AN INTELLIGENT AUTHENTICATION INFRASTRUCTURE FOR UBIQUITOUS COMPUTING ENVIRONMENTS

BY

JALAL F. AL-MUHTADI

B.C.I.S., King Saud University, 1996
M.S., University of Illinois at Urbana-Champaign, 2001

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

To my parents, who inspired me and sparked my interest to pursue higher education and who provided me with support, help and encouragement every moment along the long academic road that I followed.

Acknowledgements

This work was not possible without the help and support of many distinguished individuals. I am greatly indebted to my thesis advisors, Professor M. Dennis Mickunas, and Professor Roy Campbell, for their ongoing support, positive feedback, valuable ideas, and for putting me back on track whenever I lose my sense of direction. This dissertation has greatly benefited from their insights, valuable suggestions, and constructive comments. I would like to acknowledge the other distinguished members of my committee Professor Klara Nahrstedt and Professor Yuanyuan Zhou for their important advice, encouragement, and comments. I am thankful to many of my colleagues in the Software Research Group and in the Gaia project for all the advice and suggestions they provided me and for the valuable interactions, discussions and brainstorming sessions that I had with many of them at various points along the line, including Manuel Roman, Prasad Naldurg, Apu Kapadia, Seung Yi, Dulcineia Carvalho, Chris Hess, Chetan Shiva, Anand Ranganathan, Jim Bresler, Zahid Anwar, Jeff Naisbitt, Raquel Hill and many others. Thank you all!

Table of Contents

List of Figures

List of Tables

Introduction

These are exciting times! We are witnessing the birth of a revolutionary computing paradigm that promises to have a profound effect on the way we interact with computers, devices, physical spaces and other people. This new technology envisions a world where processing power and digital communication are extremely inexpensive commodities that are widely available. This approach eliminates time and place barriers by making services available to users anytime and anywhere. In the recent past, personal computers transformed computing into a one-to-one correspondence between people and computers, where each person had his or her own personal computer. The natural sequel to this is a computing environment that advocates a one-user to many machines computing model. In such a scenario, users are surrounded with a comfortable and convenient information environment that merges physical and computational infrastructures into an integrated habitat. This habitat features a plethora of computing devices and sensors that provide new functionality, offer specialized services, boost productivity, assist in carrying out tasks and facilitate seamless interaction with available resources. Anytime and anywhere computing support for human activities offers novel opportunities to enhance human abilities and experience in business, science, education, medicine, day to day living, and entertainment. Experiments and projects have shown that enriching physical environments with a plethora of heterogeneous devices, sensors, and networks, managed by context-aware middleware platforms results in enormous benefits and automation possibilities. We refer to this new computing paradigm as *ubiquitous computing* and we refer to the enriched physical spaces as *Active Spaces*.

More formally, we define an Active Space as a physical space coordinated by a responsive context-based infrastructure that enhances the ability of mobile users to interact and configure their physical and digital environment seamlessly.

The Gaia project [1-3] at the Department of Computer Science at the University of Illinois at Urbana-Champaign, is a large research endeavor that, as of this writing, is pioneering the notion of context-aware, programmable environments for mobile users. The Gaia project integrates ubiquitous computing devices and off-the-shelf software and hardware including applications, file systems, operating systems, PDAs, smartphones, webcams, and plasma displays, into a rich tapestry that cooperates and coordinates its activities with its mobile users. Gaia facilitates this by integrating the software and devices through middleware into a distributed, heterogeneous, computing system. Mobile users running ubiquitous computing applications on Gaia may migrate with their applications and data from one location to another or they can share their applications with mobile users at remote locations. Furthermore, dynamic resource allocation and service composition for quality of service guarantees are integrated into a flow management framework for multimedia applications. Context-awareness will allow an Active Space to take on the responsibility of serving users, by tailoring itself to their preferences as well as performing tasks and group activities according to the nature of the physical space and the tasks assigned to the users.

The realization of this computing paradigm in the real-world is not far fetched. An average person today already interacts with vast numbers of consumer devices and electronic gadgets that already have processors, microcontrollers, and memory chips embedded into them, like VCRs, TVs, washers and dryers. The vehicles we use on daily basis already have a large number of embedded computers handling different subsystems of the vehicle, like ABS (Anti-lock Braking System), ESP (Electronic Stability Program), OBD (On-Board Diagnostic system) and navigation systems. Technologies like Bluetooth [4], Wi-Fi [5], and UWB [6] make it possible to embed networking capabilities into any small device seamlessly. In effect, these technologies make networking much more general and achievable even on elementary devices, like toasters, light bulbs, and clothing articles. Soon, the human endeavors of creativity, engineering, learning and collaboration in various areas can become a crucible for new perspectives on how ubiquitous computing can impact society and serve users.

1.1 Ubiquitous Computing Vision

The term "ubiquitous computing" is a very broad term that is often overloaded to mean diverse things to different research projects. In many cases, researchers define ubiquitous computing by example, with respect to their own research. Therefore, it is important to define exactly what my vision of ubiquitous computing is. I adapt Weiser's visions of ubiquitous computing [7, 8]. More precisely, ubiquitous computing refers to a proliferation of hundreds or thousands of computing devices, sensors and embedded processors that will provide new functionality, offer specialized services, boost productivity, and facilitate seamless interaction with the surrounding environment and available resources. Ubiquitous computing allows us to realize additional abstractions that did not exist in traditional computing paradigms. The salient features of ubiquitous computing include the following.

Extending Computing Boundaries. While traditional computing encompassed hardware and software entities, ubiquitous computing extends the boundaries of computing to include physical spaces, building infrastructures, and the devices contained within. This aims to transform dull, passive spaces into interactive, dynamic, and programmable spaces that are coordinated through a software infrastructure and populated with a large number of mobile users and devices.

Invisibility and non-intrusiveness. In current computing models, computers are still the main focus of attention. In effect, people have to change some of their behavior and the way they perform tasks so that these tasks can be computerized. To boost productivity, it is important that computing machinery disappears from the spotlight. Computers should blend in the background allowing people to perform their duties without having machines at the center of their focus.

Creating smart and sentient spaces. A dust of invisible embedded devices and sensors are incorporated to turn physical spaces into active, smart surroundings that can sense, "see," and "hear," effectively, making the space sentient and personalized. Ultimately, the space should become intelligent enough to understand users' intentions and become an integral part of users' everyday life.

Context awareness. A ubiquitous computing model should be able to capture the different contexts and situational information and integrate them with users and devices. This allows the Active Space to take on the responsibility of locating and serving users and automatically tailoring itself to meet their expectations and preferences.

Mobility and adaptability. To be truly omnipresent, the ubiquitous computing environment should be as mobile as its users. It should be able to adapt itself to environments with scarce resources, while being able to evolve and extend once more resources become available.

While ubiquitous computing could have a profound impact on the way humans interact with computers and devices, unfortunately, security and privacy will be ubiquitous computing's Achilles heel and will present a significant barrier to real-world deployment of the technology.

1.2 The Need for Novel Security Mechanisms

Current research in ubiquitous computing focuses on building infrastructures for managing smart spaces, connecting new devices, or building useful applications to improve functionality. Information assurance, security, and privacy related issues in such environments, however, have not been explored in depth. Nevertheless, addressing information assurance and privacy issues in ubiquitous computing is vital to the real-world deployment of the technology. Ubiquitous computing environments raise complex security and privacy issues, which require novel security mechanisms that are able to deal with the ubiquity, the context awareness, and the rapid evolvement of such environments. In fact, the very same features that make ubiquitous computing environments convenient and powerful make them vulnerable to new security and privacy threats.

Traditional security mechanisms require much user interaction in the form of manual logins, logouts, and file permissions. These manual interactions violate the disappearing computer vision and imperil its ubiquitousness. The security requirements of an Active Space may vary according to the context of the space. Some situations (like during a confidential meeting or homeland security alerts) require greater security to be in place; while other situations may not require a very high level of security. Traditional security mechanisms are context-insensitive, i.e. they do not adapt their security policies to a changing context.

Furthermore, traditional distributed systems rely on varying degrees of technical competence on the part of the user of the system to implement and enforce security policies and mechanisms. I argue that this assumption is becoming increasing untenable. As an example, the widespread use of PDAs and cellular phones, along with their value-added services, has already challenged these assumptions severely. An average cell-phone user may not have any knowledge that his or her voice-data is encrypted to prevent other users from listening in or modifying her calls. However, the ability of the infrastructure to provide these guarantees at all times is the key to its usability. In a sense, the security mechanisms become invisible, but increasingly crucial, to the correct functioning of the technology. Furthermore, the closer interaction between the real world and the virtual world presents its own set of challenges.

I argue that ubiquitous computing forces us to think about and evaluate security technology using a radically different approach. Issues such as dependability of the components and the infrastructure, as well as the impact of failures on security protocols, now become magnified and increasingly crucial to the survivability of the system. For the reasons stated above, new revolutionary security mechanisms need to be devised. These mechanisms need to be context-aware, ubiquitous, and non-distracting. Some of the main challenges in providing information assurance in ubiquitous computing environments are outlined in the subsections below.

1.3 Security Challenges for Ubiquitous Computing

Mark Weiser envisions ubiquitous computing as a technology that will make using computers as seamless and refreshing as a "walk in the woods" [9]. However, making this journey "safe" is not straightforward. Furthermore, researchers and practitioners in this area have admitted that information assurance and privacy in ubiquitous computing are real problems. Langheinrich [10, 11] warns us about the possibility of an Orwellian nightmare in which current ubiquitous computing research continues on without considering privacy in the system. Stajano [12] notes that while researchers are busy thinking about the killer applications for ubiquitous computing, cyber-criminals and computer villains are already considering new, ingenious attacks that are not possible in traditional computing environments. Kumar [13] considers security and privacy to be among the biggest challenges to Weiser's vision of computing environments. Kagal et al. [14, 15] admit that securing pervasive computing environments presents challenges at many levels. Zakiuddin et al. [16] argue that pervasive computing environments require a revolutionized authentication mechanism, which is not only concerned with the "name attribute" of an entity, but it takes into account other attributes, like location, type, and trust level.

I argue that mere adaptation of existing information assurance, security and privacy mechanisms is not enough for securing ubiquitous computing, mainly because of the new challenges that it poses. In this section I illustrate the difficulties that make securing ubiquitous computing environments more challenging.

1.3.1 The Extended Computing Boundary

Traditional computing is confined to the virtual computing world where data and programs reside. Current distributed computing research tends to abstract away physical locations of users and resources. Ubiquitous computing, however, extends its reach beyond the computational infrastructure and attempts to encompass the surrounding physical spaces as well. Ubiquitous computing applications often exploit physical location and other context information about users and resources to enhance the user experience. Under such scenarios, information and physical security become interdependent. As a result, such environments become prone to more severe security threats that can threaten people and equipment in the physical world as much as they can threaten their data and programs in the virtual world. Therefore, traditional mechanisms that focus merely on digital security become inadequate.

1.3.2 Privacy Issues

The physical outreach of ubiquitous computing makes preserving users' privacy a more difficult task. Augmenting Active Spaces with active sensors and actuators enables the construction of more intelligent spaces and computing capabilities that are truly omnipresent. Through various sensors and embedded devices, Active Spaces can automatically be tailored to users' preferences and can capture and utilize context information fully. Unfortunately, this very feature could threaten the privacy of users severely. For instance, this capability can be exploited by intruders, malicious insiders, or even curious system administrators to track or stalk particular users electronically. As a result, the entire system becomes a distributed surveillance system that can capture too much information about users. In some environments, like homes and clinics, there is usually an abundance of sensitive and personal information that must be secured. Moreover, there are certain situations when people do not want to be tracked.

1.3.3 User Interaction Issues

One of the main characteristics of ubiquitous applications is a rich, straightforward, unobtrusive user-interface for interactions between users and the surrounding spaces, as well as interactions among the different users. A variety of multimedia mechanisms are used for input and output, and to control the physical aspects of the space. The set of users in the space affects the security properties of the space. Because of the nature of group interactions between users, users in the space cannot easily be prevented from seeing and hearing things happening in it, so this has to be taken into account while designing some security mechanisms. Thus the physical and "virtual" aspects of access control for such spaces have to be considered together.

1.3.4 Security Policies

It is important in ubiquitous computing to have a flexible and convenient method for defining and managing security policies in a dynamic, context-aware fashion, this is because the security rules of an Active Space may vary according to the context of the space. Some situations (like during a confidential meeting or homeland security alerts) require greater security to be in place; while other situations may not require a very high level of security. Traditional security mechanisms are context-insensitive, i.e. they do not adapt their security policies to a changing context. As a result, the security subsystem has to support a security policy language that is descriptive, well-defined, and flexible. The language should be able to incorporate rich context information as well as physical security awareness.

Traditional Policy Management tools that are based on scripting applications [17-19] that iterate through lists of low-level interfaces and change values of entity-specific system variables are inadequate within environments populated with many heterogeneous devices, under ever-changing situational information. Furthermore, when considering privacy issues, more challenges arise. For example, the disclosure of security policies may be a breach of security. Knowing whether the system is on the lookout for an intruder, say, could actually be a secret. Thus, unauthorized personnel should not be able to know what the security policy might become under a certain circumstance.

1.3.5 Info Ops

There is a great deal of concern over new types of threats, namely, Information Operations (info ops) and cyber-terrorism, which are natural consequences of the increasing importance of electronic information and the heavy reliance on digital communication networks in most civilian and military activities. Info ops, which can be defined as "actions taken that affect adversary information and information systems while defending one's own information and information systems," [20] is a serious concern in today's networks. In such a scenario, cyber-terrorists and other techno-villains can exploit computer networks, inject misleading information, steal electronic assets, or disrupt critical services. Ubiquitous computing provides additional leverage and adds many more capabilities to the arsenal of "info warriors," making info ops a much more severe threat.

1.3.6 An Integral Part of the Whole System

For the reasons stated above, security and privacy guarantees in ubiquitous computing environments should be considered as an integral part of the whole system. They should be specified and drafted early into the design process rather than being considered as add-ons or afterthoughts. Previous efforts in retrofitting security and anonymity into existing systems have proved to be inefficient and ineffective. The Internet and Wi-Fi are two such examples both of which still suffer from inadequate security.

1.4 Contribution

While ubiquitous computing promises to be the next revolutionary technology in computing and although security is essential for general acceptance and wide deployment of the technology, yet many pioneering research efforts in this field have either ignored security altogether, or listed security as future work [21, 22]. Some projects presented some security solutions that are customized for their particular scenario making it difficult, if not impossible, to port to other scenarios [23]. The scarce supply of research efforts that tackle security problems in ubiquitous computing in novel ways has motivated this work. In this work, I present the design, implementation and evaluation of a comprehensive autonomous framework for addressing several cornerstone security issues in ubiquitous computing. In particular, I focus on issues pertaining to authentication of entities, preserving privacy, providing calm interfaces to security services, and enriching security with context awareness. The framework assimilate identification and authentication data, in real-time, from different sensors and authentication technologies to get a more complete picture of the physical environment, its contents, and their permitted interactions.

1.5 Security Terminology in the Context of Ubiquitous Computing

To assess the security and privacy needs of ubiquitous computing, it is important to provide a more accurate definition of security terms in the context of ubiquitous computing. Here, I adopt and extend the terminology of [12, 24] to encompass security issues that make sense in ubiquitous computing environments.

1.5.1 Security Terms

A *security attack* is an action that compromises the security of information. A *security vulnerability* is a weakness in the system's defenses. Attacks are prevented through the use of *safeguards. Recovery* is the process of bringing the system back to normal operation after an attack. *Countermeasures* are remedies taken to counter an attack. A *security mechanism* is a set of protocols that are designed to detect, prevent, or recover from security attacks. *Security services* are resources in a system, which provide protocols and security mechanisms for enhancing the system's security and satisfying some security policies. Some of these services function as a safeguard against attacks, while others serve as countermeasures. *Security policies* are set of rules that guide the implementation of security in a system to match the requirements of the system. Security policies can be defined at different levels of abstraction. In an Active Space, the security policies should be flexible to capture situational changes in the surrounding environment.

1.5.2 Security Services

In this section, I mention the security services that are relevant to this research. I define these security services in the context of ubiquitous computing.

Identification is the process of linking an *entity* with an identity. This process can be initiated by the entity itself (a user typing her user ID) or inferred by the system through sensors and detection. Entities are people, programs, devices, sensors, or even physical spaces. *Authentication* provides assurance for the claimed or detected identity of an entity in the system, i.e. it verifies whether the identification of this entity is correct. I use *principal* to refer to the entity whose identity has been established. *Data authentication* provides evidence that a piece of data has originated from a particular principal. *Location authentication* provides an assurance for the claimed location of a principal. A more general form of location authentication is *context authentication* which provides an assurance for the claimed context that a particular principal operates under. Context is defined in literature as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications" [25].

Authorization or *access control* establishes the set of operations a principal is allowed to do. It should be noted that in ubiquitous computing environments these operations extend to physical spaces, so whether a principal is permitted to enter a physical room is also part of the access control policy. *Context-based* access control is an authorization policy that is dependent on the current situational information and temporal circumstances. *Confidentiality* is a security service which ensures that data is not disclosed to unauthorized principals. Confidentiality applies to stored data as well as data in transient from one principal to another over the network. *Integrity* protects against tampering to data or messages. *Nonrepudiation* prevents a principal involved in communication from later falsely denying participation in the communication. *Availability* ensures that all components of a ubiquitous computing environment, whether physical or virtual, are available to authorized principals without any disruptions or impediment. *Credentials* are special data structures that serve as evidence for the identity or authority of a principal.

Several security services are concerned with the privacy of principals in the system. These services include *anonymity* which ensures that a principal may use a resource or service without disclosing its identity *or physical location* to a set of principals (the *anonymity set*). Note that physical location is particularly important in computing systems that extend to physical spaces. *Identity anonymity* and *location anonymity* are used when I am only concerned about hiding identity or location respectively from a set of other principals. *Sender anonymity* is usually defined differently in the literature. Here I define sender anonymity as hiding the identity *and* location of a principal who is sending messages from a set of other principals. *Receiver anonymity* is the counterpart, i.e. hiding the location and identity of the intended recipient of a particular packet or message. *Pseudonymity* implies assigning pseudonyms to principals that hide their identity while still holding them accountable for their actions. *Unlinkability* ensures that a principal may make multiple accesses to resources or services without others being able to link these uses together. Similarly, unlinkability can mean that a certain relationship between two or more principals is hidden from a set of other principals. For example, communication unlinkability between *A* and *B* means that others do not know that *A* is actually communicating with *B*. *Unobservability* adds an additional privacy constraint in which a principal may use a resource or service without a set of other principals being able to observe that the resource or service is being used.

Info ops refer to actions taken that affect adversary information and information systems while defending one's own information and information systems. The term 'system' here includes communications and infrastructure [20].

2. Related Work

In this section, I present a summary of existing research and related work to my dissertation. I show how my dissertation either complements that work or provides additional contribution. The work that is relevant to my research can be categorized into three major classifications. First, I consider recent research efforts that focus on some aspects of ubiquitous computing security. Second, I consider approaches for preserving privacy on internetworked systems. Finally, I present some work on automated reasoning and some of its applications in computer security.

2.1 Security for Ubiquitous Computing

Designing and implementing a security infrastructure for ubiquitous computing environments require developing different security mechanisms that can secure the different aspects of a ubiquitous computing environment. These include omnipresent authentication for principals, protocols for securing wired and wireless links, access control models suitable for ubiquitous environments, and mechanisms for securing resource-stripped devices. I outline some research projects that tackled some of these problems.

Many approaches to securing distributed systems in general relay on the Role-based Access Control system [26] (RBAC). RBAC is based on the principle that access control decisions are based on the roles individuals take on as part of an organization. The key concept in RBAC is a role, which is a placeholder for a set of users. Each role is associated with a set of permissions, which are its rights on objects. These roles may be organized into a hierarchy to reflect the organizational hierarchy among different users or entities in a system. RBAC maintains two mappings: the User Role Assignment (URA) and the Role Permission Assignment (RPA). These two mappings can be updated independently. Users can be added to the URA without changing the RPA, automatically providing new users a predefined "role" in the system. Similarly, the RPA associates all users in the system to a limited set of permissible behaviors, and can be updated independent of the URA. The key insight in RBAC is that the URA and PRA change less frequently than the permissions of individual users. RBAC is also flexible enough to model both MAC [27] and DAC systems [28].

RBAC has been adapted for use in ubiquitous computing environments [29-31], and the concept of roles is extended to deal with context information. While networked applications have traditionally attempted to hide physical location, by providing uniform interfaces for local and remote users to access services [32], in ubiquitous computing environments, spatial location is often important to the organization of communication [33]. The Aware Home research project has extended RBAC with object and environment roles [31, 34, 35] that they use to define context-aware security policies such as those based on temporal authorizations. The Aware Home project views a home as a unique challenge, because it has abundant sensitive and personal information. Furthermore, the homeowner or operator cannot be assumed to have extensive technical knowledge. The research extends the RBAC model to develop a non-obtrusive access control system that can make use of environmental and context information. The system is meant to be usable and easy to manage for homeowners and to act as a safeguard against remote attacks or break-ins. They refer to this model as the Generalized Role-Based Access Control model (GRBAC). The authors view environmental circumstances and context information as two important factors that govern the access control policies in smart environments. Since traditional RBAC is subject-centric, it fails to capture context and environmental variables. GRBAC adds to RBAC two new concepts; environmental roles and object roles. The environmental roles are meant to capture context information, constraints, etc.

To gather such information, they employ the Context Toolkit [36]. Context widgets gather context information from sensors, and provide an interface for applications. The captured context information can then be aggregated based on the particular data that is sought and the entity in question. The model activates environmental roles and associates them with subjects and objects using a Prolog-style language. Covington et al. [37] extends the model above by illustrating how it can be used to secure rich, context-aware applications. Drawbacks of this work include the lack of implementation experience and the concentration on examples that are too simplistic. While the authors view their work as a security architecture, it is an authorization-centric scheme that does not address other security services.

Kagal et al. [14] suggests another approach to secure pervasive computing environments. The authors argue that large, open systems do not scale well with centralized security solutions. They instead, propose a security solution (Centaurus) based on trust management, which involves developing a security policy and assigning credentials to entities. Centaurus depends heavily on the delegation of trust to third parties. In this environment, users can access nearby smart devices via their handhelds which are connected using Bluetooth. The solution extends SPKI and RBAC. Every domain in the distributed system contains security agents that are responsible for authenticating and verifying entities within their domain. The agents are arranged hierarchally and use X.509 certificates for identification. The security policies are based on the roles assigned to the user accessing the service. These roles can be delegated and revoked by authorized users. This notion of delegation makes the authorization service effective, dynamic and manageable. Nevertheless, the system depends heavily on delegation of trust, which may be difficult to have in a ubiquitous environment rife with mobile users and devices that were never seen before.

Task-based Authorization Controls [38] model access control requirements from the task's point of view, different permissions are allocated to different users at different stages in the task. Georgiadis et al. [39] used a team-based access control scheme to support fine-grained policies using RBAC. Teams of users can be assigned permissions for a particular task, and some contextual information such as time and location can be considered by the access control system. Shen et al [40] provide a generalized editing model for collaborative access control, whereby users interact with a collaborative application by concurrently editing its data structures.

Several research projects concentrate on enabling ubiquitous computing. These projects address security and authentication at different levels. CoolTown [23] at Hewlett-Packard laboratories implements a ubiquitous computing framework using web technologies. Places and objects emit unique URLs wirelessly through beacons from which users access information through PDA devices. CoolTown utilizes traditional encryption techniques to protect sensitive data and digital signatures to prevent the transmission of rogue signals from unauthorized beacons. CoolTown does not address the more sophisticated security threats that come with ubiquitous computing. Aura [41, 42] focuses mainly on mobile users, associating applications with them and migrating these applications between environments while attempting to maximize resource usage and minimize user distraction. Basic levels of security are achieved by employing traditional encryption and authentication mechanisms. In contrast, the proposed framework emphasizes the construction of novel authentication and context-aware security services that address the unique demands of ubiquitous computing environments and the interaction between spaces, users, devices, and objects.

Other ubiquitous computing projects focus on specific applications or situations. The Microsoft Easy Living project [22] focuses on home and work environments, using computer vision to recognize gestures and users and to detect user location so as to customize the room and position interfaces. Classroom 2000 [43] provides an automated classroom environment and tools to capture interactions and information exchanges. The i-Land [44] and Roomware [45] projects digitally augment meeting rooms to facilitate exchanging ideas, digitally recording meeting results, searching knowledge bases, and collaborating through multimedia data exchanges. Whereas each of these projects' focus prevents it from addressing some broader issue of ubiquitous computing environments and their security, the proposed framework is a more general solution supporting any number of distributed components and covering simple, complex, and heterogeneous spaces in a single framework.

The iROS project [46, 47] from Stanford introduces a meta-operating system for ubiquitous computing environments called "Interactive Workspaces." iROS allows independent software components and devices to communicate by broadcasting events through a centralized substrate called the "Event Heap." In effect, the Event Heap is a central message repository. All components in the workspace communicate by posting and retrieving events to and from the Event Heap. Recipients of an event are determined by the contents of the event itself. In essence, the Event Heap decouples applications "referentially" and "temporally" in order to enable synchronization and interoperability between different components. To address security issues in iROS, Song et al. [48] introduce the iSecurity framework. iSecurity addresses authentication and access control issues within Interactive Workspaces. iSecurity handles authentication via a centralized mechanism referred to as the iSign server. Users must authenticate through the iSign server by conventional challenge/response mechanisms. Upon success, the user acquires a X.509 certificate, which is used for posting and retrieving events from the Event Heap. Sensitive events can be transmitted securely over SSL tunnels. iSecurity on the other hand provides for decentralized security policy enforcement where each device maintains and enforces its own security policies. While this is useful in some scenarios, however, in many cases a space-wide policy is needed. Unfortunately, iSecurity is designed to work specifically in environments that rely on a centralized message repository (the Event Heap), and therefore, is intended primarily for iROS-based applications and devices. Furthermore, the reliance on a purely centralized mechanism for communication and authentication could introduce reliability problems and security vulnerabilities in case the Event Heap is compromised. Finally, iSecurity focuses on low-level implementation details, which in the most part, rely heavily on the Event Heap programming model, and thus, is difficult to apply in other application domains. The framework that I propose attempts to provide authentication and access control mechanisms that (1) exploit the distributed and component-based nature of Gaia, reducing single points of failures and (2) emphasize usable, context-aware security by accommodating automated reasoning techniques.

Yi et al. [49] introduce security mechanisms for addressing cryptographic key management and security-aware routing in ad hoc networking environments consisting of heterogeneous nodes with limited resources. Similar to this work, Yi et al. argue for a best effort security service, in which security is not viewed merely as a binary concept of either secure or insecure. Instead, it is viewed as a continuous spectrum of values between 0 and 100%. The work introduces several probabilistic tools to determine a value for security depending on environmental situations (context) and amount of trustworthiness of nodes and communication links. This value of security is referred to as “quality of security.” How this value is interpreted is left for the applications running in the ad hoc environment. The work assumes the existence of harsh environmental restrictions, including the absence of any infrastructure, the absence of a central CA known in advance, and the possible existence of untrusted nodes and communication links. My work differs from the aforementioned work in that it addresses a different application domain, namely ubiquitous computing as opposed to an ad hoc network environment. In a ubiquitous computing environment an infrastructure exists, additionally, some assumptions can be made regarding trusted components, which help simplify some of the protocols. Furthermore, the aforementioned work does not address usability and unobtrusiveness, which are irrelevant in ad hoc networking environments but very relevant in ubiquitous computing.

In an earlier work [50] I argued in favor of extending existing security mechanisms. This included the redesigning of a well-established security mechanism such that it becomes component-based, portable, and capable of loading and unloading modules and cryptographic profiles on the fly. These extensions made it possible to provide an initial security framework for ubiquitous computing environments that are rife with a large number of devices which differ greatly in their capabilities, processing powers and security needs. SESAME [51, 52] is a security mechanism that extends Kerberos and adds support for public key technologies, role-based access control, and delegation of access rights. I introduced Tiny SESAME [50, 53], a lightweight, Java-based subset of SESAME. Tiny SESAME incorporated a component-based design for the client and service sides. In this design, the different security services, cryptographic profiles and protocols were implemented as separate components that can be loaded, unloaded, or configured on demand. To test the practicality of the system, I managed to simulate a smart home environment that is built using Jini version 1.0. The smart environment consisted of a large number of various consumer devices like TVs, VCRs, and toasters that are internetworked. I demonstrated the flexibility of Tiny SESAME further by porting it to resource-stripped PDA devices [53]. Tiny SESAME, however, had some limitations. First, Tiny SESAME may not scale well for large environments with hundreds or thousands of embedded devices and sensors. Some aspects of the authentication process were not very transparent and often required explicit interactions with the user, like entering a password. When adding new users and devices, some additional administration overhead was needed in terms of setting up users and devices with the Security Server, and installing Tiny SESAME's client side. Furthermore, Tiny SESAME is context-unaware and requires Java support on clients.

Biometric authentication techniques sparked an enormous interest lately. Biometrics show good potential for providing seamless and automated methods for determining and confirming identity while being less obtrusive. Good fingerprint recognition or face recognition techniques, for example, are faster than entering secure passwords and do not require users to carry special equipment (e.g. PDAs or badges). However, biometric authentication is plagued with several shortcomings. As described by Vielhauer et al. [54] and others, many biometric authentication techniques have overt characteristics, i.e., the authentication data is often observable to everyone (handwriting signatures can be observed and forged, and fingerprints can be extracted relatively easily). Accuracy and seamlessness of biometric authentication techniques are very dependent on hardware. Finally, biometric authentication techniques still lack a good and secure method of storing the biometric features in a way that prevents compromise of sensitive data and preserves anonymity while providing enough flexibility to accommodate partial matches and deduce a suitable confidence level. Vielhauer et al. [54] provide a generic system design for managing biometric authentication techniques in a manner that mitigates these problems. My proposed framework provide enough flexibility to accommodate biometrics authentication techniques that result in partial matches or "confidence matches."

CORBA Security specification [55] provides the model and the architecture for security in CORBA middleware systems. Its main concern is to ensure the traditional security goals that include confidentiality, integrity, accountability, availability, and non-repudiation. The specification lays down the security facilities and interfaces to application developers, security administrators, and implementers of CORBA systems. Common Secure Interoperability specification addresses the issues of secure interoperability between different CORBA implementations. The intent of these security specifications is to provide primitive security services at the middleware layer. Securing ubiquitous computing environments require leveraging these technologies with more abstract services.

2.2 Privacy Protocols

Privacy is a major concern in ubiquitous computing environments. Unfortunately, at the time of this writing, minimal efforts are exerted to address privacy issues in ubiquitous computing environments. Many efforts, however, tried to focus on building privacy mechanisms for the Internet. Some leading privacy efforts are discussed in this section.

Langheinrich et al. [10] warn us about the possibility of an Orwellian nightmare in which current ubiquitous computing research continues on without considering privacy issues. The authors proceed to describe the design principles of privacy-aware ubiquitous systems. Some of the principles proposed are yet to be implemented. Furthermore, they do not appear to be implementable with current technology but the work gives a good general guideline for privacy issues in ubiquitous computing systems. The authors mainly focus on building a privacy awareness system that is based on six principles for preserving privacy. These principles – notice, choice and consent, proximity and locality, anonymity, security and access and recourse – are based on a set of fair information practices common in most privacy legislation in use today.

In follow up work Langheinrich et al. [11] focus on implementing some aspects of a privacy awareness system by specifying privacy policies and trying to ensure that data collectors honor these policies. The focus mainly is on data usage policies, with an additional emphasis on providing guidelines for users to keep track on how their sensitive data is used in the system through accountability. Nevertheless, a practical implementation of these guidelines is yet to be demonstrated. The proposed system assumes a great deal of trustworthiness in data collectors and sensors. Furthermore, the approach does not address circumstances when rogue sensors are implanted or when data collectors lie about their information gathering practices. The latter two concerns are noteworthy particularly in an environment where sensors are meant to be invisible and non-intrusive.

Lederer et al. [56] explore the notion of "everyday privacy" in terms of consent and notice and motivate the need for modeling user preferences for privacy. However, this is still work on progress. Jiang et al. [57] introduce the notion of Information Spaces, which provides a way to organize information services and resources around important privacy relevant contextual factors. They are concerned with providing access control to information by assigning privacy tags to pieces of information. Each tag represents the access policy for the information piece it is attached to. The work allows the tag of information derived from other pieces of information to be generated automatically based on the tags of the other pieces of information. The work provides a good approach to controlling access to information; however, it appears to be focusing toward analyzing and modeling information flows between different endpoints in a ubiquitous computing environment rather than mitigating privacy risks that can occur from using value-added service, which I am trying to address in my dissertation.

Trust negotiation [14, 58-65] addresses the problem of users over a network establishing a trusted channel without revealing unnecessary information about themselves to the other party. Each party in this transaction can have privacy policies that restrict which of their credentials they are willing to reveal for this purpose. The notion of privacy in these systems is restricted to the exchange of credentials.

Sherwood et al. [66] present the P5 protocol, which utilizes peer-to-peer technology to provide anonymous communication over the Internet. The aim is to provide sender- receiver- and sender/receiver-anonymity. The protocol is customizable, allowing the administrator to strike an acceptable balance between communication latency, network bandwidth, and anonymity. The P5 protocol is meant to thwart adversaries that are capable of monitoring all or a subset of the network links. The brute force version of P5 assumes the existence of a broadcast channel over which all communication endpoints send fixed-length packets at a constant rate. These packets could be just noise packets meant to prevent traffic analysis, or they could be data packets intended for a particular recipient. In the latter case, the packets are encrypted using the target's public key. Since this naïve solution leads to a significant loss in bandwidth, the paper introduces the idea of having a logical broadcast tree, where each node represents a broadcast group (or a broadcast channel). Two values, a bit-string and a mask, are used to represent the contents of a group, and define how messages are sent. Users join one or more groups. Communication efficiency increases as the mask size increases, whereas anonymity increases as the mask size decreases. While the P5 protocol can avert traffic analysis attacks, it does not scale-well to massively distributed environments, because of its reliance on broadcast trees and its use of topology servers. The protocol appears to suffer serious performance degradation in situations where users are added or removed frequently. Since ubiquitous computing environments may include hundreds or thousands of participating nodes, some with limited bandwidth and stripped resources, the P5 protocol is unsuitable for ubiquitous applications.

Anonymizer [67] and SafeWeb [68] are two user anonymity solutions provided to World Wide Web users. Anonymizer is a centralized, 3rd party approach to hide the web users' real identities from the web servers they access. Users can enjoy anonymity by rerouting their HTTP packets through the Anonymizer, which replaces the information in the packet headers so that target websites cannot infer the users' identities or IP addresses. This approach has the problem of a centralized trusted entity. The Anonymizer site can track all the anonymous user activities and is also a single point of failure. Web and HTTP proxy software provide similar functionality to Anonymizer and suffer from the same drawbacks.

Crowds by Rubin et al. [69] is one of the approaches to anonymous communication. A Crowd is a set of voluntarily cooperating hosts. Any message that requires anonymity first channels into one of the Crowds hosts and then enters a loop during which each crowd member probabilistically forwards the packet to another until it finally gets out of the Crowd and arrives at the destination. Through statistical forwarding decisions, Crowds can effectively hide the communication pattern of a user. Crowds provides anonymity only for web clients. Furthermore, Crowds is concerned only about receiver-anonymity.

Another similar approach is Onion Routing [70]. Users can use the deployed set of Onion routers in the Internet to achieve a level of privacy similar to that of Crowds. Onion routers themselves form a ring and keep constant TCP connections between the neighboring routers, constantly transmitting packets through the routes. Also, packets are encrypted with multiple keys to form an "onion," so none of the Onion routers forwarding the packets can discover both the source and the destination information of the packet. NetCamo [71] is an approach to counter traffic analysis in real-time. NetCamo models the traffic patterns of nodes or networks and provides a real time rerouting and padding to hide the communication pattern. Both Onion Routing and NetCamo are designed to server Internet applications, and they work atop a wired TCP/IP network.

Scarlata et al. [72] present Anonymous Peer-to-peer File Sharing (APFS), which is an extension of Crowds that provides mutual anonymity for senders and receivers. It is mainly targeted for peer-to-peer file swapping application over the Internet.

Wright et al. [73] outline a number of privacy attacks that can degrade the anonymity of some protocols, like Crowds, Onion routing, and Hordes. The paper formally proves that many privacy protocols suffer anonymity degradation when a particular initiator continues communication with a particular responder across path reformations. After providing a general algorithm for an attack, the authors describe particular attacks that can be executed on particular protocols that meet certain assumptions. The work provides good insight on what should be avoided while designing a good anonymity protocol.

Concerning privacy usability issues, the Platform for Privacy Preferences (P3P) [74] attempts to provide transparent-use privacy policies for web browsing applications. However, this is presented in a very limited context and has not proven to be very popular since end-users do not understand these policies and their consequences.

With respect to specifying privacy policies, some related work includes the recent effort by the OASIS consortium sub committee to specify privacy policies within the XACML (Extensible Access Control Markup Language) standard [75]. Privacy policies in XACML for e-business transactions are specified with respect to what are called custodians and owners. The custodians are entrusted with sensitive information belonging to the owners and are bound by a set of obligations. These obligations include security safeguards, integrity constraints, use limitations, purpose specification, and limits on how data can be shared. Purpose specification is similar to the notion of intent. The XACML specification is fairly general and our formalism would refine these specifications while adding the necessary semantics.

The NIST common criteria defines the terms anonymity, pseudonymity, unlinkability and unobservability as four classes of properties of user privacy requirements [76].

2.3 Automated Reasoning

Several different approaches for automated reasoning exist. In this section I survey some of the existing tools that are relevant to this dissertation.

2.3.1 Predicate Logic

Most of the early reasoning frameworks, particularly in AI, were based on predicate logic. Logical statements written as well-formed formulas were used to represent real-world facts and available knowledge. Predicate logic's ability to accommodate variables and quantifiers made it an attractive method for knowledge representation. Simple reasoning can be done by deducing new information from known facts by using the well-defined rules of inference. However, this type of knowledge representation and reasoning assumes a precise, consistent, and unchanging model of the world and ignores the problems of uncertainty and approximate reasoning, making formal predicate logic useful only when information is precise and static.

2.3.2 Default Logic

Halpern et al. [77] and Poole [78] suggested extensions to formal logic for handling uncertainty. Poole presented *Default Logic*, where there are two types of knowledge, namely, *facts* and *defaults*. Facts are information that is always known to be true in the world, for example, a "lion is a mammal." On the other hand, defaults represent information that is uncertain, i.e., information that is not always true. For instance, the statement "birds fly," is not always true. An example demonstrating this, derived from Poole, is as follows:

```
Default 1:  person(x)   → can_walk(x)
Default 2:  machine(x)  → ¬ can_walk(x)
Default 3:  robot(x)    → can_walk(x)
Fact 1:     person(bob)
Fact 2:     machine(printer)
Fact 3:     machine(protocol-droid)
Fact 4:     robot(protocol-droid)
```

Now this type of logic admits as a theorem any expression that is valid using any of the defaults. For example, if the available expression is:

```
¬ can_walk(protocol-droid)
```

Then default logic explains it based on the stated facts and default 2. However, if we are given the expression:

```
can_walk(protocol-droid)
```

Then it is explained based on the facts stated and default 3. According to Poole, this approach views reasoning as a very simple case of theory formation, rather than deduction from available knowledge.

If a decision among the defaults is needed to solve a problem, then some other mechanism needs to be deployed. While the theory of default logic does not say much about how to choose among the different defaults, Reiter [79] discussed this issue in detail in his "default theory." In this theory, there are two sets of expressions, D and W. W represents a set of expressions in traditional first order logic. D represents a set of defaults that are expressed in a certain format. Each default rule consists of three parts; a *prerequisite* that must be proven using traditional logic, a *consistency test* which must be consistent with current rules, and the *consequence* which is the information deduced if the former parts were evaluated successfully.

Segerberg [80] and Gardenfors [81] represented uncertainty associated with default knowledge through the use of statements like $p \geq q$, which indicates that the default knowledge p is more likely than q. The $\geq$ notation is used without assigning numeric values to p and q.

2.3.3 Modal Logic

In predicate or propositional logic, statements are either true, or false. These types of logic do not accommodate other possibilities. However, in real life this may be inadequate. For instance, in natural languages we often distinguish between facts that are true in any circumstance forever, e.g. "*Alice is a person*", "*a lion is a Mammal*", or "*the square root of 4 is 2.*" On the other hand, some statements may be true in certain circumstances or certain time periods, but may not hold in the future, like "*there are nine planets in the solar system,*" or "*machines cannot walk.*" The latter two statements can be currently true, but they may not continue to be true at some point in the future, for example.

The term "modal logic" is used in a broad fashion to cover a family of related logics that are able to qualify the truth of a judgment and reason about different "modes" of truth. Different types of modal logic use a variety of different symbols, which represent unary connectives that express one or more of the different modes of truth. Among the widely used unary connectives are the necessity symbol '$\Box$' (necessarily true) and the possibility symbol '$\Diamond$' (possibly true) [82, 83]. For instance, if A is a theorem in a particular universe of discourse *(U)* then we write $\Box A$. The $\Diamond$ operator can then be defined as $\Diamond B = \neg \Box \neg B$. Logics that express a level of belief or temporal information are all types of modal logic.

One common application of modal logic is its ability to capture the notion of a multi-agent system in which the autonomous agents interact with each other. Each of these agents may have different knowledge about the environment and the knowledge of other agents. Modal logics can take into account not only the facts of the world, but also the knowledge of other agents in the universe of discourse, which may carry different levels of truth. The ability to reason within such environments is particularly useful for ubiquitous computing scenarios. This is because only partial context information may exist at different devices or sensors of an Active Space. Multi-agent modal logic techniques can help in aggregating such information and making sense out of it.

Halpern et al. [77] presented a modal logic LL to reason about *likelihood*. In their modal logic they introduced the modal operator L which captures the notion of being *likely*. For instance, L_P means that *"p is likely to be true."* This approach can be used to represent the level of uncertainty associated with a proposition without assigning numeric values.

2.3.4 Temporal Logic

Temporal logic [84, 85] extends traditional logic by adding new operators that can capture time, changes, and state transitions. Temporal logic has shown its usefulness in modeling concurrent systems as a result of its ability to express the ordering of events and define the semantics of temporal expressions in a logic framework. Temporal logic has also been used as a language for encoding temporal knowledge in expert systems and several AI fields. For instance, Allen [86] has focused on developing a general framework for all the temporal representations needed by AI applications. Galton [87] provided a survey of issues involving temporal reasoning in AI.

Although there are different variations of temporal logic, most of them introduce the following "basic" temporal operators:

- **G:** which implies "global truth", for example: "**G** *s*" implies that statement *s* is *always* true. Regardless of time.
- **F:** implies "in the future" or "eventually." Hence, "**F** *s*" implies that statement *s* must be true sometime in the future.
- **X:** implies next time unit, i.e. "**X** *s*" means that *s* will be true in the *next* time unit.
- **U:** the binary until operator. "s_1 **U** s_2" means that s_1 holds all the time *until* s_2 holds. From this point onwards we do not care about the state of s_1.
- **R:** release operator, which is the dual of the **U** operator. It requires the second statement to hold up to and including the time instant the first statement holds. However, the first statement is not required to hold eventually.

Security and privacy services are affected by the current context. Context changes with time, and thus the security and privacy requirements and deployed mechanisms will change as time progresses. A potential direction that seems to be promising is the deployment of temporal logic in the reasoning process. This enables us to capture the effect of time and the changes it introduces. Intuitively, many security mechanisms are "tied" to time. For example, if we want to query (using predicate logic) whether Bob has access to some device:

```
Access (Bob, Printer)
```

If the above statement is evaluated *before* the authentication of Bob takes place, it will evaluate to false:

```
Access (Bob, Printer) = false
```

However, if Bob is successfully authenticated to the Active Space, then only at that particular *time* the original predicate will evaluate to true. Similarly, access to other critical devices may require Bob to authenticate himself using several different mechanisms, hence, `Access (Bob, critical_device)` may continue to evaluate to false *until* Bob goes through all the necessary authentication mechanisms.

An interesting approach is to extend temporal logic by enabling it to capture other context information, i.e. develop a context-based logic!

2.3.5 Probabilistic Inference

For uncertainty reasoning, one of the most useful approaches is *Bayes' theorem*, which is obtained from the definition of conditional probability. If, for example, *E* stands for an event and *C* stands for a related condition to that event, then the probability of that event given condition *C* is stated as:

$$P(E \mid C) = \frac{P(C \mid E)P(E)}{P(C)}$$

Probability inference is based on the idea that observed events can, in general, be used to infer the probabilities of other unobserved events. However, for this to work, one has to have prior knowledge of the probability that *E* would occur, and the probability that *C* would occur given *E.* This information can be obtained by prior observations. However, since prior observations are often based on limited samples, the values assigned to the prior probability involve chance in some way. Thus, in Bayesian probability the answer to the question we are concerned with depends on chance, given that the chances of the different answers are known. If some answers are not known, then their values can be calculated by "conditioning."

Bayes' theorem gives rules for inference in cases where additional evidences are available:

$$P(H \mid E, C) = \frac{P(H \mid C)P(E \mid H, C)}{P(E \mid C)},$$

This extended form allows us to update our belief in hypotheses *H* given additional evidence *E* and the background context *C.*

Frameworks for reasoning based on this approach were laid down by different researchers. Because probabilistic dependencies in large reasoning systems could become very complex and hard to manage (e.g. imagine n binary variables, then we need to deal with 2^n probabilities!), Pearl et al. [88] introduced the idea of Bayesian networks. Bayesian networks is a framework that creates a reasoning situation where relationships among various aspects of a situation are known in terms of conditional probabilities, and the knowledge of observed events can in general be used to infer the probabilities of other unobserved events. Bayesian networks are best used in situations where (1) it is possible to reason backward from evidence to hypothesis; (2) it is possible to have a complete model of the situation where all probabilities add up to 100%. Bayesian networks use directed acyclic graphs (*DAGs*) to represent chains of probabilistic dependencies. I.e. a Bayesian network is a graphical representation of the independence relationship in the known joint probability distributions included in the Bayesian network. The DAGs nodes represent propositional variables, and the edges express casual relationships between the nodes, which translate to conditional probability. These DAGs are then used for reasoning. A sample Bayesian network is shown in Figure 1. The join probability for this DAG can be written as:

P(x6|x5) P(x5|x2,x3) P(x4|x1,x2) P(x3,|x1) P(x2|x1)P(x1)

Each term in the above is associated with a set of edges at various nodes of the network. The network represents the complete knowledge about the domain. While reasoning, when events are observed, the values are updated and the associated conditional probabilities are used to calculate the probability of unobserved events.

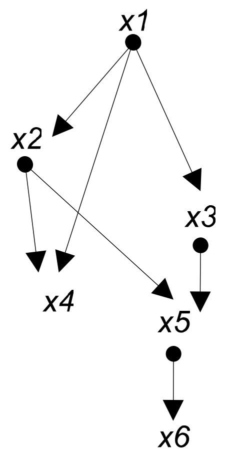


Figure 1: A Sample Bayesian Network

Methods for deploying these DAGs in expert systems and using them for reasoning and inferencing are given in Pearl et al. [88] and Neapolitan [89].

2.3.6 The Dempster-Shafer Theory of Evidence

Shafer et al. [90] developed a "calculus" for reasoning, known as "the Dempster-Shafer Theory of Evidence." Like Bayesian probability, the theory of evidence is based on prior knowledge obtained through prior observations or known probability values. However, instead of using examples where known chances are attached directly to the possible answers to the given question, the theory of evidence uses examples where known chances are attached only to the possible answers to a *related* question [90]. Furthermore, Bayesian probability (and most other probability-based approaches) limits all outcomes to two possible values only, e.g., if an event is 30% likely, then it must be 70% unlikely. In Dempster-Shafer Theory of Evidence, it is possible to have situations where an event is 30% likely, 40% unlikely and 30% uncertain. In essence, the theory takes into account the likelihood, the unlikelihood, and the uncertainty. This is illustrated in Figure 2.

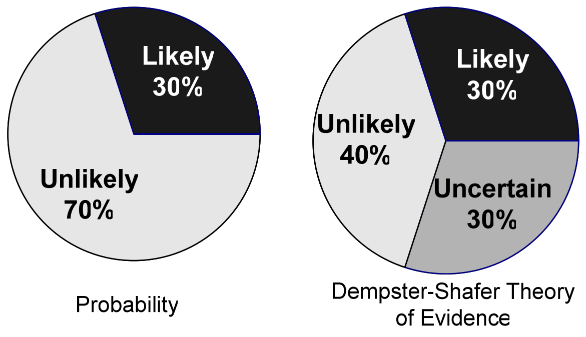


Figure 2: Probability vs. Dempster-Shafer Theory of Evidence

Dempster-Shafer theory uses available evidence to define a probability mass function. This function assigns finite amount of belief to possible subsets in a finite set of possibilities. The finite set of possibilities is called the *Universe of Discourse (U).* For every subset $S \subset U$, the theory defines three functions:

- A probability mass function *p(S)*
- A degree of belief, *bel(S),* representing the sum of all evidence that support *S* or a subset of *S.*

  $$bel(S) = \sum_{X \subset S, X \neq \phi} p(X)$$

- A *degree of plausibility, Pl(S),* which is defined by the sum of all evidence that can possibly support *S* (or its subsets):

  $$Pl(S) = \sum_{X \cap S \neq \phi} p(X)$$

In other words, *Pl(S)* expresses how much we should believe in *X* assuming all unknown facts were to support *S.* Thus, the true belief in *S* will be somewhere in the interval [*Bel(S), Pl(S)*].

If two or more evidence with probability values exist, it is necessary to combine them using *Dempster's rule of combination* for belief functions. In case of two evidences p_1 and p_2, then the combined evidence is $p_1 + p_2$, and is defined as follows:

$$p_1 + p_2(S) = \frac{\sum_{X \cap Y = S} p_1(X) \cdot p_2(Y)}{1 - \sum_{X \cap Y = \phi} p_1(X) \cdot p_2(Y)}$$

The following example shows how Dempster-Shafer theory can be used for reasoning under uncertainty. This example is derived from the one shown in Shafer [91]. Suppose that a server has two intrusion detection systems (IDS), *X* and *Y,* which try to detect possible unauthorized intrusions. Suppose further, that *X*, which is known to be accurate 80% of the time, indicates that an intrusion has occurred. *Y* on the other hand is accurate only 10% of the time. Assume that *X* detected an intrusion whereas *Y* did not detect any anomalies. Given this scenario, the Dempster-Shafer theory can be utilized to give us a level of belief to whether an intrusion actually took place or not.

Note that $U = \{yes, no\}$, which is the answer to the question "*is there an intrusion?*" Note that we have two evidences (what *X* and *Y* are reporting, let these be p_1 and p_2 respectively). Therefore, we have:

$p_1(\{yes\}) = 0.8$ (probability of *X* being accurate in reporting an intrusion).

$p_1(\{no\}) = 0$ (unknown)

$p_1(\{yes, no\}) = 0.2$

Likewise, we have:

$p_2(\{no\}) = 0.1$ (probability of *Y* being accurate in not reporting an anomaly).

$p_2(\{yes\}) = 0$ (unknown)

$p_2(\{yes, no\}) = 0.9$

We apply Dempster's rule of combination to combine both evidences. Let *m* represent the combined evidences, then:

$p_1 + p_2 (\{yes\}) = (0.72+0) / (1-0.08) \cong 0.7826$

$p_1 + p_2 (\{yes, no\}) = (0.18+0) / (1-0.08) \cong 0.1957$

$p_1 + p_2 (\{no\}) = (0.02+0)/(1-0.08) \cong 0.0217$

Thus, $Bel(\{yes\}) = 0.7826$. $Pl(\{yes\})$ = 0.7826+0.1957=0.9783. This implies that the degree of belief that an intruder has gained unauthorized access to the system is in the interval [0.7826, 0.9783]. This is illustrated in Figure 3, where *E* represents the event that an intrusion occurred in the system.

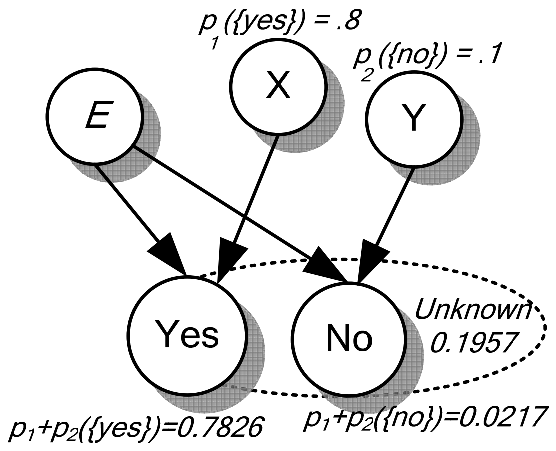


Figure 3: A Belief Network presenting the testimonies of IDS X and Y

2.3.7 Fuzzy Logic

Fuzzy logic is calculus for representing uncertainty which is based on fuzzy set theory [92, 93]. Fuzzy set theory associates a real number between 0 and 1 with the membership of a particular element in a set. This way, fuzzy logic avoids the serious limitation of classical logic where everything is based on "black-and-white" reasoning. Fuzzy logic provides much more flexibility by extending the notion of logic to capture "partial truths" allowing it to represent values that may range from "completely true" to "completely false." To achieve this, truth values are represented using elements of a given set, usually the interval [0, 1]. Unlike probability theory, the values assigned in fuzzy logic represent "degrees of membership" rather than "probability." Fuzzy logic is useful in capturing and representing imprecise notions like "tall," "trustworthy," and "confidence" and reasoning about them. For example, if S is a fuzzy set whose associated membership function is f, such that: $f\colon U \rightarrow [0, 1]$, where U is the universal set. For any $e \in U$, e is completely out of S if $f(e) = 0$, e is completely in S if $f(e) = 1$. Values between 0 and 1 represent the "degree" in which *e* belongs to *f,* for example, the fuzzy set *S* could represent "Tall people," and *f* can be the function plotted in Figure 4.

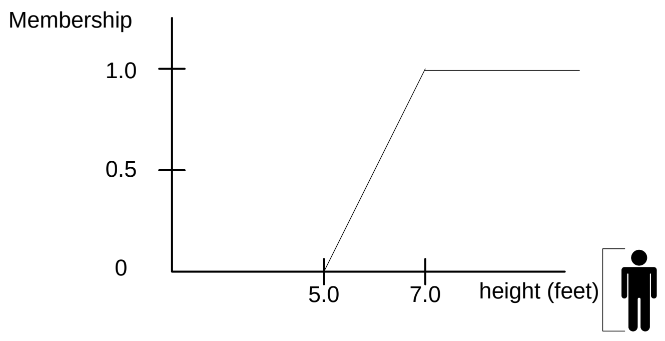


Figure 4: Membership Function for "Tall."

The function above indicates that people who are 7' tall or higher are considered to be certainly tall, whereas people 5' or shorter are not tall at all. All values heights in between represent different degrees of "tallness."

There are a number of widely accepted, basic fuzzy logic operations [93] these include:

- $a \wedge b$ = Min(a, b)
- $a \vee b$ = Max(a, b)
- $a \otimes b$ = Max (0, a+ b -1)
- $a \rightarrow$ b = Min(1, 1-a+b)
- $a \leftrightarrow$ b = $(a \rightarrow b) \wedge (b \rightarrow a)$
- $a^p = a \otimes \ldots \otimes a$ (p times)

For all $a, b \in [0, 1]$.

Although fuzzy logic provides a method for modeling imprecise and inexact information while providing powerful tools for inference and reasoning under uncertainty, it does not take “context” or “time” into account, thus the imprecision is always “fixed.” I argue that while ubiquitous computing environments contain a lot of imprecise information that require reasoning, the precision of information changes depending on context and the progress of time. Hence, combining techniques from both fuzzy logic and temporal logic is a promising research direction.

3. Problem Statement

Although several prototype implementations of some aspects of ubiquitous computing have started to appear; the construction of complete, integrated systems and their real-life deployment are still things of the future. In fact, many researchers in this new field are still investigating where the technology would go, and what would be the "killer" applications or services that could spring out of it. Security in ubiquitous computing is expected to be an integral part and a global property of the whole system, which is not done yet, and has a long way to go. Therefore, security issues and privacy concerns in this domain are open problems that require intensive research. It should be noted, however, that there is no single "magical" protocol or mechanism that can address all the security issues and meet the requirements and expectations of secure ubiquitous computing. Moreover, security itself consists of a variety of different and broad aspects each of which requires detailed research and customized solutions. For these reasons, I start off my contributions with the introduction of a "broad framework" for identification and authentication in ubiquitous computing. This framework addresses the major problems and the special requirements in securing ubiquitous computing, and provides a set of solutions that takes into account the goals and special needs that this technology demands. In addition, I attempt to provide an in-depth examination and implementation of some of the more specific security and privacy aspects. In particular, I provide an in-depth exploration of those aspects which pertain to identification, authentication, and privacy preservation, all within a framework that is unobtrusive, intelligent, and context-aware. In this dissertation I construct a middleware layer for security and privacy that incorporates automated reasoning and inferring capabilities, thus, creating an environment with ambient intelligence and security. As a proof of concept, I deploy these technologies in our Active Space testbed and assess their usefulness and practicality in real-world scenarios. Finally, I plan to evaluate the proposed solution in terms of performance, security, usability, and unobtrusiveness.

3.1 Identification and Authentication

No matter what computing paradigm is deployed, identification and authentication of entities represent the essential first step of any security system. "*Who are you?*" and "*can you prove it?*" are the first logical security questions to ask. Since security in any system is only as good as its "weakest link," deploying strong access control mechanisms and intensive auditing is weakened severely if user verification or authentication process is weak. In the last few years, we have witnessed significant technological advances in authentication mechanisms. Back when mainframes and time-sharing systems dominated the computing world, usernames and passwords were the only mechanism available for authenticating users. This technique was sufficient at that time given that each user had a single account and communication data and resource accesses had to go through controlled channels making them hard to fall into unauthorized hands. Additionally, since the mainframe itself is usually located at a physically secure location, security threats to the system were limited. Later, when computing evolved into distributed systems, password-based authentication became inadequate. This is because eavesdroppers could pick up passwords while they were transmitted over the wire or typed into a terminal. Further, it is inconvenient for a user to remember and type different passwords for different machines or services. In addition, the user is not recognized as one single user of the system; rather, he or she appears to be different users as there is no coordination of his or her use of the different servers within the distributed system. To address the new challenges, several distributed authentication mechanisms were introduced. These included Kerberos [94] and SESAME [52].

Unfortunately, these traditional authentication methods require much user interaction in the form of manual logins, logouts, and file permissions. These manual interactions violate the disappearing computer vision and imperil its ubiquitousness. The security requirements of an Active Space may vary according to the context of the space. Some situations (like during a confidential meeting or homeland security alerts) require greater security to be in place; while other situations may not require a very high level of security. Traditional security mechanisms are context-insensitive, i.e. they do not adapt their security policies to a changing context. Additionally, authentication is also concerned with associating attributes, privileges, and role names with the authenticated principal. While role-based access control systems (RBAC) assign principals role names that are based on the organizational hierarchy of the domain, these roles are often static and context-insensitive. Dynamic, context-based attributed and roles need to be defined and incorporated into the security system in order to make it as dynamic as the Active Space itself.

3.2 Privacy

The right to privacy is the basis for many social behaviors and laws. However, privacy has always been lacking in information technology systems. In ubiquitous computing environments the situation is even worse. This is because much of the infrastructure is identifying. Some sensors, for instance, are trying to track people. The transparent devices and sensors that populate Active Spaces might be capturing too much information about the inhabitants. While privacy can take various shapes, in many cases, total anonymity is not desirable. After all, for security purposes, authorized users should be authenticated and verified before accessing sensitive information. Further, access control mechanisms and security audits become useless if total anonymity is permitted. These reasons call for protocols that can strike a correct balance between privacy, while allowing entities to authenticate to the system in a secure manner.

3.3 Research Testbed

Gaia [2, 95, 96] is a component-based, middleware operating system that provides a generic computational environment. Gaia provides the necessary core services to support and manage Active Spaces and the pervasive applications that run within these spaces. By using Gaia, it is possible to construct a multipurpose, prototype Active Space. The Active Space is a large lab that consists of the Gaia middleware OS managing a distributed system composed of five 61" plasma displays, four of which have touch-screen panels attached to them to enable seamless interaction through hands or smart pens. Additionally, the room contains a HDTV display, 5.1 surround audio system (Dolby Digital version 5.1), multiple webcams, Tablet PCs, X10 devices, IR beacons, Bluetooth, gigabit Ethernet, wireless Ethernet, fingerprint scanners, Iris scanners, smart phones, RFID badges and detectors, and Ubisense™ location technology. The room is powered by 15, 1.7 GHz, Pentium-4 PCs running MS Windows™ XP and MS Mobile Windows™ based PDAs. The Gaia OS supplies services including event delivery, entity presence detection, context notification, a space repository to store information about entities in the space, a context-aware file system, a session manager and other core services. The room has been used extensively as a crucible for the new possibilities that ubiquitous computing provides for enhancing daily tasks and collaborative activities.

3.4 Proposed Solution

What is needed is a "ubiquitous," unobtrusive, and transparent security architecture for ubiquitous computing that meets the requirements stated below.

3.5 Security Requirements

3.5.1 Transparency and unobtrusiveness

The focal point of ubiquitous computing is to transform users into first class entities, who no longer need to exert much of their attention to computing machinery. Therefore, even the security subsystem should be transparent to some level, blending into the background without distracting users too much.

3.5.2 Multilevel

When it comes to security, one size does not fit all. Hence, the security architecture deployed should be able to provide different levels of security services based on system policy, context information, environmental situations, temporal circumstances, available resources, etc. In some instances, this may go against the previous point. Scenarios which require a higher-level of assurance or greater security may require users to interact with the security subsystem explicitly by, say, authenticating themselves using a variety of means to boost the system's confidence.

3.5.3 Context-Awareness

Often, traditional security is somewhat static and context insensitive. Ubiquitous computing integrates context and situational information, transforming the computing environment into a sentient space. The security aspects of it are no exceptions. Security services should make extensive use of available context information. For example, access control decisions may depend on time or special circumstances. Context data can provide valuable information for intrusion detection mechanisms. The principal of "need to know" should be applied on temporal and situational bases. For instance, security policies should be able to change dynamically to limit the permissions to the times or situations when they are needed. However, viewing what the security policy might become in a particular time or under a particular situation should not be possible. In addition, there is a need to verify the authenticity and integrity of the context information acquired. This is sometimes necessary in order to thwart false context information obtained from rogue or malfunctioning sensors.

3.5.4 Flexibility and customizability

The security subsystem should be flexible, adaptable, and customizable. It must be able to adapt to environments with extreme conditions and scarce resources, yet, it is able to evolve and provide additional functionality when more resources become available. Tools for defining and managing policies should be as dynamic as the environment itself.

3.5.5 Interoperability

With many different security technologies surfacing and being deployed, the assumption that a particular security mechanism will eventually prevail is flawed. For that reason, it is necessary to support multiple security mechanisms and negotiate security requirements.

3.5.6 Extended boundaries

While traditional security was restricted to the virtual world, security now should incorporate some aspects of the physical world, e.g. preventing intruders from accessing physical spaces. In essence, virtual and physical security become interdependent.

3.5.7 Scalability

Ubiquitous computing environments can host hundreds or thousands of diverse devices. The security services should be able to scale to the “dust” of mobile and embedded devices available at some particular instance of time. In addition, the security services need to be able to support huge numbers of users with different roles and privileges, under different situational information.

3.5.8 Intelligence and Automated Reasoning

Mark Weiser’s envisioned computing environments that are pervaded with a large number of computing devices and sensors to the extent that these devices disappear, allowing humans to focus on daily tasks rather than focusing on the underlying technology. To enable this vision, it is necessary to transform today’s machines, which are “dumb,” context-insensitive, and isolated, to intelligent, programmable, and context-aware clusters of machinery . To meet these requirements, Active Spaces must incorporate some forms of automated reasoning and advanced context-capturing and filtering, which allow them to adapt to users’ habits under different circumstances, without requiring much user intervention or manual configurations.

It is sensible to construct a security architecture that is as dynamic and as intelligent as the Active Space, which it is trying to secure. The “intelligent” security system should be able to make judgments and give assistance in securing the environment without too much intervention by users or administrators. Therefore, it is promising to explore the possibility of incorporating automated reasoning and learning into the Active Space security architecture, enabling it to perform intelligent inferences under different contexts despite the uncertainties that arise as a result of bridging the physical and virtual worlds. In order to do this, I apply some AI techniques for reasoning under uncertainty. As illustrated in Figure 5, by incorporating intelligence in the space in the form of reasoning under uncertainty it is possible to bridge the gap between the physical and computational worlds through calm interfaces without distracting users.

3.6 Artificial Intelligence in Ubiquitous Computing

Today, personal computers are intrusive and require a lot of attention from users. The context-insensitivity and the gap between the computational and physical worlds require users to utilize portion of their intelligence to interact with computers effectively (see Figure 5). One key issue of ubiquitous computing is unobtrusiveness to the extent that allows us to realize the disappearing computer vision. This is an important issue in ubiquitous computing in general and even more so when designing the security infrastructure, because security mechanisms have a tendency to obstruct usability [97].

Several branches of Artificial Intelligence are concerned with the construction of

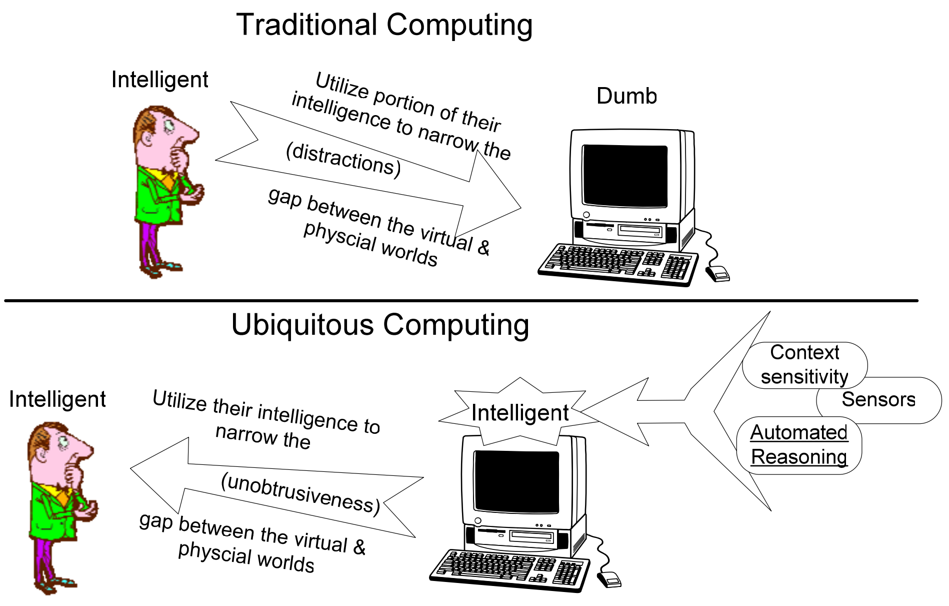

Traditional Computing
Intelligent
Utilize portion of their intelligence to narrow the
(distractions)
gap between the virtual & physcial worlds
Dumb
Ubiquitous Computing
Intelligent
Utilize their intelligence to narrow the
(unobtrusiveness)
gap between the virtual & physcial worlds
Intelligent
Context sensitivity
Sensors
Automated Reasoning

Figure 5: The need for "intelligence" in Ubiquitous Computing Environments

autonomous, thinking or decision making machines or robots. Designing those autonomous systems is usually complicated, because all the intelligence is concentrated in one machine. Moreover, this single machine has to have its own sensors, pattern matching capabilities, reasoning, and decision making algorithms to exhibit any useful behavior. For this reason, most successes were in building machines or robots that can perform very specialized tasks, i.e. a robot that can follow a colored ball. In ubiquitous computing environments, the whole surrounding environment becomes smart and intelligence is spread across the environment through the sensors, devices, smart appliances, and context-capturing capabilities that are embedded everywhere. Adding automated reasoning and decision making engines that can utilize the information provided by the surrounding infrastructure would lead to components that can effectively reduce the amount of user interactions and manual configurations needed in the ubiquitous computing environment.

Lueg [98] observes that many visions of ubiquitous computing applications borrow ideas from AI. The author notes that many of these visions lose touch with reality by envisioning infeasible scenarios that do not take into account areas that have been explored in AI or other disciplines. Therefore, my approach is build on top of approaches that are well-established in AI and proved to be effective in some areas. In particular, I attempt to reap the benefits of using an "Expert System" and its application in ubiquitous computing environments.

3.7 Expert Systems

An Expert System is a computer program that attempts to mimic the way in which a human expert reasons and make decisions to solve problems in a specific field. A clear example is an expert system which could perform some medical diagnosis, and recommend appropriate therapies for patients with bacterial infections, like the famous MYCIN system [99]. It is apparent that the computer expert system would lack much of the resources and capability that a human expert has. Nevertheless, in conventional AI, the computer expert could still be of great value when combined with a novice computer user. In this way the computer expert would rely on the human to perform actions and observations the computer expert is incapable of doing. The computer expert and the novice then could form a team which might perform as well as the human expert alone and at a probably much lower per-hour labor rate. However, I argue that an expert system can be used in a different manner within ubiquitous computing environments. As illustrated in Figure 6, instead of requiring human interactions with the expert system, the expert system can: (1) use the smart infrastructure and the context-capturing framework as inputs. (2) Perform inductions, make decisions, and take actions, based on the programmed rules and captured context and without requiring too much intervention from regular users.

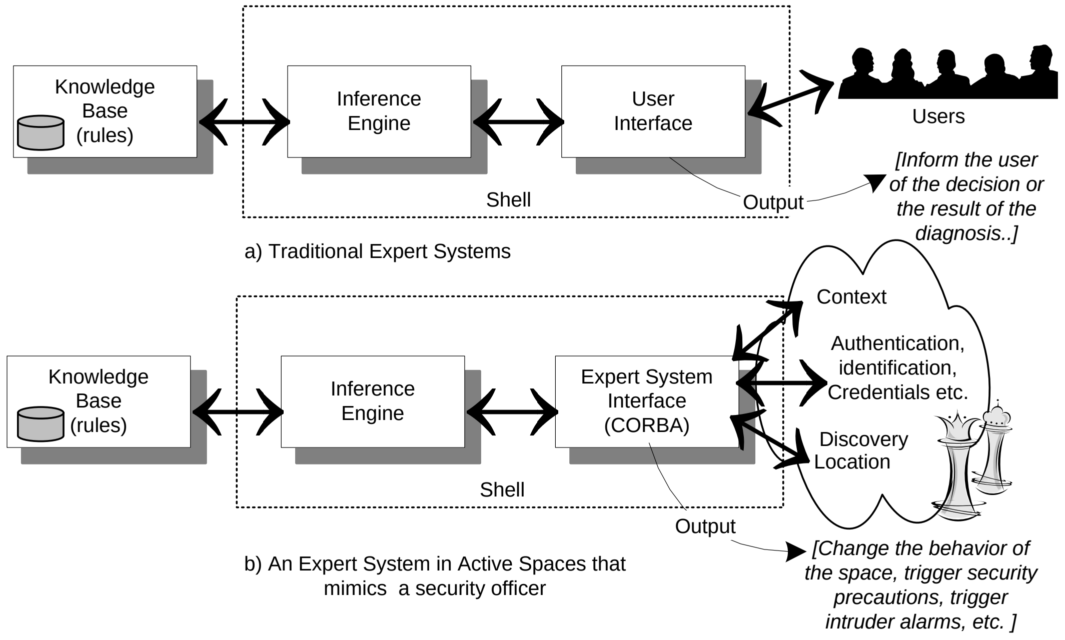

Knowledge Base (rules)
Inference Engine
User Interface
Users
Shell
Output
[Inform the user of the decision or the result of the diagnosis..]
a) Traditional Expert Systems
Knowledge Base (rules)
Inference Engine
Expert System Interface (CORBA)
Context
Authentication, identification, Credentials etc.
Discovery Location
Shell
Output
[Change the behavior of the space, trigger security precautions, trigger intruder alarms, etc.]
b) An Expert System in Active Spaces that mimics a security officer

Figure 6: Expert Systems

Any expert system consists of three components: a knowledge base, an interface, and an inference engine. Usually, the last two components are referred to as the expert system shell. The shell is independent from the domain of the expert system. Knowledge bases for different domains can be plugged into the same shell. The inference engine consults the knowledge base, checks available facts obtained from the environment, and merges facts with rules to deduce new facts or action plans. The knowledge base is the core of any expert system. It typically contains two types of information: facts and rules. The facts are pieces of information that is known before execution, or they are asserted during execution. The rules are usually domain-dependent heuristics that enable the system to deduce relevant facts or make decisions.

The construction of expert systems that can exploit information-rich Active Spaces to mimic the behaviors of security officers, access providers, trust brokers, and other security-related aspects is a new idea that has not been explored yet by researchers in ubiquitous computing.

4. An Authentication Framework for Ubiquitous Computing Environments

Authentication is a cornerstone in the security of any system. Authentication provides assurance for the claimed or detected identity of an entity in the system, i.e. it verifies whether the identification of this entity is correct. Identification is the process of linking an entity with an identity. This process can be initiated by the entity itself (a user typing his user ID) or inferred by the system through sensors. Principal refers to the entity whose identity has been established. An entity in this definition could be people, programs, devices, sensors, objects, or physical spaces.

In this chapter I argue that ubiquitous computing requires novel approaches to authentication, in which traditional authentication mechanisms need to be tailored and adapted to ubiquitous environments in a manner that preserves the environment's ubiquity and unobtrusiveness. The aim is to enrich ubiquitous computing environments with ubiquitous means for identification and authentication. These novel mechanisms will enable better security, enhanced automation, and more flexible customizations.

4.1 Elements of an Authentication System

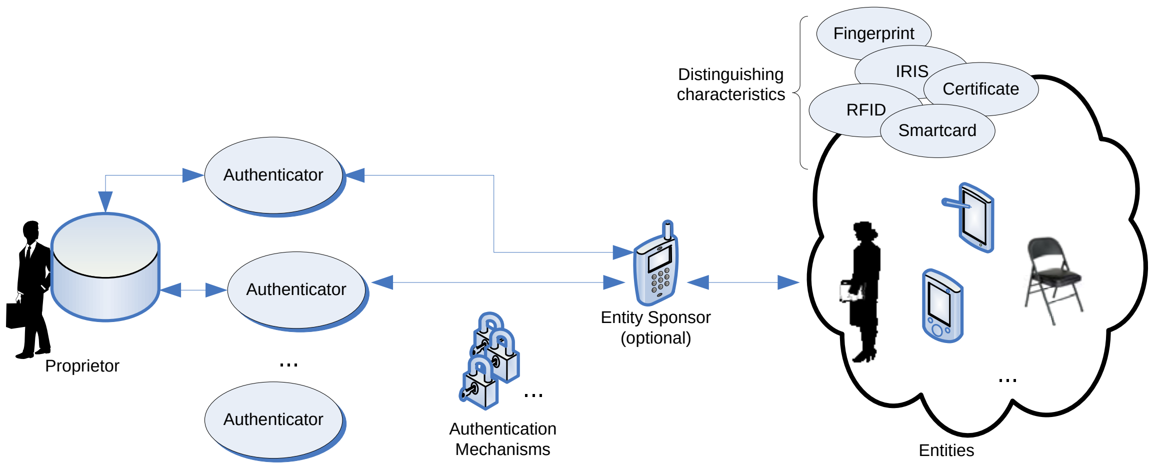


Figure 7: Basic Elements of an Authentication System

In order to understand the challenges of authentication in ubiquitous computing, it is necessary to take a closer look at the basic elements in any authentication system. Figure 1 depicts the various elements that are often present in a general authentication operation. First, there is an entity to be authenticated. The entity has one or more *distinguishing characteristics* that differentiate a particular entity from others. In some scenarios, the entity utilizes a special device (entity sponsor) to perform the authentication. E.g. in a password-based authentication the user would enter the password using a special terminal or device. In other authentication techniques, like RFID the entity does not need to interact with a sponsor device. The proprietor is the component that is responsible for authenticating users and distinguishing authorized entities from unauthorized ones. In some instances, the proprietor relies on other trusted components to authenticate identity, e.g., an iris scanner or fingerprint reader device will validate iris or finger scans. One or more authentication mechanisms are needed to verify the presence of the distinguishing characteristics. Security literature classifies the distinguishing characteristics into three types:

- *Something you know*. The distinguishing characteristic is secret information, which only the authorized entity knows, e.g., passwords and PINs.
- *Something you have*. The distinguishing characteristic is something that only the authorized entity possesses, e.g., a token, a smart card, an RFID badge, a digital certificate, or secret data embedded in a file or device. Note that this something can be physical (token) or virtual (digital certificate).
- *Something you are*. The distinguishing characteristic is some physical feature or behavior that is unique to the entity being authenticated, e.g. biometrics for humans (fingerprint, iris, voice recognition, face recognition, or DNA sequence!), or physical features of other entities, like MAC addresses or unique hardware IDs.

4.2 Key Challenges

Based on experiments and usage scenarios in our Active Space testbed, I identify a number of key issues that are required in the authentication subsystem in any generic ubiquitous computing environment. Active Spaces are mainly composed of collections of dynamically assembled components that make up services and ubiquitous applications, which fulfill the requirements of a user, or a group of users. Dynamism is probably the most important aspect of an Active Space, and requires that many of the core services be designed and implemented as a flexible component-based architecture capable of changing its own composition and state at run-time, as well as adapting to contextual changes in the environment. Given this premise, I list the unique issues and challenges that pertain to authentication in ubiquitous computing environments.

4.2.1 Unobtrusiveness

Traditional authentication mechanisms require too much user intervention in the form of manual logins and logouts. As illustrated in Figure 1, entities (particularly human users) often need to interact with an authenticator through an entity sponsor. These interactions are often required for presenting and/or validating the distinguishable characteristics. Furthermore, in many traditional authentication scenarios, entities need to log out explicitly. Taking into consideration that ubiquitous computing environments are rife with dynamic entities that leave and join frequently, it becomes impractical for users to manually log in and log out each time they enter and leave a physical room. All these unnecessary interactions could imperil the disappearing computing vision.

Eliminating authentication to lessen unnecessary interactions with the system and allow users to focus on their tasks is not a viable option. This is because ubiquitous computing environments at the very least need to be able to identify the various entities that inhabit the space, in order to utilize resources, meet expectations, and tailor itself in the best way to meet user needs and requirements. Moreover, security is an essential part in ubiquitous computing systems, and the lack of security can hinder their acceptance and widespread deployment.

4.2.2 Authenticating Animate and Inanimate Entities

In traditional authentication systems, the assumption often is that the authentication mechanism is authenticating people. However, in ubiquitous computing environments people are not the only entities that need to be authenticated. Unattended computer systems, mobile devices, PDAs, wearable devices, and even “smart furniture” may need to be authenticated. In fact, ubiquitous computing environments are made of a collection of collaborating devices each running different components of an application or a service, therefore, it is crucial to ensure that the various equipment that participate in the Active Space are authorized and are under the control of the right people or enterprise.

4.2.3 Multi-Mechanism Support

Since there are a large number of diverse authentication mechanisms and distinguishing characteristics that can be deployed for identification and authentication purposes, and as technology advances, we expect masses of new authentication mechanisms and devices to become available. This makes it necessary to offer dynamic and flexible means for accommodating new authentication technologies that can capture new types of distinguishing characteristics. Ideally, the authentication framework should provide seamless means for allowing new authentication technologies to be added to the system on the fly without downtime and without the need of reconfiguring existing applications and services.

4.2.4 Multi-Levelness

In conventional systems, the outcome of the authentication process is binary. Either the entity is authenticated through one or more distinguishing characteristics, and an identity is associated with it, or the authentication fails, and the entity is considered unauthorized. However, in an environment where users are surrounded by hundreds of devices, the notion of interacting or using a specific device or authenticating mechanism becomes inappropriate. Entities should be able to utilize different devices at different times, or use the most convenient technology at a particular time or situation. This “post PC” scenario requires a new model for authentication, which tolerates the use of different identification and authentication technologies, and builds up confidence as more distinguishing characteristics are validated.

4.2.5 Context Awareness

One of the main differences between an Active Space and a traditional distributed system is the utilization of the physical and digital context associated to the space as a default computational parameter. Context is one of the most important properties in ubiquitous computing [] and therefore authentication should accommodate situational and contextual information. Moreover, it must be able to access and alter existing context information. Context may trigger both functional and structural adaptation. As an example of functional adaptation, in emergency situations or during confidential meetings, stronger security is needed, and hence, authentication should be strengthened. In other situations, strong means of authentication may be unnecessary and minimal verifications may be adequate. An example of structural adaptation, on the other hand, involves discovering the resources available in the space and the distinguishing characteristics and the corresponding authentication mechanisms that can be used to authenticate entities best. For example, if a space has high levels of noise, then voice recognition is not the best option for authentication.

4.3 Approach

To address the issues described earlier, it is important to remodel authentication in ubiquitous computing environments in such a way that strikes a balance between authentication strength and non-intrusiveness. An active RFID badge that transmits short range radio signals, for instance, is a good non-intrusive authentication mechanism; however, it provides a weak form of authentication. A challenge-response mechanism provides stronger authentication, but may require more interactions on behalf of the user. To overcome this problem, the Gaia authentication service lets context "decide" how *strong* the authentication needs to be. This way, the Active Space does not dictate that users should carry or wear specific devices. The authentication process should enable principals to authenticate themselves to the system using a variety of means. These include the use of wearable devices, voice and face recognition, presenting a badge that contains identification information, fingerprint identification, retinal scans, etc. To enable this, I differentiate between different strengths of authentication by associating *confidence values* to each authentication process. This confidence value represents how "confident" the authentication system is about the identity of the principal. This is presented by a number in the interval [0, 1]. This confidence value is based on the authentication device and the authentication protocol used. Principals can employ multiple authentication methods in order to increase the confidence values associated with them. Access control decisions can now become more flexible by utilizing confidence information. Several reasoning techniques can be used to combine confidence values and calculate a net confidence value for a particular principal. The techniques I have considered so far include Bayesian probability and fuzzy logic.

In order to accommodate the large number of diverse authentication mechanisms and enable multi-mechanism support for authentication, it is necessary to have dynamic means for adding new authentication devices and associating them with different capabilities and protocols. Naturally, some methods of authentication are more reliable and secure than others. For example, it is easy for smart badges to be misplaced or stolen. On the other hand, the use of biometrics, iris scans for instance, is a fairly good means of authentication that is difficult to forge. Because of the various authentication methods and their different strengths, it is sensible to accommodate different levels of confidence and incorporate context and sensor information to infer more information or buildup additional confidence in a principal's identity. Furthermore, the same techniques can assist in detecting intruders and assessing their threat level.

The various means of authenticating principals and the notion of different confidence levels associated with authenticated principals constitute additional information that can enrich the context awareness of Active Spaces. This information is can be inferred and exchanged with other Gaia services and applications.

To meet the stated requirements I designed and implemented a federated authentication service that is based on distributed, pluggable authentication modules. Figure 8 provides a high-level sketch of the authentication architecture that incorporates the objectives mentioned above. PAM (Pluggable Authentication Module) [100] provides an authentication method that allows the separation of applications from the actual authentication mechanisms and devices. Dynamically pluggable modules allow the authentication subsystem to incorporate additional authentication mechanisms on the fly as they become available. The Gaia PAM (GPAM)

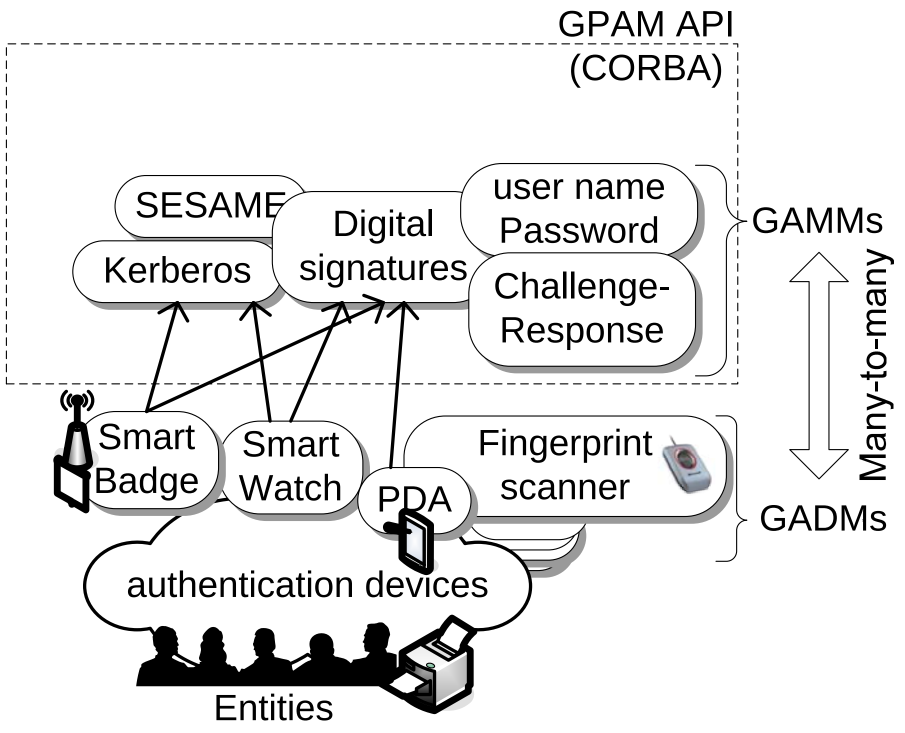


Figure 8: Gaia Pluggable Authentication Modules (GPAM)

extends traditional PAM by providing support for federated, CORBA-based authentication modules. This GPAM is wrapped by an API that is made available for ubiquitous applications, services, and other Gaia components to request authentication of entities or inquire about authenticated principals. Since the authentication service can be running anywhere in the space (possibly federated). I use CORBA facilities [55, 101] to allow the discovery and remote invocation of the authentication services that serve a particular Active Space. The authentication modules themselves are divided into two types: Gaia Authentication Mechanism Modules (GAMM), which implement general authentication mechanisms or protocols that are independent of the actual device being used for authentication. These modules include a Kerberos authentication module, a Tiny SESAME [50-52] authentication module, the traditional username-password module, a challenge-response through a shared secret module, fingerprint matcher module, etc. The other type of modules is the Gaia Authentication Device Modules (GADM). These modules are independent of the actual authentication protocol; instead, they are dependent on the particular authentication device. The GADM is comparable to traditional OS device drivers; however, they are tailored for authentication and identification purposes. Moreover, these GADM are federated components that can run on any node and not necessarily run on the authenticator device it is attached to.

This decoupling enables greater flexibility. When a new authentication protocol is devised, a GAMM module can be written and plugged in to support that particular protocol. Devices that can capture the information required for completing the protocol can use the new authentication module with minimal changes to their device drivers. When a new authentication device is incorporated into the system, a new GADM module is implemented in order to incorporate the device into the smart space, however, the device can use existing security mechanisms by using

CORBA facilities to discover and invoke authentication mechanisms that are compatible with its capabilities. In effect, this creates an architecture similar to traditional PAM but federated through the use of CORBA and secure communication channels.

In essence, the contribution of the Gaia authentication framework can be summarized in three main points. First, support for pluggable, stackable, authentication modules that can be federated and loaded only when needed. Second, the ability to combine several different identification and authentication mechanisms to build up confidence as more authentication credentials are presented and combined. Third, the utilizing of context-awareness as a key parameter in deciding how much authentication is enough, as well as enriching context with the notion of different confidence levels associated with different principals. In the next subsection, I give additional details on these three components of the Gaia authentication framework.

4.3.1 GPAM Control Flow

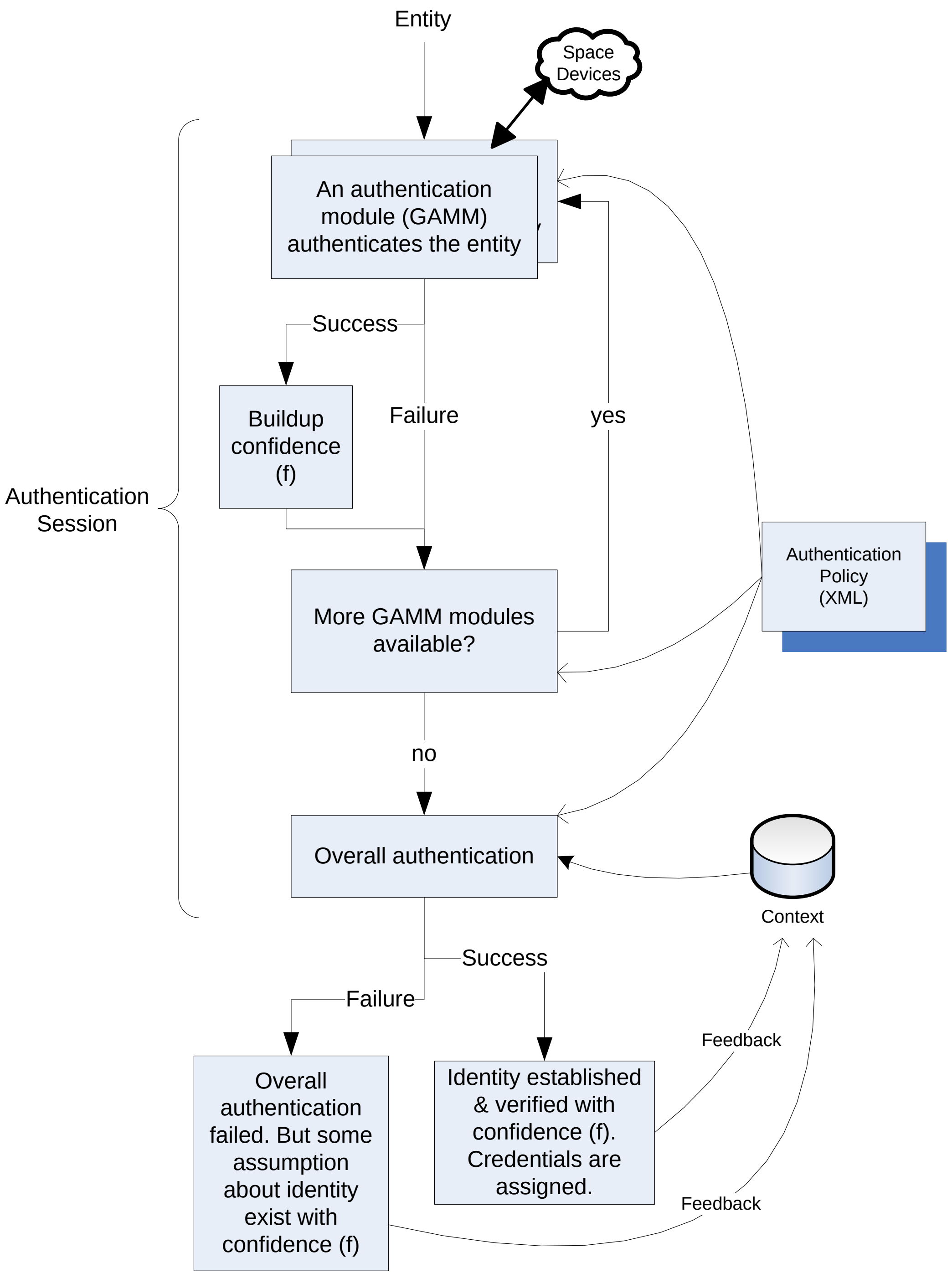

Entity
Space Devices
An authentication module (GAMM) authenticates the entity
Success
Buildup confidence (f)
Failure
yes
Authentication Session
More GAMM modules available?
Authentication Policy (XML)
no
Overall authentication
Context
Success
Failure
Feedback
Overall authentication failed. But some assumption about identity exist with confidence (f)
Identity established & verified with confidence (f). Credentials are assigned.
Feedback

Figure 9: Gaia Authentication Flow

The Gaia Pluggable Authentication Modules (GPAM) is an extension of PAM and the Java-based JAAS [100]. However, GPAM is unique in that it allows the different modules to be distributed through the use of CORBA and secure communication channels. In addition, it provides primitives to support context-awareness in the authentication process and the buildup of confidence depending on the successful authentication mechanisms. The GPAM allows authentication mechanisms to be deployed dynamically and stacked in various ways. The process of authenticating an entity in Gaia starts by initializing an *authentication session*. The control flow is depicted in Figure 9. The overall authentication process may involve authenticating using several different technologies. Each authentication technology is implemented as a GAMM module. Each GAMM module in turn (with its supported GPAMs) attempts to authenticate the entity. If an entity is authenticated successfully by more than one authentication module, a confidence value is calculated (based on the strength and the number of successful authentications). As depicted in the figure, the success of the *overall authentication* process is dependent on the individual authentication modules, the Active Space authentication policy, and the context of the space.

4.3.2 Gaia Authentication Mechanism Modules (GAMM)

The GAMM provides a generic interface that must be implemented for each authentication technology that needs to be deployed in the Active Space. Example authentication technologies that need to be implemented using GAMM include SESAME, Kerberos, RSA authentication, username-password, and fingerprint.

Multiple authentication technologies can be used at a time. Authentication technologies can be added dynamically under the proper context or when the necessary resources are available (e.g. the availability of a fingerprint scanner device). Additionally, the authentication modules are stackable in the sense that more than one module can be specified in the authentication policy file, they are called in order, and each one can add one or more principal objects and credentials, if successful.

GPAM can be configured to allow or deny logins based on which of the various authentication modules succeed. For instance, there can be a case where authentication is attempted using RSA or username-password, and if either one succeeds, the Gaia overall authentication succeeds. The administrator can customize these configurations through the authentication policy described in Section 4.3.3.

The GAMM interface supports a two-phase commit for authentication when using multiple authentication modules. The interface consists of five major methods: *setup()*, which sets up the module to be used to attempt an authentication. *Login()*, which attempts to verify the identity of the target entity according to particular mechanism being deployed. *Commit()* implements the second phase of the authentication process. In the 2nd phase, if the overall authentication is successful, *commit()* is called on all authentication modules. *Abort()* is called if the overall authentication has failed. The authentication module is expected to clean its state here. Finally, the *logout()* method logouts a subject manually, depending on how the authentication mechanisms works.

4.3.3 Authentication Policy

The framework provides system administrators with the ability to specify an authentication policy in XML format. The policy serves to identify which authentication modules should be called, in which order, and how their success or failure determines the ultimate success or failure of the overall authentication process. Multiple authentication policies can exist per Active Space. Each policy can be associated with one or more contexts, and the effective authentication policy is switched on-the-fly when the context of the space changes. The policy lists the allowable authentication modules in the space and allows specifying flags to the different authentication modules. These flags determine the role that these modules play in the overall authentication process. The flags have four possibilities: *necessary, optional, discard* and *sufficient.*

- Necessary – the authentication module must succeed for the overall authentication process to succeed. Even if it fails, however, the other authentication modules are queried for auditing purposes as well as for entity identification. In either case, the confidence value will be calculated depending on which modules succeed.
- Optional – the module success does not impact the overall authentication process. If the authentication policy does not include any necessary modules, then the overall authentication succeeds if at least one optional authentication succeeds. The optional flag is used by default if no flag is assigned to a specific authentication mechanism. This is because the optional setting provides the least obtrusiveness in cases where strong security is unnecessary.
- Discard – the result of the authentication using this module will be discarded and will not be used in calculating the confidence value for the entity. Note that in all the above flags, a successful authentication will result in calculating a net confidence value.
- Sufficient – if this module succeeds and *no necessary* module fails, the overall authentication succeeds. The confidence value up to the sufficient mechanism is calculated. No additional confidence values are calculated. This prevents excessive confidence if an entity manages to pass a large number of relatively weak authentication checks.

To summarize, the overall authenticationprocess succeeds if and only if:

$$\begin{array}{l} \forall_{m \in necessary\_set} succeed(m) \\ or \\ necessary\_set = \phi \wedge \begin{pmatrix} \exists_{m \in optional\_set} succeed(m) \vee \\ \exists_{m \in sufficient\_set} succeed(m) \end{pmatrix} \end{array}$$

Where *necessary_set, optional_set,* and *sufficient_set* correspond to the set of all necessary, optional, and sufficient modules respectively, and *succeed(m)* evaluates to true if authentication module *m* succeeds.

A sample policy is shown below. The outlined policy considers a fingerprint authentication sufficient for the overall authentication to succeed. The policy requests that the RFID badge not be used in the overall authentication or in calculating the net confidence value.

```xml
<?xml version="1.0" standalone="no"?>
<authentication_policy id="3105-1" space="3105" context="*">
<module>
 <name>fingerprint_FIU710</name>
 <flag>sufficient</flag>
</module>
<module>
 <name>RFIDBadge</name>
 <flag>discard</flag>
</module>
<module>
 <name>SmartWatch-ChalResp</name>
 <flag>optional</flag>
</module>
<module>
 <name>Ubisense_tag</name>
 <flag>optional</flag>
</module>
</authentication_policy>
```

Note that for each pair of GADM and GAMM only a single flag can be specified. Necessary modules have higher precedence over sufficient, optional, and discard. I.e., if a module is flagged as sufficient succeeds the system makes sure that no other necessary module have failed before deciding on whether the overall authentication is successful.

4.3.4 Callbacks

As described earlier, a GAMM implements an authentication mechanism. these modules can be reused for different devices. For instance, some Pocket PC devices have a built-in fingerprint scanner. The same device can be used with various authentication modules, including fingerprint verification, username-password, and RSA authentication. As shown in Figure 9, a GAMM needs to interact back and forth with the authentication device (this is also depicted in Figure 7 as the communication between the authenticator and the entity sponsor). The GADM interface implements a callback interface that allows authentication modules to retrieve credentials or capture distinguishing characteristics from the entity. For example, a callback can be used to retrieve the fingerprint template data securely from the PDA, or to get input from the user like username, password, or PIN. GPAM provides primitive callback functions for retrieving common authentication credentials, including encrypted passwords, PINs, and byte sequences (that can be used to represent fingerprint or iris templates). The exact callback implementation is specific to the device that is being used. A device may support multiple callback types, depending on its capabilities and the distinguishing characteristics it can capture. Figure 10 depicts how call backs are utilized.

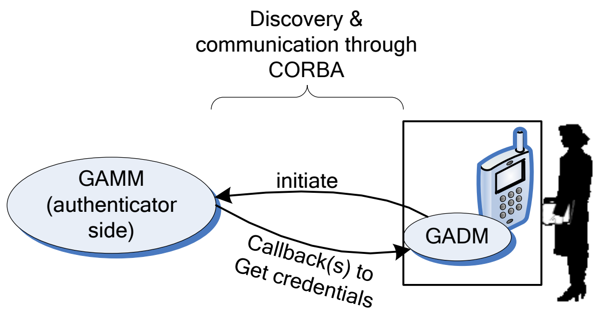

Discovery & communication through CORBA
GAMM (authenticator side)
initiate
Callback(s) to Get credentials
GADM

Figure 10: GPAM Callback Mechanism

4.3.5 Assigning Confidence to Authentication Outcome

In addition to the authentication policies for the authentication modules, the authentication framework of Gaia introduces policies for assigning confidence values to entities as they are authenticated, depending on the strength and number of succeeding authentication modules. These policies are written as rules in first order logic. There are two kinds of policies used here. One set of policies is used by the authentication server at the time of logon or authentication. These policies determine the confidence level of authentication. The other set contains access control policies, which determine whether a principal is allowed access to a particular resource.

To illustrate, the following is a simplified example of such policies. The various authentication devices are assigned confidence values, using the following rules:

```
ConfidenceLevel (smart_watch, 70%)
ConfidenceLevel (smart_badge, 10%)
ConfidenceLevel (fingerprint_scan, 90%)
…
```

These values are set by the system administrator based on the strength of the authentication device used and the protocol employed. If a principal *P* has been positively authenticated using its smart watch, say, then the authentication service inserts a new fact into the knowledge base:

```
Authenticated(P, smart_watch)
```

Similarly if the principal is authenticated using different forms:

```
Authenticated(P, password)
Authenticated(P, fingerprint)
```

We can define the confidence value (*V*) associated with an authenticated principal *P* as:

ConfidenceValue (P, V) :- $\exists_{device}$ X (Authenticated(P,X) $\wedge$ ConfidenceLevel (X, V))

Now, access control decisions can take the confidence information into account by defining rules like the following:

CanAccess (P, ColorPrinter) :- $\exists_{number}$ V (ConfidenceValue(P, V) $\wedge$ V>60%)

Here, *P* can only access the color printer if the authentication system has identified *P* with a confidence value of more than 60% (i.e. the principal has authenticated itself using at least one device whose confidence level is more that 60%). Note that in the example above, we do not calculate a net confidence value, but instead we grant access only if a user performed an authentication that grants her a confidence value of more than 60%. A more flexible way of doing this would permit us to combine multiple confidence levels and produce a net confidence value, i.e.:

CanAccess (P, ColorPrinter) :- $\exists_{number}$ V (NetConfidenceValue (P, V) $\wedge$ V>60%)

Representing system policies in first order predicate logic provides greater flexibility and dynamism while allowing rules to be evaluated efficiently.

Figure 11 contains a snapshot of the authentication policy that deals with authentication and the calculation of a net confidence value. The policy is written in Prolog syntax. The policy language is flexible enough to choose a function for combining confidence. In the simplified example shown in the figure, probability theory is employed to calculate a net confidence. I.e., if a principal receive confidence values of $V_1, V_2, .. V_n$ from different authentication methods, then the net confidence value V_{net} is calculated as follows: $V_{net} = 1 - (1\text{-}V_1)(1\text{-}V_2)\ldots(1\text{-}V_n)$, assuming that V_x are independent events. In the next section, I propose a more sophisticated and a smoother approach to reason about multiple authentications with multiple confidence values through the use of fuzzy logic rather than simplistic probability.

```
%policies related to the authentication service
clauses
  % confidence associated with authentication devices:
  %must be set by the admin based on the policy.
  confidenceLevel(“SmartWatch-ChalResp”,70).
  confidenceLevel(“SmartWatch-Passwd”,70).
  confidenceLevel(“SmartBadge”,10).
  confidenceLevel(“SmartBadgev2”,50).
  confidenceLevel(“Terminal-Passwd”,60).
  confidenceLevel(“FIU510_FingerprintScanner”,90).
  confidenceLevel(“USB_keychain”,60).
  confidenceLevel(“SpaceSelector-Passwd”, 75).
  ...
  %facts dynamically asserted by the authentication service:
  authenticated(“Bob”, “SmartBadge”).
  authenticated(“Alice”,”SmartWatch-ChalResp”).
  authenticated(“Alice”, “SmartBadge”).
  identified(“Charlie”).
  ...
predicates
  % Each principal P has a list of confidence values (CV),
  % one per authentication device he/she used
  confidence_value_list(P,CV) :-
    build_confidence_table(P, [], L).
  ...

  % building list of confidence values.
  build_confidence_table(P, L, L) :-
    not(authenticated(P,_)).
  build_confidence_table(P, L1, [E | L2]) :-
    authenticated(P, D), confidenceLevel(D, E),
    retract( authenticated(P,D) ),
    build_confidence_table(P, L1, L2).
  ...

  % calculate the net confidence per principal using
  % probability.
  netConfidenceValue (P, CV_NET) :-
    confidence_value_list(P, CV_LIST),
    calc_net_conf_prob (CV_LIST, TEMP),
    CV_NET = (1-TEMP)*100, !.
  calc_net_conf_prob ([], 1).
  calc_net_conf_prob ([CV_H | CV_REST], VALUE):-
    calc_net_conf_prob( CV_REST, TEMP),
    VALUE = (1-CV_H/100)*TEMP.
```

Figure 11: Portions of the security policy used in the Gaia testbed, written in Prolog syntax. This portion shows how confidence values are maintained and how the net confidence for a particular principal is combined. Note that some facts are asserted dynamically by the either the authentication service or the context infrastructure.

4.4 Use of Fuzzy Logic

Ubiquitous computing advocates transforming today's personal computers machines, which are "dumb," context-insensitive, and isolated, into intelligent, programmable, and context-aware clusters of machinery. However, unobtrusiveness and context awareness involve capturing and making sense of imprecise data. A salient feature of ubiquitous computing is that they interact heavily with the physical world. Physical worlds, however, are uncertain and imprecise. Ubiquitous computing environments must therefore take this uncertainty into account when dealing with the physical world. The security and authentication services in a ubiquitous computing environment are no exceptions. I argue that authentication should not be a black-and-white process, but instead, there is a need to have some room for heuristics and intuitive controls to make the environment more seamless, less obtrusive, and able to cope with uncertainty. When combining weak and strong identification and authentication mechanisms, there is a great demand to have a flexible system that can make decisions and take actions based on approximate reasoning by using fuzzy inference rules that take into account the context and the security policies of the environment.

Fuzzy Logic is a multi-valued logic that allows intermediate values to be defined between true and false. In effect, such a logic allows the use of linguistic variables used in approximate reasoning, for example, 'very high' or 'somewhat low.' In this manner, fuzzy logic is an alternative to traditional notions of set membership and Boolean logic.

4.4.1 Basic Fuzzy Definition

Let X be some set of objects, with elements x. Thus,

$X = \{x\}$.

A fuzzy set *A* in *X* is described by a membership function $\mu_A(x)$ that maps each point in *X* onto a real value in [0.0, 1.0]. As $\mu_A(x)$ approaches 1.0, the "degree of membership" of *x* in *A* increases.

A is empty iff for all x, $\mu_A(x) = 0.0$.

Equality between two fuzzy sets is defined as follows.

$A = B$ iff for all x: $\mu_A(x) = \mu_B(x)$ [or, $\mu_A = \mu_B$].

Complement of a fuzzy set *A'* is defined by the membership function:

$\mu_{A'} = 1 - \mu_A$

A is contained in B iff $\mu_A <= \mu_B$.

4.4.2 Basic Operations on Fuzzy Sets

Let *A* and *B* be fuzzy sets defined as shown in Figure 12.

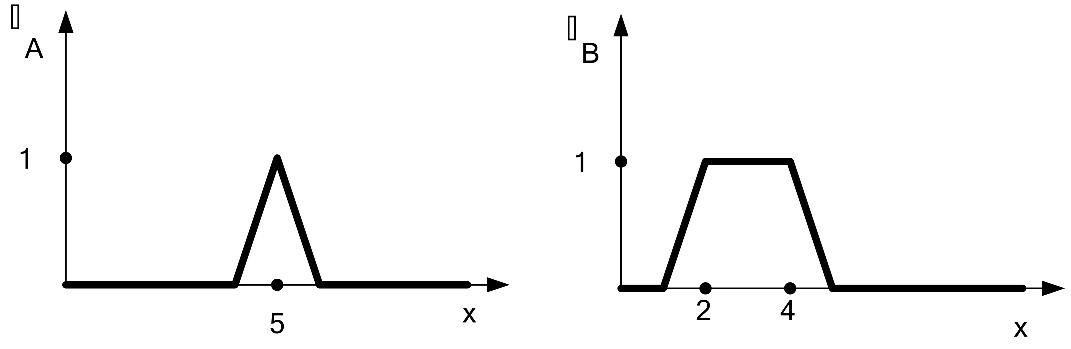


Figure 12: Sample Fuzzy Sets

The complement of a fuzzy set is defined by the membership function:

$\mu_C(x) = 1 - \mu_A(x)$

For example, the complement of μ_A is shown in Figure 13.

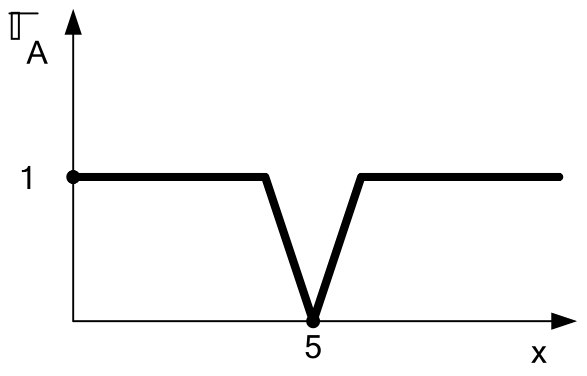


Figure 13: Fuzzy Complement Operation

The intersection of two fuzzy sets is defined by the membership function:

$\mu_i(x) = min(\mu_A(x),\ \mu_B(x))$. This is denoted as $A \cap B$, and illustrated in Figure 14.

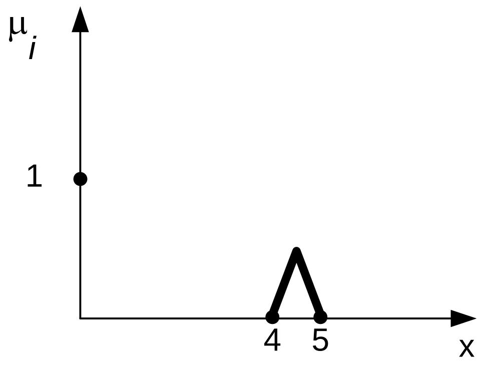


Figure 14: Intersection Operation on Fuzzy Sets

The union of two fuzzy sets is defined by the membership function:

$\mu_u(x) = max(\mu_A(x),\ \mu_B(x))$. This is denoted as $A \cup B$, and illustrated in Figure 15.

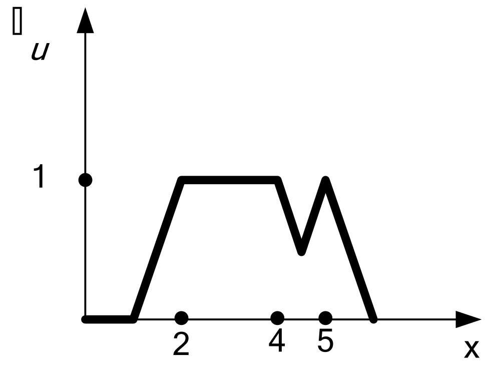


Figure 15: Union Operation on Fuzzy Sets

4.5 Inference Engine

The Inference Engine performs two kinds of tasks. First, it gives a level of confidence when a person authenticates himself. It makes use of the authentication policies as well as contextual information to assign the confidence level. Second, it evaluates queries from applications about whether a certain entity is allowed to access a certain resource. It makes use of application-specific access control policies, the credential of the entity, and contextual information to decide whether an entity has access to a resource.

The Inference Engine has access to all the authentication policies of the smart space and the access control policies of all the components in the smart space. It can also get context information from different context providers. It can either query various context providers or it can listen for events from context providers. It makes use of the Context Engine to look up various context providers. It can also get authentication information of various people in the space from the authentication service.

The authentication and access control policies are represented as first order expressions. The contextual information that the Inference Engine gets from context providers is also in the form of first order expressions. The Inference Engine evaluates queries in a way similar to how Prolog handles queries. It tries to resolve any query using the information it has about the policies and the context. Our current implementation has a very simple evaluation engine. It evaluates the query using the standard techniques of resolution and unification. If a unification that leads to all variables in the query being bound is obtained, then it returns the result to the application, else it returns nothing.

For example, a component that controls a wall display in a particular room has an access control policy that says that if there is a ubiquitous computing Seminar going on in the room, then the presenter has access to the display. The policy may look like the following.

$\forall_{People}$ *X Access(X, Display) :- SocialActivity(Room 3105, UbiComp Seminar)* $\wedge$ *IsPresenter(Ubicomp Seminar, X)*

So, when somebody (say "Bob") tries to access the display, the display component gets the credential of the person to see who it is. It then queries the inference engine to see if the person is allowed to use the display. This query would look like

?Access(Bob, Display)

To answer this query, the Inference Engine needs to know what the social activity in the room is. If it does not already know this information, it queries a context provider which knows about the social activity in the room. So, it sends a query to this context provider that looks like

?SocialActivity(Room 3105, UbiComp Seminar)

It gets back a reply of either "True" or "False."

If it gets a "True" reply, it asks about the presenter from a context provider that knows such information about the seminar. It then evaluates the rule (and any other access rules) to determine if Bob is to be given access to the display and sends this decision back to the display component.

Since a ubiquitous computing environment is very dynamic, the context of the environment changes very frequently. This affects any access control decisions that may have been made. For example, a person may have access to a certain device when there is a meeting going on in the room and he is the presenter, but not otherwise. So, if he is initially granted access to the device and later on, the activity in the space changes from “meeting” to “demo”, then he should no longer have access to the device. Applications can ask to be notified when changes in context of the space require changes in access control decisions.

In the example, described above, the display component would ask the Inference Engine to notify it whenever the following expression becomes true:

NOT Access(Bob, Display)

The Inference Engine in turn asks the social activity context provider to provide a notification when the condition NOT *SocialActivity(Room 3105, UbiComp Seminar)* becomes true. It also asks the PresentationManager Context Provider to provide a notification when the condition *NOT IsPresenter(Ubicomp Seminar, X)* becomes true. When the Inference Engine gets any such notification, it re-evaluates the rules; and if the expression *Access(Bob, Display)* no longer evaluates to true, it sends a notification to the display component.

An issue in logic programming is ensuring that the evaluation of queries can be terminated and is, hence, safe. In our system, the Inference Engine maintains only a finite set of sentences. Also quantification is done over finite sets. Thus, query evaluations will always terminate. More detailed analyses of these issues can be found in [102-104].

4.6 Fuzzy Inferencing

A shortcoming of using probability is that specific values need to be devised and given to the specific events in the system. Often, figuring out the exact value (or a very good estimate) is difficult, if not impossible. As a result, assigning the exact probability value for an event is left for the application and the scenario in hand. Fuzzy logic overcomes part of this problem by providing a set and a degree of membership in that particular set. Fuzzy logic is a key approach in situations where one wants to make a transition from a very precise but uncertain proposition to a less precise but strictly true proposition. For example, if it is not possible to construct a model to infer that the error in a face recognition device would be 3%, we may try to find a less precise version of the proposition, such that it can be concluded based on the evidence that the error in a face recognition device would be low with complete certainty.

In order to calculate net confidence when multiple authentication methods are used, without assigning precise probability values, fuzzy sets can be defined for the various possible confidence levels. I define a fuzzy set to designate the *degree of confidence* in authentication mechanisms. I assume the degree of confidence is represented in a scale from [0, 6], with the following categories: *{very low, low, medium, high, very high }*. This fuzzy set is depicted in Figure 16.

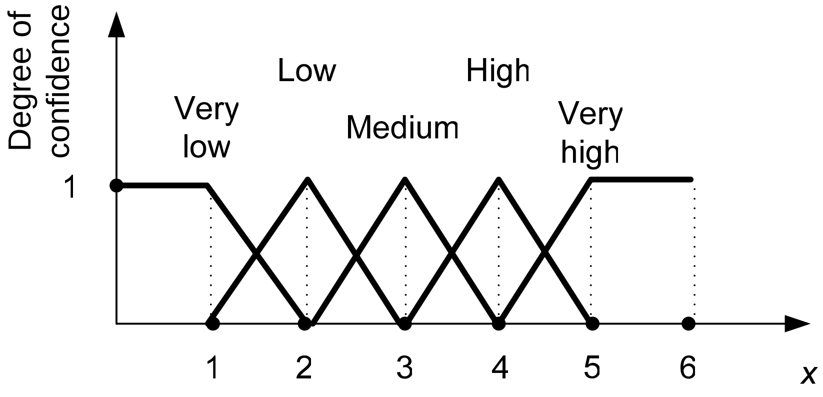


Figure 16: Degree of Confidence

To improve the Active Space ability to reason giving multiple authentication mechanisms and devices, I utilize fuzzy logic to provide a finer-grain representation of confidence in an identified or authenticated entity. The essential idea is to have an implementation of the approximate reasoning algorithm represented by fuzzy "if-then" rules, written in Prolog syntax. A single rule has the following form:

"*if X is C_i then Y is S_j*" where C_i and S_j are fuzzy sets over *X* and *Y* respectively.

Rules for building confidence based on devices and mechanisms used for authentication can be defined in the above format. Given *n* rules, R_1 to R_n, and an observation *A*, ideally we would like to conclude to a fuzzy set, and from there try to get a crisp value:

$(R_1, R_2, ..., R_n; A) \Rightarrow B$

Where *B* is the fuzzy set representing the conclusion.

I adopt a commonly used algorithm for fuzzy reasoning based on the rules given above. Given *n* rules in the above format, the algorithm is described as follows.

```
For i = 1 to n
    Let $m_i = \max_y \{ C_i(x) \cap A(x) \}$
    Let $S^A_i(y) = \min( m_i, S_i(y)\}$
Let $B(y) = 0$
For i = 1 to n
    $B(y) = B(y) \cup S^A_i(y)$
Let $b = \frac{\sum_{y_i \in Y} y_i B(y_i)}{\sum_{y_i \in Y} y_i}$
```

Where $\cap$ and $\cup$ represent fuzzy intersection and union operations respectively, $max_y\ (S)$ represents the highest y-coordinate value for set S, and $min\ (y_0, S)$ represents a subset of S consisting of all points whose y-coordinates are less than y_0. This will result in b which is the crisp conclusion, calculated by finding the center of gravity of the resulting fuzzy set. Figure 17 illustrates an example of this reasoning, where the membership functions of C_i and S_j are symmetrical and triangular for simplicity only. 'A' represents the observed fuzzy event. In the figure, $n = 3$ and the crisp value is obtained by calculating the center of gravity for the resulting fuzzy set.

Through this approach, it is possible to have flexible fuzzy rule specifications provided by administrators to specify how the system assigns confidence values to entities. The administrator in this case does not need to manually assign or guess exact values. For example, rules can be specified as follows.

```
R1: an RFID badge provides very weak authentication.

R2: Fingerprint device "FIU_700" provides somewhat strong authentication.

R3: If an entity was authenticated using a strong authentication mechanism
then assign it a high level of confidence.

R4: If an entity was authenticated using "something_it_is" and
"something_it_has" and "something_it_knows" then assign a very high level of
confidence
```

Where weak, somewhat strong, strong, and very high are linguistic values, which translate into fuzzy sets. Fuzzy reasoning can then take place in order to obtain a crisp value that represents the final confidence level.

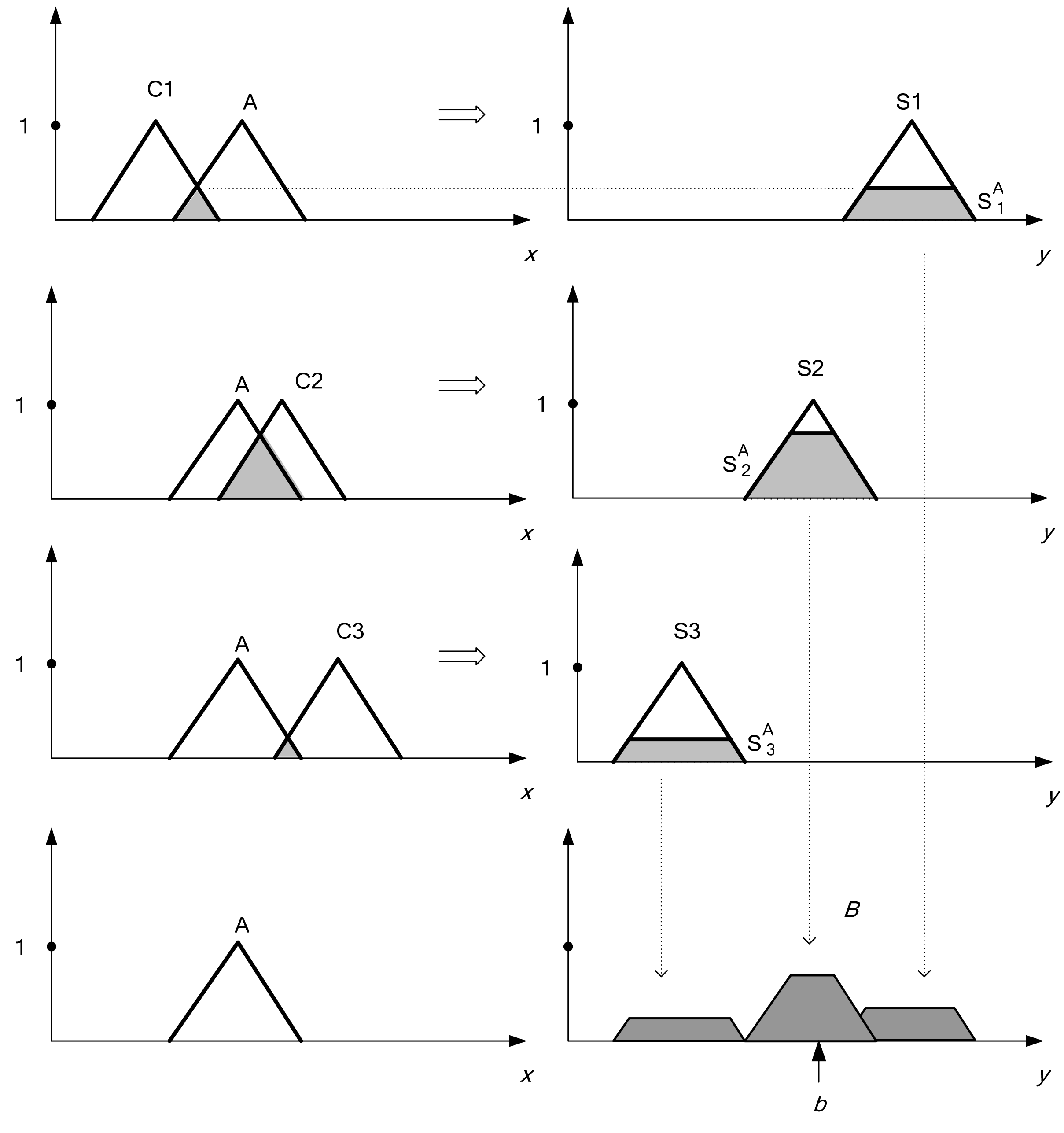


Figure 17: Fuzzy Reasoning Example

4.7 Implementation Details

GPAM and GAMM are implemented as CORBArized objects with a well-defined base interface. Developers can extend the interface with additional methods to support different callback functions or to present new capabilities for capturing different types of distinguishing characteristics. The modules themselves can be written in any programming language, as long as they are wrapped with a CORBA interface. Some lightweight devices that do not have native CORBA support employ proxies. Many CORBA implementations are heavyweight and require significant resources. To overcome this hurdle, I used the Universally Interoperable Core (UIC) on lightweight devices, which provides a lightweight, high-performance implementation of basic

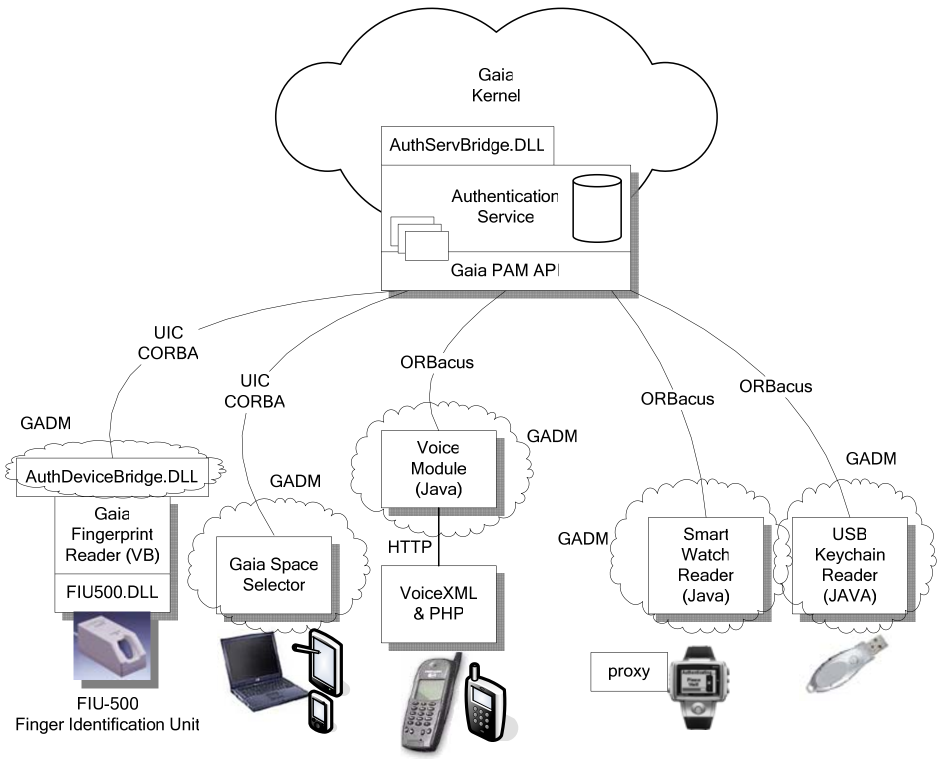


Figure 18: The existing Authentication Architecture in Gaia with all supported devices

CORBA services on mobile devices [105]. UIC is compatible with several mobile devices and PDAs.

The access control part of the security service provides an API, which ubiquitous applications and services can use to check whether principal *P* can perform a particular operation or not. The access control component forwards such inquiries to the inference engine. Depending on available context information and applicable security policies the inference engine replies with either ‘yes’ or ‘no.’ The access control component provides support for callbacks to the application, which can inform an application of possible context changes that may trigger a change in the access decision. A high-level UML diagram of the Gaia authentication framework is shown in Figure 19. The inference engine is implemented as a Gaia service and uses XSB Prolog [106]. XSB is a kind of Prolog which uses tabling and indexing to improve performance. Besides standard Prolog, XSB also allows programming in HiLog, a higher-order syntax that allows predicates to appear as arguments of other predicates. This allows unification to be performed on the predicate symbols themselves in addition to their arguments.

The implementation assumes a “close world” model to improve performance. This assumption is valid here because the identification and authentication decisions need to be taken based on the knowledge at hand only and nothing else. The inference engine deals with a finite set only, thus, query evaluation will always terminate. On a MS Windows™ XP™ machine with a 1.7 GHz processor and 1 GB of RAM, the inference engine was able to perform 32.1 MLIPS (million logic inferences per second), which is extremely fast for typical ubiquitous computing environments with hundreds of devices and sensors.

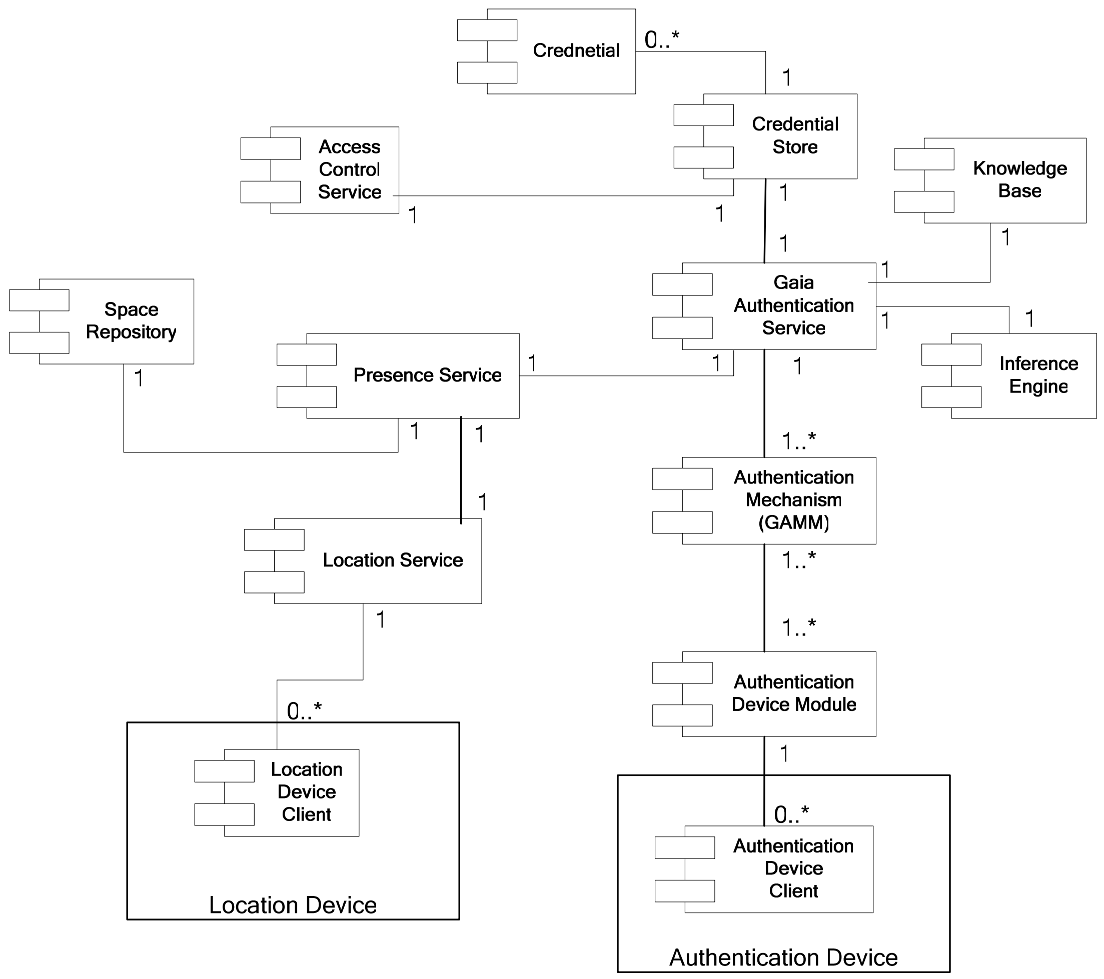


Figure 19: UML Diagram of Gaia's Authentication Framework

5. Privacy in Ubiquitous Computing Environments

After designing an elaborate ubiquitous authentication framework for Active Spaces, a new closely related challenge emerges. As the authentication framework facilitates proactive and automated identification and authentication, suitable privacy provisions must be introduced, otherwise, the authentication framework can become a ubiquitous "surveillance system" that violates basic privacy rights of users. For this reason, I argue that the development of privacy provisions has to go hand-in-hand with the development of automated identification and authentication. It is necessary to provide protocols for obfuscating the identity and/or location of the users, resources, and applications in the system. While privacy has always been lacking in information technology systems, in ubiquitous computing environments the situation is even worse. This is because much of the infrastructure is identifying. The implanted devices and sensors that attempt to make an Active Space sentient might be capturing too much information about the inhabitants. Moreover, ubiquitous environments introduce the problem of scale. Snippets of information that seems to be worthless on their own become much more sensitive when they are taken together and are correlated to the same person or activity.

5.1 Design Requirements

In this section I identify design guidelines that help establish a practical tradeoff between users' privacy on one hand, and context awareness and space automation on the other. These guidelines address issues of total anonymity, decoupling identity and location, and customizable privacy.

5.1.1 Total Anonymity

Total anonymity can be defined as the state of being unknown for an infinite amount of time. With respect to a ubiquitous computing environment, this means that neither a user's identity nor location can be inferred from context information. The user is totally anonymous and cannot be linked to any property or state. Total anonymity in a ubiquitous computing setting is not feasible because context information may reveal information about a user's identity, location, etc. Many scenarios, like classrooms and seminars, may require some level of identification, authentication, attendance recording, user-based customization, or authorization. Additionally, total anonymity could imperil the vision of the sentient, context-aware, Active Space. Furthermore, access control mechanisms and security audits become useless if total anonymity is permitted. For these reasons, it is crucial to develop mechanisms that strike a balance between preserving privacy and enabling some kind of value-added services for users based on context information. For ubiquitous computing environments *pseudonymity* represents a more practical level of privacy. Pseudonymity refers to assigning pseudonyms to entities that hide their identity or some of their properties (e.g. location, state) while still providing a way to link different actions or communication channels to the target entity. The pseudonym could be a unique nickname, ID, or even a role name. The pseudonym is temporary and valid only during a specific session.

5.1.2 Decoupling Identity from Location Privacy

Cooper et al. [107] identify three kinds of privacy: content, identity, and location. *Content privacy* is concerned with keeping data or content private. *Identity privacy* is concerned with hiding the identity of the user. *Location privacy* is concerned with hiding the location of the user. Content privacy relates to confidentiality and, in many cases, can be achieved through encryption. We propose that decoupling one's identity from one's location information provides the person with a better level of privacy. For example, if a service in the infrastructure knows that some user is located in room 3105, but without being able to positively identify the user. From the user perspective, this amount of information is less violating than positively identifying the user identity, location, and task all at once. This feature can be exploited to provide a configurable trade-off between location or identity privacy on one hand and security and value-added services on the other.

5.1.3 Customizable Level of Privacy

I believe that in ubiquitous computing environments, it is necessary to propose a system of relative privacy that allows a user to choose the level of desired privacy. The level of privacy is derived from the users' interactions with the ubiquitous environment, and the amount of personalization or value-added services that a user desires.

5.2 Approach

Two different models for preserving privacy can be identified. An *infrastructure-based model* for privacy preserving that assumes that a trustworthy infrastructure exists and users can set policies to identify who, what, when and under what context their location information is disclosed. To meet this requirement, it is possible to make use of the dynamic role-names and the context aware security policies supported by Gaia [108] to allow users, administrators, and service providers to define access control policies on how and when their location, situation, or identity information can be disclosed. This model is illustrated in Figure 20.

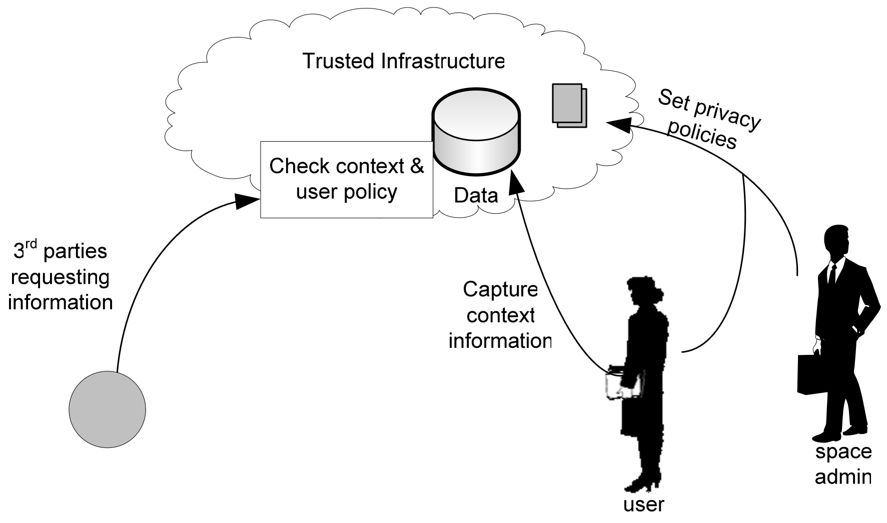

Trusted Infrastructure
Set privacy policies
Check context & user policy
Data
3rd parties requesting information
Capture context information
user
space admin

Figure 20: Infrastructure-based Model for Privacy

This centralized approach is relatively simple to implement and manage. Nevertheless, the assumption that a trusted infrastructure will take on the responsibility of protecting private information is inadequate in some settings. Furthermore, the infrastructure may be prone to insider attacks or to policy changes that could affect the privacy of users.

The other model for privacy preserving is an *ad hoc model*; where (1) the location of a user is separated from her or his identity to provide a higher degree of anonymity. In general, this approach attempts to eliminate the problem of scale introduced by ubiquitous computing. I.e., the information is distributed in the system, so that no single component has full knowledge of the entity, instead, different components only have pieces of information that are somewhat worthless on their own. (2) The location data is encrypted and sent to the user's personal device or a user agent.

In regards to decoupling sensitive information, I will focus mainly on separating identity from location, because they represent the most significant snippets on information on an entity in our Active Space scenarios. Nevertheless, the ideas stated here can be generalized for other pieces of information, e.g., action, intention, task, and role-name. Mist [109, 110] an overlay privacy preserving communication protocol can be used to implement this ad hoc model of privacy. The original Mist decouples location privacy and identity privacy in ubiquitous computing environments.

5.3 Mist Communication Overlay

The Mist overlay aim to design and implement a privacy protocol that allows users of a ubiquitous computing environment to roam and communicate freely while preserving their privacy. The privacy protocol prevents insiders, system administrators and even the system itself from tracking users and detecting their physical location. Yet, the system will enable users to communicate with other users and access computing resources in an authenticated manner without disclosing the users' physical locations or whereabouts. Further, users will be able to configure the level of privacy they wish to enjoy through the use of a user interface running on their mobile devices (e.g., mobile phone, laptop, PDA.) this is achieved by allowing the ubiquitous computing environment to maintain sensors that can detect the presence of users in a room, but without the ability to positively identify the users. Combined with our routing protocol, this creates a "mist" through which users can communicate privately. In Mist, we introduce a hierarchy of "*Mist Routers*" that perform "*handle-based routing*" to preserve privacy and hide information about the original source and the final destination. In short, we refer to this hierarchy as a "*Mist Hierarchy.*" The handle-based routing combines hop-to-hop routing based on handles with limited public-key cryptography to preserve privacy from eavesdroppers and traffic analyzers. Positive authentication and registration of users can be achieved at a higher level in the hierarchy, making it harder to infer the user's current location.

5.3.1 Overlay Design

In the Mist overlay, Mist Routers are deployed in a hierarchical fashion. The system administrator will be able to start CORBA Mist Routers on different machines. A System GUI will be able to detect all running Mist Routers, and display them graphically. The System GUI will provide the administrator with the ability to develop a hierarchy of Mist Routers and to have a "central control" to the entire Mist system. Users connect directly to one of the leaf level Mist Routers, which we call "Portals" (implemented as a subclass of a Mist Router). Through a Portal, a user (or the user's device) sets up a "Mist Circuit" upwards in the hierarchy. A Mist Circuit is a handle-based virtual circuit between the user and a special Mist Router, which we call a "Lighthouse." Since the handles for the virtual circuit is set up on a hop-by-hop basis, unless all the Mist Routers in the path collude, none of the intermediate Mist Routers can deduce the two ends of the virtual circuit. A user uses Mist Circuits to contact one of the higher level Mist Routers who is willing to serve as a contact point for that user. This contact point will only have partial information on how to route to that user. We refer to this contact point as a "Mist Lighthouse" for that user.

5.3.2 Mist Hierarchies

Mist Routers are key elements in the system. These CORBA objects conceal the identity and location of communicating parties by rerouting packets among themselves using hop-to-hop handle-based routing (which is described in more details later) We envision that Mist Routers will be deployed in hierarchical clusters organized along physical space divisions, called domains. The hierarchical organization of Mist Routers would enhance the system's flexibility and scalability, allowing it to be easily deployed over multiple domains.

Initially, a Mist Hierarchy needs to be agreed upon and constructed between the different physical space domains that are willing to cooperate and provide privacy for users roaming in them. Meeting this requirement should not be a problem; this is because most physical spaces are organized into hierarchies by nature (as illustrated in Figure 21, for example).

As illustrated in Figure 21, Mist Routers at the leaves of the hierarchy represent "Portals." Portals are viewed as the gateways that bridge the virtual world to the physical one. In other words, they are connection points where users of an active information space can connect to the system. Portals are represented by a variety of hardware that can include a fixed workstation, a sensor, an access point for wireless devices, and an RF transceiver.

As previously indicated, the original objective of an active information space is to allow seamless interactions between the various virtual and physical entities in the space. Therefore, there should be a mechanism over which these interactions can take place in spite of the existence of this mist that blurs the true identities of users and hide their physical locations. Therefore, to access the system, to communicate with others, and to use available resources while maintaining privacy, user Alice, say, has to register herself in the system as shown in Figure 22. The registration takes place through Alice's client device. The device talks directly to one of the available Portals in the surrounding physical space. The mechanism involves designating a special Mist Router for every user of the system. This special Mist Router will be referred to as a "Lighthouse" for that user. For example, a Lighthouse for Alice is a Mist Router that is an ancestor of the Portal that Alice is connecting to. Alice's Lighthouse will have knowledge of her true identity as well as partial knowledge on how to route to Alice. However, it does not know the exact physical location of Alice. Whereas the Portal knows the exact physical location of Alice, but does not "realize" that this is actually Alice and does not know who Alice’s Lighthouse is. Going back to the registration process illustrated in Figure 22, Alice’s device sends a registration request to the nearby Portal. The Portal will reply back with a list of its ancestral Mist Routers that exist at a higher level within the Mist Hierarchy and are willing to act as a Lighthouse for the user. A trusted third party can be used to vouch for the trustworthiness of some of these Mist Routers, particularly the ones that exist near the root of the hierarchy, since these Mist Routers can be accessible from different spaces. This vouching process is similar to how certificate authorities vouch for other parties on the Internet.

User Alice, through her client device, can customize the amount of privacy she wishes to enjoy by selecting a Mist Router at a suitable height in the hierarchy to be her Lighthouse. Selecting a Lighthouse is a tradeoff between performance and privacy. Choosing a Mist Router that is closer to the root of the hierarchy provides better privacy because less information is inferred about the actual physical location of Alice, and the extra rerouting provides better concealment. Whereas selecting Mist Routers closer to the Portal helps performance by limiting the number of reroutes but decreasing the level of privacy. To illustrate, in Figure 21, Alice decides to designate the Computer Science building’s Mist Router as her Lighthouse. This information implies that Alice is currently located somewhere in the Computer Science building. Bob, on the other hand, chooses the campus Mist Router as his Lighthouse. This implies that he physically can be anywhere in campus. Ultimate privacy can be achieved when a user chooses the hierarchy’s root as its Lighthouse.

Upon the selection of a suitable Lighthouse by Alice, we establish what we refer to as a “Mist Circuit” between Alice and the selected Mist Router. We discuss Mist Circuits in more detail in the Section 2.2. In any case, the Mist Circuit will make it possible for Alice’s

Lighthouse to authenticate Alice while hiding her exact physical location, and, at the same time, hiding her identity and her selected Lighthouse from the Portal she is connected to.

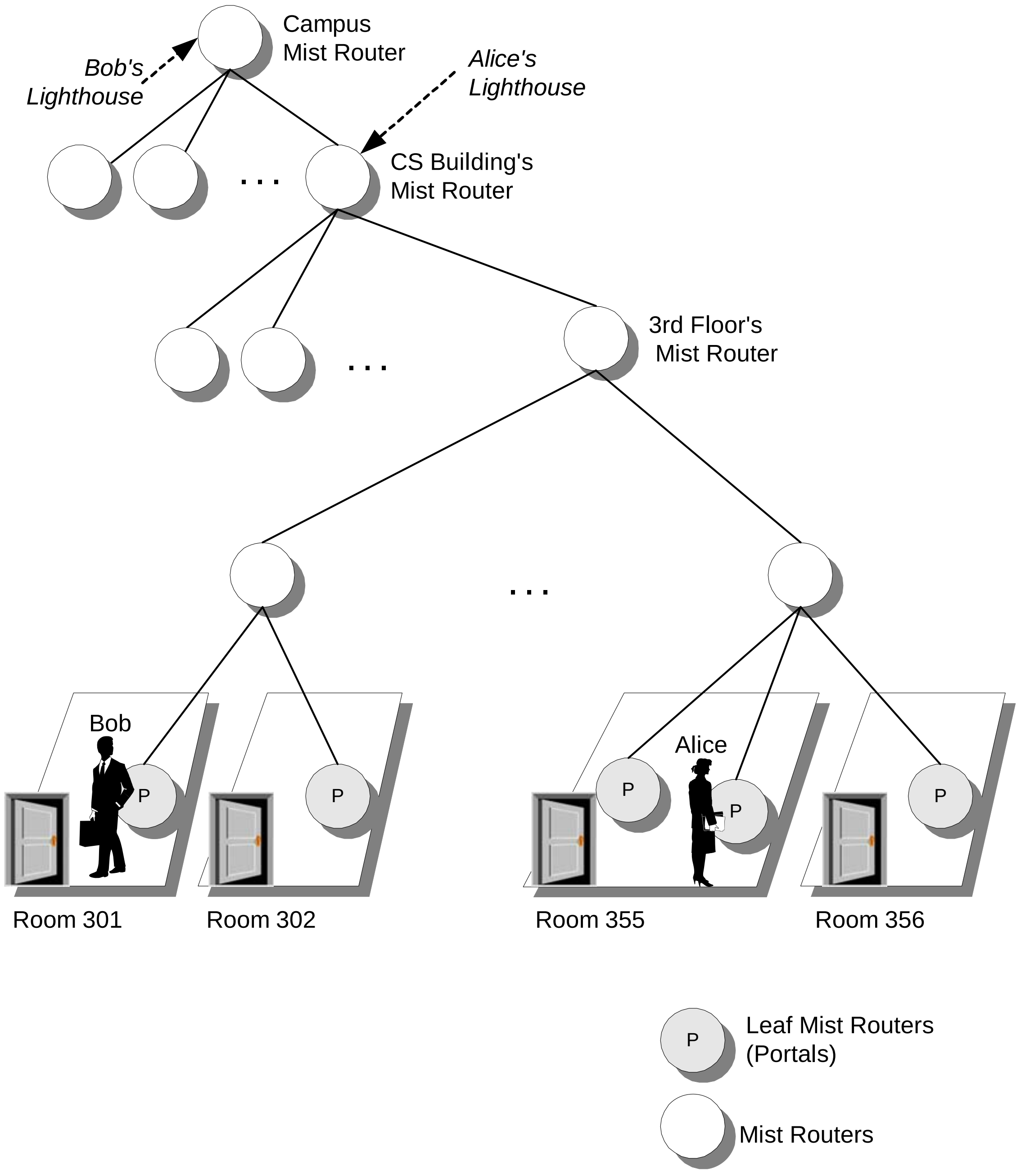


Figure 21: The Mist Hierarchy

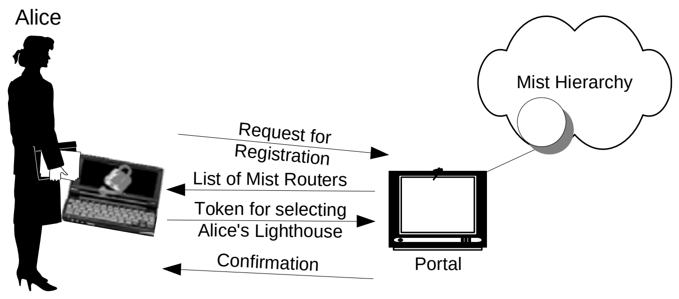


Figure 22: Registering in the System

5.3.3 Mist Circuits

Mist Circuits employ hop-to-hop, handle-based routing to send data packets back and forth between the source and destination through the mist. Combining this routing with limited public-key encryption allows data packets to be successfully routed through the mist while providing a higher degree of privacy and concealment. This prevents intermediate nodes from recognizing the identities of the actual endpoints or their physical location.

Recall that we establish a Mist Circuit between the user and its selected Lighthouse so that the user can reveal its true identity and authenticate it at the Lighthouse without disclosing physical location information. In this section we describe how a Mist Circuit is setup and used.

Going back to the example of Alice registering in the system. Her Portal fulfills her request for registration by replying back with a list of ancestral Mist Routers that are willing to act as Lighthouses. The list returned contains two pieces of information for each Mist Router. Each entry will contain an ID that uniquely identifies the Mist Router and a digital certificate for that Mist Router. The digital certificate can be issued by some trusted third party. The certificate could contain information about the how "high" in the Mist Hierarchy the associated Mist Router is. In other words, the list is of the form:

<Mist Router $_1$, Certificate $_1$>,

<Mist Router $_2$, Certificate $_2$>,

…

User Alice selects a suitable Mist Router, which she does not disclose to the Portal. To establish a Mist Circuit, Alice generates a Mist Circuit establishment packet. The general format of Mist packets are illustrated in Figure 23. The ‘Handle ID’ field represents a handle that is unique per Mist Router that helps identify the next hop on the packet’s route. A value of 0 in this field indicates that no value is assigned yet. How the handle is used is described later in this section. The ‘direction’ field is a single bit that specifies whether the packet is going upwards (toward the Lighthouse) or downwards (toward the Portal) in the hierarchy. The ‘packet type’ identifies the type of the packet, which tells the intermediate Mist Routers how they should handle the packet.

Assuming that Alice selects the Mist Router ‘Z’ in Figure 26 as her Lighthouse, then Alice’s Mist Circuit establishment packet will contain ‘0’ for the handle ID and ‘U’ in the direction field, indicating that this packet is going upwards. The type field will contain a value indicating that this is a Mist Circuit establishment packet. The payload will consist of the Message *M*:

$M = E_{public\_key\_Z} (Alice \mid\mid TS \mid\mid K_{session} \mid\mid TKN \mid\mid PP)$

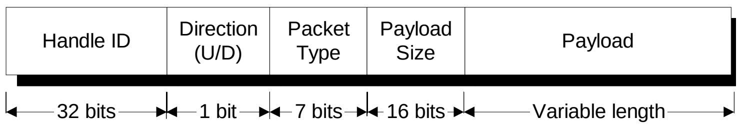
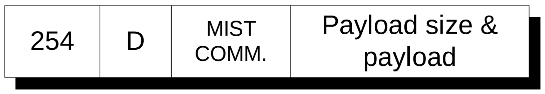
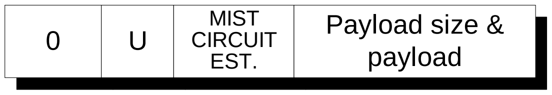


Figure 23: General Format for Mist Packets



$$Payload = E_{K_{session}}(\text{"Success"}, TS_2)$$

Figure 24: Registration Confirmation Packet



$$M = E_{public\ key\ of\ Z}(Alice \,||\, TS \,||\, K_{session} \,||\, TKN \,||\, PP)$$

$$Payload = M \,||\, S_{Alice}(M)$$

Figure 25: Alice's Mist Circuit Establishment Packet

Where:

|| stands for concatenation.

Alice: Alice’s unique ID in the active information space

TS: A timestamp to prevent replay attacks.

$K_{session}$: A random session key to encrypt further communication between the user and her or his Lighthouse. It is also used to add some additional randomness into the encrypted message.

TKN: A token to be presented to the user’s lookup service. Details about the user’s lookup service and the contents of this token are given in Section 2.3.

E_k: Means encrypt using the key ‘*k*’.

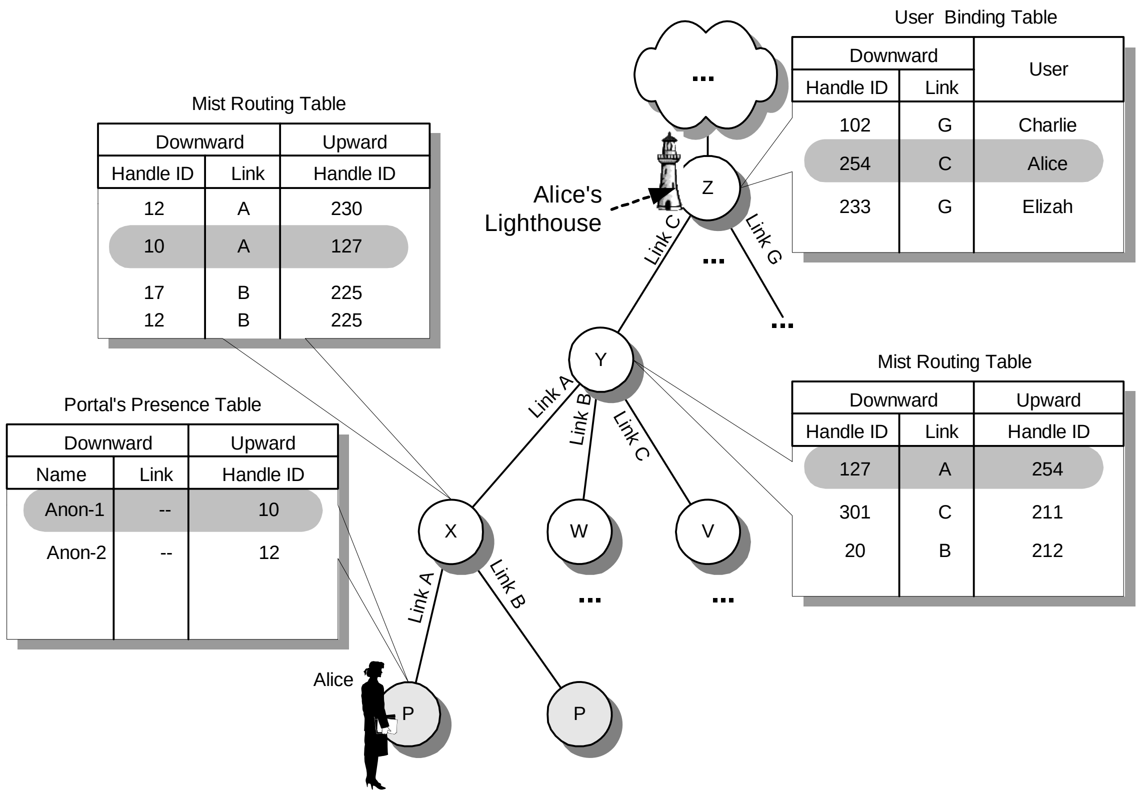
PP: A predetermined “fixed” phrase. In current implementation, the string “Mist Circuit Establishment Message” is used. The significance of this will be described below.

The actual payload is:

$Payload = M \,||\, S_{Alice}(M)$,

where $S_{Alice}(M)$ indicates Alice’s digital signature over M.

The contents of the Mist Circuit establishment packet are shown in Figure 25. Alice then transmits this packet to her Portal, without informing the Portal of the selected Lighthouse. Portals will maintain a table that is referred to as the “Presence Table.” Since the Portal detects nearby people without positively identifying them, whenever a new person is detected, he or she is entered into the Portal’s presence table as an “anonymous” person. Additionally, the Portal assigns for every user a handle ID that is unique within that table only. So in the scenario depicted in Figure 26, Alice is represented as “Anon-1” and is assigned a handle ID of 10, say. If other users exist in the same physical space and the Portal is able to communicate with them, then similarly, they will be entered into the presence table. The “link” field should contain a value that identifies the network link or port number over which the Portal can communicate



Mist Routing Table

Downward		Upward
Handle ID	Link	Handle ID
12	A	230
10	A	127
17	B	225
12	B	225

Portal's Presence Table

Downward		Upward
Name	Link	Handle ID
Anon-1	--	10
Anon-2	--	12

User Binding Table

Downward		User
Handle ID	Link	
102	G	Charlie
254	C	Alice
233	G	Elizah

Mist Routing Table

Downward		Upward
Handle ID	Link	Handle ID
127	A	254
301	C	211
20	B	212

Figure 26: Mist Circuit Setup

with the corresponding user. We assume that if a Portal supports communication with more than one physically present user, then it should be able to recognize which user sent a particular packet. Upon receipt of the Mist Circuit establishment packet from Alice, the Portal will replace the value in the packet's handle ID field with the handle ID that was assigned to Alice in the presence table, which is 10 in the example shown. Next the Portal will transmit the modified packet "upward" to its parent Mist Router.

From now on, upon receiving the circuit establishment packet every intermediate Mist Router will attempt to decrypt the encrypted portion of the payload using its private key. If the decryption fails, (the predetermined phrase can be used to indicate whether or not the decryption failed) then the Mist Router will infer that this packet is not meant for it. Instead, the packet has to be passed upward to its parent. Each Mist Router will maintain a "Mist Routing Table." This table will associate handle IDs used over downward connections with handle IDs that will be used on the upward connection. Note that within the downward column of the Mist Routing Table, the combination of Handle ID and link ID is unique per Mist Router, whereas, within the upward column the handle ID value is unique per Mist Router. The current Mist Router does a quick lookup on its Mist Routing Table to see if it has an entry for the handle ID and the link over which it received the packet. If it does not, it creates one, and associates an upward handle ID for it. The Mist Router then substitutes the value of the packet's handle ID with the newly assigned value and passes the message to its parent. The process is repeated for every intermediate Mist Router.

On the other hand, if a Mist Router successfully decrypts the encrypted portion using its private key, then this indicates that the user actually chose the current Mist Router as his or her Lighthouse. All Mist Routers that are willing to act as Lighthouses for users should maintain a

‘User Binding Table’ as shown in Figure 26. The Mist Router can now authenticate the user by verifying his or her signature and checking the freshness of the timestamp. The handle ID and the downward link above which it was used will be stored in the User Binding Table, along with the actual ID of the user.

Figure 26 shows the actual entries in the presence, routing and binding tables when user Alice registers and chooses ‘Z’ as her Lighthouse. The shaded entries in the figure represent Alice’s entries. In effect, this process has established a “circuit” over which Alice can communicate with her Lighthouse securely. Note that while Alice’s Lighthouse can infer that Alice exists somewhere in the hierarchy underneath Mist Router ‘Y’, the exact location cannot be determined unless enough Mist Routers agree to cooperate. Therefore, the longer the path between Alice and her Lighthouse the more “private” her location becomes.

To complete the Mist Circuit establishment, the Lighthouse confirms the registration of Alice by sending back a reply packet. The format of this reply is shown in Figure 24. For the example shown, the handle ID will be set to 254, because this is the value bound to Alice. The packet should be sent downward (D). The packet type is set to “MIST COMMUNICATION” which indicates that intermediate Mist Routers should not attempt to decrypt the contents, rather, they should just route it to the next hop.

$K_{session}$ is the session key between the Mist Router ‘Z’ and Alice that was transmitted through the Mist Circuit establishment packet. Note that to improve performance from this point on, we use symmetric encryption to achieve confidentiality between the user and the chosen Lighthouse. TS_2 is a timestamp to prevent replays. This packet can now be routed back to Alice in a manner similar to what was described above. Now Alice can communicate securely with her Lighthouse while preserving her privacy. In the next section, we describe how other entities in the ubiquitous computing environment can communicate with Alice.

5.3.4 Locating Users

Once the Mist Circuit-Setup has been completed, the Lighthouse Mist Router acts on behalf of the end-user. All communication with the user will take place through its Lighthouse, since only the Lighthouse knows how to route packets to the user. However, we first need to locate the current Lighthouse for a particular user. Only then can one communicate with the user. Locating users involves the *registration* of <user, Lighthouse> pairs, and the *lookup* of <user, Lighthouse> pairs.

5.3.5 LDAP Servers

RFC 1777 describes the Lightweight Directory Access Protocol (LDAP). In essence, users can register with LDAP servers, which can consequently be looked up with a subset of these attributes. Mist users will have a unique LDAP *Distinguished Name (DN)*. Mist users can look up information about other Mist users either based on their DN's, or on their attributes. For example, one could look up a user based on the last name and university, "Doe from University of Illinois." Once a user has been located, the attribute corresponding to the current Lighthouse can be retrieved.

5.3.6 Security issues

We would like to prevent malicious Lighthouses or attackers from falsely registering users with them. To achieve this, the user constructs a special token (*TKN*) signed by the user's private key. This token will contain a timestamp and the unique ID of the chose Lighthouse. This token is propagated to the Lighthouse during the Mist Circuit setup as described in Section 2.2. Once the Mist Circuit has been established, the Lighthouse presents this token to the LDAP Server. The updates will be secure, and cannot be forged or replayed by an attacker. If the timestamp has already been seen before, or if it has expired, the token will be discarded. Naturally, if the signature cannot be verified, the token is also discarded. The format of this token, *TKN*, is as follows:

$TKN = (User\ ID \ || \ Lighthouse\ ID \ || \ Timestamp \ || \ S_{User}(User\ ID \ || \ Lighthouse\ ID \ || \ timestamp)\ )$

This tells us that *TKN* contains the user ID, the Lighthouse ID (this could be the DNS name) and the timestamp are signed by the user's private key. *TKN* contents do not need to be encrypted because the contents are already known by the Lighthouse anyway. Hence, only integrity of this message, not confidentiality, needs to be guaranteed.

5.3.7 Mist Communication Setup

Once the Lighthouse for a particular user is located, we need to set up a communication channel through it. We assume that both users in the communication setup have established their own Mist Circuits and are both registered with their respective Lighthouses. Communication will now take place through the two Lighthouses. We use the notation Lighthouse_X to mean "Lighthouse of User *X*." Let us say that Bob is trying to initiate communication with Alice. Bob and Alice are registered with $\text{Lighthouse}_{\text{Bob}}$ and $\text{Lighthouse}_{\text{Alice}}$ respectively.

Bob generates the following message for its Lighthouse:

$M_{Lighthouse} = E_{Ksession}(COMM\_SETUP \ || \ Alice\text{'s } ID \ or \ attributes \ || \ TS)$

Note that all messages in this section are actually the payload of Mist Communication packets. Since handles have been set up in both directions during the Mist Circuit Setup phase, this message will travel up to Lighthouse$_{\text{Bob}}$. Note that intermediate Mist Routers are never aware of the user's Lighthouse. When the message arrives at Lighthouse$_{\text{Bob}}$ it is able to uniquely determine that the message is from Bob based on the arriving handle. It decrypts the message with session key $K_{Session}$ and determines from the *COMM_SETUP* message type that communication must be set up with Alice. If Alice's ID is included then the lookup for Lighthouse$_{\text{Alice}}$ is straightforward. However, if Bob specifies attributes, then Lighthouse$_{\text{Bob}}$ must perform a lookup based on these attributes. If a unique match for Alice is found based on these attributes, Lighthouse$_{\text{Bob}}$ can determine Alice's ID. In both cases, Alice's ID is used to lookup Lighthouse$_{\text{Alice}}$. The timestamp *TS* is used to prevent replay attacks.

Lighthouse$_{\text{Bob}}$ uses asymmetric key encryption with Lighthouse$_{\text{Alice}}$ to determine Lighthouse$_{\text{Alice}}$'s handle for Alice. Since this is straightforward, we avoid the details of this communication. We will call this the destination handle for Alice, or dest_handle$_{\text{Alice}}$. In Figure 27, we can see that Lighthouse$_{\text{Bob}}$ determines dest_handle$_{\text{Alice}}$ = 254-C. Lighthouse$_{\text{Bob}}$ then generates a unique handle that Bob can use to address Alice. We will call this handle src_handle$_{\text{Alice}}$. In Figure 27 src_handle$_{\text{Alice}}$ = 689. Lighthouse$_{\text{Bob}}$ sets up a binding of the form <src_handle$_{\text{Alice}}$, dest_handle$_{\text{Alice}}$, Lighthouse$_{\text{Alice}}$>. In Figure 27 we can see the binding <689, 254-C, Y>. We call this a Mist Communication Binding. All messages from Bob that arrive for src_handle$_{\text{Alice}}$ (689) will be tunneled to Lighthouse$_{\text{Alice}}$ (Y) and indexed with dest_handle$_{\text{Alice}}$ (254-C). Similarly, Lighthouse$_{\text{Bob}}$ will supply the handle for Bob to Lighthouse$_{\text{Alice}}$ that will set up a binding of the form < src_handle$_{\text{Bob}}$, dest_handle$_{\text{Bob}}$, Lighthouse$_{\text{Bob}}$> in the same way. In Figure 27 we can see this binding as <412, 100-A, X>.

Once Lighthouse$_{\text{Bob}}$ and Lighthouse$_{\text{Alice}}$ have setup their bindings, they need to inform Bob and Alice of the src_handles. Lighthouse$_{\text{Bob}}$ sends src_handle$_{\text{Alice}}$ to Bob in the following message:

$M_{Handle} = E_{Ksession}\ (HANDLE\_MSG\ ||\ Alice\text{'}s\ ID\ ||\ src\_handle_{Alice}\ ||\ TS)$

In Figure 27 this message corresponds to "For Alice use 689." Similarly, Lighthouse$_{\text{Alice}}$ sends src_handle$_{\text{Bob}}$ to Alice.

Now Bob can send Lighthouse$_{\text{Bob}}$ messages destined to Alice by simply using src_handle$_{\text{Alice}}$ (689), and Alice can send Lighthouse$_{\text{Alice}}$ messages destined for Bob using src_handle$_{\text{Bob}}$ (412). This is done to hide Alice's identity from intermediate routers. These intermediate routers are hence unaware of *both* the endpoints of the communication. To communicate with Alice, Bob constructs messages of the following form, where 'M' is the message for Alice:

$M_{For\_Alice} = (COMMUNICATION\_MSG\ ||\ src\_handle_{Alice}\ ||\ M)$

This message will propagate upstream until it reaches Lighthouse$_{\text{Bob}}$, which uses src_handle$_{\text{Alice}}$ (689) to determine Lighthouse$_{\text{Alice}}$ (Y) and dest_handle$_{\text{Alice}}$ (254-C). Note that the Message passes in the clear, and the use of handles does not disclose the endpoints of the communication. Alice and Bob are now free to choose an end-to-end encryption scheme if desired. Using this method, there is no duplication of encryption by the Mist. Once Lighthouse$_{\text{Alice}}$ is determined, the Message M needs to be forwarded to Lighthouse$_{\text{Alice}}$. Lighthouse$_{\text{Bob}}$ sends the following message to Lighthouse$_{\text{Alice}}$. We use the subscript of "crossing" to suggest that the message is crossing over from one Lighthouse to another.

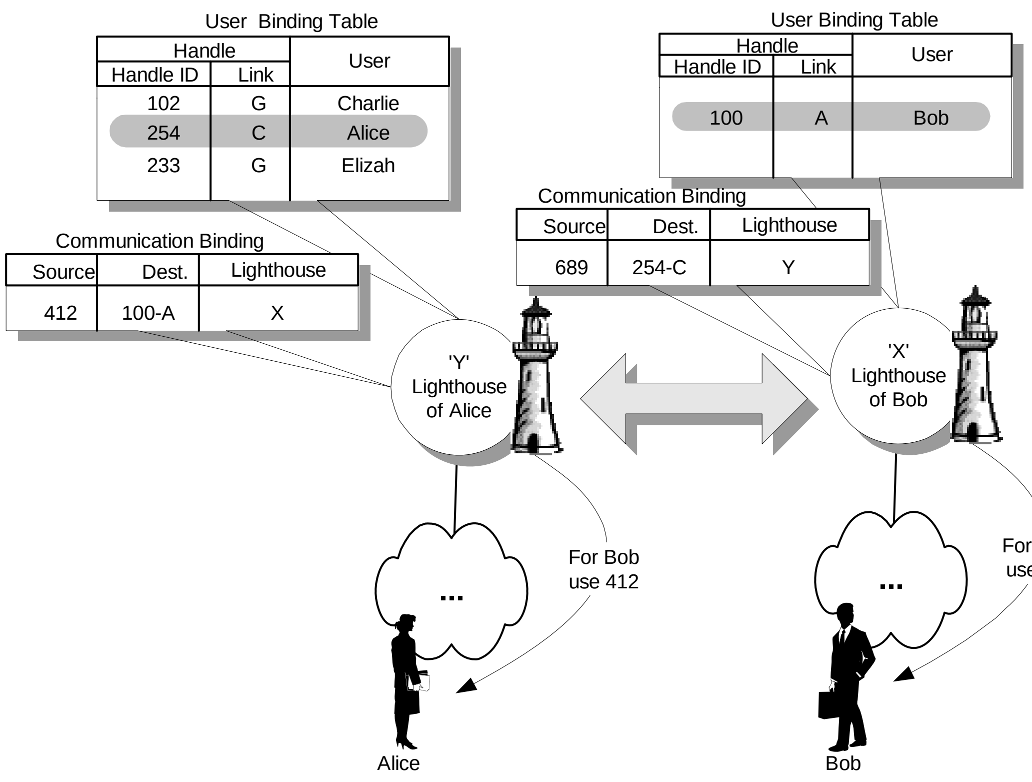


Figure 27: Mist Communication Setup

$M_{Crossing}$ = *(dest_handle$_{Alice}$, M)*, e.g., (254-C, M)

When Lighthouse$_{Alice}$ receives this message, it uses this dest_handle$_{Alice}$ to route message M to Alice. Similarly, Lighthouse$_{Alice}$ can route messages to Bob using:

$M_{Crossing}$ = *(dest_handle$_{Bob}$, M)*, e.g., (100-A, M)

Note that these "crossing" messages between Lighthouses are not the Mist communication messages described before. The Lighthouses use their own packet formats to exchange the crossing messages.

5.3.8 Discussion

Note that Lighthouse$_{Bob}$ and Lighthouse$_{Alice}$ are aware of the identities of the endpoints of the communication, but they are not aware of Alice and Bob's locations. Hence the privacy of Alice and Bob is preserved. In addition, all intermediate routers are unaware of the endpoints of the communication, and hence cannot deduce the locations of Alice and Bob. In fact Alice and Bob can communicate anonymously with respect to all other routers, with the exception of the two Lighthouses. With respect to this communication, the Lighthouses are trusted entities, and hence fully anonymous connections are not provided. In what have been described so far, we achieve the goal of preserving Alice and Bob's location privacy from all intermediate routers, including the Lighthouses. The most important thing to note is that Alice cannot deduce Bob's location, and Bob cannot deduce Alice's location. Hence communication between Alice and Bob is privacy preserving.

Note that if all the Mist Routers and Lighthouses along the path collude, then the locations of the end points can be determined. The system distributes the trust, and assumes that such routers span various domains, and collusion between such entities is not feasible. This form of privacy is stronger than that provided by a single trusted entity, and we argue that distributing trust is the best that one can do. Beyond that, if all routers collude, we cannot trust anybody in the system and privacy cannot be achieved.

5.4 Incorporating Mist into the Gaia Infrastructure

To enable users to access privileged services that require some level of authentication while preserving their location and identity privacy, I extend the Mist protocol and integrate it into Gaia's authentication framework. By doing this, the authentication framework can now utilize the privacy decoupling that Mist provides. The protocol is illustrated in Figure 28. Within every Active Space, the authenticator components of the authentication framework are considered to be "*Space Authentication Portals*" (SAPs), which corresponds to portals in the original Mist protocol. However, they are special types of Portals that can be located at the entrance of an Active Space, or other convenient places. The SAP will feature a collection of wireless and

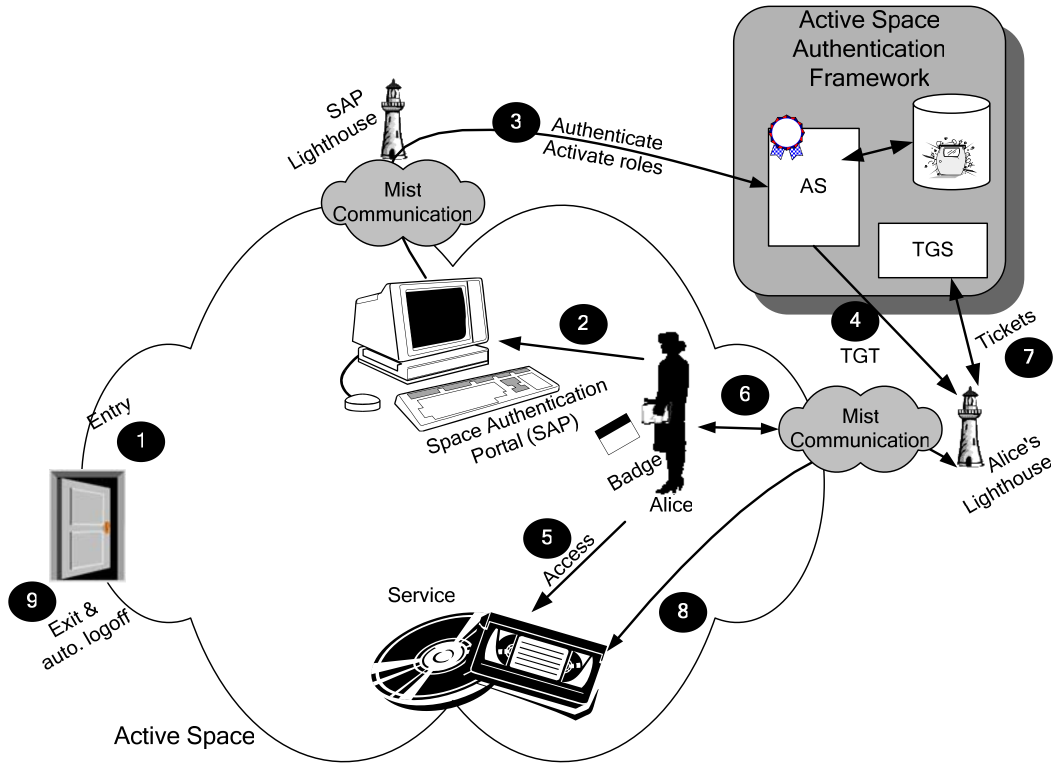


Figure 28: Integrating Mist with the Gaia Authentication Framework

wired base stations and device readers that enable users to authenticate with the Active Space using any authentication devices they are carrying or wearing.

An Active Space security service exists for every *Active Domain*. An Active Domain is a collection of Active Spaces, and the interconnecting networks, which are managed by a single administrative authority. These domains resemble Kerberos “Realms.” Like Kerberos, the security service consists of three components. The first component is the AS (Authentication Server), which provides a single sign-on point for the Active Domain, using any devices and gadgets the user currently possesses. The TGS (Ticket Granting Server) issues “tickets” that can be used by the user to access available services in that space. These tickets are signed, as a protection against tampering, using the private key of the TGS. Finally, a database is maintained that contains necessary information for the authentication of all users within the Active Domain, as well as their privileges and security attributes.

Users enter an Active Space (step 1 in Figure 3). To gain access to privileged services, users can authenticate through the *space authentication portals* (SAP) (Step 2). To achieve privacy, the SAP itself does not have sufficient information to authenticate users. However, it has a Lighthouse through which it can communicate with the security service (step 3). Mist communication is used here to prevent the security service from pinpointing the exact physical location of the authenticated user. Through its Lighthouse, the SAP contacts the security service with a set of authentication requests, each representing a different authentication device. Upon successful authentication, the AS, like Kerberos, issues a ticket granting ticket (TGT) for that user (step 4). Recall that in Mist, every user has a Lighthouse that stores his relevant information. The TGT issued for a user can be stored on his personal device (if available) or can be stored in his Lighthouse. The TGT in this system is a cryptographic data structure that contains one or more roles that the user can utilize to access a certain service (step 5). Upon accessing a secure service, the user requests a ticket to access this service, by going through the Mist to his Lighthouse. In step 7, using the TGT stored at the Lighthouse for the target user, the Lighthouse can communicate with the TGS requesting tickets to access the required service. The TGS issues the necessary cryptographic tickets. These tickets do not contain any references to the real name or identity of the owner; they just incorporate an unforgeable pseudonym or role-name. Further, these tickets contain a role name for the user that allows him to access the service without revealing his exact identity. Using the information in these tickets, the service can make a decision whether to authorize the user or not (step 8).

To illustrate how information is distributed to prevent a single component from capturing too much information about an entity, consider the simplified example depicted in Figure 29. In this scenario Alice needs access to the printing service. After successful authentication through the

Gaia authentication framework, she is presented with a credential that acts a TGT. The credential does not need to include absolute identification information for Alice. Instead, a role-name or an attribute digitally signed by the authentication suffices. When Alice wishes to access the printer service, she presents the TGT to the Printing TGS. To preserve privacy, printer access policies should be expressed in role-names or properties instead of exact user names. An RBAC-based model like the one used in Gaia [108] is ideal. Alice presents her TGT to the printing service TGS. Based on the attributes and/or role-names stored in Alice's TGT, the printing service decides whether to allow printing access to Alice. Since the TGT does not contain the exact identity of Alice, the printing service TGS only knows partial information on Alice. If the access to the printer is granted, Alice gets a TGS that allows her access to the printer. Table 1 shows how the information is decoupled, and what type of information is revealed to each entity in Figure 29.

Table 1: Information Revealed

Service	***Information Revealed***	
	Identity	***Location***
Authentication Service	All	None
Printing service TGS	Role granularity	Service granularity
Printer	None	Fine-grained

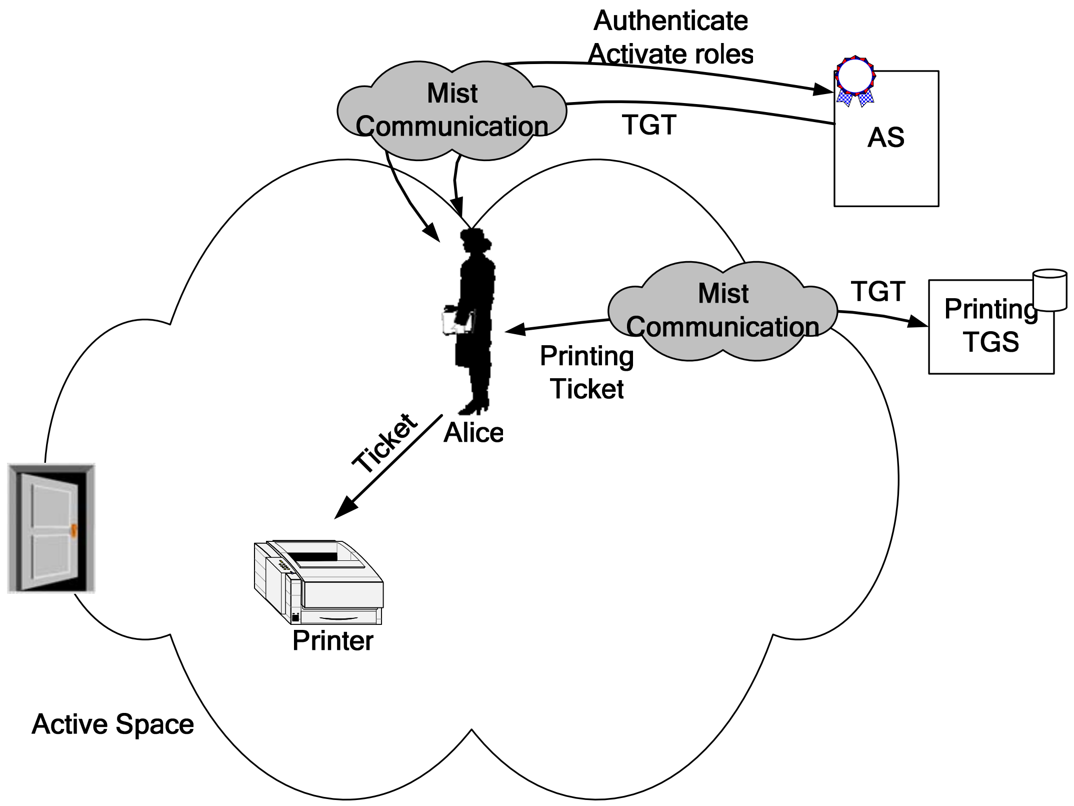


Figure 29: Sample Scenario for Printer Access

5.5 Implementation

Here I present more technical details on the design and implementation of Mist and its integration with the Gaia authentication framework.

5.5.1 UML Use Cases

Figure 30 shows the high-level use case diagram. In essence, the Mist system can be actively used by two actors. The first is a generic Gaia entity that wishes to use Mist privacy-preserving communication channels. Typically, this would be end users or services. The other actor is the system administrator who administers the system, or the Gaia communication subsystem which manages the communication hierarchy in Gaia.

Figure 31 shows the administrator's use cases. The administrator can:

1. Create or launch as many Mist Routers as he or she wants. As the number of Mist Routers increase, users' privacy increases.
2. Develop Mist Hierarchies by using the System GUI's drag and drop features to link Mist Routers to each other and build a hierarchy. Links can be added, updated and removed as necessary.
3. Monitor the Mist traffic and the communication channels established between users.

Figure 32 illustrates the possible use cases for general entities using the Mist protocol:

Step 1: User Alice wants to use the Mist system. The first step is to connect to the nearest Portal.

Step 2: Alice needs to register with a Lighthouse, so the Portal presents Alice with a list of Lighthouses. Since Alice does not want to reveal her identity to any Mist Routers other than her Lighthouse, Alice builds a Mist Circuit upwards to her selected Lighthouse.

Step 3: Now, another user, Bob, wants to talk to Alice. Bob follows the above steps to connect to a Portal and register with a Lighthouse.

Step 4: Bob "informs" his Lighthouse that he wants to talk to Alice.

Step 5: Bob's Lighthouse locates user Alice, maintains a communication binding entry in its internal tables, and provides Bob with a handle through which he can communicate with Alice. Similarly, Alice's Lighthouse sets up handles to Bob.

Step 6: Using the handle provided, Bob can communicate securely (if he chooses) and privately with Alice.

Step 7: When the communication is over, Bob or Alice can deregister from the Lighthouse and disconnect from the Portal.

Furthermore, users may query the certificate authority (CA) to obtain public keys of the users they are trying to communicate with.

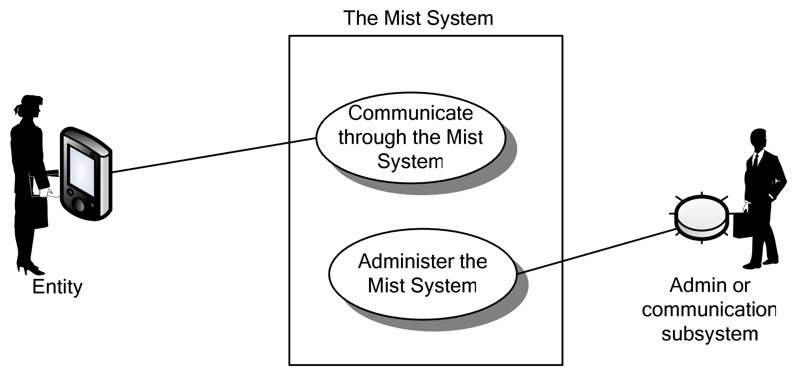

The Mist System
Communicate through the Mist System
Administer the Mist System
Entity
Admin or communication subsystem

Figure 30: High-Level Use Case Diagram

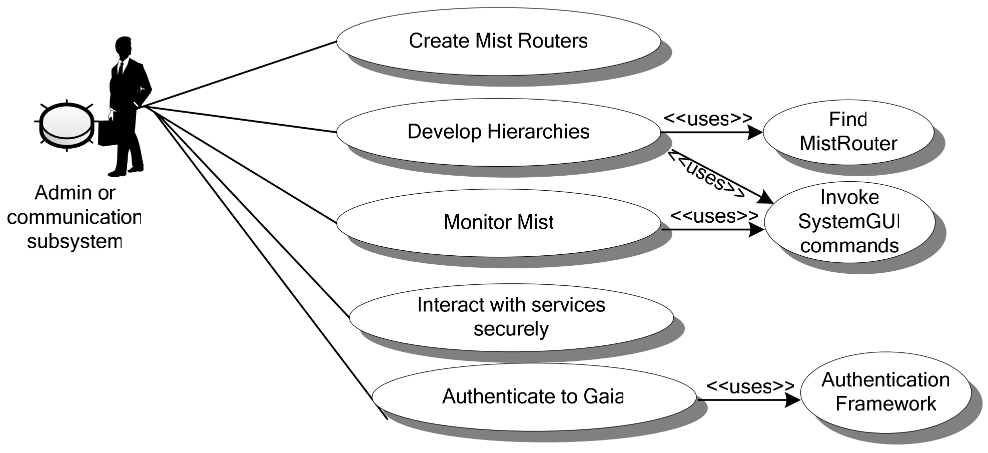


Figure 31: Administrator Use Case Diagram

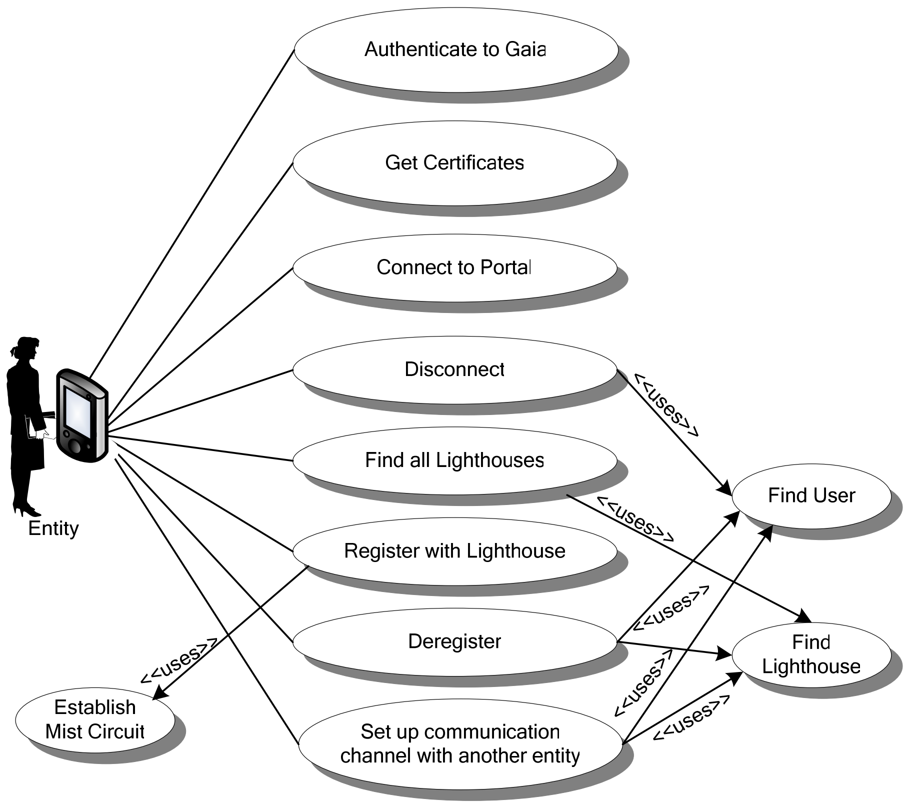


Figure 32: Mist Use Case Diagram

5.5.2 Class Hierarchies

The class hierarchies are explained briefly in this section. The different components in mist are implemented in Java and CORBArized to facilitate discovery and communication with other Gaia components and entities. Figure 33 and Figure 34 below show Mist's class diagrams. Only the essential methods are shown. The small arrow beside the labels on connectors indicates the direction over which the label applies. For example, in Figure 33 a Lighthouse "has" a MistCommBindingTable. Note that boxes with bold borders represent CORBA objects, whereas boxes with light borders represent regular classes. Brief class descriptions follow.

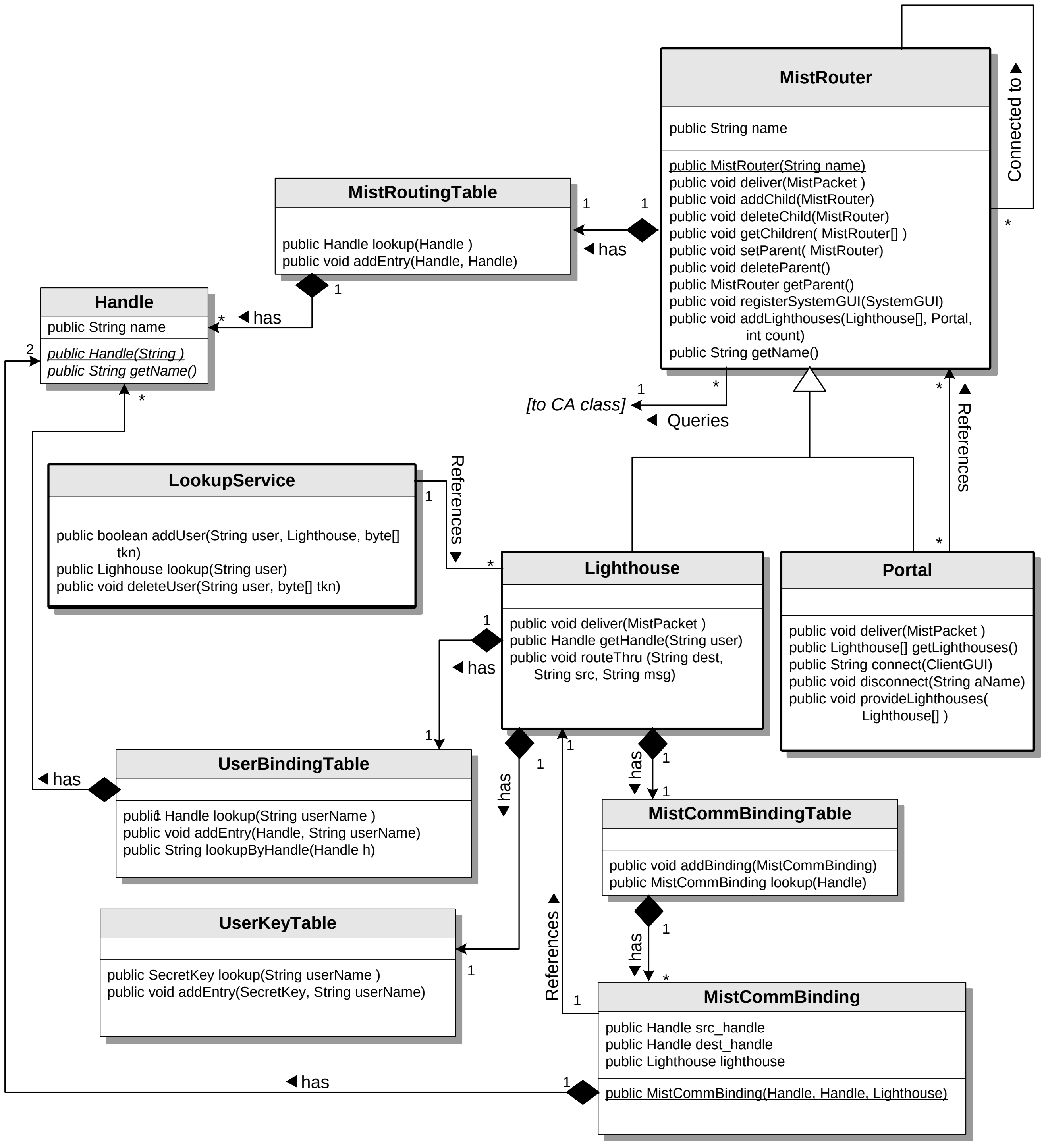

MistRouter
public String name
public MistRouter(String name)
public void deliver(MistPacket)
public void addChild(MistRouter)
public void deleteChild(MistRouter)
public void getChildren(MistRouter[])
public void setParent(MistRouter)
public void deleteParent()
public MistRouter getParent()
public void registerSystemGUI(SystemGUI)
public void addLighthouses(Lighthouse[], Portal, int count)
public String getName()
Connected to
MistRoutingTable
public Handle lookup(Handle)
public void addEntry(Handle, Handle)
has
Handle
public String name
public Handle(String)
public String getName()
has
[to CA class]
Queries
References
LookupService
public boolean addUser(String user, Lighthouse, byte[] tkn)
public Lighhouse lookup(String user)
public void deleteUser(String user, byte[] tkn)
References
Lighthouse
public void deliver(MistPacket)
public Handle getHandle(String user)
public void routeThru (String dest, String src, String msg)
Portal
public void deliver(MistPacket)
public Lighthouse[] getLighthouses()
public String connect(ClientGUI)
public void disconnect(String aName)
public void provideLighthouses(Lighthouse[])
has
UserBindingTable
public Handle lookup(String userName)
public void addEntry(Handle, String userName)
public String lookupByHandle(Handle h)
has
has
has
MistCommBindingTable
public void addBinding(MistCommBinding)
public MistCommBinding lookup(Handle)
UserKeyTable
public SecretKey lookup(String userName)
public void addEntry(SecretKey, String userName)
References
has
MistCommBinding
public Handle src_handle
public Handle dest_handle
public Lighthouse lighthouse
public MistCommBinding(Handle, Handle, Lighthouse)
has

Figure 33: Mist Class Hierarchies - Part 1

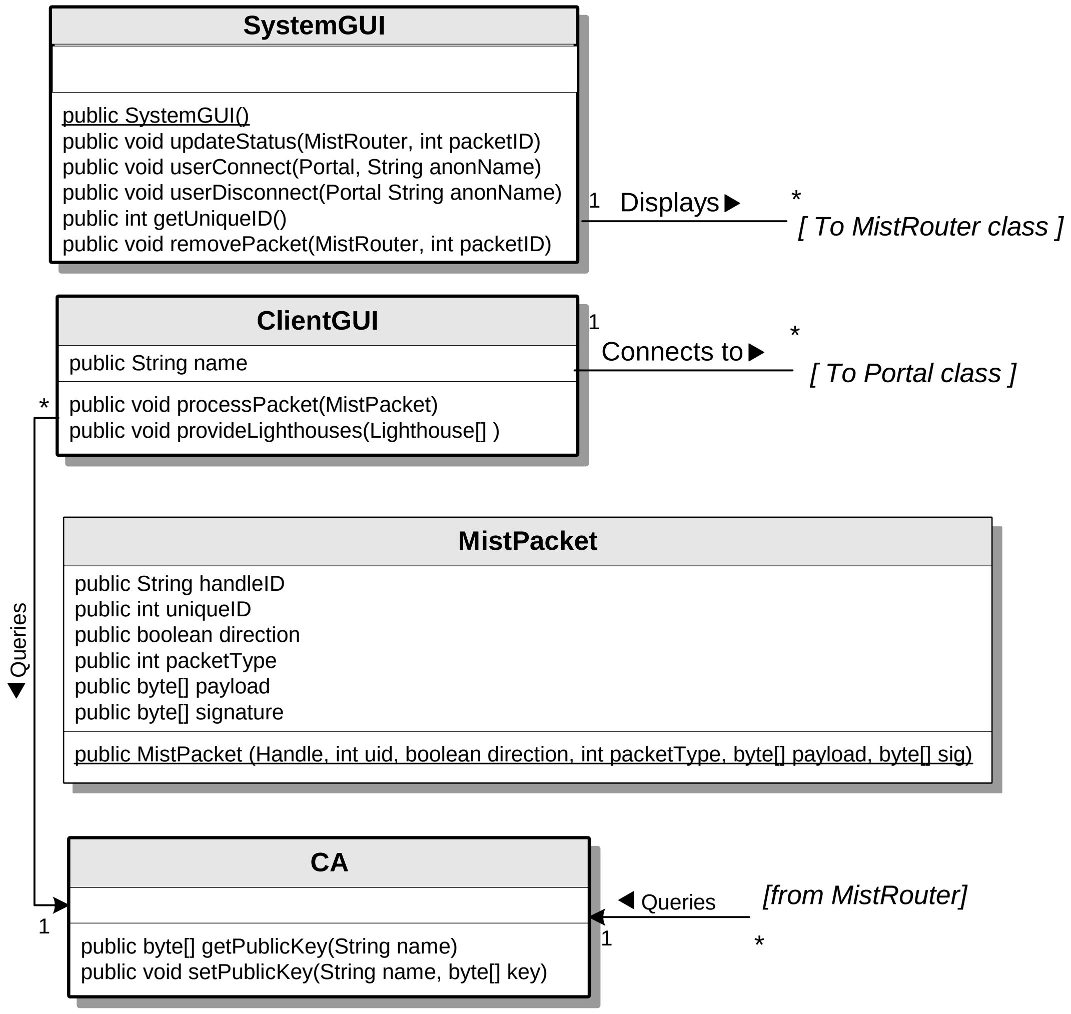

SystemGUI
public SystemGUI()
public void updateStatus(MistRouter, int packetID)
public void userConnect(Portal, String anonName)
public void userDisconnect(Portal String anonName)
public int getUniqueID()
public void removePacket(MistRouter, int packetID)
1
Displays ▶
*
[To MistRouter class]
ClientGUI
public String name
public void processPacket(MistPacket)
public void provideLighthouses(Lighthouse[])
1
Connects to ▶
*
[To Portal class]
*
◀ Queries
MistPacket
public String handleID
public int uniqueID
public boolean direction
public int packetType
public byte[] payload
public byte[] signature
public MistPacket (Handle, int uid, boolean direction, int packetType, byte[] payload, byte[] sig)
CA
public byte[] getPublicKey(String name)
public void setPublicKey(String name, byte[] key)
1
◀ Queries
[from MistRouter]
1
*

Figure 34: Mist Class Hierarchies - Part II

MistRouter Class. This class represents a Mist Router. A Mist Router has a unique name, and can be connected to other Mist Routers in a hierarchical fashion. The "deliver" method enables the delivery or routing of packets between endpoints.

Portal and Lighthouse Classes. These are special types of Mist Routers; therefore, they are subclasses of the MistRouter class. A Portal has additional functionality to allow users to connect to and disconnect from it, as well as the ability to discover ancestral Lighthouses. The Lighthouse allows users to register with itself.

Handle Class. The Handle class represents a generic “handle.” Handles are used to conceal information about the endpoints of a communication. They are instrumental in establishing Mist Circuits and enabling hop-to-hop handle-based routing.

MistRoutingTable Class. This class represents Mist Routing Tables. As described in Section 2, all Mist Routers will have a Mist Routing Table that associates incoming handles with outgoing handles. These tables are used to perform hop-to-hop routing to help conceal the identities of the endpoints.

UserBindingTable. a Lighthouse maintains a User Binding Table. This table stores the handle corresponding to users registered at this Lighthouse.

MistCommBindingTable Class. A Lighthouse maintains a Communication Binding table. This table keeps track of other users whom the registered users in this Lighthouse are communicating with. Each entry in this table consists of a “communication binding,” which is a tuple consisting of three elements: the source and destination handles for the target user, and the Lighthouse with which that user is registered.

MistCommBinding Class. This class represents a tuple within the Mist Communication Binding table.

LookupService Class. This represents the user lookup service, which is a wrapper for the LDAP service. The Lookup service will be queried by the Lighthouse during user registration, deregistration, and when locating other users’ Lighthouses. The token is a cryptographic structure generated by the user and can’t be fabricated. This token serves to “prove” that the lookup registration is actually initiated by the user and not fabricated by some Lighthouse.

ClientGUI Class. This represents the Client GUI, which will be running on the user's client. This allows the user to register/deregister with the Mist and to communicate with other users.

SystemGUI Class. This class is used for demonstration purposes. It shows an overall view of the Mist system allowing the Mist administrator to see how messages are routed through the Mist. It also enables the administrator to develop Mist Hierarchies.

MistPacket Class. This represents a Mist packet. Packets are routed hop-to-hop from one Mist Router to another.

CA Class. This class represents a Certificate Authority, which maintains the public keys of all entities in the Mist system. These include users and Mist Routers. The CA can be queried by the Client GUI or Mist Routers.

5.6 Performance Analysis

Figure 35 shows the setup time required to setup a communication channel between two entities in Gaia for 3 to 10 Mist Routers. The numbers are based on the average of 10 runs. All the Mist Routers are running on 4 different Intel Pentium-4 1.7 GHz PCs running MS Windows™ XP. Figure 36 shows the approximate roundtrip time for sending packets over Mist channels for 3 to 10 Mist Routers. Note that the packets are encrypted hot-to-hop. The measurements are based on a reference implementation that has room for optimization.

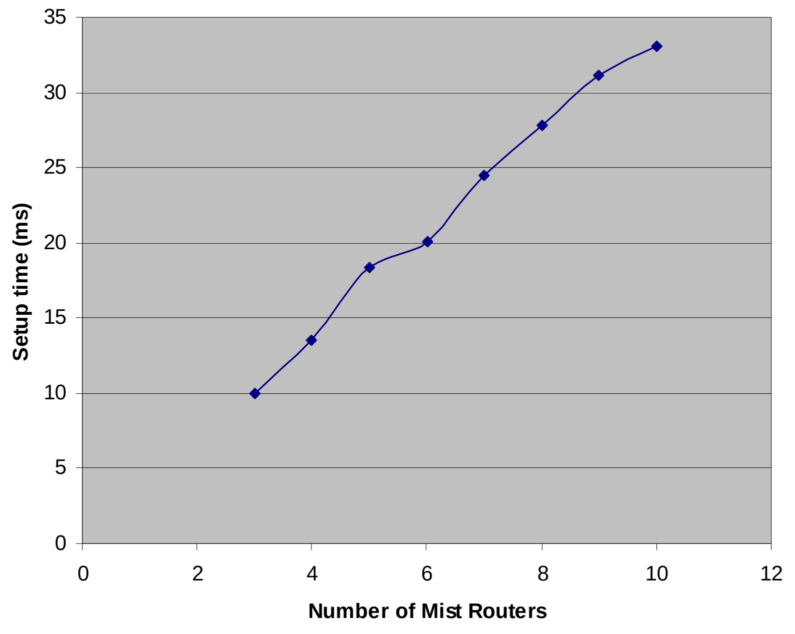


Figure 35: Time required to setup a Communication Channel

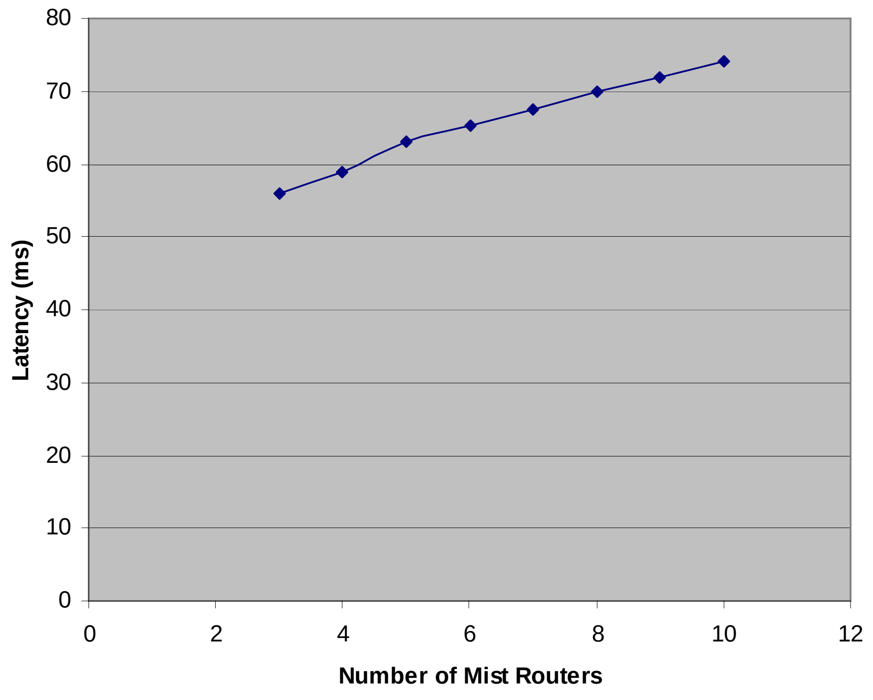


Figure 36: Round Trip times for encrypted Mist Communication Channels

6. Evaluations

Ubiquitous computing challenges conventional means of interacting and using computers. New schemes for assessing and evaluating ubiquitous computing are therefore required to guide the design and implementation of such systems and their components. In this section, I identify a number of metrics for the purpose of evaluating the security and authentication aspects of a ubiquitous computing environment. I use these metrics to perform usability studies and compare the framework with other approaches. In this section, I do not address metrics for measuring cryptographic algorithms' strengths and resilience to attacks because the authentication mechanisms employed by the proposed framework is based on algorithms and mechanisms established and evaluated in literature and deployed widely in different scenarios. Furthermore, many research efforts already address the measuring of cryptographic strengths and resilience of these protocols in details [111].

6.1 Evaluation Metrics

The security metrics I introduce here are relevant to the special needs of security and authentication in ubiquitous computing environments. These unique metrics are essential for guiding the design and implementation of security in such systems. While there are many evaluation tools that allow designers and developers to compare frameworks over a set of domain-specific quality features, however, the features emphasized in ubiquitous computing applications and components, such as context-awareness and unobtrusiveness, differ greatly from traditional monolithic applications and frameworks, making existing evaluation tools inapplicable or inadequate [112]. In essence, there is a need for metrics that can capture the ability of the security services to handle the ubiquity, context awareness, and rapid evolvement of the surrounding environment. I identify three main metrics for evaluating the security features specific to ubiquitous computing. These main metrics are: (1) the expressiveness of security policies, (2) user control over private information, and (3) unobtrusiveness of security mechanisms. Theses metrics are explained in detail in the following subsections. It should be noted however, that these metrics are not intended to be comprehensive, but rather complement traditional system and usability metrics to create a stronger and more extensive assessment of ubiquitous computing authentication and privacy components.

6.1.1 Expressiveness of Security Policies

A security policy is a set of rules that guide the implementation of security in a system to match the requirements of the system. In this setting, the expressiveness of a security policy can be measured by its ability to incorporate the following in the policy's rules.

1. *Support for mandatory and discretionary rules*. Typical pervasive computing environments are composed of a tapestry of public spaces, devices, and resources, as well as personal devices and gadgets. Therefore, it is essential to be able to support mandatory policies set by the space administrators, as well as accommodating policies defined by users for their personal devices.
2. C*ontext sensitivity.* Security rules of a pervasive computing environment may vary according to the context of the space. Hence, the security policy language should be able to incorporate rich context information.
3. *Uncertainty handling.* Often, context information is not precise. Policies should be expressive enough to define how to act under imprecise or incomplete context information.

4. *Conflict resolution.* Expressive policies have the potential to conflict with each other, particularly when different users are allowed to set policies. Some mechanism for handling conflicts is necessary.

6.1.2 User Control over Private Information

The physical outreach of ubiquitous computing makes preserving users' privacy a difficult task. Mechanisms are needed to give users control over their private information and how and when it can be disclosed. Cooper et al. [107] identify three kinds of privacy: content, identity, and location. *Content privacy* is concerned with keeping data or content private. *Identity privacy* is concerned with hiding the identity of the user. *Location privacy* is concerned with hiding the location of the user. My proposed metric takes into account these three different kinds of privacy (as illustrated in Table 2).

6.1.3 Unobtrusiveness of Security Mechanisms

Pervasive computing attempts to provide a seamless user-centric environment, where users no longer need to exert much of their attention to computing machinery. Therefore, the security subsystem should provide mechanisms that allow security services, like authentication for instance, to become transparent to some level, blending into the background without distracting users too much. In these environments, the traditional interaction metaphor of one user to one computer is broken. Users are now interacting with multiple technologies while simultaneously collaborating with other users. They are moving about the environment, constantly refocusing attention while manipulating and relocating data across devices. New interaction metaphors present new challenges for measuring the usability of the system. Through observations of users collaborating in the Active Space and through interactions with the HCI group, the following metrics for evaluating the usability and unobtrusiveness of ubiquitous environments are proposed [113, 114].

1. *User head turns*: head turns can be used to indicate how much a user's attention is divided across the space. Changing the focus of attention can be physically and mentally taxing and can be an indication of high obtrusiveness, particularly, when the head movement is in response to an alert or a message taking place in the space. The number of head turns can be measured easily and can help assess the unobtrusiveness and seamlessness of the environment.
2. *Physical movement*: physical movement that is not a part of a user's task is time consuming and causes interruption in users' thought process. Therefore, excessive physical movement can be used to indicate high obtrusiveness that impairs the everywhere, anytime access to resources which ubiquitous computing advocates.
3. *Keystrokes, clicks, actions, and other atomic input:* the number of clicks, keystrokes, input actions (e.g., placing a finger on a fingerprint reader) and other atomic input operations that are auxiliary to the main task at hand contribute to the unobtrusiveness of the space. Minimizing such input helps users to focus their attention on the task at hand.
4. *User satisfaction:* often times an accurate measure of obtrusiveness can be obtained by observing users' reactions or asking for feedback after completing tasks.
5. *Authentication mechanism latency:* this can be roughly estimated by calculating the time and attention required to go through the authentication process. An RFID tag would almost take no time at all (no action on behalf of the user is necessary),

whereas a user name / password combination requires a user to invest time and effort into entering the data manually and waiting for an answer.

Table 2 illustrates how to measure the metrics described earlier, and the measuring unit that is relevant.

Table 2: Metrics and their Measurement Units

Metrics	Unit
(1) Expressiveness of the security policy	4 different features for security policy expressiveness are identified. I propose measuring this metric by using a value of 0-4, representing the number of features supported.
(2) User control over private information	0-3, where 0 = no control provided. 1 = system provides control over the disclosure of one kind of information (content, location, or identity), 2 = system provides control over two kinds of information. 3 = system provides control over all three kinds of information.
(3) Unobtrusiveness of security mechanisms:	
Head turns	Total number of head turns auxiliary to the main task (i.e. head turns necessary to go through the authentication process and its feedback).
Physical Movement	Total number of physical movement auxiliary to the main task.
Keystrokes, clicks, actions and other atomic input	Total number auxiliary to the main task.
User Satisfaction	Subjective (1-5) scaling (5 = most satisfied)
Authentication mechanism latency	Time used for interacting with the security subsystem (e.g. authentication) auxiliary to the main task.

6.2 Evaluations

To evaluate the obtrusiveness of the proposed framework I measure the aforementioned metrics as explained in the table. The values for the first two metrics are measured and compared to key related works. To measure the third metric and its components, I perform a user study as discussed below.

6.2.1 Comparisons to other Approaches

I compare the Gaia Authentication Framework with (1) approaches that rely merely on traditional security mechanisms like SSL or public key cryptography (e.g., CoolTown [23]), (2) iROS iSecurity [48], because iROS shares a lot in common with Gaia. Table 3 outlines the result. The traditional security mechanisms employed by CoolTown and others only support mandatory policies, are context insensitive, and are unable to capture uncertainty or resolve conflicts. iSecurity has limited context sensitivity in comparison to Gaia's authentication framework and the rich context middleware [115]. Conflict resolution is not addressed by other systems. In the Gaia Authentication Framework, it is possible to detect and resolve conflicts by designing a set of suitable rules for detecting and resolving conflicts and feeding them into the inference engine. Privacy issues have not been addressed by the other systems yet, despite its importance in ubiquitous computing environments, particularly, when the surrounding environment is extremely identifying. Content privacy through encryption is achieved in both projects. The Gaia Authentication Framework also supports customizable identity and location privacy through the use of the Mist overlay.

Table 3: Comparison between Gaia Authentication Framework and other Related Work

Metric	Traditional security mechanisms (CoolTown)	iROS's iSecurity	Gaia Authentication Framework
Support for mandatory and discretionary policies	No. Only mandatory policies are supported.	Yes.	Yes.
Context sensitivity	No.	Limited.	Rich.
Uncertainty handling	No.	No.	Yes.
Conflict resolution	No.	No.	Limited.

User control over private data	Only content through encryption. Does not address other privacy issues.	Only content through encryption. Does not address other privacy issues.	Yes. Supports all three types, content, identity, and location (through the use of the Mist overlay.)

6.2.2 User Study

To measure the unobtrusiveness of the security mechanisms, I conduct a user study. In the user study, I bring an actual user, give him or her general instructions about using the Active Space, then ask the user to perform a task appropriate to his or her skills. Some tasks include running and going through a synchronized slideshow, editing a word processor document through multiple devices, playing a game that spans multiple devices, etc. Before being able to use the Active Space, the user is required to authenticate. The task is repeated several times, each time the user is asked to authenticate using a different method. The different authentication methods that have been measured include the Ubisense™ location detection system [116], active RFID tags [117], fingerprint reader (the Sony FIU-710), the Panasonic Authenticam™ iris scanner, and a user name / password combination that can be entered via a dedicated workstation in the space. The actions of the users are recorded, and the number of head turns, physical movements and atomic inputs that are auxiliary to the task are measured.

The user study is conducted on 15 volunteer users, who can be divided into three groups. The first group consists of 5 users who are students in the Engineering College and are technically savvy. The second group consists of 5 users who are students in other colleges and are less technically savvy. The third group consists of 5 middle and high school students. In order to limit the head turns, the Active Space is augmented with a text-to-speech application that gives audible and visual feedback to the result of the authentication process. The results of the study are outlined in Table 4. The numbers shown are the average values.

Table 4: Results of user study

Metric	Ubisense™	Active RFID	Fingerprint reader (FIU-710)	Authnticam™	User name / password
No. of Head turns	0	0	1.47	3.33	2.2
No. of Physical movements	0	0.2	1.27	1.4	1.13
No. of Key strokes, clicks, and other atomic inputs	0	0	1.2 (placing finger)	2.67 (alignment of eye with the camera)	Depends on user name and password length. Average was 12
User satisfaction (1-5) where 5 is best.	5	5	4.4	2	1.47
Authentication mechanism latency (seconds)	~0	1.16	2.28	9.27	6.94

The Ubisense™ and RFID appear to be most convenient, because they require the least head turns, physical movements and interactions with devices. Because Ubisense™ uses UWB it is detected almost immediately, whereas RFID required some physical movements in rare situations to get the reader to recognize the RFID card, and hence, the physical movement and latency is slightly below that of Ubisense™. The fingerprint reader required some head turns and physical movement to go to the device and place a finger. In some cases, a false negative is obtained on the first try, so some users had to place their fingers again. In some instances, users needed to perform extra head turns or physical movements. Nevertheless, users found fingerprint scanning to be convenient. The Panasonic Authenticam™ device for capturing iris scans has proven to be relatively obtrusive. The main reason for this is the immaturity of the hardware device that captures the iris image. A user needs to align his or her eye to the camera such that the eye is at a good distance and in focus with the lens. This task is not transparent and requires some practice. Users with prior experience tend to do it much faster, however, first time users have to often repeat the process several times until it succeeds. The user name / password combination scored the least in user satisfaction, probably due to the high amount of atomic input required. While Ubisense ™ and RFID have proven to be the least obtrusive, however, these two technologies are not very secure as they can be stolen or misplaced easily, therefore, it is sensible to have a context-aware system where the current context dictate how the tradeoff between convenience and security should be set, while building additional confidence in the identity of the entity as more information unfolds.

7. Conclusion

The shift to the ubiquitous computing paradigm brings forth new challenges to security and privacy, which cannot be addressed by mere adaptation of existing security and privacy mechanisms. Instead, novel security mechanisms must be devised. In this work, I have identified the challenges and requirements of security in ubiquitous computing environments. I have presented the design, implementation, and evaluation of a comprehensive framework that enriches Active Spaces with novel security mechanisms and enable cornerstone security services, including identification, authentication and privacy. The presented framework is a novel solution that provides fundamentally new possibilities in providing secure and privacy-preserving ubiquitous computing environments, without impairing the value-added services and customizability that make ubiquitous computing environments unique and powerful.

The major contributions of this dissertation are (1) The Gaia authentication framework, that combines rich context-awareness with automated reasoning under uncertainty to boost unobtrusiveness and seamlessness. (2) The Mist communication overlay that enables users to roam and communicate freely while preserving their privacy in the midst of an environment that is saturated with many sensors and identifying mechanisms that could threaten privacy.

The uniqueness of the Gaia authentication framework comes from three main points. First, support for pluggable, stackable, authentication modules that can be federated and loaded only when needed. Second, the ability to combine several different identification and authentication mechanisms to build up confidence as more authentication credentials are presented and combined. Third, the utilizing of context-awareness as a key parameter in deciding how much authentication is enough, as well as enriching context with the notion of different confidence levels associated with different principals.

The uniqueness of Mist comes from the fact that it is a novel approach that specifically targets ubiquitous environments and addresses location, identity, and content privacy. Mist allows a high degree of customization, where the end-user or the administrator can select a proper tradeoff between privacy and performance. Furthermore, Mist provides a high degree of privacy without impairing the value-added services that enrich ubiquitous computing.

The proposed system has been fully designed, implemented, deployed in test labs and real-world scenarios, and evaluated through user studies.

References

[1] M. Roman and R. H. Campbell, "GAIA: Enabling Active Spaces," presented at 9th SIGOPS European Workshop, Kolding, Denmark, 2000.

[2] M. Roman, C. Hess, A. Ranganathan, P. Madhavarapu, B. Borthakur, P. Viswanathan, R. Cerqueira, R. Campbell, and M. D. Mickunas, "GaiaOS: An Infrastructure for Active Spaces," University of Illinois at Urbana-Champaign Technical Report UIUCDCS-R-2001-2224 UILU-ENG-2001-1731, 2001.

[3] M. Roman, C. K. Hess, R. Cerqueira, R. H. Campbell, and K. Narhstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," *IEEE Pervasive Computing Magazine*, vol. 1, pp. 74-83, 2002.

[4] The Official Bluetooth Website, "Bluetooth." http://www.bluetooth.com/.

[5] "Reference number ISO/IEC 8802-11:1999(E) IEEE Std 802.11, 1999 edition. International Standard [for] Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," *IEEE*, 1999.

[6] M. Z. Win, R. A. Scholtz, and M. A. Barnes, "Ultra-wide bandwidth signal propagation for indoor wireless communications," presented at Int. Conf. Communications, Montréal, Canada, 1997.

[7] M. Weiser, "Hot Topics: Ubiquitous Computing," *IEEE Computer*, 1993.

[8] M. Weiser, "The world is not a desktop," *Interactions*, pp. 7-8, 1994.

[9] M. Weiser, "The Computer for the Twenty-First Century," in *Scientific American*, vol. 265, 1991, pp. 94-104.

[10] M. Langheinrich, "Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems," presented at ACM UbiComp 2001, Atlanta, GA, 2001.

[11] M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments," presented at 4th International Conference on Ubiquitous Computing, 2002.

[12] F. Stajano, *Security for Ubiquitous Computing*: Halsted Press, 2002.

[13] R. Kumar, "Pervasive Computing: A Safe Walk in the Woods?." Survey paper, http://www.cs.umn.edu/~richa/reports/PC.ps, 2002.

[14] L. Kagal, T. Finin, and A. Joshi, "Trust-Based Security in Pervasive Computing Environments," *IEEE Computer*, 2001.

[15] L. Kagal, J. Undercoffer, F. Perich, A. Joshi, and T. Finin, "Vigil: Enforcing Security in Ubiquitous Environments," in *Grace Hopper Celebration of Women in Computing 2002*, 2002.

[16] I. Zakiuddin, S. Creese, B. Roscoe, and M. Goldsmith, "Authentication in Pervasive Computing, Position Paper," presented at PAMPAS '02 - Workshop on Requirements for Mobile Privacy & Security Royal Holloway, University of London, 2002.

[17] J. Boyle and e. al, "The COPS Protocol." Internet Draft, Feb. 24, 1999.

[18] R. Mundy, D. Partain, and B. Stewart, "Introduction to SNMPv3." RFC 2570, April 1999.

[19] M. Stevens and e. al, "Policy Framework." IETF draft, September 1999.

[20] E. A. M. Luiijf, "Information Assurance and the Information Society," presented at EICAR Best Paper Proceedings, 1999.

[21] G. D. Abowd, "Classroom 2000: An experiment with the instrumentation of a living educational environment," *IBM Systems Journal*, vol. 38.

[22] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, "EasyLiving: Technologies for Intelligent Environments," presented at Handheld and Ubiquitous Computing (HUC), Bristol, England, 2000.

[23] "CoolTown, HP laboratories CoolTown Appliance Computing." http://cooltown.hp.com.

[24] A. Pfitzmann and M. Kohntopp, "Anonymity, Unobservability, and Pseudonymity -- A Proposal for Terminology," presented at International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, 2000.

[25] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness," presented at Workshop on the what, who, where, when and how of context-awareness at CHI, 2000.

[26] R. Sandhu, E. Coyne, H. Fienstein, and C. Youman, "Role Based Access Control Models," in *IEEE Computer*, vol. 29, 1996.

[27] M. Nyanchama and S. L. Osborn, "Modeling mandatory access control in role-based security systems," presented at IFIP Workshop on Database Security, 1995.

[28] R. S. S. a. Q. Munawer, "How to do Discretionary Access Control using Roles," presented at ACM Workshop on Role-based Access Control, 1988.

[29] B. S. Gill, "Dynamic Policy-Driven Role-based access control for active spaces," University of Illinois at Urbana-Champaign, 2001.

[30] P. Viswanathan, "Security Architecture in Gaia," University of Illinois at Urbana-Champaign, 2001.

[31] M. J. Covington, M. J. Moyer, and M. Ahamad, "Generalized Role-Based Access Control for Securing Future Applications," presented at 23rd National Information Systems Security Conference, 2000.

[32] M. Beigl and H.-W. Gellersen, "Ambient Telepresence," presented at Workshop on Changing Places, London, UK, 1999.

[33] F. Hupfeld and M. Beigl, "Spatially Aware Local Communication in the RAUM System," presented at IDMS, 2000.

[34] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad, "A Context-aware security architecture for emerging applications," presented at 18th ACSAC, Las Vegas, NV, 2002.

[35] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd, "Securing Context-Aware Applications using Environment Roles," presented at SACMAT, Virginia, USA, 2001.

[36] A. K. Dey and G. D. Abowd, "The Context Toolkit: Aiding the Development of Context-Aware Applications," presented at Workshop on Software Engineering for Wearable and Pervasive Computing, Limerick, Ireland, 2000.

[37] M. J. Covington, M. Ahamad, and S. Srinivasan, "A Security Architecture for Context-Aware Applications," Technical Report GIT-CC-01-12, College of Computing, Georgia Institute of Technology 2001.

[38] R. K. Thomas and R. S. Sandhu, "Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management," presented at IFIP WG11.3 Workshop on Database Security, Lake Tahoe, CA, 1997.

[39] C. K. Georgiadis, I.Mavridis, G. Pangalos, and R. K. Thomas, "Flexible Team-based Access Control Using Contexts," presented at 6 ACM Symposium on Access control models and technologies, 2001.

[40] H. Shen and P. Dewan, "Access Control for Collaborative Environments," presented at ACM Conference on Computer-Supported Collaborative Work (CSCW), 1992.

[41] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing," in *IEEE Pervasive Computing*, vol. 1, 2002, pp. 22-31.

[42] J. P. Sousa and D. Garlan, "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments," presented at IEEE/IFIP Conference on Software Architecture, Montreal, 2002.

[43] G. D. Abowd, "Classroom 2000: An experiment with the instrumentation of a living educational environment," *IBM Systems Journal*, vol. 38, pp. 508-530, 1999.

[44] P. Tandler, "Software Infrastructure for Ubiquitous Computing Environments: Supporting Synchronous Collaboration with Heterogeneous Devices," presented at Ubicomp 2001: Ubiquitous Computing, Atlanta, Georgia, 2001.

[45] N. Streitz, J. Geissler, and T. Holmer, "Roomware for Cooperative Buildings: Integrated Design of Architectural Spaces and Information Spaces," presented at Workshop on Cooperative Buildings (CoBuild'98), Darmstad, Germany, 1998.

[46] A. Fox, B. Johanson, P. Hanrahan, and T. Winograd, "Integrating Information Appliances into an Interactive Workspace," *IEEE Computer Graphics & Applications*, vol. 20, 2000.

[47] S. R. Ponnekanti, B. Johanson, E. Kiciman, and A. Fox, "Portability, Extensibility, and Robustness in iROS," presented at PerCom 2003, Dallas-Fort Worth, Texas, USA, 2003.

[48] Y. J. Song, W. Tobagus, D. Y. Leong, B. Johanson, and F. A., "iSecurity: A Security Framework for Interactive Workspaces," Stanford University, Technical Report September 3rd 2003.

[49] S. Yi, "Situation-Aware Security for Wireless Ad Hoc Networks," in *Department of Computer Science*: University of Illinois at Urbana-Champaign, 2005.

[50] J. Al-Muhtadi, M. Anand, D. Mickunas, and R. Campbell, "Secure Smart Homes Using Jini and UIUC SESAME," presented at 16th Annual Computer Security Applications Conference (ACSAC'2000), New Orleans, LA, 2000.

[51] M. Chandak, "UIUC-SESAME: Achieving a Portable Authentication, Access Control, and Delegation Protocol," in *Department of Computer Science*: University of Illinois at Urbana-Champaign, 1999.

[52] P. Kaijser, T. Parker, and D. Pinkas, "SESAME: The Solution to Security for Open Distributed Systems," *Computer Communications*, vol. 17, pp. 501-518, 1994.

[53] J. Al-Muhtadi, D. Mickunas, and R. Campbell, "A Lightweight Reconfigurable Security Mechanism for 3G Mobile Devices," *IEEE Wireless Communications Magazine*, vol. 9, 2002.

[54] C. Vielhauer, "Handwriting Biometrics for User Authentication: Security Advances in Context of Digitizer Characteristics," Technical University Darmstadt, 2004.

[55] OMG, "CORBA 3.0.3 Specification, formal/2004-03-01." available at http://www.omg.org/cgi-bin/doc?formal/04-03-01, 2004.

[56] S. Lederer, A. Dey, and J. Mankoff, "Everyday Privacy in Ubiquitous Computing Environments," presented at Ubicomp 2002 Workshop on Socially-informed Design of Privacy-enhancing Solutions in Ubiquitous Computing, 2002.

[57] X. Jiang and J. Landay, "Modeling Privacy Control in Context-aware Systems," *IEEE Pervasive Computing*, vol. 1, 2003.

[58] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote Trust-Management System." RFC 2704, 1999.

[59] Y. Chu, J. Feigenbaum, B. A. LaMacchia, P. Resnick, and M. Strauss, "REFEREE: Trust Management for Web Applications," *WWW6 / Computer Networks*, vol. 29, pp. 953-964, 1997.

[60] N. Li, W. Winsborough, and J. Mitchell, "Distributed Credential Chain Discovery in Trust Management (Extended Abstract)," presented at 8th ACM Conference on Computer and Communications Security, Philadelphia, PA, 2001.

[61] T. Yu, M. Winslett, and K. E. Seamons, "Interoperable Strategies in Automated Trust Negotiation," presented at ACM Conference on Computer and Communications Security, Philadelphia, PA, 2001.

[62] T. Yu, M. Winslett, and K. E. Seamons, "Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation," *ACM Trans. Inf. Syst. Secur.*, vol. 6, pp. 1-42, 2003.

[63] T. Yu and M. Winslett., "A unified scheme for resource protection in automated trust negotiation.," presented at Symposium on Security and privacy, 2003.

[64] P. Bonatti and P. Samarati, "Regulating Service Access and Information Release on the Web," presented at 7th ACM Conference on Computer and Communications Security, Athens, Greece, 2000.

[65] P. A. Bonatti and P. Samarati, "A uniform framework for regulating service access and information release on the Web," *Journal of Computer Security*, vol. 10, pp. 241-271, 2002.

[66] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5: A Protocol for Scalable Anonymous Communication," presented at IEEE Symposium on Security and Privacy, 2002.

[67] "Anonymizer." http://www.anonymizer.com.

[68] "SafeWeb." http://www.safeweb.com.

[69] M. Reiter and A. D. Rubin, "Crowds: Anonymity for Web Transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, 1998.

[70] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas in Communication, Special Issue on Copyright and Privacy Protection*, 1998.

[71] Y. Guan, C. Li, D. Xuan, R. Bettati, and W. Zhao, "Preventing Traffic Analysis for Real-Time Communication Networks," presented at The IEEE Military Communication Conference (MILCOM) '99, 1999.

[72] V. Scarlatta, B. Levine, and C. Shields, "Responder anonymity and anonymous peer-to-peer file sharing," presented at IEEE International Conference on Network Protocols (ICNP), 2001.

[73] M. Wright, M. Adler, B. N. Levine, and C. Schields, "An Analysis of the Degradation of Anonymous Protocols," presented at Proceedings of Symposium on Network and Distributed Systems Security, 2002.

[74] "Platform for Privacy Preferences." www.w3.org/TR/P3P.

[75] "OASIS Privacy Profile of XACML." http://docs.oasis-open.org/xacml/access_control-xacml-2_0-privacy_profile-spec-cd-01.pdf: Comittee Draft 01, 2004.

[76] "NIST Common Criteria V 2.1, Annex I Privacy (FPR)." http://docs.oasis-open.org/xacml/access_control-xacml-2_0-privacy_profile-spec-cd-01.pdf.

[77] J. Y. Halpern and M. O. Rabin, "A logic to reason about likelihood," *Artificial Intelligence*, vol. 32, pp. 379-405, 1987.

[78] D. Poole, "A logical framework for default reasoning," *Artificial Intelligence*, vol. 36, pp. 27- 47, 1988.

[79] R. Reiter, "Nonmonotonic Reasoning," *Annual Review of Computer Science*, vol. 2, 1987.

[80] K. Segerberg, "Qualitative Probability in a Modal Setting," presented at J. E. Fenstad, ed. Proceedings of the Second Scandinavian Logic Symposium, North Holland, Amsterdam, 1971.

[81] P. Gardenfors, "Qualitative Probability as intensional logic," *J. Philos. Logic*, vol. 4, pp. 171-185, 1975.

[82] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*: Cambridge University Press, 1999.

[83] G. Hughes and M. Cresswell, *A New Introduction to Modal Logic*: London: Routledge, 1996.

[84] J. van Benthem, *The Logic of Time*, second edition ed. Boston and London: Kluwer Academic Publishers, 1991.

[85] J. van Benthem, "Temporal Logic," *D. M. Gabbay, C. J. Hogger, and J. A. Robinson, Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 4, pp. 241-350, 1995.

[86] J. F. Allen, "Towards a general theory of action and time," *Artificial Intelligence*, vol. 23, pp. 123-154, 1984.

[87] A. P. Galton, "Time and Change for AI," *D. M. Gabbay, C. J. Hogger, and J. A. Robinson, Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 4, pp. 175-240, 1995.

[88] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, pp. 241-288, 1986.

[89] R. E. Neapolitan, *Probabilistic reasoning in expert systems*. New York, USA: John Wiley and Son Inc., 1990.

[90] G. Shafer, *A mathematical theory of evidence*: Princeton University Press, 1976.

[91] G. Shafer, "Probability Judgment in Artificial Intelligence," presented at Uncertainty in Artificial Intelligence, Amsterdam, North-Holland, 1986.

[92] L. Zadeh, "Fuzzy sets as basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3-28, 1978.

[93] V. Novak, *Fuzzy Sets and their Applications*. Ostrava, Czech Republic: Czechoslovak Academy of Sciences, 1989.

[94] B. N. a. T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," in *IEEE Communications Magazine*, vol. 32, 1994, pp. 33-38.

[95] M. Román, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," *IEEE Pervasive Computing (accepted)*, 2002.

[96] M. Roman and R. Campbell, "GAIA: Enabling Active Spaces," presented at 9th ACM SIGOPS European Workshop,, Kolding, Denmark, 2000.

[97] A. Whitten and J. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," presented at 8th USENIX Security Symposium, 1999.

[98] C. Lueg, "On the Gap between Vision and Feasibility," *LNCS 2414*, pp. 45, 2002.

[99] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*: MA: Addison-Wesley, 1984.

[100] V. Samar and R. Schemers, "Unified Login with Pluggable Authentication Modules (PAM)," RFC 86.0, 1995.

[101] OMG, "CORBA, Architecture and Specification," Common Object Request Broker Architecture (CORBA) 1998.

[102] A. K. Chandra and e. al., "Horn Clauses Queries and Generalization," *J Logic Programming*, 1985.

[103] O. Shmueli, "Decidability and expressiveness aspects of logic queries," presented at sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of database systems, San Diego, CA USA, 1987.

[104] M. Jarke and e. al, "An Optimizing PROLOG Front-End to a Relational Query System," presented at ACM SIGMOD '84 Conference, Boston, MA, 1984.

[105] M. Roman, F. Kon, and R. H. Campbell, "Reflective Middleware: From Your Desk to Your Hand," *IEEE Distributed Systems Online Journal, Special Issue on Reflective Middleware*, 2001.

[106] "XSB Prolog." http://xsb.sourceforge.net.

[107] D. A. Cooper and K. P. Birman, "Preserving Privacy in a Network of Mobile Computers.," presented at IEEE Symposium on Research in Security and Privacy, 1995.

[108] G. Sampemane, P. Naldurg, and R. Campbell, "Access Control for Active Spaces," presented at the Annual Computer Security Applications Conference (ACSAC), Las Vegas, NV, 2002.

[109] J. Al-Muhtadi, R. Campbell, A. Kapadia, D. Mickunas, and S. Yi, "Routing Through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments," presented at International Conference of Distributed Computing Systems (ICDCS 2002), Vienna, Austria, 2002.

[110] J. Al-Muhtadi, R. Campbell, A. Kapadia, D. Mickunas, and S. Yi, "Routing through the Mist: Design and Implementation," UIUCDCS-R-2002-2267, March 2002.

[111] "Security Metrics Guide for Information Technology Systems," NIST Special Publications, 2003.

[112] G. Abowd, "Software Engineering Issues for Pervasive computing," presented at Proceedings of the 21st International Conference on Software Engineering, 1999.

[113] J. Biehl and B. Bailey, "ARIS: An Interface for Application Relocation in an Interactive Space," presented at Graphics Interface 2004, 2004.

[114] M. Roman, J. Al-Muhtadi, B. Ziebart, R. Campbell, and M. D. Mickunas, "System Support for Rapid Ubiquitous Computing Application Development and Evaluation," presented at System Support for Ubiquitous Computing Workshop (UbiSys '03) in conjunction with UbiComp '03, Seattle, WA, 2003.

[115] A. Ranganathan and R. H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," presented at Middleware 2003 (submitted) http://choices.cs.uiuc.edu/~ranganat/Pubs/MiddlewareForContext.pdf, 2003.

[116] UbiSense, "Local position system and sentient computing." http://www.ubisense.net/.

[117] RFId, "Radio Frequency Identification, (RFID)," http://www.aimglobal.org/technologies/rfid/.

Author’s Biography

Jalal Al-Muhtadi was born in Riyadh, Saudi Arabia. He graduated from King Saud University, Saudi Arabia in 1996 with a Bachelor of Computer and Information Sciences, Computer Science field. He was ranked first in his graduating class and awarded the “First Class Honor Award” and the “Best Graduation Project Award” from the College of Computer and Information Sciences. He was also awarded a scholarship to pursue graduate studies abroad. He enrolled in the graduate program at the Department of Computer Science, University of Illinois at Urbana-Champaign in 1998. He obtained his Master of Science Degree from the University of Illinois in Computer Science in 2001 and his Doctor of Philosophy Degree in Computer Science, in October 2005. Throughout his graduate studies, he worked as a Research Assistant in the Gaia project at the University of Illinois at Urbana-Champaign. His research interests include ubiquitous computing, distributed systems, middleware and infrastructure security and privacy issues.