scalability. Given the efficiency of computing trust across multiple experiences and the straightforward method for generating consensus, we believe our model is scalable and can accommodate input from a large number of sensors. In future work, we will provide experience with simulated sensor interactions that will further test the scalability in a real-world setting.

Design Principle 8: Robustness. This design principle aims to produce a resilient authentication parameter that is not subject to manipulations by misbehaving entities. Figure 12 shows the result of a consensus operation that has been performed by one form of misbehavior—a "random," or inconsistent, sensor. In this example, a random sensor is defined as one with an initial trust value of \( t = 0.5 \) and an accuracy reading of \( a = 0.5 \). These values yield a "random opinion" of:
$$\omega_s = \begin{cases} 
    \beta_s = t \cdot a = 0.25 \\
    d_s = t \cdot (1 - a) = 0.25 \\
    \alpha_s = (1 - t) = 0.50 
\end{cases}$$

Figure 12: 10 Random Sensors forming Consensus

As indicated in the figure, uncertainty decreases as more random sensors contribute an opinion to the authentication process. However, the randomness of the sensors results in an equal, but slow, rise of belief and disbelief values. Given the relatively low accuracy and trust values associated with a random sensor, the consensus operation does not allow these sensors to have a significant impact on the overall AP that results. In fact, the AP value does not increase beyond 0.5 even when many sensors are polled.

4.5 Discussion

We have introduced the concept of Parameterized Authentication and detailed the process for computing a measure that captures the overall quality of an authentication process. In this section, we present some noteworthy aspects of parameterized authentication that did not receive sufficient discussion in previous sections.
4.5.1 Authentication Paradigms and Historical Data

In general, our model supports two basic scenarios through which a user can be authenticated. First, there is the push model in which the user "pushes" authentication credentials, in combination with an access request, to a service that will process her request. The second model is the pull model in which the user makes a request and the authentication service retrieves, or pulls, authentication credentials from sensors in the environment. The second model allows for a more dynamic approach to authentication and also requires mechanisms that support efficient sensor usage. For instance, if the user makes a request that can be granted without perfect authentication, it may be possible to "pull" information from a limited number of sensors. This procedure reduces some of the processing and bandwidth usage associated with the authentication process.

In these rich, interactive settings, a user's context, such as her location and the people and objects around her, can be very useful in making security-related decisions. Context, in this setting, is composed of any relevant information about the entities in the current interaction – including historical state information that details previous interactions.

We believe that the use of historical state information would supplement and enhance the accuracy of a security subsystem. In essence, we propose an authentication cache that would allow authentication decisions to be based on identification information currently stored in the system. Traditional implementations of caching technology were able to store often-requested information at the edge of the network, therefore speeding up site performance and lowering connection costs. Given the overwhelming success of caching, we propose that similar technology be applied to the security services in interactive environments such as the home.

Such a cache would pull user identification information from sensors in the environment and use it when making future access control decisions. For example, in the Aware Home users will be identified using a number of devices and technologies. Suppose that Alice swipes a smart card in order to gain access to the house via the front door. As she
enters, a camera is able to identify her with 86% accuracy and the Smart Floor is able to track her movements and maintain a positive identification as she moves throughout the house. When Alice attempts to access a restricted resource, the system is able to use all of the historical state information, including the data obtained via the smart card’s key signature, the camera, and the Smart Floor, to obtain a positive identification, and therefore an accurate authentication, for Alice, a legitimate resident and authorized user of the resource.

To our knowledge, the use of historical information in security-related decisions has never been included as an enhancement to the overall system. Instead, historical data has been viewed as stale and inconsequential because security parameters and policies are subject to dynamic change and variability. However, in the case of the Aware Home and other computationally rich environments, historical data can offer something that is highly desirable – more timely and efficient access to critical information. In addition, historical information can act as a supplement in cases where the current or active data set is inaccurate or incomplete. In the Aware Home example, the key signature belonging to Alice’s smart card may possess the authorization codes that are required for Alice to use the television. By supplementing the current identification results with historical information, the system can grant her access without requesting a second swipe of the smart card.

4.5.2 Authentication Refinement

There are many obstacles that must be overcome before we can build non-intrusive systems that provide security, while at the same time maintaining ease-of-use, functional requirements, and user privacy. In terms of user interaction, identification and authentication are essential services that support the system’s ability to enforce access rights and respond appropriately to user’s needs. A balance must be achieved between the requirements of the resource and the capabilities of the system. For instance, issues such as cost, practicality and privacy must all be addressed when providing security services for a resource. Over-protecting resources is as harmful to a system as under-protection. For example, a service
that is available to anyone may not require the user to be “fully authenticated” — this allows authentication resources to be reserved for the protection of more important resources. When protection is desired but privacy must be maintained, authentication of user “roles” instead of identities allows for the enforcement of access policies without infringing on a users privacy.

Other authentication services, such as Kerberos [51], enforce session limits that force tickets or credentials to expire after a specified period of time has elapsed. We provide a mechanism to enforce similar session limitations, but use a method of enforcement that compliments our model for parameterized authentication. Since the user credential, or AP, is actually a consensus of input from multiple sources that could potentially be collected over a period of time, credentials cannot simply timeout. Also, it may be desirable to have a credential based on a lower AP value expire before one that was based on more reliable sensor input.

To address these concerns, our model provides a decay function that decrease the authentication value over time to enforce session limits and ensure the authenticity of user credentials. The effect of this decay on an AP’s value can be modeled using a decay function $f(n)$ such that the AP decreases after some specified time has passed; the rate of decay should increase as more time passes. Higher AP values (e.g., those based on more trustworthy or accurate sensor inputs) are typically more reliable than lower AP values. After $t$ time-periods, the AP value will be equal to $y$.

We make use of the decay function to enforce session limits and ensure that AP values are fresh and up-to-date. As an AP is reduced and user access to resources is restricted, the AP can be refined through additional sensor polling. This approach guarantees that access to restricted resources is provided only to users with sufficient authentication credentials that are current, accurate and trustworthy.
4.6 Summary

We have introduced a new model for user authentication and have described how it can be used to secure dynamic applications in pervasive computing environments. The major benefit provided by this authentication model is its ability to provide adaptive services when faced with incomplete, inaccurate or unreliable security information.

We introduced the concept of parameterized authentication and have illustrated how it can allow continued functionality in a damaged, compromised or fundamentally inaccurate system. Furthermore, our authentication scheme is not limited by being binary in nature – users who cannot be fully authenticated by the system may still be given access rights based on their role and the level of confidence the system has in their identification.

Our notion of parameterized authentication provides a metric, the authentication parameter, that is based on trust and accuracy values associated with each authentication sensor and the output provided by those sensors. The authentication parameter is used to provide knowledge of the authentication process to an authorization service as a single, well-understood metric. We have presented several design principles that guided the design of our model and have used those principles to evaluate the model itself.

In addition, we have explored an enhancement to the Generalized Role-Based Access Control model that provides extensions to support the authentication parameter. We have discussed our initial evaluation of the model and provided experimental results regarding its effectiveness in securing context-aware applications.
CHAPTER 5

A CONTEXT-AWARE SECURITY ARCHITECTURE

We describe an approach to building security services for context-aware environments. Specifically, we focus on the design of security services that incorporate the use of security-relevant "context" to provide flexible access control and policy enforcement. We previously presented a generalized access control model that makes significant use of contextual information in policy definition. This chapter provides a concrete realization of such a model by presenting a system-level service architecture, as well as early implementation experience with the framework. Through our context-aware security services, our system architecture offers enhanced authentication services, more flexible access control and a security sub-system that can adapt itself based on current conditions in the environment. We discuss our architecture and implementation and show how it can be used to secure several sample applications.

5.1 Overview

The context-aware applications previously discussed present new and interesting security challenges. The transparent nature of a pervasive computing environment motivates the need for a security architecture that will transparently determine the sources of requests and handle a high degree of context changes. We can no longer assume that user "sessions" will persist for extended periods with the same authentication and authorization credentials. Thus, we must look at new access control models and authentication techniques that will operate effectively in these next-generation environments.
5.1.1 Context-Aware Authorization

Security policies in an information-rich environment like the Aware Home can potentially be quite complex. A policy can restrict access to information or resources based on several factors, including attributes pertaining to the subject, the resource or the environment. For example, subjects can be classified into roles such as "resident" or "guest." Access rights can depend on the subject's classification (e.g., "resident"), as well as on his or her actual 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.

While time and location are natural examples of environmental states 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 [68] 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 or if the request is an emergency. Unlike traditional access control models where requests are made explicitly by subjects, requests in the Aware Home may be triggered by changing environmental conditions. For instance, if an emergency situation is detected, a request for professional assistance can be generated and should be granted based on the context of the situation.

While GRBAC presents a powerful and flexible model for expressing access control policies for context-aware applications, the model itself clearly requires a more complex system architecture to support the extended roles and intricate policies that are made possible. In particular, the system architecture must support mechanisms to securely collect contextual information that is used to enforce access control policies. In addition, GRBAC requires that a separate authorization component be available to bind subject, object and environment roles together with an operation and corresponding permission. In Sect. 6, we present our approach to building this architecture and discuss the various components that provide the security infrastructure to context-aware applications.
5.1.2 Non-Intrusive User Authentication

Another challenge presented by a pervasive computing environment like the Aware Home involves relieving the user from the "burden" of authentication. Ideally, information available from sensors in the home should be used to automatically infer a subject’s security-relevant attributes (e.g., identity, role, location, etc.). Although it is possible for a resident to use a physical authentication token, it is undesirable to expect the user to carry such an object around at all times. Previous work with physical identification tokens such as the Active Badge system [99] have yielded useful results but are less practical for home use and unreliable for authentication (users can assume a different identity by simply carrying another person’s badge).

Several sensor-based technologies such as voice and face recognition can be deployed in the Aware Home to non-intrusively identify humans and track their movements. Many such techniques can establish the identity of a subject with only a partial level of certainty. Such "partial authentication" has important implications for access control models. In particular, some identification mechanisms are known to be more reliable than others. Our model for context-aware user authentication takes these differences into account and provides a mechanism for "parameterized authentication." Parameterized authentication allows a legitimate user to maintain access to a system even when the overall quality of his authentication is not 100%. We do this by granting subsets of access rights based on the current "authentication parameter," a metric that is based on trust in the devices that provide authentication data and the inherent accuracy of those devices. A related notion of various levels of authentication for a user was recently proposed in [33].

5.1.3 Secure Context Collection

Traditionally, interactive applications are limited to using input that is provided by users explicitly interacting with the system. As users move away from the traditional desktop computing model and move toward ubiquitous computing environments like the Aware
Home, there is a greater need for applications to leverage from implicit information, or context. These pervasive computing environments are rich in context, with users and devices moving around and interacting, computational services becoming available or disappearing over time, and a seamless flow of information that is available "wherever, whenever." This contextual information is usually not available to applications but can be useful in adapting the way in which they perform services and in changing the services they make available. In addition to making this information available to context-enabled applications, we envision it being used to enable context-aware security services.

The extensions found in GRBAC unify ideas from several existing access control models into a single model that captures all security-relevant state in a system. The unification of all relevant state into a single concept—that of a role—makes access control policies more flexible and easier to understand than those found in other models.

Clearly, our ability to secure applications using context is limited by our ability to securely collect context. If we use contextual information that is provided by a malicious entity, the security of our system is compromised. In addition to managing role complexity, any context-aware security architecture must also provide support for secure context collection.

![Logical Framework for Securing Context-Aware Applications](image)

**Figure 13:** Logical Framework for Securing Context-Aware Applications

92
5.2 Components

To address the problem of providing security to context-aware applications, we describe a Context-Aware Security Architecture (CASA). CASA provides a security infrastructure upon which emerging applications can be built. Figure 13 provides a high-level overview of the various logical components that comprise this security architecture.

We have implemented a prototype system that uses CASA to provide security services to applications running in an information-rich computing environment. Our implementation is built using the Java 2 Standard Edition Software Development Kit (J2SE SDK)[94]. In the following sections, we describe our instantiation of CASA and provide details on making these services available to context-aware applications.

```xml
<GRBAC_TABLES>
  <POLICY>
    <SROLE> Child </SROLE>
    <ROLE> Dangerous Appliance </ROLE>
    <ACTION> ALL </ACTION>
    <EROLE> Working Hours </EROLE>
    <PERMS> Deny </PERMS>
  </POLICY>
</GRBAC_TABLES>
```

Figure 14: Example GRBAC Policy Specification in XML

5.2.1 Security Management Service

The Security Management Service (SMS) is responsible for managing and storing system policies and role relationships, as specified by the security administrator. We separate the functionality of policy storage and runtime policy evaluation so as to allow for a more distributed and efficient system design. Policies that are enforced using the GRBAC model are defined in terms of roles. The SMS manages the relationships that exist between roles and provides appropriate mechanisms for secure storage and retrieval of policies.

In our architecture, policy enforcement is provided by an Authorization Service, while
environment role activation is managed by a separate logical service. The SMS allows for role manipulation to be performed and ensures that any associated roles or policies are updated accordingly. For example, if a security administrator were to make policy changes to an active system, the SMS would notify any role activation service (subject, object, environment) affected by the change.

In addition, the SMS provides a central location for backup and recovery. Since it is responsible for all policy management and for the bootstrapping of other components, the SMS can be distributed and protected more efficiently and effectively than an architecture that would distribute policy and management functionalities. For instance, the SMS could be implemented on top of a distributed data repository such as the Secure Store [53] that would provide data replication and enhanced availability in a potentially hostile environment.

Our implementation provides an SMS that is comprised of two separate components - a persistent storage mechanism that is responsible for storing policy-related data and a front-end processor that provides a communication interface between this storage component and other services in the framework. This modular design allows storage mechanisms to be easily replaced without requiring changes to the communication interface.

We have built an SMS that uses a relational database to provide storage for policies and associated data. Based on a local access control policy, other services can manipulate the database tables by making the appropriate calls into the SMS communication interface. In addition, the SMS is configured to provide limited support to the Authorization Service. For instance, an Authorization Service can request that the SMS process internal tables and return a hierarchical structure of role definitions.

CASA provides a distributed and modular architecture design. Although the SMS provides a centralized location for policy storage and retrieval, no other components are structured in the same way. Authorization, authentication and role-activation services can be centralized or distributed based on the environment in which they operate. In the Aware

94
Home, some resources may instantiate a local Authorization Service to perform their own resource access checks. Other limited-capability resources may opt to offload authorization to a central service. In either case, the SMS serves as the central console for policy updates and ensures consistency and well-formedness of policy and role definitions.

5.2.2 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. The Authorization Service retrieves policies from the SMS when a request is received and is consulted to determine if access to a particular operation should be granted or not.

Ubiquitous computing environments such as the Aware Home consist of many devices and services that will be centrally administered. The authorization service ensures that access rules are consistent and allows for all resources – regardless of processing capabilities – to enforce security policies.

The Authorization Service is responsible for retrieving policy definitions from the SMS and for determining whether a particular request should be granted or denied. Our implementation supports an Authorization Service that can boot in two different modes. The first mode does not store any authorization information locally. For every request, the Authorization Service must contact the SMS to retrieve a copy of the relevant policy. In the second mode, the Authorization Service stores all relevant policy definitions in a local runtime structure. The second mode provides greatly enhanced performance but also requires more resources at the Authorization Service.

To avoid the communication overhead associated with establishing connections to the SMS, the second mode stores a local copy of subject and object role definitions, as well as relevant policy in its runtime structure. When the service is started, it retrieves the relevant policy-related data from the SMS and flattens the hierarchical structures to store
them locally.

Since our system allows for multiple Authorization Services, it is useful for each Authorization Service to maintain a cached list of active environment roles (ERoles) that have already been evaluated. By assuming synchronized clocks between the various services in our architecture, we define a lifetime parameter that can be used to determine the lifetime of cached ERole status. In our implementation, the Authorization Service caches only active ERoles.

When an access request is received by the Authorization Service, it first determines what roles are active for the object being accessed. Based on this active object role set, policies are checked and the Authorization Service must determine if it requires information from the Authentication Service or ERAS. If so, appropriate calls are made to determine active subject roles and/or environment roles that are relevant to the access request. Policies are stored in the form of a tuple:

\(< SRole, ORole, Action, ERole, Perms >\)

where \( SRole \) specifies a subject role, \( ORole \) specifies an object role, \( Action \) specifies an operation, \( ERole \) specifies environment role(s) and \( Perms \) determine whether the action is granted or denied.

5.2.3 Environment Role Activation Service

```
ERole ::= < L.Exp > | < L.Exp > < L.Opr >
< L.Exp > ::= True | False | < M.Exp > < M.Exp > < C.Opr >
< M.Exp > ::= < Meta > | < Meta > < Meta > < M.Opr >
< Meta > ::= < Constants > | < Environment.Variable >
< L.Opr > ::= AND | OR | NOT
< C.Opr > ::= = | <= | > | >= | < | <=
< M.Opr > ::= + | - | * | /
```

*Figure 15: Environment Role Definition*
The Environment Role Activation Service (ERAS) maintains information on system state and manages role activation and deactivation based on conditions that are held by specified environment variables. It is through this component that an administrator specifies the variables that define environment roles and the conditions that must be met in order for the roles to be activated. This service interacts with one or more Context Management Service/sensors to ensure that system state is collected and the appropriate roles are activated when necessary.

Environment roles (ERoles) specify and capture environmental conditions that are relevant to access control. The Environment Role Activation Service (ERAS) is responsible for evaluating the status of ERoles. Environment roles are activated when certain environmental conditions are met. These conditions can change dynamically and hence the active role set is also subject to change. In our system, we have implemented an ERAS that evaluates role status on-demand. Our model maintains two possible states for each ERole, active and inactive, and we represent these with boolean values. We have chosen to implement ERoles as logical expressions. The major function of the ERAS is to evaluate these logical expressions and determine the state of any given ERole.

In a previous implementation, we used Conjunctive Normal Form (CNF) to represent ERoles (the logical expressions) but found that this approach made it difficult to specify additional operations on environment variables. We have since modified our implementation to one that utilizes a postfix format and allows all mathematical and logical operations (detailed below) to be treated in a unified fashion. In terms of our implementation, an ERole is defined as shown in Figure 15.

As initially described in Figure 15, <Environment.Variable> identifies the particular variable that is provided by a sensor, and <Constant> represents the constant values that need to be evaluated together with environment variables.

CASA data types include integer, float, string, and extended type. The data type extended type is introduced to group the types that need special operations (other than "$\div\$"
and "<>") on specially formatted data. We support four extended types in our current implementation of the system. They are *time, date, dayofweek,* and *month.* 

The *<Meta.Operation> will be performed according to the semantics of the *<Meta> s that participate the operation. For example, comparisons can be made on month values to determine that "December" > "March". Contained within the ERAS is a table that details all legal operations that are permitted on each data type and any combination of data types.

Figure 16 provides a formal definition for the *Party* ERole (equivalent to "Person.LivingRoom >= 10 and NoiseLevel.LivingRoom_DB >= 50"). ERoles can also inherit definitions from existing ERoles. For example, if ERoles *Party* and *Weekend* have already been defined in the system, a new ERole *Weekend.Party* can be defined simply through logical "AND" of the logical expressions from both of the existing ERoles. The newly created ERole inherits definitions from both of its parents and can also specify additional conditions that must hold in order for the role to be activated. Further discussion of the environment role model and related properties can be found in [20].

<table>
<thead>
<tr>
<th>Party</th>
<th>L.Exp1 L.Exp2 AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>L.Exp1</td>
<td>M.Expr11 10 &gt;=</td>
</tr>
<tr>
<td>L.Exp2</td>
<td>M.Expr21 50 &gt;=</td>
</tr>
<tr>
<td>M.Expr11</td>
<td>Person.LivingRoom</td>
</tr>
<tr>
<td>M.Expr21</td>
<td>NoiseLevel.LivingRoom_DB</td>
</tr>
</tbody>
</table>

*Figure 16: Example Environment Role Definition: Party*

In order to avoid communication delays that may be associated with contacting the SMS, the ERAS has been designed to maintain a local copy of ERole definitions (downloaded during a bootstrap process with the SMS). When contacted by the Authorization Service, the ERAS evaluates the necessary ERole(s) and returns the current status.

Experimenting with this initial configuration yielded interesting results - we discovered that not only was it necessary to account for time spent evaluating an ERole status, but we also had to take into consideration the time required to establish connections to each
CMS/sensor, retrieve the necessary data and perform evaluations for postfix expressions. Primarily, we found that the time required to establish a secure connection added significant overhead to the time spent completing a request.

To overcome the costs associated with external communication with both the SMS and sensors, we have implemented an ERAS that uses a caching mechanism to maintain the status of an evaluated ERole. Since environmental variables are subject to dynamic fluctuations in value, it is not appropriate to cache such variables. However, once an ERole is evaluated, it may be possible to cache its status for future use. Our implementation assigns lifetime and inactive_lifetime attributes to each ERole. Lifetime specifies the amount of time for which the cached value of the active ERole will remain valid. Inactive_lifetime specifies the amount of time for which the cached value of the inactive ERole will remain valid. In other words, ERoles can be cached regardless of status. In either case, a zero lifetime indicates that the ERole must be evaluated each time it is used. In addition, both lifetime values are static. We are currently investigating other methods that would allow us to derive a more dynamic lifetime value for ERole caching.

With caching enabled, the ERAS is implemented as follows: a request from the Authorization Service will trigger the ERAS to evaluate a list of ERoles. The ERAS will first check whether the ERole requested is already in the cache. If so, it will then determine the “freshness” of the cached copy. If the lifetime is still valid, the cached status is sent to the Authorization Service along with a new expiration time. If a cached copy is not found, or if it is too old, the ERAS will then contact the appropriate sensor(s) for current state values which will be used to (re-)evaluate the ERole. A request timestamp is associated with each (re-)evaluated ERole when they are placed into the cache.

Using a revised implementation of the ERAS, our experiments show that the number of sensors contacted has significant impact on the cost of ERole evaluation. Although this is due, primarily, to the communication overhead, the same experiments demonstrate how an ERole caching mechanism can significantly decrease the amount of time it takes the ERAS
to return the requested role status.

5.2.4 Authentication Service

We have developed an approach that allows for the secure and transparent authentication of users in context-aware environments. By making use of security-specific measurements in our calculations, we are able to take into account the likelihood of a sensor device being compromised, as well as its inherent ability to accurately identify users in the system. Such information is used to determine the level of authentication that can be provided to the source of a request.

Using this approach, we have built a collection of services to support the transparent authentication of users in the Aware Home. The application programming interface (API) for the Authentication Service allows for close interaction with the Authorization Service. When a request arrives at the Authorization Service, it contacts the Authentication Service to provide details on the source of the request. The Authorization Service can specify the quality it requires in the identification of the source. For example, a routine access control check such as one that requests access to the family’s electronic newspaper subscription does not require complete authentication of the source. Having the system identify the user with 100% accuracy is unnecessary and a waste of resources. However, an authorization check that involves a request to access the family’s medical history should involve a more thorough authentication procedure.

The authentication service for CASA supports two models of authentication: push and pull. The push model is similar to the traditional model where a user sends credentials to the system at the time access is requested. The pull model, on the other hand, only requires authentication to be performed when necessary. In the above example, the Authorization Service may determine that any user in the home may access the family newspaper subscription and, therefore, no authentication needs to be performed. However, if the Authorization Service were to receive a credential-less request to access the medical database,
it could dynamically pull authentication information from the environment based on the source of the request.

The CASA authentication service expects requests for either credential verification or credential retrieval. In verification, credentials are passed to the authentication service where their status (validity, expiration, etc.) is checked and a user ID is returned, which can be used to activate subject roles. For credential retrieval, the authentication service is sent information pertaining to the source of the access request and a least-acceptable authentication parameter. The authentication parameter details the level or quality of authentication that is required by the Authorization Service.

Additional details regarding the authentication service and its interaction with the Authorization Service are provided in Chapter 6.

5.2.5 Context Management Services

To facilitate the collection of environment variables and their associated values, we make use of Context Management Services (CMS). The ERAS monitors one or more such services to maintain a snapshot of current environmental conditions. Example CMSs include services based on the Simple Network Management Protocol (SNMP), the Context Toolkit (CTTK) [27] and similar services that monitor environmental conditions (e.g., system, network, etc.) and maintain a record of environment state.

Sensors are placed throughout the environment to collect useful security-relevant data. Sensors can include motion detectors, fingerprint scanners, cameras and numerous other sensing devices. In addition to authentication-related data, sensors can also collect information related to environmental state, such as temperature, ambient noise in a room, network bandwidth and CPU usage. Sensors communicate directly with one or more CMSs that are responsible for managing the received data.

Our current implementation makes use of the Context Toolkit [27, 20] and a collection of distributed sensors for the purpose of managing environmental context. 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. Our implementation of CASA makes several important changes to the Context Toolkit to ensure that security-relevant environmental context is collected in a secure fashion. This section provides details on those changes and the Context Toolkit in general.

In the CTTK, Context widgets represent abstractions over sensors that hide details of how sensing and interpretation of the environment occurs. Widgets are essentially wrappers around an underlying sensor or service; they provide an interface to automatically deliver information to interested components or services in the system. 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). Finally, Interpreters are responsible for abstracting low-level context to higher-level information. An interpreter can convert state information to another format or meaning. For example, a complex interpreter can take location, identity and sound information and subsequently determine that a meeting is underway.

The Context Toolkit was originally 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 secure version of the CTTK that will allow us to collect environment information in a manner that is both secure and reliable. 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. However, the same may not be true for communication links between components.
We have built a certificate-based PKI to support component authentication and data encryption. Specifically, we use HTTP over SSL to allow for authenticated and encrypted sessions. Given the component-based nature of the Context Toolkit, this approach allows us the flexibility to select either secure or insecure channels, depending on the properties of the communication that is taking place. We are currently expanding our implementation to support multiple communication protocols (in addition to HTTP) and a more flexible key-sharing infrastructure (such as SPKI).

In addition to the changes described here, CASA can be used to provide security services to components within the Context Toolkit. In this scenario, components of the Context Toolkit would make access requests to the Authorization Service using the API described in this document.

5.3 Summary

In this chapter we have introduced a new model for securing context-aware environments and describe why we believe it will be useful for securing applications in the highly-connected world of tomorrow. Much of this work is focused on providing more adaptive security services than those found in traditional computing environments.

We have presented a framework for providing authorization services in context-aware environments and applications. This framework supports the collection of contextual information from resources, the environment and the users who interact in that environment. In addition, we have explored, through the context aware security architecture, an implementation of the Generalized Role-Based Access Control model.
CHAPTER 6

SYSTEM IMPLEMENTATION AND EVALUATION

Our context-aware security architecture — through the incorporation of GRBAC and our enhanced authentication techniques — presents a powerful approach for enforcing security policies in a ubiquitous computing environment. This chapter shows how our security infrastructure can be applied, in practice, to the home environment. It also demonstrates some of the additional benefits not offered by traditional approaches to system security. We present an operational scenario that requires an application to leverage the security services of CASA. In addition, we provide a performance analysis of the architecture.

6.1 Application Scenario

To best demonstrate the functionality of our context-aware security architecture, we have used it to build a secure implementation of AudioIM, an instant messaging (IM) application that has been deployed in the Aware Home. AudioIM is one member of a family of applications [68] that are being used to explore how technology can enhance distributed conversation among family and close friends. Similar to text-based IM, which is used in the office to start semi-synchronous talks, AudioIM extends IM into the home with a touchscreen interface and audio messaging. AudioIM strives to provide instant messaging services where desktops are not suitable.

In order to identify users interacting with the system, the original AudioIM application provides a password verification scheme. Pictured in Figure 17, the login component requires the user to authenticate herself with a username and password combination. This approach, though effective, uses a desktop model to achieve user authentication for an application that was designed specifically for pervasive computing environments.
As we prepared our design for an enhanced AudiolM application, we identified two primary goals for our task. First, we wanted to provide infrastructure support that would allow the application to utilize more natural identification technologies. Second, we wanted to provide a rich access control model that would allow flexible access policies to be written by the security administrator.

6.2 Implementation

The following sections describe the various components that comprise our implementation of the context-aware security architecture. We describe each independently and end the chapter by providing a description of the secured AudiolM application.

6.2.1 Policy Definition

A Generalized Policy Definition Language (GPDL) is described in [65] for defining GR-BAC policies. We have found that in practice, however, GPDL can 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 a clear-cut manner.

We have built a prototype graphical editor in order to explore how it can help to define and explain complex security policies. Such an interface is necessary as our access model is deployed in the Aware Home. By using this graphical editor, we are able to display complex security policies using simple constructs and an intuitive layout. This GUI allows a security administrator to associate permissions with various combinations of roles. For example, a child can be denied access to a category of resources that is classified using a single role, dangerous appliance, during certain environmental conditions (e.g., during a parent’s working hours). In a more intricate example, we could specify complex, method-level access control on object resources in the system. For instance, the family physician may have access to read and write medical databases in the home, while the family attorney can only obtain a limited view of such records in the event of an emergency. A screenshot of our policy manager is shown in figure 18.

6.2.2 Runtime Policy Representation

In our implementation, policies are defined through the graphical management tool and encoded into eXtensible Markup Language, or XML [103]. XML is used to specify access policies, role definitions and relationships and is also used as a common representation to share data between the various services in the architecture. Figure 14 shows the XML-encoded policy that restricts a user in the child role from accessing a dangerous appliance during specified environmental conditions.

Given well-structured rule sets, XML provides an efficient structure for storing the policy that is generated and enforced by our security services. We take advantage of XML’s robust, non-proprietary and verifiable file-format by using it to transmit policies and related information (e.g., environment role definitions) between services in our architecture. Each
component in our infrastructure can construct its own efficient runtime structures for local processing, but XML is used to transfer information between services. This allows us to standardize on a platform-independent policy specification that can be read, verified and processed by any authorized component.

We acknowledge that XML encoding and decoding adds a processing cost that is significantly higher than those of other encoding formats due to the conversion of data from binary to ASCII and vice versa. In addition, an XML approach has higher network transmission costs due to records being encoded in an ASCII format. However, XML provides a communication flexibility that is not found in other data encoding formats – rather than transmitting data in a binary form, XML uses an ASCII-based format that provides header and footer information identifying fields in each record. This allows applications to communicate without previous knowledge of one another. In addition, XML is widely-supported among a variety of systems and applications.
The following section provides a working example of GRBAC policy definition, including an XML encoded ruleset, and discusses processing time for GRBAC transactions.

4.2.3 Transaction Processing

Using our graphical Management Tool, we begin by creating a simple environment role hierarchy. In our example, we define a series of basic time-related environment roles that capture different times in an academic calendar. We also specify the CMS/sensors that will be used to collect the required information to determine the ERole status. Finally, we provide subject role and object role information that is subsequently used to form a policy statement. When complete, our role definitions (and the relationships that exist between them) and policy statements are encoded into an XML format similar to that in Fig 19.

Using this access control policy we have run several experiments to show the performance of CASA when processing authorization requests from an application. The four primary components (SMS, ERAS, Authentication and Authorization) were started as separate services on distributed machines. The experiments were conducted on a cluster of workstations using dual-2.20GHz Intel Xeon processors, running RedHat Linux 7.2, all connected by a 100 Mb Ethernet switch. The Java virtual machine was part of the J2SE SDK version 1.4 from Javasoft.

For our first set of experiments, we generated a series of access requests using different sets of active ERoles. There were a total of eleven access requests sent to the Authorization Service. The first request involved a policy that granted access regardless of environmental state; no ERoles needed to be active and, therefore, no check with the ERAS was necessary. All other requests in the series involved access checks that made use of more complex policies; each policy specified an environment role that used from one to ten unique variables (sensors) in its definition. In addition, authentication services were provided to verify credentials that were "pushed" with the access requests.

This first set of experiments was run using HTTP as the transport protocol and allowed
Figure 19: Example XML encoding of Role Definitions and Policy Statements

us to generate an initial set of measurements to demonstrate the efficiency of our implementation. Table 1 shows “round-trip time” for the experiment, starting with the time that the access request is generated and ending with receipt of an access response from the Authorization Service.

To illustrate the performance increase that was observed through the use of caching mechanisms in both the ERAS and the Authorization Service, the same set of experiments was run, first with only an ERAS cache and secondly with a fresh and fully-populated cache at the Authentication Service. As demonstrated here, the CASA implementation can provide access requests in a range of 1 to 40 milliseconds, depending on cache state and contents. Clearly, when the number of ERoles requiring evaluation increases, the time spent evaluating ERole status also increases. This applies to both uncached and cached values. However, we feel that our implementation is highly efficient in determining ERole status—the majority of time spent evaluating ERole status is actually spent in communicating with
Table 1: Generating Access Requests in CASA over HTTP (in milliseconds)

<table>
<thead>
<tr>
<th>ERole Defs</th>
<th>No Cache</th>
<th>ERAS Cache</th>
<th>AS Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>No ERoles</td>
<td>25</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1 Variable</td>
<td>26</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>2 Variables</td>
<td>29</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3 Variables</td>
<td>32</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>4 Variables</td>
<td>30</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>5 Variables</td>
<td>35</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>6 Variables</td>
<td>36</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>7 Variables</td>
<td>39</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>8 Variables</td>
<td>57</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>9 Variables</td>
<td>37</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>10 Variables</td>
<td>38</td>
<td>16</td>
<td>5</td>
</tr>
</tbody>
</table>

the environmental sensors that monitor variable conditions.

In order to ensure that policies are protected and that credentials are not observed during transmission, we also support encrypted communication channels in the CASA implementation. We have performed a second set of experiments in which we generate the same access requests as before but use an SSL-encrypted HTTP channel over which the services communicate. As before, there were a total of eleven access requests sent to the Authorization Service. Table 2 shows the results of our application test.

The results presented here were all obtained using SSL-encrypted HTTP communication channels and, when compared to the previous results, demonstrate the cost of establishing and using secure communications. This protocol selection resulted in a delay that was caused by SSL-related key exchange and session establishment. The overhead introduced by the processing at the services is not significant when compared to the connection establishment and communication times. We are currently working on a communication subsystem that utilizes session key-reuse; we expect this to eliminate a significant portion of the overhead cost associated with the HTTPS-based communication.

The experimental results show that services in our architecture can authorize requests
Table 2: Generating Access Requests in CASA over HTTPS (in milliseconds)

<table>
<thead>
<tr>
<th>ERole Defs</th>
<th>No Cache</th>
<th>ECRAS Cache</th>
<th>AS Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>No ERoles</td>
<td>775</td>
<td>536</td>
<td>212</td>
</tr>
<tr>
<td>1 Variable</td>
<td>885</td>
<td>485</td>
<td>221</td>
</tr>
<tr>
<td>2 Variables</td>
<td>935</td>
<td>490</td>
<td>230</td>
</tr>
<tr>
<td>3 Variables</td>
<td>1181</td>
<td>438</td>
<td>206</td>
</tr>
<tr>
<td>4 Variables</td>
<td>1610</td>
<td>462</td>
<td>215</td>
</tr>
<tr>
<td>5 Variables</td>
<td>1946</td>
<td>438</td>
<td>217</td>
</tr>
<tr>
<td>6 Variables</td>
<td>1737</td>
<td>500</td>
<td>275</td>
</tr>
<tr>
<td>7 Variables</td>
<td>1925</td>
<td>426</td>
<td>243</td>
</tr>
<tr>
<td>8 Variables</td>
<td>2173</td>
<td>436</td>
<td>227</td>
</tr>
<tr>
<td>9 Variables</td>
<td>2417</td>
<td>525</td>
<td>213</td>
</tr>
<tr>
<td>10 Variables</td>
<td>2450</td>
<td>415</td>
<td>193</td>
</tr>
</tbody>
</table>

in a reasonable amount of time. Furthermore, by taking advantage of our caching framework and other runtime efficiencies, such as leaving established (secure) connections open between requests, context-aware security does not present any significant overhead to the application.

6.3 Securing the Application Scenario

Using all of the components that comprise our context-aware security architecture, we have built an enhanced version of the AudioIM application. We began by removing the desktop-influenced authentication support and replaced it with a scheme that utilizes the parameterized authentication model. The enhanced application, pictured in Figure 20, no longer requires the user to log in. Instead, the AudioIM console provides a collection of functions that are available for use. Should one of them be protected by an access control policy, the authentication support would be activated in the background. If there is adequate sensor information available on the current user, no explicit communication is necessary to obtain user credentials.

With authentication no longer required, we moved on to build support for rich access control policies into the enhanced application. The original AudioIM application could
only restrict usage based on username; if an unauthorized user attempted to use the system, he would be rejected upon failing to pass the login prompt. The enhanced AudioIM application, however, allows complex policies to be defined and enforced using the Generalized Role-based Access Control model.

In Figure 21, we illustrate a scenario in which an unknown (non-authenticated) user attempts to access a protected resource. Because the policy for that resource allows access for any user, at any time, no authentication data needs to be retrieved.
Figure 21: Using AudioIM without Authentication

Similarly, Figure 22 illustrates a scenario in which authentication is required. In addition, environment roles must be checked before access can be granted. In this scenario, all of the components of our architecture are utilized to illustrate all of the steps that must be completed before an access request can be granted (or denied).

Figure 22: Using AudioIM with Enhanced Authentication and Access Control

6.4 Summary

This chapter has described the implementation of our context-aware security architecture and our experience working with applications that utilize the security services. Our findings indicate that both the GRBAC and parameterized authentication models are capable of
supporting real-world usage scenarios. Processing overhead for GRBAC has been shown to be minimal and simulations with authentication indicate that it is efficient and scalable.
CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This dissertation has described a collection of flexible security services that are appropriate for pervasive computing environments and context-aware applications. In this final chapter, we summarize our research contributions and briefly describe some areas that merit future research.

7.1 Research Summary

The dynamic nature of pervasive computing motivates the need for a security architecture that can operate effectively in a context-rich environment. In order to maintain the properties that are inherent to ubiquitous computing, the system should provide natural interfaces that utilize context to enhance the quality of the user experience and to reduce the amount of explicit input required by the system.

An examination of existing security solutions, including a variety of authorization models and authentication schemes, has revealed that current approaches cannot provide natural protection for pervasive computing applications. Most notably, current access control solutions cannot meet the demands of a context-aware application because they fail to provide any extensive use of context in authorization. We have also illustrated how current approaches to authentication can be burdensome, intrusive and insufficient for intelligent environments like the Aware Home.

To meet the needs of a context-aware computing environment, we have introduced the Generalized Role-Based Access Control Model and demonstrated how it can express complex access control policies. The benefit of GRBAC over many current models is its combination of simplicity and expressiveness. GRBAC is simple because it is based on only
one central concept: the role. Yet, GRBAC is very expressive because it applies the role concept uniformly to subjects, objects, and environmental states.

We have also presented a model that addresses the inherent imperfections of the authentication process by introducing a metric to capture the overall "quality" of authentication. The major benefit provided by this authentication model is its ability to provide flexible services when faced with incomplete, inaccurate or unreliable security information.

By providing an authentication scheme that is not limited by being binary in nature, users who cannot be fully authenticated by the system may still be given access rights based on their role and the level of confidence the system has in their identification. Our notion of parameterized authentication provides a metric, the authentication parameter, that is based on trust and accuracy values associated with each authentication sensor and the output provided by those sensors.

In addition, we have validated our authentication model by revisiting its design principles and have also explored an enhancement to the GRBAC model that provides extensions to support the authentication parameter.

Finally, we have discussed the design and and implementation of our context-aware security architecture (CASA). We detailed some deficiencies that were initially observed in our framework and discussed our approach to correcting the problems. We have implemented our security architecture and demonstrated its effectiveness with several usage scenarios and applications.

7.2 Thesis Contributions

In summary, the contributions of this thesis are:

- The introduction of a new authorization model, Generalized Role-Based Access Control (GRBAC), that is useful for securing applications in context-aware environments such as the Aware Home.
• The introduction of parameterized authentication, an authentication scheme that is not limited by being binary in nature and through which authentication “quality” is captured, refined and shared. This model introduces a robust component to authentication by allowing continued functionality in a damaged, compromised or fundamentally inaccurate system.

• A security architecture for providing authorization and authentication services to context-aware environments and applications. This framework supports the collection of contextual information from users and resources, and the environment in which they interact.

• An implementation of the GRBAC model and validation of the Parameterized Authentication approach. These experiences led to improvements in the models by identifying strengths and deficiencies in our original approach.

7.3 Future Research Directions

The study of security in pervasive computing environments has a large scope and we obviously have not addressed all of the relevant issues. We will now discuss some issues that warrant further investigation.

7.3.1 Application Deployment and Evaluation

The implementation and evaluation of CASA has relied heavily on a limited set of context-aware applications to test functionality and effectiveness. In addition, we have used software sensors in place of real physical sensors to build and test our security services. Despite recent efforts to provide programming models that support and ease the development of context-aware applications [25], we still lack infrastructure support and a core set of applications to properly demonstrate security issues in pervasive computing. More work needs to be done with real-world applications and physical sensors to further validate our approach. When a variety of these physical sensors become available and once appropriate
applications are built, the architecture requires no change to make use of them. Regardless, practical experience with these applications is desirable, as it will provide more insight into possible enhancements to the model.

7.3.2 GRBAC Transaction Efficiency

Through our experiences with implementing the security architecture, we have learned how to improve the efficiency of a transaction in GRBAC. By identifying the need for multiple caches, we provided a significant increase in response time and are now able to process rather complex policies in a short period of time.

Furthermore, we have learned that significant performance increases can also result from a restructured policy definition. By using postfix notation to represent ERoles, we achieved faster response time and greater flexibility in role definition than with the original CNF format.

The implementation of GRBAC has provided us with a wealth of experience that was unavailable when considering the model alone. However, more work needs to be done to validate and improve this model. A better understanding of the appropriateness and impact of different policy management schemes is also needed.

7.3.3 Enhancing Trust

We have introduced the concept of Parameterized Authentication and detailed an approach to capturing the overall quality of an authentication process. Our model provides a mechanism that evolves trust in sensors through a series of experiences and interactions.

This approach is limited in the sense that we look only at "local" experiences at the sensor level. It would be desirable to consider rule-based systems that can detect malicious activity similar to those found in Intrusion Detection Systems (IDS). For instance, SRI International's System Design Laboratory developed a Next-Generation Intrusion Detection Expert System (NIDES) [3] in which the system performed real-time monitoring of
user activity on multiple target systems connected via Ethernet. NIDES searched for activity that may indicate unusual or malicious user behavior by utilizing two complimentary detection units: a rule-based signature analysis subsystem and a statistical profile-based anomaly-detection subsystem.

NIDES was part of a larger research effort referred to as Computer Misuse and Anomaly Detection (CMAD) in which participants were focused on the detection and correlation of malicious activity [56, 89, 57, 100, 91, 85, 5, 42] from a variety of sources.

We believe that related work like CMAD could be used with Parameterized Authentication to provide an enhanced mechanism for evolving trust. By utilizing a variety of mechanisms to identify malicious activity in the system, we expect that the authentication service could be less reactive and more aggressive in preventing negative experiences from occurring. In its current implementation, the authentication service can only detect malicious activity after it has occurred.

7.4 Conclusions

This dissertation has studied the challenges involved with providing security services to pervasive computing environments and context-aware applications. We have presented the requirements for and an implementation of an architecture that supports the protection of resources in these intelligent environments. Our architecture provides a simple, yet robust access control model that can express complex security policies using roles to capture context related to subjects, resources and the environment in which they interact. We have also presented an authentication scheme that can maintain service when faced with incomplete, inaccurate or unreliable identification information. Our model for parameterized authentication ties in to the authorization framework by authenticating users into roles when insufficient data is available for perfect authentication. We have studied our security models and have provided implementation experience to demonstrate the effectiveness of our approach.
REFERENCES


121


123


[58] Leyden, J., "Biometric sensors beaten senseless in tests," The Register, May 2002.


124


125


Michael J. Covington was born on March 1, 1976, in Annapolis, Maryland. In 1997, Michael received his Bachelor of Science degree in Computer Science and graduated with honors from Mount Saint Mary’s College in Emmitsburg, Maryland. Subsequently, he was awarded his Masters of Science in Computer Science degree from the Georgia Institute of Technology in 1999. He was a 1999-2000 Intel Foundation Graduate Fellow.

Mr. Covington’s research efforts focusing on building secure, scalable, and dependable systems for pervasive computing environments led to his doctoral dissertation entitled: “A Flexible Security Architecture for Pervasive Computing Environments”. Mr. Covington received his Ph.D. from the College of Computing at the Georgia Institute of Technology in 2004.