A CERTIFICATE BASED, CONTEXT AWARE ACCESS CONTROL MODEL FOR MULTI DOMAIN ENVIRONMENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AHMET YORTANLI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

FEBRUARY 2011

Approval of the Graduate School of Informatics

Prof. Dr. Nazife Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Yasemin Yardımcı Çetin
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Altan Koçyiğit
Supervisor

Examining Committee Members

Prof. Dr. Nazife Baykal	(METU, II)	
Assist. Prof. Dr. Altan Koçyiğit	(METU, II)	
Dr. Ali Arifoğlu	(METU, II)	
Assist. Prof. Dr. Erhan Eren	(METU, II)	
Assist. Prof. Dr. Cüneyt Sevgi	(Işık Ün., IT)	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: AHMET YORTANLI

Signature :

ABSTRACT

A CERTIFICATE BASED, CONTEXT AWARE ACCESS CONTROL MODEL FOR MULTI DOMAIN ENVIRONMENTS

Yortanlı, Ahmet
M.Sc., Department of Information Systems
Supervisor : Assist. Prof. Dr. Altan Koçyiğit

February 2011, 79 pages

A certificate based approach is proposed for access control operations of context aware systems for multi domain environments. New model deals with the removal of inter-domain communication requirement in access request evaluation process. The study is applied on a prototype implementation with configuration for two different cases to show the applicability of the proposed certificate based, context aware access control model for multi domain environments. The outputs for the cases show that proposed access control model can satisfy the requirements of a context aware access control model while removing inter domain communication needs which may cause some latency in access request evaluation phase.

Keywords: Ubiquitous Computing, Context Aware Computing, Access Control Model, Certificate Based Access Control Model, Multi Domain Access Control Model

ÖZ

ÇOKLU ORTAMLAR İÇİN SERTİFİKA TABANLI, BAĞLAM BİLİNÇLİ BİR ERİŞİM KONTROL MODELİ

Yortanlı, Ahmet
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi : Yar. Doç. Dr. Altan Koçyiğit

Şubat 2011, 79 sayfa

Bu tezde çoklu ortamlar için bağlam bilinçli sistemlerin erişim kontrol işlemleri için sertifika tabanlı bir yaklaşım önerilmiştir. Yeni model erişim isteğinin değerlendirilmesi sürecinde ortamlar arası iletişim gereksiniminin ortadan kaldırılmasını hedeflemektedir. Çoklu ortamlar için sertifika tabanlı, bağlam bilinçli bir erişim kontrol modelinin gerçekleştirilebilirliği bir ilk örnek uygulama üzerinde iki farklı senaryonun yapılandırılması ile gösterilmiştir. Prototip üzerinde uygulanan senaryoların sonuçları sunulan erişim kontrol modelinin, bağlam bilinçli sistemlere ait erişim kontrol modellerinin gereksinimlerini karşılarken aynı zamanda giriş isteğinin değerlendirilmesi aşamasında ihtiyaç duyulan ortamlar arası iletişim gereksiniminin ortadan kaldırılmasını sağladığını göstermiştir.

Anahtar Kelimeler: Aynı Anda/Heryerde Bilişim, Bağlam Bilinçli Sistemler, Erişim Kontrol Modelleri, Sertifika Tabanlı Erişim Kontrol Modelleri, Çoklu Ortam İçin Erişim Kontrol Modelleri

To my wife ...

ACKNOWLEDGMENTS

I would like to thank my supervisor Assist. Prof. Dr. Altan Koçyiğit for his guidance, support and motivation throughout the development of this thesis; and for giving me freedom I needed for my graduate studies.

I would like to express my thanks to the jury members, Prof. Dr. Nazife Baykal, Dr. Ali Arifoğlu, Assist. Prof. Dr. Erhan Eren and Assist. Prof. Dr. Cüneyt Sevgi for reviewing and evaluating my thesis.

I would like to thank to TÜBİTAK UEKAE BİLGEM / G222 for supporting my academic studies.

I would like to thank to my family for bringing me up and making me who I am with their love, trust, understanding and every kind of support throughout my life.

Finally special thanks to my wife, Jale Betül Yortanlı, for her support and patience during my academic studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

During 80's and beginning of 90's mainframes satisfied the main computing needs of mankind. After the mainframes where groups of people share the same computer, computing devices became smaller and cheaper and desktop computing paradigm took place. Everyone can own his/her own computing device with desktop computing. With the rise of Internet and network technologies, people start to use more than one computer device at the same time. Eventually, computing devices have become much smaller and cheaper and even they have become invisible in some cases. Today, ubiquitous computing paradigm is rapidly taking place and computing devices are invading into daily life of people without interrupting them and a group of computing devices start to share individual users.

Weiser and Brown [1] describe the evolution of computing as outlined above. As the computing trends show, ubiquitous computing will become the main type of computing in the following years.

1.1 Overview

Context aware computing, one of the main component of ubiquitous computing environments, refers to computing systems which can sense the environment and sense changes in its environment, and adapt their states accordingly. By definition, context is any information that can be used to define the situation of an entity and context aware application is an application that uses context to provide suitable information or services to the users [2].

Similar to standard network based applications, security is one of the most important problems in context aware computing. With standard security issues like protection from unauthorized

access, issues like protection from processing irrelevant context information, sharing of security information with other domains, uninterrupted service access for user while the user is moving between domains are also very important for context aware applications.

One of the sub research topics of security for context aware computing is context aware access control models which mainly deal with controlling resource access requests of users according to their privileges and current context information. Studies in this field can be categorized according to the number of domains involved. There mainly two kinds of models: single domain access control models and distributed (or multi domain) access control models. Single domain models are designed for systems serving a single domain and in such systems there is not so much interrelation with other domains. Therefore such models mainly deal with local problems of access control such as finding more flexible policy evaluation techniques to support various context types. On the other hand distributed (or multi domain) access control models deal with issues in access control across multiple domains such as serving other domains' users, providing uninterrupted service access, retrieving user privileges from the home domains. In this thesis, we propose a context aware access control model for multi domain environments.

1.2 Objectives and the scope of the study

One of the problems of context aware access control models for multi-domain environments is the latency caused by the need for inter-domain interaction during the access request evaluation phase. The objective of the research presented is to devise a context aware access control model for multi-domain environments that eliminates the requirement of inter-domain communication during access request evaluation phase. To satisfy this objective, a certificate based access control mechanism is proposed.

Our approach is based on making an abstraction of users of different domains in access control operations and managing certificate related issues across multiple domain environments. For access control evaluation, we define policy rules and we propose methods to share these rules between domains and to remember and use the policy rules of other domains in each domain. As we primarily cover access control mechanisms using contextual information and employ simple context matching methods, context data acquisition and other context matching related

issues are left out the scope of this thesis.

In order to demonstrate the validity and practicability of proposed model we implemented prototype system that can be used to provide context aware access control service in multi-domain environments. We also explore a set of scenarios to show the applicability of the proposed model.

1.3 Thesis Organization

The outline of the thesis is as follows. In Chapter 2, firstly, the notion of ubiquitous computing, context aware computing, research fields in context aware computing field and definition of context aware access control model are presented. Then context aware access control models proposed in literature are surveyed. Finally, certificates, certificate based authentication techniques, certificate revocation techniques and digital signature technologies are discussed. Chapter 3 introduces the proposed access control model by detailing the main components. In Chapter 4, two different scenarios for the proposed model are defined and their evaluation with a sample prototype application is done to validate the applicability of the model. Then the implementation details of the sample prototype application are introduced in Chapter 5. Finally, in Chapter 6, summary of the proposed model and point to the possible future extensions are introduced.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter gives background information about the research domain and presents the related studies in the literature. Firstly the Ubiquitous Computing is introduced. Then, the Context Aware Computing and its sub research fields are discussed. It is followed by the studies related to the context aware access control models. Then, the proposed access control model is compared to the other models proposed in the literature. The chapter is finished with general background information on certificate technologies and certificate based authentication and access control mechanisms.

2.1 Ubiquitous Computing

Ubiquitous computing is the next generation computing model after the desktop model of computing. It states that computing is integrated with objects and activities around users. In ubiquitous computing, users perform their daily activities oblivious to the computers around them. In other words, computers move into background of our lives in ubiquitous computing [3].

According to Weiser and Brown [1], ubiquitous computing is the third phase of computing. First phase is the mainframes era in which many people share the same computer. Second phase is the personal computers era where each user has a computer. Third and last phase is the ubiquitous computing era where lots of computers share each of us. Now we are in a transition phase from second to third phase where internet and distributed computing take place. The major trends in computing across years are demonstrated in the Figure 2.1.

Some people confuse ubiquitous computing and virtual reality but ubiquitous computing is

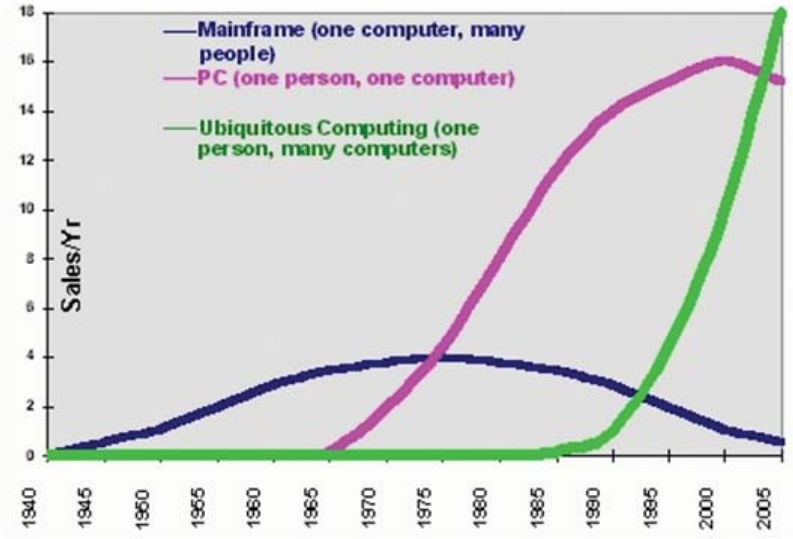

Figure 2.1: The Major Trends in Computing [3]

the opposite of virtual reality. *"Virtual reality puts people inside a computer-generated world; ubiquitous computing forces the computer to live out here in the world with people."* [3]

2.2 Context Aware Computing

Context aware computing is one of the major parts of ubiquitous computing. Abowd and Mynatt [4] categorized researches in ubiquitous computing into three parts. These are natural interfaces, context awareness and automated capture and access for live experiences. Natural interfaces researches are related with speech, hand writing, facial expression, etc. Automated capture and access researches are related with recording of daily life experiences in order to remove the humans' burden of recording events. Finally, context awareness researches are related with applications and systems that sense external environment and act according to the current context. Context information is one of the main inputs for most of the ubiquitous computing systems.

2.2.1 Context

Before defining context aware computing, it is better to define the term 'context'. There are so many different definitions of context. Lieberman and Selker define context to be any input other than the explicit input and output [5], where explicit input is users' intentional action such as key strokes and mouse clicks. Schilit and Theimer [6] define context as location and identities of nearby people and objects and also changes on those objects. Like the definition of Schilit and Theimer, most of the definitions of context depends on the context it is defined. One of the most general definitions for context is made by Day and Abowed: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."* [2] Thus, it can be said that information is a context if it can be used to describe an entity situation from any point of view.

2.2.2 Context Categories

Context can be categorized into several types. Categorization may be helpful for application developers to decide which pieces of context are suitable in their application. Categorization of Ryan et al. suggest location, environment, identity and time as context types [7]. Context types defined by Ryan et al, in practice, are the most important among the context types defined by others. On the other hand, Schilit et al. categorize context information according to where you are, who you are with and what resources are nearby [8]. Abowd and Mynatt suggest categorizing context information according to five 'W's which are: who (user is with), when (as time), where (as location), what (is user doing) and why (is user doing, i.e., intention) [4].

2.2.3 Context Awareness and Context Aware Computing

A system can be characterized as context aware if it can extract context information and then react based on that information. According to Fickas et al. context aware systems should monitor changes in the environment and adapt their behavior according to predefined or user-defined guidelines [9]. Dey and Abowd [2] give a more general definition for context aware

applications. They defined a system as context-aware if it uses context to provide relevant information or services to the user. Besides, context aware applications are categorized as active and passive by Chen and Kotz [10]. If an application changes its behavior according to context changes then it is an active context aware application. If application serves context information directly to the user without any action then it is a passive context aware application.

2.3 Research Topics for Context Aware Computing

Since context aware computing is a relatively new research field, there are lots of many places available for research and development. Some of the main research areas in context aware computing are described, by Kocaballı [11], as system architecture, context representation and ontology, sensors, context fusion, context matching and security and privacy based researches.

System architecture researches for context aware computing deal with modeling the architecture of context aware applications. Most of the researches done in this area are application specific and therefore there is no standardization in modeling context aware systems. Blackboard model [12] where client and server communicate by reading and writing to the same place and distributed services model [13] can be given as sample system architecture models for context aware applications.

Context representation and ontology researches deal with formatting and grouping context data and defining relationships among them in order to put them into standardization. By this way, contextual information is formed into a standard and become useful for applications [14].

Sensors related researches mainly deal with hardware related issues of context aware applications. They deal with sensing and retrieval of the context data in the physical environment. Some research subjects in this area can be described as usability, robustness, portability and reliability of sensors and sensed context data [15].

Context fusion is another research area as defined above. Context fusion provides the chance of extracting reliable context data from unreliable set of context sources by fusing data provided by them [16]. Besides, by context fusion, researchers can define some abstractions in

context by fusing lower level context data in order to provide high level ones.

Context matching researches are conducted to find better context matching mechanisms between provided and desired context data. Most of the time, context sources are not suitable for direct use. They have to be transformed into some well defined, mostly, application specific structures before using them.

Security and privacy are the last research subjects to be mentioned. Like desktop computing, security and privacy are very important subjects for context aware computing field. Besides, security and privacy issues have much more different aspects in context aware computing because of their distributed and un-intrusive nature [17]. One of the research subjects in that area is context aware access control models. Context aware access control deals with controlling user privileges and access requests according to the current context of the user. Like in all other research topics, most of the context aware access control models are application and/or domain specific.

2.4 Context Aware Access Control Models

In this thesis, we propose a certificate based context aware access control mechanism for multi-domain environments. However, before going on, it is better to examine some approaches on access control models in both context aware and certificate based environments.

2.4.1 Access Control Models

At first, mainly in desktop style computing, user access controls are achieved by storing sessions. In session based approach, each user starts with an authentication to the system. Then sessions are created only for some period of time. Users can access the resource for a predefined time interval. By defining user groups, providing different levels of permissions for different user types can be achieved. In spite of the fact that an access control mechanism can be provided in this way, it is a very limited approach for the cases with complex privilege requirements. Even it is not suitable to give different permissions to users without defining them in group context.

In 1996, Sandhu et al. presented a new approach for authentication and access control which

is called Role Based Access Control Model (RBAC) [18]. In RBAC model, permissions are associated with roles instead of users and users are associated with those roles. RBAC model simplifies the security administration and it provides the ability of defining different security policies. It is still insufficient for applying complex security policies since the roles defined in RBAC are static and defining flexible rules are not possible in role based access control models.

2.4.2 Context Aware Access Control Models

After the RBAC model, different types of extensions are proposed for RBAC model in many studies. One of the models that extend RBAC is called Temporal Role Based Access Control Model [19]. Bertino et al. extend RBAC with time dependency in Temporal RBAC. They provide role enabling and disabling capabilities according to defined time intervals. By using the time context, Temporal RBAC can be categorized as a context aware access control model. Similar to Temporal RBAC, Covington et al. [20] extend RBAC model with location and system status in addition to time domain. In both access control models, permissions for roles are configured according to the current context.

In Generalized Role Based Access Control Model (GRBAC) [21], Moyer and Abamad extended the notion of role. They defined roles not only for users but also for other system components. They introduced subject, object and environment roles in the access control decision phase. Subject roles are the traditional user related roles. Object Roles are roles that are related to the properties of objects to be accessed. Finally, environment roles are the context related roles that capture the users' current environment status.

In 2006, Zhang and Parashar presented the dynamic role based access control model (DRBAC) [22] in which they demonstrated a dynamic role enabling and disabling mechanism that works as a role state machine. When a user logs into a system, related role set and a local context agent is assigned for the user. This agent monitors the user context and dynamically adapts roles' and permissions' state according to the context information.

2.4.3 Models for Distributed and Multi Domain Environments

In distributed systems, in addition to authentication and authorization, some other issues like usage of different types of devices for authentication, management of user roles according to user trust levels need to be considered. In Cerberus [23], Muhtadi et al. proposed a distributed context aware access control schema which separates authentication module from the application in order to provide an abstraction to the devices used for authentication. They gave different confidence values to different devices and, as a result, they provided multi-level authentication to access resources in distributed context aware environment.

Hu and Weaver proposed a distributed, web service based context aware access control mechanism in 2004 [24]. They introduced trust level approach where users get different trust levels according to the device or technology they use during authentication. When user tries to access a request, not only user context but also given trust level is taken into account in access policy evaluation phase.

Another similar approach is published by Seon et al. They introduced COBAR in 2007 which is a distributed context aware access control model [25]. Similar to Hu and Weaver's work, they dealt with the usage of different authentication devices to access resources in context aware environments. They introduced the term 'Authentication Confidence Index' (ACI) which is assigned to system users according to the confidence value of the device they use for authentication. If a user's ACI value is higher than the role activation value, related role is enabled for that user. Otherwise, the user cannot access some resources until he/she uses a more confident device to authenticate.

The approaches summarized above provide dynamic, context aware security models but they are proposed for single domain environments. In multi-domain environments issues like federation of security information of users to other domains or uninterrupted service access in different domains should also be taken into account.

In 2005, Nishiki and Tanaka [26] proposed a context aware access control model for multi-domain environments. They introduced authentication and access control agents which are assigned for each domain that the system is implemented. When logging into a domain, an authentication ticket is given to a user. At the same time an access ticket that holds evaluated user roles is generated for the user and stored in the server. When user tries to access

a resource, his/her context information and stored access tickets are used to evaluate access request decision. When user moves into another domain, he/she provides the given authentication ticket to the domain access control agent and then access control agent retrieves the user access ticket from the domain that really authenticated the user. Given model is mainly implemented for adjacent domains where user can pass from one to another before the session is expired. Access control agents find each other by sending search messages to the nearest domain agents which slows down the search process.

2.4.4 Certificate Based Authentication and Access Control Models

In this thesis, a context aware access control model is introduced for multi-domain environments. Problems like federation of security information among domains and fast and uninterrupted resource access of users are tried to be resolved by proposing a certificate based solution. In this section, some certificate based authentication and access control models are investigated and some background information about certificates and certificate based authentication and authorization techniques will be given.

Koufi and Vassilacopoulos [27] suggested a model in 2008 which defines a certificate based context aware access control model. In the proposed model, user roles are stored in certificates. When a user wants to access a resource, the user's context information is evaluated with those roles to decide whether the user is authorized to use the resource or not. Proposed model defines a certificate based access control mechanism but it is only concentrated on single domain systems. Therefore there is no infrastructure suggested for supporting multi-domain systems.

Another certificate based technique was proposed by Wang et al. [28], in 2007. They introduced a multi-domain certificate based authentication system. By using certificates provided, users can authenticate one or more domains according to their environment. Given solution provides a suitable certificate based authentication mechanism for multi-domain environments but it only focuses on authentication to domains. Access control decisions are not done using certificates.

Thompson et al. offered a certificate based access control mechanism for distributed resources [29]. They proposed a certificate based, multi domain access control model. Users authenti-

cate to systems and access to resources by using their certificates. Certificates store user roles and access request decisions are evaluated by using those roles. However, this solution is not designed for context aware systems.

2.5 Certificates and Certificate Based Authentication

In our proposed model, similar to the certificate based solutions above, certificates are used to access resource. Therefore a certificate validation mechanism is required in access request evaluation phase. In this part of the chapter, information on certificates and certificate based authentication will be presented. First of all, security problems and main threat types in networks will be described. Then the public key cryptography is described and encryption and decryption techniques are summarized. Finally, certificate technologies are described.

There are three main security issues in communication over a network. These are eavesdropping, tampering and impersonation [30]:

Eavesdropping: Eavesdropping threat type is related with reaching private information without authorization. Information does not change with eavesdropping but its privacy is broken. For instance, someone may learn your credit card number or intercept confidential information.

Tampering: Without authorization of the owner, changing information is called tampering. When information is sent, someone could change it before it is delivered to the recipient. Changing someone's vote in an election or changing the order of a company without their knowledge can be the examples for tampering.

Impersonation: Impersonation, as a threat type, is imitating potential recipient in order to access some information or to do something. Impersonation has two types. First one is spoofing in which a person acts as if he/she were someone else. For example, sending an email to someone by using the mail address of a company is a spoofing type of impersonation. Second type is misrepresentation where a person represents himself/herself as if he/she were someone else. Publishing a fake online store with the aim of collecting users' credit card numbers is a misrepresentation type of impersonation.

2.5.1 Public Key Cryptography

Public key cryptography is a set of techniques and standards that make it easy to take measures to communication threats listed above by using a cryptographic approach [31]. It uses asymmetric key algorithms. Public key cryptography may help to achieve following tasks:

Tamper Detection: Allows the recipient to verify whether data sent is changed or not during communication. Any change in the original message can be detected.

Authentication: Allow recipient to detect whether data is sent by the sender who is claimed. That is, it approves the identity of the sender.

Non-repudiation: Prevents the information sender denying the information is sent.

Encryption and decryption: Includes encoding operation of data sent and decoding operation of received data. Encryption and decryption solve the problem of eavesdropping. Encryption is a process to make information unintelligible and decryption is vice versa. Encryption and decryption is done, most of the time, by using a cryptographic algorithm [31] which is a mathematical function pair to encrypt and decrypt information. Usually cryptographic algorithm, itself, is not secret but some sort of secret keys is used by cryptographic algorithms. There are mainly two approaches of key based encryption schemes available. These are called symmetric and asymmetric key encryption [32].

2.5.1.1 Symmetric Key Encryption

In symmetric key encryption algorithms, the same key is used for both encryption and decryption of the data. Symmetric key encryption provides a degree of authentication because the data encrypted with a key cannot be decrypted with a different key. As a result, communicating parties can transfer data securely as long as they keep the key secure. If some third parties get the key, they can not only decrypt sent data but can also send different data after encrypting it with the key as if they are one of the parties communicating. Symmetric key encryption is illustrated in Figure 2.2.

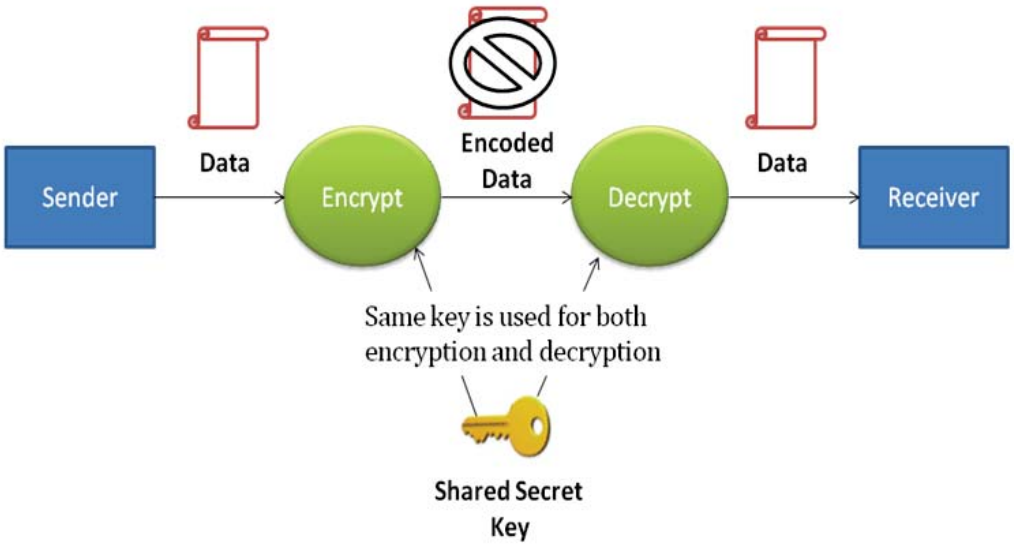

Figure 2.2: Symmetric Key Encryption

2.5.1.2 Asymmetric Key Encryption

In asymmetric key encryption algorithms, two different but somehow related keys are used for encryption and decryption of the data. Data encrypted with one of the keys can only be decrypted with the other key. In that approach public key is published to receiver and private key is kept secret. Receiver decrypts data with the public key of sender and if someone wants to send secure data to the key owner he/she should encrypt the data with the public key of the recipient. This time encrypted data can only be decrypted by using private key. However, public key encryption is much more complex compared to the symmetric key encryption techniques. Therefore, most of the times, it is not practical to encrypt, decrypt large amount of data by using public key encryption technique. In such cases, symmetric key algorithms are used with session keys exchanged by using public key encryption techniques. Besides encryption with a private key is not so much secure since anyone that has public key can read data to be send. Nevertheless, public key encryption is really useful to prove the identity of the key owner. Therefore it is one of the important requirements of today's electronic commerce applications. Asymmetric key encryption is illustrated in Figure 2.3.

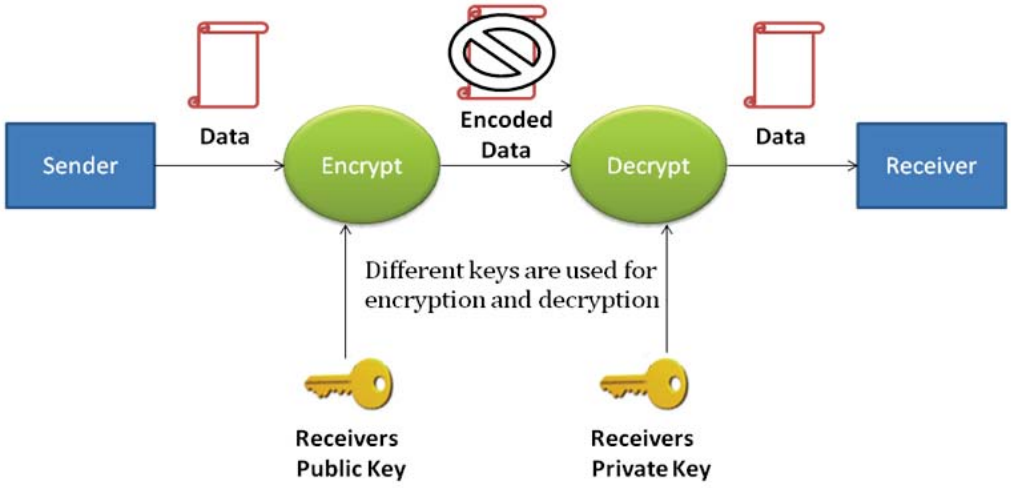

Figure 2.3: Asymmetric Key Encryption

2.5.2 Digital Signature

Digital signature is used to prevent third parties from modifying the data sent to recipient without detection. It uses a method called one way hash (also called digest method) [33]. Key owner applies a hash function to the data and then encrypt the hash with his/her own private key. Finally he/she attaches encrypted hash together with the original data and sends that combination (called digitally signed data) to the recipient. When recipient gets digitally signed data, he/she uses public key of the sender to decode the encrypted hash. After decoding, recipient applies the same hash function to the received data and compares the result with the decoded hash. If the two hashes match, the receiver will be sure that the data has not changed since it was signed. If they don't match, the data must have been tampered with since it was signed. Signing and verification phases for digital signature is demonstrated in Figure 2.4.

2.5.3 Certificate

A certificate is a digital document that identifies an entity like an individual, a server or a company to associate that entity with a public key. Similar to driver's licenses, certificates prove the identity of an individual. Therefore, public key cryptography uses certificates to

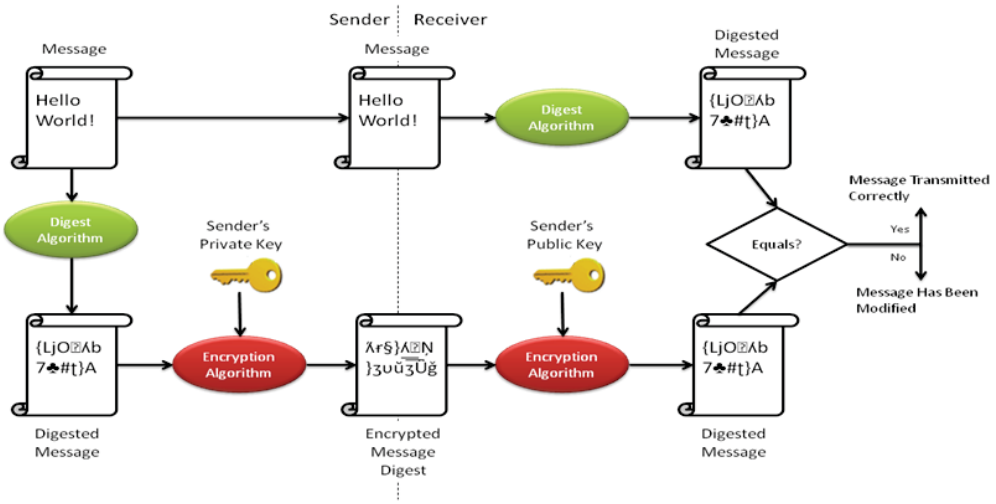

Figure 2.4: Signing with Digital Signature and Verification

protect communication from impersonation.

Similar to other real world identity cards (like driver license); certificates are managed by authorities called Certificate Authorities (CA). CA's validate identities and publish certificates. These authorities are independent third parties or organizations. Each CA has its own certificate validation technique.

A certificate is composed of two parts. First part is the data section. It includes a public key and the cryptographic algorithm, name of the entity identified by certificate, an expiration date, the name of the CA publishing the certificate, a serial number, and other optional information. Second part is the signature section. It includes the digital signature of the CA which is obtained by encrypting the whole certificate data with CA's private key. It also indicates the algorithm used by CA to sign the certificate. Sample certificate content is given in Figure 2.5.

2.5.3.1 Certificate Based Authentication

Digital signatures can be used for authentication. With digital signatures, client authentication and server authentication can be done in various different ways. For instance, adding digital signature to an email or an html page and attaching the certificate to the message/page can provide a strong evidence of sender's or publisher's agreement. In addition to authentication,

```
Certificate:
    Data:
        Version: v3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: PKCS #1 MD5 With RSA Encryption
        Issuer: OU=Ace Certificate Authority, O=Ace Industry, C=US
        Validity:
            Not Before: Fri Oct 17 18:36:25 1997
            Not  After: Sun Oct 17 18:36:25 1999
        Subject: CN=Jane Doe, OU=Finance, O=Ace Industry, C=US
        Subject Public Key Info:
            Algorithm: PKCS #1 RSA Encryption
            Public Key:
                Modulus:
                    00:ca:fa:79:98:8f:19:f8:d7:de:e4:49:80:48:e6:2a:2a:86:
                    ed:27:40:4d:86:b3:05:c0:01:bb:50:15:c9:de:dc:85:19:22:
                    43:7d:45:6d:71:4e:17:3d:f0:36:4b:5b:7f:a8:51:a3:a1:00:
                    98:ce:7f:47:50:2c:93:36:7c:01:6e:cb:89:06:41:72:b5:e9:
                    73:49:38:76:ef:b6:8f:ac:49:bb:63:0f:9b:ff:16:2a:e3:0e:
                    9d:3b:af:ce:9a:3e:48:65:de:96:61:d5:0a:11:2a:a2:80:b0:
                    7d:d8:99:cb:0c:99:34:c9:ab:25:06:a8:31:ad:8c:4b:aa:54:
                    91:f4:15
                Public Exponent: 65537 (0x10001)
        Extensions:
            Identifier: Certificate Type
                Critical: no
                Certified Usage:
                    SSL Client
            Identifier: Authority Key Identifier
                Critical: no
                Key Identifier:
                    f2:f2:06:59:90:18:47:51:f5:89:33:5a:31:7a:e6:5c:fb:36:
                    26:c9
    Signature:
        Algorithm: PKCS #1 MD5 With RSA Encryption
        Signature:
            6d:23:af:f3:d3:b6:7a:df:90:df:cd:7e:18:6c:01:69:8e:54:65:fc:06:
            30:43:34:d1:63:1f:06:7d:c3:40:a8:2a:82:c1:a4:83:2a:fb:2e:8f:fb:
            f0:6d:ff:75:a3:78:f7:52:47:46:62:97:1d:d9:c6:11:0a:02:a2:e0:cc:
            2a:75:6c:8b:b6:9b:87:00:7d:7c:84:76:79:ba:f8:b4:d2:62:58:c3:c5:
            b6:c1:43:ac:63:44:42:fd:af:c8:0f:2f:38:85:6d:d6:59:e8:41:42:a5:
            4a:e5:26:38:ff:32:78:a1:38:f1:ed:dc:0d:31:d1:b0:6d:67:e9:46:a8:
             dd:c4
```

Figure 2.5: Certificate Content [30]

digital signature attached to any data also provides non-repudiation.

Client authentication is the essential element of network security. There are two types of authentication in client based authentication: password based authentication and certificate based authentication. Password based approach is out of the scope of this thesis. In certificate based authentication, server generates a random piece of data and sends the data to the client. Then client digitally signs retrieved random data and sends both its certificate and the signed data across the network. The server uses techniques of public-key cryptography to validate

the signature and confirm the validity of the certificate. Certificate based client authentication is part of the SSL (Secure Sockets Layer) protocol.

2.5.3.2 Revoking Certificates

Certificates are issued for some period of time. They contain the data that defines validity start and end dates. However, in some circumstances, certificates may require revocation before the expiry date. In such cases, CA should inform entities that trust those certificates published by it.

Certificate revocation can be handled in several different ways. For some organizations, authentication process includes checking the certificate validity directly from the servers of the CA. In that approach, CA setups some server and publishes all valid certificates in directories. When an administrator revokes a certificate, the certificate is removed from the directory, and authentication attempts with that certificate will fail even though the certificate remains valid from other aspects. Another approach involves publishing a certificate revocation list (CRL). CRL is the list of revoked certificate. In that approach, CRL is published to a directory at regular intervals and authentication process includes checking the CRL list.

CHAPTER 3

PROPOSED MODEL

In this chapter, a model to provide a context aware access control for multi-domain environments is introduced. In order to evaluate resource access requests of users in a context aware environment, a domain should collect user roles, privileges and other user related information from the home domain of the user. To access user specific information, most of the available models deal with communication with the home domain in access request decision phase. Since ubiquitous computing applications require quick response time by their nature, such communication latencies may cause some usability problems.

The proposed model focuses on reducing network traffic between different domains in access request decision phase. That is, it works in a multi-domain environment but access control decisions are done as if the environment is a single domain. By this way, it gains the advantage of minimizing complexity and speeds up access control decisions in multi-domain environments. The model is based on certificates to achieve the required goals. In the proposed model, instead of searching user validity from remote sources in access request decision phase, a certificate is given to each user and user validity check is done locally by controlling those certificates. Besides, user requests are evaluated against local policy rules which includes not only policy rules for host domain users but also policy rules for other domains that has an inter-domain service level agreement with host domain. Proposed model is illustrated in Figure 3.1. The primary focus of this study is on certificate based access control in multi domain environments. Therefore some of the related issues like context data retrieval and some of the issues in context aware computing beyond basic context matching are out of the scope of the thesis.

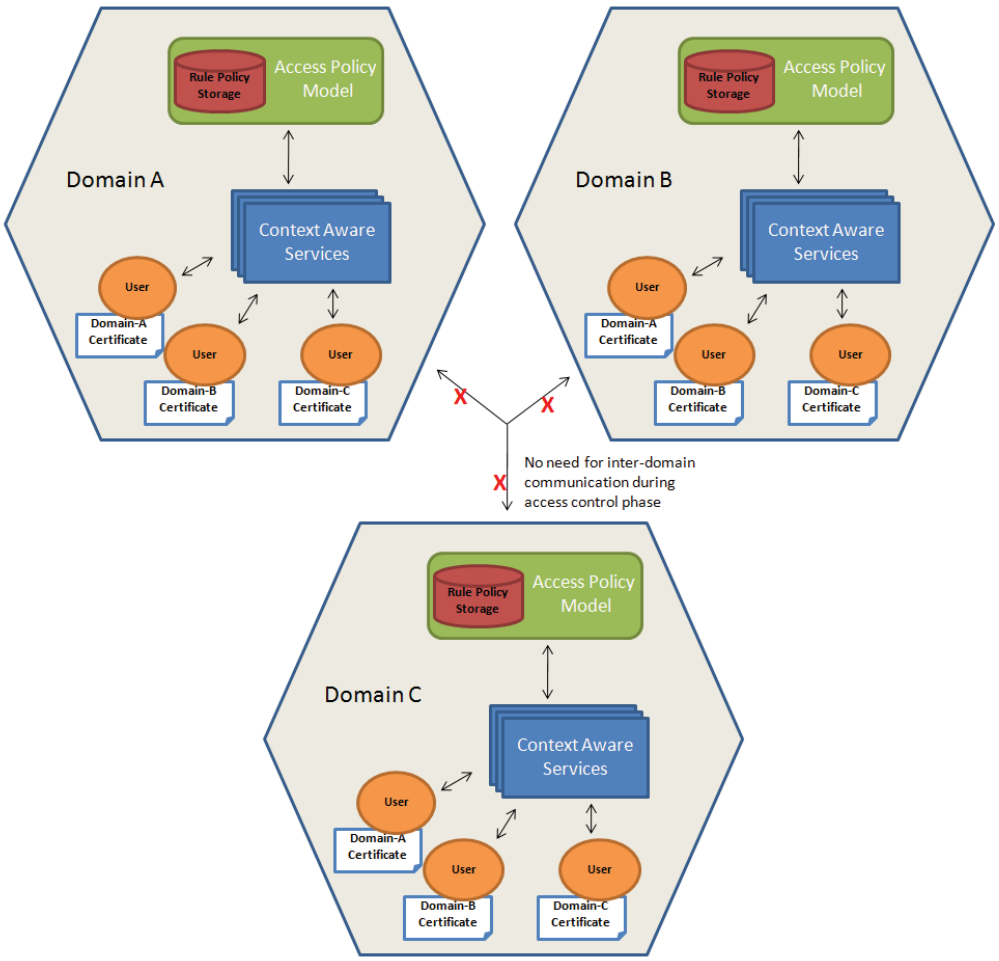

Figure 3.1: Context Aware Access Control Model

3.1 Salient features of the Proposed Model

First of all, proposed model is for context aware systems. That is, each access request of a user is evaluated according to the current context of the user. Secondly, proposed model is certificate based. Users ask for accessing resources with their certificates issued by their own domains. Finally, proposed model is used in multi-domain environments. It can be used in only one domain but it is mainly modeled for multiple domains that are interrelated with each other and have their own domain users accessing resources in other domains too.

In the proposed model, a certificate is given to each user by their own domains. Users try to access resources in their domains by using these certificates. If their context data is suitable to

reach the resource, then they are allowed to access the resource. On the other hand, a domain can make an inter-domain service level agreement with other domains so that users of that domain can reach the resources of other domains. In such situations, domains define extra access control rules for other domains. As a result, when other domain users try to access resources in the agreed domain by using their certificates, they can access the resources if their context data is suitable according to access control rules defined by the domains. These cases for access control check in the proposed model are illustrated in Figure 3.2.

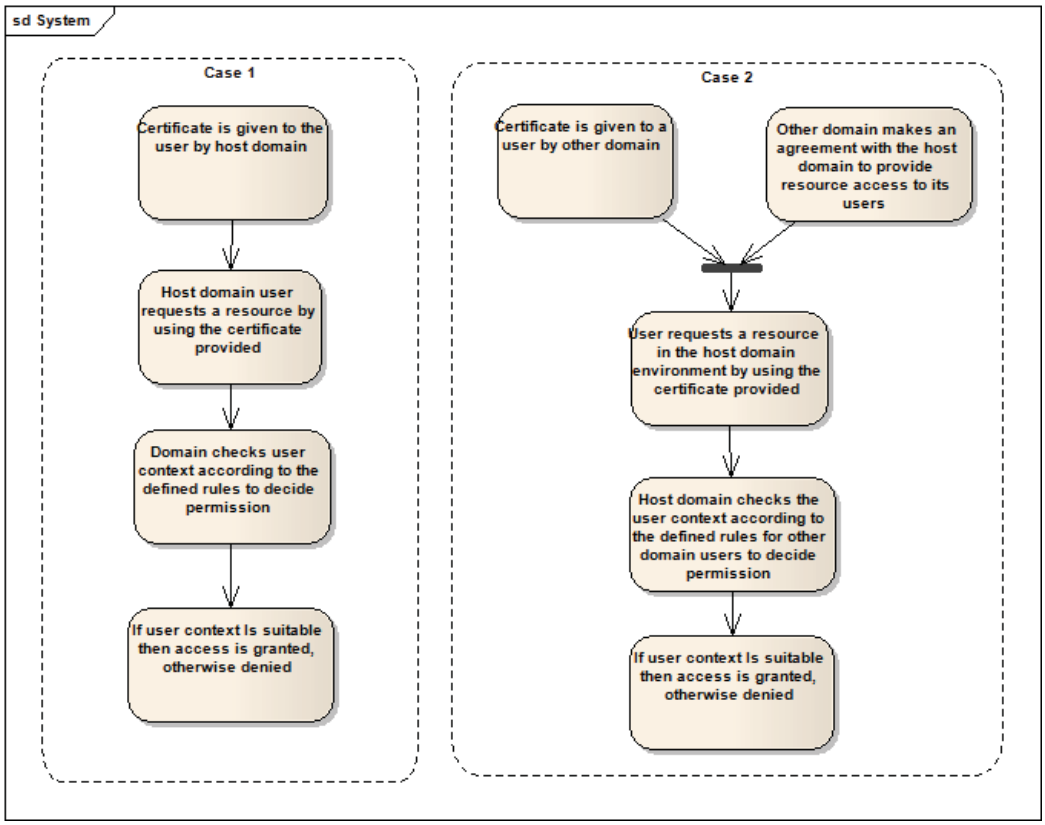

Figure 3.2: Cases for Accessing Resource for Different Domain Users

In a domain, the following type of rules can be defined to cover different cases:

- A domain can define access control rules for its own users. A user can access resources with his/her certificate if the user's context matches with the rules defined in the domain.

- A domain can define access control rules for users of another domain. Users of other domain can access resources with the certificates provided by their own domain if their context data fits to the rules defined for their domains.

- A domain can define access control rules for specific users of another domain. Such users can access resources with their own certificates supplied by their domain if those users' context data is suitable according to those rules defined specifically for them.

- A domain can define groups for both its own users and other domain users and then it can define access control rules for those groups. Users in those groups can access resources with their own certificates supplied by their own domain if their context data fits to those group rules.

In order to cover all these different cases, a domain abstraction is done in the proposed model. All domains including the host domain are treated as external domains. As a result, in a domain, a rule can be defined for a specific user, for a domain (including itself) or for a group.

3.2 Advantages of Using Certificates

If proposed model were implemented without a tool like certificate, then systems would check the identity of users for each access control by communicating with the home domain of the user. Or, at least, each domain should have provided an authentication mechanism before the access decision for requests are made in order to validate user from the home domain. Such a validation mechanism would require communication between the servers of the domains and this would consume tangible amount of time and network bandwidth.

By using certificates, users carry their own authentication credentials with them so that communication between domains to validate users can be minimized. This certificate based approach provides a faster access control compared to the approach based on authentication via the home domain. Moreover, by using certificates, users not only carry their own identity but also carry their domain's identity with them. As a result, inter-domain service level agreement rules to access the resources can be defined in domain identity level and access control evaluations can be done based on domain identity when a user tries to access a resource with his/her certificate. That is; instead of storing users' identity in rules, storing certificate provider's identity in rules is enough in order to evaluate individual user requests.

3.3 Major Components of the Proposed Model

Proposed model consists of five components working in the application layer and one component working in the service layer. Application layer components are policy rule engine, context data service, context rule service, domain communication agent and policy management interface. Service Layer has only one component which is the persistence manager. The component diagram of the proposed model is given in the Figure 3.3.

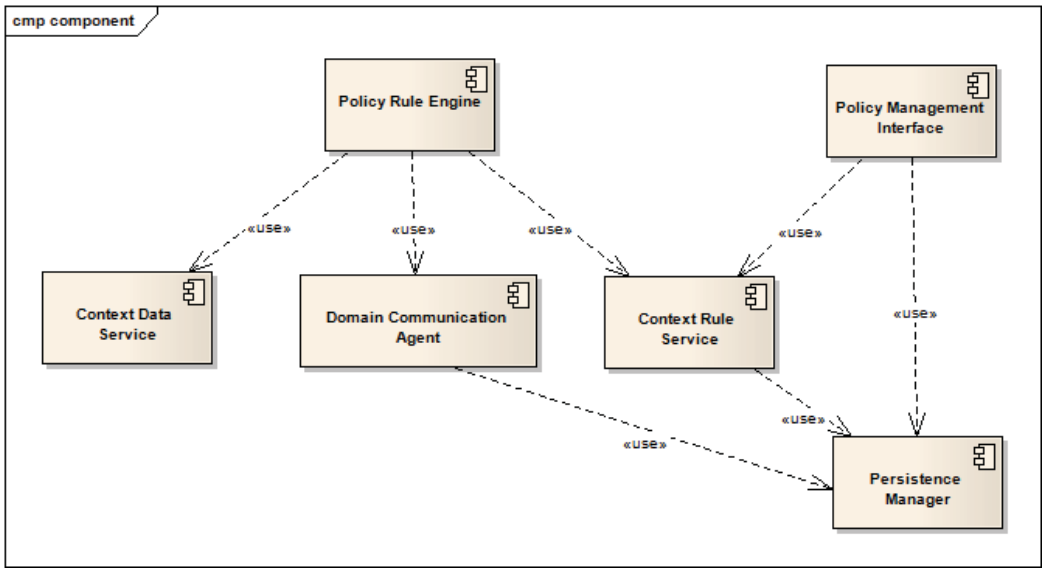

Figure 3.3: Model Component Diagram

The policy rule engine is the central component of the model. Requests are retrieved by the policy rule engine and it coordinates the tasks to be done and manages other components of the system. The policy rule engine retrieves context data from the context data service, access policy rules from the context rule service and certificate validity information from the domain communication agent in order to process the incoming user requests. After the evaluation, it returns the response for the access request back to the user and also grants access to the resource requested.

The context data service is responsible for context data retrieval and context matching for users. It determines context data for the context types requested by the policy rule engine and returns those data back to it. Context matching and context data retrieval is a huge research area. There are so many approaches/methods available for these. Therefore, context retrieval

and matching issues are out of the scope of this thesis. It is assumed that context data service does the context data retrieval and context matching tasks for the given user in defined contexts when requested. It is left as a separate module in order to make it pluggable, i.e., different implementations can be employed for context retrieval and matching in context data service easily.

Each domain using the proposed model should contain some rules that define access policies for the users in different contexts. Context rule service manages those access policy rules and provides rules when they are requested. Whenever access policy rules are requested for a given user and a given resource, it immediately finds the related rules and sends them back to requester possibly after filtering and processing those rules. Besides, context rule service is responsible for validating access policy rules that are inserted or updated by the system administrators over Policy Management Interface.

Domain communication agent stores the list of revoked certificates in order not to allow those users who has a certificate with valid data but revoked by certificate provider for some reasons. When a user certificate is tried to be checked by domain communication agent, it controls certificate from the certificate revocation lists retrieved from certificate providers. Domain communication agent is also responsible for updating those lists from certificate providers' domains.

Policy management interface provides a graphical user interface for the administration tasks of the system. System administrators can insert or update access policy rules into the system via the policy management interface. Also some system parameters can be defined by using policy management interface.

Last module is the persistence manager. The persistence manager provides an interface to other modules in order to ensure that they reach database. All database related operations are done by the persistence manager. There are lots of different implementations available for persistence manager and one of them could be adapted to use in this model.

Each component in proposed model is designed as a pluggable separate module and, for better modularity, system components interact via service oriented architecture. As a result, each module defines its methods as web services.

3.4 General Work Flow for the Proposed Model

When a user tries to use a resource by providing his/her certificate, user certificate data and resource data are retrieved by the policy rule engine. Then policy rule engine checks the certificate validity by sending certificate data to the domain communication agent. The domain communication agent first checks certificate validity and then checks whether certificate is revoked or not from certificate revocation lists and finally sends result back to the policy rule engine. If certificate is valid then the policy rule engine asks access policy rules from the context rule service. The context rule service searches given certificate and resource data from database over the persistence manager. Found rules are filtered and processed in order to get rid of problems such as rule conflicts. Filtered and processed rules are sent back to the policy rule engine. Next, policy rule engine determines context types used in the access policy rules and asks context data in for those context types for the user from the context data service. Context data service retrieves context data for the specified user and requested context types. After retrieval of context data, it sends back those data to policy rule engine. Finally, policy rule engine evaluates retrieved access policy rules with fetched context data and sends decided access request decision back to the requested resource. Activity diagram for general work flow of the proposed system is given in the Figure 3.4.

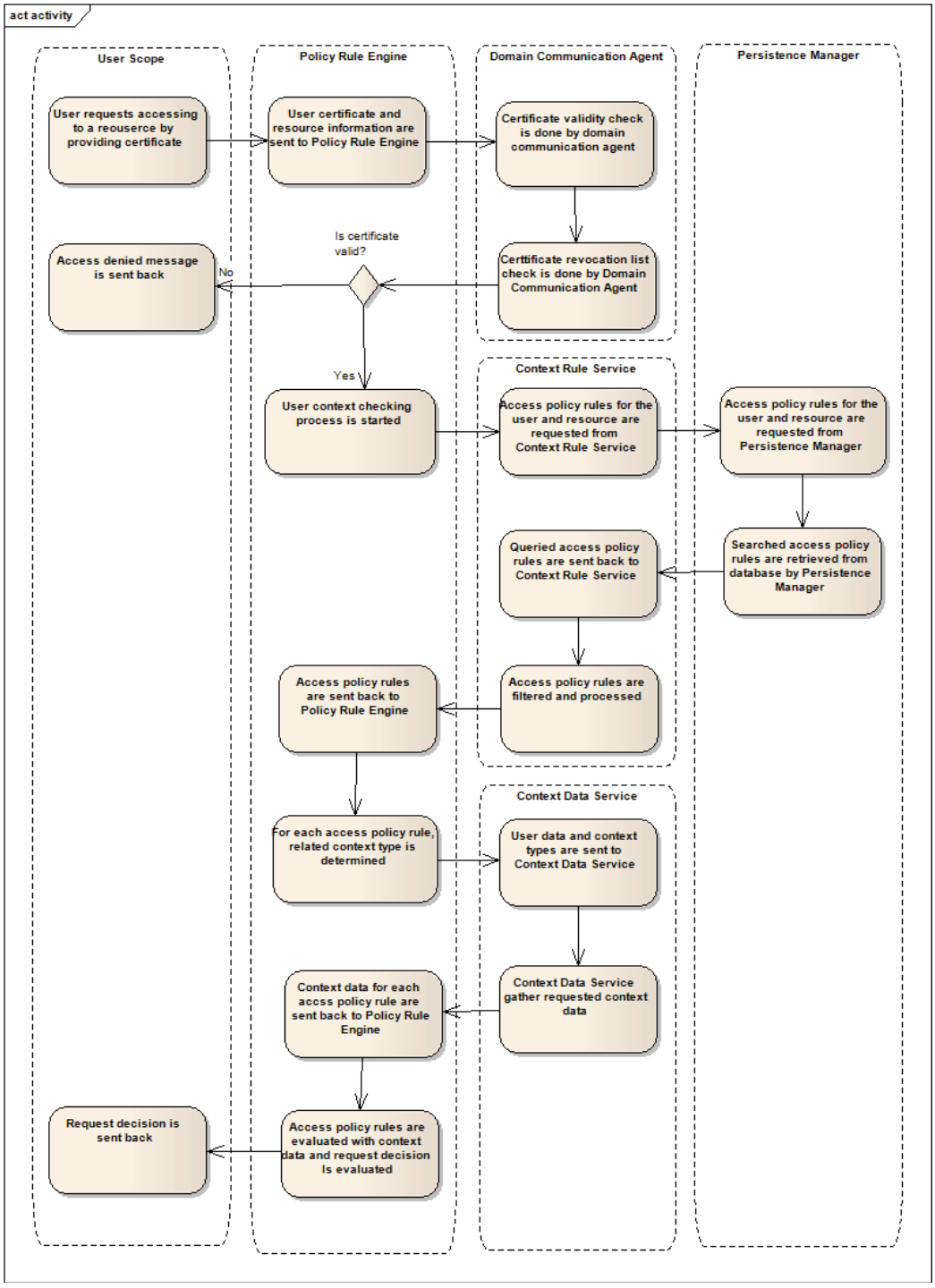

Figure 3.4: Activity Diagram for Proposed System

In the following sections Context Rule Service, Domain Communication Agent, Policy Rule Engine and Policy Management Interface components are explained in detail.

3.5 Context Rule Service

Each domain should store some access rules to decide whether the usage of requested resource is allowed or denied in the current context of the requester. The Context Rule Service acts as the main policy rule storage for domains. The Context Rule Service contains pre-defined rules and provides those rules when they required for access control decision. These policy rules are called Access Policy Rules.

Two main operations are handled by the Context Rule Service. Firstly, it validates Access Policy Rules before they are inserted or updated via Policy Management Interface. Secondly, when the Policy Rule Engine requests some Access Policy Rules, the Context Rule Service retrieves those rules from the data storage by using the Persistence Manager and sends requested Access Policy Rules to the Policy Rule Engine after processing them. The Context Rule Service component diagram is given in the Figure 3.5.

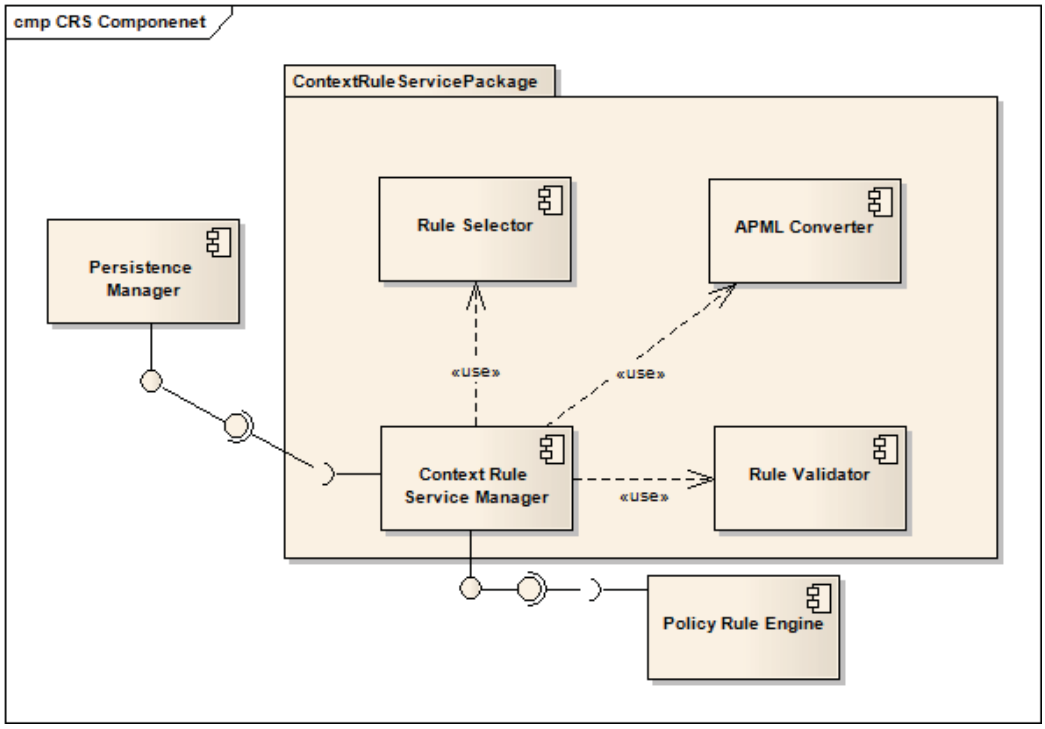

Figure 3.5: Context Rule Service Component Diagram

Access Policy Rules are defined and stored in Access Policy Markup Language (APML) format which is defined as an extension to the eXtensible Markup Language (XML) format

for Access Policy Rules. Access Policy Rules are defined in an XML based structure because it can easily be transferred among different services in a transparent way and also it is easy to manipulate and read.

When a new or an updated context rule is sent to the Context Rule Service in APML Format, the Context Rule Service Manager extracts rule data by sending incoming context rule to the APML Converter. After conversion, it validates the rule data through the Rule Validator and finally it sends the validated rule data to the Persistence Manager in order to persist it to the database.

In the case of rule request condition, certificate id, certificate provider id and resource id are received by Context Rule Service and they are sent to the Persistence Manager to find the related rules. The rules found are sent to the Rule Selector to decide unsuitable rules that should be filtered in the case of a conflict. Then selected rules are sent to APML Converter to convert them into APML format. And finally selected rules are sent to the requester in APML format. The more detailed sequence of an incoming request will be discussed shortly. Structure of the Access Policy Markup Language and some Access Policy Rule examples are given in Figure 3.6.

```
<apr>
  <subject type="user_certificate">1568797</subject>
  <resource type="resource">printer48</resource>
  <context type="time">Working Hours</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="certificate_provider">1658</subject>
  <resource type="group">cs_department_printers</resource>
  <context type="location">cs_department</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="group">1214</subject>
  <resource type="group">metu_printers</resource>
  <context type="time">weekend</context>
  <permission>allow</permission>
</apr>
```

```
<apr>
  <subject type="group">764</subject>
  <resource type="resource">shopping_mall_cinema_discount%50</resource>
  <context type="time">wednesday</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="certificate_provider">45678</subject>
  <resource type="group">shopping_mall_floor1_stores_discount%25</resource>
  <context type="location">shopping_mall</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="user_certificate">77769548</subject>
  <resource type="group">shopping_mall_floor1_stores_discount%25</resource>
  <context type="time">working_days</context>
  <permission>deny</permission>
</apr>
```

Figure 3.6: Structure of APML and some APR samples in APML format

3.5.1 Structure of an Access Policy Rule

Each Access Policy Rule contains four different data types. These data types are subject, resource, context and permission.

Subject is the identity of the requester. It can be a certificate id of a user or it can be the

identity of the certificate provider or even it can be the identity of a certificate group that is pre-defined by administrators. A student with a certificate, or a university as a certificate provider, or instructors of CS department that is defined as group can be given as different subjects in a campus domain. Similarly, a customer of a telecommunication company with certificate, the telecommunication company itself as certificate provider, or a customer group that benefit from some special service of the telecommunication company can be given as different subjects in a telecommunication company domain. A subject can be a certificate id, certificate provider id or a group. Groups contain a list of subject; i.e., groups can contain a list of id or can contain other sub groups. Structure of the subject is given in the Figure 3.7.

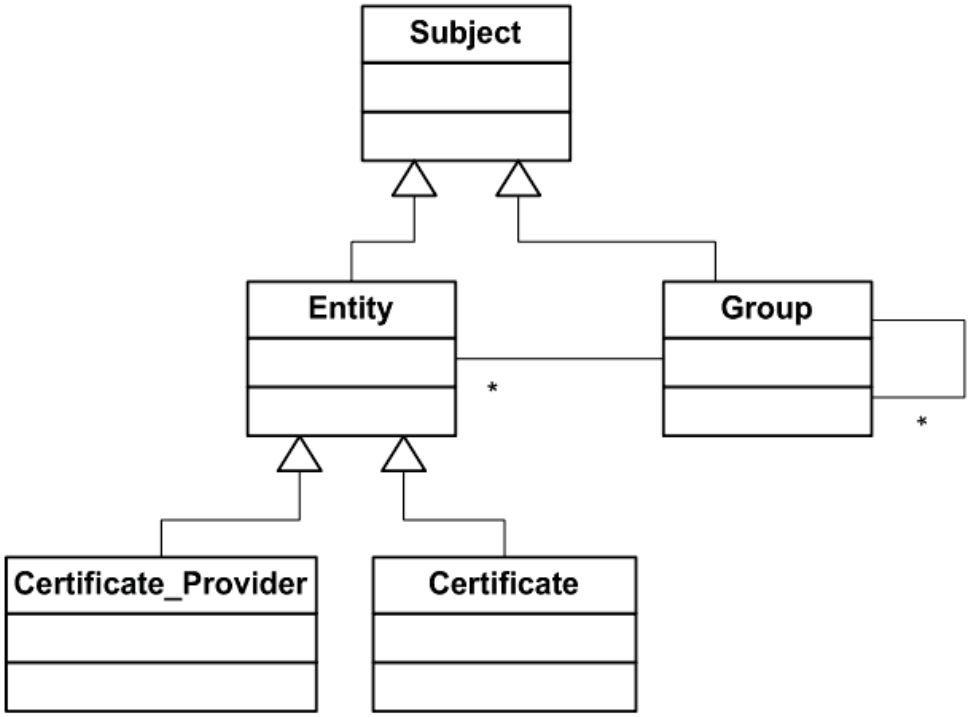

Figure 3.7: Structure of the Subject in APR

A resource is the object that is requested to be used/accessed by the subjects. Similar to a subject, a resource can be a single resource or can be a resource group that is defined by administrators. In a campus domain, all printers in the campus, all printers in a specific department or a single printer in a department can be defined as a resource. Similarly; a promotion in a market, a promotion in a floor of a shopping mall or a promotion in all stores of a shopping mall for telecommunication company costumers can be defined as resources in

the telecommunication company domain. A resource can be a single resource or a group of resources. A group can include a list of sub groups as well as resources. The structure of the resource is given in Figure 3.8.

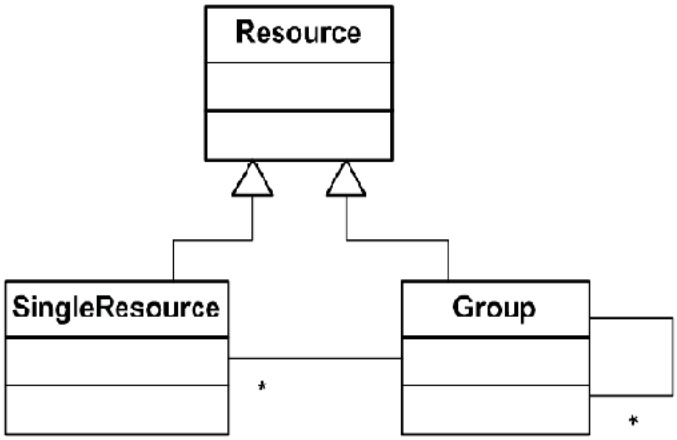

Figure 3.8: Structure of the Resource in APR

A context is the current context information of the subject. A context can be any contextual information related to the subject. Each context element contains two different data. These are context type and context name. Working hours in the time context and CS Department as the location context can be given as samples to context names. Context data of the related context names are stored in another table in the persistent storage. Context names and context data can be defined by administrators.

Permission defines whether Access Policy Rule is related with giving access right to the requester or denying requester from accessing a resource. There are two kinds of permission data available. These are 'Allow' and 'Deny'. While there may a more general rule that allows user to access a resource, another specific rule can also be defined to deny a request for some specific sub-group, sub-resource or sub-context. The rule selection algorithm is called 'More Specific Rule Wins' as it is explained in the following sections.

3.5.2 Policy Rule Storage

Access Policy Rules are stored in a relational database. Each rule is stored as a single entity in the database. Besides access policy rules, subject group definitions, resource group definitions and also context types and context definitions are stored in the database.

Policy Rule table contains the Access Policy Rules. Each table entity contains a rule id, subject id, resource id, context name, and permission data. Rule id is a unique number that defines a rule uniquely. Subject id is the identity of a user or user group. Subject id can be certificate id of the user, certificate provider id of the user or id of a group. Resource id is the id of a resource which can be a single resource entity or a group of resources. Context name is the name of the context checking data which can also be empty if there is no context rule for accessing a resource. Context names are mapped to context checking data in the Context table defined below. Finally, permission field stores whether the rule is an access rule or deny rule.

Subject group definitions are stored in a subject group table. It contains group id, subject id and subject type. Group id is the unique number that defines a group. Subject id is either a certificate id, certificate provider id or an id of another sub group. Subject type stores the type of the subject. If subject is certificate id or certificate provider id then subject type is entity. If subject is id of another group then subject type is group.

Resource definitions are stored in a resource table. Resource table contains resource id, resource name and other necessary resource data. Resource group definitions are stored in the resource group table. Similar to subject groups, resource groups are composed of group id, resource id and resource type.

Context table contains name of the context, type of the context, context checking condition which can be a r(ange) or an e(quality) check, context checking data and context data formatter. <WorkingHour, 'time', 'r', '09:00,18:00-hh:mm'> can be a sample context table entity. Context types can be defined in the implementation phase. They cannot be defined in application runtime because there should be some context evaluation algorithms implemented for each context type. If, later, it is decided to add a new context type then the related context evaluation algorithms should be implemented for that context type. Context evaluation techniques for some selected context types will be defined in the scope of the policy rule engine

module.

3.5.3 Rule Selection Technique

When an Access Policy Rule is requested from it, the Context Rule Service works as follows:

1.Access Policy Rules of a resource with resource_id and a subject with certificate_id and certificate_provider_id are asked from the Context Rule Service.

2.The ContextRuleServiceManager sends retrieved resource_id to the PersistenceManager.

3.Resource groups are searched by the PersistenceManager in order to find resource_group_ids that contain the given resource_id.

4.The ContextRuleServiceManager sends retrieved certificate_id and certificate_provider_id to the PersistenceManager.

5.Subject groups are searched by the PersistenceManager in order to find subject_group_ids that contain the given certificate_id or certificate_provider_id.

6.Access Policy Rules are searched with the resource-subject pairs to find the related rules. Resource_id and found resource_group_ids are used as the resource and certificate_id, certificate_provider_id and found subject_group_ids are used as the subject in query.

7.Found rules are categorized according to the context types of the rules and each group is filtered with 'More Specific Rule Wins' algorithm in order to eliminate the rule conflict situations. This algorithm will be presented in the following section.

8.Context names in selected rules such as 'Working Hours' for time context and 'CS Department' for location context are converted to the real context data by searching context names from the Context table in the database.

9.Selected Rules are converted to APML.

As a result, selected Access Policy Rules are served to the requester in the format of APML.

3.5.4 Rule Conflict Situation and 'More Specific Rule Wins' Algorithm

Since there is no limitation in defining a new access policy rule for a user or a user group, there may be some rule conflicts. For example, while a rule is in use for allowing the usage of a resource for a specific user, there may exist another rule with the same context condition that deny the usage of same resource for him/her. For such conflicting situations, an algorithm to resolve the conflict is required. Context rule service applies 'More Specific Rule Wins' algorithm. According to the algorithm, when a conflict for a subject is caused by more than one rule for the same resource with the same context condition, the rule with the more specific subject definition is selected. When the subject levels are the same, the rule with the more specific resource definition is selected. If both subject and resource levels are different then only the subject definitions are examined for the rule selection. The rule selection decision for different rule types is given in Table 3.1. Note that the algorithm is applied when all rules under examination have the same context condition.

Table 3.1: APR Selection Table

	U1 - R1	Un - R1	U* - R1	U1 - Rn	Un - Rn	U* - Rn	U1 - R*	Un - R*	U* - R*
U1 - R1	X	<--	<--	<--	<--	<--	<--	<--	<--
Un - R1	↑	X	<--	↑	<--	<--	↑	<--	<--
U* - R1	↑	↑	X	↑	↑	<--	↑	↑	<--
U1 - Rn	↑	<--	<--	X	<--	<--	<--	<--	<--
Un - Rn	↑	↑	<--	↑	X	<--	↑	<--	<--
U* - Rn	↑	↑	↑	↑	↑	X	↑	↑	<--
U1 - R*	↑	<--	<--	↑	<--	<--	X	<--	<--
Un - R*	↑	↑	<--	↑	↑	<--	↑	X	<--
U* - R*	↑	↑	↑	↑	↑	↑	↑	↑	X

↑	= Rule in the up side must be selected
<--	= Rule in the left side must be selected

U1	**= subject for a specific user u**
Un	**= subject for a small group that contains specific user u**
U*	**= subject for a larger group that contains specifi user u**

R1	**= resource for a single entity r**
Rn	**= small resource group that contains specific resource entity r**
R*	**= larger resource group that contains specific resource entity r**

3.6 Domain Communication Agent

In the proposed model, each user is required to use a certificate that is provided by a certificate authority known by the system. In order to access resources in the domain, users should provide their certificates and the system evaluates the request according to requester's certificate id and certificate authority id. In ordinary situations, there is no problem about accessing and evaluation of a request. However, for some reason, if any of the certificates is cancelled by the certificate provider, how can the system be informed about the revoked certificate? Without any communication with the certificate authority, users with the revoked certificates can still access the resources like the other users that have valid certificates. Therefore a communication mechanism with the certificate providers is required in order to prevent such situations.

Each domain should contain a Domain Communication Agent in proposed model. The Domain Communication Agent has two main duties: When a request comes, Domain Communication Agent should first check the validity of the certificate and then should check whether it is revoked by the certificate provider or not.

3.6.1 Certificate Validity Check

Certificate validity check is related to controlling a certificate in order to answer the following questions. Is the certificate is still valid? Is the certificate really given by a certificate provider? Is the certificate being used by its owner? Certificate validity check contains 3 steps.

1.Checking certificate's validity interval: A Certificate contains a validity period data. When a certificate is retrieved for validity check, it is first controlled to determine whether it is valid currently or not. If it fails the next steps are omitted.

2.Checking a certificate is really given by a known provider or not: As defined in previous chapter, certificates store an encoded data which is signed version of certificate with certificate provider's private key. Decoding that data with certificate provider's public key should provide the same output that certificate contains. If result is successful, then it is guaranteed that certificate is given by a trusted certificate provider and certificate is not modified after published by provider.

3.Checking certificate is being used by its owner: When a user tries to access a resource,

system generates a random number (nonce) and sends it back to the user. Then user signs the random number and sends it to the system with his/her certificate. Then system decodes encoded data with public key inside certificate and check whether decoded data and retrieved random number are the same. If checking is successful then it is guaranteed that certificate is really sent by the one whom has the private key that should only be known by the certificate owner.

If all three steps are successfully passed then certificate is accepted as a valid certificate.

3.6.2 Certificate Revocation Check

Revoked certificates should be published by the certificate providers. Certificate revocation can be handled in several different ways. Two of the most commonly used techniques are discussed here.

First technique assumes that a server is set up in each domain and all valid certificates are stored in a directory. When an administrator revokes a certificate, it will automatically be removed from the directory. Later on, whenever a user with revoked certificate tries to authenticate his/her certificate, it is searched in the directory in the home domain and authentication request of the user fails. [34] First revocation technique is summarized in Figure 3.9.

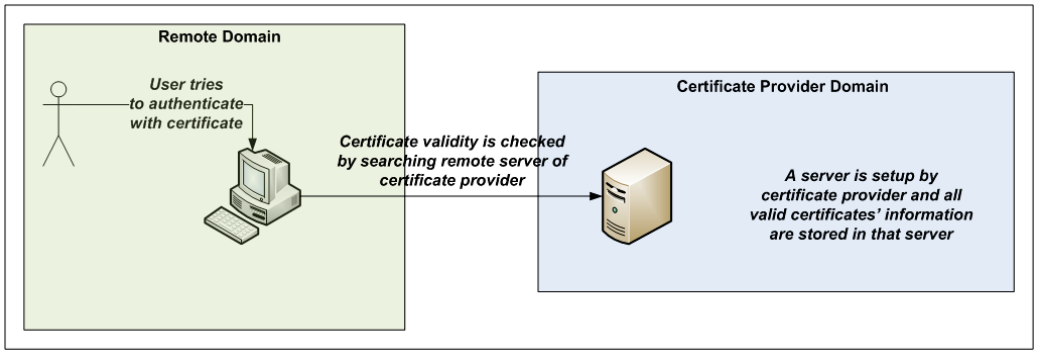

Figure 3.9: First Approach of Revoked Certificate Check

Second technique involves publishing a certificate revocation list (CRL). Certificate revocation list contains the list of revoked certificates. It is published at regular intervals. When a user tries to authenticate, its certificate is searched in the published revocation lists instead of directly checking the certificate from the system of the certificate provider. [3] Diagram for

the revocation technique with CRL is given in the Figure 3.10.

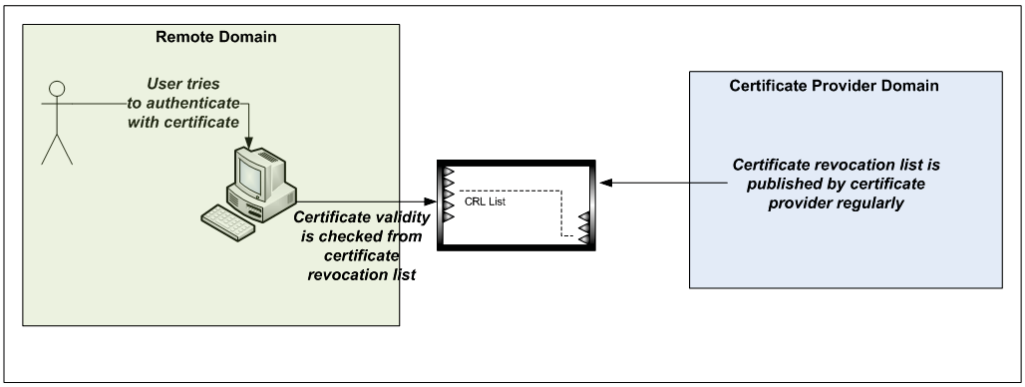

Figure 3.10: Second Approach of Revoked Certificate Check

In the first technique, the certificate authority should provide a system for the authenticators and authenticators should check the certificate directly from certificate authority but it supports the real time status checking; i.e., when a certificate is revoked, it immediately starts to fail in authentications. In the second technique, publishing a list is enough for authenticators and authentication procedure does not need to connect to certificate authority but the list may not contain the current revoked certificate lists because CRL is not immediately refreshed but it is refreshed in some regular intervals. According to the benefits and losses, different types of techniques may be selected.

For the proposed system, using first technique requires a direct network access to all certificate providers. While a user waits for the access request response, the system should check validity of the certificate from home system which may cause a long latency. Besides, there may be many access requests at the same time. As a result, very high network bandwidth may be required. Therefore, choosing the first technique is not suitable for the proposed system because it may require some extra hardware and latency for the user may be frustrating.

On the other hand, the second technique also causes some problems because certificate revocation lists are stored in the certificate providers and each access request needs a certificate validity check from home systems. However with some modification in the second technique remote certificate validity checks can be avoided.

Domain communication agents use the certificate revocation lists of the certificate providers in order to check validity of the requesters' certificate. Nevertheless, instead of remote checking,

certificate revocation lists are copied to local domain at regular intervals and consequently certificate validity checks are done locally. Disadvantage of proposed solution is that revoked certificates are not copied to local lists in real time. However, by defining the updating interval appropriately, revoked certificates are transferred to local CRL's faster and difference between local and remote CRLs are minimized.

Domain communication agent has two main tasks in the proposed system. Firstly, when a user tries to access a request, his/her certificate data is send to domain communication agent and it checks the validity and also checks certificate status from local certificate revocation list of the related certificate provider. Secondly, domain communication agent updates local CRLs from the remote CRLs' of the certificate providers in some predefined time intervals. The activity diagram of the domain communication agent is given in the Figure 3.11.

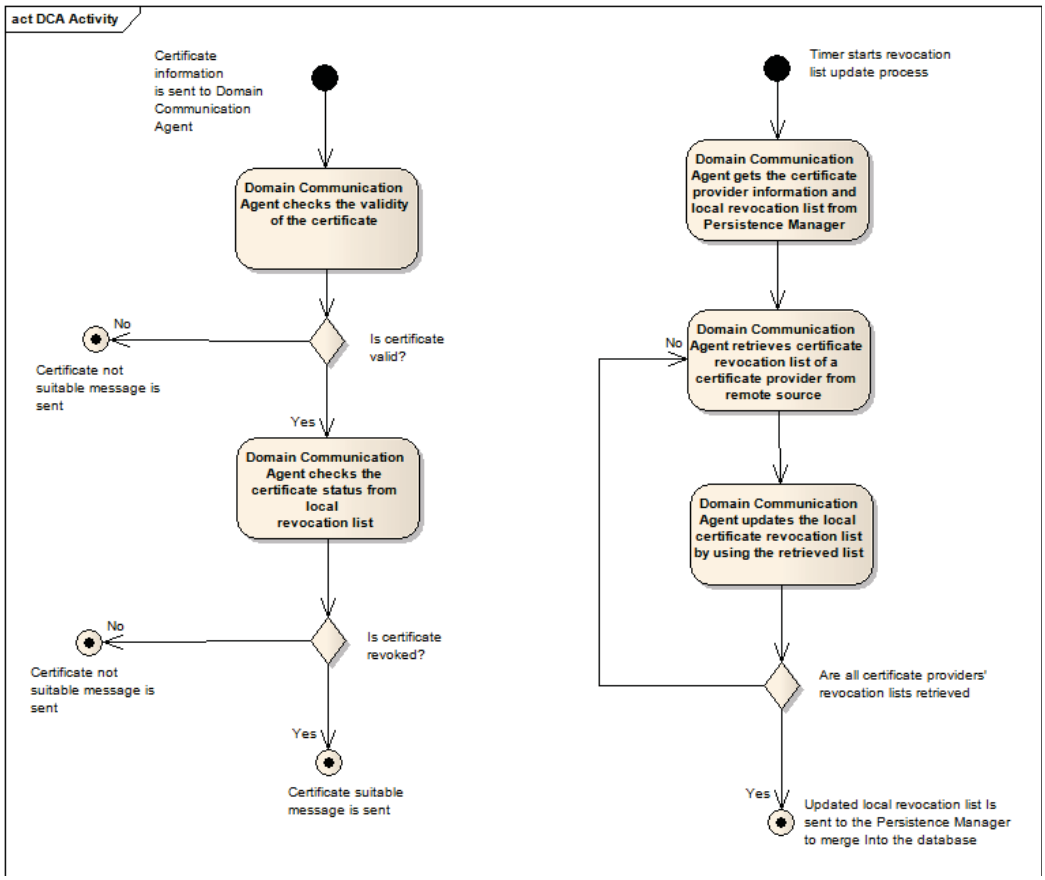

Figure 3.11: Activity Diagram of Domain Communication Agent

3.7 Policy Rule Engine

Policy rule engine is the core component of the system. An incoming access request is retrieved by the policy rule engine and by coordinating other components the policy rule engine evaluates the access requests and sends decision to the requester.

When a user tries to access a resource with his/her certificate, the requested resource and the certificate data are retrieved by the policy rule engine. Policy rule engine sends certificate data to domain communication agent in order to check the validity of the certificate. In spite of the fact that all certificate data are valid, it may have been cancelled by the certificate provider. Domain communication agent controls certificate validity by checking whether it is revoked or not. After the validity test, the result is sent back by the domain communication agent to the policy rule engine. If the certificate of the requester cannot pass the validity test then access request directly fails and user cannot access the resource. If certificate of the requester pass the validity test then a session is created for the certificate and requested resource data and requester's certificate data are sent to context rule service. After querying persistent storage, context rule service returns the related access policy rules to the policy rule engine. After gathering access policy rules, the policy rule engine initiates the policy evaluation mechanism. The policy evaluation mechanism includes the acquisition of the context data as it will be described in the next part in detail. According to the result of the evaluation of access policy rules, policy rule engine returns the access request result to the user and the resource requested if access to the resource can be granted. If evaluation is passed user can access the requested resource and if evaluation is failed then user can not access the requested resource.

The next time user tries to access a resource in domain, policy rule engine checks the user session and if user session is still available then certificate validity check step in domain communication agent is skipped. Flow diagram and component diagram of the policy rule engine is given in the Figure 3.12 and Figure 3.13.

3.7.1 Policy Evaluation Mechanism

After access policy rules about the certificate of the user and the requested resource are queried from the context rule service, the matching access policy rules are returned to policy rule engine and policy evaluation mechanism is initiated with the found access policy rules list. If

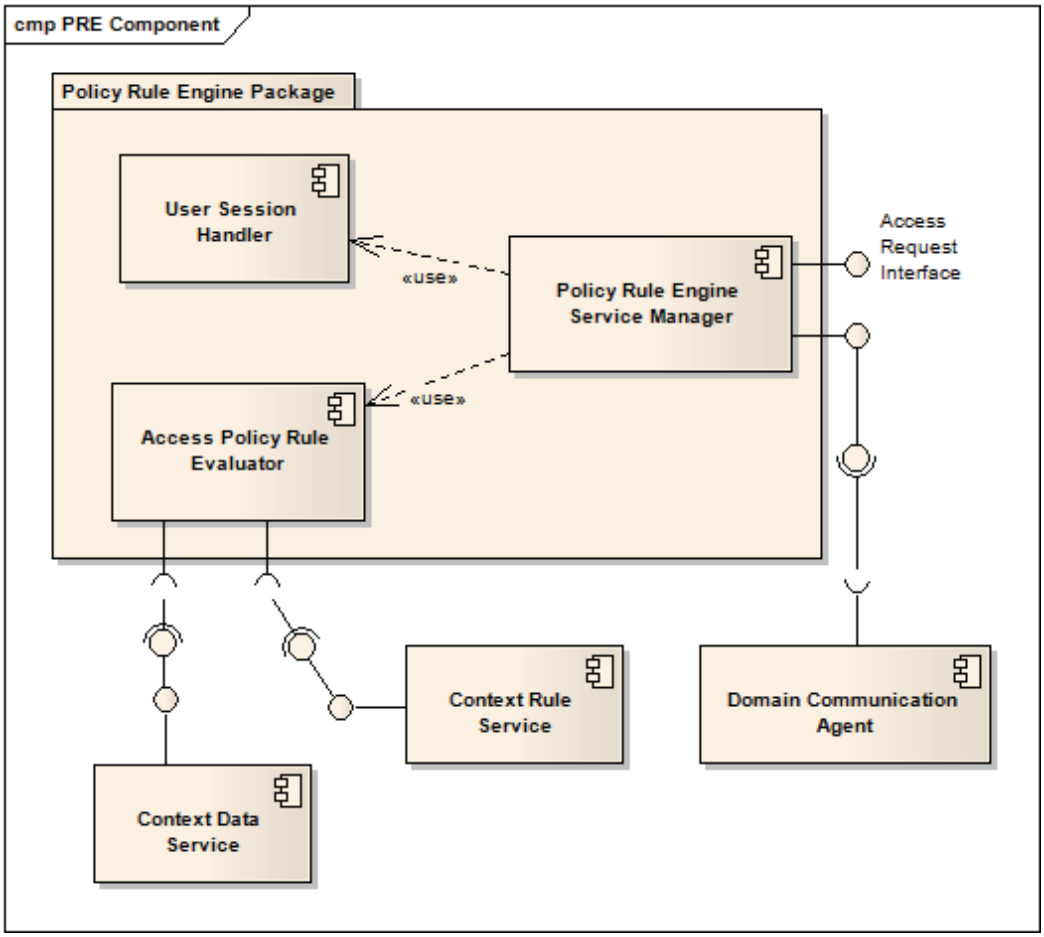

Figure 3.12: Component Diagram of Policy Rule Engine

the returned access policy rule list is empty then policy evaluation mechanism will directly return a failure result.

If the returned access policy rule list is not empty then for each rule, related context type is identified and for the given certificate user and identified context type, context data is asked from context data service. In proposed system context data service is kept as a separate module and context data retrieval and matching is assumed to be done by context data service internally.

After the retrieval of context data, context evaluation is started for the given access policy rules. Each context type has a different evaluation mechanism. Therefore, when a new context type is desired to be added to the system, it is also required that context evaluation algorithm for the new context type should be implemented. The context evaluation techniques for time

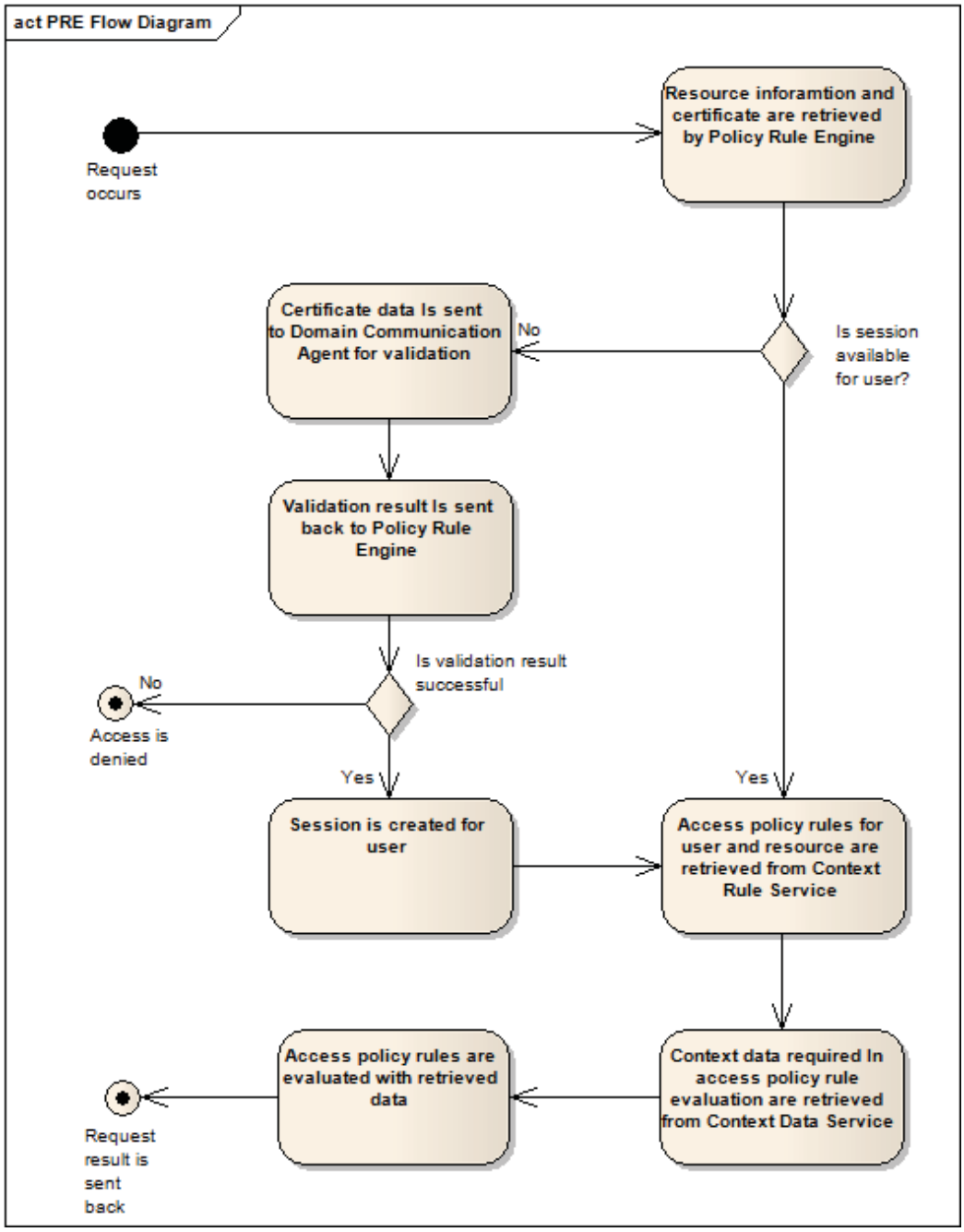

Figure 3.13: Policy Rule Engine Flow Diagram

and location context will be described here and for other context types, similar techniques can be implemented.

3.7.1.1 Time Context Evaluation

For time context, each rule contains a date range or a specific date. For date range, a lower and an upper time limit are given. Besides context data, a data format to be checked for the context data is contained in access policy rules. For example <r,09:00-18:00,HH:mm> means this is a range context data and also lower time limit is 09:00, upper time limit 18:00 and retrieved context data is checked only in hours and minutes. Similarly, <e, Sunday, EEEE> means, this is a specific date and only day of week field of the retrieved context data is checked. Java Date/Time patterns [35] are used as the date format in proposed system. The full list of date format patterns is given in the Appendix A.

3.7.1.2 Location Context Evaluation

For location context, each rule contains a range or a specific location similarly to time context described above. The specific location may be defined by using wildcards for matching the precision. In range type <r,36:20:10N72:27:43W-36:20:15N72:27:40W> and in specific location <e,36:20:**N,72:27:**W> are sample location context rules. First rule means, context data should be in the range of 36:20:10 North and 36:20:15 North latitudes and in the range of 72:27:43 West and 72:27:40 West longitudes. On the other hand, second rule means, context data should be equal to 36:20 North latitude and 72:27 West longitude without considering the precisions in seconds. Sample access policy rules for time and location context are given in Figure 3.14. Note that access policy rules are given in APML format for readability.

3.7.2 Multiple APR for Access Request

In ordinary situations, it is expected that one access policy rule is found in database about a user for requesting a resource. In those cases, access policy rule is evaluated with context data according to the algorithm for related context type. However there may be more than one rule available for the related user and the resource. In such cases, a multiple rule evaluation technique is required.

In proposed system, it is guaranteed that context rule service never returns more than one rule for the requester in the same context condition. If there exists multiple access policy rules for

```
<apr>
  <subject type="user_certificate">1568797</subject>
  <resource type="resource">printer48</resource>
  <context type="time" condition="range" format="hh:mm">09:00-18:00</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="certificate_provider">1658</subject>
  <resource type="group">cs_department_printers</resource>
  <context type="location" condition="range">36:20:10N72:27:43W-36:20:15N72:27:40W</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="group">1214</subject>
  <resource type="group">metu_printers</resource>
  <context type="time" condition="equality" format="EEEE">SATURDAY,SUNDAY</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="group">764</subject>
  <resource type="resource">shopping_mall_cinema_discount%50</resource>
  <context type="time" condition="equality" format="EEEE">WEDNESDAY</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="certificate_provider">45678</subject>
  <resource type="group">shopping_mall_floor1_stores_discount%25</resource>
  <context type="location" condition="equality">36:20:**N72:27:**W</context>
  <permission>allow</permission>
</apr>

<apr>
  <subject type="user_certificate">77769548</subject>
  <resource type="group">shopping_mall_floor1_stores_discount%25</resource>
  <context type="time" condition="range" format="EEEE">MONDAY-FRIDAY</context>
  <permission>deny</permission>
</apr>
```

Figure 3.14: Access Policy Rules in Time and Location Context

the same context condition in the database, context rule service applies a filtering mechanism and eliminate all rules except for one. Filtering mechanism was discussed in the Context Rule Service section. Hence, all rules retrieved by policy rule engine are in different context condition. For instance, if there is a rule with 'weekend' context condition, it is guaranteed that there is not any other rule retrieved with same 'weekend' context condition.

When there is more than one rule, each rule is separately evaluated and final evaluation result is found by 'Logically Or'ing same context type rules and then 'Logically And'ing the result of different context type rules. For example, if there are four different rules that say user can access resource in Wednesday, user can access resource in Thursday, user can access resource in location X and user can access resource in location Y then final result will be that user can access resource in Wednesday or Thursday and in location X or location Y.

Context data acquisition and matching is a costly task therefore eliminating some of the rules before gathering context data for all rules may be useful. 'Logically And'ing and 'Or'ing the result of each rule gives a chance of giving precedence to each context types. By this way, if we give more precedence to context types that are more easily acquired, we can get rid of some more complex context data retrieval procedure. For example, acquiring time data is much more easy then acquiring users' location information so giving high precedence to time context and evaluating time context related rules first before evaluating location related context rules may get rid of the system from some unnecessary location related context data retrieval procedure. Let's assume that there is a rule that says user can access resource in morning and another rule says user can access resource in location X. Then, if user tries to access resource in evening, system should not unnecessarily evaluate location related rule and also should not try to retrieve location data of the user by firstly evaluating time related access policy rule.

Meanwhile, deny rules have more precedence than allow rules. Because, if one rule is denying user from accessing a resource and his/her context meets the rule requirements then evaluating other rules becomes unnecessary task. As a result, if there are two different rules that have same precedence levels then deny type rules should evaluated first.

To sum up, considering time and location context, if there is more than one rule available in access request decision evaluation phase, following rule evaluation order is applied. Firstly, time context related access policy rules that deny user from accessing resource are evaluated. If evaluation succeeds then deny message is directly sent. Secondly, time context related access policy rules that allow user to access resource are evaluated. If evaluation fails, then denying message is sent. Then same procedure applied for location context related access policy rules. Finally, if process can be finalized without a denying message, an allowing message is sent. Evaluation procedure is given as a formula in the Figure 3.15.

3.8 Policy Management Interface

Policy Management Interface works as an administration interface in proposed system. Through it administrators maintain system definitions and properties. Main task of administrators can be described as defining new access policy rules into the system. As it is described before,

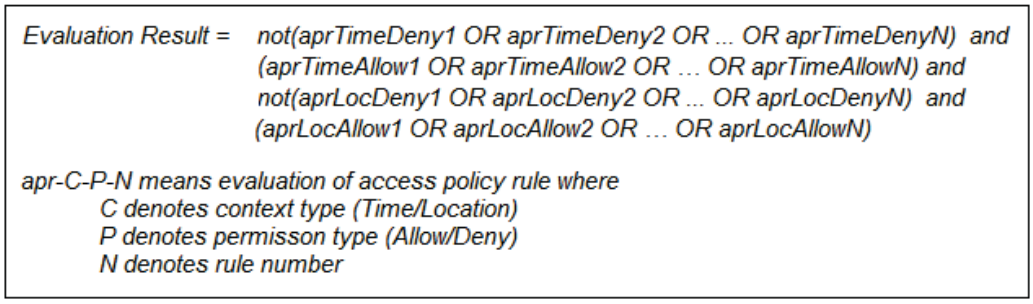

Figure 3.15: Formule for the Access Policy Rule Evaluation

access policy rules are encapsulated into APML format. For administrators, describing rules in xml based structure can be difficult. With Policy management interface, they can describe new access policy rules, or update/delete existing access policy rules via a user friendly graphical interface. When they insert or update a rule, related rule is validated first according to rule structure by Context Rule Service before it is written into persistent storage.

User certificates and certificate providers' certificates are not needed to be stored in persistent storage entirely. Availability of their identity in related rules is enough to access resources in domain. However, subject groups in rules required to be defined in subject table. For that reason, administrators can define new subject groups for domain over the policy management interface.

Administrators are also required to define resource and resource groups in resource table in database. Resource definition tasks are also maintained over policy management interface.

Table 3.2: Tasks Performed By Policy Management Interface

Task Name	Explanation
Manage Access Policy Rules	Includes Add, Remove and Update Operations For Access Policy Rule In DB
Managing Subject Groups	Management of Subject Groups that stores group information For Users
Managing Resource	Domain resource definition Operations
Managing Resource Groups	Defining Resource Group For Resources Operations
Managing Parameters	Definition of Parameters such as revocation list update interval
Managing Certificate Providers	Management of Certificate Provider Information That Are Known By Domain

In order to hinder revoked certificate owners from accessing resource, certificate revocation

lists in domain communication agents should be updated periodically in defined time intervals. Besides, URL of certificate revocation list for a certificate provider can be changed and therefore it should be parametric in order to maintain those changes. On the other hand, certificate of domain provider should also be stored in database to validate user certificates. Policy management interface provides an interface to administrators to maintain certificate providers information defined above.

To sum up, policy management interface provides a graphical user interface for administrators to carry on management tasks. List of tasks can be done over policy management interface is given in Table 3.2.

CHAPTER 4

SAMPLE USAGE SCENARIOS FOR PROPOSED MODEL

In this chapter, two sample usage scenarios will be examined to demonstrate the applicability of the proposed model. Scenarios may be realized by using the prototype application that will be analyzed in the next chapter. Firstly, the scenarios will be defined and then configuration details and application of sample cases on the prototype are examined for each scenario.

4.1 Scenarios for Proposed Model

In this section two different usage scenarios for the proposed model will be introduced. First scenario is the application of the proposed model in a university campus environment where users try to access different resources in the campus. Second scenario is the application of the proposed model in a shopping mall where users get different discounts in different stores of the shopping mall.

4.1.1 Scenario 1: University Campus

In this scenario, we assume that the host university is Middle East Technical University (METU) and it has some inter-domain service level agreement with Istanbul Technical University (ITU) to provide some services to ITU users in METU campus. Both METU and ITU issue certificates to their students and staff. In this scenario, the following requirements will be assumed and satisfied by the proposed model.

- User Groups: METU Computer Science (CS) Users, METU Business Administration (BA) Users.

- Locations: METU Campus, CS Department, BA Department, Library.
- Time Data: February, Weekends and Academic Terms.
- Resources: Printers and online services.
- METU Users can access printers if they are in the campus environment.
- METU CS Users can access online services only if they are in CS Department or Library.
- METU BA Students can access online services only if they are in BA Department or Library.
- ITU Users can access printers only if they are in CS Department, BA Department or Library.
- ITU Users can access online services only if they are in Library.
- ITU Users cannot access printers and online services at weekends.
- METU Users can access printers and online services only in academic terms.
- METU User, Ahmet Doğan, should not access any resource in February.
- ITU User Ayşe Koç's certificate is revoked for some reasons.

4.1.2 Scenario 2: Shopping Mall

In this scenario, a shopping mall provides some discount as a resource to its customers. It is assumed that shopping mall has an inter-domain service level agreement with telecommunication companies Turkcell, Vodafone and Avea. All telecommunication companies issue certificates to their users. In this scenario, the following requirements will be satisfied:

- Locations: Shopping Mall, Electronic Shop, Supermarket and Cinema (which is also on the 1st floor).
- Time Data: Weekends, Wednesday and Tuesday.
- Turkcell and Vodafone users get %10 discount if they are in shopping mall.
- Turkcell users get %20 discount if they are in Electronic Shop.
- Vodafone users get %20 discount if they are in Supermarket.
- Turkcell users get %50 discount if they are in Cinema in Wednesday.
- Vodafone users get %40 discount if they are in Shopping Mall in Tuesday.
- In weekends, all discounts are cancelled for Vodafone and Turkcell.
- Turkcell user, Ali Yıldırım, certificate is revoked by Turkcell for some reason.

4.2 Scenario Implementations

In this section, implementation details are given and some example cases are investigated for the example scenarios. It is assumed that certificates similar to the certificate given in Figure 2.5 are used.

4.2.1 Campus Scenario

In campus scenario, university members are defined as users. METU campus is selected as host domain and ITU is selected as beneficiary domain.

4.2.1.1 Configuration Details

In the university campus scenario, METU and ITU are defined as certificate providers. It is assumed that they both issue certificates to their members and also they publish revoked certificate list for the requesters. Suppose that revocation lists are checked in every 60 seconds.

Two different subject groups are defined in the scenario. These are Metu Computer Science users and Metu Business Administration users. On the other hand, following resource groups are defined: printers and online services. Printers also consist of computer science printers and business administration printers as sub resource groups. Similarly online services contain online library as sub resource group.

Seven different contexts are defined for this scenario. Defined contexts are given in Table 4.1. Note that if RangeCheck value is '1' then related context defines a range value, otherwise it defines an equality value. Location context data are defined in terms of latitudes and longitudes. On the other hand, time context data are defined according to Java Date/Time Patterns defined in Appendix A.

Table 4.2 contains the access policy rules defined in this scenario. Each access policy rule contains a context condition, subject id , resource id, and a permission type which can be 'allow' or 'deny'.

Finally, system is configured such that certificate of a METU user with subject id hasanb and ITU user with subject id aysek are revoked and published in revocation lists in their certificate

Table 4.1: Contexts for Campus Scenario

ContextID	ContextType	RangeCheck	CheckingData
MetuCampus	Location	1	40:20:10N35:10:00E-40:25:10N35:20:00E
CSDepartment	Location	1	40:22:00N35:12:00E-40:23:00N35:13:00E
BADepartment	Location	1	40:24:00N35:12:00E-40:25:00N35:13:00E
Library	Location	0	40:21:**N35:18:**E
Weekend	Time	1	Saturday-Sunday,EEEE
AcademicTerm	Time	1	January-June,MMMM
February	Time	0	February,MMMM

Table 4.2: Access Policy Rules for Campus Scenario

ContextID	SubjectID	ResourceID	PermissionType
MetuCampus	METU	Printers	allow
CSDepartment	METU_CS_Users	OnlineServices	allow
Library	METU_CS_Users	OnlineServices	allow
BADepartment	METU_BA_Users	OnlineServices	allow
Library	METU_BA_Users	OnlineServices	allow
CSDepartment	ITU	Printers	allow
BADepartment	ITU	Printers	allow
Library	ITU	Printers	allow
Library	ITU	OnlineServices	allow
Weekend	ITU	OnlineServices	deny
Weekend	ITU	Printers	deny
AcademicTerm	METU	OnlineServices	allow
AcademicTerm	METU	Printers	allow
February	ahmetd	Printers	deny

providers.

4.2.1.2 Sample Cases

In this section, different access requests to the resources and their evaluations are examined.

Case 1: METU user with subject id 'ahmetd' tries to access a printer in METU when he/she has following context:

40:22:10N35:13:43E as location context (inside METU Campus)

06.01.2011-14:45:43 as time context (in Academic Term, but not February)

Result for Case 1: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
MetuCampus	METU	Printers	allow
AcademicTerm	METU	Printers	allow
February	ahmetd	Printers	deny

Case 2: METU Computer Science user with subject id 'velik' tries to access online library service when he/she has following context:

40:22:34N35:12:42E as location context (inside METU Computer Science Department)

06.01.2011-14:45:43 as time context (in Academic Term)

Result for Case 2: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
CSDepartment	METU_CS_Users	OnlineServices	allow
Library	METU_CS_Users	OnlineServices	allow
AcademicTerm	METU	OnlineServices	allow

Case 3: METU Business Administration user with subject id 'akifb' tries to access online library service when he/she has following context:

40:24:36N35:12:23E as location context (inside METU Business Administration Department)

06.01.2011-19:42:54 as time context (in Academic Term)

Result for Case 3: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
BADepartment	METU_BA_Users	OnlineServices	allow
Library	METU_BA_Users	OnlineServices	allow
AcademicTerm	METU	OnlineServices	allow

Case 4: ITU user with subject id 'mustafat' tries to access a printer in METU when he/she has following context:

40:24:36N35:12:23E as location context (inside METU Business Administration Department)
06.01.2011-19:42:54 as time context (not in Weekend)

Result for Case 4: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
CSDepartment	ITU	Printers	allow
BADepartment	ITU	Printers	allow
Library	ITU	Printers	allow
Weekend	ITU	Printers	deny

Case 5: ITU user with subject id 'mustafat' tries to access online library service when he/she has following context:

40:21:36N35:18:23E as location context (inside METU Library)
06.01.2011-12:34:24 as time context (not in Weekend)

Result for Case 5: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
Library	ITU	OnlineServices	allow
Weekend	ITU	OnlineServices	deny

Case 6: ITU user with subject id 'mustafat' tries to access online library service when he/she has following context:

40:21:36N35:18:23E as location context (inside METU Library)
08.01.2011-09:21:05 as time context (in Weekend)

Result for Case 6: System returns an access denied message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
Library	ITU	OnlineServices	allow
Weekend	ITU	OnlineServices	deny

Case 7: METU user with subject id 'ahmetd' tries to access a printer in METU when he/she has following context:

40:22:10N35:13:43E as location context (inside METU Campus)

06.08.2011-14:45:43 as time context (not in Academic Term)

Result for Case 7: System returns an access denied message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
MetuCampus	METU	Printers	allow
AcademicTerm	METU	Printers	allow
February	ahmetd	Printers	deny

Case 8: METU user with subject id 'ahmetd' tries to access a printer in METU when he/she has following context:

40:22:10N35:13:43E as location context (inside METU Campus)

06.02.2011-14:45:43 as time context (in Academic Term, in February)

Result for Case 8: System returns an access denied message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
MetuCampus	METU	Printers	allow
AcademicTerm	METU	Printers	allow
February	ahmetd	Printers	deny

Case 9: ITU user with subject id 'aysek' tries to access a printer in METU when he/she has following context:

40:21:36N35:18:23E as location context (inside METU Library)

06.02.2011-14:45:43 as time context (not in Weekend)

Result for Case 9: System returns an access denied message to the user. Following access policy rules are available but are not evaluated since user is in revoked certificate list.

ContextID	SubjectID	ResourceID	PermissionType
CSDepartment	ITU	Printers	allow
BADepartment	ITU	Printers	allow
Library	ITU	Printers	allow
Weekend	ITU	Printers	deny

Case 10: METU user with subject id 'cemilt' tries to access a printer in METU when he/she has following context:

40:21:36N35:18:23E as location context (inside METU Library)

06.02.2011-14:45:43 as time context (in Academic Terms)

Result for Case 10: System returns an access denied message to the user. Following access policy rules are available but are not evaluated since user certificate is not in valid time period.

ContextID	SubjectID	ResourceID	PermissionType
MetuCampus	METU	Printers	allow
AcademicTerm	METU	Printers	allow

4.2.2 Shopping Mall Scenario

In shopping mall scenario, telecommunication company users are defined as users. Shopping mall is selected as host domain. Turkcell and Vodafone are selected as beneficiary domains. It is assumed that shopping mall only serves as a domain providing services. On the other hand, Turkcell and Vodafone are assumed that they only act as certificate provider by not actually providing a host domain to their own users. Their users only benefit from domains like shopping mall that has an inter-domain service level agreement with them.

4.2.2.1 Configuration Details

In this scenario, Turkcell and Vodafone are defined as certificate providers. It is assumed that they both give certificate to their subscribers. Also it is assumed that they publish revoked certificate list for the requesters. Revocation lists are updated at every 60 seconds.

Only one user group called 'All Users' which includes all users in the shopping mall is defined in this scenario. Similarly, one resource group called 'discount' which describes discounts in stores of shopping mall is defined.

Seven different contexts are defined for this scenario. Defined contexts are given in Table 4.3. Note that if RangeCheck value is '1' then related context defines a range value, otherwise it defines an equality value. Location context data are defined in terms of latitudes and longitudes. Time context data are defined according to Java Date/Time Patterns defined in Appendix A.

Table 4.4 contains the access policy rules defined for shopping mall scenario. Each access

Table 4.3: Contexts for Shopping Mall Scenario

ContextID	ContextType	RangeCheck	CheckingData
ShoppingMall	Location	1	40:20:10N35:10:00E-40:25:10N35:20:00E
ElectroShop	Location	1	40:22:00N35:12:00E-40:23:00N35:13:00E
Supermarket	Location	1	40:24:00N35:12:00E-40:25:00N35:13:00E
Cinema	Location	0	40:21:**N35:18:**E
Weekend	Time	1	Saturday-Sunday,EEEE
Wednesday	Time	0	Wednesday,EEEE
Tuesday	Time	0	Tuesday,EEEE

policy rule contains a context condition, subject id, resource id, and a permission type which can be 'allow' or 'deny'.

Table 4.4: Access Policy Rules for Shopping Mall Scenario

ContextID	SubjectID	ResourceID	PermissionType
ShoppingMall	Turkcell	discount_10	allow
ShoppingMall	Vodafone	discount_10	allow
ElectroShop	Turkcell	discount_20	allow
Supermarket	Vodafone	discount_20	allow
Cinema	Turkcell	discount_50	allow
Wednesday	Turkcell	discount_50	allow
ShoppingMall	Vodafone	discount_40	allow
Tuesday	Vodafone	discount_40	allow
Weekend	All_Users	discount	deny

It is assumed that Turkcell user with subject id aliy and ITU user with subject id hasanb are revoked and published in revocation lists of their certificate providers.

4.2.2.2 Sample Cases

Case 1: Vodafone user with subject id 'mahmutg' tries to get

40:21:10N35:11:00E as location context (inside Shopping Mall)

06.01.2011-14:45:43 as time context (Thursday)

Result for Case 1: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
ShoppingMall	Vodafone	discount_10	allow
Weekend	All_Users	discount	deny

Case 2: Turkcell user with subject id 'kamila' tries to get

40:22:10N35:14:00E as location context (inside Shopping Mall)

06.01.2011-14:45:43 as time context (Thursday)

Result for Case 2: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
ShoppingMall	Turkcell	discount_10	allow
Weekend	All_Users	discount	deny

Case 3: Turkcell user with subject id 'kamila' tries to get

40:22:15N35:12:30E as location context (inside Electro Shop)

06.01.2011-14:45:43 as time context (Thursday)

Result for Case 3: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
ElectroShop	Turkcell	discount_20	allow
Weekend	All_Users	discount	deny

Case 4: Vodafone user with subject id 'mahmutg' tries to get

40:24:10N35:12:20E as location context (inside Supermarket)

06.01.2011-14:45:43 as time context (Thursday)

Result for Case 4: System returns an access granted message to the user. Following access policy rules are evaluated.

Case 5: Turkcell user with subject id 'kamila' tries to get

ContextID	SubjectID	ResourceID	PermissionType
Supermarket	Vodafone	discount_20	allow
Weekend	All_Users	discount	deny

40:21:16N35:18:47E as location context (inside Cinema)

05.01.2011-14:45:43 as time context (Wednesday)

Result for Case 5: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
Cinema	Turkcell	discount_50	allow
Wednesday	Turkcell	discount_50	allow
Weekend	All_Users	discount	deny

Case 6: Vodafone user with subject id 'mahmutg' tries to get

40:21:10N35:11:00E as location context (inside Shopping Mall)

04.01.2011-14:45:43 as time context (Tuesday)

Result for Case 6: System returns an access granted message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
ShoppingMall	Vodafone	discount_40	allow
Tuesday	Vodafone	discount_40	allow
Weekend	All_Users	discount	deny

Case 7: Vodafone user with subject id 'mahmutg' tries to get

40:21:10N35:11:00E as location context (inside Shopping Mall)

08.01.2011-14:45:43 as time context (Saturday, Weekend)

Result for Case 7: System returns an access denied message to the user. Following access policy rules are evaluated.

ContextID	SubjectID	ResourceID	PermissionType
ShoppingMall	Vodafone	discount_10	allow
Weekend	All_Users	discount	deny

Case 8: Turkcell user with subject id 'aliy' tries to get

40:21:10N35:11:00E as location context (inside Shopping Mall)

06.01.2011-14:45:43 as time context (Thursday)

Result for Case 8: System returns an access denied message to the user. Following access policy rules are available but are not evaluated since user certificate is in revoked certificate list.

ContextID	SubjectID	ResourceID	PermissionType
ShoppingMall	Turkcell	discount_10	allow
Weekend	All_Users	discount	deny

Case 9: Turkcell user with subject id 'tugceo' tries to get

40:21:10N35:11:00E as location context (inside Shopping Mall)
06.01.2011-14:45:43 as time context (Thursday)

Result for Case 9: System returns an access denied message to the user. Following access policy rules are available but are not evaluated since user certificate has not a valid time period.

ContextID	SubjectID	ResourceID	PermissionType
ShoppingMall	Turkcell	discount_10	allow
Weekend	All_Users	discount	deny

CHAPTER 5

PROTOTYPE IMPLEMENTATION

In this chapter, a prototype implementation will be presented for context aware, certificate based access control model for multi domain environment presented in Chapter 3. The aim of this prototype is to validate and show the applicability of the proposed access control model. For simplicity, only two context types, location and time, are considered in prototype implementation. Prototype will also be configured for two different cases defined in the previous section in order to analyze proposed model's applicability. Firstly high level requirements for the prototype will be given. Then system architecture and detailed design of the prototype will be presented.

5.1 High Level Requirements

High level requirements of the prototype developed are as follows:

- System shall work with two different context types: location and time.
- System shall consist of five different modules: Context Data Service Module, Context Rule Service Module, Policy Rule Engine Module, Domain Communication Service Module and Policy Management Interface Module.
- All system components shall communicate over web services, the application architecture shall support service oriented architecture.
- System shall get user certificate information and resource information that is requested as input.
- System shall return output access request. Result shall be 'allow' or 'denied'.
- System shall provide an administration interface to define new access policy rules, subject

groups, resources, resource groups and context information.
- System shall provide an interface to define URL of the certificate provider in order to get certificate revocation lists from remote sources.
- System shall update certificate revocation lists from remote sources in defined time intervals.
- For simulation/testing purposes, Context Data Service Module shall generate pre-defined context information for defined users. It shall generate context data by using defined context data lists.
- All requirements regarding the model defined in the previous section shall be satisfied by the system with the following exceptions(For the sake of simplicity of the prototype).
* System will not store user session information.
* System will not transfer access policy rules in the format of access policy markup language among modules. Access policy rules are transferred among modules directly as an entity objects.
* Policy Management Interface will not be implemented. Instead, all parameters and values are directly inserted into the database directly.
* Certificate validation will not be implemented. There are lots of different library's that can be used for this purpose, and implementing them may introduce complexity to the prototype. Instead, certificate validity time interval and availability of certificate subject id and provider id will be checked in prototype implementation.

5.2 Activity Diagrams for the Primary Execution Scenarios

Activity Diagram for the primary execution scenarios are given in this section. Scenarios are chosen for different cases that the prototype should satisfy. Three main scenarios are chosen to describe the primary control flows in the system. All these scenarios are depicted in the activity diagram in Figure 5.1. Main responsibility of each module is also shown in the activity diagram.

Scenario 1: User request is accepted because he/she has valid certificate and valid context to reach requested resource.

When a user requests to access a resource, firstly, the Policy Rule Engine sends certificate data of the user to the Domain Communication Agent. The Domain Communication Agent

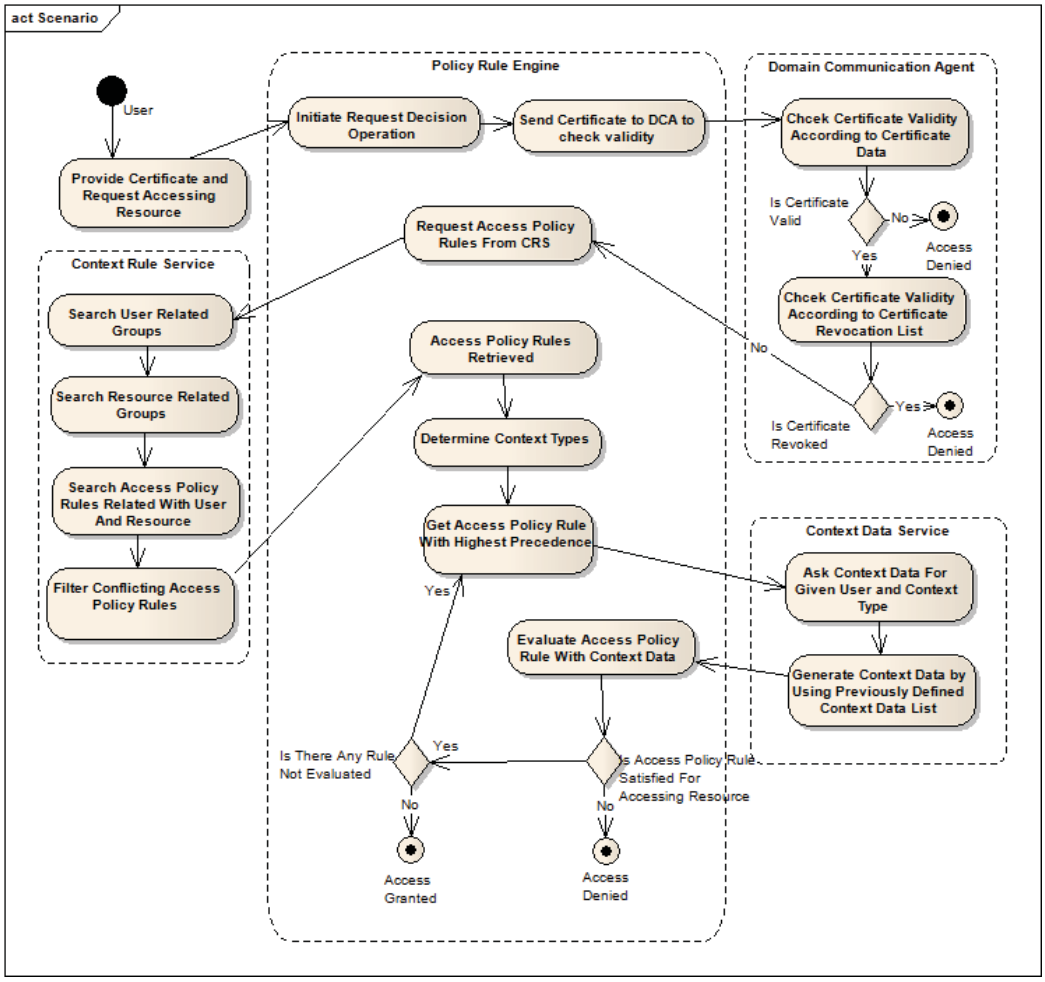

Figure 5.1: Activity Diagram for the Primary Execution Scenarios

checks certificate validity and also checks certificate status from certificate revocation lists. After checking certificate validity, it sends back 'valid' message to the Policy Rule Engine.

Secondly, the Policy Rule Engine asks access policy rules for requested resource and user from the Context Rule Service. The Context Rule Service retrieves user related subject groups and resource related resource groups from the database over the Persistence Manager interface first. Then it searches the access policy rules related to the given user/user group and the resource/resource group. Finally, it filters the conflicting access policy rules according to the 'More Detailed Rule Wins' algorithm defined in the Chapter 3 and returns filtered access policy rules to the policy rule engine.

The Policy Rule Engine determines the context types of each access policy rule and orders

them according to the precedence of the context types. For the prototype implementation there are only two context types defined and the time context has a higher precedence then the location context because time context data are much more easily grasped according to the location context. Besides, denying type access policy rules has more precedence then allowing type rules. Therefore, rules in the same context type are ordered according to their permission types also.

For access policy rule with highest precedence, policy rule engine requests context data from the Context Data Service. The Context Data Service generates a suitable context data for the requested user and context type by using the context data list defined earlier and then returns context data back to the Policy Rule Engine. Policy Rule Engine evaluates access policy rule according to the retrieved context data and since the access policy rule is satisfied to reach resource, it continue with the next access policy rule with the highest precedence. For each access policy rules in the list, same evaluation technique is applied and then access granted message for the requester user is send back to the resource.

Scenario 2: User request is denied because he/she has valid certificate but context is not appropriate to access the requested resource.

Similar steps in scenario 1 are conducted in scenario 2. However, context data service generates an inconvenient context data and after evaluation of access policy rule Policy Rule Engine does not continue to evaluate other access policy rules and return access denied message to the user.

Scenario 3: User request is denied because he/she does not have a valid certificate in spite of the fact that he/she has a valid context to reach resource.

When user requests to access a resource, Policy Rule Engine, firstly, sends certificate data of user to Domain Communication Agent. Domain Communication Agent checks certificate validity and also checks certificate status from certificate revocation lists. Since user certificate is in a certificate revocation list, 'invalid' message is sent back to Policy Rule Engine and Policy Rule Engine sends an access denied message for the requester user to the resource.

5.3 System Design

Prototype system is implemented on top of J2EE technology. Since all modules will be deployed independent of each other, they are implemented as enterprise java beans (EJB) and deployed as EJB web services into the application server, Oracle Weblogic 10.3. For database, Microsoft Access is chosen because of its usage simplicity and easy deployment.

Prototype is implemented as combination of several modules. When the system is considered as black box it has interactions with three external actors. First one is the resources to be accessed, second is the administrator who defines parameters and rules and last one is the remote systems that provide certificate revocation lists. The interaction with the external components is depicted as a diagram in Figure 5.2.

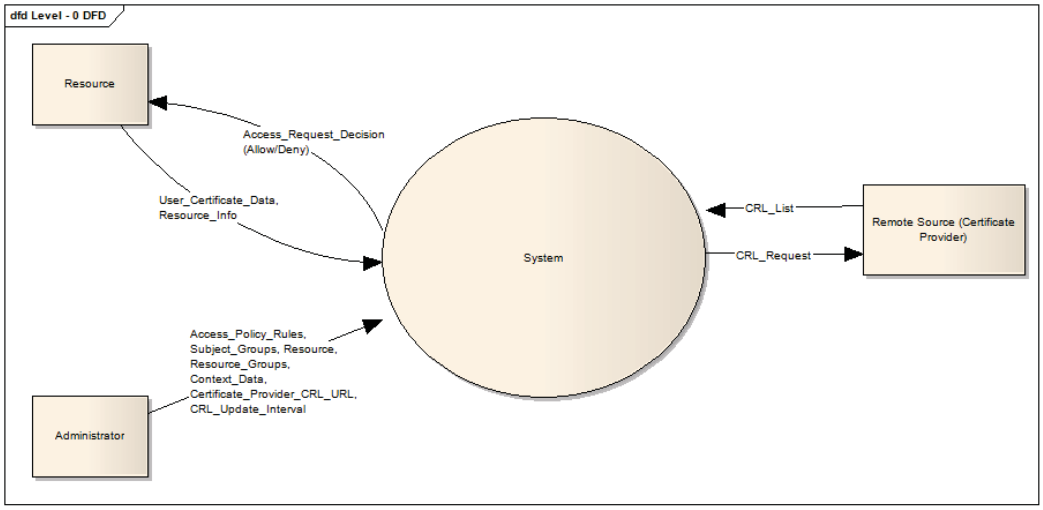

Figure 5.2: Interaction with the External Components

Resources send user certificate data and their own identity information to the system and system returns decision result back to resources. Administrator defines Access Policy Rules, Subject Groups, Resources, Resource Groups, Context Data, Certificate Revocation List URLs and their update intervals to the system. Finally, system sends certificate revocation list data to the remote certificate providers and they send back certificate revocation list for their users.

System is composed of five components. These are Context Data Service, Context Rule Service, Policy Rule Engine, Domain Communication Agent, and Central Database of System with a Persistence Manager module.

5.3.1 Context Rule Service

Context Rule Service searches access policy rules for the users and resources. When a request is retrieved, access policy rules that are related to the user and requested resource are determined and sent to the Context Rule Service.

Context Rule Service is an internal component of the system. That is, it does not have any interface with the external entities. It has interfaces with only other components of the system. First interface of the Context Rule Service is with Persistence Manager. Context Rule Service sends user data and/or resource data to the Persistence Manager and, in return, Persistence Manager sends queried user groups, queried resource groups or queried access policy rules that are related with user and/or resource. Second interface is with Policy Rule Engine. Policy Rule Engine sends user certificate data and requested resource information to the Context Rule Service. Then, Context Rule Service sends back access policy rules to the Policy Rule Engine after processing them.

Context Rule Service is responsible for performing the following tasks. Querying user related subject groups from persistence manager, querying resource related resource groups from persistence manager, querying access policy rules according to user/user groups and resource/resource groups from persistence manager and filtering access policy rules that are conflicting with others. Class diagram for the context rule service is given in Figure 5.3.

Only noteworthy classes and public interfaces of those classes are given in the class diagram. Private/protected classes and classes used internally are not shown for simplicity. Other components reach Context Rule Service over ContextRuleServiceSoap WSDL interface class. Similarly, Context Rule Service use Persistence Manager over CRSPersistenceServiceSoap WSDL interface class which is provided by Persistence Manager.

ContextRuleServiceManager acts as a component router. It uses other classes to generate a request result. PersistenceServiceQueryHandler makes requests from persistence manager to get query results from database. Finally AccessPolicyRuleConflictFilter is responsible for eliminating conflicting access policy rules according to the 'More Detailed Rule Wins' algorithm defined in previous section. Related algorithm is also defined in AccessPolicyRuleConflictFilter class.

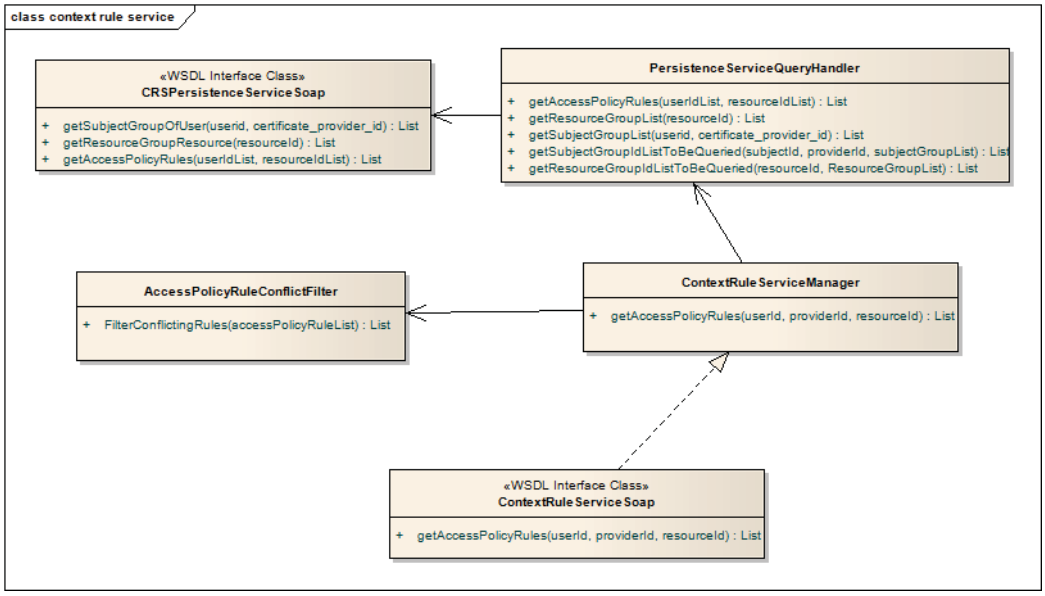

Figure 5.3: Context Rule Service Class Diagram

5.3.2 Domain Communication Agent

Domain Communication Agent checks validity of certificates which are distributed by either host domain or external domains that are known by the host domain. It is responsible for checking real time validity of user certificate. Moreover, it checks revoked certificate lists from each certificate providers' domain in predefined time intervals.

Domain Communication Agent has two interfaces. First interface of the Domain Communication Agent is with Policy Rule Engine. Policy Rule Engine sends user certificate data to Domain Communication Agent and it sends back validity check result after examining certificate data internally. Second interface of the Domain Communication Agent is with an external interface which is remote sources of the external certificate providers. Domain Communication Agent request revoked certificate list from each of the certificate provider and certificate providers send back their revoked certificate lists to the Domain Communication Agent.

Domain Communication Agent is responsible for performing the following tasks. Validating certificate data, checking certificate cancellation status from certificate revocation lists, initiating certificate revocation list retrieve operation and retrieve certificate revocation lists from remote sources. Class diagram for the Domain Communication Agent is given in Figure 5.4. Note that only noteworthy classes and public interfaces of those classes are given in the class

diagram. Private/protected classes and classes used internally are not shown for simplicity.

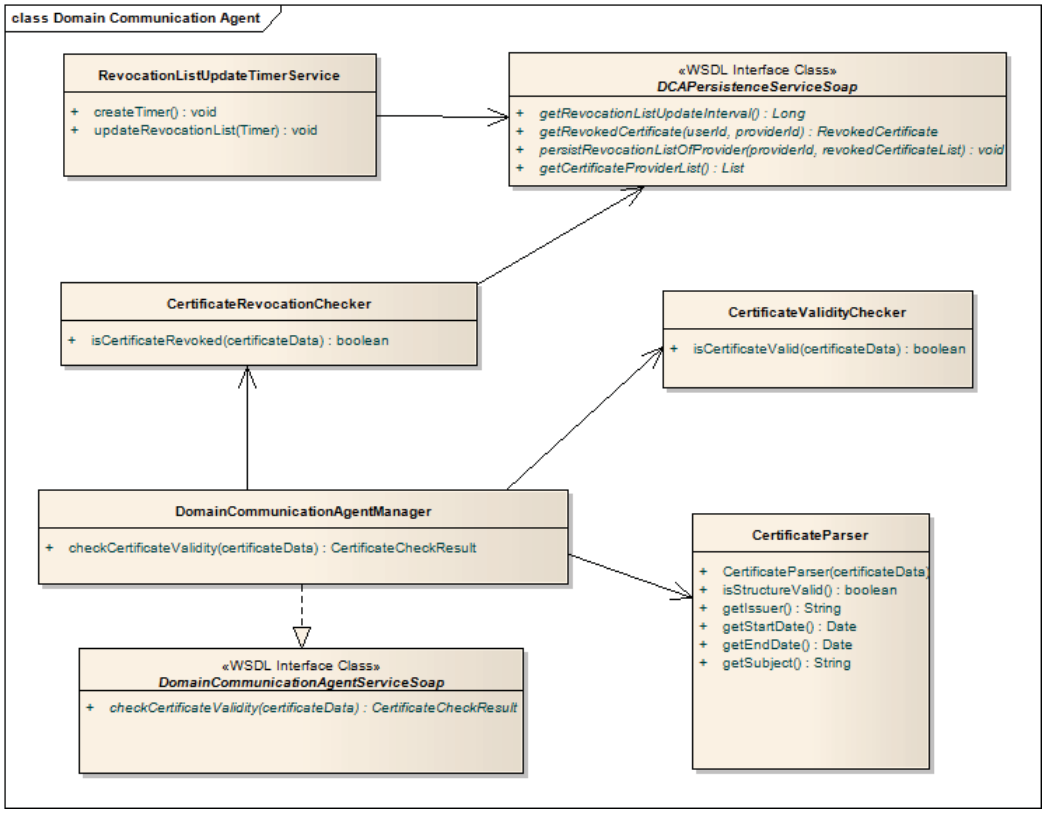

Figure 5.4: Domain Communication Agent Class Diagram

Domain Communication Agent is accessed over DomainCommunicationAgentService WSDL interface class. Similarly, Domain Communication Agent uses Persistence Manager over DCAPersistenceServiceSoap WSDL interface class which is provided by Persistence Manager.

DomainCommunicationAgentManager controls the main data flow of the component. It uses other classes to generate a result for requests coming from other components. CertificateValidityChecker class checks certificate validity. It checks certificate to decide whether it is a real certificate or not. On the other hand, CertificateRevocationChecker class checks certificate in order to determine whether they are revoked or not by querying them from local revocation lists stored in Database. It reaches local revocation lists over DCAPersistenceManagerServiceSoap WSDL interface class. RevocationListUpdateTimerService class controls and manages revocation list update operations. RevocationListUpdater class is responsible for retrieving revocation list of a domain that is defined with a URL in database. Finally

CertificateParser class parse given certificateData and return required fields of it.

5.3.3 Context Data Service

Context Data Service is responsible for generating context data for given user and context type. For real service implementation, it should responsible for context matching and retrieving user context data. However, in proposed system, context data retrieval is out of scope and therefore Context Data Service in prototype implementation is only responsible for generating context data for given user and context types to simulate sample scenarios.

Context Data Service has only one interface that is with Policy Rule Engine. Policy Rule Engine sends user data and context type to the Context Rule Service and it sends back a context data generated by using a predefined data list.

Context Data Service contains three different classes in prototype implementation. First one is ContextDataServiceSoap WSDL interface class. Policy Rule Engine access Context Data Service by using it. Second class is ContextDataServiceManager which takes user data and context type and returns generated context data after encapsulating it. Last class, ContextDataGenerator, produce a context data by using predefined data list. Class diagram for the Context Data Service is given in Figure 5.5.

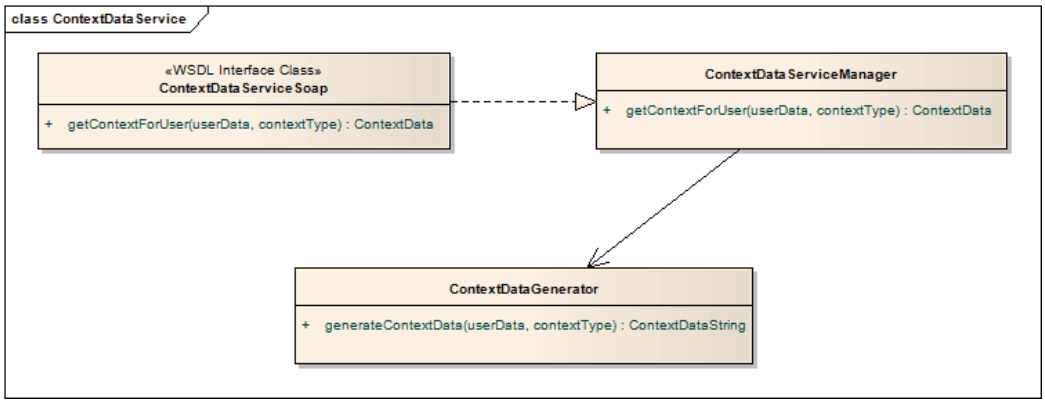

Figure 5.5: Context Data Service Class Diagram

5.3.4 Policy Rule Engine

Policy Rule Engine retrieves user requests and returns access request decision. Policy Rule Engine is the orchestrating component of the system. It evaluates access requests by using other components of the system.

Policy Rule Engine has three different interfaces with other internal components and one interface with an external entity. External entity interface of the Policy Rule Engine is with resources. Actually, it is the interface over which user requests are coming. When user tries to access a resource, resource sends user certificate data and resource info and after evaluating the access request result, Policy Rule Engine sends back request result to the resource. First internal interface of the Policy Rule Engine is with Domain Communication Agent. When Policy Rule Engine sends certificate data to the Domain Communication Agent it checks validity of the certificate and sends validity check result back to Policy Rule Engine. Second internal interface of the Policy Rule Engine is with Context Rule Service. In order to evaluate user access request, Policy Rule Engine needs access policy rules. To get access policy rules, it sends user data and resource info to the Context Rule Service and Context Rule Service sends back related access policy rules back to Policy Rule Engine in access policy markup language format. Last internal interface of the Policy Rule Engine is with Context Data Service. To evaluate access policy rules, Policy Rule Engine should get related context data of the user. Context Data Service supplies user related context data when requested by Policy Rule Engine.

Policy Rule Engine is responsible for carrying out the following tasks. Retrieving user access requests, obtaining access policy rules and related context data, evaluating context data according to access policy rules, sending user access request decisions to the resource requested. Policy Rule Engine class diagram is given in Figure 5.6. Note that only noteworthy classes are given in the class diagram. Private/protected classes and classes used internally are not shown for simplicity.

Requests are done over PolicyRuleEngineServiceSoap WSDL interface class. That is, in prototype requests are sent to system over a web service interface. Policy Rule Engine also accesses other component of the system over web service interfaces of the components. DomainCommunicationAgentServiceSoap, ContextRuleServiceSoap and ContextDataService-

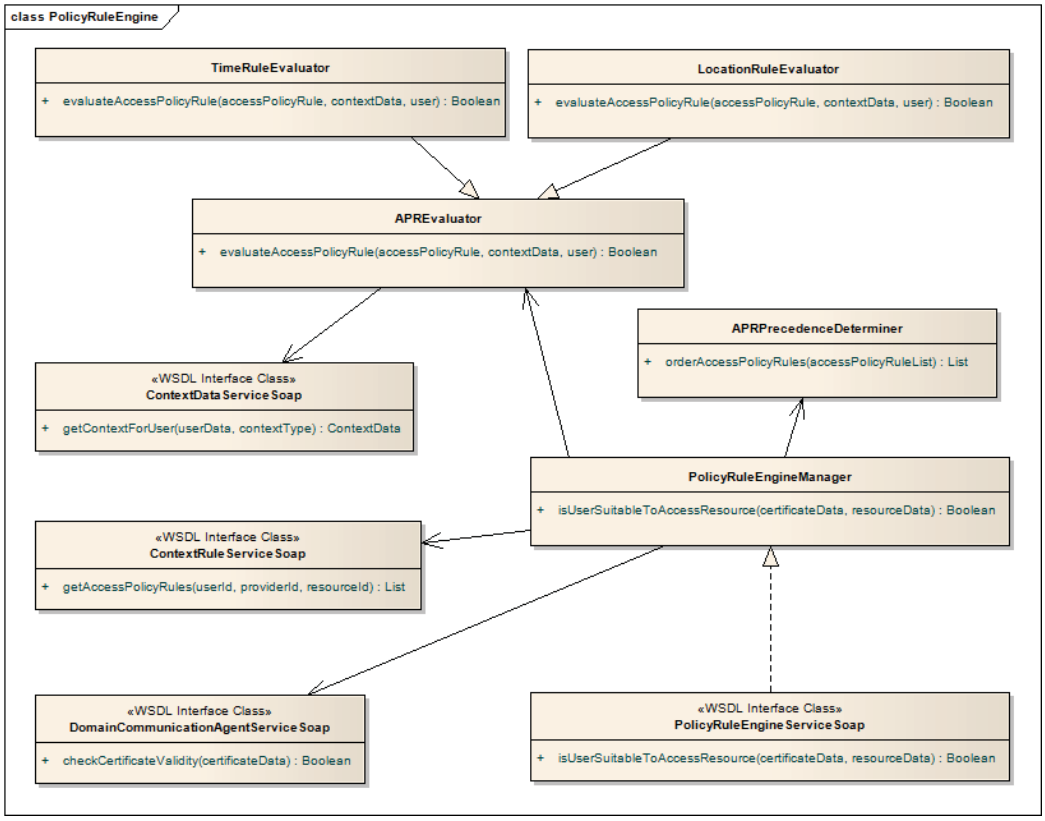

Figure 5.6: Policy Rule Engine Class Diagram

Soap WSDL interface classes are responsible for providing access to related components.

PolicyRuleEngineManager controls the main evaluation mechanism. It calls other objects to calculate request decision. APREvaluator and its extension classes gets an access policy rule and evaluates that rule after retrieving related context data. Finally, APRPrecedenceDeterminer order the access policy rules according to their context types in terms of precedence of context types that are defined in the previous section.

5.3.5 Persistence Manager

Persistence Manager is responsible for retrieving requested resources from database. It works as a database interface for other internal system modules. The Persistence Manager module contains three main classes. First one is the DCAPersistenceManagerServiceSoap WSDL interface class which provides interface to query Domain Communication Agent needs. Second class is CRSPersistenceManagerServiceSoap WSDL interface class which provides interface

to query Context Rule Service requirements. Final class of the Persistence Manager module is the Entity Manager class. It connects to the database and run incoming queries.

5.3.5.1 System Storage

System datastore is also a sub entity of Persistence Manager. Datastore contains six different data tables. These are PolicyRule Table, Subject Table, SubjectGroup Table, Resource Table, ResourceGroup Table, Context Table, RevokedCertificateTable, CertificateProvider Table and Parameter Table. Entity relationship diagram of the system datastore is given in Figure 5.7.

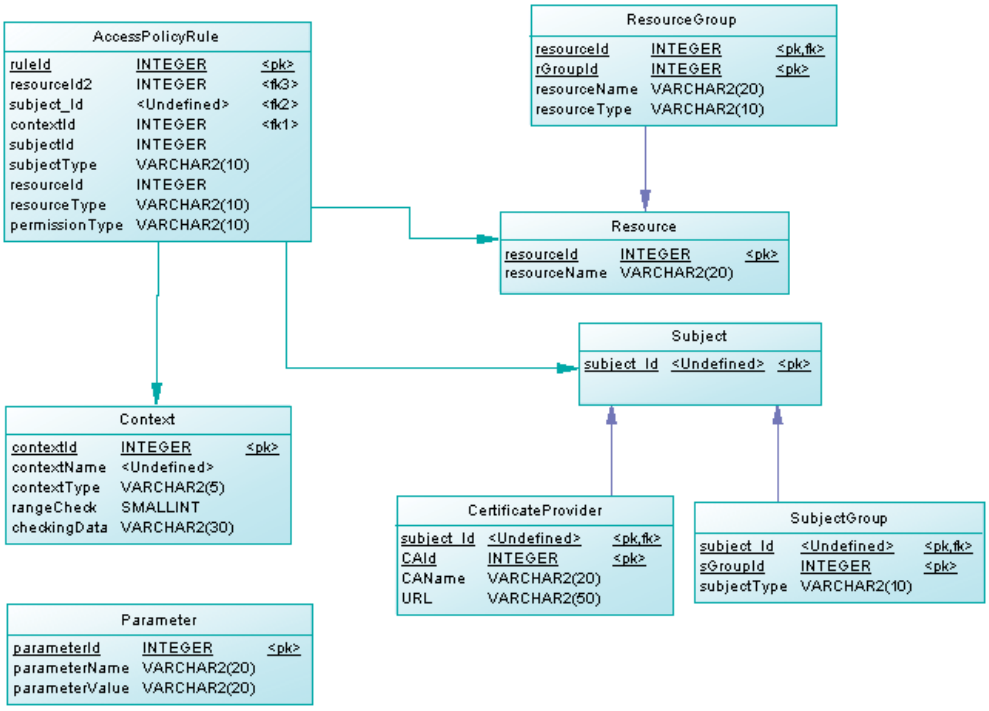

Figure 5.7: ER Diagram of the Prototype

PolicyRule table stores defined access policy rules. It contains subject information, resource information, context information and permission type information. SubjectGroup table stores group information of subjects if available. Besides it also stores group information that contains other sub groups. Resource table stores resource related information and ResourceGroup table stores resource group definitions. Context table is used for mapping context data with defined context names. For example, time context data 'Saturday and Sunday' can be defined

as 'Weekend' in Context Table. CertificateProvider table stores certificate provider information such as name of the provider and revocation list URL of the certificate providers. Finally, Parameter table stores system parameters that can be changed by administrators.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this thesis, a study on context aware access control models is presented with a special emphasis on multi-domain environments. A certificate based approach is proposed to mitigate the inter-domain communication related problems of access control in multi domain environments.

The primary problem that is tackled is the network latency in access request evaluation phase in multi-domain access control. It is crucial to provide fast response to users in ubiquitous systems. When a user from another domain tries to access a resource, ordinary access control models either authenticate and/or retrieve user privileges from the user's home domain. According to the environmental conditions, these interactions may cause significant latency. In order to increase the usability of ubiquitous systems, it is possible to improve the response time of the system by lowering or avoiding such latencies.

In the literature, there are several models that can be used for single domain access control, multi domain access control, and certificate based authentication and access control models. The model proposed in this thesis work exploits the techniques proposed for and applied in different fields to provide a streamlined certificate based, context aware access control model for multi domain environment as a solution.

Our method completely avoids inter-domain communication needs during the access request evaluation phase which is one of the main problems for most of the current multi domain access control models. For this purpose, users of the domains are supplied with certificates and inter-domain service level agreements are stored in local repositories. Besides, due to the fact that the user certificates can be revoked, an inter-domain communication mechanism is proposed to exchange revoked certificate information regularly from home domains. Hence,

users can access resources in different domains without the authorizing them from the home domains if they have valid certificates.

A prototype system based on the proposed model is developed to assess the validity and applicability of the solution. For this purpose, two different configurations are done on the prototype system to support two different scenarios. Firstly, a campus based scenario where members of different universities try to access resources in a campus of a university is examined. Then a shopping mall scenario where users of different telecommunication companies try to access discounts (resource) in stores of a shopping mall is introduced.

Our observation with the prototype implementation and application of the model in two different scenarios using that prototype indicates that the proposed certificate based, context aware access control model considerably improves the access request processing speed by minimizing the inter-domain communication needs.

In the proposed model, certificate validation is done locally without checking the certificate revocation status from the remote sources. This is achieved by regularly retrieving revocation list of each certificate provider and storing them locally in predefined time intervals. Therefore, there may be some obsolete revocation data in locally stored revocation list while remote list has already been changed by the certificate provider. In the proposed model, if the environment is sensitive to such conflicts in local and remote revocation lists, update time interval should be defined small enough to mitigate of such problems considerably.

Proposed model does not differentiate users as local users or guest users. It generalizes all users as 'certificate users'. That is, each user is treated according to his/her user id and the associated certificate provider, and group information defined in his/her certificate. In each domain, the access policy rules for all users of all domains, including the domain's own users, are stored in a single repository. Therefore, during access control, user id and the associated certificate provider is determined and the request is evaluated according to the data provided in certificate and locally stored access policy rules. Such a generalization also enables the proposed solution to be applied in the single domain environments. Even the proposed solution can be applied in environments where host domain has no users but it only provides services to remote domain users (like the second scenario discussed in Chapter 4).

In this thesis work, proposed model makes use of only time and location context. Besides, for

each context dimension, a simple matching method is proposed. In order to incorporate new context dimensions to the system, some simple adjustments is necessary. One of the future works regarding this study might be the generalization of the context matching mechanism to make it easily extendible to introduce new context dimensions and matching algorithms.

Proposed model is designed for multi domain environments. It is expected that it works well in all types of environments and it is scalable to larger domains. However, analyzing the performance of the proposed model in multi domain environments and assessing its scalability might be another future work. Comparing the performance of the proposed model with the performance of a model that communicates with home domain of the requester for authentication and authorization instead of using certificates might be another future research direction.

Only the individual user and group based access policy rules can be defined in the proposed model. However, role definition and role activation or passivation is not covered in the given model. Instead of storing group definitions and evaluating access policy rules according to those groups, storing user roles in certificates and applying role based activation and passivation in access control might be another future research direction.

Another future work might be retrieving domain access policy rules from related domains in regular intervals with a similar approach done in this thesis work for revocation lists of domains. With such an approach, complexity of handling access policy rules in the model can be reduced.

REFERENCES

[1] Mark Weiser and John Seely Brown. *The coming age of calm technolgy*, pages 75–85. Copernicus, New York, NY, USA, 1997.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, 1999. Springer-Verlag.

[3] Mark Weiser. Ubiquitous computing. http://www.ubiq.com/hypertext/weiser/UbiHome.html, March 1996.

[4] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact.*, 7:29–58, March 2000.

[5] H. Lieberman and T. Selker. Out of context: computer systems that adapt to, and learn from, context. *IBM Syst. J.*, 39:617–632, July 2000.

[6] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994.

[7] N. S. Ryan, J. Pascoe, and D. R. Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. In V. Gaffney, M. van Leusen, and S. Exxon, editors, *Computer Applications in Archaeology 1997*, British Archaeological Reports, Oxford, October 1998. Tempus Reparatum.

[8] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[9] Stephen Fickas, Gerd Kortuem, and Zary Segall. Software organization for dynamic and adaptable wearable systems. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers*, ISWC '97, pages 56–, Washington, DC, USA, 1997. IEEE Computer Society.

[10] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.

[11] A. Baki Kocaballı. Granular best match algorithm for context-aware computing systems. Master's thesis, Graduate School of Informatics, Middle East Technical University, Turkey, January 2005.

[12] Terry Winograd. Architectures for context. *Hum.-Comput. Interact.*, 16:401–419, December 2001.

[13] Jason I. Hong and James A. Landay. An infrastructure approach to context-aware computing. *Hum.-Comput. Interact.*, 16:287–303, December 2001.

[14] Anand Ranganathan and Roy H. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Middleware '03, pages 143–161, New York, NY, USA, 2003. Springer-Verlag New York, Inc.

[15] Albrecht Schmidt. *Ubiquitous Computing - Computing in Context*. PhD thesis, Computing Department, Lancaster University, U.K., 2002.

[16] Rene Mayrhofer, Harald Radi, and Alois Ferscha. Recognizing and predicting context by learning from user behavior. *Radiomatics: Journal of Communication Engineering, special issue on Advances in Mobile Multimedia*, 1(1):30–42, May 2004.

[17] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10 –17, August 2001.

[18] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29:38–47, February 1996.

[19] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. Trbac: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4:191–233, August 2001.

[20] Michael J. Covington, Wende Long, Srividhya Srinivasan, Anind K. Dev, Mustaque Ahamad, and Gregory D. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the sixth ACM symposium on Access control models and technologies*, SACMAT '01, pages 10–20, New York, NY, USA, 2001. ACM.

[21] M.J. Moyer and M. Abamad. Generalized role-based access control. In *Distributed Computing Systems, 2001. 21st International Conference on.*, pages 391 –398, April 2001.

[22] Guangsen Zhang and Manish Parashar. Dynamic context-aware access control for grid applications. In *Proceedings of the 4th International Workshop on Grid Computing*, GRID '03, pages 101–, Washington, DC, USA, 2003. IEEE Computer Society.

[23] Jalal Al-Muhtadi, Anand Ranganathan, Roy Campbell, and 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, pages 489–, Washington, DC, USA, 2003. IEEE Computer Society.

[24] Junzhe Hu and Alfred C. Weaver. A dynamic context-aware security infrastructure for distributed healthcare applications. In *5th Workshop on Pervasive Security Privacy and Trust (PSPT)*, MA, Boston, 2004.

[25] Seon-Ho Park, Young-Ju Han, and Tai-Myoung Chung. Context-aware security management system for pervasive computing environment. In *Proceedings of the 6th international and interdisciplinary conference on Modeling and using context*, CONTEXT'07, pages 384–396, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] K. Nishiki and E. Tanaka. Authentication and access control agent framework for context-aware services. In *Applications and the Internet Workshops, 2005. Saint Workshops 2005. The 2005 Symposium on*, pages 200 – 203, January 2005.

[27] Vassiliki Koufi and George Vassilacopoulos. Context-aware access control for pervasive access to process-based healthcare systems. In *MIE*, pages 679–684, 2008.

[28] Chun-Dong WANG, Li-Chun FENG, and Qiang WANG. Zero-knowledge-based user authentication technique in context-aware system. In *Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering*, MUE '07, pages 874–879, Washington, DC, USA, 2007. IEEE Computer Society.

[29] Mary Thompson, William Johnston, Srilekha Mudumbai, Gary Hoo, Keith Jackson, and Abdelilah Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8*, pages 17–17, Berkeley, CA, USA, 1999. USENIX Association.

[30] Introduction to public key cryptography. http://docs.sun.com/source/816-6154-10/contents.htm#1041271, October 1998.

[31] Public key cryptography. http://en.wikipedia.org/wiki/Public-key_cryptography, November 2010.

[32] Gustavus J. Simmons. Symmetric and asymmetric encryption. *ACM Comput. Surv.*, 11:305–330, December 1979.

[33] Digital signature. http://en.wikipedia.org/wiki/Digital_signature, November 2010.

[34] Jim DeRoest. Certificate based authentication. Sun Expert Magazine: Jun. 1997, pp. 87-89.

[35] Java 1.6 simpledateformat class date time patterns. http://download.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html#number.

APPENDIX A

JAVA DATE TIME FORMATTER PATTERNS

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General Time Zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC822 Time Zone	-800

Pattern letters are usually repeated, as their number determines the exact presentation:

Text: For formatting, if the number of pattern letters is 4 or more, the full form is used; otherwise a short or abbreviated form is used if available. For parsing, both forms are accepted, independent of the number of pattern letters.

Number: For formatting, the number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount. For parsing, the number of pattern letters is

ignored unless it's needed to separate two adjacent fields.

Year: If the formatter's Calendar is the Gregorian calendar, the following rules are applied.

For formatting, if the number of pattern letters is 2, the year is truncated to 2 digits; otherwise it is interpreted as a number.

For parsing, if the number of pattern letters is more than 2, the year is interpreted literally, regardless of the number of digits. So using the pattern 'MM/dd/yyyy', '01/11/12' parses to Jan 11, 12 A.D.

For parsing with the abbreviated year pattern ('y' or 'yy'), SimpleDateFormat must interpret the abbreviated year relative to some century. It does this by adjusting dates to be within 80 years before and 20 years after the time the SimpleDateFormat instance is created. For example, using a pattern of 'MM/dd/yy' and a SimpleDateFormat instance created on Jan 1, 1997, the string '01/11/12' would be interpreted as Jan 11, 2012 while the string '05/04/64' would be interpreted as May 4, 1964. During parsing, only strings consisting of exactly two digits, as defined by Character.isDigit(char), will be parsed into the default century. Any other numeric string, such as a one digit string, a three or more digit string, or a two digit string that isn't all digits (for example, '-1'), is interpreted literally. So '01/02/3' or '01/02/003' are parsed, using the same pattern, as Jan 2, 3 AD. Likewise, '01/02/-3' is parsed as Jan 2, 4 BC.

Otherwise, calendar system specific forms are applied. For both formatting and parsing, if the number of pattern letters is 4 or more, a calendar specific long form is used. Otherwise, a calendar specific short or abbreviated form is used.

Month: If the number of pattern letters is 3 or more, the month is interpreted as text; otherwise, it is interpreted as a number.

General time zone: Time zones are interpreted as text if they have names. For time zones representing a GMT offset value, the following syntax is used:

GMTOffsetTimeZone: GMT Sign Hours : Minutes
Sign: one of + -
Hours: Digit — Digit Digit
Minutes: Digit Digit
Digit: one of 0 1 2 3 4 5 6 7 8 9

Hours must be between 0 and 23, and Minutes must be between 00 and 59. The format is locale independent and digits must be taken from the Basic Latin block of the Unicode standard.

For parsing, RFC 822 time zones are also accepted.

RFC 822 time zone: For formatting, the RFC 822 4-digit time zone format is used:

RFC822TimeZone: Sign TwoDigitHours Minutes
TwoDigitHours: Digit Digit

TwoDigitHours must be between 00 and 23. Other definitions are as for general time zones.

For parsing, general time zones are also accepted.

Examples

Date and Time Pattern	Result
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 AD at 12:08:56 PDT
"EEE, MMM d, ''yy"	Wed, Jul 4, '01
"h:mm a"	18:08 PM
"hh 'o''clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700
"yyyy-MM-dd'T'HH:mm:ss.SSSZ"	2001-07-04T12:08:56.235-0700