TRUST-BASED SECURITY INFRASTRUCTURE FOR UBIQUITOUS COMPUTING SYSTEMS

by

u-Security Team[1]

Technical Report V.1.1

RTMM, Kyung Hee University

28 Feb. 2006

Approved by ______________________________
Project Supervisor

____

Date : 28 Feb 2006

[1] Hassan Jameel, Le Xuan Hung, Nguyen Ngoc Diep, Pho Duc Giang, Riaz Ahmed Shaikh, Sungyoung Lee, Tran Van Phuong, Young-Koo Lee, and Yuan Weiwei,. (Note: All in alphabetical order)

Ubiquitous Computing Lab, Kyung Hee University

ABSTRACT

TRUST-BASED SECURITY
INFRASTRUCTURE FOR UBIQUITOUS
COMPUTING ENVIRONMENTS

by u-Security Team

Project Supervisor: Professor Sungyoung Lee
Department of Computer Engineering

*Ubiquitous Computing (*Ubicomp) is a revolution of computing paradigm that promise to have a profound affect on the way we interact with computers, devices, physical spaces and other people. This new technology envisions a world where embedded processors, computers, sensors, and digital communications are inexpensive commodities that are available everywhere. This eliminates time and place barriers by making services available to users anytime and anywhere. Ubicomp will surround users with a comfortable and convenient information environment that merges physical and computational infrastructures into an integrated habitat. This habitat will feature a proliferation of hundreds or thousands of computing devices and sensors that will provide new functionality, offer specialized services, and boost productivity and interaction. Context-awareness will allow this habitat 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. We term this dynamic, information-rich habitat a "*smart space"*. Within this space, individuals may interact with flexible applications that may follow the user, define and control the function of the space, or collaborate with remote users and applications.

Ubicomp imposes peculiar constraints compared with other systems, for example in terms of connectivity, computational power and energy budget, which make this case significantly different from those contemplated by the canonical doctrine of security in distributed systems. A well-established taxonomy subdivides computer security threats into three categories, according to whether they threaten confidentiality, integrity or availability.

Confidentiality is the property that is violated whenever information is disclosed to unauthorized principals. Everyone realizes that wireless networking is more vulnerable to passive eavesdropping attacks than a solution based on cables: by construction, information is radiated to anyone within range. It is natural to expect that the security requirements of a wireless system will include addressing this concern.

Integrity is violated whenever information is altered in an unauthorized way. This applies both to information within a host and to information in transit between hosts. Imagine a wireless temperature sensor on your roof that relays its measurements to a display inside your house. If an attacker modifies either the sensor's firmware or the transmitted messages so that the displayed temperature is off by 10 degrees then, if you are sufficiently gullible, you may be cheated into wearing the wrong type of clothes for that day's weather.

Availability is the property of a system which always honors any legitimate requests by authorized principals. It is violated when an attacker succeeds in denying service to legitimate users, typically by using up all the available resources. As we remarked about integrity, the fact that Ubicomp implies unattended devices opens the door to many abuses. If we envisage that these ubiquitous hosts might accept mobile code that roams from one of them to another, then denial of service might also be caused by malicious programs that lock up the host device

In order to fulfill such security requirements for Ubiquitous Environments, we have proposed USEC architecture with seven major components (Entity Recognition, Trust/Risk Management, Trust-based Access Control, Privacy Control, Intrusion Detection, and Home Firewall) along with Policy Database and Lightweight Cryptography Library.

Entity Recognition component supports flexibly various devices such as Smart Badges, iButtons, Smart Watchs, PDAs. This component integrates different type of authentications, ranging from conventional authentication approaches (Username/Password, PKI, Kerberos, etc) to emerging identity recognition technology. The Resurrecting Duckling security policy model is an example of entity recognition; ducklings know that their mother is the first entity who sent the imprinting key when they were born. Trust/Risk Management supports Access Control. It provides trust collaborations and interactions which usually occur among roaming entities. By modeling trust relationships in smart spaces environments, unknown entities from different domains can interact and request services and resources from a given domain in secure and privacy manner. Risk evaluator and Trust value cooperate with each other to make decision of entity interaction. Trust-based Access Control is critical to preserve confidentiality and integrity. Conventionally, the condition of confidentiality requires that only authorized users can read information, and the condition of integrity requires that only authorized users can alter information and in authorized ways. In USEC, Trust-based Access Control extends scopes of users by using Trust/Risk Management. By doing this, it supports not only pre-registered users but also not pre-registered users. Privacy Control is integral part in this convenience but obtrusive environment. It provides location privacy, anonymous connections and confidentiality of information to users. USEC infrastructure also provides Home Firewall to defend against outsider potential attackers.

Table of Contents

C h a p t e r 1

INTRODUCTION TO SECURITY FOR UBIQUITOUS COMPUTING

1 Introduction to Security for Ubiquitous Computing

1.1 Ubiquitous Computing: A Revolution of Computing Paradigm

In the year 1991, *Mark Weise* originally introduced the term *'Ubiquitous computing'* (*'Ubicomp'*) [1.1]. Since then, Ubicomp technology has been growing rapidly. Many researchers have spent a lot of effort to bring this technology to reality in order to facilitate the human-life. In this chapter, we briefly introduce Ubicomp, we point out principal security concerns that we shall facing (more on this in next chapter), and we give some guideline that motivates us to build up our security infrastructure.

Ubicomp is a revolution of computing paradigm that promise to have a profound affect on the way we interact with computers, devices, physical spaces and other people. This new technology envisions a world where embedded processors, computers, sensors, and digital communications are inexpensive commodities that are available everywhere. This eliminates time and place barriers by making services available to users anytime and anywhere [1.1]. Ubicomp will surround users with a comfortable and convenient information environment that merges physical and computational infrastructures into an integrated habitat. This habitat will feature a proliferation of hundreds or thousands of computing devices and sensors that will provide new functionality, offer specialized services, and boost productivity and interaction. Context-awareness will allow this habitat 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. We term this dynamic, information-rich habitat a "*smart space*" Within this space, individuals may interact with flexible applications that may follow the user, define and control the function of the space, or collaborate with remote users and applications.

Smart Spaces (Smart Home, Smart Office) have been the first targets of Ubicomp technology. A lot of Ubicomp systems have been proposed and built up so far such as CAMUS (Context-Aware Middleware for Ubicomp Systems) [1.2], SOCAM (Service-oriented Context-Aware Middleware) [1.3], CASS (Context-awareness sub-structure) [1.4], CoBrA (Context Broker Architecture) [1.5], CORTEX (Context-Aware Middleware for Pervasive and Ad Hoc Environments) [1.6], GAIA (A Middleware Infrastructure to Enable Active Spaces) [1.7], etc. Security and Privacy are crucial and indispensable parts of an Ubicomp system, but among these, a few takes those aspects into account. In next section, we'll point out some principal security problems in such smart environments.

1.2 And Its Security Problems

Ubicomp imposes peculiar constraints compared with other systems, for example in terms of connectivity, computational power and energy budget, which make this case significantly different from those contemplated by the canonical doctrine of security in distributed systems. A well-established taxonomy subdivides computer security threats into three categories, according to whether they threaten confidentiality, integrity or availability.

We briefly summarize these three fundamental security properties given the preconditions of Ubicomp that has been mentioned in [1.8].

1.2.1 Confidentiality

Confidentiality is the property that is violated whenever information is disclosed to unauthorized principals. Everyone realizes that wireless networking is more vulnerable to passive eavesdropping attacks than a solution based on cables: by construction, information is radiated to anyone within range. It is natural to expect that the security requirements of a wireless system will include addressing this concern.

1.2.2 Integrity

Integrity is violated whenever information is altered in an unauthorized way. This applies both to information within a host and to information in transit between hosts. Imagine a wireless temperature sensor on your roof that relays its measurements to a display inside your house. If an attacker modifies either the sensor's firmware or the transmitted messages so that the displayed temperature is off by 10 degrees then, if you are sufficiently gullible, you may be cheated into wearing the wrong type of clothes for that day's weather. If this does not look like a terribly dramatic security violation, imagine instead that the sensor is monitoring a patient's temperature in a clinic or, even better, that it is part of an alarm system for a nuclear power plant.

As happens with confidentiality, the wireless nature of communications increases the vulnerability of the system to integrity violations: if the receiver listens to the strongest signal that "looks right", an attacker wishing to substitute forged messages for the original ones only needs to shout loudly enough, without having to splice any cables. As for the integrity of hosts, as opposed to that of messages in transit, the Ubicomp vision of unattended devices ready to communicate with whoever comes in range clearly makes it likely that an attacker will sooner or later tamper with such unattended devices if this can bring her any benefits.

1.2.3 Availability

Availability is the property of a system which always honors any legitimate requests by authorized principals. It is violated when an attacker succeeds in denying service to legitimate users, typically by using up all the available resources. As we remarked about integrity, the fact that Ubicomp implies unattended devices opens the door to many abuses. If we envisage that these ubiquitous hosts might accept mobile code that roams from one of them to another, then denial of service might also be caused by malicious programs that lock up the host device.

While illustrating the three fundamental security properties of confidentiality, integrity and availability we have repeatedly referred to "authorized principals". It follows that a fundamental prerequisite of a secure system is the ability to establish whether any given principal is or is not authorized to perform the action it is requesting. To define "who is authorized to do what" is the duty of the security policy, a concise specification of the security goals of the system. In order to ascertain whether the policy authorizes a principal to perform an action, there is also a need for identification (finding out who the principal claims to be) and particularly authentication (establishing the validity of this claim). Authentication is one of the foundations of security: it is easy to come up with examples that demonstrate that, in its absence, the three fundamental properties can be trivially violated.

1.3 References

[1.1] M. Weiser: Scientifc America. The Computer for the 21st Century. (Sept. 1991) 94-104; reprinted in IEEE Pervasive Computing. (Jan.-Mar. 2002) 19-25.

[1.2] Hung Quoc Ngo, Anjum Shehzad, Kim Anh Pham Ngoc, Sungyoung Lee, and Manwoo Jeon, "Research Issues in the Development of Context-aware Middleware Architectures," The 11th IEEE International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA 2005), 17-19 August 05, HongKong, http://www.comp.hkbu.edu.hk/~rtcsa2005/

[1.3] T. Gu, H.K. Pung, and D.Q. Zhang, "A service-oriented middleware for building context-aware services," Journal of Network and Computer Applications, Vol. 28, No. 1, 2005, pp. 1-18.

[1.4] Patrick Fahy and Siobhán Clarke. Cass: Middleware for mobile, context aware applications. In Workshop on Context Awareness at MobiSys 2004, June 2004.
[1.5] Chen, H.L., Finin, T., Joshi, A.: A context broker for building smart meeting rooms. In Schlenoff, C., Uschold, M., eds.: Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium, AAAI Press, Menlo Park, CA (2004) 53–60
[1.6] Thirunavukkarasu Sivaharan, Gordon S. Blair, Adrian Friday, Maomao Wu, Hector A. Duran-Limon, Paul Okanda and Carl-Fredrik Sørensen. "Cooperating Sentient Vehicles for Next Generation Automobiles ". ACM/USENIX MobiSys 2004 International Workshop on Applications of Mobile Embedded Systems, Boston, USA. June 2004.
[1.7] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganat, R. H. Campbell, and K. Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. IEEE Pervasive Com puting,vol.1,pp.74-82,2002, 2002..
[1.8] F. Stajano: Security for Ubiquitous Computing, John Wiley & Sons, Chichester, UK, 2002.

C h a p t e r 2

SECURITY CHALLENGES AND REQUIREMENTS

2 Security Challenges and Requirements

In this section, we briefly mention about the major challenges and requirements for securing pervasive computing environments which Roy C. *et al* outlined in their paper [2.1]

2.1 Challenges

As mentioned before, the additional features and the extended functionality that pervasive computing offers make it prone to additional vulnerabilities and exposures. Below, we mention these features that add extra burden to the security subsystem.

2.1.1 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. Pervasive computing, however, extends its reach beyond the computational infrastructure and attempts to encompass the surrounding physical spaces as well. Pervasive 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.

2.1.2 Privacy Issues

The physical outreach of pervasive computing makes preserving users' privacy a much 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 electronically stalk particular users. The entire system now 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.

2.1.3 User Interaction Issues

One of the main characteristics of pervasive applications is a richer user-interface for interaction between users and the space. A variety of multimedia mechanisms are used for input and output, and to control the physical aspects of the space. At any point of time, the set of users in the space affects the security properties of the space. Because of the nature of these interactions, 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 access control mechanisms. We believe that the access control mechanisms should allow groups of users and devices to use the space in a manner that facilitates collaboration, while enforcing the appropriate access control policies and preventing unauthorized use. Thus the physical and "virtual" aspects of access control for such spaces have to be considered together.

2.1.4 Security Policies

It is important in pervasive computing to have a flexible and convenient method for defining and managing security policies in a dynamic and flexible fashion. Policy Management tools provide administrators the ability to specify, implements, and enforces rules to exercise greater control over the behavior of entities in their systems. Currently, most network policies are implemented by systems administrators using tools based on scripting applications [2.2,2.3] that iterate through lists of low-level interfaces and change values of entity-specific system variables. The policy management software maintains an exhaustive database of corresponding device and resource interfaces. With the proliferation of heterogeneous device-specific and vendor-specific interfaces, these tools may need to be updated frequently to accommodate new hardware or software, and the system typically becomes difficult to manage. As a result, general purpose low-level management tools are limited in their functionality, and are forced to implement only generic or coarse-grained policies [2.4].

Since most policy management tools deal with these low-level interfaces, administrators may not have a clear picture of the ramifications of their policy management actions. Dependencies among objects can lead to unexpected side effects and undesirable behavior [2.5]. Further, the disclosure of security policies may be a breach of security. For example, knowing whether the system is on the lookout for an intruder could actually be a secret. Thus, unauthorized personnel should not be able to know what the security policy might become under a certain circumstance.

2.1.5 Information Operation

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,*" [2.6] 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. Pervasive computing gives extreme leverage and adds much more capabilities to the arsenal of info warriors, making info ops a much more severe threat.

2.2 Security Requirements

To deal with the new vulnerabilities introduced by pervasive computing, security and privacy guarantees in pervasive computing environments 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. In this section, we briefly mention the important requirements needed for a security subsystem for pervasive computing environments.

2.2.1 Transparency and unobtrusiveness

The focal point of pervasive 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.

2.2.2 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 system's confidence.

2.2.3 Context-Awareness

Often, traditional security is somewhat static and context insensitive. Pervasive 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 context information available. 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 basis. 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.

2.2.4 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.

2.2.5 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.

2.2.6 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 becomes interdependent.

2.2.7 Scalability

Pervasive 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.

2.3 References

[2.1] Roy Campbell, Jalal Al-Muhtadi, Prasad Naldurg, Geetanjali Sampemane, M. Dennis Mickunas, "Towards Security and Privacy for Pervasive Computing." in Theories and Systems, Mext-NSF-JSPS International Sympsoium, ISSS 2002, Tokyo, Japan, November 2002. pp. 1-15, G. Goos, J. Hartmanis, and J. vanLeeuwen (editors) in Lecture Notes in Computer Science.

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

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

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

[2.5] P. Loscocco and S. Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating System," presented at Proceedings of the FREENIX Track of the 2001 USENIX, 2001.

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

C h a p t e r 3

RELATED WORK

3 Related work

Ubiquitous Computing seeks to move computers into the background while using them to enhance human endeavors. Nevertheless, will society accept and totally trust the security associated with background computational activities? Are there new security threats to person and property as a consequence of this successor to interactive computing? Can we seamlessly leverage current/ classical security concepts and solutions to an augmented world? Ubiquitous computing security research may still be considered a "hot topic", as there still remain some open questions. There are many projects to address problems related to ubiquitous computing security [3.1].

3.1 SHAD – A Lightweight Security Scheme.

SHAD - a lightweight security scheme [3.2] avoids the use of centralized entities and it is designed to be agile in a Peer-to-Peer environment. SHAD meet the following requirements:

1. Independence of centralized services or authentication servers.
2. Ease of use and the non-obtrusiveness.
3. Supporting of disconnections and delegation.
4. Minimizing of power consumption and the processing limitations of mobile devices.
5. Ease of deployment.

It is based on a personal device called PCM (Personal Command Module) – a mobile device that represents the user in the system. It acts as both client and server as well in the system. The PCM allows the user to control his activities and to use the pervasive elements nearby. The PCM stores and manages the user's secrets and it is the entity in charge of granting or refusing service requests. Using the PCM to authenticate users, they can comply with the necessity of independence of centralized services. Two principles can mutually authenticate and exchange tickets.

The user has not to introduce passwords or to be using authentication devices (e.g. fingerprint reading devices) all the time, because the PCMs are able to negotiate keys without human supervision. The PCM stores the secrets used by its owner to use resources that do not belong to him. A principal can use a resource (that does not belong to him) when the owner is online. Later, the principal will be able to keep using the resource although the resource owner disconnects, because the PCM may handle disconnections by transferring session keys to devices. SHAD uses role based access control (RBAC) policy as assigning roles to the humans. So it has several different access privileges in each side depending on the level of trust. Because of minimizing the power consumption in mobile devices, it uses symmetric cryptography.

SHAD is a new idea for security scheme in Peer-to-peer environment. But actually, they have some drawbacks. SHAD needs an owner for each devices (like an administrator), so that it is not flexible enough. We need a manual configure at the first time to response to other requests and of course we need to reconfigure if we want to change the rights of others. If the key is disclosed, and be stolen, then someone can use the devices without permissions. Another problem is how to use a devices if we do not know about where is the owner, who is he?

3.2 Cerberus - A Security Architecture for Context-Aware Middleware GAIA

Security Service of Gaia project comes with different approach. Cerberus [3.3], the security architecture for Gaia, would not work for old applications not using Gaia. Approaches based on libraries and frameworks that allow programmers to reach context information that ancient software does not work and it has to be re-implemented. Contrary to SHAD, this is centralized approach.

This security architecture is dynamic, context-aware security architecture for Active Spaces. This features a federated authentication system that is based on distributed, pluggable, "CORBArized" authentication modules. This module-based service allows the separation of applications from the actual authentication mechanisms and devices. The dynamically pluggable modules allow the authentication subsystem to incorporate additional authentication devices and mechanisms on the fly as they become available.

The access control system is designed to automate the creation and enforcement of access control policies for different configurations of an active space. The system explicitly recognizes different modes of cooperation between groups of users, and the dependence between physical and virtual aspects of security in Active Spaces. They support both discretionary and mandatory access control policies, and use role-based access control techniques for easy policy administration. Their model dynamically assigns permissions to user roles based on context information. Dynamic protection domains allow administrators and application developers the ability to customize access control policies. The model preserves the principle of least privilege, promotes separation of duty, and prevents rights-amplification.

The Cerberus core service of Gaia aims to capture as much context information as possible by deploying different devices and sensors, identifying entities and reasoning automatically in order to provide an unobtrusive computer environment. Cerberus consists of four major components: (1) the security service, (2) the context infrastructure, (3) a knowledge base that stores various security policies, and (4) an inference engine, which performs automated reasoning and enforces the security policies.

One new aspect that Cerberus has comparing to other security projects is they do the authentication associated with confidence values. This confidence value represents how confident the authentication system is about the identity of the principal. And it is used as a parameter of the context.

Security policies in Cerberus are written as rules in first order logic. There are two kinds of policies used in Cerberus. 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.

The Inference Engine performs two kinds of tasks:

- 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.
- 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 Provider Lookup Service to look up various context providers. It can also get authentication information of various people in the space from the authentication service.

3.3 SECURE-Secure Environments for Collaboration among Ubiquitous Roaming Entities

Another approach to secure systems in ubiquitous environment is trust-based one. SECURE (Secure Environments for Collaboration among Ubiquitous Roaming Entities) [3.4] is a project like that.

The heart of the SECURE project is the development of a computational model of trust that will provide the formal basis for reasoning about trust and for the deployment of verifiable security policies. The result of the project is the development of a software framework encompassing algorithms for trust management including algorithms to handle trust formation, trust evolution and trust propagation.

They designed a scheme called APER which is usable to recognize peers on a network. APER assumes that the network supports some form of "broadcast" or "multicast" messaging, for example using IP broadcast or multicast addresses, or adopting an application layer broadcast approach. APER has not (yet) undergone peer review for its security properties, they only indicate the properties that they assume it to have, which is fine for current purposes since the scheme is really a proof-of-concept for the "recognition is enough" argument.

It has three levels of recognition, any of which can be sufficient, depending on higher level policies. Each level will have some associated parameters (e.g. the number of claims seen), which may also impact on how the recognition is treated.

- *Level 1:* Claimants signature verified over a set of recently seen claims
- *Level 2:* Level 1 and claimants recent claims are "fresh", based on the "last-n-hashes" mechanism
- *Level 3:* Level 2 and the claimant successfully responded to a challenge

To provide adaptability to an entity's capabilities and to legacy authentication, it uses pluggable solution so that entity recognition module into which different recognition schemes can be plugged. The design of that pluggable recognition module (PRM) leverages from other work such as PAM. PAM allows for the use of different legacy authentication schemes: Kerberos, smart cards, etc. In order to get dynamic enrolments, policies - regarding which authentication scheme or combination of authentication schemes should be used - cannot require an administrator to be effective. For pervasive computing, the degree of auto-configuration has to be increased. To achieve this, the appropriate recognition scheme must be negotiated, which means that alternative recognition schemes are also possible. The negotiation will use the degree of trust needed.

At the end, it evaluates the end-to-end trust, which is the result of trust in technology and in other entities. The formula is like this:

$$\text{End-to-end trust} = (\text{Trust in infrastructure}) * (\text{Trust in entities})$$

For trust in the underlying technology, it uses metrics, dynamically calculated or statically defined by a group of experts. For trust in entities, trust would be calculated based on the human notion of trust, probably thanks to direct observations, past history, and careful use of recommendation and reputation.

3.4 References

[3.1] Workshop on Security in Ubiquitous Computing,UBICOMP 2002, Göteborg Sweden

[3.2] Enrique Soriano Salvador, "SHAD: A Human Centered Security Architecture for Partitionable, Dynamic and Heterogeneous Distributed Systems", ACM 2005

[3.3] Jalal Al-Muhtadi et al, "Cerberus: A Context-Aware Security Scheme for Smart Spaces", Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 03)

[3.4] Jean-Marc Seigneur et al, "Entity Recognition Scheme (SECURE Project)", 2003

Chapter 4

USEC SECURITY INFRASTRUCTURE: COMPONENTS AND ACTIVITIES

4 USEC Security Infrastructure: Components and Activities

4.1 Motivation

Some day in very near future, we'll be staying in ubiquitous environments where human are facilitated in every day life. Security is critic for these environments: users must be authenticated, confidentiality and integrity have to be assured (mainly in wireless communications) and the access to the resources must be controlled.

Traditional security schemes, like Kerberos [4.6] and Sesame [4.7, 4.8], are based on centralized entities like key distribution centers and authentication authorities. Other security schemes that have been proposed for ubiquitous environments are heavily [4.9,4.10] based on them. In most ubiquitous schemes [4.11-4.15], it is necessary to reach centralized servers to query context and location information too. For these systems to work principals need to be online and they must have connectivity with the centralized server if they want to communicate. What would then happen when two users meet at a remote isolated place? Another problem is that most of these systems are hard to administrate because they need accounts for users that require management. Moreover, the centralized services are unique failure points for the ubiquitous environment. Many approaches are based on middleware. These systems have problems with existing or native applications, because in most cases these applications have to be modified or wrapped. An example is Cerberus [4.16], the security architecture for Gaia that would not work for old applications not using Gaia. Approaches based on libraries and frameworks that allow programmers to reach context information have the same problem: ancient software does not work and it has to be re-implemented. Moreover, most of these architectures fail to work in P2P settings.

Therefore, a new architecture is needed for Ubicomp environments, because the current ones introduce complexity, obtrusiveness and centrality. Our aim is to build up a security infrastructure in order to fulfill such weaknesses as we mentioned earlier. This infrastructure along with our context-aware middleware (CAMUS) will provide Secure Smart Space Environments. For the first proposed architecture, we are dealing with six major components: Entity Recognition, Access Control, Intrusion Detection, Trust/Risk Management, Privacy Control, and Firewall. Entity Recognition is novel approaches of Authentication for Ubicomp. Entity Recognition and Access Control support flexibly and variously devices and methods, such as Smart Badges, iButtons, Smart Watchs, PDAs, integrating conventional authentication approaches (Username/Password, PKI, Kerberos, etc) and emerging identity recognition technology. The Resurrecting Duckling security policy model is an example of entity recognition; ducklings know that their mother is the first entity who sent the imprinting key when they were born. Trust Management supports trust collaborations and interactions which usually occur among roaming entities. By modeling trust relationships in smart spaces environments, unknown entities from different domains can interact and request services and resources from a given domain in secure and privacy manner. Besides, Privacy Control is integral part in this convenience but obtrusive environment. It provides location privacy, anonymous connections and confidentiality of information to users. Since smart spaces are deployed by various wire and wireless devices and systems, it is likely to be attacked by outsider. Thus, Firewall is good solution to defend against such threats.

4.2 USEC Overview

USEC (A Trust-based Security Infrastructure) is a component-based security architecture supporting ubiquitous computing systems. It is composed of seven security components along with Policy Database and Cryptography Library as depicted in Fig. 4.1: Entity Recognition, Trust/Risk Management, Hybrid Access Control, Privacy Control, Intrusion Detection, and Home Firewall.

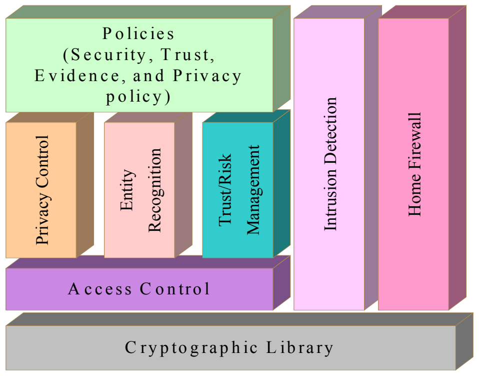


Fig 4.1 USEC Infrastructure Components

Entity Recognition is a novel authentication technology for ubiquitous computing paradigm. In USEC architecture, Entity Recognition supports flexibly various devices such as Smart Badges, iButtons, Smart Watchs, PDAs. This component integrates different type of authentications, ranging from conventional authentication approaches (Username/Password, PKI, Kerberos, etc) to emerging identity recognition technology. The Resurrecting Duckling [4.2] security policy model is an example of entity recognition; ducklings know that their mother is the first entity who sent the imprinting key when they were born.

Trust/Risk Management [4.1] provides trust value to the Access Control Manager. It supports trust collaborations and interactions which usually occur among roaming entities. By modeling trust relationships in smart spaces environments, unknown entities from different domains can interact, request services and resources from a given domain in secure and privacy manner. Risk evaluator and Trust value cooperate with each other to make decision of entity interaction. Hybrid Access Control (HAC) [4.3] is the core part of USEC infrastructure. This is hybrid of Role-based (RBAC), Context-based (CBAC) and Trust-based Access Control (TBAC) to solve different shortcomings of RBAC, CBAC, and TBAC. HAC is critical to preserve confidentiality and integrity. Conventionally, the condition of confidentiality requires that only authorized users can read information, and the condition of integrity requires that only authorized users can alter information and in authorized ways. In USEC, HAC extends scopes of users by using Trust/Risk Management. By doing this, it supports not only predefined entities but also un-predefined entities. Privacy Control is integral part in this convenient but obtrusive environment. It provides *location privacy, anonymous connections* and *confidentiality* of information to users. In USEC infrastructure, we also integrate Home Firewall [4.4] to protect smart space against potential outside attackers. Fig 4.2 shows the relationships and interactions among these components.

Intrusion Detection System is deployed in order to defend against unauthorized access and who has legitimate access to the system but abuse privileges. In ubiquitous environments, this usually occurs due to ubiquity and wireless communication of the systems. In the sensor network layer, USEC provides a lightweight cryptography mechanism in order maintain secure communication among sensors and between sensors and context-aware systems [4.5]. Trust/Risk Management, Intrusion

Detection System, Home Firewall, and Sensor Network Security are together supports Entity Recognition.

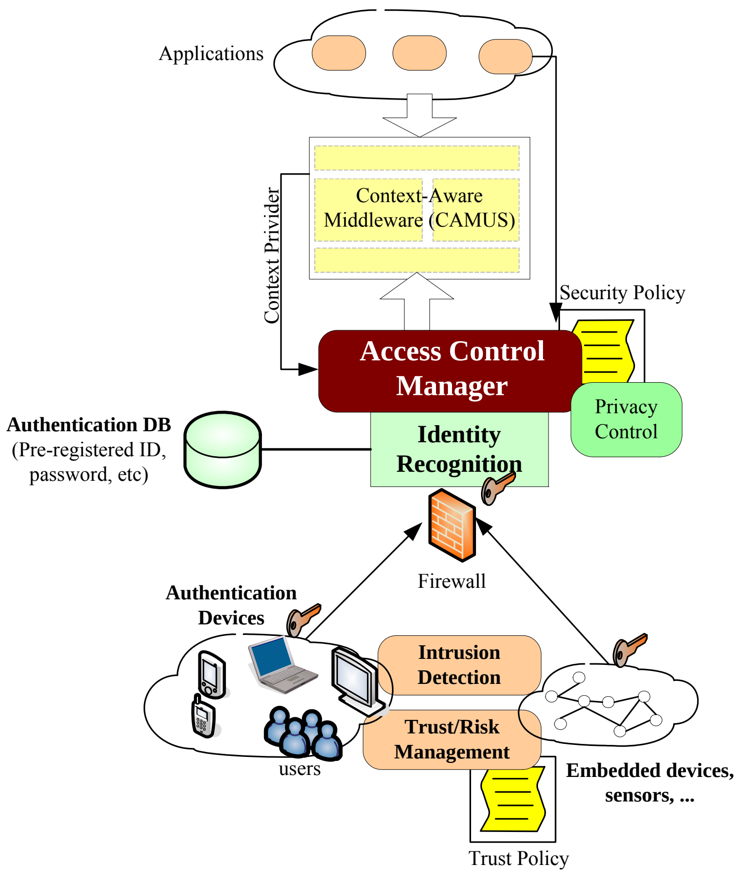


Fig 4.2 USEC Architecture and Its Component Interactions

In the following sections, we focus on important components but have not paid much effort by researchers. We will discuss about its current security problems and propose efficient solutions.

C h a p t e r   5

HYBRID ACCESS CONTROL

5 Hybrid Access Control

5.1 Introduction

Ubiquitous Computing (*ubicomp*) is a revolution of computing paradigm that promises to have a profound affect on the way we interact with computers, devices, physical spaces and other people [5.1]. This new technology envisions a world where embedded processors, computers, sensors, and digital communications are inexpensive commodities that are available everywhere. This eliminates time and place barriers by making services available to users anytime and anywhere. Ubicomp will surround users with a comfortable and convenient information environment that merges physical and computational infrastructures into an integrated habitat. This habitat will feature a proliferation of hundreds or thousands of computing devices and sensors that will provide new functionalities, offer specialized services, and boost productivity and interaction. Context-awareness will allow this habitat 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.

However, the *ubiquity* and *invisibility* characteristics of ubicomp itself arises many security problems, especially in the area of resources/services access control. There are several important issues that needed to be addressed in access control design:

- Ubicomp environment is composed of large scale of users, mobile devices, as well as services. Therefore, determining user's access privilege to resources must be based on *role* or *group of role*, instead of individuals.
- *User's context* (e.g. user location, user need, etc) and *system's context* (e.g. CPU usage, network bandwidth, etc) in ubicomp environments dynamically changes over time. Authorization of user's access based on such contextual information is required for proper enforcement of the required policies.
- In context aware environments users might not know what credentials needed to provide to access a specific service. In that case, the delivery service must support some interacting mechanism to communicate with users to acquire explicitly what kinds of credentials that the user has to show in order to access a certain resource.
- There are hundreds to thousands of entities roaming across different domains in ubiquitous network. These entities are usually unknown by the system that they are located in. Traditional access control approaches purely deny every request from such entities to protect the local systems. Therefore, access control for ubicomp environments must support all of these situations in order to provide *ubiquity* for users (i.e. users can access resources and services in anywhere at anytime).

Lots of work has been done in the area of access control so far. Most of these works have followed four main approaches: Role-based Access Control (RBAC), Policy-based Access Control (PBAC), Context-based Access Control (CBAC), and Trust-based Access Control (TBAC) [5.2-5.10]. However, each of these approaches itself can not fulfill security requirements of ubicomp environments, as we point out in the next section. Therefore, we propose a Hybrid Access Control (HAC) model to tackle the problems of these approaches while takes their major advantages. HAC adopts contextual information as principal design of access control rules. Whenever user's context or system's context changes, user access privileges will be dynamically changed. In addition, HAC applies the models of *deduction* and *abduction* to support flexible interaction with users. Trust management is also integrated to provide services to users based on trust level.

The rest of this section is organized as follows. In the Section 2, we discuss about access control's state of the art, mainly focus on four approaches RBAC, PBAC, CBAC, and RBAC. We also point out the shortcomings of these models. In Section 3, we present our access control architecture, including its overview, workflow, and some examples in reality to explain how it works.. Section 4 discusses about our approach and its promising benefit. Finally, Section 5 concludes the paper and outlines our future work.

5.2 Access Control Approaches for Ubicomp Systems

5.2.1 Role-Based Access Control (RBAC)

Role-based Access Control (RBAC) is a technology that is attracting a great deal of attention, particularly for commercial applications, because of its potential for reducing the complexity and cost of security administration in large network applications [5.11]. It is determined by premise that most real world access control decisions are determined by a person's position or job title within an organization. The fundamental of RBAC is concept of *role*. Basically, a role is a grouping mechanism that is used to categorize *subjects* (i.e. individual users) based on various properties. Another two fundamental concepts of RBAC is *object* and *action*. An object is any resource/service in a system that a subject can access to. An action is a serial of access to a set of one or more objects. Actions can be as simple as reading a file in a home computer, or can be as complex as aiming and firing a missile from a Navy destroyer. Fundamentally, a subject relates to an object through an action as illustrated in Fig 5.1.

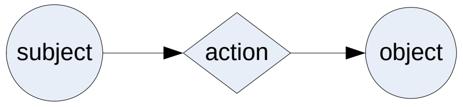


Fig 5.1 Fundamental relationship between a subject, object, and action in RBAC

Though the core of RBAC is very simple, its policy in practice is quite complex and raises many problems. Two major problems are *separation of duty* and *role precedence*. Separation of duty is a case in which the set of access privileges granted by multiple role possession can cause a conflict of interest. For instance, a student can view his academic record. What if he is also a staff of school administrative office? With the privilege of both "student" and "administrative staff", he can perform some illegal actions for instance he can modify his score as his desire. Role precedence stems from inconsistent rules between two roles that a subject posses. For example, Bob is authorized to own "professor" and "faculty". Supposed that professor role is authorized to access and modify data in a computer, but faculty role is prohibited to do so. Thus conflict between "professor" and "faculty" occurs.

So far there have been lots of works published in the area of role-based access control for difference types of applications including traditional computer networks, mobile grids, and ubiquitous computing systems [5.2-5.4]. Jason Reid *et al* [5.3] present a novel model based on RBAC to protect privacy in distributed health care information systems. In this paper, the authors argue that RBAC does not support policy for this application's type with sufficient flexibility and propose a novel adaptation of RBAC principal to address this shortcoming. The main contribution of this work is to propose a modified RBAC in which a set of privileges held by a role can be allowed or denied to another role without using traditional RBAC constraints concepts such as static and dynamic separation of duty.

In [5.4], Covington *et al* proposes a *Generalized Role-Based Access Control* model (GRBAC). GRBAC is an extension of the traditional RBAC model for securing application in the highly connected home as well as in other environments. The major benefit of this model is its combination of usability and expressiveness. It solves the problems in RBAC approach by introducing three different kinds of role: *subject roles* (e.g. 'adult', 'child'), *object roles* (e.g. image, source code, streaming video), and *environment roles* (e.g. daytime, nighttime). However, the model is not a complete security solution itself. To be useful in real-work environment, it must be integrated carefully into a trusted computer system. On the other hand, it proposed policy language based on logic appears to be too simplistic.

5.2.2 Policy-based Access Control (PBAC)

Policy-based network management already requires a paradigm shift in the access control mechanism (from identity-based access control to trust management and negotiation), but this is inadequate for across domain autonomic communications. For many services, no partner may guesses a prior what credentials will be sent by the clients and the clients may not know a prior which credentials are required in order to get an access privilege to a certain services. In [5.12], the authors propose a new concept of *interactive access control* to tackle such problems. This paper discusses about the following three operations for making decision and defining policies:

- ***Deduction***: Deduction is the default process that most access control techniques employ. In deduction, given a defined policy and a set of credentials provided by the client, it is decided whether to grant the access. It checks if a request for access can be permitted or not.
- ***Abduction***: In abduction, given a defined policy and a request to access some resource or service, it is decided what minimum credentials are required so that the given request can be allowed. This process is specifically useful in context aware environments where the client might not know what credentials it needs to provide to access a specific service. In that case, the delivery service will request the client to provide the missing credentials and if the clients provide valid credentials, the request is granted, else denied.
- ***Induction***: Induction utilizes a heuristic function along with some positive examples of scenarios in which the request should be granted and some negative examples of scenarios where the request should be denied. Basing on this information, the induction process tries to identify the policy that satisfies the validity of the granted requests. The induction process is useful in the case where a single static policy can not be defined. This is true for context aware systems since the context changes at run-time so should the policy to incorporate the new situation.

By utilizing these three operations, an interactive and autonomous access control mechanism can be put in to effect. It improves the chances of service matching by allowing the client to provide missing credentials instead of plainly denying the request [5.5]. In our hybrid access control model, we integrate the deduction and abduction functions to increase the usability of the system. Induction is left for future implementation due to its complexity and high overhead causing to the system.

5.2.3 Context-based Access Control (CBAC)

The underlying technology of ubiquitous computing systems is context-awareness. Access control to ubiquitous services calls for novel solutions based on various context information e.g. user location, device properties, user needs, local resource visibility. Context-based Access Control (CBAC) has become well suitable for such ubiquitous computing environments since this approach adopts context as a design principle to rule access to resources. CBAC dynamically grants and adapts permissions to users according to current context. For instance, if the user is in the presentation room and using the projector, then he will be allowed to use the computer as well without any credential provision because the system recognizes him as a presenter.

A. Corradi *et al* proposes a novel model of context-based access control, called UbiCOSM (Ubiquitous Computing Context-based Security Middleware) [5.6]. UbiCOSM uses the context as a foundation for security policy specification and enforcement processes. Unlike traditional access control models, permissions are directly associated with contexts, instead of user identities/roles: any mobile user/device acquires a set of permissions by entering a specific context. Instead of managing subjects and their permissions individually, a system administrator defines for each context the set applicable permissions. When a subject operates in a specific context, she instantaneously acquires the set of permissions active for the related context. When she changes her operating context, her previous permissions are automatically revoked and new permissions acquired.

G. Zhang *et al* [5.7] extends the role-based access control model to provide dynamic context-aware access control for ubicomp systems, called DRBAC. DRBAC addresses two keys requirements that (1) A user's privileges must be changed when the user's context changes and (2) A resource must adjust its access permission when the system's context (e.g. CPU usage, network bandwidth, memory usage) changes. The model dynamically adjusts *Role Assignments* and *Permission Assignments* based on context information. In this approach, each user is assigned a role subset (by the authority service) from the entire role set. Similarly the resource has permission subsets for each role that will access the resource. During a secure interaction, state machines are maintained by delegated access control agents at the subject (*Role State Machine*) to navigate the role subset, and the object (*Permission State Machine*) to navigate the permission subset for each active role. The state machine consists of state variables (role, permission), which encode its state, and commands, which transform its state. These state machines define the currently active role and its assigned permissions and navigate the role/permission subsets to react to changes in the context.

Another work proposed by A. Tripathi *et al* [5.8] is system architecture to support ubiquitous computing for mobile users across different environments by transparently performing context-based discovery and binding of resources. The focus of this paper is on policies for resource access in ubiquitous and context-aware ubicomp environments. The environment is able to proactively discover and transparently bind the resources required by the user. And the environment is able to negotiate the security policies specified by the user and the security polices associated with the resources. The goals of this paper are similar to many of the current research activities in the field of ubiquitous computing [5.13].

5.2.4 Trust-based Access Control (TBAC)

For decades, Trust Management has been well investigated by many researchers in different fields such as computer networks, Internet, Mobile Grid, etc. However, applying trust in security for ubiquitous computing systems just started several years ago. Few works that adopts the notion of trust for access control mechanism has been done and the others still are in progress [5.14-5.16]. Ubiquitous computing environments lead to new security-challenges for trust collaboration between participants because these participants usually are roaming across domains and are unknown by the system where they are located in [5.14]. TBAC is the idea of using the human notion of trust and community as a principal concern for assigning privileges. This means that the user can be granted some access permissions without any credentials if he is trusted at some level by the system. The most challenge of TBAC is how to define mathematics model of trust, how much of trust is enough to grant certain permission. For instance, how much trust do I need in my colleague to let him borrow my PDA?

The SECURE (*Secure Environments for Collaboration among Ubiquitous Roaming Entities*) project [5.15] investigates the design of security mechanism for ubiquitous computing based on trust. It addresses how an entity in and unfamiliar ubicomp environment can overcome initial suspicion to provide secure collaboration. SECURE introduces a novel model of autonomic decision-making based on economic theory. In SECURE, the access control manager grants or denies permissions of a *principal* to execute an *action* based on trust value in the principal and the cost if the outcome occurs. An explicit cost-benefit analysis is used to determine how much trust is required to offset the risk. While the trust framework is to calculate a trust-value for a *principal* p, the access control manager looks up the outcome costs for the action and checks any specified environmental constraints (for example, time of day), then evaluates a series of predicates which compare trust-values to costs.

William J. Adams *et al* [5.16] proposes a decentralized trust-based access control for dynamic collaboration. The goal of this research is to create a decentralized trust-based access control system for a mobile ad-hoc collaborative environment. User permissions can be determined and assigned by using behavior grading without the need for pre-configured, centrally managed role hierarchies or permission sets. The system provides trust-based access control suitable for deployment in a rapidly assembled, highly fluid, collaborative environment.

5.2.5 Summarized Shortcomings of Existing Approaches

We have introduced four main different approaches of designing access control mechanism for ubiquitous computing environments. Each approach has its own advantages and disadvantages. RBAC is used in large scale organization for scalability and managing complexity. However, it can not enforce fine-grained access control in a context-aware environment where users are not predefined and the access rights and constraints are dynamic. Though [5.4] proposed GRBAC in order to tackled problems in RBAC, but still it does not fit well to ubiquitous environments where *subject* and *object* are highly mobile and usually unknown to the system. CBAC is an extension of RBAC that grants access privileges based on current context of users and the system. Though its advantage is to provide a flexible security mechanism according to context change, a scalability problem arises in which a vast amount of resources needed to be monitored and adjusted permission regularly. In PBAC mechanisms, the communicating parties agree on some *policy* to allow access to resources/services. The shortcoming of this approach is that policies are not dynamic and can not accommodate the changing security requirements of a context aware system with time. TBAC seems to be more applicable for ubiquitous environments. [5.15,5.16] are examples of applying trust in access control mechanism. However, TBAC is not scalable and flexible for making decision if context-awareness and interaction with users are not complemented.

Motivated by such shortcomings, we introduce Hybrid Access Control to tackle these problems while takes advantages of each existing approach to provide a flexible and scalable access control mechanism for ubiquitous computing environments.

5.3 Hybrid Access Control (HAC) Architecture

5.3.1 HAC Overview

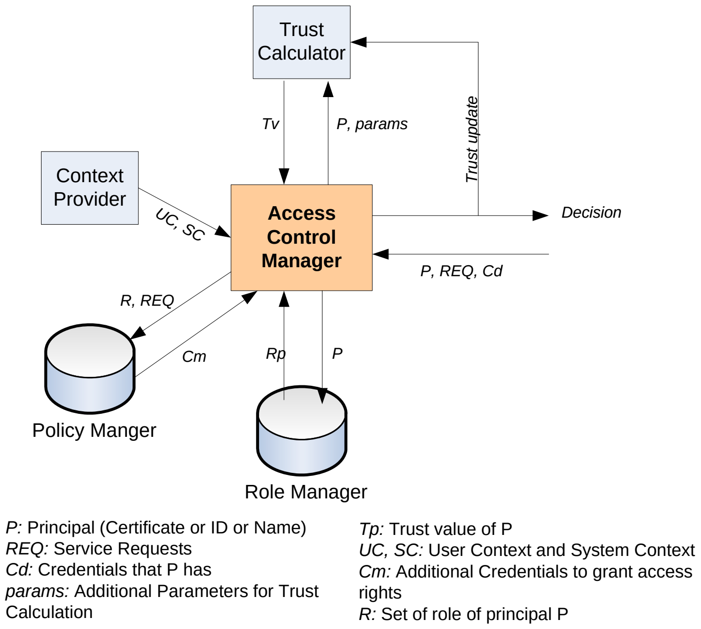
Our Hybrid Access Control (HAC) model, as its name suggests, is hybrid of RBAC, CBAC, PBAC and TBAC. By applying *role*, *context*, *policy*, and *trust* in *decision-making* mechanism, HAC fulfills security requirements and solves the shortcomings of RBAC, CBAC, PBAC, and TBAC approaches.

Fig 5.2 depicts HAC model including *Trust Calculator (TC), Context Provider (CP), Policy Manager (PM), Role Manager (RM),* and the core part *Access Control Manager (ACM)*. Trust Calculator is responsible for calculating trust value on principal P. When it receives a request from Access Control Manager (ACM) along with principal (*P*) and additional parameters (*params*), it computes the trust value on P (T_V) based on a our trust model [5.10]. In this trust model, we use a vector of trust values composed of four elements: *Peer Recommendation (PR), Confidence level (CF), History of Past Interaction (PI), and Time-based Evaluation (TE)*. The trust value of Q_i on Q_j (t_{QiQj}) is computed as:

$$t_{Q_i,Q_j} = \frac{w_1\left(PR_{Q_i,Q_j}\right)\left(\frac{CF_{Q_i,Q_j}+TE_{Q_i,Q_j}}{2}\right)+w_2\left(PI_{Q_i,Q_j}\right)}{\sum_{i=1}^{2} w_i} \tag{1}$$

where w_i is weighted values which can be adjusted to meet different security requirements of deferent systems.

After the ACM made a decision either allow or deny the request, it sends a feedback to Trust Calculator to update *History of Past Interaction* value.



Trust Calculator
Tv
P, params
Trust update
Context Provider
UC, SC
Access Control Manager
Decision
P, REQ, Cd
R, REQ
Cm
Rp
P
Policy Manger
Role Manager
P: Principal (Certificate or ID or Name)
REQ: Service Requests
Cd: Credentials that P has
params: Additional Parameters for Trust Calculation
Tp: Trust value of P
UC, SC: User Context and System Context
Cm: Additional Credentials to grant access rights
R: Set of role of principal P

Fig 5.2 Hybrid Access Control Model

Context Provider module provides *user's context* (UC) and *system's context* (SC) to ACM. User's context here includes location and time that the user accesses resources or uses services. This information serves as implicit credentials which are evaluated against the activity policies to access a given resource/service. System's context involves system information such as CPU usage or network bandwidth. System's context helps in selecting policy applicable in the given situation from a policy pool.

Policy Manager maintains *system policies* and *service policies*. System policy deals with the permission over resource access on the system level. For example, when to generate alarms in case of emergency situation. In a context-aware environment, we define multiple level policies and the most applicable policy basing on current context is chosen from that pool and applicable as long as the system context doesn't change or an exception doesn't occur. By selecting basing on context, we can have dynamic policy-based access control mechanism. The most important attributes of a policy are "*Triggering Context*" and "*Terminating Context*". These two attributes help the system in finding the most suitable policy for the current system context. Service policy is local policy of each service. It defines what action can be performed on the service and who is authorized to perform them. These policies can be static or dynamic depending on the particular service security's requirements.
Role Manager stores *client roles* in hierarchical structure. Using role hierarchical structure can deal with *separation of duty* and *role precedence* problems of traditional RBAC. Role hierarchical structure helps

manage role complexity to exploit commonality among roles. For example in an university, all professors have certain set of role "faculty" even though they belongs to different departments. Here we classify *client role* into *subject roles* and *object roles*. Subject role is analogous to traditional RBAC approach. A subject can be a user, a mobile device, or even an application which is attempting to access services/resources. A subject can have more than one role assigned to it. For example a person may have "student" role and "graduate office staff" role as well. *Object* here includes services, resources that a subject can access to. Object role can be classified based on any classifiable properties of object, such as object type, sensitive level, or even the context of the object. By using object role, we can easily structure accessing policy according to the properties of resources/services. In HAC, Role Manager maintains accessing rules by using contextual graph structure as an example in Fig 3. In this, subjects and objects connect with each other through *action*, i.e. particular access privilege that a subject can use to operate on an object. For instance, a employee can view the working schedule only, but he can not modify it (*modify* here includes *add, update,* and *delete* records). However, a foreman can view as well as modify it, plus he can view salary record and employee resume as well.

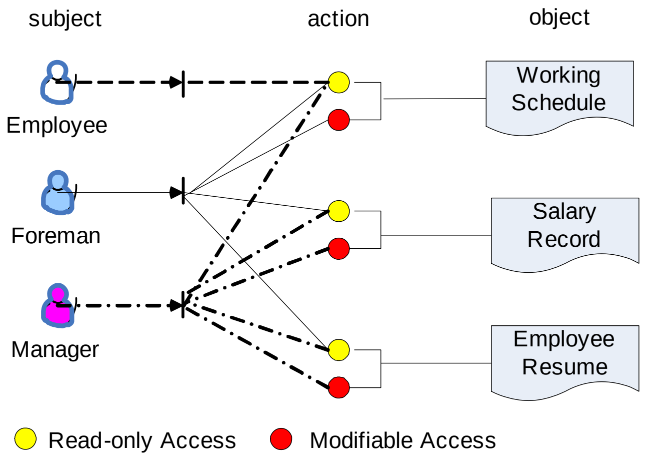


Fig 5.3 An example of accessing rules by using contextual graph structure.

5.3.2 HAC Workflow

Fig 5.4 depicts the workflow of HAC including three major operations: *deduction*, *abduction*, and *trust comparison*. When a user sends a service request (R_S) to the ACM along with his credentials (C_P), the ACM firstly perform *deduction* operation (step 1, written in the circle). It evaluates the service request by using policy rules from Policy Manager and context (*user context UC* and *system context SC*) from Context Provider, and it makes a decision whether this request is permitted or not. For instance, Bob is in his office. The system recognizes Bob (by some authentication method) and automatically assigns him the role "professor". Now he sends a request to access the fax machine. There is no one using the fax machine, i.e. system context is ok. Therefore, according to policy rules, his service request is definitely allowed. One moment later, he leaves his office. By tracking his location, the system knows that his context has been changed. Thus it automatically switches his role to "faculty". Doing this makes Bob not able to access fax machine in other professor's offices because his role no longer is "professor".

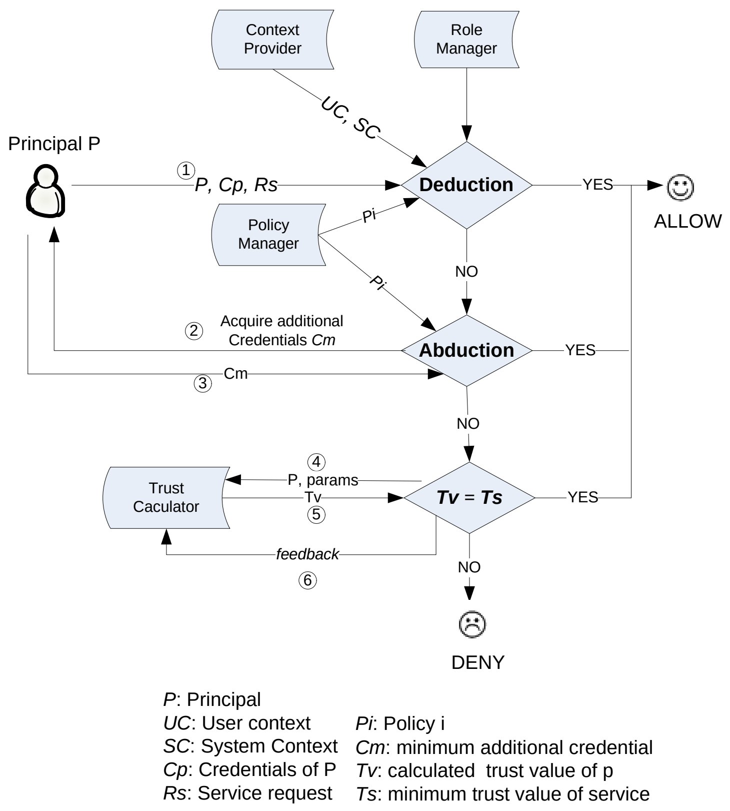


Fig 5.4 Hybrid Access Control workflow

If the service request is not allowed due to limited privileges of his role and credentials, the system will pass this request to *abduction* operation. By checking the request and system policies, the abduction operation is performed to find the minimum additional credentials (C_m) that the user must provide more in order to get access permission to given services/resources. This additional credential requirement is sent to the user (step 2). If the user provides such credentials, then there is no problem for him to access the resource (step 3). For instance, Bob now comes to the training bureau and he wants to access to the teaching schedule in order to assign some classes to faculty members. However, with "faculty" role, his request is not permitted. The system then explicitly asks him for "dean", or "dean's secretary" credential. He sends the "dean" credential to the system. Now he is free to modify the teaching schedule.

So far we have presented the cases wherein the user is known by the system and he can provide sufficient credentials to the system either by implicitly or explicitly manner. However, such cases do not cover all the circumstances in ubiquitous computing environments. Usually, the user is unknown by the system and he is not able to provide such required credentials that the system requests. HAC deals with this problem by supporting Trust Comparison operation. Trust Calculator computes trust value on this principal (T_V) based on recommendations of other principals, history of its interaction with the system, and other factors as mentioned above. It then passes this value to Trust Comparison (step 5). If the trust value exceeds the predefined trust value of given service/resource, he will be permitted to access to that service/resource, otherwise denied. After making decision, ACM sends a feedback to Trust Calculator to update interaction information (step 6). We give an example by assuming that Bob now comes to his friend's company, Alice. Since Alice is attending meeting, so he has to wait in her office. To save time, he uses his laptop to print some documents and he also wants to fax his report to his department. He has never registered with Alice's office system, so obviously the system does not know Bob. Also, he can not respond any credential request from the system. Therefore the system cannot grant him any access privilege. Bob sends a request to Alice's PDA to ask for a recommendation. In addition, by checking history of interaction, the system knows that Bob used to use the printer. Gathering all this information, the system computes trust value on Bob as T_V = 0.7. We assume that the printer has assigned trust threshold $T_{printer}=0.6$ and the fax's is $T_{far}=0.8$ (i.e. only when the system trusts the principal with the trust value is not less than *0.6* (*0.8*), it permits this principal to access printer (fax machine). Since $T_{faxr}>T_V > T_{printer}$, Bob is allowed to use the printer only, but not the fax machine.

5.4 Discussion

The major goal of our hybrid access control model is to tackle problems of existing approaches while takes their advantages. Our target is to provide a flexible and scalable access control mechanism for ubiquitous computing environments.

Controlling user's accesses to resources based on role is the foremost important and widely used approach. By adopting role hierarchical structure, HAC can reduce the complexity and cost of security administration. On the other hand, role hierarchical structure helps to deal with *separation of duty* and *role precedence* problems in RBAC. HAC also uses context as principal design to dynamically assign and adapt permissions to users. Whenever the context changes, role of user will be changed. By doing this, HAC provides flexible, convenient, and high secure accessing to resource. In addition, we adopt two fundamental operations of Interactive Access Control approach, *Deduction* and *Abduction*. Deduction and abduction operations help users specify implicit as well as explicit credential requirements of the systems. It deals with the problems in ubiquitous computing environments that a user usually does not know a *prior* what kinds of credential needed to provide so that he can access to certain resources. In this case, he is requested by the systems for additional credentials so that the systems can provide as many services as possible to him. Last but not least, trust is indispensable factor that ubiquitous computing systems must take into account in architecture design. In HAC, trust management is involved as the final operation to deal with the problems of *uncertainty* between different domains (i.e. users from foreign domains are *unknown* to the system and traditional security mechanism purely denies every requests from such users for securing the system). By applying trust management in to our access control mechanism, we want to provide to *unknown* users as many services as possible even though they can not show evident credentials.

5.5 Conclusion and Future Work

We have presented the Hybrid Access Control (HAC), a flexible and scalable access control model for ubiquitous computing environments. HAC is a hybrid of Role-based Access Control, Policy-based Access Control, Context-based Access Control, and Trust-based Access Control. The major benefit of HAC over existing access control models is its combination of flexibility and scalability. HAC takes into accounts most of important factors that are role, context, interactive, and trust to provide more services and resources to users in applicable and secure manner.

Though HAC was introduced as a promising approach for ubiquitous computing environments, it is not a complete security model itself. There are many work that we are still working on. At this stage, we are using Prolog to define policy specification for HAC, but we will extend to RDF [5.17] in our future work. Currently, we are developing a Trust-based Security Architecture for Context-aware Systems (USEC) [5.18] for our CAMUS context-aware middleware [5.19]. After completing our architecture design, we will implement HAC as a core part of USEC.

5.6 Reference

[5.1] M.Weiser, “Hot Topics: Ubiquitous Computing” IEEE Computer, 1993.
[5.2] David F. Ferraiolo, D. Richard Kuhn, R. Chandramouli. Role-Based Access Control. Artech House, INC 2003
[5.3] Jason Reid, Ian Cheong, Matthew Henricksen, Jason Smith. A Novel Use of RBAC to Protect Privacy in Distributed Health Care Information Systems. 8th Australasian Conference on Information Security and Privacy, Australia July 9-11, 2003 (LNCS volume 2727 pp. 403-415)
[5.4] M. J. Covington, M. J. Moyer, and M. Ahamad, "Generalized Role-Based Access Control for Securing Future Applications," 23rd National Information Systems Security Conference, 2000.
[5.5] Maria Riaz, Saad Liaquat Kiani, Sungyoung Lee, and Young-Koo Lee. Incorporating Semantics Based Search and Policy-Based Access Control Mechanism in Context Service Delivering. IEEE Fourth Annual ACIS International Conference on Computer and Information Science (ICIS 2005),14-16 July 05, Jeju - Korea, pp.175-180
[5.6] A. Corradi, R. Montanari, and D. Tibaldi, “Context-based access control management in ubiquitous environments,” Proc. Third IEEE International Symposium on Network Computing and Applications, (NCA’04), pp.253–260, Aug. 2004.
[5.7] G. Zhang and M. Parashar. Context-aware dynamic access control for pervasive computing. In 2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04), San Diego, California, USA, January 2004.
[5.8] Anand Tripathi, Tanvir Ahmed, Devdatta Kulkarni, Richa Kumar, and Komal Kashiramka. Context-Based Secure Resource Access in Pervasive Computing Environments. In Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW’04) 2004.
[5.9] Adams, W.J.; Davis, N.J., IV. Toward a decentralized trust-based access control system for dynamic collaboration. The IEEE Workshop on Information Assurance and Security, NY, 2005.
[5.10] Hassan Jameel, Le Xuan Hung, Umar Kalim, Ali Sajjad, Sungyoung Lee and Young-Koo Lee. A Trust Model for Ubiquitous Systems based on Vectors of Trust Values. 3rd International IEEE Security in Storage Workshop San Francisco, California USA, December 13, 2005
[5.11] David F. Ferraiolo, D. Richard Kuhn, R. Chandramouli. Role-Based Access Control. Artech House, INC 2003
[5.12] Koshutanski, H., Massacci, F.: Deduction, Abduction and Induction, the Reasoning Services for Access Control in Autonomic Communication. In: proceedings of the 1st IFIP TC6 WG6.6 International Workshop on Autonomic Communication (WAC 2004), October 2004, Berlin, Germany. Springer, 2004
[5.13] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Toward Distraction-Free Pervasive Computing. IEEE Pervasive computing, 1(2):22–31, April-June 2002.
[5.14] Dimmock, N.: How much is `enough'? Risk in trust-based access control. In: IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises - Enterprise Security. (2003) 281-282
[5.15] Cahill, V., et al.: Using trust for secure collaboration in uncertain environments. IEEE Pervasive Computing 2 (2003) 52-61.
[5.16] Adams, W.J.; Davis, N.J., IV. Toward a decentralized trust-based access control system for dynamic collaboration. The IEEE Workshop on Information Assurance and Security, NY, 2005.
[5.17] Resource Description Framework (RDF), http://www.w3.org/RDF/
[5.18] Hassan J., Le Xuan Hung, N.N. Diep, P.D. Giang, Riaz A.S., SY Lee, T.V. Phuong, Y-K. Lee, and Y. Weiwei. Trust-based Security Infrastructure for Ubiquitous Computing Environments. Technical Report, Real-time and Multimedia Lab. Department of Computer Engineering, Kyung Hee University, Korea
[5.19] Anjum Shehzad, Hung N. Q., Kim Anh P. M. , Maria Riaz, Saad Liaquat, Sungyoung Lee, and Young Koo Lee. Middleware Infrastructure For Context-Aware Ubiquitous Computing Systems. Technical Report. Real-time and Multimedia Lab. Kyung Hee University, Korea.

C h a p t e r   6

PLUGGABLE RECOGNITION MODULE

6 Pluggable Recognition Module (PRM)

6.1 Introduction

The Ubicomp paradigm foresees communicating and computational devices embedded in all parts of our environments, from our physical selves to our home, our office, our streets and so forth. In this new paradigm devices will need to interact, almost spontaneously, with certain other devices in an environment that is both unknown and changing. In traditional approaches, the interaction of two (or more) devices is secured by an authenticated key exchange, where authentication usually means entity authentication. However, we feel that security, based on entity authentication, is likely to be inadequate in the pervasive computing paradigm, for two principle reasons:
Names of entities will probably be unknown – a rather fundamental obstacle for entity authentication!

Authenticating an entity (supposing that its identity can be reliably determined) is not likely to give us much confidence about what that device will do.

6.2 Pluggable Recognition Module (PRM)

In computer security, *Authentication* is defined as the process by which a computer, computer program, or another user attempts to confirm that the computer, computer program, or user from whom the second party has received some communication is, or is not, the claimed first party. In Ubicomp environment where there are various devices and users, the condition of authentication requires more than that. It has to support many kinds of devices and users with different authentication modules. On the other hand, it has to offer a balance between authentication strength and non-intrusiveness. For example, a smart badge that transmits a short range signal is a good non-intrusive authentication mechanism, but provides a week form of authentication. Motivated from this, Jalal A. *et al* proposed a GAIA Pluggable Authentication Module (GPAM) scheme for smart spaces [6.1]. Conventionally, PAM 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 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. However, as we mentioned earlier, this approach only supports entities which have been registered to the system. This can not deal with unknown entities comes from other domains that always occurs in ubiquitous environment.

We extend GPAM by providing a Pluggable Recognition Module to resolve this weakness. PRM can incorporate with conventional authentication schemes such as Digital Signature, Username/Password, Kerberos, etc or advanced entity recognition schemes such as Resurrecting Duckling, μTESLA, ZCK, etc. Our PRM architecture is depicted by Fig 6.1.

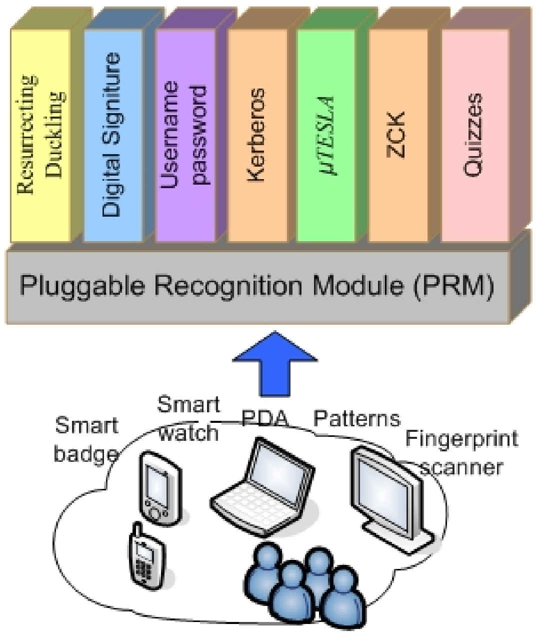


Fig 6.1 Pluggable Recognition Module

6.3 References

[6.1] Jalal Al-Muhtadi, Anand Ranganathan, Roy Campbell, and M. Dennis Mickunas. Cerberus: A Context-Aware Security Scheme for Smart Spaces. In IEEE International Conference on Pervasive Computing and Communications (PerCom 2003), Dallas-Fort Worth, Texas, March 23-26, 2003

C h a p t e r 7

TRUST MANAGEMENT

7 Trust Management

7.1 Introduction

Ubiquitous computing premises a massive network world supporting diverse but cooperating mobile entities where autonomous operation is necessary due to lack of central control. Due to the highly dynamic and unpredictable of ubiquitous computing environments, there are several problems in the ubiquitous security. First, traditional authentication and access control are only effective if the system knows in advance which user is going to access and what their access rights are. Secondly, security information in different domains is subject to inconsistent interpretations in such an open, distributed environment. Thirdly, portable handheld and embedded devices have severely limited resources.

Based on the above security challenges, computational models of trust have been proposed for use in ubiquitous environments to decide whether to provide service to the service requestor. Trust is the belief or willingness to believe an entity based on its competence (e.g. goodness, strength, ability) and behavior within a specific context at a given time. It is an area of study in which people with various backgrounds have tried to base their own views on their own circumstances and backgrounds. It is subjective and situation specific. Also trust in one environment does not transfer to another environment. So we need a method to measure trust. Translation of this subjective concept into a language understood by computing entities is the main objective needed to be solved. At the same time, when we evaluate the trust, prior probability is obviously very important that should be involved.

7.2 Related work

Since mid '90s the research community has outlined the key role of trust management models to develop more complex and dependable computer systems. From this, the importance of trust model was first highlighted by Blaze et al in their seminal paper [7.1]. Subsequently, Josang [7.2] presented an interesting classification of trust relationships and its implication to traditional security concepts. Until now, several trust models have been proposed in the literature for different distributed systems [7.3]. For the Grid scenario, X.509 [7.4] and SPKI [7.5] seem adequate which propose a central Certificate Authority (CA) based trust model. However, there are a number of issues related to proxy/delegation certificates that are serious drawbacks of these models. A two-level trust model for Grid based on graph topology was proposed in [7.6]. They use different trust evaluation metrics for centralized grid domains and distributed Virtual Organizations (VO). A peer recommended trust model was proposed in [7.7] for ubiquitous computing systems. Their trust management scheme through recommendation lacks certain aspects such as the weighted recommendation of peers based on their prior interactions.

In [7.2], a decentralized trust and reputation model for multi agent systems has been proposed whereas a probabilistic trust model is proposed in [7.8] for mobile agents. Both these models lack a fundamental requirement, i.e., very old recommendations should not be relevant in predicting the behavior of an entity. Another probabilistic trust model called the Beta Reputation System (BRS) [7.9] works by giving ratings about other users in the system. All these trust models can be generally categorized into probabilistic models and others in which the trust evaluation formulae are tuned to give the desired result. In the fields of Ubiquitous Computing, research has paid much attention to build autonomous trust management as fundamental building block to design the future security framework. Up to now, research has focused mainly on the propagation and composition of trust information [7.10, 7.11, 7.12, 7.13] while paying less attention to how direct trust information is actually built. Though focused on distributed trust computation, [7.14, 7.15] face the problem of building trust from past experience. Michiardi et al [7.16] proposed an organic reputation-based framework to enforce collaboration in ad-hoc networks. Peer reputation is built by evaluating a mix of directly collected information, undirected feedback, and eventually multiple interaction classes.

7.3 Example scenario

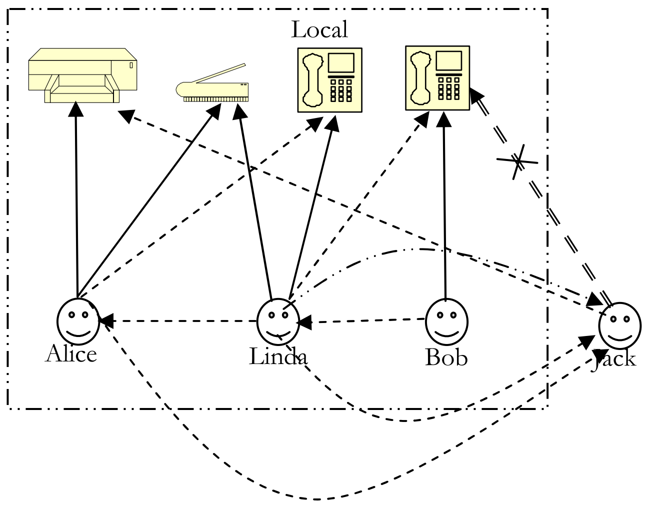


Fig 7.1 example scenario

One of the example scenarios which can use the trust model mentioned in this paper is as follows:

As showed in figure 7.1, consider in a Smart Office Building, everyday there are lots of people working in the building or drop by for business. Many services are available in this building, e.g. using the copy machine, scanner, telephone to make local call and phone to make international call.

Different user has different rights to use different services, e.g. Alice, the desk clerk, can use the copy machine and scanner. Linda, the secretary, can use the copy machine, scanner and telephone to make local call. Besides these services, Bob, a manager, can also use phone with international service. As showed in Figure 1 with real lines.

If Linda want to use the international telephone to dispose some urgent situation, since she doesn't have the right to use this, she must have recommendation form the users who know her and have used the service before or have the right to use that, like Bob. As showed in figure 1 with dash line. One of the possible results is that her request is accepted and manages to use the international phone call service.

Jack, a business man drops by and never use any of the facilities in this building before, wants to use some service, then he should have the recommendation from the reliable entities in this environment. E.g. he uses the scanner with the recommendation from Alice and Linda, who know him. But if Jack wants to use the international phone call service to dispose some personal issues, he asks Linda, who is also his private friends, to give him recommendation. Even though, Linda has successful used this service before and she strongly recommend Jack, Jack's request to use this service will be rejected, as showed in Fig 7.1 using double dash line. But if Jack has the same recommendation from same person, Linda, to use the local phone service, he will probably success.

Also, if the local phone itself has very unsuccessful communication with Alice before, even though Alice is recommended as trustable by Linda and Bob, the request will probably be rejected. At the same time, if Linda's communication to Alice just took place yesterday. But Bob's was 4 month before. Of course in this case Linda's recommendation will be more useful.

Furthermore the devices in the building are not fixed since the highly dynamic of the ubiquitous environment. Suppose Jack knows 10 people in this building, but 7 of them are not in the building because of business, so the service provider should use the limited information only from 3 people's devices to make a decision as accurate as possible.

7.4 Our Trust Model

7.4.1 The factors Considered in Our Trust Model

Whenever two principals want to interact, they should be able to evaluate the amount of trust on each other using some evaluation metric. This metric should include the recommendations of other principals that had past experiences with these principals; the more the experiences, the higher the weight of these recommendations. Moreover older experiences should have less impact on this evaluation. Finally the interacting principals' past experiences with each other should obviously have a say in this evaluation. These metrics are precisely developed in the following sections.

7.4.1.1 Peer Recommendation

We assume each principal in the system has its own unique identity. Suppose n is the total number of principals in the system. Each principle has a trust value for any other principal it interacted with before. Let Q1;Q2;…;Qn denote the principals in the system. In this section we will model and formulate how to calculate the trust value of a principal requesting some action by asking the principal's reputation from other principals in the system. The other principals might lie and give a false recommendation for some mutual benefit. We will suppose a very reasonable assumption that principals with high trust values will not send false recommendations.

This peer recommendation can be used to calculate how much trust the two principals can put upon each other. The peer recommendation will be higher if the peers have more trust on the principals and vice versa. Thus gives a good idea about the reputation of the two principals. Notice that this value is the same for both the interacting principals. The peer recommendation involves a dot product of vector elements, one of which is the trust that the principal Qi has on the other one and the second one is the trust that the other principal has on Qj Thus if Qi has a low trust value for that principal, then its recommendation will be highly minimized. Consequently, based on our assumption, a principal who gives a false recommendation about Qj will not get any advantage as it will have a low trust value.

7.4.1.2 Confidence

Intuitively, the peer recommendation value calculated above should have a higher weight if the number of peers common to both the interacting principals is higher. Likewise principals with more interactions with a particular principal should have a higher say in recommendation. This introduces the notion of confidence over the peer recommendation value. The confidence level should be a maximum if the number of common peers and the number of individual interactions of these peers are greater than a threshold value.

7.4.1.3 History of Past Interactions

An important factor in deciding this confidence is the history of the past interactions. Two interacting principals should keep in mind their past experiences when calculating the trust value. We can generically define successful and unsuccessful interactions between two principals based on their past behaviors, where an unsuccessful interaction means that the principal has betrayed the trust bestowed upon it. The nature of an interaction might reflect more than just a successful and unsuccessful interaction. For example, a principal might behave totally contrary to the expectations whereas another one might diverge to a lesser extent. However, as this transition is really cumbersome to model and might differ from every principal's perspective, we restrict ourselves to the two outcomes; successful and unsuccessful. Furthermore, the outcome of an interaction might be different in the view of the two principals. What one conceives as a success, the other might regard as a failure.

7.4.1.4 Time based evaluation

Intuitively, very old experiences of peers should have less weight in peer recommendation over new ones. In other words, peer recommendations older than a threshold time interval should have less weight over the others. We can put this desired property in our evaluation model if every principal keeps a time stamp with its latest interaction with every other principal.

7.4.2 Trust Evaluation Metric

Based on the aforementioned metrics, we are now ready to describe our trust evaluation metric. The trust metric is defined as a weighted arithmetic mean of PR, CF, TE, and PI.(peer recommendation, confidence, time based value and past interaction history) More precisely, the trust between two principals Qi and Qj who want to interact can be calculated as:

$$t_{Q_i,Q_j} = \frac{w_1(PR_{Q_i,Q_j})(\frac{CF_{Q_i,Q_j} + TE_{Q_i,Q_j}}{2}) + w_2(PI_{Q_i,Q_j})}{\sum_{t=1}^{2} w_i}$$

where $f_i \in N$ and they can be adjusted to a suitable value if more weight is to be given to a specific metric. For example, past interactions evaluation should be given more importance over the others. The PR value is weighted over CF and TE.

7.5 Conclusion

This model for trust is based on the vectors of trust values of different entities in ubiquitous computing. Distinguished from previous trust model, our trust model takes uncertainty of trust into account with a precise computation model. Besides basic factors of trust computation such as peer reputation, confidence, and history of past interaction. We additionally include time based evaluation factor to calculate trust value and efficiently handle false recommendations. The calculation of the trust depends upon the recommendation of peer entities common to the entities which are weighted according to the number of past interactions and the time of last interaction. The model can calculate trust between two entities in situations both in which there is past experience among the interacting entities and in which the two entities are communicating for the first time. Several tuning parameters are suggested which can be adjusted to meet the security requirement of a distributed system. This highly secure system can adjust these parameters such that entities with high reputation and recommendation are allowed to perform requested actions.

7.6 References

[7.1] J. Blaze, M. Feigenbaum. Decentralized trust management. In Proceedings of the 1996 IEEE Symposium on Security and Privacy, pages 164–173, 1996.

[7.2] A. Josang. The right type of trust for distributed systems. In New security paradigms workshop, pages 119–131, 1996. [11] M. Kamvar, S.D. Schlosser. The eigentrust algorithm for reputation management in p2p networks. In WWW'03: Proceedings of the 12th international conference on World Wide Web, pages 640–651. ACM Press, 2003.

[7.3] N. Huynh, T.D. Jennings. Developing an integrated trust and reputation model for open multi-agent systems. In AAMAS- 04 Workshop on Trust in Agent Societies, 2004.

[7.4] S. Mendes. A new approach to the x.509 framework: Allowing a global authentication infrastructure without a global trust model. In Proceedings of NDSS 95, 1995.

[7.5] C. Ellison. Spki certificate theory. Internet Request for Comments: 2693, 1999.

[7.6] H. Li, T.Y. Zhu. A novel two-level trust model for grid. In ICICS, pages 214–225, 2003.

[7.7] M. Carbone. A formal model for trust in dynamic networks. In International Conference on Software Engineering and Formal Methods (SEFM'03). IEEE, 2003.

[7.8] N. Patel, J. Jennings. A probabilistic trust model for handling inaccurate reputation sources. In iTrust, pages 193–209, 2005.

[7.9] R. Ismail. The beta reputation system. In Proceedings of the 15th Bled Conference on Electronic Commerce, 2002.

[7.10] R. Matthew, R. Agrawal. Trust management for the semantic web, 2003.

[7.11] S. Hung, N. Q. Kiani. Developing context-aware ubiquitous computing systems with a unified middleware framework. In 2004 International Conference on Embedded and Ubiquitous Computing. Springer Verlag, 2004.

[7.12] J. Theodorakopoulos, G. Baras. Trust evaluation in ad-hoc networks. In WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security, pages 1–10. ACM Press, 2004.

[7.13] R. Guha, R. Kumar. Propagation of trust and distrust. In WWW'04: Proceedings of the 13th international conference on World Wide Web, pages 403–412. ACM Press, 2004.

[7.14] R. Michiardi, P. Molva. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, pages 107–121. Kluwer, B.V., 2002.

[7.15] S. Ganeriwal and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. In SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, pages 66–77. ACM Press, 2004.

C h a p t e r   8

Light weight Security Framework for Sensor Networks Layer

8 Light weight Security Framework for Sensor Networks Layer

8.1 Introduction

Wireless networks are relatively more vulnerable to attacks than wired networks due to the broadcast nature of communication [8.1]. Wireless sensor networks which usually consist of a large number of small size sensor nodes deployed in the observed environment. Sensor nodes have smaller memory (8K of total memory and disk space) and limited computation power (8-bit, 4 MHz CPU) [8.2]. They usually communicate with a powerful base station which connects sensor nodes with external networks. The limited energy in senor nodes creates hindrances in implementing complex security schemes. There are two major factors of energy consumption

1. Transmission and reception of data
2. Processing of query request.

In order to implement security mechanism in sensor networks, we need to ensure that communication overhead is less and consumes less computation power. With these constraints it is impractical to use traditional security algorithms and mechanism that are built for powerful workstations.

Sensor networks are vulnerable to a variety of security threats such as DoS, Eavesdropping, Message injection, Message replay, message modification, malicious code, side channel analysis, etc. In order to secure sensor networks against these attacks, we need to implement message confidentiality, authentication, message integrity, non-repudiation, and intrusion detection, etc. Encrypting communication between sensor nodes can partly solve some of the problems but it requires a robust key exchange and distribution scheme.

In general, there are three types of key management schemes [8.2, 8.3]: Trusted Server scheme, self enforcing scheme and key-predistribution scheme. Trusted server schemes relies on a trusted base station, that is responsible for establishing the key agreement between two communicating nodes as described in [8.4]. It uses symmetric key cryptography for data encryption. The main advantages of this scheme are, it is memory efficient, nodes only need to store single secret key and it is resilient to node capture. But the drawback of this scheme is that it is energy expensive, it requires extra routing overhead in the sense that each node need to communicate with base station several times [8.3]. Self enforcing schemes use public key cryptography for communication between sensor nodes. This scheme is perfectly resilient against node capture and it is fully scalable and memory efficient. But the problem with the traditional public keys cryptography schemes such as DSA [8.5] or RSA [8.6] requires complex and intensive computations which is not possible to perform by sensor node having limited computation power. Some researchers [8.7,8.8] uses Elliptic curve cryptography as an alternative to traditional public key systems but still not perfect for sensor networks. Third scheme is key pre-distribution scheme that is based on symmetric key cryptography, in which limited numbers of keys are stored on each sensor node prior to their deployment. This scheme is easy to implement and does not introduce any additional routing overhead for key exchange. The degree of resiliency of node capture is dependent on the pre-distribution scheme [8.3].

Quite recently some security solutions have been proposed in [8.9-8.13] especially for wireless sensor networks but each suffers from various limitations such as higher memory and power consumptions that are discussed in detail in section 4.

Keeping all these factors in mind we have proposed a security framework- LSF - for wireless sensor networks. LSF combines the features of trusted server scheme and Self Enforcing security schemes. Our main contribution is the designing and implementation of LSF that provides

- Authentication and Authorization of sensor node.
- Simple Secure key exchange scheme.
- Confidentiality of data.
- Secure defense mechanism against anomalies and intrusions.
- Usage of both symmetric and asymmetric schemes.

The rest of this section is organized as follows. Section 2 describes the details of LSF. Section 3 presents the simulation results and evaluation of LSF. Section 4 presents the comparison of LSF with other security solutions. Section 5 contains conclusion and future directions.

8.2 Light weight Security Framework (LSF)

The basic objective of LSF is to provide light weight security solution for wireless sensor networks in which all nodes can communicate with each other. LSF can support both static and mobile environment, which may contain single and multiple Base Stations (BS). Basic system architecture is shown in Fig. 8.1. LSF uses both symmetric and asymmetric schemes for providing secure communication in wireless sensor networks.

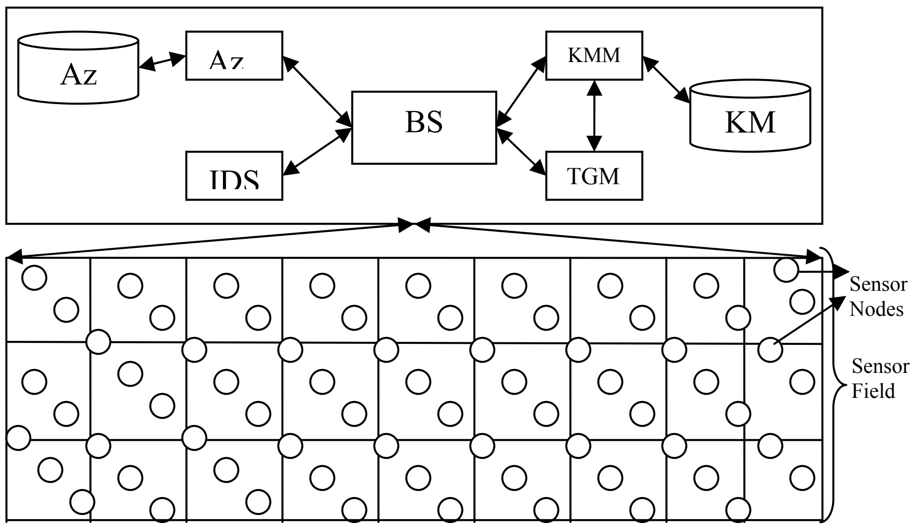


Fig 8.1 LSF System Architecture

Key Management Module (KMM) is used to store public and shared secret key of each node with BS to the database. Token Generator Module (TGM) is used to generate the tokens for the requesters, which will be further used by the other communicating party for the authentication of requester node. Authorization Module (AzM) is used to check whether particular node is allowed to communicate with other node or group. Light weight mobile agents will only be installed on Cluster heads which sends alerts messages to intrusion detection system (IDS), which is responsible for detecting any anomaly or intrusion in the network. Basic assumptions and rules of LSF are given below.

8.2.1 Assumptions

1. Base Station (BS) is the trusted party and it will never be compromised. Compromising the Base station can cause the entire sensor network to be useless and it is the only point from where sensor node can communicate with external networks.
2. Only Base Station (BS) knows the Public keys (Pk) of all the sensor nodes in the network. Communicating nodes will know each others public key during the time of connection establishment.

8.2.2 Rules

- Asymmetric scheme will only be used for sharing ephemeral secret key between communicating nodes.
- For every session new random secret key will be used.
- Data will be encrypted by using symmetric schemes because these schemes are considered to be execute 3-4 times faster than asymmetric schemes [8.14].

8.2.3 LSF Packet Format

LSF packet format is shown in Fig 8.2. Currently LSF uses seven types of packets, 'Request', 'Response', 'Init', 'Ack', 'Data', 'Update Group Key' and 'Alert' packet. All six packets are distinguished by 'type' field in the LSF packet. IDsrc field contain the id of sending node and last encrypted portion contain the information depending upon the type of packet, as shown in table 1.

The distribution of bits to different fields (as shown in Fig 8.2), introduces some upper limits, such as, size of source address is of 2 bytes, it means our LSF is only works in the environment where sensor nodes are not more than 218. Length of Nonce (unique random number) field is of 3 bytes, so LSF can maximum allow 224 connections at a time. The length of public key and private key is of exactly 128 bits and the length of secret key is of exactly 64 bits. Only stream cipher encryption algorithms are allowed to use because of a fixed length size of packets. MAC is of 64 bits.

Table 8.1 LSF: Type field

Type	ID_{src}	Encrypted Portion
Request	Any (sensor node)	$EK_{A\text{-}BS}$ (Intended-ID_{dest} , N)
Response	BS	$EK_{A\text{-}BS}$ (R-type, Intended-ID_{dest} , N , Pk, token | R)
Init	Any (sensor node)	EK_B^+(N, Pk, token)
Ack	Any (sensor node)	EK_A^+(N,sk)
Data	Any (sensor node)	EKsk (data)
UpdateGroupKey	Any CH sensor node	EK_G (GroupID, new Key), MAC
Alert	Any CH sensor node	$EK_{CH\text{-}BS}$ (Alert-type), MAC

$EK_{A\text{-}BS}$ = Encrypt with the secret key shared between node A and BS
EK_A^+ = Encrypt with the public key of node A
EK_B^+ = Encrypt with the public key of node B
EK_{sk} = Encrypt with the shared secret key
EK_G = Encrypt with group key
$EK_{CH\text{-}BS}$ = Encrypt with the secret key shared between Cluster head and BS

R-type = Response type (positive or negative response)
R = Reason of negative acknowledgement
Intended-ID$_{dest}$ = ID of Intended Destination
Pk = public key ID$_{src}$ = ID of source node
N = Nonce (Unique Random Number) MAC = Message Authentication Code
CH = Cluster Head

Type	Src	Encrypted Packet

Fig 8.2(a): LTSS Packet Format

4	2 bytes	2 bytes	3 bytes

Fig 8.2(b): Request Packet

4	2 bytes	1 bit	2 bytes	3 bytes	16 bytes	4 bytes

Fig 8.2(c): Response Packet

4	2 bytes	3 bytes	16 bytes	4 bytes

Fig 8.2(d): Init Packet

4	2 bytes	3 bytes	8 bytes

Fig 8.2(e): Ack Packet

4	2 bytes	30 bytes

Fig 8.2(f): Data Packet

4	2 bytes	8 bytes	8 bytes	8 bytes

Fig 8.2(g): Update Group Key Packet

4	2 bytes	8 bytes	8 bytes

Fig 8.2(h): Alert Packet

Fig 8.2 LSF Packet Format

8.2.4 Procedure

LSF works in three phases, authentication and authorization phase, key distribution phase and Data transmission phase. Authentication and authorization is performed during the exchange of "Request" and "Response" packet by using symmetric scheme. Key distribution phase involves sharing of random secret key in a secure manner by using asymmetric scheme. In this phase "INIT" and "ACK" packets will be exchanged. Data transmission phase involves transmission of data packet in an encrypted manner.

Let's suppose node A wants to communicate with the node B. It will first send request packet to Base station, in order to get token and public key of node B. The request packet is encrypted with the secret key shared between node A and BS. BS first checks in the database via AzM that either node A has rights to establish connection with node B. If yes it generates the token which will be further used by the node B for the authentication of node A. That token is encrypted with secret key shared between node B and BS, so that node A will not able to decrypt token. BS will sent back a response packet that contains token, public key of node B and Nonce (Unique Random Number) that was present in request packet. Nonce will ensure node A that packet came from genuine BS. When node

A gets the positive response from BS it sent the INIT packet to node B that contains Nonce, its own public key and token generated by BS. The whole INIT packet is encrypted with the public key of node B. When node B gets INIT packet it first check token, if it is correct it will generate the secret key and sent it back to node A in an encrypted manner. When node A gets ACK packet, it deletes the public key of node B from its memory and sent data to node B by using new session secret key. When data transmission complete both nodes delete that session key. For group communication, each node uses the group secret key for data transmission in a secure manner. Cluster head will update this key after periodic interval.

8.3 Simulation and Performance Analysis

We have tested our LSF protocol on Sensor Network Simulator and Emulator (SENSE) [8.15]. In sensor node we introduce the middleware between application layer and network layer as shown in Fig 8.3.

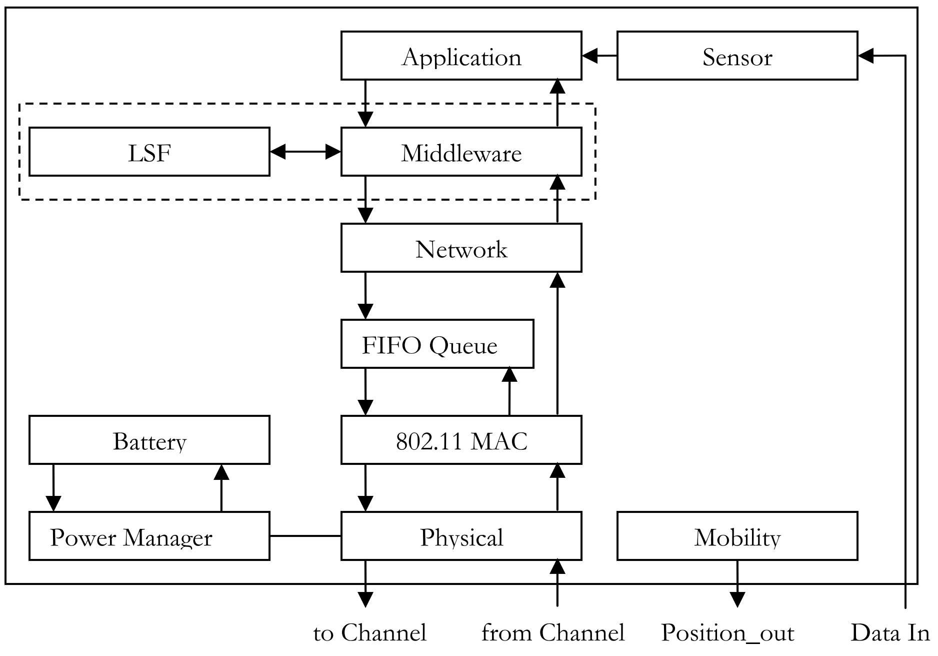


Fig 8.3 Sensor Node Architecture

That middleware uses LSF for the enforcement of security in the sensor network. At application layer we use constant bit rate component (CBR) that generate constant traffic during simulation between two communicating sensor nodes. For the demonstration and performance evaluation of LSF, we run CBR with and with out LSF. We randomly deployed 100 sensor nodes plus one Base station (BS) in 1000 by 1000 terrain. Basic simulation parameters which we use are given in table 2.

Table 8.2 Simulation Parameters

Terrain	1000x1000
Total Number of Nodes	101 (including BS)
Initial battery of each sensor node	1x106J
Power consumption for transmission	1.6W
Power consumption for reception	1.2 W

Idle power consumption	1.15W
Carrier sense threshold	3.652e-10W
Receive power threshold	1.559e-11W
Frequency	9.14e8
Transmitting & Receiving antenna gain	1.0

8.3.1 Performance Analysis of Communication Overhead

In our simulation scenario, application sent data packets of size 30 bytes in a periodic interval. The overall communication overhead of LSF for one to one communication is decreases with the increase in transfer of number of data packets as shown in Fig 8.4. Communication Overhead (C0 %) is calculated as

$$CO(\%) = \left(\frac{Nc * 74.125}{\sum_{i=1}^{n} N_i^P * 30}\right) * 100$$

Where as 'Nc' is the total number of connections. $\sum_{i=1}^{n} N_i^P$ is the total number of packets transferred by all connections.

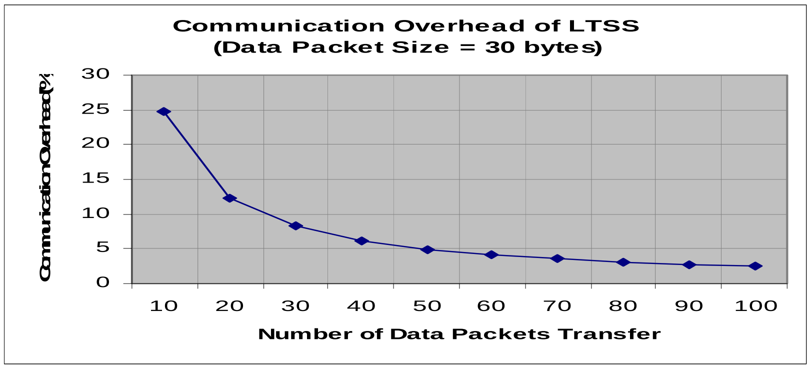


Fig 8.4 Communication Overhead (%) of LSF

8.3.2 Performance Analysis of Power Computation

Power Computation primarily depends upon the symmetric and asymmetric scheme applied. If we assume that computation power required for symmetric encryption and decryption scheme is CSE and CSD respectively and computation power of asymmetric encryption and decryption scheme are CAE and CAD respectively. Then the total power consumption required by single node during first two phases is

Power Computation = (CSE + CSD) + (CAE + CAD)

Computation power required by a single node during data transmission phase is calculate as,

Power Computation= (TNSP*CSE) + (TNRP*CSD)

Where TNSP is the Total Number of Sent data packets and TNRP is the Total Number of received data packets.

8.3.3 Performance Analysis of Memory Consumption

Every sensor node needs to store only six keys, three of them are permanent and three are ephemerals. Permanent keys are public and private keys of its own and public key of BS. Ephemerals keys are Group key, public key of other node and session secret key. In order to save these keys only 72 bytes are needed. Details are given in table 3. This approach will make sensor network memory efficient.

Table 8.3 Storage Requirement of Keys

S/No	Keys	Size (in bytes)
	Permanent Keys	
1	Public key of node	16
2	Private key of node	16
3	shared secret key b/w Node & BS	8
	Ephemeral Keys	
4	Group Key	8
5	Public key of other node	16
6	Session key	8
Total Storage size Required		72 bytes

8.3.4 Performance Analysis of Energy Consumption

The main cause of energy consumption in sensor node is dependent on transmission and reception cost. We have used the SENSE that consumed energy in four different modes: TRANSMIT, RECIEVE, IDLE, and SLEEP. Energy consumption rate of each mode is given in Table 8.2. For each connection, LSF per connection exchange four control packets of cumulative size 74.125 bytes that requires for authentication, authorization and key exchange mechanism. That is an acceptable tradeoff between energy and security. Simulation result of energy consumption is shown in Fig 8.5.

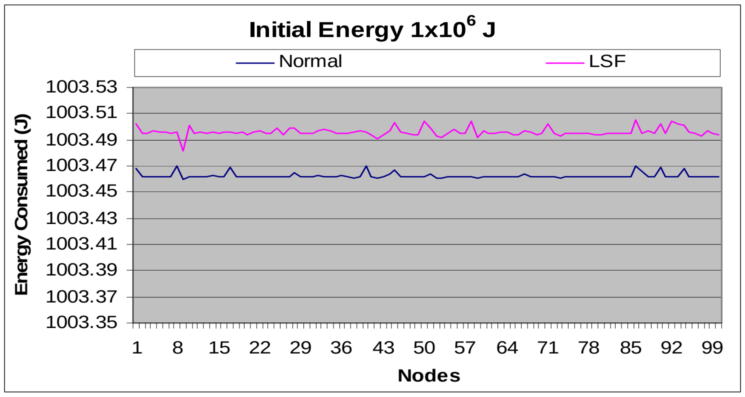


Fig 8.5 Energy Consumptions

8.3.5 Resilience against Node Compromise

Single node compromised will not expose the whole communication in the network. Only those communication links will expose that are established with compromised node. Let's suppose 'Ncn' is the set of nodes that establish connections and 'Ncp' is the set of compromised nodes. Then Ncn ∩ Ncp will give us the set of nodes that are compromised as well as connected. Then the maximum number of connections that can be exposed only if all compromised nodes connected to uncompromised nodes. On the other hand minimum numbers of links that can be exposed only if all compromised nodes are connected with each other.

$$Max : Ncn \bigcap Ncp$$

$$Min : \left(\begin{array}{ll} \dfrac{Ncn \bigcap Ncp}{2} & for \longrightarrow even \\ (\dfrac{Ncn \bigcap Ncp + 1}{2}) & for \longrightarrow odd \end{array} \right)$$

If we assume that sensor networks consists of 1000 nodes and total 500 connections established between pair of nodes then the total links that can be minimum and maximum compromised is shown in Fig 8.8.

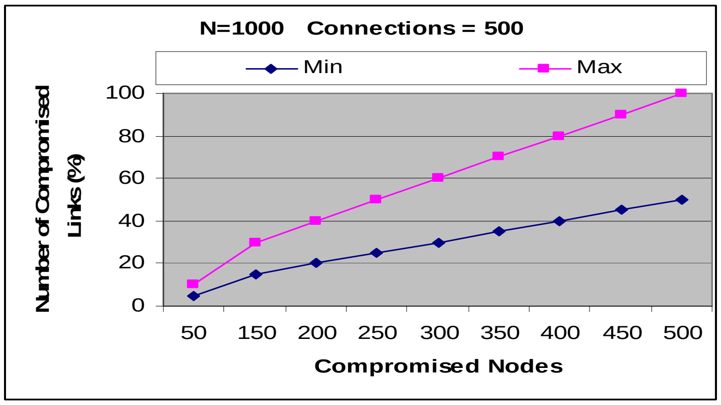


Figure 8.6 Percentages of Compromised Links

8.4 Comparison of LSF with Other Security Solutions

Table 8.4 Comparison of LSF with other security solutions

		SPINS	TinySec	LiSP	LSF
Memory Requirement with respect to storage of keys		3	Depended on KMS[2]	≥ 8	6
Transmission Cost	During key exchange (bytes)	--	Depended on KMS	12.6*TNN[3]	74.125*TNC[4]
	During Data Transmission	20%	10%	> 20%	8.33%
Public Key Cryptography Support		No	No	No	Yes
Symmetric key cryptography Support		Yes	Yes	Yes	Yes
Intrusion Detection mechanism		No	No	Yes	Yes
Authentication support		Yes	Yes	Yes	Yes
Authorization support		No	No	Yes	Yes

[2] KMS: Key Management scheme
[3] TNN: Total Number of Nodes

[4] TNC: Total Number of Connections

Data Integrity support	Yes	Yes	Yes	No
Confidentiality support	Yes	Yes	Yes	Yes
Availability support	No	No	Yes	No

Quite recently some security solutions have been proposed in [8.9, 8.10, 8.12] specially for wireless sensor network but each suffers from various limitations. Adrain Perrig et. al [8.9] have proposed security protocols suite called SPINS for wireless sensor networks. SPINS consist of two building blocks SNEP and uTESLA. SNEP provides data confidentiality, two party data authentication and data freshness where as uTESLA provides authenticated broadcast for severally resource constraint environment. For data confidentiality they use symmetric encryption mechanism in which secret key called master key is shared between sensor node and base station. SNEP uses one time encryption key that produces from the unique master key. SNEP uses MAC function for two party authentications and checking data integrity. SPINS is based on binary security model means either it provides maximum security or no security. There are number of drawbacks associated with SPINS such as, it can only work in non-anonymous environment in which all nodes have some unique id. Because of the usage of source routing scheme in SPINS they are making the network vulnerable to traffic analysis [8.16]. It does not address security in the Physical layer therefore they are unable to provide defense mechanism against physical layer attacks such as jamming etc [8.11].

Chris Karlof et. al [8.10] have proposed TinySec architecture for wireless sensor networks. TinySec is a link layer security protocol that provides authentication, integrity and confidentiality by adding less than 10% of energy, latency and bandwidth overhead. TinySec does not provide access control and non-repudiation. It also does not provide protection against physical layer attacks. The major drawback of this solution is that it is tightly coupled with Berkeley TinyOS and can not be use for general sensor network model [8.17]. Like SPINs it can only work in non-anonymous environment in which all nodes have some unique id.

Taejoon Park and Kang G. Shin [8.12] have proposed Light weight Security protocol (LiSP) that's makes a tradeoff between security and energy consumption through efficient re-keying mechanism. LiSP achieves authentication, confidentiality, data integrity, access control and availability. Another important feature of LiSP architecture is the ability to detection intrusions. By using LiSP each node need to save eight keys.

General Comparison of all above discussed schemes with LSF is given in Table 8.4. We have compared from the perspective of memory requirement, transmission cost, and some other basic security parameters such as authentication, authorization, confidentiality etc.

8.5 Conclusion and Future Directions

We have proposed Light weight security framework (LSF) for wireless sensor networks, which provides authentication and authorization of sensor node. It also provides simple secure key exchange scheme, as well as confidentiality of data. LSF is highly scalable and memory efficient. It uses 6 keys, which takes only 72 bytes memory storage. It introduces 74.125 bytes of transmission and reception cost per connection. It also provides simple secure defense mechanism against compromised nodes. In future we try to solve the issue related to the neighboring nodes of the BS, that were suffered from higher communication overhead by forwarding request and response packets during authentication and authorization phase.

8.6 References

[8.1] C. Karlof and D. Wagner, “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures”, Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (WSNA’03), May 2003, pp. 113- 127

[8.2] Wenliang Du, Jing Deng, Han, Y.S., Shigang Chen, Varshney P.K, “A key management scheme for wireless sensor networks using deployment knowledge”, proceeding of INFOCOM 2004, Mar 2004

[8.3] Lydia Ray, “Active Security Mechanisms for Wireless Sensor Networks and Energy optimization for passive security Routing”, PhD Dissertation, Dep of Computer Science, Louisiana State University, Aug 2005

[8.4] J. Kohl and B. Clifford Neuman, “The Kerberos Network Authentication Service (v5)”, RFC 1510, Sep 1993

[8.5] W. Diffie and M.E. Hellman, “New Directions in Cryptography”, IEEE Transaction on Information Theory, vol. 22, Nov 1976, pp. 644-654.

[8.6] R. L. Rivest, A. Shamir, L.M. Adleman, “A method for obtaining Digital Signatures and Public key cryptosystem”, Communication of ACM, vol. 21(2), 1978, pp. 120-126

[8.7] Erik-Oliver Blaß and Martina Zitterbart, “Towards Acceptable Public-Key Encryption in Sensor Networks”, 2nd International Workshop on Ubiquitous Computing, ACM SIGMIS, May 2005

[8.8] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary, “Wireless sensor network security: A Survey”, Technical Report MIST-TR-2005-007, July, 2005

[8.9] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. D. Tygar, “SPINS: Security protocols for sensor networks”, proceedings of 7th annual international conference on Mobile computing and networking, Rome, Italy, Aug 2001, pp 188-189

[8.10] Chris Karlof, Naveen Sastry, and David Wagner, “TinySec: a link layer security architecture for wireless sensor networks”, Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, Nov 2004, pp 162-175

[8.11] K. Jones, A.Wadaa, S. Oladu, L. W|son, and M. Etoweissy, “Towards a new paradigm for securing wireless sensor networks”, Proceedings of the 2003 workshop on New security paradigms, Ascona, Switzerland, Aug 2003, pp 115 - 121

[8.12] Taejoon Park, and Kang G. Shin, “LiSP: A Lightweight Security Protocol for Wireless Sensor Networks’ ACM Transactions on Embedded Computing Systems, Vol. 3, No. 3, Aug 2004, pp. 634–660

[8.13] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia, “LEAP: Efficient Security Mechanism for Large-Scale Distributed Sensor Networks”, Proceedings of the 10th ACM conference on Computer and communications security, Washington, USA, 2003, pp. 62-72

[8.14] Elaine Shi and Adrian Perrig, “Designing Secure Sensor Networks”, IEEE Wireless Communications, Dec 2004, pp. 38-43

[8.15] Sensor Network Simulator and Emulator (SENSE) http://www.cs.rpi.edu/~cheng3/sense/

[8.16] Jeffery Undercoffer, Sasikanth Avancha, Anupam Joshi, and John Pinkston, “Security for Sensor Networks”, Proceeding of 2002 CADIP Research Symposium, Baltimore, MD, Oct 2002.

[8.17] Adrain Perrig, John Stankovic, and David Wagner, “Security in wireless sensor networks”, communications of ACM, vol 47(6), Jun 2004, pp. 53-57

C h a p t e r 9

INTRUSION DETECTION SYSTEM

9 Intrusion Detection System

9.1 Introduction

Intrusion detection (ID) is defined as "the problem of identifying individuals who are using a computer system without authorization and those who have legitimate access to the system but are abusing their privileges". The intrusion detection field has grown considerably in the last few years, and a large number of intrusion detection systems have been developed to address different needs. Intrusion detection is clearly more necessary in ubiquitous networks where other secure systems such as firewall are not applicable. In this Smart Spaces scenario, we focus on designing an IDS architecture for sensor network side.

Wireless sensor networks (WSNs) constitute a new paradigm of ambient monitoring with many potential applications. Typically formed by thousand of nodes of small dimension, they use ad-hoc communication and have scarce resources regarding energy, bandwidth, processing capacity and storage.

WSNs are typically designed to gather data in inhospitable places and might be involved in critical applications. Wealth environment mapping and enemy's movement monitoring in a battlefield are some examples of critical applications they are used for. In these applications, WSNs are of interest to adversaries. WSNs are susceptible to some types of attacks since they are deployed in open and unprotected environments and are constituted of cheap small devices. Preventive mechanisms can be applied to protect WSNs against some types of attacks. However, there are some attacks for which there is no known prevention method, such as wormhole. Moreover, there are no guarantees that the preventive methods will be able to hold the intruders. For these cases, it is necessary to use some mechanism of intrusion detection. Besides preventing the intruder from causing damages to the network, the intrusion detection system (IDS) can acquire information related to the attack techniques, helping in the development of prevention systems.

Intrusion detection poses many challenges to WSNs, mainly due to the lack of resources. Besides, methods developed to be used in traditional networks cannot be applied directly to WSNs, since they demand resources not available in sensor networks. WSNs are typically application-oriented, which means they are designed to have very specific characteristics according to the target application. The intrusion detection assumes that the normal system behavior is different from the behavior of a system under attack. The several possible WSN configurations make difficult the definition of the "usual" or "expected" system behavior. Since common nodes are designed to be cheap and small, they do not have enough hardware resources. Thus, the available memory may not be sufficient to create a detection log file. Moreover, a sensor node is designed to be disposed after being used by the application and it makes difficult to recover a log file due to the possible dangerous environment in which the network was deployed. The software stored in the node must be designed to save as much energy as possible in order to extend the network lifetime. Finally, another challenge to the design of an IDS is the frequent failures of sensor nodes when compared to processing entities found in wired networks. Given all these characteristics, it is important to detect the intrusions in real time. In this way, we could hold the intruder and minimize the application damages.

9.2 Related work

Lots of work has been done in the field of intrusion detection for Wireless Ad-hoc networks. In [9.1], they pointed out the challenges of intrusion detection and proposed the first distributed and cooperative architecture to suite the needs of mobile ad-hoc networks. In that architecture, every node is installed an IDS agent which is responsible for detecting signs of intrusion locally and independently. However, IDS agents in neighboring nodes can collaboratively participate in global intrusion detection actions. In [9.2], architectures using clustering were first proposed to reduce the number of monitoring nodes. The protocol for securing clusters was presented in [9.3] where every node has equal chance to be elected as a cluster head. Here, abnormal detection & identifying attack type based on statistics of packets. Clustering technique was also used in [9.4] to organize and maintain a dynamic hierarchy of intrusion detection components. The advantages of a hierarchy is "its potential scalability to large networks, since it can provide rapid and communication-efficient detection for local cooperative attack recognition, while still allowing data sharing for more widely-distributed cooperative intrusion detection algorithms." The hierarchy is dynamic because it always is reconstructed to adapt to the flexibility of mobile ad-hoc networks. Mobile agents were mentioned in [9.5, 9.6] to solve problems of memory limit and intermitted connection in ad-hoc networks.

However, all above techniques can not be applied directly to intrusion detection in Wireless Sensor Networks because of their some crucial characteristics. The Decentralized Intrusion Detection System [9.7] has some nodes installed intrusion detection agent (called monitoring nodes). These nodes were selected to cover all the networks. Each intrusion detection agent uses some simple rules to detect anomaly behavior. However, this architecture lacks of cooperation between monitoring nodes. An algorithm to detect anomalies based on packets' receive power was discussed in [9.8] where a packet is considered anomaly if its receive power is below the min or above the max of a certain value. "The New Intrusion Prevention & Detection Approaches" presented in [9.9] can reduce monitoring time & reduce energy but it is only used for Clustering-based Sensor Network.

In summary, these intrusion detection architectures proposed above did not work well & had high rate of false alarm due to the limit in computation, memory and energy of sensor nodes.

9.3 Proposed architecture

Our proposed architecture takes full advantage of the secure & powerful server in Smart Space scenario. It includes two parts: ID server and ID agents. Most of the intrusion detection processing is done in the server side to reduce processing amount in the client side. ID agents are installed in every sensor nodes to collect data and detect anomaly behaviors.

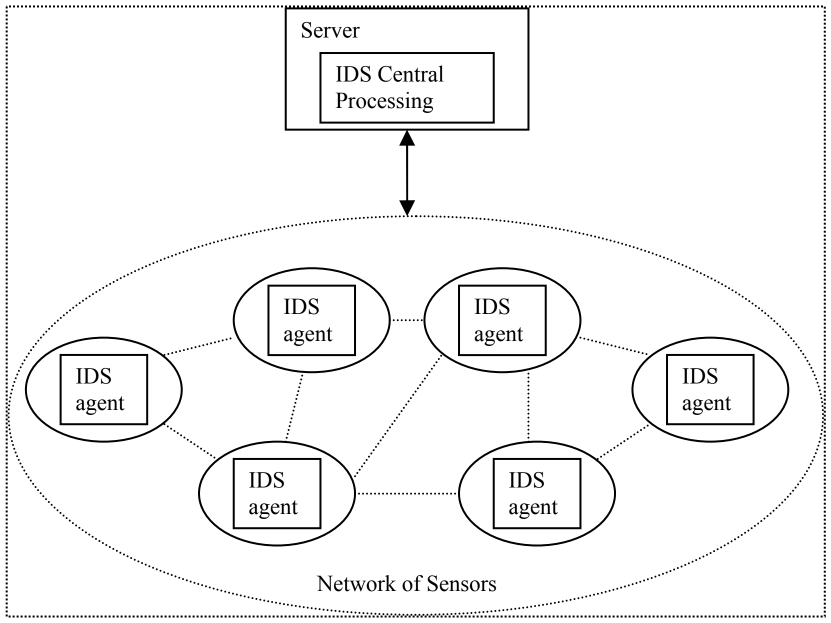


Fig 9.1 Intrusion Detection Architecture

9.3.1 ID Agents

Because sensor networks lack of central point to collect data so there is no doubt that every intrusion detection architecture must install detection agents on many nodes in order to cover the entire network. The important thing is how agents should do to save energy & other resources of the host nodes, however, still guarantee the security for the host nodes & all network.

ID agent is responsible for monitoring the host node & its neighbors for anomaly behaviors. Because of limited resources of sensor nodes, ID agent must be as lightweight as possible. In our architecture, agents use some simple rules to detect intrusion.

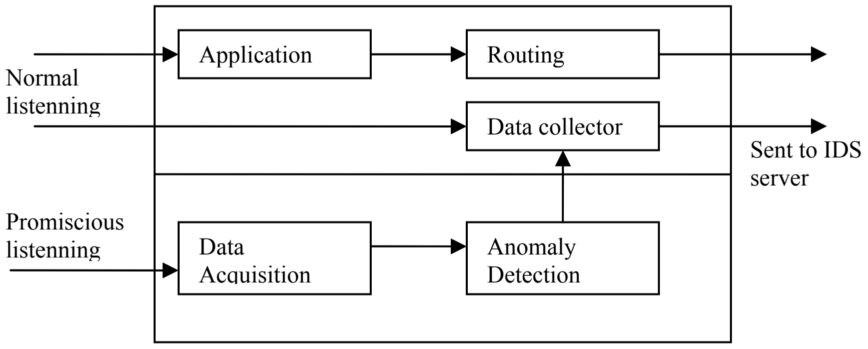


Fig 9.2 IDS agent architecture

Data Acquisition

In this phase, messages are listened to in promiscuous mode by the monitor mode and the important information is filtered and stored for subsequent analysis. Important information includes message fields that might be useful to the rule application phase. Thus, we use less memory and less processing time, saving energy. Messages to which no rules can be applied are not stored.

Data extracted from the messages are stored in an array data structure and discarded after a given period of time or when there is no space left in memory.

Anomaly Detection:
This module detects anomaly behavior in neighboring nodes based on statistical data of packets received by promiscuous listening. This module supplements Rule Application module.

Data Collector Module:
This module is in charge of collecting & sending necessary data to the IDS server for father processing. However, in the sake of saving resource, this module is active only when the IDS agent detects something abnormal in the network. IDS agents rarely go to the final decision of intrusion detection. This is in charge of IDS server.

9.3.2 IDS Server

IDS server is the central processing in this architecture. IDS server will collect all data from necessary sensor nodes, aggregating & making the final intrusion decision. IDS server can use both signature based and anomaly detection technique.

9.4 Proposed anomaly detection algorithm

9.4.1 Background

9.4.1.1 Anomaly Attack Detection

All of the intrusion detection techniques are classified into one of two methodologies: misused detection or anomaly detection. Misused detection techniques, sometimes referred to as signature-based detection techniques, look for behavior that matches a known attack scenario by analyzing the information in the network, comparing it to a large database of known attacks (signatures). Any new attack which is not in the database can not be detected so the database must be kept up to date, which is not easy to do in sensor networks. Anomaly detection techniques look for behavior that deviates from normal system activities. These techniques do not require knowledge of know attacks and can detect new types of intrusion which is considered more suitable for sensor networks.

The key question in anomaly detection techniques is how to distinguish anomalies from normal. Which factors of behavior used to know whether one behavior is normal or not is the most important thing in an anomaly detection system. Some systems use the distribution of commands that users used in their session, some use statistics about system calls … Generally, it depends on characteristics and common types of attacks against each systems.

9.4.1.2 Attacks on sensor networks

In order to construct an anomaly detection algorithm in sensor networks, it is necessary to analyze some of the most common attacks in sensor networks including: wormhole, black hole, HELLO flood attack, Jamming … Most of them focus on vulnerabilities of routing protocols.

- Wormholes: By some ways, an adversary creates communication links between some pairs of compromised sensor nodes. This may attract more sensor nodes to send their traffic via these links. After that, the adversary can eavesdrop, alters or simply drops these packets.
- Black holes: a black hole is formed when a node tries to advertise a zero-cost route to all other nodes in the network. As a result, more sensors will send traffic through this zero-cost route and will be unsuccessful.
- HELLO flood attack: an adversary broadcasts HELLO packets with large enough transmission power to lure other sensor nodes that the adversary is their neighbor.
- Exhaustion: a sacrificed node keeps transmitting packets to another node to exhaust the target's battery power.

To realize the anomaly characteristics of these attacks, we divide them into two major categories: (1) attracting other nodes to send their traffic to a compromised node and (2) exhausting a node's resources by sending many packets to the target. It is straightforward to see that attacks in each category makes the network traffic deviated from that in normal condition in different ways. If the network is under attacks in the first category, traffic to some nodes (compromised nodes) will be suddenly increased and attacks in the third category is realized by the increasing amount of outgoing traffic related to one node. Therefore, we can detect attacks in sensor networks by monitoring these anomalies. They are the changing in (1) the number of incoming packets to a node and (2) the number of outgoing packets from a node.

9.4.2 Proposed Algorithm

A lot of techniques have been done for anomaly detection such as: neural network, audit data analysis and mining, statistical models … Each of them has their own pros and cons. Here, we used a widely-used anomaly detection algorithm, Cumulative Sum (CUSUM). CUSUM is suitable to deploy in sensor network because it is a strong, light-weight and less memory consuming statistical model.

9.4.2.1 CUSUM algorithm

CUSUM is one of some change point detection algorithms used widely to detect the change of mean value of a random sequence (see [9.11, 9.12] for good survey). In brief, CUSUM detect changes based on the cumulative effect of the changes made in the random sequence instead of using a single threshold to check every variable. To detect abrupt changes in a random sequence {Xn}, CUSUM requires a parametric model for {Xn} which it not easy in some cases. Thus, a new approach called non-parametric CUSUM proposed by Wang [9.13] is used more popular especially in attack detection. Assume that {Xn} has a negative mean in normal condition and become large positive in anomaly operation, we set:

$$y_0 = 0$$

$$y_n = (y_{n-1} + X_n)^+ \quad (n \geq 1)$$

where

$$(x)^+ = x : x > 0$$
$$= 0 : \text{otherwise}$$

y_n can be canculated in another way:

$$y_n = S_n - \min S_k,$$

$$S_k = \sum_{i=1}^{k} x_i$$

In normal operation, the mean of {Xn} is negative so $y_n \sim 0$. In anommaly condition, Xn will become positive. $\{y_n\}$ will accumulate with time. A large $\{y_n\}$ is a strong indication of abrupt changes. In attack detection, we set $d_n(y_n)$ be the decision function. $d_n()$ can be defined as:

$$d_n(y_n) = 0 \text{ if } y_n \leq N$$
$$= 1 \text{ if } y_n > N$$
(N is the threshold of the attack detection)
We can describe the CuSum algorithm in brief as following:

```
CuSum := 0
n := 0
Repeat
        n := n + 1
        CuSum := CuSum + Xn
        If CuSum > ThresHold then
                Signal attack indication
Until Finished
where n is the n^th sampling period
```

The algorithm is straightforward. The most important thing is how to model {Xn}. In next parts, we will discuss the way to model {Xn} to detect abrupt changes in the number of incoming packets, the number of collision packets and the number of outgoing packets from a node.

9.4.2.2 Detecting changes in the number of incoming packets

Let { Δn , n = 0, 1, …} be the number of incoming packets to the monitored node collected within one sampling period. However, { Δn } depends on the size of sampling period and the density of the monitored node's vicinity. To normalize, we simply define $Z_n = \Delta n / \overline{F}$ where $\overline{F}$ is the average number of incoming packets to the monitoring node in a sampling period. $\overline{F}$ can be calculate recursively as following:

$$\overline{F}(n) = \alpha \overline{F}(n-1) + (1 - \alpha) \text{ INC}(n)$$

where INC(n) is the number of incoming packets to the monitoring node in the nth sampling period. α is a constant lying between 0 and 1 indicating the memory in the estimation.
Thus, the mean of $\{Z_n\}$ is close to 1 in normal condition. To satisfy the assumption (2), we transform $\{Z_n\}$ to another random sequence without loss of any statistical feature.

$$X_n = Z_n - \beta$$

where β is a constant parameter depending on the network condition to produce $\{X_n\}$ with a negative mean. In general, β is selected to be larger than the mean of $\{Z_n\}$ during normal conditions.
So, we can apply nonparametric CUSUM with a random sequence $\{X_n\}$ to detect changes in the number of incoming packets to the monitored node.

9.4.2.3 Detecting changes in the number of outgoing packets

Similarly, let { Δn , n = 0, 1, …} be the number of outgoing packets to the monitored node collected within one sampling period. We define $Z_n = \overline{F} / \Delta n$ where $\overline{F}$ is the avarage number of outgoing packets to the monitoring node in a sampling period.

$$\overline{F}(n) = \alpha \overline{F}(n-1) + (1 - \alpha) \text{ OUT}(n)$$

where OUT(n) is the number of incoming packets to the monitoring node in the nth sampling period. α is a constant lying between 0 and 1.
Thus, the mean of $\{Z_n\}$ is close to 1 in normal condition & become larger under attack. To satisfy the assumption (2), we set

$$X_n = Z_n - \beta$$

β is selected to be larger than the mean of $\{Z_n\}$ during normal conditions.

Algorithm 2 summarizes our algorithm described above.

```
all CuSum = 0
n = 0
repeat
    n = n + 1
    for each neighbor i do
        CuSum(inc i) : = (CuSum(inc i) + Xn(inc i))^+
        CuSum(out i) : = (CuSum(out i) + Xn(out i))^+
        if any CuSum > Its Threshold then
            Signal attack indication
    end for
until Finished
```

where n is the nth sampling period. inc i means the number of incoming packets of ith neighbor. *out i* means the number of outgoing packets of ith neighbor. Collision means the number of collision packets of the monitor node.

9.5 Conclusion

Our architecture is decentralized because IDS agents are installed on every node of the network. In some certain cases, this is considered redundant & resource-consuming. However, in order to save resources, we can install agents on certain sensor nodes so that these nodes can still cover all the network. The key point in our architecture is the cooperative intrusion detection performed by the IDS server. Because the server is very powerful so this architecture will both reduce the resource needed in sensor end and increase the effectiveness of the IDS system.

9.6 References

[9.1] Zhang Y, Lee W & Huang Y: "Intrusion Detection Techniques for Mobile Wireless Networks", ACM Wireless Networks, 9, 2003

[9.2] Oleg Kachirski and Ratan Guha. "Effective Intrusion Detection Using Multiple Sensors in Wireless Ad Hoc Networks." IEEE HICSS in Hawaii, Jan. pp57 (2003)

[9.3] Yi-an Huang, Wenke Lee A Cooperative Intrusion Detection System for Ad Hoc Networks, ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03) October 31, 2003 George W. Johnson Center at George Mason University, Fairfax, VA, USA

[9.4] Daniel F. Sterne, Poornima Balasubramanyam, David Carman, Brett Wilson, Rajesh Talpade, Calvin Ko, Ravindra Balupari, Chin-Yang Tseng, Thomas F. Bowen, Karl N. Levitt, Jeff Rowe: A General Cooperative Intrusion Detection Architecture for MANETs. IWIA 2005: 57-70

[9.5] P. Kannadiga, M. Zulkernine, and S. Ahamed, Towards an Intrusion Detection System for Pervasive Computing Environments, Proc. of the International Conference on Information Technology (ITCC), pp. 277-282, IEEE CS Press, Las Vegas, Nevada, USA, April 2005

[9.6] Guy Helmer , Johnny S. K. Wong , Vasant G. Honavar , Les Miller , Yanxin Wang, Lightweight agents for intrusion detection, Journal of Systems and Software, v.67 n.2, p.109-122, 15 August 2003

[9.7]A. P. R. Da-Silva, M. H.T Martins, B. Rocha,A. Loureiro, L. Ruiz, and H. C. Wong University of Bel-Horizon, Brasil, Decentralized Intrusion Detection in Wireless Sensor Networks

[9.8] An Intrusion Detection System for Wireless Sensor Networks, Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks

[9.9] Mong-Fong Horng, Ko-Ming Chang, Chien-Chung Su and Yau-Hwang Kuo, "The New Intrusion Prevention and Detection Approaches for Clustering-based Sensor Networks," accepted by IEEE Wireless and Communications and Networking Conference, New Orleans, Mar. 2005

Chapter 10

HOME FIREWALL

10 Home Firewall

10.1 Introduction

The vision of ubiquitous computing, with devices seamlessly integrated into the life of everyday users, and services readily available to users anywhere all the time [10.1,10.2] is becoming now a reality. A ubiquitous computing environment consists of a various range of hardware (user devices, sensors etc), applications and services which can predict the demand of users and act on their behalf proactively. These gadgets enable the seamless integration of computing resources and physical spaces, and surround users with a convenient, information-rich atmosphere that we refer to as a smart space [10.3].

However, ubiquitous computing increases more concerns related to security and privacy. The emergence of smart spaces computing to physical environments causes the information and physical security issues become interdependent. In addition, it is the dynamism and mobility absolutely necessary for smart spaces that can yield extra chances for attackers to exploit vulnerabilities in the system invisibly. Therefore, designing a sufficient and suitable security perimeter mechanism for home and office smart spaces becomes an urgent demand.

Many of related research activities [10.3-10.5] have been focused on how to efficiently protect networks and data from attackers, and even more study is still in progress. Numerous methods related to secure communication, authentication, and authorization issues are being presented [10.6,10.7,10.8]. Also, quite much research work has been dedicated to firewall and its connected applications [10.9-10.12].

However, these efforts have focused on the area of fixed wire networks and just few of them on the area of wireless ones. And at the time of this writing, to our knowledge, there has been no published work on applying firewall technology to smart spaces.

The freedom of mobile and pervasive users to wirelessly and invisibly connect to their home systems from anywhere, at any time raises complicated security risks. The communication between the users and the environment, and the transaction inside the environment itself are more easily intercepted; malicious software is more easily installed into the systems; and the space is more susceptible to attacks. This enforces additional requirements on security factors. Researchers must find techniques to reduce the security risks as much as possible from every aspect of the smart medium. Firewall technology is a logical approach that can help them accomplish this troublesome task.

Generally, a firewall is a hardware or software barrier placed between the network of concern and the rest of the world to prevent unwanted and damaging intrusions of the network [10.13]. Unfortunately, due to the pervasiveness and context-awareness characteristics of the smart space, conventional firewall is not suitable for this environment. In traditional network systems, this kind of firewall only has to deal with normal TCP/IP packets and can place at the network gateway to trace the abnormal signatures. However, in ubiquitous computing environments, the firewall not only has to encounter with ordinary TCP/IP packets but also faces with context-aware packets and other command packets in/out the service servers. Alternatively, it is not easy to put the firewall at the gateway to control the whole space since various types of networks, such as wireless networks, wireless sensor networks, and fixed wire networks, are deployed in the smart medium. Therefore, how and where can the common firewall efficiently examine context-aware or control packets that are in improper condition or modified by the attackers to deny them?

In this paper, we introduce a new concept of context-aware host-based firewall called Home Firewall. A concrete scheme is also presented. The suggested solution combines with other security methods to protect the central service server deploying our current context-aware middleware, namely CAMUS [10.14]. In the CAMUS project, we present the design and approach to a middleware solution that expedites context-awareness in a ubiquitous computing environment. CAMUS envisions a comprehensive middleware solution that not only focuses on providing context composition at the software level by masking the inherent heterogeneity of environment sensors. The CAMUS core architecture consists of the core services that make up the smart spaces. We believe that security and context awareness are two necessary core services for any ubiquitous environment.

The remaining paper is organized as follows. In Section 2, we describe a smart home scenario as an example of our security model. In Section 3, we present the proposed method. Section 4 provides the discussions. Section 5 describes the related work. Finally, in Section 6, conclusion and future work are given.

10.2 A Smart Home Scenario

In order to elaborate the functional aspects of the model proposed in the next section, we consider the following scenario.

Firstly, we assume that a morning routine of one user consist of doing his personal hygiene, listening (or reading) to his related news, having breakfast, and leaving for office on his car. All these things can be automatically and timely done in a smart home by a central service server running context-aware applications or services. When he wakes up, windows drapes open, lights in rooms turn on/off according to his location, coffee pot heats up coffee. His PDA can well-timed inform him about his important events such as the stock condition, daily rate exchange information, etc. Before leaving home, the surveillance camera and alarm system automatically start controlling and detecting events of interest (e.g., thieves actions, enemy vehicles, house fires.)

Now, we consider the infrastructure model supporting the smart space (the smart home, smart office, etc, described in Fig 10.1) in which it has one machine playing the role as a central server and variants of clients. The house is equipped with a number of different useful sensors including location, temperature, light, humidity sensors in the rooms. For instance, the camera and alarm system placed on the ceiling of living room or to the suitably unobtrusive places is used to supervise the user's condition and state. The sensors and camera system are assumed to connect to the central server deploying a context-aware middleware, like CAMUS, through a base station to control the user behaviors.

Mobile and wireless devices (PDAs, Laptops, Mobile phones, etc) are supposed to wirelessly connect to a Wi-Fi network through a Wireless Access Point (Wireless Gateway) in order to query or manage information supplied by the server.

Applications or services performing on the server on the servers requires a user's electronic profile to control the behavior of these devices, such as issuing control commands to start up his PDA to query proper data, close the front-door, and turn on the surveillance camera and alarm system when he left, to make the space become a smart environment.

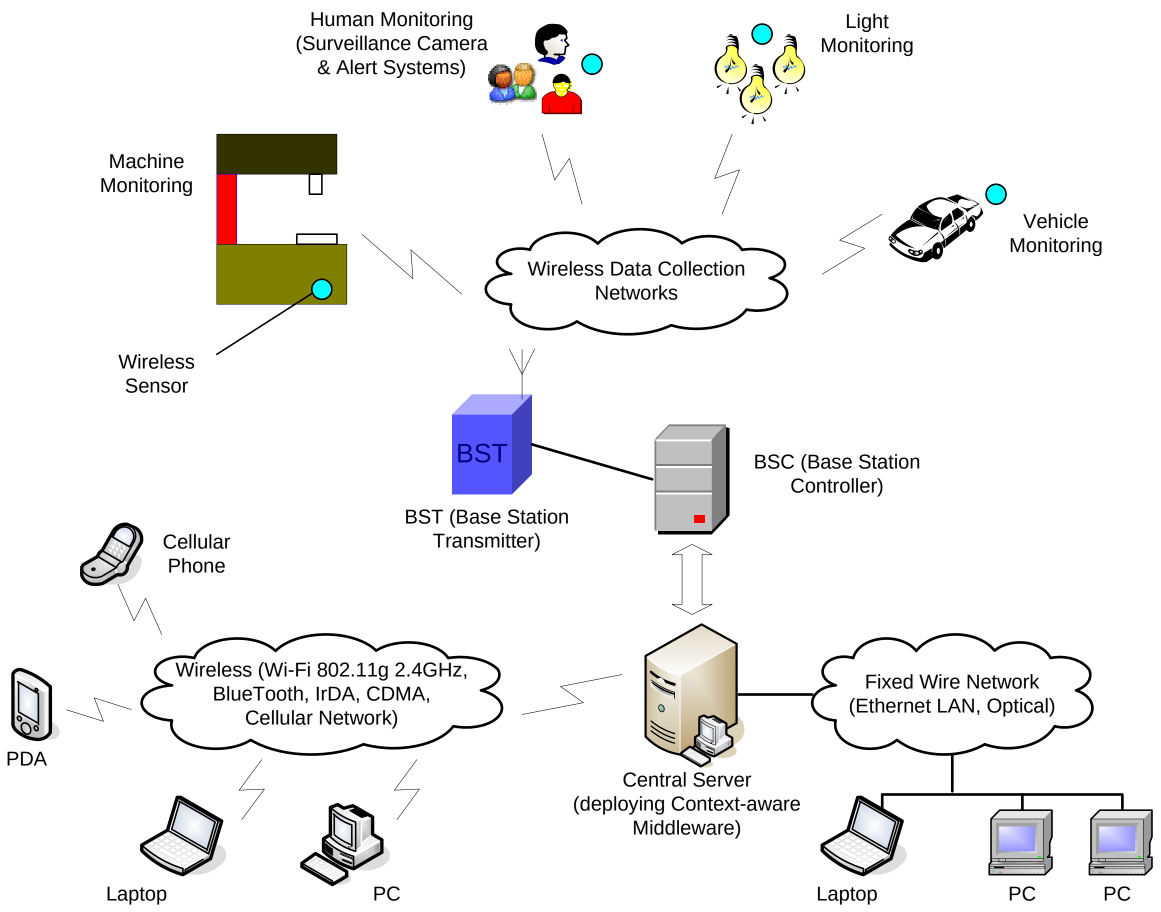


Fig 10.1 An example for a smart home infrastructure

10.3 Proposed Methodology

As mentioned above, ubiquitous computing environments raise a great number of security issues. Two main causes of these matters are exchange of information and mobility. Ubiquitous information is transmitted invisibly and wirelessly, so it can more easily intercepted than in other spaces. The problem gets worse when users and their devices permit to be mobile. We can take a visual example by supposing that the user in the scenario noted earlier joins his laptop into an unprotected network already infected with viruses, worms, or Trojan horses at his office, his laptop is then infected with the kind of plague. Later, he brings the laptop into his home network, a protected environment, and connecting to the central server through the Wireless Gateway. In this case, packets from his laptop are sent to the servers without any verification and they are thus free to corrupt the entire system.
In this section, we focus on the security problems, discuss some firewall basic knowledge, and present our solution in a concrete way.

10.3.1 Threats to the Central Server

Before any security policies can be implemented, we identify the security holes in the smart environment being protected. Obviously, the main server is one of the most difficult-to-secure systems in the space. This central system has the highest risk of becoming infected viruses, worms or Trojan horses, and other client-born threats that take advantage of out-of-date or improperly configured security software. Once being compromised, it loses productivity and put its personal and corporate information, and the smart space itself at risk.

One of the most severe security threats to the server coming from wireless sensor networks is base station spoofing. In a common smart area, the wireless sensor networks often gather and relay data to the server via a gateway or base station. This base station typically has resource-rich capabilities in term of its computation, energy and storage. It will be capable of both wired connectivity to the Internet as well as wireless connectivity to the sensor network [10.15]. An attacker gains unauthorized access to the environment by making it appear that a malicious message has come from the base station by spoofing the IP and/or MAC address [10.16,10.17] of that machine. In this way, the attacker can deceive the central server in order to getting all its control information that should have sent to the station to cancel the server's command to the sensor system. Therefore, instead of sending the control packets to the base station, for example to turn the surveillance camera and alarm system on, in order to alert strangers breaking into the house, the server delivered messages to the hacker's machine.

Moreover, dangers to the main server coming from wireless networks are wireless device compromise. The aforementioned example of the virus-infected laptop is a good case of this. Harmfully, if this laptop is compromised by a hacker, it will then be used as a jumping-off place to launch attacks to the server.

Also, risk to the principle server resulting from applications or services implemented on the system can be exploited by attackers because they missed crucial security patches. Once these programs compromised, the system control right will be taken over by hackers. Our server may be planted viruses, opened back doors for serving the intruder's remote control demands.

So what if the central server is compromised? Our front-doors, for example, will automatically open without our permission, camera systems will suddenly be turned off when we are absent from the environment and the smart space is no longer smart at all. And how can we efficiently protect the server from those threats?

10.3.2 Firewall Background

The firewalls that have been discussed in [10.12,10.18,10.19] are network based. We placed them on borders between subnets to manage network traffic crossing from one segment to another. However, our proposed firewall is applied at the host level to control how packets enter and leave the central server.

A firewall security policy is a list of ordered filtering rules that define the actions performed on packets that satisfy specific conditions [10.20]. A rule usually consists of filtering fields and an action field. The filtering fields play an important role in evaluation a packet matching a rule or not. The action is either to allow, which permits the packet coming in or out the secured server, or to deny, which causes the packet to be dropped.

10.3.3 Our Approach

In this section, we propose an approach of using a Home Firewall to protect the smart space from such security problems. In this approach, we rely on following assumptions:

- A unique central server is deployed for serving and supervising all the services in the space.
- All entities of the smart space including sensors, mobile and wireless devices, and applications are not compromised simultaneously.

The proposed home firewall has some characteristics like a host-based firewall which should not be confused with the hardware firewall that is commonly found on usual network perimeters. Rather, it is set of related programs installed and administered on the server to protect a single computer from intruders [10.21]. The Home Firewall acts as the extra logical line of defense against penetration attacks. It enforces the system's defenses by:

- Monitoring incoming traffic and block malicious code
- Screening outgoing packets that infect other resources
- Preventing IP and/or MAC address spoofing
- Blocking different types of reconnaissance probes used to discover vulnerable applications or services running on the system

Basically, the Home Firewall security policy is similar to a traditional firewall one which can be thought of as a pair of policies: one which exists to "deny" default all the traffic, and the other which exists to "allow" default all the traffic except the traffic enforcing explicit policies.

In the former case, the user only needs to take care of deployed services and check those for security risk since the final firewall rule is configured to "deny all" traffic that is not specifically permitted. This put him in pro-active mode. Conversely, the latter case requires that the user should have a comprehensive understanding his system in order to allow default all the traffic from/to his central server. Obviously, the first one is the most appropriate one, and it is the policy that we have used in our proposed model.

10.3.3.1 Models and Configurations

In this section, we present our basic design of the proposed Home Firewall in the smart home described above. This general scheme is shown in Fig 10.2.

In order to clearly explain about our proposed solution, we separate the system into three different sub models as follows:

- Model 1: the system is composed of one Central Server, our Home Firewall, a Wireless Access Point (WAP) for serving wireless connections, and mobile devices (Laptops, PDAs)

- Model 2: the system is composed of one Central Server, our Home Firewall, and Applications and Services implemented on the server

- Model 3: the system is composed of one Central Server, our Home Firewall, one Base Station, and sensing devices such as sensors, camera and alarm systems.

In the proposed firewall designed for all these three models, we enforce the following conditions:

- Username/password authentication mechanism is installed to identify the user who wants to change the current security policies

- The packet filtering module allows the firewall to accept connections from the users and to respond to the user requests

- The decision of how many services, what kind of service, and other parameters such as interval for serving each service should be provided depends on the firewall rule set.

In addition, the Home Firewall holds the following rule table for packet filtering:

Table 10.1 An entry of the proposed firewall rule table

Order	Source MAC Address	Destination MAC Address	Controlled Entity	Interval	Command	Action

The firewall rule table indicates which machine the server can or cannot communicate with, and which service the server can or cannot provide. If a rule field is not used, it will get the default "any" value.

We begin considering the first model. The other models will be argued in turn. In this model (Fig 10.3), our Home Firewall and the Wireless Access Point (WAP) share responsibility and work co-operatively to provide security to the Central Server. Commonly, the WAP controls the Data Link level access whereas the Home Firewall supplies 2-layer defense mechanism including Network layer and Application layer protection.

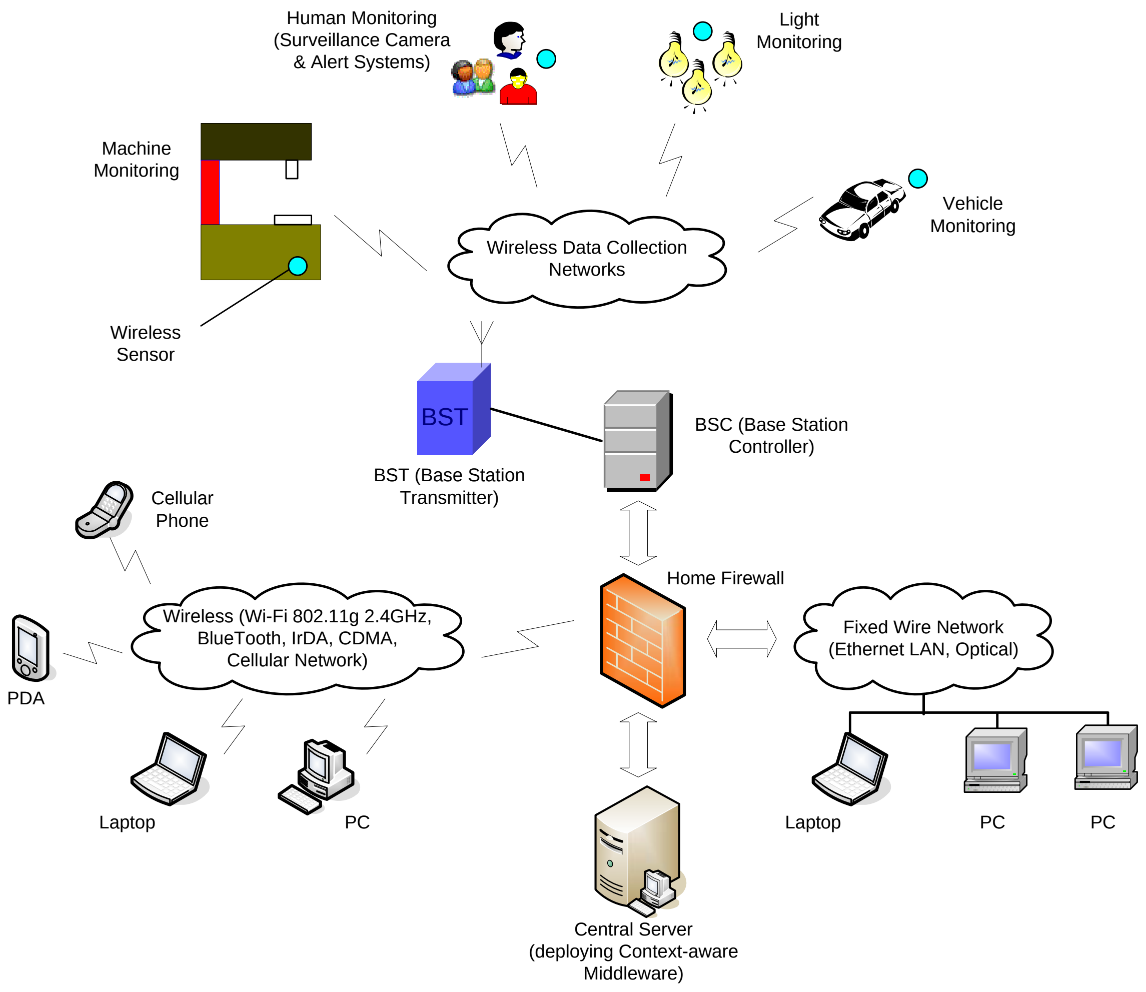

Human Monitoring (Surveillance Camera & Alert Systems)
Light Monitoring
Machine Monitoring
Vehicle Monitoring
Wireless Data Collection Networks
Wireless Sensor
BST
BST (Base Station Transmitter)
BSC (Base Station Controller)
Cellular Phone
Home Firewall
PDA
Wireless (Wi-Fi 802.11g 2.4GHz, BlueTooth, IrDA, CDMA, Cellular Network)
Fixed Wire Network (Ethernet LAN, Optical)
Laptop
PC
Laptop
PC
PC
Central Server (deploying Context-aware Middleware)

Fig 10.2 A smart space with Home Firewall support

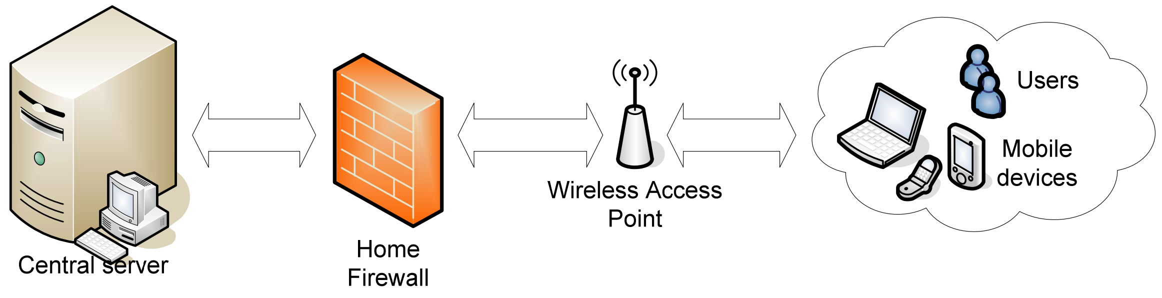


Fig 10.3 The first model of our proposed solution

As indicated above, if our security perimeter is limited to the WAP with some restricted default security policies, the compromise of the WAP will leave the main server vulnerable. This violates the "defense in depth" principle. So the appearance of our proposed Home Firewall in addition to the WAP will make the security perimeter stronger. The firewall manages all the transactions between the user's mobile devices and the central server. If the WAP and/or user's mobile devices are compromised, attackers still have no way to change the behavior of our central server since they don't know the username/password to change the firewall policies. For instance, if a firewall rule is configured as follows:

Table 10.2 The content of an entry in a firewall security policy

Order	Source MAC Address	Destination MAC Address	Controlled Entity	Interval	Command	Action
01	Any	Any	Camera and Alarm Systems	11P.M-6A.M	On	Allow
02	Any	Any	Camera and Alarm Systems	7A.M-10P.M	Off	Allow
03	Any	Any	Any	Any	Any	Deny

Without the protection of our firewall, the central server can be comprised by the attacker after he controlled the user's laptop. He who installed a backdoor for abusing the user's laptop mentioned above will not have a chance to modify the firewall security policy to turn all the camera system off even though they are mastering the user's device if our proposed firewall is deployed. For example, the attacker wants the camera system to be turned off from 11P.M-6A.M in order to successfully break into the house by delivering a request from the user's laptop to the server. However, these command packets from the central server could not be sent to this system because they did not satisfy the rule set and they would be dropped at the firewall layer. Therefore, along with built-in security techniques implemented on the WAP [10.22] and the Home Firewall installed in the server, this model brings better solution for protecting the smart space.

In the second model, depicted in Fig 10.4, the firewall is located in front of the applications and services implemented on the server. It helps preventing these server's programs from being compromised by stopping common hacker's reconnaissance port scanning techniques [10.23]. An attacker who can successfully conduct recon on your server has a much higher likelihood of attempting to compromise your system than an attacker whose recon attempts are thwarted. Once the attacker knows what services that the server is running and their related information, he can activate attacks based on exploiting software vulnerabilities to our server.

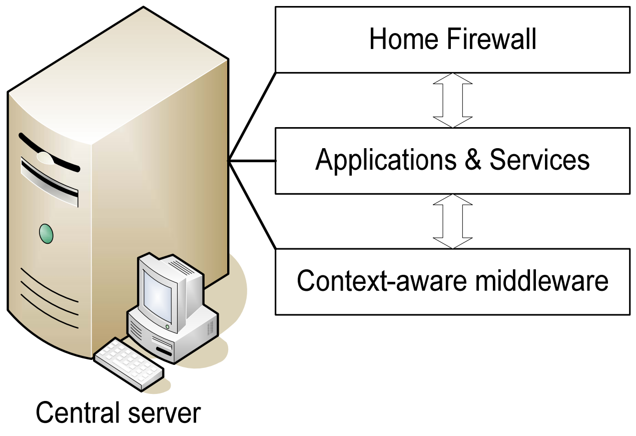


Fig 10.4 The second model of our solution

In order to defend our server from these kinds of potential threats, such as ICMP scanning, TCP scanning, UDP scanning [10.24], we deploy an anti-scanning security policy as follows:

Table 10.3 The content of an entry in an anti-scanning security policy

Order	Source MAC Address	Destination MAC Address	Protocol/ Packet Type	Direction	Action
01	Any	Any	ICMP/Echo Request	In	Allow
02	Any	Any	ICMP/Echo Reply	Out	Deny
03	Any	Any	ICMP/Port Unreachable	Out	Deny

The two first rules are coordinately used for preventing ICMP scanning technique by prohibiting the ICMP replying packets sent back to the attacker. The third rule is used to deny ICMP Port Unreachable packets transmitted back to him for protecting UDP scanning technique. It's because all these packets are exploited to notice the attacker about the status of the server's services. In order to detect the TCP scanning signature, we might say that if there are more than 5 SYN packet attempts to non-listening ports in one minute, then an alarm SMS message should be automatically triggered to the user's cell phone. Therefore, our suggested firewall helps us hiding sensitive application information from malicious probes.

The last model in our explanation, shown in Fig 10.5, is for the circumstance of base station spoofing in the smart space. In such attacks, the adversary intercepts a legitimate communication between the server and the base station. The malicious host then controls the message stream and can eliminate or modify the information sent by the base station without the knowledge of either the server or the base station. In this way, the attacker can deceive the central server into disclosing confidential information by spoofing the identity of the base station, which is seemingly trusted by the server.

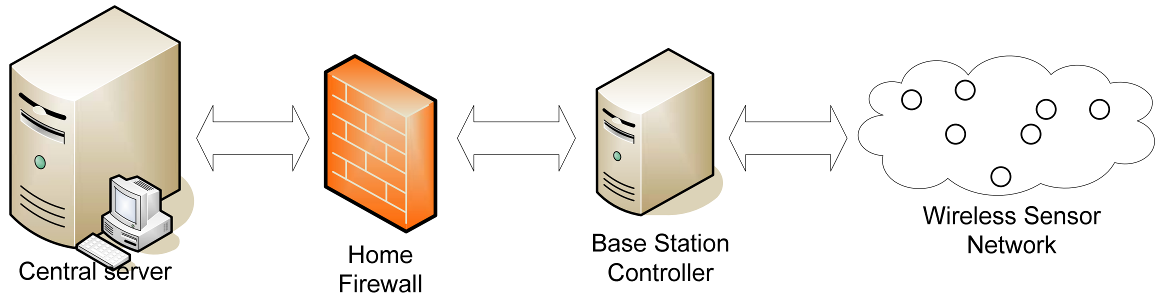


Fig 10.5 The third model of our solution

In order to prevent this kind of attack, we suggest an additional module included in the Home Firewall, the MAC Address Refining (MAR) module. This module is responsible for real-time selecting trusted MAC addresses of available confident base stations in the space. The selected addresses are maintained in an admission list. The MAR module periodically sends an RARP (Reverse Address Resolution Protocol) [10.25] packet to each address in the list. The function of RARP is mapping a MAC address into an IP address. Following this, Reverse ARP should reply one IP address for one network device. If multiple IP addresses return, it means that the MAC address is being exploited by more than one device.

The MAC address is also called a hardware address, and it supposed to be permanent, following the NIC (Network Interface Controller) card attached to a network device. Since an intruder is trying to identify and duplicate existing MAC address of the base station, our defense firewall can quickly detect this duplication. The performance of MAR module makes sure that the MAC addresses in the admission list are trusted addresses and cannot be forged. This list is then employed by the firewall security policies to verify the source MAC address of the packets sent to it for certain that these packets really come from a legitimate base station.

10.4 Discussions

The firewall solution proposed in this paper clearly demonstrates to be as an efficient defense for a smart space. There are many advantages in our model. This solution applies the principle of "defense in depth". For example, if one defensive component of the security perimeter is compromised, like Wireless Gateway, still will our Home Firewall cease the intruder, or at least stretch attacker's energetic and vigorous activities until the user perceives the security violation.

Since the scheme does not introduce any modification to the existing model, it just provides an extra security level for the smart space. Its simplicity and compatibility in implementation could be seen as a few advantages. The Home Firewall is deployed on the central server, so other entities of the smart environment are not affected. It can prevent the server from various attacks, such as base station spoofing, application reconnaissance security scanning, and user wireless device compromise, to make the space more confident.

Nevertheless, this suggested firewall is not a complete solution for all kinds of attack yet. A tricky attacker can use unknown variant of probing techniques for gathering server sensitive information is a good example of this. Additional, our solution supposed to protect a unique service server deployed in the space. Other security techniques to guard a group of server go beyond the scope of this scheme. Also, our model is only suitable for the situation when all entities of the space are not compromised concurrently.

10.5 Related Works

The security in ubiquitous computing would be a major issue as individual, groups, and organizations are unlikely to put personal, important, and mission-critical information over an infrastructure that is either not secure or is not perceived to be secure [10.26]. The security weaknesses of wireless and mobile infrastructure stem from both the use of multiple "incompatible" security schemes and due to "inherent" weaknesses in certain wireless security algorithm (such as wireless LANs) [10.26].

Security problems and vulnerabilities related to ubiquitous computing are generally concerned by Stajano [10.27]. Covington et al. [10.28,10.29] tackled the problem of securing a smart home environment. They refer to this environment as the "Aware Home". However, at the time of writing this paper, there has been no published work on proposing a firewall solution to smart spaces. Most of the suggested firewall models mainly focus on fixed wire networks [10.4,10.5], and a few ones support for wireless networks [10.12]. Thus, the problems arise from the smart space are still open. The act of proposing the Home Firewall to the smart surroundings is the contribution of this work.

10.6 Conclusions and Future Work

Security for smart spaces is really a fascinating and challenging research topic. The inherent features of the ubiquitous computing environment such as dynamism, mobility, and pervasiveness raise new difficult and stimulating tasks. Firewall security, like other security technologies, requires appropriate management and operation in order to safely protect context-aware middleware or services.

In this paper, we formally defined some firewall security policies and proved that these can efficiently defend our context-aware server in risky circumstances in which some smart space entities are compromised. Hence, if our proposed firewall is implemented and administered in a well-defined way, attacks upon the server deployed in the space can be maximally reduced.

We believe that there is lots of work to do in firewall implementation area. Our future research plan includes implementation of major Home Firewall modules, such as packet filtering module for verifying traffic in/out the central server, MAC Address Refining module for protecting base station spoofing, and rule editing module.

10.7 References

[10.1] M. Weiser, "The Computer for the 21st Century," Scientific America, pp. 94-104, Sept. 1991; reprinted in IEEE Pervasive Computing, pp. 19-25, Jan.-Mar. 2002.

[10.2] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," IEEE Personal Communications, pp. 10-17, Aug. 2001.

[10.3] Jalal Al-Muhtadi, Anand Ranganathan, Roy Campbell, M. Dennis Mickunas, "Cerberus: A Context-Aware Security Scheme for Smart Spaces", in Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03).

[10.4] Bryan, J. "Build a Firewall", BYTE, pp. 91-96, April 1995.

[10.5] Bellowin, S. M. and W. R. Cheswick "Network Firewalls," IEEE Communication Magazine, pp. 50-57, Sep. 1994.

[10.6] Xinyi Huang, Susilo W., Yi Mu, Futai Zhang, "Identity-based ring signcryption schemes: cryptographic primitives for preserving privacy and authenticity in the ubiquitous world," Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on Volume 2, pp. 649-654, 28-30 March 2005.

[10.7] Al-Muhtadi J., Ranganathan A., Campbell R., Mickunas M.D., "A flexible, privacy-preserving authentication framework for ubiquitous computing environments," Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference, pp. 771-776, 2-5 July 2002.

[10.8] Sailer R., Giles J.R., "Pervasive authentication domains for automatic pervasive device authorization," Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference, pp. 144-148, 14-17 March 2004.

[10.9] Bellowin, S. M. and W. R. Cheswick. "Network Firewalls," IEEE Communication Magazine, pp. 50-57, Sep. 1994.

[10.10] Cheswick, W. R. and S. M. Bellowin, "Firewall and Internet Security: Repelling the Wily Hacker," Addison-Wesley Publications, 1994, Reading, MA.

[10.11] Ranum, Marcus J., “Thinking About Firewalls,” Trusted Information Systems, Inc. Glenwood, Maryland. Available through WWW at www.tis.com/home/NetworkSecurity/Firewalls.
[10.12] U. Murthy, O. Bukhres, W. Winn, E. Vanderdez. "Firewalls for Security in Wireless Networks," hicss, p. 672, Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7, 19910.
[10.13] Bryan, J. “Build a Firewall”, BYTE, pp. 91-96, April 1995.
[10.14] Hung Q. Ngo, Anjum Shehzad, Kim Anh Pham, Maria Riaz, Saad Liaquat, and S. Y. Lee, “Developing Context-aware Ubiquitous Computing Systems with a Unified Middleware Framework,” Embedded and Ubiquitous Computing 2004.
[10.15] Jing Deng, Richard Han, and Shivakant Mishra, “Enhancing Base Station Security in Wireless Sensor Networks,” Technical Report CU-CS-951-03, Department of Computer Science, University of Colorado, April 2003.
[10.16] CERT, “IP Spoofing Attacks and Hijacked Terminal Connections, CA-95:01,” Computer Emergency Response Team, Carnegie Mellon University, Pittsburgh, Pennsylvania, Sept. 1991.
[10.17] Lorri Ely Berend, “Wireless Network Hacking,” GSEC Practical Assignment, v1.4b, Avaiable through WWW at http://www.giac.org/certified_professionals/practicals/gsec/2372.php
[10.18] Schuba C.L., Spafford E.H., “A reference model for firewall technology,” Computer Security Applications Conference, 1997. Proceedings., pp. 133-145, 13th Annual 8-12 Dec. 1997.
[10.19] Yi Xu, Lee H.C.J., “A source address filtering firewall to defend against denial-of-service attacks,” Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th, pp. 3296-3300 Vol. 5, 26-29 Sept. 2004.
[10.20] Al-Shaer E.S., Hamed H.H., “Discovery of policy anomalies in distributed firewalls,” INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, pp. 2605-2616, 7-11 March 2004.
[10.21] Noakes-Fry K., and Diamond T., “Personal firewalls: technology overview,” Retrieved December 19, 2004 from http://www.gartner.com, Document G001211810.
[10.22] John Tsai, “Wireless Access Point/Router Security,” Available through WWW at http://www.rescomp.berkeley.edu/besecure/how-to/wireless/
[10.23] Andy Millican, “Network Reconnaissance – Detection and Prevention”, GSEC v1.4b, 23 January 2003, Available through WWW at http://www.giac.org/certified_professionals/practicals/gsec/2473.php
[10.24] Prabhaker Mateti, “Port Scanning,” College of Engineering & CS, Wright State University, Dayton, Ohio 45435-0001, 29 June 2001. Available through WWW at http://www.cs.wright.edu/~pmateti/Courses/499/Probing/index.html.
[10.25] Finlayson, Mann, Mogul, Theimer, “RFC-903 Reverse Address Resolution Protocol,” Network Working Group. June 1984.
[10.26] Upkar Varshney, “Network Access and Security Issues in Ubiquitous Computing,” CIS Department, Georgia State University, Atlanta.
[10.27] F. Stajano, Security for Ubiquitous Computing: Halsted Press, 2002.
[10.28] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd, "Securing context-aware applications using environment roles," presented at Proceedings of the Sixth ACM Symposium on Access control models and technologies, Chantilly, Virginia, United States, 2001.
[10.29] 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.

C h a p t e r 1 1

PRIVACY CONTROL

11 Privacy Control

11.1 Introduction

The major advances in distributed systems and mobile computing have converged to enhance global interconnectivity. This has fueled the idea of ubiquitous computing and active information spaces where users can access services, run programs, utilize resources, and harvest computing power anytime and anywhere. This new generation of ubiquitous computing enables the delivery of integrated services and multimedia-enabled applications that are no longer bound by time or location barriers. Ubiquitous computing promotes the proliferation of embedded devices, smart gadgets, sensors and actuators. These devices will be everywhere, performing regular tasks, providing new functionality, extending the reach of traditional computing to physical spaces, and allowing users to interact seamlessly with the surrounding environment.

Physical spaces augmented with sensors and actuators that can locate users, detect their presence, and track their whereabouts will be commonplace in this new and exciting computing paradigm. These sensors will play a major role in bridging the virtual computing world with the physical world and boosting the productivity of users and the availability of computing resources. However, these very features could severely threaten the privacy of users. For instance, the mentioned services can be exploited by intruders, malicious insiders, or even "curious" system administrators to track or electronically stalk particular users. Although encryption provides confidentiality by hiding information flowing through communication channels from eavesdroppers (e.g., an insider or a system administrator), an eavesdropper can still gather the network addresses or physical locations of the communicating parties. The lack of privacy in today's networks and distributed systems is well-documented [11.1,11.2]. Similar concerns arise for ubiquitous computing environments [11.1]. While several approaches have tried to address these problems (see Section 11.2) the solutions presented are either only concerned with anonymous web browsing or with trusted third parties that store the location information of users and only disclose it to authorized principals.

This report tried to serve as an introductory reading for the interested computer science researcher, especially in the field of ubiquitous computing. It gives a brief background on privacy – its issues surrounding it, touches on various legal implications, and tries to develop a comprehensive set of mechanisms and guidelines for designing privacy-aware ubiquitous systems.

11.2 Related work

Privacy is actually a fuzzy term that is often overloaded to mean a large variety of things. Therefore, before proceeding any further, it is important to clarify the scope of user privacy that we strive to achieve in a ubiquitous computing environment. Our goal is to achieve the following:

● Location privacy: Neither the system nor the users of the system will be able to know the exact physical location of a user, unless that user decides to disclose such information or if another person physically sees that user at that location.

● Anonymous connections: If two parties decide to communicate with each other, then other users in the system will not know who the communicating parties are, unless one of the communicating endpoints decides to disclose such information.

● Confidentiality: If both endpoints of a communication agree, they can make the content of their communication confidential, such that neither the system nor other users in the system can read the contents of the communication [11.4].

In this section, we present some of the existing research that relate to our problem. Compared to the amount of research efforts directed towards ubiquitous computing, very little attention has been paid to the security aspects of ubiquitous computing so far. However, in this section, we will consider some of the approaches that attempt to achieve anonymity on the Internet. Some projects try to provide a way to hide a user's identity while communicating over an open network while others try to provide a communication channel that is immune to traffic analysis, hence, providing anonymity from eavesdroppers. We describe some of the representative works in this section.

In [11.3], Marc Langheinrich warns us about the possibility of an Orwellian nightmare in which current ubiquitous computing research continues on without considering privacy protection in the system. He proceeds to describe the design principles of privacy-aware ubiquitous systems. Some of the principles proposed are yet to be implementable with current technology but the paper gives a good general guideline for privacy issues in ubiquitous computing systems. The crucial point of this paper is that unless you consider the privacy concerns since the initial stages of a ubiquitous system design, it is very likely to end up becoming a ubiquitous surveillance system. Our approach fits the spirit of this paper in the sense that we integrate the privacy concerns into the routing itself.

Previous research on privacy and anonymity on the Internet can be classified into roughly two categories: user anonymity and anonymous communication. User anonymity aims at providing the users anonymity while they are using the network by letting them hide their identity from the communicating peers. Research on anonymous communication focuses on providing a communication channel that is immune to traffic analysis so that the communicating parties can be anonymous against the eavesdroppers.

Anonymizer [11.5] and SafeWeb [11.6] are two user anonymity solutions provided to World Wide Web users. Anonymizer is a centralized 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 the websites cannot infer the users' identities. 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.

Crowds [11.7] by Aviel Rubin et al. is one of the approaches on anonymous communication. A Crowd is a set of voluntarily cooperating hosts. Any message that requires anonymity first channels into one of the Crowd shosts and then enters a loop until it finally gets out of the Crowd and arrives at the destination. Using statistical forwarding decisions, Crowds can effectively hide the communication pattern of a user. Another similar approach is Onion Routing [11.8]. Users can use the deployed set of Onion routers in the Internet to achieve a level of privacy similar to that of Crowds. One difference, however, is that the 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 [11.9] is an approach to counter traffic analysis in real-time. NetCamo models the traffic patterns of nodes or networks and provide a real time rerouting and padding to hide the communication pattern.

11.3 Description

11.3.1 Mechanisms for Privacy Protection

Before setting out to assemble a technical infrastructure for privacy protection, we need to take care of the array of mechanisms available to us – both those working in our favor, and against. Building a technical infrastructure, whether for privacy protection or other purposes, cannot be done in isolation from the legal and social realities that inevitably surround it. Otherwise we might easily run the risk of creating unworkable or unacceptable solutions.

The field of ethics in general, and technology assessment in particular, can provide valuable insights into the requirements and limits of any privacy solution, as it reflects the moral realities of how much or how few privacy is deemed desirable. Another important component is trust, since data collection systems require some basic trust in either the technology itself, the entities collecting or using the data, or law enforcement mechanisms that allow interactions with un-trusted parties.

Corresponding privacy legislation can often help strengthening any privacy conserving system. While some basic similarities exist, legal protections differ substantially around the world. The sectorial framework in the US have seen a number of recent additions that specifically address issues such as location privacy , while European law with its more comprehensive protection still requires corresponding updates that take into account the recent technological developments.

Of the existing technical solutions, maybe the most prominent ones are those for encryption and authentication. While often used synonymous with privacy tools in general, such security mechanisms cover an important part of technical privacy protection, though not the complete range of issues. Anonymization and pseudonimization are another building block in providing privacy when the full disclosure of one's identity is not necessary. These mechanisms are complemented with transparency and trust tools, such as the Web technology P3P, which allow data collectors to describe their collection policies in a machine-readable format and communicate these to their data subjects.

By being aware of the full range of mechanisms that are at work in the field of privacy – social mechanisms such as moral, ethics, and trust; legal mechanisms such as laws and regulations; and technical mechanisms for solving different distinctive problems – we can hope to build a comprehensive solution that solves the right problem, in the right manner, with the right mechanisms.

11.3.2 Guiding Principles

With the wealth of mechanisms in mind, we can set out to draw up a number of principles that are to guide technical development. A a starting point for such guidelines, we draw from a well-established set of practices with more than thirty years of experience: the Fair Information Principles, drawn up in 1973 in a report by the US Department of Health, Education, and Welfare (HEW), which form the basis of practically all modern privacy laws today.

Among the most fundamental requirements is that of notice and disclosure: There should be, simply stated, no hidden data collections. Ubiquitous computing systems will per definition be ideally suited for covert operation and illegal surveillance; no matter how much disclosure protocols are being developed. It will always take special detection equipment to be reasonably sure that a certain room or area is not being monitored by others. But for those who want (and are bound by law) to "play it by the book," some kind of announcement system would be helpful that would allow them to openly announce otherwise covert data collections to customers, employees, and visitors, but also to family members and friends.

Given that individuals know about data collections taking place, they can exercise another fundamental requirement of data collection regulations: choice and consent. Again, the area of pervasive computing poses new challenges in this respect, as not even a button-click – the established means of giving consent on the Web – will be available in most of these smart environments. Users will need delegation mechanisms that allow for an automated pickup privacy announcements and subsequent decision-making on the basis of previously established preferences.

Should an offered service be not to the use's liking (with respect to his or her privacy), a choice should exist involving anonymity and pseudonymity. While several anonymization schemes for Internet service access exist, their deployment in future computing scenarios is made difficult by the fact that real-world data is much harder to anonymize than virtual data. Especially the realm of location anonymity and pseudonymity would need to be part of any privacy protection scheme for ubiqutious computing.

Adequate security, i.e. encryption of electronic communication and storage, as well as authentication as access control, must of course also be involved whenever data collection takes place, otherwise promised collection and handling practices can hardly be guaranteed. Although alarge number of encryption mechanisms and security procedures exist, finding the right balance between security and usability will be a challenge for any application involving invisible computers.

Trusting a system, and especially a system as far-reaching as a ubiquitous one, requires a set of regulations that separate acceptable from unacceptable behavior, together with a reasonable mechanism for detecting violations end enforcing the penalties set forth in the rules. Technology can help implementing specific legal requriements such as access and recourse, so that data subjects can see for themselves what information about them is on file and potentially correct of delete it.

Even with a ubiquitous computing systems supporting all of the above requirements, situations may arise where getting the explicit consent from a subject beforehand will be difficult, if not infeasible. Complementary mechanisms such as principles of proximity and locality should be embedded in the underlying infrastructure in order to not only prevent accidental data collections (e.g., a memory amplifier recording without its owner being present) but also limit data dissemination (e.g., keeping sensory data stored close to its collection place).

Whether the above six points – notice and disclosure, choice and consent, anonymity and pseudonymity, adequate security, access and recourse, and proximity and locality – can be realized in future computing systems, will of course depend to a large extend on the intricate interplay between technology, social norms, and legal obligations that together will form the design space of any such environment. What we can hope to achieve is building as system that complements, rather than replaces, these mechanisms. We call this privacy awareness, rather than privacy protection, indicating that its effectiveness rests on supporting existing social and legal tools, not on replacing them.

11.4 Summary

Ubiquitous computing is an emerging research area with great potential. However, without careful consideration for user privacy from the ground up, there is a fair possibility of creating a ubiquitous 'surveillance' system instead. To avoid this undesirable future, we contend that the privacy and anonymity of users in ubiquitous computing environments should be considered seriously and carefully from the very beginning of the system design phase.

11.5 References

[11.1] M. Reiter and A. D. Rubin, "Crowds: Anonymity for Web Transactions," ACM Transactions on Information and SystemSecurity (TISSEC) Volume 1, Issue 1, November 1998.
[11.2] J. Schwartz, "As Big PC Brother Watches, Users Encounter Frustration," The New York Times, September 5, 2001.
[11.3] M. Langheinrich, "Privacy by Design – Principles of Privacy-Aware Ubiquitous Systems," ACM UbiComp 2001, Atlanta, GA, 2001.
[11.4] J. Al-Muhtadi, Roy Campbell, Apu Kapadia, M. Dennis Mickunas, Seung Yi, "Routing Through the Mist: Privacy Preseving Communication in Ubiquitous Computing Environments", University of Illinois at Urbana-Champaign.
[11.5] Anonymizer, http://www.anonymizer.com
[11.6] SafeWeb, http://www.safeweb.com
[11.7] M. Reiter and A. D. Rubin, "Crowds: Anonymity for Web Transactions," ACM Transactions on Information and System Security (TISSEC) Volume 1, Issue 1, November 1998.
[11.8] 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.
[11.9] Y. Guan, C. Li, D. Xuan, R. Bettati, and Wei Zhao, "Preventing Traffic Analysis for Real-Time Communication Networks," Proceedings of The IEEE Military Communication Conference (MILCOM) '99, November 19911.

Chapter 12

SECURITY POLICY

12 Security Policy

12.1 Introduction

Ubiquitous computing (ubicomp) integrates computation into the environment, rather than having computers which are distinct objects. We hope that embedding computation into the environment would enable people to move around and interact with computers more naturally than they currently do. One of the goals of ubiquitous computing is to enable devices to sense changes in their environment and to automatically adapt and act based on these changes based on user needs and preferences [12.1], as long as system's security policies are not violated.

Ubiquitous computing has unique features that make it different from other computer science domains. They are ubiquity, invisibility, sensing, heterogeneous and with many resource-constrained devices. With these features, ubiquitous environment is not only the virtual world as traditional computing environment but the strong combined environment of virtual and physical world. Therefore, it also raises the risk of privacy and security.

Traditional security focuses on authentication, access control, confidentiality, integrity, availability, and trust. In the new environment, these problems are much more complex since ubiquitous environment is more dynamic, more distributed, more invisible. Therefore, we need to view security problems in a new paradigm and explore them thoroughly under the above effects.
Security requirements in ubiquitous computing

The ubiquitous computing bring us a new environment, it also imposes new requirements on security and privacy. Due to its unique features, security services will also have unique features compared with traditional security services. They are ubiquity, context-awareness, invisibility (non-intrusive and transparent), and light-weight. They need to be adaptability and multilevel for supporting a heterogeneous environment and the dynamic characteristic of the environment.

The security system 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 [12.2].

12.1.1 Definitions

In this section, we will clarify some terms related with security and security policy in ubiquitous computing.

As we know, a security policy is a statement that partitions the state of the system into a set of authorized (or secure) states, and a set of unauthorized (or non- secure) states. In which, a secure system is a system that starts in an authorized state and cannot enter an unauthorized state. Thus, a breach of security occurs when a system enters an unauthorized state.

And, to precisely describe the security requirements of the model and to provide a framework for the specification, implementation, and verification of the security properties of the network, it is necessary to have a mathematical formulation of the model. We call it security model. We can have its definition in RFC 2828 as follows: "a schematic description of a set of entities and relationships by which a specified set of security services are provided by or within a system".

From the above section, we knew about one unique feature of security of ubiquitous computing, that is context-aware. Here, we need to define a new term: security context. Security context as a set of information collected from the user's environment and the application's environment and that is relevant to the security infrastructure of both the user and the application (Kouadri and Brezillion).

We also consider context explicitly as a guide to deduce which mechanisms to enforce in each situation (security context) that we call "context-based security".

12.2 Background and related work

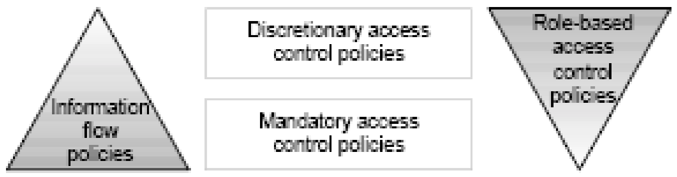
12.2.1 Security Policy Overview

The definitions most frequently proposed for computer security identify three primary objectives for security: confidentiality (sometimes called secrecy) related to the disclosure of information, integrity related to the modification of information, and availability related to the denial of access to information. To achieve these objectives three mutually supportive technologies are used: Authentication, Access Control and Audit. Access control is concerned with limiting the activity of legitimate users who have been successfully authenticated, and is the process of ensuring that every access to a system and its resources is controlled and that only those accesses that are authorized can take place. There are three basic components to an access control system: the subjects, the targets and the rules which specify the ways in which the subjects can access the targets. The set of high-level rules according to which access control must be regulated are traditionally called access control policy [Samarati et al. 2000]. The study of access control has identified a number of useful access control models, which provide a formal representation of security policies and allow the proof of properties about an access control system. Note that the use of the term policy is often used in the literature to refer to both high-level security policies as defined above, and actual authorization rules to be enforced.

Access Control Models

Access control policies have been traditionally divided into discretionary and mandatory policies. Discretionary policies are concerned with the specification of authorization rules to govern the access of users to the information, whereas mandatory policies are mostly concerned with controlling information flow between the objects of a system. Information flow policies are often described as a separate type of policy, and are directly related to the issue of data confidentiality. Recently role-based access control policies are attracting increasing attention, particularly in commercial applications, and are often seen as an alternative to traditional discretionary and mandatory access control. The following figure shows a relationship between the four generic models mentioned.

We view role-based policies as more closely applying the principles of discretionary access control. On the other hand information flow policies are more closely related to mandatory access control.


Information flow policies
Discretionary access control policies
Mandatory access control policies
Role-based access control policies

Over the years other, often more sophisticated security models have been proposed to formalize security policies required for commercial applications. The most well known is the Clark-Wilson model [Clark et al. 1987], which attempts to present in a formal, abstract way commercial data processing practices. Its main goal is to ensure the integrity of an organization's accounting system and to improve its robustness against insider fraud. The Clark-Wilson model recommends the enforcement of two main principles, namely the principle of well-formed transactions where data manipulation can occur only in constrained ways that preserve and ensure the integrity of data, and the principle of separation of duty. The latter reduces the possibility of fraud or damaging errors by partitioning the tasks and associated privileges so cooperation of multiple users is required to complete sensitive tasks. Authorized users are assigned privileges which do not lead to execution of conflicting tasks. This principle has since been adopted as an important constraint in security systems. Other models include a security policy model that specifies clear and concise access rules for clinical information systems [Anderson 1996]. This model is based on access control lists and his authors claim it can express Bell-LaPadula and other lattice-based models. Finally the Chinese-wall policy (see description in [Anderson et al. 2001]) was developed as a formal model of a security policy applicable to financial information systems, to prevent information flows which cause conflict of interest for individual consultants. The basis of the model is that people are only allowed to access information which is not held to conflict with any other information that they already possess. The model attempts to balance commercial discretion with mandatory controls, and is based on a hierarchical organisation of data. It thus falls in the category of lattice-based access control models.

Recent proposals include a trend towards languages able to express different access control policies in a single framework in order to provide a common mechanism able to enforce multiple policies. This enables uniform specification and composition of access control policies across administrative domains and for a number of different platforms.

Another direction is certificate-based access control aimed at specifying trust policies for access to resources from un-trusted sources e.g. over the Internet. Trust has long been tied to authorization: "Access control consists in deciding whether the agent that makes a statement is trusted on this statement; for example, a user may be trusted (hence obeyed) when he says that his files should be deleted." [Abadi et al. 1993]. However, its only very recently that work on certificate-based authorization has been intensified, as part of trust management systems.

Policy Specification Approaches

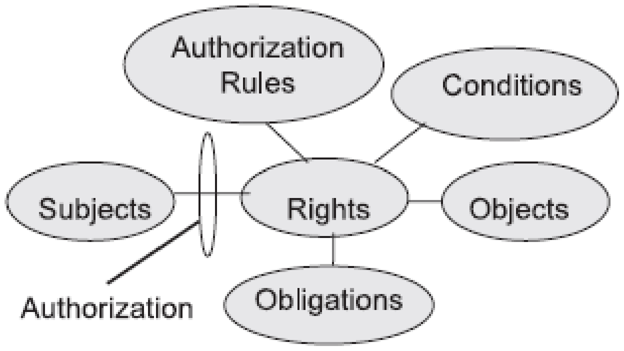
There are three main categories: policy specification languages, rule-based specifications, and formal logic languages. From a human input standpoint, the best way to specify policies is using a policy language because it provides considerable flexibility compared to the other approaches. However, the use of a generic high-level language compromises the ability to analyze policy specifications, a process that can be made considerably simpler with the design of declarative languages. In the rule-based approach policies are specified as sequences of rules of the form: if condition then action, and are mostly applied to quality of service management in IP networks. Finally, logic-based approaches are driven by the need to analyze the policy specification, but generally fail to directly map to an implementation and are not easily interpreted by humans. Formal logic is mostly used in the specification of security policies.

There are many ways to divide the discussion on the various policy specification approaches, e.g. based on the granularity of specification, based on the functionality, or based on the application domain. To specify security polices, we can use logic-based languages, high-level languages [12.3].

12.2.2 Related work

Recently, there are some works about access control model in ubiquitous computing.

One new access control model is the Usage Control (UCON) concept. The UCON model encompasses traditional access control, trust management, and digital rights management (DRM) to control the access to and usage of digital information objects. The UCON model consists of three core components and three additional components that are mainly involved in the authorization process. Core components comprise subjects, objects, and rights while the additional components include authorizations, conditions, and obligations.



Subjects hold and exercise certain rights regarding objects. A subject can be a user, a group, a role, or a process. Objects are entities to which subjects hold rights allowing the subjects to access or use objects. Objects can be either privacy sensitive or privacy non-sensitive. Examples of objects include documents and audio, video, and executable files. Rights are privileges that a subject can hold regarding an object. Rights consist of a set of usage functions that enable a subject to access objects. Rights include rights of access (allowing access to and use of objects) and rights for delegation of rights.

Authorization rules are a set of decision factors used to check whether a subject is qualified to use certain rights with respect to an object. Examples of authorization rules include identity or role verification, proof of payments, metered payment agreements, and usage log report agreements. Conditions are a set of decision factors that the system should verify during the authorization process along with authorization rules before allowing the use of rights regarding a digital object. Conditions differ from authorization rules in that conditions are used to check whether existing limitations and the status of usage rights regarding an object are valid. Some examples of dynamic conditions are the number of times an object can be used (e.g., a video file can be viewed five times) and the accessible time period (e.g., during business hours). Finally, obligations are mandatory actions that a subject has to perform after obtaining or exercising rights with respect to an object. Traditional access control has paid little attention to the obligation concept. The UCON model is a promising approach for the next generation of access control because it covers both security and privacy issues concerning current business and information system requirements in a systematic way.

On the other hand, the UCON model leaves open the architecture and mechanisms for providing trusted attributes. This is one of the important challenges as we look ahead. The delegation of rights is among the crucial issues that should be covered by the UCON model. In addition, there should be a clear description of administration issues.

Another new access control model is UbiCOSM (Ubiquitous Context-based Security Middleware). As well as allowing security administrators to specify system access control policies to prevent illegal access to local resources, it also allows users to specify privacy requirements to regulate the disclosure of their personal information when entering a new context. UbiCOSM uses the context as a foundation for security policy specification and enforcement processes. Unlike traditional access control models, permissions are directly associated with contexts, instead of user identities/roles: any mobile user/device acquires a set of permissions by entering a specific context. UbiCOSM exploits

an RDF-based standard format to express access control permissions. The definition of permission includes a Name that identifies the permission, an Action that specifies the allowed operation, a Target representing the resource the specified action can be applied to, and a Kind representing the positive or negative meaning of the permission. UbiCOSM also allows permissions to be associated with an individual context (simple association) or to multiple contexts composed of and, or, and dependence relationships. The policy semantics change according to the type of association. Thus, access control policies are enforced on any mobile client currently operating within that context.
One another project that bases on context is SESAME dynamic context-aware access control mechanism for pervasive Grid applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context. The underling dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC).

"Context-Based Security Policies: A New Modeling Approach" is also a project that use context as additional material for access control. It shows a solution for modeling security policies by using graph, called "Contextual graphs for modeling context based security policies".
Context Sensitive Access Control is another interesting project that proposes an authentication solution by verifying context.

In case of peer-to-peer and ubiquitous computing systems, we have a novel and interesting authentication problem in wireless networks, secure transient association. One solution to enable secure transient association is the Resurrecting Duckling security policy model. The name of this model was inspired by ducking behavior whereby a duckling emerging from its egg will recognize as its mother the first moving object it sees that makes a sound. This phenomenon is called imprinting. The Resurrecting Duckling security policy model enables a duckling (a slave device) to imprint upon itself a mother duck (a master) through the transfer of an imprinting key, or "soul". Once the slave device is imprinted, it remains faithful to the master as long as that soul persists. When the duckling dies, the soul dissolves, and the duckling's body is ready for imprinting with respect to a new mother duck. During the imprinting phase, a shared secret must be established between the duckling and the mother. If at least one of the two principals (the mother and duckling) can perform the public key operations (decrypt and sign), the other device then simply generates a random secret and encrypts it under the public key of the powerful device from which it gets back a signed confirmation [12.4].

Although most of the efforts on security policy specification focus on the use of formal logic, some approaches have been proposed for high-level security languages.

Lalana Kagal et. al describes Rei - A Policy Language for Pervasive Computing Environment. Rei is based on demonic concepts and includes constructs for rights, prohibitions, obligations and dispensations (deferred obligations). The language consists of a few simple constructs that are extremely flexible and allows different kinds of policies (security, privacy, management, conversation etc.) to be specified. The policy language is not tied to any specific application and permits domain specific information to be added without modification. As our policy language is geared towards environments that consist of several domains we believe that there is a potential for policy conflicts. The language includes two constructs for specifying meta-policies that are invoked to resolve conflicts; setting the modality preference (negative over positive or vice versa) or stating the priority between policies. For example, it is possible to say that in case of conflict the Federal policy always overrides the State policy. Rei models speech acts like delegation, revocation, request and cancel that allow policies to be less exhaustive and allow for decentralized security control.

Rei includes three types of constructs: (i) policy objects, (ii) meta policies and (iii) speech acts. (i) The policy objects represent rights, obligations, prohibitions and dispensations. The *has* policy construct allows these objects to be associated with different entities creating policy rules. This allows for reuse of policy objects. For example, the same right to read a certain file could be associated with different entities in different policy domains. (ii) The policy language contains meta-policy specifications for conflict resolution. These include constructs for specifying precedence of modality and priority of policies. (iii) Rei models four speech acts that can be used within the system to modify policies dynamically: delegate, revoke, cancel and request. In order to make correct policy decisions, we assume the presence of a monitoring service that sends all relevant speech acts to the policy engine. Associated with the policy language is the policy engine that interprets and reasons over the policies, speech acts and domain information to make decisions about users rights and obligations [12.5].

12.3 Description

12.3.1 Issues and challenges of security policy in ubiquitous computing

A ubiquitous computing environment creates new challenges that cause data security to differ from traditional system protection. First, the environment is often unfamiliar to the users. They will not have a trust relationship with the owners of the environment – as they might have with a local system administrator – that is appropriate for handling their security and private information. Second, user access rights change dynamically with respect to their relationship with the mechanisms by which data are generated and sometimes the users cannot be predetermined. For example, a number of users can record a meeting using a camera that is administered by the environment. These users should have access to the video produced during the meeting period, but not other video segments. The system must be able to associate a piece of information with the correct set of users while it is being produced.

12.3.2 Proposals

12.4 Summary

12.5 References

[12.1] http://en.wikipedia.org

[12.2] Jalal Al-Muhtadi, Anand Ranganathan, Roy Campbell, M. Dennis Mickunas, “Cerberous – a context-aware security scheme for smart spaces”, Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom’03), 2003

[12.3] Nicodemos C. Damianou, “A Policy Framework for Management of Distributed Systems”, 2002

[12.4] Shigeki YAMADA, Eiji KAMIOKA, “Access Control for Security and Privacy in Ubiquitous Computing Environments”, IEICE TRANS. COMMUN., VOL.E88–B, NO.3 MARCH 2005

[12.5] Lalana Kagal et al, “A Policy Language for a Pervasive Computing Environment”