MASTER'S THESIS

Context Aware
Security Policy Enforcement
CASPER

by
Yonggang Han

Supervisors:
Dr. E.P.de Vink (TU/e)
Ir. F.Fransen (TNO-ICT)

Eindhoven Oct. 2005
Acknowledgement

During this project, many people gave me kind help and supports, such as my supervisors, my friends in TNO-ICT Groningen and in TU/e, and my family, I would like to thank all of them.

I appreciate ir.Frank Fransen for giving me such an opportunity to do research in this new field. He spent lots of time on discussing the project with me and gave me so much valuable instructions and advices. He is really a wise mentor and kind friend. It’s is one of the most important experience for me to work with Frank and other colleagues in TNO-ICT, Groningen.

I am really grateful to dr. Erik de Vink for his patient instructions on my project and thesis. His precise research attitude and constructive opinions help me improving my thesis in many aspects. He also impressed me within last two year for his outstanding teaching work and kind help in Information Security Technology master program. I also want to thank dr. Reinder Bril for evaluating my thesis.

I was very luck to meet lots of new colleagues and friends: Paul de Jager, Maaike Blikslager, Richard Kerkdijk, Achteren van Leender, F.Mevissen, Dick Welfing and Fabian Walraren etc. in TNO-ICT. They gave me many helps and an inspiring work environment.

Last but not the least I want to thank my parents for their love and support during my study. Although far away from home, my family is always the most reliable support in the bottom of my heart. Special thanks to my girl friend for her love and company, it is her support and appreciation helped me getting through those the most difficult period, when we can only see each other at weekend.
Abstract

The increasing of mobile devices using heterogeneous networks makes it possible to provide new kind of ubiquitous application: context awareness. This kind of applications use current dynamic context information of users, devices and environment such as time, location, user’s event, network environment etc. to provide dynamic intellectualized applications.

The CASPER (Context Aware Security Policy Enforcement) project focus on introducing context awareness into security mechanism to let the enforcement of security policy dynamically adjusted according to the changes of context information, consequently providing dynamical, advanced and seamless security setting.

In the project, we survey the recent researches about context awareness and context-aware security. Based on the survey results, we analyze the requirements of CASPER system. Then we use the knowledge from analysis and up-to-date technologies to design the CASPER model. Furthermore, we apply this model on access control to let this security mechanism to be context aware.

We believe the CASPER model is valuable for providing new context-aware security applications in ubiquitous computing environment.
# Table of Contents

1 INTRODUCTION........................................................................................................................................... 1  
1.1 THE GOALS OF PROJECT .......................................................................................................................... 1  
1.2 THESIS STRUCTURE ............................................................................................................................... 2  
2 BACKGROUND AND STATE-OF-THE-ART ............................................................................................... 3  
  2.1 CONTEXT-AWARENESS ............................................................................................................................ 3  
  2.1.1 The definition of context ..................................................................................................................... 3  
  2.1.2 The definition of Context aware computing ...................................................................................... 4  
  2.1.3 Context Management ......................................................................................................................... 4  
  2.1.4 Context Ontology ............................................................................................................................ 5  
  2.1.5 Other aspects of context related to security ..................................................................................... 7  
  2.1.6 Survey on Context aware Systems ................................................................................................. 7  
  2.1.7 Conclusion ................................................................................................................................... 12  
  2.2 CONTEXT AWARENESS & SECURITY ................................................................................................. 12  
  2.2.1 The Security of Context-awareness System .................................................................................... 12  
  2.2.2 Context Aware Security .................................................................................................................. 13  
  2.2.3 Survey of context aware security .................................................................................................. 13  
  2.2.4 Conclusion ................................................................................................................................... 17  
  2.3 THE MOTIVATION OF CASPEr ............................................................................................................ 17  
3 INTRODUCTION TO CASPER ................................................................................................................ 19  
  3.1 WHAT IS CASPEr .................................................................................................................................. 19  
  3.2 CONTEXT-AWARE SECURITY SCENARIOS ......................................................................................... 20  
  3.2.1 Business Travel ............................................................................................................................... 20  
  3.2.2 Healthcare .................................................................................................................................... 21  
  3.3 CONTEXT AWARE SECURITY MECHANISMS .................................................................................... 22  
  3.3.1 Access control ................................................................................................................................. 22  
  3.3.2 Secure communication .................................................................................................................... 23  
  3.4 CASPEr REQUIREMENTS ..................................................................................................................... 23  
  3.4.1 Analysis of functional requirement ............................................................................................... 24  
  3.4.2 Analysis of non-functional requirement ....................................................................................... 25  
4 CASPER DESIGN ...................................................................................................................................... 26  
  4.1 GENERAL CASPER MODEL .................................................................................................................. 26  
  4.1.1 Motivations ..................................................................................................................................... 26  
  4.1.2 CASPER model and components ................................................................................................. 27  
  4.1.3 Policy decision making process ...................................................................................................... 31  
  4.1.4 Conclusion ................................................................................................................................... 34  
  4.2 ADAPT CASPER MODEL TO ACCESS CONTROL ........................................................................... 34  
  4.2.1 Traditional access control ............................................................................................................... 34  
  4.2.2 Introduce context awareness into access control .......................................................................... 38  
  4.2.3 Policy language candidates for CASPER ..................................................................................... 39  
  4.3 EXTENDING XACML FOR CASPER ................................................................................................. 47  
  4.3.1 CASPER access control policy language model - Solution 1 ........................................................ 48  
  4.3.2 CASPER access control policy language model - Solution 2 ....................................................... 50  
  4.3.3 Detail of the policy language .......................................................................................................... 51  
  4.4 EVALUATION PROCESS ..................................................................................................................... 55  
  4.4.1 Attribute evaluation ....................................................................................................................... 56
APPENDIX C XML SCHEMA FOR CONTEXT AWARE EXTENDED XACML

APPENDIX D OVERVIEW OF SAML

APPENDIX E XML SCHEMA EXTENSION OF SAML
List of Figures

Figure 1-1 Common application and Context awareness application ................................................................. 1
Figure 2-1 Common architecture of context management ....................................................................................... 4
Figure 2-2 Semantic Web Stack and Ontology [Den00] .............................................................................................. 5
Figure 2-3 Standard Ontology for Ubiquitous and Pervasive Applications [CPJ04] .................................................. 6
Figure 2-4 The SOUPA policy ontology [CPJ04] ...................................................................................................... 7
Figure 2-5 Context Toolkit components [Dey00] ...................................................................................................... 8
Figure 2-6 Context Toolkit Architecture [Che03, 04] .................................................................................................. 9
Figure 2-7 Context Broker Architecture [Che03, 04] ........................................................................................... 10
Figure 2-8 CoBrA implementation in EasyMeeting demo [Che04] ........................................................................ 10
Figure 2-9 Context Management Subsystem Architecture [Jia05] ........................................................................ 10
Figure 2-10 Threats for context aware system ........................................................................................................ 13
Figure 2-11 CAAC Security Infrastructure [Hu04] ................................................................................................. 15
Figure 2-12 Policy based active CAS system architecture [Jea03] ......................................................................... 16
Figure 2-13 TEANU scenarios [Jea03] .................................................................................................................. 16
Figure 3-1 The concept of CASPER ....................................................................................................................... 19
Figure 4-1 Context Aware Security Policy Enforcement model ........................................................................... 27
Figure 4-2 Tasks of PDP and PAP .......................................................................................................................... 28
Figure 4-3 Tasks of Context Broker ...................................................................................................................... 29
Figure 4-4 CASPER Policy Decision Making Process Sequence Diagram .......................................................... 30
Figure 4-5 PEP-PDP interaction structure ........................................................................................................... 32
Figure 4-6 Interaction between PDP-PAP and PDP-Finder .................................................................................. 32
Figure 4-7 Policy Decision Information collecting process ............................................................................... 33
Figure 4-8 ISO/IEC 10181-3 access control framework ..................................................................................... 35
Figure 4-9 Mapping between CASPER access control model and ISO/IEC framework ..................................... 38
Figure 4-10 WS-Policy Model ............................................................................................................................... 41
Figure 4-11 XACML context [XACML] .................................................................................................................. 42
Figure 4-12 XACML Policy Language Model [XACML] ....................................................................................... 47
Figure 4-13 Context information component and context types ........................................................................ 49
Figure 4-14 Context information component and context types in language model ........................................... 50
Figure 4-15 Main elements of “Rule” .................................................................................................................... 51
Figure 4-16 Main elements of “Policy” .................................................................................................................. 53
Figure 4-17 Main elements of “PolicySet” ................................................................................................................. 54
Figure 4-18 XACML Request Message ................................................................................................................. 55
Figure 4-19 XACML Response Message .............................................................................................................. 55
List of Acronyms and Terminology

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>ADI</td>
<td>Access Control Decision Information</td>
</tr>
<tr>
<td>ADF</td>
<td>Access Control Decision Function</td>
</tr>
<tr>
<td>AEF</td>
<td>Access Control Enforcement Function</td>
</tr>
<tr>
<td>CASPEr</td>
<td>Context Aware Security Policy Enforcement</td>
</tr>
<tr>
<td>CB</td>
<td>Context Broker</td>
</tr>
<tr>
<td>CI</td>
<td>Context Information</td>
</tr>
<tr>
<td>CIP</td>
<td>Context Information Provider</td>
</tr>
<tr>
<td>CMS</td>
<td>Context Management Subsystem</td>
</tr>
<tr>
<td>CoBrA</td>
<td>Context Broker Architecture</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>PAP</td>
<td>Policy Administrator Point</td>
</tr>
<tr>
<td>PDP</td>
<td>Policy Decision Point</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy Enforcement Point</td>
</tr>
<tr>
<td>PDI</td>
<td>Policy Decision Information</td>
</tr>
<tr>
<td>P3P</td>
<td>W3C Platform for Privacy Preferences</td>
</tr>
<tr>
<td>RBAC</td>
<td>Role Based Access Control</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SOUPA</td>
<td>Standard Ontology for Ubiquitous and Pervasive Applications</td>
</tr>
<tr>
<td>WSPL</td>
<td>Web Service Policy Language</td>
</tr>
<tr>
<td>WS-Policy</td>
<td>Web Service Policy</td>
</tr>
<tr>
<td>XACML</td>
<td>eXtensible Access Control Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 The Goals of Project

The rapid development of the heterogeneous communication network let more and more users under ubiquitous computing environments. Using the (mobile) computing devices will be as using other common tools such as pen and notepad in our daily life. Computing service will be anywhere and anytime. In the near future, more and more smart devices should be able to react to the changes of the various environment factors (contexts). These contexts, such as location, time, network environment, computing environment and user’s event will be continuously changed under dynamic environment. In order to handle changes and perform auto-configuration within ubiquitous computing environments, context awareness was introduced. More and more applications will use the context information such as location, time, role, temperature etc. to characterize the situation of entity (e.g. user, device), and perform some adaptive actions. There are some examples of context awareness application, the most prevalent one is car navigator - a location based service. The car navigator utilizes the GPS position service and surrounding traffic information to provide appropriate travel route. In this application, the information of location, traffic of context will affect user’s car navigator to make the adaptive route selection. The general definition of context and context awareness can be found in section 2.1.

![Diagram of traditional and context-aware applications](image)

**Figure 1-1 Common application and Context awareness application**

In the ubiquitous computing environment, security is crucial to many applications. It is necessary and possible to make the security of an application adapt to the changes of environment (context) of the user, device or application. An example of this concept is displaying sensitive information within a public place. The mobile device user is able to access all the data in his/her mobile device within the company (trusted area). However, according to the requirements of company’s security policy, the confidential document can not be displayed within public place (hostile area). The context information such as location of the device is required for automatically enforcing the security policy. Compared to the traditional static security policy enforcement methods, the mechanism we just mentioned has several advantages:

- The security configuration can adapt to new environment (context).
- The security setting process is transparent to user. User is not bothered by applying security mechanisms before or during his/her activities.
The security configuration will become more seamless and simpler to common user who is not the expert on computer science and information security.

The goal of this thesis is to develop a **Context Aware Security Policy Enforcement (CASPeR)** mechanism for innovative applications within ubiquitous computing environment. A generic architecture for CASPeR is designed within the project, along with a flexible method for describing context aware security policy. The work mainly focuses on context aware access control.

### 1.2 Thesis Structure

This thesis is outlined in 5 chapters. The first two chapters give an introduction and background of the context awareness and context aware security. The chapter 2 “Background and Stat-of-the-art” is based on the survey of recently active projects and research papers. So we could have a view and analysis of the context awareness applications and relevant security issues.

In chapter 3, we provide two scenarios that have close relation to context-aware security issues. One is “business travel”, another is “healthcare”. The story line, context information category and the relevant security mechanisms are presented. From the two scenarios, we can find out why context awareness and context-aware security is useful and significant to our life. We give a summary and analysis of CASPeR system based on them, and then the requirements of CASPeR are identified.

The CASPeR architecture design is represented in chapter 4. We provide a generic CASPeR model, and then adapt the model on access control mechanism to make it to be context aware. Moreover, we provide a solution for context aware access control policy language model. We also provide the blueprint for how to implement CASPeR access control and a threats model in the appendix. At the end of the thesis, my contributions in CASPeR project are summarized and future works are recommended.
2 Background and State-of-the-art

In ubiquitous computing environments, where situations of the user and the devices change frequently, it is desired to have mechanisms that are able to adapt to these changes. This will provide users with more seamless, easy use services. The paradigm of tailoring applications, services, communication and connectivity to the user’s current situation and needs, is referred to as context awareness.

The goal of CASPEr is to develop an architecture that enables the enforcement of security policy to be adapted to context information. The focus will be on the security mechanisms access control and privacy protection.

Before we start the CASPEr design, this chapter provides background and state-of-the-art survey of context awareness concepts, technologies, applications and architectures. We also address two aspects of context awareness and security, and the state-of-the-art of context aware security researches.

2.1 Context-Awareness

2.1.1 The definition of context

Within the ubiquitous computing domain, there are several various definitions of context have been put forward. The generic and reasonable definition given by Dey and Abowd [Dey99] is nowadays, however, used as reference in most of literatures on context awareness:

“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 application themselves.”

However, the definition in [Dey99] is focus on user. Except this definition, we also need to consider the context relate to device and environment in CASPEr. Dey and Abowd also give four primary context categories:

- Identity (who’s);
- Location (where’s);
- Time (when’s);
- Activity (what’s).

Examples of context information are location, temperature, the device type, the nearby resources (e.g. a printer or display), the available networks, the state of a workflow process, and the capabilities of a device. Context of an entity thus includes a lot of different types of information. Several categorizations have been proposed to structure context information. Often they are oriented on a particular application domain. An example of generic categorizations is given in [Che00]:

- **Computing context** - deals with any kind of context information related to a computing system. Computer CPU, network, IP address, status of a workflow, etc are example of computing system context.
- **User context** - refers to any kind of context information related to the user. User context information can be user’s age, user profile, location, medical history, etc. User context can be also related to his emotions [Pic03], even if this is difficult to capture in a computing system.
- **Physical context** - consist in any kind of context information related to the physical environment, in opposition to the computing environment. Physical context information is lighting, temperature, noise level, weather, etc.
- **Temporal context** - defines any kind of context information related to time. Time of day, date, and season are typical temporal context information.
2.1.2 The definition of Context aware computing

Dey and Abowd define context awareness in [Dey99] as follows:

“A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task.”

There are three categories of context aware functions, related to the presentation of information, the execution of services, and the storage of context information attached to other captured information for later retrieval [Dey00]. The first category: Presenting information and services, refers to applications that either present context information to user, or use context to propose selections of actions to user. The second category: Automatically executing a service, describes applications that trigger a command, or re-configure the system on behalf of the user according to context changes. The third category: Attaching context information for later retrieval, applications tag captured data with relevant context information.

Within many research projects context-aware applications as meant within this definition have been developed and demonstrated. There are, however, not many real context-aware applications available to users yet. At the beginning, most context aware projects and applications focus on using location information, such as the “Shopping Assistant [AGK94], “Cyberguide [LKA96, AAH 97]” and “Location-aware Information Delivery [MS00].”

Lately more and more research involves using other types of context information to provide added value to users, like the “Adaptive GSM phone and PDA [SAT 99],” use the context information of user’s activity, light level, pressure, and proximity of other people. In the PDA scenario, the font size is adapted to the user’s activity (a large font when the user is walking, small font when stationary) as well as to environmental conditions (such as light level).

2.1.3 Context Management

In order to make a system to become context aware, it needs to be able to handles context information. In addition acquires context, assesses context, and performs action according to assessment results. This is commonly referred to as context management. Context management consists of all the following steps:

- **Context acquisition** - During runtime of a context-aware system, context acquisition is necessary to compile the current context. This is a continuous process that has to be separated every time context changes and when the device is turned on. Thereby, all context information of the set is acquired separately and provided to the assessment entity [Pax99, Pax99].
- **Context assessment** - The result of context acquisition of the previous step describes a current context uniquely. Context assessment now compiles and decides what action has to be performed according to the context acquired. This task is triggered by a new context acquired and decision is made according to a set of rules – the policy.
- **Performing an action** - The action chosen by context assessment is performed.
2.1.4 Context Ontology

In context aware system, the context is needed to define, store and represent in a machine understandable form. Ontology is used to identify objects by classifying them and characterizing them with properties. In the scope of context-aware system, context ontology classifies context information and establishes relationships between them. The Web Ontology Language (OWL) is a Semantic Web language for use by applications that need to process the content of information instead of just presenting information to humans. This language is developed in part of the Semantic Web initiatives sponsored by World Wide Web Consortium (W3C). Based on the XML standard, OWL is a knowledge representation language for defining and instantiating ontology.

![Semantic Web Stack and Ontology](image)

Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) is a promising ontology which provides knowledge sharing, context information reasoning and inter-operability in ubiquitous environment. Expressed using OWL, SOUPA ontology consists of two set of ontology: SOUPA Core and SOUPA Extension. SOUPA core propose of vocabularies for the nine following concepts: person, agent, belief-desire-intention, action, policy, time, space and event. For each concept, SOUPA Core defines a typical vocabulary associated to the concept. For example, for the person concept, SOUPA Core defines attributes for a person such as first name, last name or contacts. SOUPA extension enables the definition of new set of vocabulary. Existing SOUPA extensions propose ontology for location, image capture or region connection calculus.

The SOUPA policy ontology (see Figure 2-3) is designed based on the following principle: policies are rules for regulating the permissions for computing entities to perform actions. In a pervasive computing environment, policies are defined by the human users to permit or forbid computing entities to perform different types of actions. As we have discussed in CoBrA system architecture section, the SOUPA policy ontology supports policy reasoning using description logic. So the policy reasoning implementation can be used on hand. But, it has many limitations, such as the SOUPA does not support the definition of variables and rule conditions that are logical expressions (e.g., conjunctions and disjunctions). Consequently, users can not define conditional policy rules, or define meta-policies.
Within the following CASPEr design sections, we try to provide a policy mechanism that is more flexible and fine-grained, which can provide more complex policy representation and policy combination. Details are described in chapter 4.
2.1.5 Other aspects of context related to security

When we take a close look at context information, many characteristics can be identified that have to be taken into account when developing a context-aware system. For the design of CASPEr the following characteristics are relevant.

- **Imperfection of context information** – Context information is imperfect, which means that the context information at the time of use is inaccurate, or even incorrect. This is for instance caused by the delay between the time of sensing and the time of use. In combination with the highly dynamic character of some types of context, at the time of use the context can already have changed. Another example is the fact that some sensors have a measuring error and thus provide raw sensor data with a certain inaccuracy. Also during the handling of context information the integrity could be affected, for instance due to a malicious act.

  Gray and Salber [Gra01] identified the following information quality attributes:

  - **coverage** – the amount of the potentially sensed context about which information is delivered
  - **resolution** – smallest perceivable element
  - **accuracy** – range in terms of a measure of the property
  - **repeatability** – stability of measure over time
  - **frequency** – sample rate; the temporal equivalent of resolution
  - **timeliness** – range of the measure in time; the range of error in terms of the time of some phenomena; the temporal equivalent of accuracy

- **Representation** – Much of the context information within ubiquitous computing can be represented in different ways. For instance, a location can for instance be represented by latitude/longitude coordinates or by an address, building name and room number. Also different scales can be used to represent a value (e.g. metric system, British units). In [Hel02], requirements for the representations of context information are given. Context representation should be structured, composable/decomposable, interchangeable, uniform, extensible, and standardized.

- **Higher level context** – A higher context is context information that is derived as a result of the merger of other context information. Higher context can also be derived as a result from interpretation of context information to raise the level of abstraction.

- **Historical context** – Historical context information is context store over a time span. Based on historical context it would for instance be possible to predict user behavior in a particular context. It is, however, necessary to keep in mind that storing all possible context information of an entity would be inefficient. The historical context required by the application will therefore be defined up front. Also the history of context could be used to estimate the quality of some context information.

2.1.6 Survey on Context awareness Systems

The survey on following typical context awareness systems will give you an overview of the state-of-the-art on context-aware systems. We will first give an overview of the three context-aware systems, and then a comparison and conclusion.

- **Context Toolkit**

  Context Toolkit (http://www.cs.berkeley.edu/~dey/context.html#overview) that developed by Anind K.Dey and Gregory D.Abouwd is an earlier toolkit for context-aware applications. The basic components of this architecture are context widgets, interpreters, aggregators and the discoverer (are shown in figure 2-2).
- **Context widgets** are software components that provide applications with access to context information while hiding the details of context sensing.
- **Interpreters** may combine multiple pieces of context to produce higher-level context information.
- **Aggregators** gather context information related to an entity for easy access by applications.
- **Discoverers** allow applications (and other components) to determine the capabilities of the environment and to take advantage of them.

![Figure 2-5 Context Toolkit components](image)

The interaction relation between Context Toolkit components are shown in Figure 2-6. Widgets are either queried or use notification to inform their clients (applications, aggregators or other widgets) of changes. Aggregators in turn act as a bridge between widgets and applications. Interpreters can be solicited at any stage and by any widget, aggregator or application. Services are triggered primarily by applications. Discoverers communicate with all the components, acquiring information from widgets, interpreters, and aggregators, and providing this information to applications.

![Figure 2-6 Context Toolkit Architecture](image)

The Context Toolkit provides the services of: encapsulation of sensors, access to context data through a network API, abstraction of context data through interpreters, sharing of context data through a distributed infrastructure, storage of context data, and basic access control for privacy protection.

All the Context Toolkit components are independently from applications, allowing for the constant acquisition of context and use by multiple applications. So the usage of components and their communication protocols and languages are decided by implementing developer, depends on the system purpose. The context toolkit implements the main functions of context management.
• **Context Broker Architecture (CoBrA)**

The **Context Broker Architecture**, so-called CoBrA, is an agent based architecture for supporting context-aware systems in smart spaces (e.g., intelligent meeting rooms, smart homes, and smart vehicles). CoBrA provides the function of discovery, acquisition and reasoning context information.

The most significant character of CoBrA is that it uses Semantic Web standard (W3C Web Ontology Language OWL) to define ontologies of context (e.g. people, agents, devices, events, time, space, etc.). In the traditional context aware systems, most of them use the data structure (such as Java class) to modeling context information. But the method often tight couples the context information representation to the specific system implementation. The method can not effectively support context knowledge sharing and reasoning. It will limit the interoperability between different components and different systems. Via semantic web ontology, it is able to providing an explicit representation of contexts for reasoning and knowledge sharing.

CoBrA also provides a resource-rich context broker to maintain a shared model of context for all computing entities. Moreover, it allows the users to define privacy policy to control the sharing and the use of their situational information. CoBrA adopts policy-based approach to protect privacy, and they use SOUPA (see section 2.1.4) policy ontology to define the privacy policies. CoBrA allows users to control the granularity of the information that a context broker can share with other agents. The SOUPA policy ontology supports policy reasoning using description logic. So the policy reasoning implementation can be used on hand. However, it has many limitations, such as the SOUPA does not support the definition of variables and rule conditions that are logical expressions (e.g., conjunctions and disjunctions). Consequently, users can not define conditional policy rules, or define meta-policies.

![Figure 2-7 Context Broker Architecture](Che03_04)
In [Str05], M.Strimpakou and I.Roussaki et al describe a Context-Aware Enabling System, also named Context Management Subsystem (CMS). This is depicted in figure 2-9. The CMS establishes the context-awareness related functionality in the IST DAIDALOS [Far04] pervasive services platform. It provides users a uniform way to access context information, thus hiding the complexity of context management. It offers streamlined mechanisms for large-scale distribution and synchronization of personal repositories across multiple administrative domains.

**Context Management Subsystem (CMS)**

In [Str05], M.Strimpakou and I.Roussaki et al describe a Context-Aware Enabling System, also named Context Management Subsystem (CMS). This is depicted in figure 2-9. The CMS establishes the context-awareness related functionality in the IST DAIDALOS [Far04] pervasive services platform. It provides users a uniform way to access context information, thus hiding the complexity of context management. It offers streamlined mechanisms for large-scale distribution and synchronization of personal repositories across multiple administrative domains.
The Context Management Subsystem responsible for the following:
- Acquisition of raw context data from network resources;
- Inference of additional and not directly observable context information;
- Control of access to the context data;
- Sharing and manipulation of context data.

The Context Broker (CoB) answers context requests and serves as the access point to the context data. When context consumers request context from the broker, a negotiation, which based on the requester’s requirements (e.g. type, accuracy, cost, time of update, priorities), the context owner’s authorization setting (e.g. privacy), and the available context and its quality parameters, will be processed. According to the decision, the appropriate context will be retrieved from multiple sources and providers, such as local context store, inference engine, peer CoBs or other appropriate context provider.

The Inference Engine (IE) infers additional and not directly observable context information, which have high added value for both consumers and providers. A chain of components is set up to allow the enrichment of context, by combining, converting or refining context. A typical example of inference is that of estimating a person’s activity from some lower level sensor data.

The Sensor Manager (SM) keeps track of the sensors which are associated with or attached to the local host, and updates new sample data in context system.

The Context Store retrieves context from the SM (raw data) and the IE (inferred data), processes the data, stores it and updates the context databases. Context Store provides context (context system’s uniform format) to the CoB on demand.

As shown in Figure 2-6, there are two main interfaces are offered to actors or other entities outside the CMS by the CoB: the User Context Management Interface and the Enabling Services Interface. The User Context Management Interface allows a context consumer to configure any data previously defined. The Enabling Services Interface is offered to the rest of the components in the pervasive system architecture in order to support and control access to the context data maintained by the platform.

We compare these context awareness systems in the following aspects that we are concerned about, include context modeling, context reasoning, development toolkit and security. This is given in table 1.

**Table 2-1 Compare several context awareness systems**

<table>
<thead>
<tr>
<th>Context Toolkit</th>
<th>Context modeling or Ontology</th>
<th>Context reasoning</th>
<th>Development toolkit or Prototype</th>
<th>Security consideration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Toolkit</td>
<td>Attribute-value-tuple, encoded in XML</td>
<td>Context Interpretation: low level context to high level context</td>
<td>Context Toolkit</td>
<td>Simple access control and privacy.</td>
</tr>
<tr>
<td>CoBrA</td>
<td>OWL</td>
<td>OWL &amp; Assumption-based Reasoning</td>
<td>CoBrA Demo Toolkit</td>
<td>Privacy</td>
</tr>
<tr>
<td>Context Management Subsystem</td>
<td>unknown</td>
<td>Context Inference</td>
<td>Prototype implementation on OSGi Service Platform.(Simplified version of CMS)</td>
<td>Use other security management component in DAIDALOS</td>
</tr>
</tbody>
</table>
Look into these several context-awareness architectures, you will find out that more and more attention is paid on this research area. However, there still are many fields need more research such as ontology and security consideration.

When we implement the CASPeR system, the toolkit such as Context Toolkit or CoBrA Demo Toolkit are possible component that can be embedded into CASPeR. We believe Context Toolkit is more appropriate for research work due to its general usage. The CoBrA is more suitable for specific application such as easy meeting.

2.1.7 Conclusion

From the state of the art survey on context awareness, we could find out that the context awareness is a very promising technology within the future ubiquitous computing environment. It is a reasonable trend that more and more service infrastructures and mobile devices will be context aware.

Moreover, based on the survey of context awareness, we can see that there are several aspects we should take into account for CASPeR design. Firstly, the context category should be extensible in policy description, and the acquisition of context should according to the specific application. Secondly, the representation and context level should be standardized in context aware security policy. So we need a component that can merge, reason and abstract origin context to higher level context information, which can be used for evaluate security policy decision. Thirdly, the imperfection of context information is another important issue for security system. The correctness of context will affect the policy decision of the CASPeR, and the manipulation of context information may compromise the system directly. Consequently, there still are many aspects and problems that need to be studied further, e.g. the quality of context, context ontology and security.

2.2 Context Awareness & Security

When we consider the security issues of context awareness, we identify two points of view: security of Context Awareness, and Context-aware security. The two concepts are different, but they have close relation to each other.

2.2.1 The Security of Context-awareness System

The use of context information in context awareness system may introduce new vulnerabilities and exposures, which must be taken into account. So the context information should be reliable and integrated. Moreover, the context information always relates to user’s privacy, such as user’s location, preference and health status.

The context awareness systems need to satisfy following common security requirements: confidential, integrity and availability. The confidentiality, integrity and availability of the context information need to be ensured during all the phase of the context management to a predefined require level.

In most of context awareness researches, the main security threat in context-aware environments is basically the violation of the user privacy. Indeed, contextual information is provided by many sensors that can be invisible to the users. It is obvious that these sensors, gathering information about people without being noticed, can be a threat to privacy.

Furthermore, reasoning about context information raises other security challenge. Indeed, reasoning may get some context information for processing other context information. For example, from the patient’s heart rate and blood pressure, we can evaluate the patient’s health condition. Assuming that patient’s heart rate and blood pressure has different level of trust and security, so it will affect the trust level and security about the patient’s health condition. The correctness of low level context will affect the high level context directly.
2.2.2 Context Aware Security

With context-aware security we mean that enforcement of a security mechanism is dynamically adjusted to an entity's context, were an entity can be a person, place, or physical or computational object that is considered relevant to the enforcement of the security mechanism. The ability to use context information to extend or replace traditional user credentials for security purposes is beneficial for making security less intrusive. It can provide seamless security setting, to make the process easier to user (in most of cases, the user is not the expert with computer and security).

In nature, there are several aspects need to be considered: such as access control, communication and privacy. Let’s consider a healthcare scenario: the patient’s medical record could only be accessed by patient’s attending physicians within hospital in common situation. Vice versa, the record should be able to access by any physicians under the emergency situation. The scenario refers to both privacy and access control. The context information in this example is the role of the entity requesting access, and patient health status.

Several research papers haven been published on this topic. Most of them focus on context aware access control. In the following section, an overview is given of some of these papers.

2.2.3 Survey of context aware security

The survey in this section will illustrate some context-aware security systems. Some kinds of context information have been used in these systems to help making security configuration purposes. A major part of them are focus on access control, e.g. GRBAC, Gaia and CAAC; others are addressed various aspects such as communication (TEANU) and trust (LOCAS). They will be discussed as following.

- GRBAC
Other research papers on context aware access control are as following: Covington et al. [Cov02] propose the Generalized RBAC model. This is an extension to the traditional Role-Based Access Control (RBAC). In RBAC, permissions are associated to subjects’ roles (i.e. roles associated to entity requesting access to a certain object). In Generalized RBAC the concept of roles not only applies to subjects, but also to objects and system states. These new type of roles are called environment role. GRBAC allows defining new types of roles which are more flexible and easy to understand which a deep knowledge of the underlying computing system. An implementation, so-called CASA, of the environmental roles has been developed.

Covington et al [Cov00], [Cov01] defines their own policy management tool for the policy specification and context representation. This tool is using a graphical representation of XML encoded rules to specify access policies, role definitions and relationships. Their security policy is only focused on the access control policy. This access control model removes the subject centric limitation, by adding the possibility to define the policy from a subject-centric, object-centric, and environment-centric perspective. The notion of environment role allows the policy administrator to add roles to the surrounding environment of the system using a set of contextual information. For example, GRBAC enable to define an environment role corresponding to each day of the week.

- **Gaia**

Roman et al [Rom02] define a generic computational environment that captures physical spaces into a responsive context-base software architecture so-called Gaia. A physical space is a geographic region with limited and well defined physical boundaries, containing physical objects, heterogeneous networked devices, and users performing a range of activities. Derived from the physical space, Active Space proposes to the user a computing representation of physical space. Active Space helps the user to interact with the physical space. In the scope of Gaia [Jal03], Al-Muthadi et al define a framework for context-aware identification, authentication and access control and reasoning about context, so-called Cerberus. Based on Kerberos authentication [Nee94], and as Gaia is a user-centric architecture, Cerberus targets user’s identification via user’s context information such as fingerprint, voice and face recognition.

- **CAAC**

Junzhe Hu et al [Hu04] describe an extended RBAC – CAAC (Context-Aware Access Control) by associating access permissions with context-related constraints. Every constraint is evaluated dynamically against the current context of the access request. So the model is capable of making authorization decisions based upon context information in addition to roles. Moreover, in their design, administrators have great flexibility to specify complex context-aware authorization policies. The authorization engine can enforce any context-aware policy automatically because it is not statically bound to any application. And every context type definition and context implementation is independent of the specification of the access rules. The details of the system will be described following.

Within the Junzhe Hu’s paper[Hu04], they propose a context aware access control architecture for health care web application. In general, the access control policy is based on the definition of a set constraint related to a tuple <U, A, O>. The tuple is composed by user U, object O, and action A that can be performed. When we apply it to context aware system, the extra context constrain C should be added to tuple. In order to make the context information be used in make the access decision, they give four definitions:

- **Context Type:**
  It can be seen as context category of application. There are five primitive context types: Time, Location, User ID, Object Type and Object ID.

- **Context Constraint:**
  It is defined as a regular expression based on logical operator, which is capable of specifying any complex context related constraint to describe security requirement.

- **Authorization Policy:**
It is defined as $AP = (S, P, C)$ where $S$ is subject, $P = <M, O>$ is the target permission, and $C$ is context constraint. If $C$ is empty, then the policy reverts to RBAC.

- **Data Access:**
  It is defined as $DA = (U, P, RC)$ where $U$ is a user, $P$ is the permission this user want to acquire, and $RC$ (runtime context) is a set of values for every context type of the application.

The authors give the algorithm for context evaluation. Only when the context constrains $C$ evaluated to true under runtime context $RC$, the data access is granted. Figure 2-11 is the security infrastructure of CAAC, it is used to establish a healthcare web service, where physicians and patients can access the medical data via a web service portal. WS-Policy is used as access control policy specification language for CAAC. Furthermore, for using the context information, authors customize it to specify contextual constrains.

- **LOCAS**
  In [Zas02] Alexander Ng, Arkady Zaslavsky present a system, called Location Adaptive Security system (LOCAS), that attempts to adapt the level of security enforcement depending on the mobile client’s location and resource usage in a wireless network. The system distinguishes between “Trusted Zones” and “Untrusted Zones”. “Trusted Zones” can be generally categorized as networks that are enforced with tools such as firewalls, Intrusion Detection Systems, anti-virus scanners, etc. and provide protection for the mobile client. Typically, organizational wide WANs or office LANs fall into this category. An “Untrusted Zone” however, is a public network, such as a wireless ‘hotspot’ zone, where networking security policies are not strictly imposed. The mobile client’s is capable of determining whether it is in a trusted or untrusted zone. If the mobile client is in an untrusted zone additional security mechanisms will be enforced.

  The enforcement of security mechanisms can also be adapted to the status of the devices resources. For example, if the battery power is low, less power consuming encryption (e.g. use smaller key size) could be enforced. The author points out that choosing between full security or none at all, may not be the best solution.

- **TEANU**
  Several works on context aware policy-based security in the mobile networks also could be found in the literature. Recently, they were specifically focused on the next generation networks (NGN). NGN represent the integration of the 3G networks (GPRS/UMTS) with the WLAN. Jean et al [Jea03] propose a policy based context-aware service methodology; they choose the policy-based method because policies are ideal for context modeling. Their goal was to propose a solution to facilitate the provision of context aware services in a secure manner by implementing a policy-based network management (PBNM). This PBNM focuses more on the security mechanisms in the network layer than the application layer. The authors applied their security policy on a network-centric context-aware service called TEANU (Transparent Enterprise Access for Nomadic Users). This service offers to the user the possibility to access to their workstation from anywhere using the NGN technologies. In their scenario (Figure 2-12), the context aware service uses the location of the user and the
properties of the network to send the appropriate policy to the network domain (WLAN domain for example). Depending on the security policy, different strategies will be used to secure the communication between the user and his office (for example IPsec VPN while on the WLAN).

![Policy based active CAS system architecture](image)

**Figure 2-12 Policy based active CAS system architecture**

![TEANU scenarios](image)

**Figure 2-13 TEANU scenarios**

- **Other Systems**

  In [Rod04] a relevant example is presented: hospital workers are highly mobile; they are constantly changing location to perform their daily work, which includes visiting patients, locating resources, such as medical records, or consulting with other specialists. The information required by these specialists is highly dependent on their location. Access to a patient's laboratory results might be more relevant when the physician is near the patient's bed and not elsewhere.

  In [Wul04] context-aware authorization architecture is described designed to augment existing network security protocols in an Intranet environment. This authorization architecture, based on Kerberos authentication, enables
to activate or deactivate role assigned to a user depending on his context. For example, if a user in an un-secure place such as airport terminal, the access to sensitive data is denied whereas in the corporate building of the user, he has access to the confidential data.

Christine Julien et al developed Context-Sensitive Access Control for Open Mobile Agent Systems [Jul04]. The access control model is different from RBAC. The access control function is based on a security agent that mediates the access to the data residing on the device it is responsible for. To achieve a flexible access control, each agent defines a single access control function that takes some parameters:

- Credentials – allowing an agent, which is requesting access, to convey information about itself, e.g. Agent's id or a third party authentication.
- Operation – the access control function can allow one set of operations for some agents, but different operations from others.
- Pattern – the pattern provides information about an application’s prior knowledge of data. The security agent may allow access only to the agents that know the correct way to access data.

2.2.4 Conclusion

One the one hand, we should consider the security issues of context awareness system. Normally, the context awareness system has the general security problems as well as the common systems, such as authentication, authorization, confidentiality, integrity and non-repudiation. However, the use of context information will bring more vulnerabilities and exposure. So the protection of the context information is very important to whole system. Moreover, the compromise to context information also will lead privacy related problems. So we should pay more attention on the security issues when we design a context awareness system.

On the other, introducing context information into security mechanism gives us an opportunity to provide more advanced and dynamic means. The context awareness could become a new method for security configuration. It will let the security setting simpler and transparent to common user. However, it is still a new field in information security study.

2.3 The motivation of CASPEr

From the overview of the state-of-the-art in the previous sections we could find that context awareness give the system ability of taking advantage of context information (which characterize the situation of an entity) to provide adaptive services. We present several systems that are context aware. It is clear that in the past several years the researches and implementations of context awareness system have increased enormously. Other than the location based service, more and more context types are introduced in context awareness application.

Security issues are always important in pervasive computing systems. However, for the context awareness, security is still an area that needs more researches on it. Some context awareness systems have paid attention to security issues, most of which address on privacy.

Use context awareness as a method to enhance the security mechanism is really a new field. Several research projects try to provide adaptive security mechanism via context information, such as CAAC system. The context aware security will provide more seamless and advanced security mechanisms to user. Of course, the security of context information is very important in context aware security system, because the security mechanisms are dependent on context information.

Within the CASPEr project, we try to extend the state-of-the-art of context aware security by developing a policy based, context aware security model.
TNO research questions and refined questions

The research questions that are addressed by TNO within CASPEr project are:

- What is the State-of-the-art on Context Awareness for Security Enforcement? Note, this is broader than the concept and mechanisms that are actually called Context Aware, since this term is new to the security domain.
- What is the state-of-the-art on description language for security policies?
- What threats should a system that uses Context Aware Security Policy Enforcement have to be able to resist?
- What are (important) requirements for a system that uses Context Aware Security Policy?
- How can context information be trusted?

Based the survey and study of the current context-awareness, context-aware security researches and other assistant technologies, we refine the project research questions on the following aspects:

- What is the state-of-the-art of context awareness and context aware security?
- What is the state-of-the-art on description languages for security policies?
- What are the requirements for a context aware security policy enforcement system?
- How to adapt security policy language to context-aware security?
- What are the threats that a context aware security system should be able to resist?

Within the project, a generic architecture for Context-Aware Security Policy Enforcement needs to be designed along with a flexible method for describing context aware security policies. It aims to let the security configuration more seamless, more advanced and simpler to common users.
3 Introduction to CASPEr

3.1 What is CASPEr

Within the project, we define the Context Aware Security Policy Enforcement (CASPEr) as:

The enforcement of security policies is dynamically adjusted according to relevant context information. A system’s security policy consists of several rules, which are chosen due to a specific context of an entity; security decisions are made and enforced in context-aware method.

Figure 3-1 The concept of CASPEr

Via CASPEr, we try to provide more advanced security mechanisms under the ubiquitous computing environment within user’s daily working and living. As showed in the figure 3-1, we could introduce context information into security mechanism decision process. The security mechanism could be access control, secure communication, privacy protect, trust management etc. The context aware policy evaluation is against context information. For instance, an email encryption mechanism could make a decision to encrypt an email or not according to the current network environment and the confidential level of the email.

Except the value that bring by CASPEr, we also should note that CASPEr may introduce more risks. Context-aware security system need utilize various kinds of context information to provide context-aware security service, which may introduce more potential security vulnerabilities. The context information should be reliable and integrate. The threats may occur in several stages, such as acquiring and message transmit. The tamper of the context information will compromise the security policy decision.

Furthermore, how to protect user’s context information that collected by the application has become new kind of privacy protection problem. The disposal of the personal or corporation’s context information may cause the privacy relevant problem, even the law problems.

Two related scenarios are described in next section to present the usages of context aware security policy enforcement mechanism.
3.2 Context-aware Security Scenarios

3.2.1 Business Travel

The “business travel” scenario is presented following. The story line of the scenario is described firstly. The context aware security scenes are marked by the italics and the number with bracket. Moreover, the context aware security scenes are explained in more detail after the story.

“Hans is a technical director of an international telecom service and equipment provider corporation. They have many branches and R&D laboratories around world. So he needs to travel to different R&D center and client’s organization frequently.

After the Easter holiday, Hans is back to start the work of a new week. On 10:00 Am, Hans leaves his office and goes to meeting room. After he leaves, his computer is locked automatically. At the meeting, Hans and his colleagues inspect the milestone of the product development because the corporation has got some potential order and they want to launch the market as soon as possible. After meeting, Hans makes the decision, amended product plan, and send them to his boss and technical staff.

In the afternoon, Hans drives to another ABC laboratory in a university. He needs to discuss the newest research achievement with a professor. He could enter the lab freely and exchange the data with lab database with his laptop and PDA.

According to Hans’s agenda, an important client meeting is added into agenda. He was asked to meet with his boss and join an important client meeting. He will make a presentation on Wednesday morning. His digital assistant reserves a train ticket. Hans prefers the train so he can continue his work on the road. This point has been added to his profile in PDA. The digital assistant will make a decision according to the travel distance, agenda and Hans’ preference.

Hans reaches the train station at Tuesday noon, starts the trip to Germany. Hans opens the notebook to continue to prepare the meeting content. The notebook security setting will warn Hans that he is in a public environment, so he should pay more attention on the content that are showing on his notebook. When Hans arrives the Hanover, there are still several hours private time to visit the city. Hans goes to the mall to buy some gifts for his family. When he goes through the store, the merchandise information will be provided to Hans PDA in terms of his family’s preference.

After a good night sleep, Hans and his boss sit in front of their client. Hans make an attractive presentation about their new product. He also invites the professor from University via video conference to present the future plan of their innovation. Their clients are attracted by their plan, but they want to know more detail information. Since Hans don’t bring the materials with him, he accesses his corporation’s database and gets some up-to-date commercial information. The client is satisfied with the product’s new features, especially the new product plan and timeline. Now Hans could make relax and back to home happily.”

Now we will explain more about the context aware security scenes mentioned above:
[1]: After Hans leaves his office, the sensors could collect some context information (user’s identification token, user location) to determine that the user is without the control range of his/her device, so the user’s device will locked automatically.
[2]: The documents are classified into different confidential levels. When they are sent by email, the email system will decide using encryption or not according to the security level labels of the documents.
[3]: Via location context, the mobile device realizes it enter the partially trusted domain, it will apply role based access control and default of security setting of cooperating organization.
[4]: Han’s some personal data such as his shopping preference, his hobby and his personal agenda are kept secret to company. These personal contexts could be used in some context aware services, and the usage should be decided by Han’s privacy policies.
[5]: Within the public place, the mobile devices find they are under hostile network environment via location and network context. The firewall of Hans’ device is set in high secure level. Compare with within the company’s trusted environment, the different security setting are utilized.
[6]: User could control his privacy information to determine what kind of information could be access for context aware service (such as shopping service and tourist service) according to privacy policies.
[7]: Based on Hans’s important commercial visitor role, he can use partial resource of the client’s company, like printer, projector and wireless network.
[8]: Based on location and network context, secure mechanisms should be used to protect the communication channel, and the sensitive data are accessed based on user roles.

3.2.2 Healthcare

We will describe the “healthcare” scenario story firstly. The context aware security application scenes are marked by italic and the number with bracket. Furthermore, more details of the context aware security scenes are discussed.

“Bill gets heart disease in his sixties, so he often needs to be taken care by others. Bill buys a mobile healthcare assistant, which is a mobile device similar with PDA, however, with some sensors, healthcare functions, and be able to communicate with medical center. The healthcare assistant is able to collect Bill’s health status data such as heart rate, blood pressure, body temperature... via sensors. These data are continually monitored.

One day, Bill’s feel uncomfortable suddenly when he is walking out a supermarket. The healthcare assistant gives a warning and some advices to Bill. Bill follows the assistant’s instructions to take the medicine. Unfortunately, it doesn’t work. The health monitor data indicate that the situation may get worse. The healthcare assistant makes a judgment that Bill’s condition is very. Then the medical emergency center is automatically informed by the health assistant, and Bill’s location and the health status recorder are sent to the medical center immediately. Furthermore, the device broadcast the help message, which can be recognized by mobile device of people who hold physician role.

Maggie is a physician, she is going to buy some food after work. She receives an emergency message via her PDA, which says there is a patient who is suffering heart disease nearby. Maggie reaches Bill immediately, check his medical recorder within the health assistant and do some treatments according to the medical data[9]. After several minutes, Bill’s status become stable, and they have heard the ambulance is coming.

When Bill wakes up, he is laying in the sickroom, just recover from the event. His medical data is displayed on his attending physician’s computer[10]. With the professional medical care, Bill can have a good rest now.”

Now we will explain more about the context aware security scenes mentioned above:

[9] Health assistant adjust the access control policy of Bill’s medical recorder. Now anyone who has physician role and close to Bill can read Bill’s medical data.
[10] In normal situation only Bill’s attending physician can access Bill’s medical data. Other related staff like nurses only can access part of the medical recorder in their work time and within hospital.
3.3 Context aware security mechanisms

3.3.1 Access control

Description:
Access control is the set of mechanisms and processes that aim at protecting assets of a computer system from unauthorized entities. Access control determines whether an access request to a resource or data is granted or denied [Sam01]. A context-aware access control mechanism uses contextual information to grant or deny access to a certain object.

Based on the two scenarios mentioned above, several context-aware access control applications can be identified on context aware security points [3], [7], [9] and [10].

Context Information:
Within above application points, similar or different kinds of context information are adopted to help system make the access authorization decision.

[3]: The role of user; Location;
[7]: The role of user;
   Location;
   Activity
[9]: The role of user (requestor);
   Requestor and patient location;
   Patient’s health status (heart rate, blood pressure, body temperature…)
[10]: The role of user (requestor);
   Requestor and patient location;
   Patient’s health status;
   Time

Security Mechanism:

Table 3-1 Map Context information to access control

<table>
<thead>
<tr>
<th>Context Information</th>
<th>Policy Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>“Time&gt;=8:00” &amp; “Time&lt;=17:00”: allow read; append; delete; update</td>
</tr>
<tr>
<td></td>
<td>Time ≠ Work time: access deny</td>
</tr>
<tr>
<td>Role</td>
<td>Role &lt;= physician: access deny</td>
</tr>
<tr>
<td>(Role + Time)</td>
<td>Role = attending physician: allow read; append; update</td>
</tr>
<tr>
<td></td>
<td>“Role = manager” &amp; “Time = Work time”: allow read; append; delete; update</td>
</tr>
<tr>
<td>Location</td>
<td>Location = trusted area (within organization) : read, append, update</td>
</tr>
<tr>
<td></td>
<td>Location = hostile area (public place) : access deny</td>
</tr>
<tr>
<td>Role</td>
<td>Role = owner; <em>permit access (read) to personal medical recorder</em></td>
</tr>
<tr>
<td>Role + time</td>
<td>“Patient Status = normal” &amp; “Role &gt;= attend physician”: <em>permit access (read, append, update) to patient medical recorder</em></td>
</tr>
<tr>
<td>Role + activity (status)</td>
<td>“Patient Status = emergency” &amp; “Role &gt;= healthcare staff”: <em>permit access (read, append, update) to patient medical recorder</em></td>
</tr>
<tr>
<td></td>
<td>“Patient Status = emergency” &amp; “Role &gt;= healthcare staff” &amp; “Location = nearby”: <em>permit access to (read) patient medical recorder</em></td>
</tr>
</tbody>
</table>
There already are several research projects and mechanisms on context-aware access control, such as the GRBAC (General Role Based Access Control) concepts and CAAC (Context Aware Access control) architecture for health care web application. More information please see section 2.2.3.

Our research project will also address on this aspect, try to provide a policy based context-aware access control architecture.

3.3.2 Secure communication

Description:
All the communications related to sensitive content are required via secure connection. Especially the connections exceed the corporation’s perimeter. The high mobility is a distinct character of ubiquitous computing. Via context-aware mechanism, the enforcement of secure communication is based on context information (such as user’s location, network environment, mobile device computing capability, device’s battery level…).

Based on the two scenarios, several context-aware secure communication applications can be identified on context aware security points [2], [3], [5], [7] and [8].

Context Information:
Within above context-aware application points, various kinds of context information are adopted to help system to establish secure communication channel.

[2]: Security level label.
[3], [7]: Location; Role; Network type
[5], [8]: Location; Role; Network type; Within trust domain or not; Device computing ability

Security Mechanism:
Table 3-2 Map Context information to secure communication

<table>
<thead>
<tr>
<th>Context Information</th>
<th>Policy Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Type</td>
<td>“Network Type = 802.11” ∩ “CPU Type &gt; middle”: Using WPA</td>
</tr>
<tr>
<td></td>
<td>“Network Type = 802.11” ∩ “CPU Type &lt;= middle”: Using WEP</td>
</tr>
<tr>
<td>CPU Type</td>
<td>CPU Type &gt; middle: PKI</td>
</tr>
<tr>
<td>Battery Level</td>
<td>“Battery Level &gt;=middle”: using SSL connection; Encrypt email</td>
</tr>
<tr>
<td>Battery Level + Location</td>
<td>“Battery Level &lt;= low” ∩ “Location = hostile area”: No Encryption protect, No connection &amp; Can’t send email</td>
</tr>
</tbody>
</table>

3.4 CASPEr requirements

Based on the previous researches and according to the analysis of the context-aware security application scenarios, we identify the following context aware related requirements of CASPEr.
3.4.1 Analysis of functional requirement

The context-aware security policy enforcement system needs to adopt a set of context aware security policy. For using the policy on various platforms, and making the communication between the different platforms easily, the security policy should be described via fine grained, specific and standard policy language. Furthermore, the security policy should specify the method to describe the context information. The context type should be able to be extended and modified according to the specific application. Based on the above reasons, we identify the requirement $R_1$ as following:

- $R_1$ CASPEr should be able to adopt standard context-aware security policy language.
  - $R_{1.1}$ CASPEr should adopt standard security policy language.
  - $R_{1.2}$ CASPEr should specify standard input/output message format.
  - $R_{1.3}$ CASPEr should specify standard description method of context information.
  - $R_{1.4}$ CASPEr should specify the methods to modify and extend context type.

CASPEr should be able to acquire context information which is needed for the security policy evaluation. It can be fulfilled by communicating with a context broker, and acquire needed context information from it. The CASPEr should have function to find out what context information types are needed for the policy evaluation, and inquire it from Context Broker. The acquisition of raw context information and the reasoning of context information should be accomplished within Context Broker. This Context Broker may be part of the CASPEr, or generic service that is part of the middleware of the system. The corresponded requirements are described in $R_2$ as following:

- $R_2$ CASPEr should be able to acquiring context information from Context Broker.
  - $R_{2.1}$ CASPEr should be able to communicate with Context Broker.
  - $R_{2.2}$ Context Broker should be able to acquiring raw context information from sensor or context information provider.
  - $R_{2.3}$ Context Broker should be able to reasoning context information.

One of the key requirement of CASPEr is it has the evaluation engine to evaluate security policy against the input content and the context information. CASPEr policy evaluation process should be able to perform various kind evaluations such as arithmetic evaluation, match evaluation, expression evaluation etc. Furthermore, the policies should be able to be combined into policy set.

- $R_3$ CASPEr should be able to evaluate context-aware security policy against context information, and make a policy decision of how to enforce the security mechanisms.
  - $R_{3.1}$ CASPEr should be able to evaluate security policy against input and context information via different methods (match evaluation, expression evaluation, arithmetic evaluation etc.).
  - $R_{3.2}$ CASPEr should be able to combine a set of security policies.
  - $R_{3.3}$ CASPEr should be able to make an explicit policy decision.
  - $R_{3.4}$ CASPEr policy should be able to contain one or more obligations, which should be performed when the policy decision take effect.

Since the mechanism of CASPEr relies on the context information and context-aware security policy, so the faults caused by the two key points will affect the whole system. The CASPEr should be able to handle exception of the system such as the unavailability of context information, the syntax errors of policy, unsupported functionality for evaluation engine etc.

- $R_4$ CASPEr should be able to handle exception of the system.
  - $R_{4.1}$ CASPEr should be able to detect exception of unavailability of context information.
  - $R_{4.2}$ CASPEr should be able to handle the syntax error or datatype error of policies.
  - $R_{4.3}$ CASPEr should provide set of default policies under indeterminate situation.
3.4.2 Analysis of non-functional requirement

In addition to the above functional requirements, the CASPEr also needs to fulfill some other non-functional requirements. As we discussed above, the availability of the context information is important to the CASPEr. On the other side, the CASPEr itself also faces many kinds of threats in security aspects such as message transmit and storage. Furthermore, the usage environments (OS system, network type) of CASPEr will be heterogeneous under ubiquitous computing environment. Consequently, the CASPEr should be interoperable with various kinds of systems. So we give the following non-functional requirement R5:

- **R5 CASPEr should be able to fulfill following non-functional requirements**
  - R5.1 Availability of the context information
  - R5.2 Security of CASPEr system
  - R5.3 Interoperability (Standard message and policy)
  - R5.4 Robustness of CASPEr system
  - R5.5 Lightweight of CASPEr system
4 CASPEr Design

In this chapter, the design of CASPEr is presented. Firstly, a generic model of CASPEr is given together with the policy decision making process. Then we move on context-aware access control issues; apply the CASPEr model on access control mechanism. The specific policy language model for context-aware access control is demonstrated. Furthermore, we explain how it works through several policy examples.

4.1 General CASPEr Model

4.1.1 Motivations

To meet the requirements identified within chapter 3.4, a generic model is designed for context aware security policy enforcement, as shown in figure 4-1. This architecture achieves the following purposes: evaluating the security policy against input and context information, and returning the security policy decisions; acquiring and reasoning context information; and administrating security policies.

When we design the model, following characteristics of system are taken into account. From the software engineering point of view, the application should not be intermingled with the management and storage of its context. So decoupling mechanism for acquisition, interpretation and provision of context information is a significant principle. Moreover, the context description method should be generic, standard and extensible to meet the requirements of various concrete applications.

Another important character of context awareness in CASPEr design is the mobile property. As we discussed within the scenarios of chapter 3, a large mount of context aware security applications are under mobile environments, where the context of the application is highly dynamic. One thing we must notice is the resource limitation of mobile devices (such as CPU computing capability, memory capacity and battery capacity). Furthermore the network connection quality is not always guaranteed under mobile environment, therefore CASPEr couldn’t always rely on remote server.

These features of the CASPEr computing environment affect our design of CASPEr model. Some points are described following:

- Our design should support stand-alone applications when device is disconnected from network. So the main components of CASPEr should be located on a single device so that to make the system does not depend on remote communication. It will increase the robustness of CASPEr.
- The architecture should consider the resource limitation of mobile device. Consequently, the CASPEr system should be lightweight, such as just acquiring context information that needed for specific policies. Moreover, storage of history context information is not compulsory requirement.
- For the sake of the limited context acquisition capability of mobile device, CASPEr should be able to obtain context from remote context information provider and be able to share context information with other mobile devices.

To sum up above mentioned considerations and CASPEr requirements, we make the following decisions in CASPEr model design:

Firstly, for the purpose to make the security policy enforcement process to be context aware, the application must let input go through the CASPEr system. So we setup a Policy Enforcement Point – PEP (the term is defined in [RFC 3198] and [RFC2753]) component between security input and output to prevent the input bypass CASPEr.

Secondly, in order to meet the requirement R3 in section 3.4, CASPEr needs an evaluation engine that is able to make policy decision against context-aware security policies. This core function will be performed by the Policy
Decision Point – PDP (the term is defined in [RFC3198] and [RFC2753]) component. The tasks of PDP are shown in figure 4-1 and 4-2.

Thirdly, to meet the requirement R1 of section 3.4, CASPEr needs a component to edit and store security policies, and this component makes the policies available to PDP. We let Policy Administrator Point - PAP to realize these functions, as shown in figure 4-1 and 4-2.

Fourthly, the system should be lightweight as discussed above. So we do not collect all kinds of context information that provided by sensors, but the appropriate context information according to the needs of specific security policies evaluation. That means there should be a component to communicate with PDP before acquiring context information and other policy decision information (such as subject ID or object ID). For this reason, CASPEr Finder component is responsible for the task, as shown in figure 4-1.

Finally, in order to meet requirement R2 of section 3.4, the Context Broker component is added into architecture for context information acquisition and reasoning work. Via this component, we can decouple mechanisms for acquisition, reasoning and provision of context information to security policy evaluation engine. Besides, Context Broker is also responsible for sharing context information with other peer Context Brokers. The tasks of Context Broker are shown in figure 4-3.

### 4.1.2 CASPEr model and components

Based on the reasons mentioned above, we design following CASPEr model, which is illustrated in figure 4-1.

Figure 4-1 Context Aware Security Policy Enforcement model
Within CASPEr model, we identify the following basic components:

- **PEP (Policy Enforcement Point):**
  This component is responsible for creating the native requests against input, communicating with PDP and enforcing the output policy decisions. The PEP is placed between input and output of application so that the mechanism enforcement process must through the CASPEr PEP.

  PEP has two interfaces. One is used to accept input and output the policy decisions. Another is responsible for communicating with PDP, sending the native format request message to PDP and receiving native format policy decision response message from PDP.

- **PDP (Policy Decision Point):**
  This component is the core of CASPEr, it evaluates applicable policy and renders policy decision against policy decision request come from PEP, policy decision information and context information which is collected via various methods (such as from system itself, sensors and context information provider).

  As shown in figure 4-2, PDP accepts policy decision request message from PEP firstly, and then Policy Loader gets appropriate policies via PAP according to the information specified within the request message. Through analyzing the policies, PDP finds out what else policy decision information (include context information) is needed for evaluation, and then Policy Decision Information Loader asks information from CASPEr Finder. While PDP obtains all needed policy decision information, the policies and policy decision information are sent to Policy Evaluation Engine to perform the policy evaluation process.

  ![Figure 4-2 Tasks of PDP and PAP](image)

  **Figure 4-2 Tasks of PDP and PAP**

  PDP has three interfaces. One is used to accept/response message with PEP. The Second is responsible for getting policies from PAP. The other one is used to communicate with CASPEr Finder.

  We suppose that the PDP and PEP is tight coupled (located in one device) in mobile application. This mechanism has advantages in efficiency and timeliness, it can decrease the delay time between two components. Furthermore, another important advantage of the mechanism is avoiding the cost of authentication and protecting the communication between PDP and PEP.
• **PAP** (Policy Administrator Point):
  This component creates policies and makes policies available to **PDP**. **PAP** has one interface for communicating with **PDP** to provide policies. The **PAP** is also responsible for policies storage. For the sake of robustness requirement of CASPEr system, **PAP** also should provide a set of strict default policies to ensure the system works in case the context information is unavailable.

• **CASPEr Finder**:
  This component queries context information and other policy decision information that needed for policy decision evaluation. It has three interfaces for communicating with **PDP**, **Context Broker**, and collecting other needed policy decision information.

• **Context Broker**:
  This component is responsible for acquiring and reasoning the context information. As shown in figure 4-3, the Context Acquisition module is responsible for collecting raw context information from sensors and Context Information Provider. The Context Reasoning module deduces the high-level context information from raw context information, such as making the judge that whether patient is under emergency situation according to the value of his/her pulse, blood pressure and body temperature. Then the high-level context information is provided to **CASPEr Finder** (application layer) via Context Broker Interface. Besides, peer **Context Brokers** of different devices are able to share their context information via CB Interface. Furthermore, the history of context information can be stored in Context Repository for later usage. However, as we discussed formerly, the storage of context information is optional because of the resource limitation of mobile device.

  Context Broker has three interfaces for communicating with sensors and context information provider, **CASPEr Finder** and other **Contest Brokers**. Moreover, the **Context Broker**’s function also can be a generic service that act as middleware.

![Figure 4-3 Tasks of Context Broker](image)

The CASPEr model shows us the functions of CASPEr components. Furthermore, CASPEr’s policy decision making process is described in the sequence diagram 4-4.
Figure 4-4 CASPeR Policy Decision Making Process Sequence Diagram
4.1.3 Policy decision making process

The sequence diagram shown in figure 4-4 illustrates the entire process that how CASPEr system make policy decision according to security policies against context information and other policy decision information. The whole process is described following:

1. **PEP** accepts system’s input, which is various for different security mechanisms. For instance, the input is the access request that sent by Initiator for access control mechanism.
2. **PEP** translates input to standardized native format **PolicyDecisionRequest** message (XML based), and sends it to **PDP**.
3. **PAP** writes policies and makes them available to the **PDP**. **PDP** send **PolicyRequest** message to **PAP** for requiring one or more policies. These policies represent the complete policy for a specified target.
4. **PAP** returns **PolicyResponses** message that includes the XML based policy statement to **PDP**.
5. Based on the information that derived from policies, **PDP** makes sure what kinds of context information and other policy decision information is needed, and send request to **CASPEr Finder**.
6. **CASPEr Finder** collects needed policy decision information such as subject attributes, object attributes, action attributes etc.
7. **CASPEr Finder** sends **ContextInformationRequest** to **Context Broker**.
8. **Context Broker** collects related context information from Sensors, Context Information Providers or other Context Brokers.
9. **Context Broker** reasoning the raw context information to higher level context information, which can be conveniently used by **PDP**.
10. **Context Broker** returns **ContextInformationResponses** to **CASPEr Finder**.
11. **Finder** sends the collected **PolicyDecisionInformation** (attributes and context information) back to **PDP**.
12. **PDP** evaluates the security policy against context information and attributes.
13. **PDP** translates the result of policy decision evaluation to the native response format, and returns **PolicyDecisionResponse** message to the **PEP**.
14. **PEP** sends output to application and fulfills the policy decision.

For discussing policy decision making process in detail clearly, the whole process can be separated to three parts: PEP-PDP interaction and policy evaluation process part, PDP-PAP and PDP-Finder interaction part, and Policy Decision Information collecting process part. We will represent these three parts in following text and figure 4-5, 4-6 and 4-7, and the numbers of message sequences are in accordance with figure 4-4.

Firstly, the PEP-PDP interaction part is shown in figure 4-5. This sequence structure maintains the basic PEP-PDP interaction structure that described in [RFC2753]. Within CASPEr model, for the purpose to make system to be context aware, we set PEP between application’s input and output. This ensures CASPEr system become the essential part of the whole system, so the executing of application can not bypass CASPEr system. The PEP plays a media role between CASPEr policy evaluation engine - PDP and the application. That’s the reason why the first input message [1] is sent to PEP and last output message [14] is come out from PEP. PEP is also responsible for translating original input message into CASPEr native format PolicyDecisionRequest message [2], which is formal specified XML-based message. In PEP-PDP structure, PDP takes care of security policy evaluation process [12], makes the policy decision against security policies and Policy Decision Information (which include context information), then return the PolicyDecisionResponse message [13] back to PEP. While PEP receives policy decision it sends output or performs corresponding actions to application. For the sake of PEP and PDP are located in a single device (the reasons are described in section 4.1.1) and timeliness, the PEP keeps the active state during waiting for the result of policy decision making process.
Secondly, the interactions between PDP-PAP and between PDP-Finder are shown in figure 4-6. The task of this part is ensuring PDP obtain the needed information for security policy evaluation. Some information includes appropriate security policies and policy decision information (which includes context information) is needed for PDP evaluation engine. As we analyzed in section 4.1.1, the CASPEr does not always collect all kinds of the context
information because the resource is limited for mobile system. That is the reason why PDP interact with PAP and try to get the suitable policies ([3], [4]) according to the information that derived from PEP’s policy decision request message in the earlier stage. Via these fine-grained and formal specified policies (XML based), PDP is able to discover what kinds of context information and other PolicyDecisionInformation are needed for policy evaluation. Consequently, PDP is able to query these kinds of information via CASPeR Finder on clear purpose instead of aimlessly.

Furthermore, as shown in figure 4-6, the next task of PDP is to query and obtain policy decision information via interacting with CASPeR Finder ([5], [11]). The concrete steps of how to collect context information and other information are discussed in next part.

We should notice that if the PAP’s job is interrupted, the entire CASPeR will be abnormal, because PDP couldn’t find the essential element – security policy for evaluation work. PAP is also a key point for protecting security policy to prevent disclosure and tamper threats from outside of system. More considerations of safeguards can be found in appendix B.

---

**Figure 4-7 Policy Decision Information collecting process**

Thirdly, the process of collecting context information and other policy decision information is illustrated in figure 4-7. As mentioned above, PDP need many kinds of information, which according to security policies, for policy evaluation. The CASPeR Finder and Context Broker are responsible for collecting that information. We should point out that the context information collection process ([7], [8], [8’], [9] and [10]) and attribute (other policy decision information) collection process ([6] and [6’]) have no specific sequence. The two processes also can perform simultaneously.

The Context Broker is an independent component that performs the context acquisition ([8] and [8’]) and context reasoning ([9]) tasks. We can regard Context Broker as middleware between policy evaluation modules and low-level sensors layer. Context Broker is also responsible for context reasoning work, which deduces higher level context information from raw context information that come from sensors and context information provider. The
higher level information is more convenient and can be used easily for PDP. A simple example is that Context Broker should be able to infer that whether the user’s device is under hostile (public) environment through context information such as the network type, network domain or location.

An important problem that should be taken into account is that how to handle the exception while the CASPEr cannot collect needed context information. So there should be a set of default security policies that can be used when the system detect that context information is unavailable. Default policies should be strict security setting to protect entire system. As long as the interactions between PEP-PDP-PAP are not interrupted, the system is still able to work.

4.1.4 Conclusion

Within section 4.1 we demonstrate the generic CASPEr model and policy decision making process. Based on the requirements of CASPEr system and our analysis, we believe that the model and decision making sequence process is reasonable. In the following sections, the consideration will be focus on applying CASPEr model on context-aware access control. Because access control is very important aspect of security field and context awareness will be very useful in this mechanism within ubiquitous environment, which is represented in our scenarios within chapter 3. It is valuable and comparatively easy to introduce context awareness into this mechanism to provide dynamic and advanced access control. The detail contents are represented in section 4.2 to section 4.5.

4.2 Adapt CASPEr model to Access Control

From the scenarios of section 3.2.1 and 3.2.2, we could find that the context aware access control is valuable in many mobile applications, such as business and healthcare. Especially for healthcare applications, ensuring the privacy and security of sensitive healthcare data such as patient’s medical recorder is very important, and it is still an open question. When the mobile technology is widely used in this field, the dynamically changed environment makes the issue more complicated, but also give us the chance to import context awareness into this kind of applications to provide more advanced and valuable usage. Context aware access control can make the healthcare system to specify more flexible, precise and fine grained access authorization policies for the applications that under dynamically changed situation.

A typical example we mentioned within healthcare scenario is: “in order to perform the medical treatment in time under emergency situation, any physician closed to patient can access his/her medical record. However, in normal situation, only patient’s attending physician can access his/her medical record.” The context aware access control policy provides privacy and security protection on patient medical record, as well as does not delay the rescue under emergency.

Within this section, we will introduce traditional access control framework and mechanisms firstly. Then we focus on how to adopt context information on access control. Based on the survey of nowadays policy languages, we choose two policy languages as the candidates: XACML and WS-Policy. They are compared in several aspects and we will demonstrate which one it better for context aware access control. Furthermore, in section 4.3, we identify how to extend XACML language model to make it appropriate for the context aware access control. Finally, several policy examples are given in section 4.4.

4.2.1 Traditional access control

Access control is a set of mechanisms and processes that aim at protecting computer system from unauthorized entities. Access control determines whether an access request to a resource or data is granted or denied. It can provide protection against unauthorized disclosure, unauthorized use, unauthorized creation, modification and destruction, etc.

A general access control framework (figure 4-8) is given by ISO 10181-3. The access control could be worked on many types of entities, such as files, database, programs, enterprise resources and some real systems. The basic entities within this framework are: initiator, target, Access Control Enforcement Function (AEF) and Access
Control Decision Function (ADF). Here AEF corresponds to PEP concept, and the ADF corresponds to PDP concept.

![Diagram of access control framework](image)

**Figure 4-8 ISO/IEC 10181-3 access control framework**

*Initiator* can be human user or computer based entity that tries to access the *target(s)*. The *Targets* represent the entities that initiators try to access to. *Targets* could be file, database, program or real system (such as electronic door).

Within the figure 4-8, the access request represents the operations and associated operands. The AEF and ADF ensure that only valid initiator’s accesses can be performed on *targets*. When AEF receives an access request on a *target*, it communicates with ADF and requires ADF to make the access authorization decision. The ADF make the decision based on the access request and following types of access control decision information:

- *Initiator ADI* (Initiator Access Control Decision Information), which is derived from the access control information that bound to initiator. Examples of initiator access control information include: access control identity of an individual; identifier of the group; identifier of roles; integrity marking etc.

- *Target ADI* (Target Access Control Decision Information), which is derived from the access control information that bound to the target. Examples of target access control information include: target access control identities; integrity markings; identifier of the target container etc.

- *Access request ADI* (Access request Access Control Decision Information), which is derived from the access control information bound to the access request. Examples of access request information include: the class of operation, such as read, write; integrity level that required for operation; data type of the operation etc.

- *Access control policies*, which are from ADF’s security domain authority.

- *Contextual information*, which is needed to interpret the access control decision information or policy. There is little attention is paid to contextual information on traditional access control mechanism.
Based on the above mentioned inputs and access control decision information, the ADF decides the authorization decision that permits or denies initiator’s access request, and send the result back to AEF. AEF performs the decision to allow or deny the access to target, furthermore, may take other add-on actions (obligations).

We want to represent a simple access control example that use above framework. We assume a manager want to view company’s sensitive financial data from his company’s database, and he could only through company’s access control system to access the database. Firstly, the manager sends an access request to AEF. When AEF receives the request, it communicates with ADF and requires ADF to make the authorization decision. Then ADF make the decision base on initiator ADI, target ADI, access request ADI and access control policies. The evaluation of policies is based on initiator ADI (such as the manager’s identity, his role in the company and his security token, which can be username + password or X.509v3 certificate), target ADI (such as the identifier of the database or concrete piece of data), and access request ADI (the operations that manager want to perform on the data, in our example is read). According to company’s specific policy, the manager is able to view financial data. ADF completes the policy evaluation process and renders the decision that access is granted. Then it sends the authorization decision back to AEF. AEF sends the decision to database server and set up the communication between manager’s client application and database server.

**Access Control Policy**

Within ISO framework, access control policy plays an essential role. Access control policy is a kind of security policy that handles the access control services and applications. Access control policy uses a set of rules to express the security requirement in access control mechanism. These rules can be utilized by ADF to render the access control decision. The traditional access control policies can be categorized in two types: rule-based access control policy and identity-based access control policy.

Rule based access control policies are intended to apply to all access requests by any initiator on any target in security domain [Mic03]. A set of rules determine who has the access right to resources. Security label access control method is a particular type of rule based policy. The access decisions are made by comparing the initiator and target security labels. What kinds of accesses that can be performed are determined by rules according to specified security labels.

Identity based access control policies is based on the rules that are specified to certain initiator, a group of initiators or the role of initiator. A particular well known example of identity based access control is RBAC – Role Based Access Control. A given role may be applied to a single user, such as employer, or be applied to a group of users, such as employees. Moreover, employer may have more privileges than employees hold, so the hierarchical structure can be used here to cooperate with role and group access control. We should note that user’s role is also a very important kind of context information within context aware application.

**Traditional Classification of Access Control Mechanism**

Traditional access control mechanism may have several typical access control schemes, such as access control list (ACL), capability scheme and label based scheme. Some characteristics of these mechanisms are summarized within [Mic03]. Furthermore, the context-based schemes will play more and more important role nowadays, and this kind of new method is what we are interested in our research.

- **Access Control List (ACL)**
  
  When we consider the access control from the initiator identity viewpoint, and use a set of \([\text{initiator identity}, \text{operation type}]\) pairs as target access control information, then we get access control list under certain access control policy. The characteristics of ACL are:
  
  - Access control is managed as a list of \([\text{initiator qualifier}, \text{operation qualifier}]\) pairs.
  - Individual, group or role identifier is initiator access control information.
  - ACL is convenient in case targets are dynamic.
  - ACL is convenient in case initiators, or the groups or roles of initiators are not many.
ACL is easy to revoke access to targets.
ACL is convenient in case the access control management is set on the target side.

The *initiator qualifier* is the distinguishing identifier of initiator (individual, group or role) to which the *operation qualifier* is applied. The *operation qualifier* represents the operations (in access request) that are allowed or denied for associated *initiator qualifier*. Via matching algorithm, the initiator access control decision information and the operation that derived from the access request are compared with the \([\text{initiator qualifier}, \text{operation qualifier}]\) entry of ACL. The decision is made on whether or not the match is made.

However, ACL mechanism has its weak point in some cases. From above mentioned ACL’s features, we could find out that when there are too many individual or groups of initiators, or the initiators changes frequently, the ACL scheme will be not appropriate. Under this kind of situation, another mechanism known as capability scheme is appropriate.

### Capability scheme

On the contrary of the ACL scheme, if we consider the access control from the target viewpoint, use \([\text{target identity}, \text{operation type}]\) pairs as initiator access control information, and use target identities as target access control information, then we can get capability scheme. The characteristics of capability scheme are:

- Access control is managed in terms of initiator bound access control information that defines a set of allowed operations (the capability of initiator) on identified (set of) targets.
- Capability scheme is convenient when the number of targets is not large.
- Capability scheme is convenient in case the access control management is set on the initiator side.
- Capability scheme is convenient when the initiators and targets are in different security domains.

Within capability scheme, the initiator access control information is a set of *capabilities*, which have two components: *identity* of target or targets set, and the *operations* authorized on the target. Capabilities can be conveyed by access control certificate. The initiator binds access control certificate on access request. Then ADF evaluates the target access control information to verify whether the target identity in the *capability*, and whether the operation is in the *capability*. The access is permitted when both verifies are made.

We should notice that capability scheme also has negative point, it is not appropriate for the system that with a large mount of targets, or the targets are dynamic change frequently. While the negative point of capability scheme is just the positive point of ACL scheme, the ACL is a better choice under this kind of environment. So we can see that the ACL and Capability schemes are compensatory to each other.

### Label-based scheme

The label based scheme use the security labels, which can be assigned to initiator and target. It is useful when the system has many initiators and targets and the access requests are not too much. In label based scheme, the operations are not explicitly included in the initiator access control information or target access control information. The operations are defined in certain security policies. Because this method is appropriate for the system has both many initiators and many targets, so that when the ACL and capability schemes are both encounter difficulty, label-based scheme maybe a solution.

Except these kinds of access control scheme, the context based scheme is get more consideration recently. However, there are not many works focused on it. A practical access control system may need various schemes to cooperate together. For the sake of context awareness could provide more dynamic information to the access control decision making process, it will let the access control mechanism become more dynamic and advanced. Via introducing context information into access control, we are able to produce more complex and diverse access control policies for different situations and environments. The context aware access control can be more appropriate for some special dynamic mobile applications such as health care under emergency. Following sections will focus on how to make the access control to be context aware.
4.2.2 Introduce context awareness into access control

We find that it is possible to adapt CASPEr model on access control mechanism to achieve the dynamic auto-configuration access control. We could introduce context information into access control and make it takes affect in policy decision making process.

Within ISO 10181-3 access control framework, it also point out the contextual information concept. But there are not many concrete works on it. As we described in CASPEr generic model, we utilized the PEP-PDP structure, which can be mapped to AEF-ADF framework. In fact, when the CASPEr generic model is used for access control, the CASPEr PEP and PDP will play the similar functions compared with AEF and ADF separately. The mapping between two models is shown in figure 4-9.

![Figure 4-9 Mapping between CASPEr access control model and ISO/IEC framework](image-url)
Comparing two models shown in figure 4-9, we can find out that PDP’s function is same with ADF, they are both responsible for making policy decisions; and PEP’s function is same with AEF, they both play the role of sending access request to PDP and enforcing the policy decision. Furthermore, within CASPEr model, the structure is more specific: the access control policies will be provided by PAP, and three kinds of Access Control Decision Information (initiator ADI, target ADI, access request ADI), which is represented by ISO/IEC framework, are mapped to several kinds of Policy Decision Information (subject attribute, resource attribute, action attribute etc.) in CASPEr model, and these attributes are collected through CASPEr Finder.

However, we need more components for introducing context information into the framework. Firstly, the acquisition and reasoning work of context information are essential for context-aware access control system. As we discussed within section 4.1, we decide to decouple mechanisms for acquisition, interpretation and provision of context information. That’s the reason why we need Context Broker component in the CASPEr model. Secondly, a set of standard context-aware policy language is the key point for CASPEr access control. In the following sections, we will provide a solution of it.

Table 4-1 Mapping between CASPEr and ISO access control model

<table>
<thead>
<tr>
<th>CASPEr access control model</th>
<th>ISO access control framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy decision making component</td>
<td>Performed by Policy Decision Point (PDP)</td>
</tr>
<tr>
<td>Policy enforcement component</td>
<td>Performed by Policy Enforcement Point (PEP)</td>
</tr>
<tr>
<td>Access control policy supply component</td>
<td>Performed by Policy Administrator Point (PAP)</td>
</tr>
<tr>
<td>Information provided to policy decision making process</td>
<td>Policy Decision Information (Subject attributes, Resource attribute, Action attribute etc.)</td>
</tr>
<tr>
<td>Policy decision information acquiring component</td>
<td>CASPEr Finder</td>
</tr>
<tr>
<td>Context information acquisition component</td>
<td>CASPEr Context Broker</td>
</tr>
</tbody>
</table>

For designing CASPEr access control policy language, we survey the technologies of nowadays policy languages, and decide to choose XACML and WS-Policy as the two candidates for the policy language basis. These two policy languages are most prevalent and mature, and can achieve good interoperability on different platforms. In the next section 4.2.3, we will compare the candidates of policy language on various aspects that are significant for context awareness, and we chose XACML as the basis of the CASPEr access control policy language eventually. Moreover, we give more introductions of other needed technologies such as XACML and SAML in the appendix. After section 4.2, possible extension models of access control policy languages are provided in sections 4.3.

4.2.3 Policy language candidates for CASPEr

For introducing context awareness into access control, we need to develop an appropriate context-aware access control policy language. As we analyzed within CASPEr requirement R1, this language should have the following characteristics:
- the policy language should specify standard policy format that can exchange between CASPEr access control components
- the policy language should specify standard description method of context information
- the policy language should be able to modify and extend context information type
- the policy language should specify standard access request/response message format
- the policy language should support rich set of operations for context-aware policy evaluation
- the policy language should be able to work with different platforms

Because CASPEr needs to work under ubiquitous computing environment and heterogeneous networks, so the standardization and interoperability is essential for the policy language. The XML based policy language can facilitates cooperation between heterogeneous platforms. In fact, the features that required by CASPEr policy language are also essential for web service requirements. So we try to utilize some technologies that developed for web service. Another important aspect that inspired us is that some designs of web service policy language are grow out from access control model. Based on our survey, we find that several policy languages (such as XACML, WS-Policy) that developed for web service fulfill part of CASPEr language required characteristics. Moreover they are XML based and extensible, so they are appropriate for cross platforms usage. Therefore, we need to find the most appropriate one which is suitable for extending for context aware access control purpose.

In [And05], author presents several policy languages that have been proposed at W3C workshop as the basis for new standard are based on Boolean combinations of predicates for web service policy language. These languages include: Web Service Policy Framework (WS-Policy), Web Services Description Language (WSDL) with the addition of compositors, XACML profile for web services (WSPL) and a language outline from INOA Technologies [INOA04]. In WS-Policy, the predicates are Assertions that return a Boolean result, but are not defined in the policy framework itself. In WSDL compositors, the predicates are WSDL Boolean Features, Properties or nested compositor expressions. However, the Features and Properties are not further defined. In XACML WSPL, the predicates are XACML functions that return a Boolean result and operate on Attribute and literal values. The Attribute may be a (name, type, value) triple or a node in XML that identified by XPath expression. In IONA outline, the predicates are simple XML elements, with at most Yes/No parameter, and the process of defining the elements to be used is not specific in the outline.

Within these four languages, both WSDL and IONA policy language outline don’t describe their predicate in detail. Furthermore, the structures of XACML and WS-Policy have been standardized, and they are comparative mature in research and implementation. Therefore, we focus the discussion on these two languages and choose one of them as the language basis for CASPEr access control.

4.2.3.1 WS-Policy

WS-Policy, which is developed by Microsoft\IBM\BEA\SAP for web service, is an XML based policy language. It defines an XML framework to exchange authentication and authorization information for accessing web service or other web resource. It specifies an XML-based structure called a policy expression containing policy information, and the grammar elements to indicate how the contained policy assertions apply. WS-Policy includes three specifications: WS-Policy (Web Service Policy Framework), WS-Policy Assertions, and WS-Policy Attachment. WS-Policy Framework provides a generic model and grammar for describing and communicating the policies of a web service. WS-Policy Assertions and other specification (such as WS-Security) provide specific applications of the grammar in their domains. WS-Attachment is responsible for associating policies with web services.

Within WS-Policy, a policy is a collection of one or more policy assertions. WS-Policy provides a flexible and extensible grammar for expressing capabilities, requirements, preference and general characteristics of Web Service entities. For access control usage, it represents the properties that have to be satisfied to policy subject associated with assertions. The XML representation of a policy assertion is known as policy expression. The element wsp:Policy is the container of a policy expression. Every assertion is associated with an obligatory usage qualifier: Usage, which specifies how the assertions should be processed. Five types Usage values are:

- wsp: Required – the assertion must be applied to the subject.
- wsp: Rejected – the assertion is not supported. The present of this kind assertion will cause failure.
- wsp: Optional – the assertion may be applied, but not required.
- wsp: Observed – the requestor of the service will be informed that the policy will be applied.
wsp: Ignored – the assertion is processed, but ignored. No action will be taken, the requestor will be informed.

- wsp:All - requires that all of its child elements be satisfied.
- wsp:ExactlyOne - requires that exactly one of its child elements be satisfied.
- wsp:OneOrMore - requires that at least one of its child elements be satisfied.

In case of there are many possibility for granting a access, attribute wsp:preference can be used to decided the order of preference. WS-Policy also provides another method to share policy expressions between different policies, via element wsp:PolicyReference. The position with PolicyReference will be replaced by the referenced policy expression. A simple access control example that expressed by WS-Policy can be found in following comparison section 4.2.3.3, Example 1.

For the sake of WS-Policy is developed without consideration well defined model, it make WS-Policy become an ambiguous language that is subject to different interpretations and uses under some situations. Given a set of policies that expressed by the WS-Policy syntax, the evaluation may have ambiguous results. This will be a very serious problem in access control usages, because access control must make out an explicit policy decision result. WS-Policy is still a proprietary specification of its development companies, though it has been submitted to W3C, but still not be approved as a standard when we do the survey work.

### 4.2.3.2 XACML

The OASIS eXtensible Access Control Markup Language [XACML] is a powerful, standard language that specifies schemas for authorization policies and authorization decision requests and responses. It also specifies how to evaluate policies against requests and other attributes to compute a response.

When a resource access is attempted, the Policy Enforcement Point (PEP) sends a description of the attempted access to a Policy Decision Point (PDP) in the form of an authorization decision request. The PDP evaluates this request against its available policies and attributes, and produces an authorization decision that is returned to the PEP. The PEP is responsible for enforcing the decision. The PDP may augment the PEP's description of the access request with additional attributes. The PDP may obtain policies from on-line Policy Administration Points (PAP) or from Policy Repositories into which PAPs have stored policies. XACML can serve as a standard format for exchanging information with these entities when combined with other standards.

#### XACML context

XACML is intended to be suitable for a variety of applications. The core language is separated from the application environment by the XACML context, as shown in figure 4-11. The XACML context is defined in XML schema, describing a canonical representation for the inputs and outputs of the PDP. Attributes referenced by an instance of XACML policy may be in the form of Xpath expressions over the XACML context, or attribute designators that identify the attribute by subject, resource, action or environment and its identifier, data-type and (optionally) its issuer. The implementation of XACML should convert between the attribute representations in the application.
environment (e.g., SAML, J2SE, etc.) and the attribute representations in the XACML context. (Note: the XACML context used here is not the context information that represented by us in the thesis, but is request/response message.)

Figure 4-11 XACML context

Rule and Policy Combining

XACML defines three top-level policy elements: <Rule>, <Policy> and <PolicySet>. The <Rule> element contains a Boolean expression that can be evaluated in isolation, but that is not intended to be accessed in isolation by PDP. So it is not intended to form the basis of an authorization decision by itself. The <Policy> element contains a set of <Rule> elements and a specified procedure for combining the results of their evaluation. It is the basic unit of policy used by the PDP, and so it is intended to form the basis of an authorization decision. The <PolicySet> element contains a set of <Policy> or other <PolicySet> elements and a specified procedure for combining the results of their evaluation. It is the standard method for combining separate policies into a single combined policy.

XACML defines a number of combining algorithms that can be identified by a RuleCombiningAlgId or PolicyCombiningAlgId attribute of the <Policy> or <PolicySet> elements, respectively. The rule-combining algorithm defines a procedure for arriving at an authorization decision given the individual results of evaluation of a set of rules. Similarly, the policy-combining algorithm defines a procedure for arriving at an authorization decision given the individual results of evaluation of a set of policies. Standard combining algorithms are defined as:

- Deny-overrides (Ordered and Unordered) (both for rule and policy combining),
- Permit-overrides (Ordered and Unordered) (both for rule and policy combining),
- First-applicable and (both for rule and policy combining)
- Only-one-applicable (only for policy combining).

More details of combining algorithms can be found in section 4.3.3 and [XACML]. If necessary, users may define their own combining algorithms.

4.2.3.3 Comparison with XACML and WS-Policy

The main difference between XACML and WS-Policy is that XACML is based on a well-defined model and has fine-grained formal specification, but WS-Policy is developed without consider this modeling phase and has no specified formal representation. The following table 4-2 gives a mapping of elements between XACML and WS-Policy Framework.

Table 4-2 Mapping elements of XACML and WS-Policy

<table>
<thead>
<tr>
<th>XACML</th>
<th>WS-Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Policy attachment</td>
</tr>
<tr>
<td>Policy set</td>
<td>Policy inclusion</td>
</tr>
<tr>
<td>XACML Policy</td>
<td>WSPF Policy</td>
</tr>
<tr>
<td>Rule</td>
<td>Policy assertion</td>
</tr>
<tr>
<td>Effect</td>
<td>Usage/Preference</td>
</tr>
</tbody>
</table>
Then via a policy example, which implement both in XACML and WS-Policy, we can see more difference between the two policy languages. The following example describe a simple policy: the access is granted if exactly one security token among Kerberos certificate or X.509 v3 certificate is provided.

**Example 1 (WS-Policy):**

```xml
<wsp:Policy xmlns:wsp="..." xmlns:wsse="...">
  <wsp:ExactlyOne>
    <wsse:SecurityToken wsp:Usage="wsp:Required">
      <wsse:TokenType>Kerberosv5TGT</wsse:TokenType>
    </wsse:SecurityToken>
    <wsse:SecurityToken wsp:Usage="Required">
      <wsse:TokenType>X509v3</wsse:TokenType>
    </wsse:SecurityToken>
  </wsp:ExactlyOne>
</wsp:Policy>
```

**Example 2 (XACML):**

```xml
<Policy PolicyId="policy:1" RuleCombiningAlgorithm="&permit-overrides;">
  <Rule RuleId="rule:1" Effect="Permit">
    <Condition FunctionId="&function;string-is-in">
      <AttributeValue DataType="&string;">Kerberosv5TGT</AttributeValue>
    </Condition>
  </Rule>
  <Rule RuleId="rule:2" Effect="Permit">
    <Condition FunctionId="&function;string-is-in">
      <AttributeValue DataType="&string;">X509v3</AttributeValue>
    </Condition>
  </Rule>
</Policy>
```

The following table gives a detailed comparison between XACML and WS-Policy from various aspects. Some of these aspects, includes standard structure, policy type, operations, data types, composition algorithms, extensibility, evaluation method, development support etc. are very important for extending the language to be context aware.

**Table 4-3 comparison between XACML and WS-Policy**

<table>
<thead>
<tr>
<th></th>
<th>XACML</th>
<th>WS-Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Structure Support</strong></td>
<td>Yes - XML based; - Formally specified, fine-grained semantics.</td>
<td>Yes - XML based;</td>
</tr>
<tr>
<td><strong>Policy Types Support</strong></td>
<td>Yes - Can distinguish different types of policy rules, such as authentication, privacy etc.</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Data Types Support</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Based on the survey of two languages, we found that WS-Policy has some weak points when used to express access control policy.

Firstly, WS-Policy doesn't define a formal model and well defined specification, it just be described only by means of XML XSD schemas. So the description of WS-Policy may cause ambiguities in different implementations and usages. In [Ard04], author point out one of the ambiguity of WS-Policy. If a policy that use operator <All> to include one or more policy assertions for which the attribute Usage has value "Optional", the ambiguity arises. It is not sure whether the “Optional” assertion should be considered or not, such as the following policy Example 3.

Example 3 (ambiguous combination of WS-Policy)
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
    <wsp:All>
        <wsse:SecurityToken wsp:Usage="Required">
            <wsse:TokenType>Kerberosv5TGT</wsse:TokenType>
            ...
        </wsse:SecurityToken>
        <wsse:SecurityToken wsp:Usage="Required">
            <wsse:TokenType>X509v3</wsse:TokenType>
            ...
        </wsse:SecurityToken>
    </wsp:All>
</wsp:Policy>

<table>
<thead>
<tr>
<th>Operations</th>
<th>- Standard data types;</th>
<th>- Don’t have standard data types;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Rich set of data types;</td>
<td>- Don’t have rich data types;</td>
</tr>
<tr>
<td></td>
<td>- Data types can be extended to create new types.</td>
<td>- Data types can be extended.</td>
</tr>
<tr>
<td></td>
<td>- Standard operations;</td>
<td>- Don’t have rich set of operations;</td>
</tr>
<tr>
<td></td>
<td>- Rich set of operations;Can perform exact matches, comparison, of value-range matches.</td>
<td>- Operations are extensible.</td>
</tr>
<tr>
<td></td>
<td>- Operations can be extensible.</td>
<td></td>
</tr>
<tr>
<td>Merging</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Merging based on value and semantic of policies;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Rich set of comparison operators.</td>
<td></td>
</tr>
<tr>
<td>Rule Composition Algorithm</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Rules combining algorithms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Policies combining algorithms</td>
<td></td>
</tr>
<tr>
<td>Multiple Subjects Support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy Intersection</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensibility</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Extensible attribute</td>
<td>- Completely dependent on extensions</td>
</tr>
<tr>
<td></td>
<td>- Extensible datatype</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Extensible new function</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Extensible rule and policy combine algorithm</td>
<td></td>
</tr>
<tr>
<td>Policy Evaluation</td>
<td>- Support standard evaluation engine</td>
<td>- Require custom evaluation engines</td>
</tr>
<tr>
<td>Standardization</td>
<td>- OASIS standard</td>
<td>- It is not yet an approved standard, but already be submitted to W3C committee.</td>
</tr>
<tr>
<td></td>
<td>- Related WSPL (Web Service policy language) draft is in OASIS XACML TC</td>
<td></td>
</tr>
<tr>
<td>Implementation Support</td>
<td>- Sun Java implementation support for XACML</td>
<td>- .Net development support for WS-Policy</td>
</tr>
<tr>
<td></td>
<td><a href="http://sunxacml.sourceforge.net/">http://sunxacml.sourceforge.net/</a></td>
<td></td>
</tr>
</tbody>
</table>
As we mentioned within section 4.2.3.1, <All> operator means that all assertions must be satisfied. But on the other hand, attribute Usage “Optional” means the assertion may be applied, but not required. Therefore, the policy may cause the different interpretations according to different implementations. Whether the policy operator takes precedence over the attribute Usage or on the contrary is not clear.

Moreover, when you use the <ExactlyOne> and Usage value “Required” together, it will also cause the conflict between Usage flags and operators, such as:

```xml
<ExactlyOne>
  Value=A Usage="required",
  Value=B Usage="required",
</ExactlyOne>
```

Similar problem will arise when combination performed with policy operator <All> and attribute Usage value “Ignored”. For access control usage, the ambiguity is a very serious weak point. However, this kind of problems does not exist in XACML, because XACML use the semantics that have been formally analyzed.

The second problem is that WS-Policy doesn’t define the assertion comparison functionality itself. Users must to specify every value for a fine-grained assertion themselves. On the contrary, XACML provides a rich set of comparison and match operators such as equal, greater than, greater than or equal to, less than, less than or equal to, etc. All the comparison operators are strongly typed and must agree with the data types that specified for the function parameters. Therefore, we can perform exact matches, comparison and value-range matches easily. Furthermore, the operations are extensible according to user’s concrete requirement. Comparing with WS-Policy, this character of XACML is very convenient for the context information matching process.

The third problem is that WS-Policy’s “Ignored” Usage, which means assertions with this Usage value must be processed and then the results must be ignored, seems to be useless. And the element “Preference” also seems be unnecessary especially under the control of <All> or <OneOrMore> operators.

Of course, XACML also has weak point compared with WS-Policy. From the Example 1 and 2, we can find that XACML is a little verbose. It will use more sentences than WS-Policy when states a same policy. So the policy writing work is more complex than WS-Policy, but it is not a big problem if we utilize the policy authoring tools. Nevertheless, as we analyzed above, we also get many advantages from this verbose, formal and specific semantics.

### 4.2.3.4 Conclusion

As discussed above, we found that XACML has many merits for context-aware extension compared with the WS-Policy:

- XACML uses standard datatypes and can be extended to introduce context information.
- XACML defines rich set of comparison operators, and it is also extensible. It can perform exact matches, logic comparison or value-range matches. So the rich set of operations is valuable for the context constrain evaluation.
- XACML can be extended via new attribute types, functions and combine algorithms. So we can perform enough extension to let it appropriate for the context-aware access control.
- XACML is already a standard policy language. It is fine grained and formally specified. So it can be cooperated with various kinds of platforms.
- XACML has Java open-source implementation support by Sun. It can be the basis for the implementation and let the development process easily.
Another advantage that we want to adopt XACML is that it can easily cooperate with Security Assertion Markup Language (SAML) to provide security communication. SAML is an XML-based framework for exchanging security information defined by OASIS organization, and it has defined the method for binding with SOAP. Therefore, this combination will let the secure message communication easily to be implemented.

The XACML already meets part of our policy language requirements, such as it has standard language model and specification, it has standard request/response message format, it supports rich set of operation functions, and it is able to work across different platforms. Fulfilling these policy language requirements let XACML is more appropriate to be extended for CASPEr access control, so we prefer XACML as the basis for CASPEr access control policy language. However, other CASPEr language requirements of supporting description and evaluation of context information are need to be extended based on XACML. The XACML extension solution will be discussed in next section 4.3.
4.3 Extending XACML for CASPEr

Through comparing policy language candidates, we decide to use XACML as the basis of CASPEr access control policy language and extend it to be context aware. Two extension methods are designed in our research, and we will identify which is better. Before demonstrating our policy language model, the original XACML language model is shown in figure 4-12.

![Figure 4-12 XACML Policy Language Model](XACML)

The main difference between XACML and other XML-based access control languages is that XACML relies on a model that provides a precise representation of the access control security policy and it works [Ard04]. This modeling phase ensures that XACML is a specific and unambiguous language to different interpretations and users. As shown in figure 4-12, an XACML policy has root element “Policy” or “PolicySet”. An XACML policy consists of a set of rules, targets, obligations and rule combining algorithm. A “Rule” specifies a “permit” or “deny” for a subject to perform an action on an object. The “Target” consists of a set of “conditions” for Subject, Resource, Action and
Environment} that must satisfied for a policy to apply to a given request. An important feature of XACML is that a rule is based on the definition of attributes corresponding to specific characteristics of subject, resource, action or environment. If all the conditions of a Target are satisfied, then its associated policy decision applied to the request. If a policy applies to all entities that belong to a group or type, then an empty element, named AnySubject, AnyAction, AnyResource is used to replace the original one.

For achieving more powerful policy evaluation, XACML supports attribute evaluation process. XACML define policies based on properties, named attributes, associated with subjects and resources other than identities. So we are able to write more generic properties associated with subjects and resources. XACML provides elements: SubjectAttributeDesignator, ResourceAttributeDesignator, ActionAttributeDesignator and EnvironmentAttributeDesignator which together with SubjectMatch, ResourceMatch, ActionMatch and EnvironmentMatch elements, they allow to identify a particular subject and resource attribute.

In section 4.1, we provide a generic CASPEr model. We apply this model on access control mechanism and use the extended XACML as the access control policy language, so that to achieve policy based context aware access control. In CASPEr model, the Context Broker is responsible for acquiring context information from physical sensors, virtual sensors, even from remote context information provider and peer Context Brokers. The Context Broker also does the context information reasoning work, and provides higher level context information to PDP via CASPEr Finder. Within policy evaluation process, PDP makes the authorization decision by evaluating extended XACML policy against the attributes that come from access request and context information that provided by the Context Broker. This process requires that the policy language - extended XACML must be able to represent the context information and the context constrains evaluation operation functions. The following two sections 4.3.1 and 4.3.2 will provide the solutions that make the XACML be able to represent context aware access control policies.

4.3.1 CASPEr access control policy language model - Solution 1

For extending the XACML to be context aware, we need import context information into XACML policy language model to describe the context constrains in XACML policy. The extended language must be able to represent context information, and let it can be interpreted and evaluated by PDP. The context information should be able to be matched or compared via operators such as equal, greater than, greater than or equal to, less than, less than or equal to etc. All the operators are strongly typed and agree with data types that specified for function parameters.

Because an important character of XACML evaluation process is that a rule is based on the definition of attributes corresponding to specific characteristics of a subject, resource, action or environment, in order to make the authorization decisions, the attributes that composed within request will be compared with attribute values in policy. Attributes are identified by the <SubjectAttributeDesignator>, <ResourceAttributeDesignator>, <ActionAttributeDesignator> and <EnvironmentAttributeDesignator>. These elements use <AttributeValue> element to define the value of particular attribute. So, our first design of extending XACML language model is adding “Context Information” element under “Target” element, on the same level with “Subject”, “Resource”, “Action” and “Environment”, as shown in figure 4-13.

“Context Information” includes two main types context information: “Subject context” and “Resource context”. Via this element, we can import five types of primary context objects: Time, Location, User Context, Device Context, and Network Context.

- Time: represents current time, can be provided by system clock of user’s device.
- Location: represents current location of the device. This context can be obtained via GPS or other kind of sensors, or from mobile service provider.
- User Context: represents user related context information, such as user’s name, ID, role, user’s action etc.
- Device Context: represents device identifier, device type, CPU type, battery level etc.
- Network Context: represents available network type, network bandwidth etc.
For different applications, more specific context objects can be defined and added to the language model. The definition of the types of Context Object is according to the concrete application.

This solution maintains classic context category method. However, this language model will make large change with original XACML language model. Is there simpler way to achieve our purpose? Another alternative solution is proposed in the next section.
4.3.2 CASPEr access control policy language model - Solution 2

We have described an XACML extended method in the above section. The method works, but it changes the XACML language model largely and will let policy evaluation work more complex. So we work out another simpler model to introduce context information, which is shown in figure 4-14. In this way, we break out the classic context category method, do not add new element under Target, but add different kinds of context information objects under each Target sub elements: Subject, Resource, Action and Environment. In fact, this context information category method is enough for our usage. So we get four types context information elements:

- **Subject Context information**, such as subject’s Identifier, subject’s role, subject’s location etc, which is similar to former user context.
- **Resource Context Information**, such as resource identifier, object’s location, resource network type etc.
- **Action Context Information**, the information related to access actions.
- **Environment Context Information**, such as time, location, current situation.

For instance, the Subject Context Object can be the role of subject, the location of the subject. The Environment Context Object can be time or situation, such as whether the patient is in state of medical emergency.

Compare with the solution 1, this solution gives us a simpler way to import context information into access control police. In this way, the value of context information can be assigned in the new attribute types of Subject, Resource, Action and Environment. And the context evaluation functions can be assigned under <Condition> via <Apply> element. This method makes less change to original XACML language model, so it is more easily to be

---

**Figure 4-14 Context information component and context types in language model**

For instance, the Subject Context Object can be the role of subject, the location of the subject. The Environment Context Object can be time or situation, such as whether the patient is in state of medical emergency.

Compare with the solution 1, this solution gives us a simpler way to import context information into access control police. In this way, the value of context information can be assigned in the new attribute types of Subject, Resource, Action and Environment. And the context evaluation functions can be assigned under <Condition> via <Apply> element. This method makes less change to original XACML language model, so it is more easily to be
implemented. We just need to defined new datatypes for describing context information, furthermore providing new evaluation functions when the standard operations and functions are not enough. More details about implementation are described in Appendix A.

Although the two solutions are both works for CASPer access control, we prefer the second one because of the simplicity. For the sake of it makes little change of original XACML, it can achieve better interoperability, and let the development process simply based on the existed XACML implementation. Moreover, the evaluation process of solution 2 is also simpler than solution 1. The comparison of the two evaluation process is represented in section 4.4.7. Consequently we will use solution 2 as the policy language model.

4.3.3 Detail of the policy language

As the result of above two sections, we prefer the second solution as our context aware access control policy language model. The detail of the language is presented in this section.

The main components of the policy language are Rule, Policy and PolicySet. A policy has root element either a Policy or a PolicySet, and a PolicySet is a combination of Policy or PolicySet.

**Rule:**

A *rule* is the most elementary unit of *policy*. The *rules* must be encapsulated in a *policy* in order to exchange between components.
As shown in figure 4-15, the main components of a rule are:

- **Target**
- **Effect**
- **Condition**

The *Target* of rule (also of policy or policy set) defines as the set of:

- **Resources**
- **Subjects**
- **Actions**
- **Environment**

The PDP verifies that whether the matches defined by the target are satisfied by the subjects, resource, action, environment attributes in the access request. The *Target* definitions are discrete, in order that applicable rules may be efficiently identified by the PDP. When several rules are combined to form a policy, the target element could be absent. In this case, the *rule target* is same as that of the *target* of parent policy element.

Every kind of (subject, resource, action and environment) Context Information components has several Context Object; and Context Object may be context types such as time, subject location, resource location, subject Role, network type etc. The definition of the context type should be able to extend according to concrete application.

The *Effect* component of rule indicates the consequence of a “true” evaluation for the rule. Two values are allowed: “Permit” and “Deny”.

The `<Condition>` component of rule represents a Boolean expression that refines the applicability of the rule beyond the predicates implied by its target. For the rule to be applicable to request, the conditions must evaluate to “True”. This element is very useful for the context constrain evaluation. We can use `<Apply>` element, which is the sub element of `<Condition>` to call the standard or extended operations functions to evaluate the context constrain that specified by rules against context information. The `<Apply>` element denotes application of a function to its arguments, thus encoding a function call via `FunctionId`. There are two examples in section 4.4.1 and 4.4.2.

**Policy:**

The *Rules* can not be exchanged among system components. PAP is responsible for combining *rules* in a *policy*. As shown in figure 4-16, a *policy* comprises following components:

- a *Target*
- a set of *Rules*
- a *Rule-combining algorithm* identifier
- *Obligations*

Same with the *target* that we discussed within the *rule* element, the *policy target* will play the role while the *rule target* is absent. In case that *rule target* is absent or the *rule* and *policy* have the same *target*, the *rule* elements inherit the *target* of the *policy* in which they are contained. It always happens when we combine several rules in a policy.
**Rule-combining algorithm** indicates the method that how to combine the rule evaluation results when evaluating a policy. The standard rule-combining algorithms include “deny-overrides (ordered and unordered)”, “permit-overrides (ordered and unordered)”, “first-applicable” and “only-one-applicable”.

“Deny-overrides” algorithm: If a single <Rule> element is encountered that evaluates to “Deny”, then regardless of the evaluation result of the other <Rule> elements in the applicable policy, the combined result is “Deny”.

“Permit-overrides” algorithm: If a single “Permit” result is encountered, then regardless of the other <Rule> elements, the combined result is “Permit”.

“First-applicable” algorithm: The combining result is the same as the result of evaluating the first <Rule> element in the list of rules whose target is applicable to the decision request.

Policy may also contain obligations element. When the effect of policy or policy set is evaluated match the value of “FulfillOn” attribute of the obligation, it returns certain of obligations that should be fulfilled with the policy decision, to PEP in response.

**Policy Set:**

As shown in figure 4-17, a policy set include following components:

- a Target
- a set of policies
- a policy-combining algorithm identifier
- Obligations

![Diagram of XACML PolicySet Type]

**Figure 4-17 Main elements of “PolicySet”**

Similar with the policy is the set of rules, the policy set is the set of policies or policy sets. Most usage of elements within the policy set are similar to which have been discussed within policy element.

Different with rule-combining algorithm, policy-combining algorithm has one more algorithm: “Only-one-applicable”. This algorithm only applies to policies. The result of this combining algorithm ensures that one and only one policy or policy set is applicable in terms of their targets. If no policy or policy set applies, then the result is “NotApplicable”, but if more than one policy or policy set is applicable, then the result is “Indeterminate”. When exactly one policy or policy set is applicable, the result of the combining algorithm is the result of evaluating the single applicable policy or policy set.

**Request, Response message:**

The request message consists of three mandatory elements Subject, Resource and Action, and optionally may have Environment element. Same method with the extending that performed in policy language model, the request message also need be able to represent context information under its four elements: Subject, Resource, Action and Environment. An initiator’s request message can be sent with SOAP HTTP channel.
The response message is same with standard XACML format. From the discussion of next section 4.4 (Evaluation Process), we can found that the decision result within response message maybe “Permit”, “Deny”, “NotApplicable” or “Indeterminate”, which made by PDP. It also convey status information of the decision making process. For the security purpose, we recommend use “Deny” replace “Indeterminate” and “NotApplicable” result as the final access control decision.

4.4 Evaluation Process

We already have the context aware access control policy language model, but how to use the policy and how to evaluate the policy to make the authorization decision. Within this section, the policy evaluation process is represented.
4.4.1 Attribute evaluation

In our model, attributes and context information are represented in the request message by deriving from original access request and CASPer Finder. They are referred to policy by subject, resource, action and environment attribute designators and attribute selectors. A named attribute within the policy is the term used and specified by subject, resource, action and environment attribute designators and selectors, use to refer to particular attributes in the subject, resource, action and environment elements of the request message.

Original XACML defines implicit collections of its data types. XACML refers to a collection of values that are of a single data type as a bag. Bags of data types are needed because request message or selections of nodes from an XML resource may return more than one value. The <AttributeDesignator> element illustrates matching methodology for attributes in the request message. XACML also defines <AttributeSelector> element uses an XPath expression to specify the selection of data from an XML resource.

Attribute matching

A named attribute within policy includes specific criteria with which to match attributes in the request message. An attribute specifies an AttributeId and DataType, and a named attribute also specifies the Issuer, however the Issuer is optional. A named attribute shall match an attribute if the values of their respective AttributeId, DataType and optional Issuer attributes match within their particular element: subject, resource, action or environment of the request message. The AttributeId of the named attribute must match the AttributeId of the corresponding request message attribute by URI equality. The DataType of the named attribute must match the DataType of the corresponding request message attribute by URI equality. More about matching method is presented in 4.4.2.

4.4.2 Match evaluation

The match evaluation provides the methods that used within target (attributes) evaluation and expression evaluation. Attribute matching elements appear in the <Target> element of rules, policies and policy sets. The attribute matching elements include: <SubjectMatch>, <ResourceMatch>, <ActionMatch> and <EnvironmentMatch>.

These elements represent Boolean expressions over attributes of the subject, resource, action and environment, respectively. A matching element contains a MatchId attribute that specifies the function to be used in performing the match evaluation, an <AttributeValue> and an <AttributeDesignator> or <AttributeSelector> element, which specifies the attribute in the request message that is to be matched against the specified value.

The MatchId attribute shall specify a function that compares two arguments, and return a Boolean type result (http://www.w3.org/2001/XMLSchema#boolean). The attribute value specified in the matching element shall be supplied to the MatchId function as its first argument. The element of the bag returned by the <AttributeDesignator> or <AttributeSelector> element shall be supplied to the MatchId function as its second argument. The DataType of the <AttributeValue> shall match the data type of the first argument expected by the MatchId function. The DataType of the <AttributeDesignator> or <AttributeSelector> element shall match the data type of the second argument expected by the MatchId function.

The XACML standard functions that represented by MatchId are:
- urn:oasis:names:tc:xacml:2.0:function::type-equal
- urn:oasis:names:tc:xacml:2.0:function::type-greater-than
- urn:oasis:names:tc:xacml:2.0:function::type-greater-than-or-equal
- urn:oasis:names:tc:xacml:2.0:function::type-less-than
- urn:oasis:names:tc:xacml:2.0:function::type-less-than-or-equal
- urn:oasis:names:tc:xacml:2.0:function::type-match.

The evaluation semantics for a matching element is as follows:
- If an operational error occurs while evaluating the `<AttributeDesignator>` or `<AttributeSelector>` element, then the result of the entire expression shall be "Indeterminate".

- If the `<AttributeDesignator>` or `<AttributeSelector>` element were to evaluate to an empty bag, then the result of the expression shall be "False".

- Otherwise, the `MatchId` function shall be applied between the `<xacml:AttributeValue>` and each element of the bag returned from the `<AttributeDesignator>` or `<AttributeSelector>` element. If at least one of those function applications were to evaluate to "True", then the result of the entire expression shall be "True".

- Otherwise, if at least one of the function applications results in "Indeterminate", then the result shall be "Indeterminate".

- Finally, if all function applications evaluate to "False", then the result of the entire expression shall be "False".

Moreover, it is also possible to express the semantics of a target matching element in a `<condition>`.

### 4.4.3 Expression evaluation

The expression evaluation is the main method for context evaluation. The `policy` specifies expressions in terms of the elements listed below, of which the `<Apply>` and `<Condition>` elements recursively compose greater expressions. These elements include:

- `<xacml:AttributeValue>`
- `<xacml:SubjectAttributeDesignator>`
- `<xacml:ResourceAttributeDesignator>`
- `<xacml:ActionAttributeDesignator>`
- `<xacml:EnvironmentAttributeDesignator>`
- `<xacml:AttributeSelector>`
- `<xacml:Apply>`
- `<xacml:Condition>`
- `<xacml:Function>`

The function of `<AttributeValue>`, `<AttributeDesignator>` and `<AttributeSelector>` has been mentioned in above section. The types of each of the elements contained within `<Apply>` and `<Condition>` elements shall agree with the respective argument types of the function that is named by the `FunctionId`. The resultant type of the `<Apply>` or `<Condition>` element shall be the resultant type of the function. Original XACML defines an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an operational error occurring during the evaluation of the expression. For the `rule` to be applicable, the `condition` must evaluate to “True”. We define the context constrains as part of the `condition` so that let context information affect the policy decision.

### 4.4.4 Target evaluation

We have represented the evaluation methods above, and then we can use them together to perform the evaluation on `policies`. In fact the evaluation of `policy` is a set of operations to combine policy’s contents (target and rules) evaluation results. So we will represent how to evaluate `<Target>` element firstly.

The `target` value shall be “Match” if the subjects, resources, actions and environments specified in the `target` all match values in the request message. If any one of the subjects, resources, actions and environments specified in the `target` are “Indeterminate”, then the `target` shall be “Indeterminate”. Otherwise, the `target` shall be “No match”. The match rules are shown in table 4-4.
Table 4-4 Target evaluation match table

<table>
<thead>
<tr>
<th>Subjects value</th>
<th>Resources value</th>
<th>Actions value</th>
<th>Environments value</th>
<th>Target value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Match”</td>
<td>“Match”</td>
<td>“Match”</td>
<td>“Match”</td>
<td>“Match”</td>
</tr>
<tr>
<td>“No match”</td>
<td>“Match” or “No match”</td>
<td>“Match” or “No match”</td>
<td>“Match” or “No match”</td>
<td>“No match”</td>
</tr>
<tr>
<td>“Match” or “No match”</td>
<td>“No match”</td>
<td>“Match” or “No match”</td>
<td>“Match” or “No match”</td>
<td>“No match”</td>
</tr>
<tr>
<td>“Match” or “No match”</td>
<td>“Match” or “No match”</td>
<td>“No match”</td>
<td>“Match” or “No match”</td>
<td>“No match”</td>
</tr>
<tr>
<td>“Match” or “No match”</td>
<td>“Match” or “No match”</td>
<td>“Match” or “No match”</td>
<td>“No match”</td>
<td>“No match”</td>
</tr>
<tr>
<td>“Indeterminate”</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>“Indeterminate”</td>
</tr>
<tr>
<td>Don’t care</td>
<td>“Indeterminate”</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>“Indeterminate”</td>
</tr>
<tr>
<td>Don’t care</td>
<td>Don’t care</td>
<td>“Indeterminate”</td>
<td>Don’t care</td>
<td>“Indeterminate”</td>
</tr>
<tr>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>“Indeterminate”</td>
<td>“Indeterminate”</td>
</tr>
</tbody>
</table>

The *subjects*, *resources*, *actions* and *environments* shall “match” values in the *request message* if at least one of their `<Subject>`, `<Resource>`, `<Action>` or `<Environment>` elements matches a value in the *request message*. Likewise, a *subject*, *resource*, *action* or *environment* shall match a value in the *request message* if the value of all its `<SubjectMatch>`, `<ResourceMatch>`, `<ActionMatch>` or `<EnvironmentMatch>` elements, respectively, are “True”.

### 4.4.5 Rule evaluation

*Rule evaluation* involves separate evaluation of the *rule’s target* and *condition*. After we get the target and condition evaluation result via the evaluation methods mentioned above, a *rule* evaluation result can be calculated by evaluating its contents. The *rule* evaluation truth table is shown in Table 4-5.

Table 4-5 Rule evaluation truth table:

<table>
<thead>
<tr>
<th>Target</th>
<th>Condition (include the context evaluation)</th>
<th>Rule Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Match”</td>
<td>“True”</td>
<td>Effect</td>
</tr>
<tr>
<td>“Match”</td>
<td>“False”</td>
<td>“NotApplicable”</td>
</tr>
<tr>
<td>“Match”</td>
<td>“Indeterminate”</td>
<td>“Indeterminate”</td>
</tr>
<tr>
<td>“No-match”</td>
<td>Don’t care</td>
<td>“NotApplicable”</td>
</tr>
<tr>
<td>“Indeterminate”</td>
<td>Don’t care</td>
<td>“Indeterminate”</td>
</tr>
</tbody>
</table>

### 4.4.6 Policy and Policy Set evaluation

The policy evaluation involves separate evaluation of the *policy’s target* and *rules*. The *policy’s target* shall be evaluated to determine the applicability of the *policy*. If the *target* evaluates to “Match”, then the value of the *policy*
shall be determined by evaluation of the policy’s rules, according to the rule combining algorithm. If the target evaluates to “No-match”, then the value of the policy shall be “NotApplicable”. If the target evaluates to “Indeterminate”, then the value of the policy shall be “Indeterminate”.

Table 4-6 Policy evaluation truth table:

<table>
<thead>
<tr>
<th>Target</th>
<th>Rule values</th>
<th>Policy Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Match”</td>
<td>At least one rule value is its Effect</td>
<td>Specified by the rule-combining algorithm</td>
</tr>
<tr>
<td>“Match”</td>
<td>All rule values</td>
<td>“NotApplicable”</td>
</tr>
<tr>
<td>“Match”</td>
<td>At least one rule value is “Indeterminate”</td>
<td>Specified by the rule-combining algorithm</td>
</tr>
<tr>
<td>“No-match”</td>
<td></td>
<td>“NotApplicable”</td>
</tr>
<tr>
<td>“Indeterminate”</td>
<td></td>
<td>“Indeterminate”</td>
</tr>
</tbody>
</table>

Furthermore, the policy set evaluation process is similar to policy evaluation. The evaluation result value of a policy set shall be determined by evaluation of its contents: policy set’s target, policies and policy sets, according to the specified policy combining algorithm. However, if the final evaluation result of policy or policy set is “Indeterminate”, we suggest use “Deny” to replace it for the security reason as we mentioned before.

Table 4-7 Policy Set evaluation truth table:

<table>
<thead>
<tr>
<th>Target</th>
<th>Policy values</th>
<th>Policy Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Match”</td>
<td>At least one policy value is its Decision</td>
<td>Specified by the policy-combining algorithm</td>
</tr>
<tr>
<td>“Match”</td>
<td>All policy values</td>
<td>“NotApplicable”</td>
</tr>
<tr>
<td>“Match”</td>
<td>At least one policy value is “Indeterminate”</td>
<td>Specified by the policy-combining algorithm</td>
</tr>
<tr>
<td>“No-match”</td>
<td></td>
<td>“NotApplicable”</td>
</tr>
<tr>
<td>“Indeterminate”</td>
<td></td>
<td>“Indeterminate”</td>
</tr>
</tbody>
</table>

4.4.7 Compare evaluation process of two extended policy language model solutions

While the system collects context information and represents it in the request message, there are some differences of policy evaluation process with our two language model solutions represented in section 4.3.1 and 4.3.2. For the solution 1, the context information, which derived from Context Broker and provided by CASPEr Finder, will be categorized according to the context information structure of figure 4-7, and mapped into <Context Information> element of request message. The different type context information is represented by corresponding <Context Object>. Consequently, the evaluation process needs to handle more matching and operations due to there are more sub elements of <Target> besides <Subject>, <Resource>, <Action> and <Environment>. Furthermore, new <ContextInformationAttributeDesignator> is needed for the evaluation process.
For the solution 2, the context information will be categorized according to the structure illustrated in figure 4-8, and mapped into different context information attribute type within subject, resource, action and environment elements of request message. According to the above mentioned policy evaluation methods in section 4.4.3, we are able to evaluate the context constrain that is represented via <Condition> element of the rules. This method is more direct and simpler compared with solution 1. It's even need not to change the evaluation process, just need to map the context information into new context information attributes of subject, resource, action and environment, furthermore to provide corresponding operation functions. Compared with solution 1, the solution 2 has the following advantages:

- Simplicity. The context information is classified into “subject context” and “resource context” directly according to the original language structure of XACML. This context category method is sufficient for our context information representation requirement for context aware access control.

- It makes less change to the original XACML language model. The context information can be mapped into different context information attributes of subject, resource, action and environment element. So we just need to provide new attribute type (datatype) and corresponding operation functions to support the context constrains evaluation.

- The existed open source implementation of XACML can be utilized by extending it according to our method, so that to shorten the development period and reduce the implementation cost.

In the next section, two policy examples and corresponding explanations are described to explain how the context aware access control policy works.

### 4.5 CASPEr Access Control Policy Example

In this section, we present two CASPEr access control policy examples. These policy examples illustrate how the context aware access control works with context information. The first one is <Condition> part of a policy, which shows us how to via the function within <Condition> element and <Apply> sub element to evaluate the context constrain. Then a medical care policy is described in second example.

#### 4.5.1 CASPEr Context Constrain

This example is not a full policy but only <Condition> part. Within this example, we evaluate “time” context constrain via functions within <Condition> element. When we consider the policy such as the access can be perform between 9:00 and 17:00, the Context constrain:= ( CurrentTime >= 9:00 and CurrentTime <= 17:00 ). So it can be described via <Condition> and <Apply> element.

```xml
... 01 <Condition>
02   <Apply FunctionId="&function;and">
03       <Apply FunctionId="&function;time-greater-than-or-equal">
04           <Apply FunctionId="&function;time-one-and-only">
05               <EnvironmentAttributeDesignator
06                   AttributeId="&environment;current-time"
07                       DataType="&xml;time"/>
08       </Apply>
09        <AttributeValue
10           DataType="&xml;time">09:00:00</AttributeValue>
11   </Apply>
12 </Apply>
13 <Apply FunctionId="&function;time-less-than-or-equal">
14      <Apply FunctionId="&function;time-one-and-only">
15         <EnvironmentAttributeDesignator
16             AttributeId="&environment;current-time"
17               DataType="&xml;time"/>
18      </Apply>
19      </Apply>
```
Within this time constrain example,

**Line 01** starts `<Condition>` element. The *condition* must evaluate to “True” for the *rule* to be applicable. It contains embedded `<Apply>` element.

**Line 02** `<Apply>` element introduces operation function “and” that used to connect the lower and upper boundary of time constrain.

**Line 03** introduces the operation function “≥”.

**Line 04 – 10** compares values of two sub elements. The first element uses “time-one-and-only” to ensure that the value selected by environment context exactly one value of type “&xml;time” (http://www.w3.org/2001/XMLSchema#time”). The current time is evaluated by selecting ”&environment;current-time” via EnvironmentAttributeDesignator.

**Line 12 - 20** introduces the operation function “≤”.

**Line 22** closes the `<Condition>` element.

### 4.5.2 CASPeR medical care access control policy

In this part, we illustrate a full access control policy:

- **Rule 1:** During work time (between 9:00 AM and 5:00 PM), allow any healthcare staffs who is close to patient to read patient’s medical recorder.
- **Rule 2:** On other time, only the authorized person (such as whose email address is …@physician.example.com) is able to read patient’s medical recorder."

The policy is present via extended XACML as following:

```
<?xml version="1.0" encoding="UTF-8"?>
PolicyId="research:CASPeR:example:Medical:Recorder:Access:Policy"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:ordered-permit-overrides">
  <Description>
    Between 9am and 5pm local time, allow any healthcare staff who is close to patient to read patient’s medical recorder. All other times, only allow authorized people (ie, those with an email address @physician.example.com) read patient’s medical recorder. Deny all other cases.
  </Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"
```
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  urn:example:med:schemas:patient-medical-recorder
</AttributeValue>
<ResourceAttributeDesignator
  DataType="http://www.w3.org/2001/XMLSchema#string"
  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"/>
</ResourceMatch>
</Resource>
</Resources>
<Actions>
  <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
      <ActionAttributeDesignator
        DataType="http://www.w3.org/2001/XMLSchema#string"
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
    </ActionMatch>
  </Action>
</Actions>
</Target>

<Rule RuleId="AnyDuringWorkHoursDistanceCloseTo" Effect="Permit">
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment;current-time"
            DataType="http://www.w3.org/2001/XMLSchema#time"/>
        </Apply>
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment;current-time"
            DataType="http://www.w3.org/2001/XMLSchema#time"/>
        </Apply>
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
      </Apply>
    </Apply>
    <Apply FunctionId="http://research.casper.com/function#distance-less-than">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator
          DataPositionType="http://research.casper.com/datatype/casper#location"
          AttributeId="casper:xacml:subject:current-location"/>
      </Apply>
      <ResourceAttributeDesignator
        DataPositionType="http://research.casper.com/datatype/casper#location"
        AttributeId="casper:xacml:resource:current-location"/>
      <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
    </Apply>
  </Condition>
</Rule>
Within this policy example:

**Line 01** is a standard XML document tag indicating the XML version and the character encoding type.

**Line 02 - 03** introduces the XACML policy, which includes XML namespace, URN of XACML policy schema. It also specifies the name of this policy, which indicated by PolicyId. The name of policy has to be unique for a given PDP so that there is no ambiguity if one policy is referenced from another policy. Furthermore, the rule combining algorithm is also specified by RuleCombiningAlgId. The ordered-permit-overrides algorithm says that if any rule is evaluated to “Permit” according to the rules order, then the policy returns “Permit”, and if all rules are evaluated to “Deny”, then the policy returns “Deny”. The description of rule combining algorithms are represented in section 4.3.3.

**Line 04 - 09** is the text description of policy.

**Line 10 - 37** describes the target of the policy. If the subject, resource, action and environment in a decision request don’t match the values specified in the policy target, then the remainder of the policy does not need to be evaluated.

**Line 11 - 13** indicates the subjects of the policy. In our example, the <AnySubject/> means we don’t point out a specific subject, and constrains of the subjects will be specified via <Condition> part in following rules.

**Line 14 - 25** represents the Resources of the policy. Resources element contains one or more attributes of the resource to which the subject(s) has want to access. The resource is patient medical recorder within our example. The match compares the target namespace of the requested document with the value “urn:example:med:schemas:patient-medical-recorder”.

**Line 16** The MatchId indicates the matching function.

**Line 18** is the Literal attribute value to match.

**Line 20 - 22** The <ResourceAttributeDesignator> element selects the target namespace from the resource contained in the request. The attribute name is specified by the AttributeId. Other than the target-namespace, the AttributeId also can be resource-id, URI or xpath.
Line 26 - 36 represents the access Actions that can be performed. The <ActionMatch> element compares it first (<AttributeValue>) and second (<ActionAttributeDesignator>) child elements according to the matching function that specified by MatchId. The match is positive if the value of the first argument matches any of the values selected by the second argument. In our example, the value of the action-id attribute in the request is compared with the literal value “read”.

Line 38 - 72 represents the rule 1. Within rule 1, the <Target> is omitted. This means rule inherits the <Target> of policy.

Line 38 “RuleId” identify the rule 1’s identifier. The “Effect” declares the rule effect is “Permit”, which means it emits the value “Permit” when a rule evaluates to “True”. This value is then combined with the “Effect” values of other rules according to the rule combining algorithm that identified by policy.

Line 39 - 71 The <Condition> element. For the rule to be applicable to request, the condition must evaluate to “True”. The condition in this example evaluates whether current time is between 9:00 and 17:00, and whether the subject is close to patient.

Line 40 - 59 Using time context to set condition. We use “time-greater-than-or-equal” and “time-less-than-or-equal” functions to specify that the access time constrain is between 9:00 and 17:00.

Line 60 - 70 Using location context to set condition. We use “distance-less-than” function to specify that the distance between subject and resource should less than 5 meters.

Line 72 closes rule 1.

Line 73 - 93 represents the rule 2. The RuleId is identified in line 73, and the Effect is “Permit” if rule 2 evaluates to “True”.

Line 75 - 85 The <Subjects> element within rule Target identify that only the person with the physician role (any user with an email name in the “physician.example.com” namespace) is allowed to perform the action “read” on resource (patient medical record) specified by the policy.

Line 93 closes rule 2.

Line 94 closes the policy.

4.6 Conclusion

Within this chapter, a general CASPEr model is designed and applied on context aware access control. Based on our survey, we decide to use XACML as the basis of CASPEr access control policy language, because it has many good features (such as extensible datatype, rich set of operators, standard semantic structure…) that are more appropriate for introducing context information. In the following we provide two methods of extending XACML policy language model to be context aware. We prefer solution 2 because it makes the less change of XACML, and it will let the implementation simple. Finally, two policy examples are presented to demonstrate how the context aware access control policy works.
5 Conclusion and Future Work

5.1 Conclusion

Recently, more and more attention is paid to researches and applications on context awareness. But context awareness is still a new area to security field. Our work provides architecture for context aware security policy enforcement. The results are based on the survey and research on state-of-the-art of context awareness and context-aware security. The survey results demonstrate that more and more technologies will support the context aware applications, and it is possible to introduce context awareness into security mechanism. Context aware security will become an enhancement of security mechanisms. However, most security related researches on context awareness are mainly focus on privacy protection, and very limited research are carried out on context aware security, it is still a new open topic.

With respect to our contributions, we surveyed the state-of-the-art of context awareness and context-aware security firstly. We identified the requirements of CASPeR system, and then designed a generic CASPeR model according to these requirements and further analysis. Furthermore, we applied this model on access control mechanism; because policy based access control is a basic important security mechanism that could produce added value via introducing context awareness. In succession, the possible policy language candidates were compared, and we demonstrated to use XACML as the basis for context aware access control policy language, and extended it to become context aware.

In this thesis, we represented the generic CASPeR model, and extension solution of XACML to be CASPeR access control policy language model. Moreover, we also illustrated the implementation methods based on the sun XACML implementation within appendix.

We believe that context aware security will get more and more attention within context awareness and security field. Our CASPeR model solution is valuable for providing more advanced and seamless security configuration, which adapting to dynamic context information. It will be useful to common users who are not experts on computer and information security field.

5.2 Future Work

In the future, there are many works could be extended. CASPeR could be adapted to more mechanisms other than access control, such as communication security set and trust management.

Moreover, the security of context aware system should be taken into account in more details, especially on trust and quality of context information. We represent a treat model and some safeguard methods in appendix B, and there should be more protection methods on the sensor side. Because sensors could be attacked easily to let it providing inaccurate context information, which may affect the policy decision result.

On the other hand, we might consider making CASPeR to cooperate with P3P in order to provide more functions on privacy protection.

Furthermore, as the methods discussed in Appendix A, the Sun’s Java open source solution of XACML can be modified and extended for CASPeR implementation.
References


[Bha04] Rafae Bhatti, Elisa Bertino, Arif Ghafoor, “A Trust-based context-aware access control model for web-services”, Center for Education and Research in Information Assurance and Security, Purdue University, Aug. 2004


[Che04] Harry Chen et al., “Intelligent Agents Meet the Semantic Web in Smart Spaces”, IEEE Internet Computing, November 2004


http://www.w3.org/2004/08/ws-cc/rbcc-20040902


[Jul04] Christine Julien, Jamie Payton, and Gruia-Catalin Roman “Context-Sensitive Access Control for Open Mobile Agent Systems” Department of Computer Science and Engineering Washington University in St. Louis, 2004


[Mic03] E.F. Michiels, “Telematics systems security”, Sept 2003, Distributed and Embedded systems group, University of Twente


[Zui03] Martijn Zuidweg, José Gonçalves Pereira Filho, Marten van Sinderen “Using P3P in a web services-based context-aware application platform” Centre for Telematics and Information Technology, University of Twente, 2003
Appendix A Implementation Blueprint

One of the reason why we choose XACML as the access control policy language is that there is implementation to support the XACML specification, known as Sun java XACML implementation (http://sunxacml.sourceforge.net/). This open-source implementation is possible to be extended to adapt to context-aware access control. We will introduce the structure of the XACML implementation, and demonstrate how to extend it for context-aware usage.

A.1 Overview of APIs

The API is broken into several packages:

- **xacml (include PDP class)**
  The core package, which contain the logic for Target matching, rule evaluation (include context evaluation), policy and policyset handling and other related features. It contains the PDP class, which is the entry point for other components.

- **xacml.attr**
  The package supports all the standard XACML attribute data types, as well as designators, selectors, and factors that used to create new attribute values. Standard interfaces and abstract class are provided to define new attributes types.

- **xacml.combine**
  The package includes all the combination algorithms.

- **xacml.cond**
  Provides all the condition and function logic, support all the standard functions. Furthermore, standard interfaces and classes are provided for define new functions, such as context evaluation.

- **xacml.ctx**
  The package represents the types define in XACML context schema, such as the request and response format. It can be used to build a simple PEP.

- **xacml.finder**
  This package supports for retrieving things that the PDP needs. It can be used for finding policies, retrieving attributes not provided in the request, resolving resource identifiers, or generating real-time values (such as context information). A set of standard classes lets the application writer use different kinds of finder modules to suit their tasks. So the finder will be responsible for collecting policy related context information from Context Broker.

![Figure A-1 Packages of XACML implementation](image)

A.2 The context-aware extensible point
Based on the basic API, we are able to extend the system to be context aware. The new, attribute types, functions, combining algorithms and finder modules could be produced via “factories” mechanism. In sun xacml implementation 1.2, we can define our won implementations of factories and use them in our code. The AttributeFactory, FunctionFactory and CombiningAlgFactory are located in xacml.attr, xacml.cond and xacml.combine separately.

We should note that in the most cases we need not to extend the functions, because XACML has provide rich set of standard functions for the comparison and match purpose. Moreover, most of the context reasoning work should have been done within Context Broker. We just need to add appropriate context attribute types and datatypes to represent context information within the policy. However we could extend new functions that when we need do some special evaluation of context information.

### A.2.1 Factory

Within the XACML implementation, a simple system of *factories* is used to support all datatypes, functions, and combining algorithms. There are generic factory instances, so it is possible to add new attribute-types, functions by defining your own implementations of factories. When we want to augment the default factory to support new datatype, function or algorithm, we should create a new factory instance which still supporting standard functionality.

For **datatypes**:

```java
StandardAttributeFactory standardFactory = StandardAttributeFactory.getFactory();
final BaseAttributeFactory newFactory = new BaseAttributeFactory(standardFactory.getStandardDatatypes());
...
// modify the contents of the new factory
...
AttributeFactory.setDefaultFactory(new AttributeFactoryProxy() {
    public AttributeFactory getFactory() {
        return newFactory;
    }
});
```

For **combining algorithms**:

```java
StandardCombiningAlgFactory standardFactory = StandardCombiningAlgFactory.getFactory();
Final BaseCombiningAlgFactory newFactory = New BaseCombiningAlgFactory (standardFactory.getStandardAlgorithms () );
...
// modify the contents of the new factory
...
combiningAlgFactory.setDefaultFactory (new CombiningAlgFactoryProxy() {
    public CombiningAlgFactory getFactory() {
        return newFactory;
    }
});
```

For **functions**, the method is more complex, but there is a convenient method is provided to do it:

```java
FunctionFactoryProxy proxy = StandardFunctionFactory.getNewFactoryProxy();
...
// modify the contents of the new factory through the proxy
...
FunctionFactory.setDefaultFactory(proxy);
```
When we discuss the extension of attribute type, function and algorithm in following part, the factories should be modified and setup as the methods mentioned above.

A.2.2 Add new attribute types

XACML supports rich set of standard datatypes such as strings, integers, dates, URIs etc.. Since all the attributes in XACML need to specify their type, it is easy to extend the default types, and include values of the new types in policies request and response. For our using, add a new (context) datatype is very easy.

All attribute types are as subclasses of the abstract `AttributeValue`. So we need to create a new class that extends `AttributeValue`, by providing the new type’s identifier, implementing the `encode`, `equals` and `hashCode` methods. Here is an example of an attribute that represents simple string value:

```java
public class NewSampleAttribute extends AttributeValue{
    public static final String Identifier = "http://research.CASPeR.com/datatype#newSampleString";
    private String value;
    public NewSampleAttribute(String value) throws URISyntaxException{
        super(new URI(Identifier));
        this.value = value;
    }
    public String encode() {
        return value;
    }
    public Boolean equals(object o) {
        return value.equals(o);
    }
    public int hashCode() {
        return value.hashCode() ;
    }
}
```

Via the above method, we already get a new type class. Then it needs to be added to the `AttributeFactory`, as we have talked about in section 7.2.1. The `AttributeProxy` is called each time a new value is created.

```java
    AttributeFactory factory = AttributeFactory.getInstance();
    factory.addDatatype(http://research.CASPeR.com/datatype#newSampleString, new AttributeProxy() {
        public AttributeValue getInstance(Node root) throws Exception {
            return NewSampleAttribute.getInstance(root);
        }
        public AttributeValue getInstance(String value) throws Exception {
            return new NewSampleAttribute (value);
        }
    });
```

A.2.3 Add new functions

XACML has already provide rich set of functions from simple Boolean operations (integer-equal, and, or, not, date-less-than etc.) to complex Bag-related functions. However, we are able to extend the new functions if this set of standard functions is not sufficient, such as when we need to handle some special context information.
FunctionBase class is an easy way to implement the Function interface to help add a new function. The new function can be evaluated against a set of inputs, can be asked if a set of inputs is acceptable, and provides information about its return type and identity. As an example, the following new function is added to take a Boolean and a string, and compares them to see if the text of string is equal to the value of the Boolean.

```java
Public class BoolTextCompare extends FunctionBase {

    public static final String NAME = "bool-text-compare";

    private static final String params [] = { BooleanAttribute.identifier, StringAttribute.identifier }

    private static final Boolean bagParams [] = {false, false};

    public BoolTextCompare() {
        super (NAME, 0, params, bagParams, BooleanAttribute.identifier, false);
    }

    public EvaluationResult evaluate (List inputs, EvaluationCtx context) {
        AttributeValue [] argValues = new AttributeValue[inputs.size()];
        EvaluationResult result = evalArgs(inputs, context, argValues);
        if (result != null)
            return result;
        BooleanAttribute bool = (BooleanAttribute) (argValues [0]);
        StringAttribute str = (StringAttribute) (argValues [1]);
        boolean evalResult;
        if (bool.getValue()) {
            evalResult = str.getValue().equals("true");
        } else {
            evalResult = str.getValue().equals("false");
        }
        return EvaluationResult.getInstance(evalResult);
    }
}
```

Like what we should do with new attribute type, we need to add this new function into the factory. We add the function to the factory as a Target function, so it can be used in a Target, a Condition or in any general situation.

```java
FunctionFactory factory = FunctionFactory.getTargetInstance();
factory.addFunction( new BoolTextCompare());
```

More detail of adding new functions can be found in [Sun’s XACML] and it’s JavaDocs of xacml.cond package part.

### A.3 Using SAML as transport solution for CASPER

#### A.3.1 Introduction

XACML itself defines the content of the messages necessary to implement the access control model, but does not define protocols or transport mechanisms. XML-based Security Assertion Markup Language (SAML) is a standard suitable for providing the assertion and protocol mechanisms for XACML. SAML defines schemas for requesting and responding with various types of security assertions. The schemas also include information needed to identify and validate the contents of the assertions, such as the identity of the issuer, the validity period of the assertion, and
the digital signature of the assertion. Moreover, SAML provide associated specifications that describe the binding to other standards, such as SOAP over HTTP. So we can use SAML to transmit the messages securely.

![Figure A-2 The SAML Assertion between XACML components](SAML-XACML)

When we apply SAML with XACML to protect, transport XACML schema instances, there are 6 types of queries and statements (also see Figure 5-2):

- **AttributeQuery** – A standard SAML Request used for requesting one or more attributes from an Attribute Authority (such as TCIP).
- **AttributeStatement** – A standard SAML Statement that contains one or more attributes. This statement may be used in a SAML Response from an Attribute Authority (such as TCIP).
- **XACMLPolicyQuery** – A SAML Request extension, new defined for XACML. It is used for requesting one or more policies from a Policy Administration Point (PAP).
- **XACMLPolicyStatement** – A SAML Statement extension, new defined for XACML. It may be used in a SAML Response from a PAP, or it may be used in a SAML Assertion as a format for storing policies in a policy repository.
- **XACMLAuthzDecisionQuery** – A SAML Request extension, new defined for XACML. It is used by a PEP to request an authorization decision from an XACML PDP.
- **XACMLAuthzDecisionStatement** – A SAML Statement extension, new defined in this profile. It may be used in a SAML Response from an XACML PDP.

### A.3.2 Attributes Mapping

The SAML assertion schema defines an Attribute Assertion. The SAML protocol schema defines an AttributeQuery used for requesting instances of Attribute Assertions, and a Response that contains the requested instances. Systems using XACML may use instances of these SAML elements and SAML AttributeQuery protocol to request, transmit and store attributes. In order to be used in an XACML Request Context, the SAML Attribute shall be mapped to an XACML Attribute.
A SAML Attribute Assertion is a `<saml:Assertion>` instance that contains one or more `<saml:AttributeStatement>` instances, each of which may contain one or more `<saml:Attribute>` instances. Each SAML Attribute in the SAML Attribute Assertion shall comply with XACML Attribute Profile, namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`, in SAML profiles [SAML-PROFILE].

When we use the SAML Assertion to transport attributes, the `<xacml-context:Attribute>` shall be constructed from the corresponding `<saml:Attribute>` element in a SAML Attribute Assertion as described following:

- **XACML AttributeId**
  The value of the `<saml:Attribute>` `Name` shall be used.

- **XACML DataType**
  The value of the `<saml:Attribute>` `DataType` shall be used. If the `<saml:Attribute>` `DataType` is missing, the XACML `DataType` shall be `http://www.w3.org/2001/XMLSchema#string`.

- **XACML Issuer**
  The string value of the `<saml:Issuer>` shall be used.

- **XACML `<xacml-context:AttributeValue>`**
  The `<saml:AttributeValue>` value shall be used.

We should note that not all `<saml:Attribute>` instances in SAML Assertion need to be mapped to `<xacml-context:Attribute>` element. The `Issuer` of the `<saml:Assertion>` is used as the `Issuer` for each `<xacml-context:Attribute>` element. The `<xacml-context: Attribute>`, which created from `<saml:Assertion>` shall be assigned to `<xacml-context:Resource>`, `<xacml-context:Subject>`, `<xacmlcontext:Action>`, or `<xacml-context:Environment>` element according to the `<saml:Subject>` in SAML Attribute Assertion. For instance, if the SAML Assertion Subject contains a `<saml:NameIdentifier>` element, which value matches the value of the `<xacml-context:Attribute>`'s `resource:resource-id`, then the created `<xacml-context:Attribute>` instances shall be assigned into the `<xacml-context:Resource>` element.

Furthermore, the semantics defined by SAML should be checked before performing the mapping. These checks include:

- **NotBefore and NotOnOrAfter** attributes in `<saml:Assertion>` shall be valid with respect to the `<xacml:Request>`. The `NotBefore` and `NotOnOrAfter` attribute values shall be consistent with the `current-time`, `current-date`, and `current-dateTime`.

- The mapping shall ensure that the semantics defined by SAML for any `<saml:AudienceRestrictionCondition>` or `<saml:DoNotCacheCondition>` elements have been adhered to.

- If there is `<ds:Signature>` in the SAML Assertion, then the mapping shall ensure that the signature is valid and the SAML `<Issuer>` element is consistent with any `<ds:X509IssuerName>` value in the signature.
A.3.3 Authorization Decisions

In order to use the SAML Request and Response syntax with full support the XACML Request Context and Response Context, two SAML extensions need to be defined.

- `<xacml-samlp:XACMLAuthzDecisionQuery>` is a SAML Query that extends the SAML Protocol Schema. It allows a PEP to submit an XACML Request Context in a SAML Request, along with other information.
•  `<xacml-saml:XACMLAuthzDecisionStatement>` is a SAML Statement that extends the SAML Assertion schema. It allows an XACML PDP to return an XACML Response Context in the Response to an `<XACMLAuthzDecisionStatement>`, along with other information. It also allows an XACML Response Context to be stored or transmitted in the form of a SAML Assertion.

A.3.4 Policies

SAML does not define any Protocol or Assertion schemas for policies. So two new SAML extensions: `<XACMLPolicyQuery>` and `<XACMLPolicyStatement>` need to be defined. The instances of the new elements can be used to request, transmit and store XACML `<Policy>` and `<PolicySet>` instances.

A.4 Conclusion

As described above, it is possible to extend the XACML implementation to be context aware via adding new context attribute types and adding new evaluation function if needed. Our suggestion is to complete the context reasoning work within Context Broker (such as via ContextToolkit) and avoid adding too much new evaluation function into the PDP. However, the extension of the evaluation function can be done if you need to handle some special context information within PDP.

Appendix B  Security consideration of Context Aware Security Policy Enforcement System

When we implement the CASPEr system, there may be many threats to the system itself. These threats could compromise the security policy decision enforced by the PEP. Furthermore, the threats may also expose confidential data like security policy and user’s context information, which may cause the privacy invasions.

B.1 Threat model

We assume there is an adversary that can compromise the communication channel between CASPEr components. This adversary might be able to eavesdrop, insert, delete and modify messages. Other possible vulnerable components of the system include PEP, PDP and PAP. Compromise of each of these components may lead to incorrect policy decisions. But since these components are not specific for context awareness, they are out of scope of our security analysis.
As shown in the figure 5-1, some old style attacks may compromise PEP, PDP, PAP and the communication when Finder collects attributes. There are other kinds of attacks that may compromise the context information acquisition process. The attacker may attack the context brokers and sensors directly, or try to compromise the communication on the perimeter of the CASPEr system. They may have following purposes:

- Try to affect the security policy decision of CASPEr;
- Try to intercept context information that may include privacy sensitive information, such as people’s location;
- Try to disrupt CASPEr system service itself, such as Deny-of-Service attack.

### B.1.1 Eavesdropping

The messages that transmitted between different components of CASPEr system should be protected to avoid eavesdropping. The unauthorized disclosure of the messages and security policies will compromise the security policy decision and the data or system that protected by CASPEr. Moreover, the disclosure of context information and other information in the subject request may relate to privacy problem. These kinds of disclosure may cause serious problem in case of the information related to medical recorder or commercial information.

Another possible attack should be considered is that attack may fake the input of the CASPEr system to let sensor collect some valuable context information (such as position), so this kind of information may be intercepted and captured by attacker.

### B.1.2 Message replay
The message replay attack may cause denial of service or impersonation. The adversary replays the legitimate messages between the components of CASPEr systems, such as access control request replay. Moreover, attacker may use context information replay during the context acquisition. We should note that the encryption method is no help for preventing message replay attack. For preventing attacks we could use freshness authentication to check it.

### B.1.3 Message insertion, deletion and modification

The adversary may insert or delete the messages transmitted between components of context aware security system. Furthermore, the adversary may intercept and capture a message between components, and change its content. The modification of the message may cause the alteration of policy decision and the information disclosure. Authentication, encryption, checking the message sequence integrity and message integrity are possible solutions for the Message attack.

The context aware system may also be attacked by modifying or inserting fake context information by attack the sensors, context information provider or communication at the system perimeter. The incorrect context information will lead to incorrect policy decision.

### B.1.4 NotApplicable results

The “NotApplicable” result means that PDP of context aware security system couldn’t address an appropriate policy match the input information and context information. The PDP couldn’t find the appropriate policy, or unavailable of context information may cause the result.

A set of restrict default security policy are recommended to be set to handle this kind of exception. For instance, within the access control, the “Deny” is recommended to return instead of the “NotApplicable”.

### B.2 How to protect CASPEr system

#### B.2.1 Authentication

For the CASPEr system, authentication should be performed between the context broker and context information provider, and between the different context brokers. The source of context information should be reliable and trusted.

Authentication methods such as digital signature or protected communication channel can be utilized. Authentication can be performed as part of the communication protocol that used to exchange the message between system components.

#### B.2.2 Confidentiality

Confidentiality ensures that the content of the message only can be read by the desired recipients. Others who capture the message while transmit are not able to understand it. Confidentiality of message in transmission and the confidentiality of the statement level should be addressed.

The communication confidentiality can be performed via secure communication mechanism such as SSL/TLS. Furthermore, the confidentiality of statement level also can be performed via XML Encryption Syntax and Processing Candidate Recommendation of W3C [XML Enc]. The XML Encryption can be used to encrypt all or parts of an XML document, such as the policy file, and the decision message.

#### B.2.3 Policy integrity
The policy decision of the system is evaluated depends on the context-aware security policy. So the integrity of the policy is important for the whole system. We need to ensure that the policy not be altered, inserted or deleted. This can be achieved by communication security mechanism (such as SSL) between the system components during the transmission process. Otherwise, the policy could be signed by issuer via XML Signature Syntax and processing standard of W3C [XML Sign] to ensure the integrity.

B.2.4 Protect context information

Within CASPeR system, the context information is essential. So we need to ensure the availability, integrity, privacy protection of context information. We need to authenticate the context information acquisition components, ensure the integrity and confidentiality of the message during the communication.

Table 5-1 Threats and safeguards of CASPeR

<table>
<thead>
<tr>
<th>Safeguard Threat</th>
<th>Authentication</th>
<th>Confidentiality (encryption)</th>
<th>Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack 1: Eavesdropping</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Attack 2: Message reply</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attack 3: Message insertion, deletion and modification</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Attack 4: Fake context information</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attack 5: Fake “request”, and eavesdropping CI</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Appendix C  XML schema for context aware extended XACML

access_control-xacml-2.0-policy-schema-CASPeR-extended.xsd (part)

```xml
<!—
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
  <xs:sequence>
    <xs:element ref="xacml:Subjects" minOccurs="0"/>
    <xs:element ref="xacml:Resources" minOccurs="0"/>
    <xs:element ref="xacml:Actions" minOccurs="0"/>
    <xs:element ref="xacml:Environments" minOccurs="0"/>
    <xs:element ref="xacml:ContextInformation" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<!—>
<xs:element name="Subjects" type="xacml:SubjectsType"/>
<xs:complexType name="SubjectsType">
  <xs:sequence>
    <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!—>
```
<xs:element name="Subject" type="xacml:SubjectType"/>
<xs:complexType name="SubjectType">
    <xs:sequence>
        <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
    <xs:sequence>
        <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
    <xs:sequence>
        <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="Actions" type="xacml:ActionsType"/>
<xs:complexType name="ActionsType">
    <xs:sequence>
        <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
    <xs:sequence>
        <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="Environments" type="xacml:EnvironmentsType"/>
<xs:complexType name="EnvironmentsType">
    <xs:sequence>
        <xs:element ref="xacml:Environment" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="Environment" type="xacml:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
    <xs:sequence>
        <xs:element ref="xacml:EnvironmentMatch" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="ContextInformation" type="xacml:ContextInformationType"/>
<xs:complexType name="ContextInformationType">
    <xs:sequence>
        <xs:element ref="xacml:ContextInformationObject" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="ContextInformationObject" type="xacml:ContextInformationObjectType"/>
<xs:complexType name="ContextInformationObjectType">
    <xs:sequence>
<xs:element ref="xacml:ContextInformationObjectMatch" maxOccurs="unbounded"/>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
<xs:complexType name="VariableDefinitionType">
  <xs:sequence>
    <xs:element ref="xacml:Expression"/>
  </xs:sequence>
  <xs:attribute name="VariableId" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="ExpressionType" abstract="true"/>

<xs:complexType name="VariableReferenceType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="VariableId" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="AttributeSelectorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="RequestContextPath" type="xs:string" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="AttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="Issuer" type="xs:string" use="optional"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="SubjectAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="SubjectId" type="xs:anyURI" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="ResourceAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="ResourceId" type="xs:anyURI" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="ActionAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="ActionId" type="xs:anyURI" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="EnvironmentAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="EnvironmentId" type="xs:anyURI" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="ContextInformationObjectAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="ObjectContextId" type="xs:anyURI" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
access_control-xacml-2.0-context-schema-CASPER-extended.xsd

<?xml version="1.0" encoding="UTF-8"?>
<!—
xs :element name="Request" type="xacml-context:RequestType"/>
<xs:complexType name="RequestType">
<xs:sequence>
  <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
  <xs:element ref="xacml-context:Resource" maxOccurs="unbounded"/>
  <xs:element ref="xacml-context:Action"/>
  <xs:element ref="xacml-context:Environment"/>
  <xs:element ref="xacml-context:ContextInformation"/>
</xs:sequence>
</xs:complexType>
<!—
xs :element name= "Response "  type= "xacml-context :ResponseType "/>
<xs:complexType name="ResponseType">
<xs:sequence>
  <xs:element ref="xacml-context:Result" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<!—
xs :element name= "Subject "  type= "xacml-context :SubjectType "/>
<xs:complexType name="SubjectType">
<xs:sequence>
  <xs:element ref="xacml-context:Attribute" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="SubjectCategory" type="xs:anyURI" default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
</xs:complexType>
<!—
xs:element name="Resource" type="xacml-context:ResourceType"/>
...
<!—
xs:element name="ResourceContent" type="xacml-context:ResourceContentType"/>
...
<!—
xs:element name="Action" type="xacml-context:ActionType"/>
...
<!—
xs:element name="Environment" type="xacml-context:EnvironmentType"/>
...
<!—
xs:element name="ContextInformation" type="xacml-context:ContextInformationType"/>
<xs:complexType name="ContextInformationType">
<xs:sequence>
  <xs:element ref="xacml-context:Attribute" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
Appendix D Overview of SAML

One standard suitable for providing the assertion and protocol mechanisms needed by XACML is the OASIS (SAML), Version 2.0 [SAML]. SAML defines schemas intended for use in requesting and responding with various types of security assertions. The SAML schemas include information needed to identify and validate the contents of the assertions, such as the identity of the assertion issuer, the validity period of the assertion, and the digital signature of the assertion. The SAML specification describes how these elements are to be used. In addition, SAML has associated specifications that define bindings to other standards. These other standards provide transport mechanisms and specify how digital signatures should be created and verified.

SAML information is expressed through security assertions. An assertion is defined as a statement (or declaration of facts) about a subject made by an issuer. It provides two XML schemas: one for the definition of security assertions and another for the definition of a request/response protocol used to exchange assertions. SAML also
defines bindings for using assertions with standard transport and messaging services (currently defines a binding for SOAP over HTTP). SAML provides three different kinds of assertion statements:

- Authentication: the subject has been authenticated by some means at a given time.
- Attribute: the subject is associated with the given attributes and values.
- Authorization Decision: response to an access request, whether the access has been granted or denied.

SAML assertions are issued by SAML authorities, so in relation to the previous assertions types, we find: authentication authorities, attribute authorities and policy decision points. Following figure shows a sample domain model provided in SAML.

---

**Appendix E XML schema extension of SAML**

**E.1 Authorization Decisions:**

**Element `<XACMLAuthzDecisionQuery>`**

The `<XACMLAuthzDecisionQuery>` element MAY be used by a PEP to request an authorization decision from an XACML PDP. It allows a SAML Request to convey an XACML Request Context instance.

```xml
<xs:element name="XACMLAuthzDecisionQuery"
    type="XACMLAuthzDecisionQueryType"/>
<xs:complexType name="XACMLAuthzDecisionQueryType">
    <xs:complexContent>
        <xs:extension base="samlp:RequestAbstractType">
            <xs:sequence>
                <xs:element ref="xacml-context:Request"/>
            </xs:sequence>
            <xs:attribute name="InputContextOnly"
                type="boolean"
                use="optional"
                default="false"/>
            <xs:attribute name="ReturnContext"
                type="boolean"
                use="optional"
                default="false"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The `<XACMLAuthzDecisionQuery>` element is of XACMLAuthzDecisionQueryType complex type. This element is an alternative to the SAML-defined `<samlp:AuthzDecisionQuery>` that allows a PEP to use the full capabilities of an XACML PDP.

The `<XACMLAuthzDecisionQuery>` element contains the following XML attributes and elements:

- **InputContextOnly** [Default “false”]

---
This XML attribute governs the sources of information that the PDP is allowed to use in making its authorization decision. If this XML attribute is “true”, then the authorization decision SHALL be made solely on the basis of information contained in the <XACMLAuthzDecisionQuery>; No external attributes MAY be used. If this XML attribute is “false”, then the authorization decision MAY be made on the basis of external attributes not contained in the <XACMLAuthzDecisionQuery>.

**ReturnContext** [Default “false”]
This XML attribute allows the PEP to request that an <xacml-context:Request> element be included in the <XACMLAuthzDecisionStatement> resulting from the request. It also governs the contents of that <xacml-context:Request> element. If this XML attribute is “true”, then the PDP SHALL include the <xacml-context:Request> element in the <XACMLAuthzDecisionStatement> element in the <XACMLResponse>. This <xacml-context:Request> element SHALL include all those attributes supplied by the PEP in the <XACMLAuthzDecisionQuery> that were used in making the authorization decision. The PDP MAY include additional attributes in this <xacml-context:Request> element, such as external attributes obtained by the PDP and used in making the authorization decision, or other attributes known by the PDP that may be useful to the PEP in making subsequent <XACMLAuthzDecisionQuery> requests.

If this XML attribute is “false”, then the PDP SHALL NOT include the <xacml-context:Request> element in the <XACMLAuthzDecisionStatement> element of the <XACMLResponse>.

**<xacml-context:Request>** [Required].
An XACML Request Context.

**Element <XACMLAuthzDecisionStatement>**
The <XACMLAuthzDecisionStatement> MAY be used by an XACML PDP to return a SAML Response containing an XACML Response Context to a PEP in response to an <XACMLAuthzDecisionQuery>. It may also be used in a SAML Assertion as a format for storage of an authorization decision in a repository.

```xml
<xs:element name="XACMLAuthzDecisionStatement" type="xacml-saml:XACMLAuthzDecisionStatementType"/>
<xs:complexType name="XACMLAuthzDecisionStatementType">
  <xs:complexContent>
    <xs:extension base="saml:StatementAbstractType">
      <xs:sequence>
        <xs:element ref="xacml-context:Response"/>
        <xs:element ref="xacml-context:Request" MinOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexType>
</xs:complexType>
```

The <XACMLAuthzDecisionStatement> element is of XACMLAuthzDecisionStatementType complex type. This element is an alternative to the SAML-defined <samlp:AuthzDecisionStatement> that allows a SAML Assertion to contain the full content of the response from an XACML PDP.

The <XACMLAuthzDecisionStatement> element contains the following elements:

**<xacml-context:Response>** [Required]
The XACML Response Context created by the XACML PDP in response to the <XACMLAuthzDecisionQuery>.

**<xacml-context:Request>** [Optional]
An <xacml-context:Request> containing XACML Attributes returned by the XACML PDP in response to the <XACMLAuthzDecisionQuery>. This element SHALL be included if the ReturnResponse XML attribute in
the `<XACMLAuthzDecisionQuery>` is “true”. This element SHALL NOT be included if the ReturnResponse XML attribute in the `<XACMLAuthzDecisionQuery>` is “false”.

*Element* `<XACMLAuthzDecisionQuery>` for a description of the XACML `<Attribute>` values that SHALL be returned in this element.

### E.2 Policies

*Element* `<XACMLPolicyQuery>`

The `<XACMLPolicyQuery>` element is used by a PDP to request one or more XACML Policy or PolicySet instances from an on-line Policy Administration Point as part of a SAML Request.

```xml
<xs:element name="XACMLPolicyQuery"
            type="XACMLPolicyQueryType"/>
<xs:complexType name="XACMLPolicyQueryType">
  <complexContent>
    <xs:extension base="samlp:RequestAbstractType">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="xacml-context:Request"/>
        <xs:element ref="xacml:Target"/>
        <xs:element ref="xacml:PolicySetIdReference"/>
        <xs:element ref="xacml:PolicyIdReference"/>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The `<XACMLPolicyQuery>` element is of XACMLPolicyQueryType complex type. The `<XACMLPolicyQuery>` element contains one or more of the following elements:

- `<xacml-context:Request>` [Any Number]
  Supplies an XACML Request Context. All XACML Policy and PolicySet instances applicable to this Request SHALL be returned. The concept of “applicability” in the XACML context is defined in the XACML 2.0 Specification [XACML].

- `<xacml:Target>` [Any Number]
  Supplies an XACML `<Target>` element. All XACML Policy and PolicySet instances applicable to this `<Target>` SHALL be returned.

- `<xacml:PolicySetIdReference>` [Any Number]
  Identifies an XACML `<PolicySet>` to be returned.

- `<xacml:PolicyIdReference>` [Any Number]
  Identifies an XACML `<Policy>` to be returned.

*Element* `<XACMLPolicyStatement>`

The `<XACMLPolicyStatement>` is used by a Policy Administration Point to return one or more XACML `<Policy>` or `<PolicySet>` instances in a SAML Response to an `<XACMLPolicyQuery>` SAML Request. The `<XACMLPolicyStatement>` may also be used in a SAML Assertion as a format for storing the `<XACMLPolicyStatement>` in a repository.

```xml
<xs:element name="XACMLPolicyStatement"
            type="xacml-saml:XACMLPolicyStatementType"/>
<xs:complexType name="XACMLPolicyStatementType">
  <!-- Snippet omitted for brevity -->
</xs:complexType>
```
The `<XACMLPolicyStatement>` element is of XACMLPolicyStatementType complex type. The `<XACMLPolicyStatement>` element contains the following elements. If the `<XACMLPolicyStatement>` is issued in response to an `<XACMLPolicyQuery>`, and there are no `<xacml:Policy>` or `<xacml:PolicySet>` instances that meet the specifications of the associated `<XACMLPolicyQuery>`, then there SHALL be no elements in the `<XACMLPolicyStatement>`.

`<xacml:Policy>` [Any Number]  
An `<xacml:Policy>` instance that meets the specifications of the associated `<XACMLPolicyQuery>`, if any.

`<xacml:PolicySet>` [Any Number]  
An `<xacml:PolicySet>` instance that meets the specifications of the associated `<XACMLPolicyQuery>`, if any.