Forschungsbericht Nr. 03 - 5

Implement Role-Based Access Control with Attribute Certificates

Wei Zhou, Christoph Meinel
Forschungsgruppe Institut für Telematik
FB IV-Informatik
Universität Trier, 54286, Trier, Germany
{zhou, meinel}@ti.uni-trier.de

IMPLEMENT ROLE-BASED ACCESS CONTROL WITH ATTRIBUTE CERTIFICATES

Wei Zhou, Christoph Meinel

FG Institut für Telematik, Universität Trier,
D 54286 Trier, Germany
{zhou, meinel}@ti.uni-trier.de

Abstract

Nowadays more and more activities are performed over the Internet. But as more people are involved in the transaction circle, security and authorisation control becomes one of the biggest concerns. Hence, we are motivated by the need to manage and to enforce a strong authorisation mechanism in large-scale Web-environment. As well known, Role Based Access Control (RBAC) provides some flexibility to security management. Public key Infrastructure (PKI) provides a strong authentication. Privilege Management Infrastructure (PMI) as a new technology can provide strong authorisation. To satisfy mentioned security requirements, we have established a role based access control infrastructure and developed a prototype that uses X.509 public key certificate (PKCs) and X.509 attribute certificates (ACs). Access control is performed by an access control policy that is written in XML, roles and policies are stored in ACs. PKCs and ACs are all stored in a LDAP server. Our prototype includes an access control engine and an administration tool. The access control engine is implemented by a Java servlet. The access control engine provides a service that mediates the data between the users and the database, which also executes the function of authentication and authorisation. The administration tool can create key pair, X.509 PKCs, X.509 ACs, manage users' information, and some other functions.

Keyword: Role Based Access Control, X.509, Public Key Certificates, Public Key Infrastructure, Attribute Certificates, Privilege Management Infrastructure, XML

1. Introduction

Nowadays more and more activities are performed over the Internet. This trend creates new business opportunities and posts new technical challenges. One of the most challenging problems in managing large networked systems is the complexity of security administration, particularly access control. The goal of access control is to counter the threat of unauthorised operations involving computer or communication systems. The traditional Access Control List (ACL) can not always provide satisfied quality of security management, when there are many subjects and objects. So new access control mechanisms are needed to be developed in order to cater for various applications in Internet.

Recently, research has been focused on Role Based Access Control (RBAC) [1,2]. The basic idea is that access privileges are given to roles rather than to individual users. The users get corresponding right from acting different roles. So RBAC can provide the missing flexibility to security management over the traditional approach which use user and group identifiers. In fact RBAC addresses many needs of the commercial and government sectors.

Another important technology that can be used for access control is Privilege Management Infrastructure (PMI). It was specified by the ITU-T and ISO/IEO [3]. The main function of PMI is providing a strong authorisation after the authentication has taken place. The basic data structure in a PMI is a X.509 attribute certificate (AC). Like Public Key Certificate (PKC) strongly binds a public key to its subject, AC strongly binds a set of attributes to its holder. PKI and PMI infrastructures are linked by information contained in the identity and attribute certificates. Some research and development efforts have been done in this area [4,5,6,7]. But these efforts are still in primary phase, and to day no authorisation mechanism is widely accepted.

We were motivated by the need of using PKI, PMI and RBAC concepts to construct an authorisation mechanism. Avoiding to start from zero, we have assessed several works that have done in this area [5,8,17,20,26], and we decided to adopt the PERMIS [5] model to start our work. The main idea of the PERMIS model is that user's role is stored in ACs, access control decisions are driven by an authorisation policy, and the authorisation policy is also stored in an AC. We have developed an access control engine that can make decisions according to an access control policy, and an administration tool, that can generate key pairs, X.509 PKCs, X.509 ACs and bind ACs to PKCs. A demo system has also been developed.

The paper is organised as follows. Section 2 overviews RBAC and PMI technologies. Section 3 presents the most important related works. Section 4 describes our approach while Section 5 describes the demo system. Finally, Section 6 summarises the conclusions and mentions some future work.

2. Introduction to the Basic Technologies Used

2.1 Introduction to Role Based Access Control (RBAC)

There are two basic types of access control mechanisms used to protect information from unauthorized access: discretionary access controls (DAC) and mandatory access controls (MAC). DAC permits the granting and revoking of access control privileges to be left to the discretion of the individual users. A DAC mechanism allows users to grant or revoke access to any of the objects under their control. MAC requires all those who create, access, and maintain information to follow rules, set by administrators. MAC is a means of restricting access to objects, based on the sensitivity of the information contained in the objects and the formal authorization of subjects to access information of such sensitivity. These policies for access control are not particularly well suited to the requirements of government and industry organizations, that process unclassified but sensitive information.

Role-based access control (RBAC) emerged rapidly in the 1990s as a proven technology for managing and enforcing security in large-scale enterprise wide systems. The central notion of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. This greatly simplifies management of permissions. Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted by new permissions, as new applications and systems are incorporated, and permissions can be revoked from roles as needed.

A general RBAC model was defined by Sandhu [1]. It is summarised in Figure 1. The model is based on three sets of entities called users (U), roles (R), and permissions (P). A user is a human being or an autonomous agent. A role is a job function or job title within the organisation with some associated semantics regarding the authority and responsibility conferred on a member of the role. A permission is an approval of a particular mode of access to one or more objects in the system.

The user assignment (UA) and permission assignment (PA) relations of Figure 1 are both many-to-many relationships (indicated by the double-headed arrows). A user can be a member of many roles, and a role can be assigned to many users. Similarly, a role can have many permissions, and the same permission can be assigned to different roles.

Role hierarchy (RH) in RBAC is a natural way of organizing roles to reflect the organisation's lines of authority and responsibility. The hierarchy is partially ordered, so it is reflexive, transitive, and anti-symmetric. Inheritance is reflexive because a role inherits its own permissions. Transitivity is a natural requirement in this context, and anti-symmetry rules out roles that inherit from one another and are therefore redundant.

Figure 1 shows a set of sessions S. Each session relates one user to possibly many roles. A user establishes a session during which he or she activates some subset of roles that he or she is a member of (directly or indirectly by means of the hierarchy). The double-headed arrow from a session to R indicates that multiple roles can be simultaneously activated. The permissions available to the user are the union of permissions from all roles activated in that session. Each session is associated with a single user, as indicated by the single-headed arrow from the session to U. This association remains constantly for the life of a session. A user may have multiple sessions open at the same time, each in a different window on the workstation screen for instance. Each session may have a different combination of active roles. The concept of a session equates to the traditional notion of a subject in access control. A subject (or sessions) with different permissions are active at the same time.

Finally, Figure 1 shows a collection of constraints. Constraints can apply to any of the preceeding components. An example of constraints are mutually disjoint roles, such as purchasing manager and accounts payable manager, where the same user is not permitted to be a member of both roles.

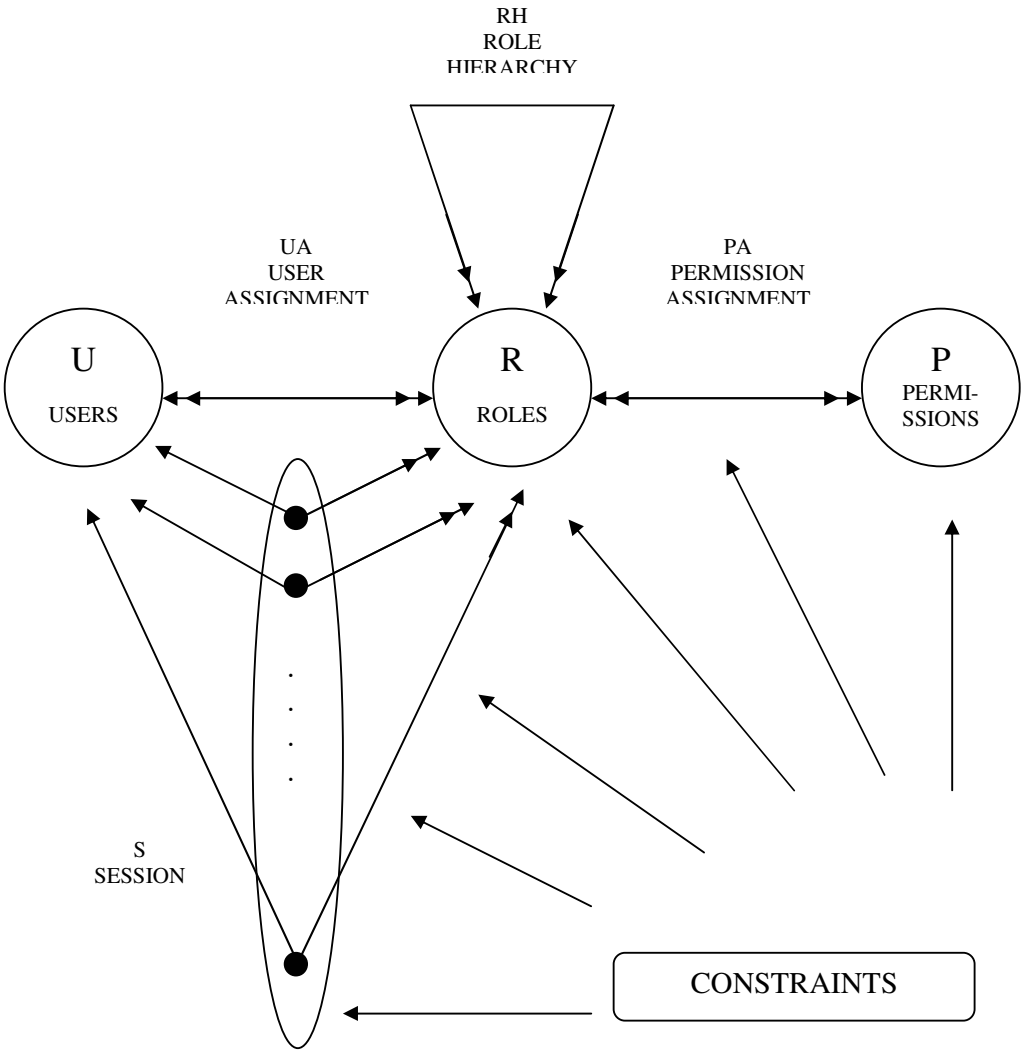

Figure 1. Basic RBAC model

RBAC is a proven alternative to traditional discretionary and mandatory access controls; it ensures that access is given to certain data or resources only to authorized users. It also supports some important security principles: least privilege, separation of duties, and data abstraction. Least privilege is supported, because RBAC can be configured such that only those permissions are assigned to the role required for the tasks conducted by members of the role. Separation of duties is achieved by ensuring that mutually exclusive roles must be invoked to complete a sensitive task, such as requiring an accounting clerk and account manager to participate in issuing a check. Data abstraction is supported by means of abstract permissions such as credit and debit for an account object, rather than read, write and execute permissions typically provided by the operating system

RBAC is policy neutrally. It is a means for articulating policy rather than embodying a particular security policy. Role is a semantic construct forming the basis of access

control policy. With RBAC, system administrators can create roles, grant permissions to those roles, and then assign users to the roles on the basis of their specific job responsibilities and policy. In particular, role-permission relationships can be predefined, making it simple to assign users to the predefined roles. Access control policy is embodied in RBAC components such as user-role, role-permission, and role-role relationships. These RBAC components determine whether a particular user is allowed to have access to a specific piece of system data. Moreover, the access control policy can evolve incrementally over the system life cycle. The ability to modify policy to meet the changing needs of an organisation is an important benefit of RBAC.

2.2. Introduction to X.509 (2000) Privilege Management Infrastructure (PMI)

Readers familiar with PKI will know that the standard format for a public key certificate (PKC) is specified in the X.509 standard, published jointly by the ITU-T and ISO/IEC [3]. During the past, practice has shown the need for additional granularity of the specification of certificates. Today, the current version v.3 of X.509 certificates allows to use a lot of extensions, where some of them can be used for propriety purposes, e.g. for defining attributes of an entity like roles, rights or authorisations information. But there are at least two important reasons why the placement of authorisation information in PKC is usually undesirable. First, authorisation information often does not have the same lifetime as the binding of the identity and the public key. For example, the persons authorised to perform a particular function in a company may vary monthly, weekly, or even daily. It is also possible that they are authorised to perform some temporary task for only several hours. Contrary to that, public key certificates are typically designed to be valid for a long time. Too frequently revoking and reissuing public key certificates will generate severe problem in certificate management system. Secondly, the PKC issuer is not usually authoritative for the authorisation information. It results in additional steps for the PKC issuer to obtain authorisation information from the authoritative source. Sometimes it is unacceptable or unpractical.

Recognising that X.509 public key certificates are not always the best vehicles for carrying authorisation information, the U.S. American National Standards Institute (ANSI) X9 committee developed an alternative approach, known as attribute certificates (ACs). Similar to public key certificate, that bind a public key to its subject, an AC binds a set of attributes to its holder. The released edition 4 of X.509 is the first edition of X.509 to fully standardize a strong authorisation mechanism, which it calls privilege management infrastructure (PMI). The primary data structure in PMI is X.509 attribute certificate (AC) [9] (see Figure 2). The ACs' issuer is called attribute authority (AA). ACs are digitally signed by the AA, so they are tamper-resistant. The trusted root is called source of authority (SOA). When a user's authorisation permissions need to be revoked, AA will issue an attribute certificate revocation list (ACRL) containing the list of ACs no long to be trusted.

The primary purpose of PMI is to provide a strong authorisation function after the authentication, provided by PKI, has taken place. That means the attribute certificates and public key certificates should be bound by some ways. One option is that the AC's holder field contains the PKC's serial number and issuer. Other options are using the entity name, for example the LDAP distinguished name, or the object digest

information, for example the hash value of the public key or the hash value of the PKC itself.

```
AttributeCertificate ::= SEQUENCE {
acinfo                  AttributeCertificateInfo,
signatureAlgorithm   AlgorithmIdentifier,
signatureValue        BIT STRING
}

AttributeCertificateInfo ::= SEQUENCE {
version                 AttCertVersion -- version is v2,
holder                  Holder,
issuer                  AttCertIssuer,
signature               AlgorithmIdentifier,
serialNumber            CertificateSerialNumber,
attrCertValidityPeriod   AttCertValidityPeriod,
attributes              SEQUENCE OF Attribute,
issuerUniqueID          UniqueIdentifier OPTIONAL,
extensions              Extensions OPTIONAL
}
```

Figure 2. Attribute certificate.

There are two primary models for distribution of attribute certificates: The “push” or “pull” model. In the “push” model, the ACs are pushed from the client to the server. In the “pull” model, the server pulls the ACs from a repository. The choice to use the “push” or the “pull” model depends on the system requirements. The “push” model is especially suitable in inter-domain cases where the client’s rights should be assigned within the client’s “home” domain, whereas the “pull” model is especially suitable for inter-domain cases where the client’s rights should be assigned within the server’s domain.

Attribute certificate strongly binds a set of attributes to its holder, and public key certificate strongly binds a public key to its subject. So in an application, attribute certificates depend on public key certificates to finish the authentication function. The typical verification sequence is: the application validates if a user's public key certificate is signed by a trusted CA, then it verifies the identity of the user by using his/her public key certificate. According to the information of user’s public key certificate it obtains his/her attribute certificates from a repository (in pull model), then it verifies if it is issued by a trusted AA and not expired. After the AC is proven a valid AC, the attributes within the attribute certificate will be extracted and used for access control or some other aims.

2.3 Lightweight Directory Access Protocol (LDAP)

User information is often a fragment across the enterprise. It leads to data that are redundant, inconsistent, and expensive to manage. Directories are viewed as one of the best mechanisms to make enterprise information available to multiple different systems within an organisation. Directories also make it possible for organisations to access information over the Internet. LDAP is a protocol that enables X.500-based directories to be read through Internet clients. It is a new technology for the Internet. The original idea is that LDAP server can store data and retrieve them later on, mostly in the same as a database. LDAP-server are mainly used in situations where that they are accessed much more often than they are updated [19]. It has many other advantages such as quick and advanced search, quick response, easy maintenance and a hierarchy view of data. Besides, LDAP server can be used in a large-scale network system over Transmission Control Protocol (TCP) / Internet protocol (IP) as well as in a distributed system. It also can be utilized to many other applications.

3. Related Work

3.1. Akenti Authorisation Infrastructure

Akenti [Akenti] is an authorisation infrastructure from the Lawrence Berkeley National Laboratory in the USA [8]. Akenti is an access control system designed to address the issues raised in permitting access to distributed resources that are controlled by multiple remote stakeholders. Akenti enables stakeholders securely to create and to distribute instructions authorising access to their resources. Akenti make access control decisions based on a set of digitally signed documents that represent the authorisation instructions. Public Key Infrastructure and secure message protocols provide confidentiality, message integrity, and user identity authentication, during and after the access decision process.

The Akenti policy is distributed and hierarchical. It comprises two components: Use-Condition certificates and Policy Certificates. Akenti issues attribute certificates to users that hold their privileges. But the ACs that are used by Akenti are not in a standard format. For more information about Akenti one can refer to http://dsd.lbl.gov/security/Akenti/.

3.2. PERMIS Research Project

PERMIS is an authorisation infrastructure from the European Commission (EC) funded PrivilEge and Role Management Infrastructure Standards validation (PERMIS) project [Permis]. The work was done at the Information Security Institute of University of Salford, UK. Its objective is to implement an X.509 role based Privilege Management Infrastructure (PMI), and it concentrates on solving identification and authorization problems [4, 5, 12]. The PERMIS group has examined various policy languages concluding that XML is the most appropriate candidate for policy specification language.

The PERMIS PMI architecture comprises a privilege allocation subsystem and a privilege verification subsystem. The privilege verification subsystem is responsible for allocating privileges to the users. The privilege verification subsystem is responsible for authenticating and authorizing the users. PERMIS PMI uses X.509 attribute certificates (ACs) to store the users' roles. All access control decisions are driven by an authorization policy, which is itself stored in an X.509 AC guaranteeing its integrity. All the ACs can be stored in one or more LDAP directories, thus making them widely available. Authorization policies are written in XML according to a DTD, that has been publishing at XML.org. Authentication is performed in an application-specific manner, but authorization is performed in an application-independent manner according to the PERMIS RBAC authorization policy. In this way one policy can control access to all resources in a domain. For more information about PERMIS one can refer to http://www.permis.org/index.html.

3.3. The European HARP Project

The HArmonisation for the secuRity of web technologies and aPplications (HARP) Project is funded by European Commission and driven by six countries (GR, UK, NL, DE, N, IL) [6]. The overall objective of HARP is to develop and harmonize Trusted Third Parity Services (TTPs), technologies and tools for the integration of applied and emerging Web oriented security systems for the telemedicine sector. The main idea of HARP is to let doctors visit patients documents in Europ.

Within the European HARP project, the HARP Cross Security Platform (HCSP) has been specified to design and to implement trustworthy distributed applications for health over the open Internet enabling both, communication and application security services. Certified servlets, composed and attributed according to the user's authorization create certified and signed XML messages. From those messages, user-role-related applets are generated. The needed Privilege Management Infrastructure has been established by an Attribute Authority and a policy server. The HCSP components are distributed installed over all countries involved. The role-based authorization has been defined according to the policy deploying the user's attribute certificates. The HARP solution has been practically implemented for a Clinical Study demonstrator at Magdeburg University Hospital, Germany.

The basic components of the HARP Cross Security Platform as depicted in Fig. 3 are:

- *A client environment*. This is fully under server control and accessible only to players holding the appropriate smartcard.
- *A web server*. It is the entrance-point for the user. It is also the Application server, user task are delegated to servlets; therefore an application server must host a web server.
- *Policy server*. Policies and policy related functions are provided.
- *Attribute Authority*. The Attribute Authority provides and manages (i.e. issuance, revocation, etc.) attribute certificates.
- *Database server*. All medical data is stored here. Control of access to data is policy-regulated.
- *Archive server*. All messages communicated are stored for accountability reasons.

For more information about HARP we refer to

http://telecom.ntua.gr/~HARP/HARP.htm.

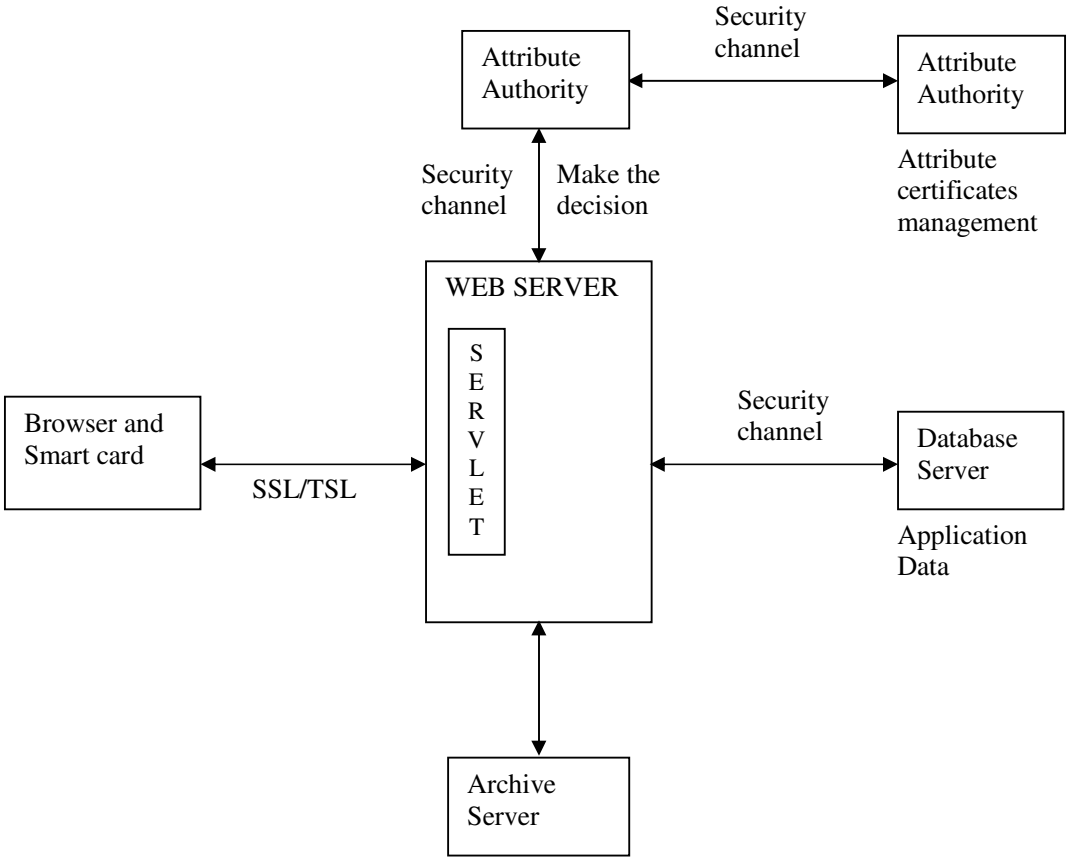

Figure 3. HARP Cross Security Platform Architecture

After assessing these important works and some other researches in the area of access control using attribute certificates, we decided to adopt the PERMIS authorization model to start our work, because it uses standard based open technologies. We have constructed a Role Based Access Control Infrastructure that uses PKI and PMI. An access control engine has been developed and used for constructing a demo system. We have also implemented a delegation mechanism using PKCs and ACs.

4. Implementation

4.1. System Overview

Our access control system has been developed to support role based access control using X.509 public key certificates and X.509 attribute certificates. The authentication function is implemented by PKI, and the authorization function is implemented by

PMI. All the access control decisions are made based on some policies. Our access control system has been designed to support multi-policies, so it is easy to add new control policy without influence to the old one. We have also developed a delegation system using X.509 public key certificates and X.509 attribute certificates and integrated it into the access control system. A picture of the access control system is depicted in Fig. 4.

These are the components:

- Administration tool - creating key pairs and their public key certificates, managing roles that will be used in a RBAC system, inserting a XML format access control policy into a policy attribute certificate and binds it to SOA's public key certificate. Creating a role attribute certificate and assigning it to a user through, binding it to a user's public key certificate, then the user gets some permission. For managing users' public key certificates and other information a user entity in a LDAP server is created. All the public key certificates and attribute certificates will be stored into one or several LDAP servers.

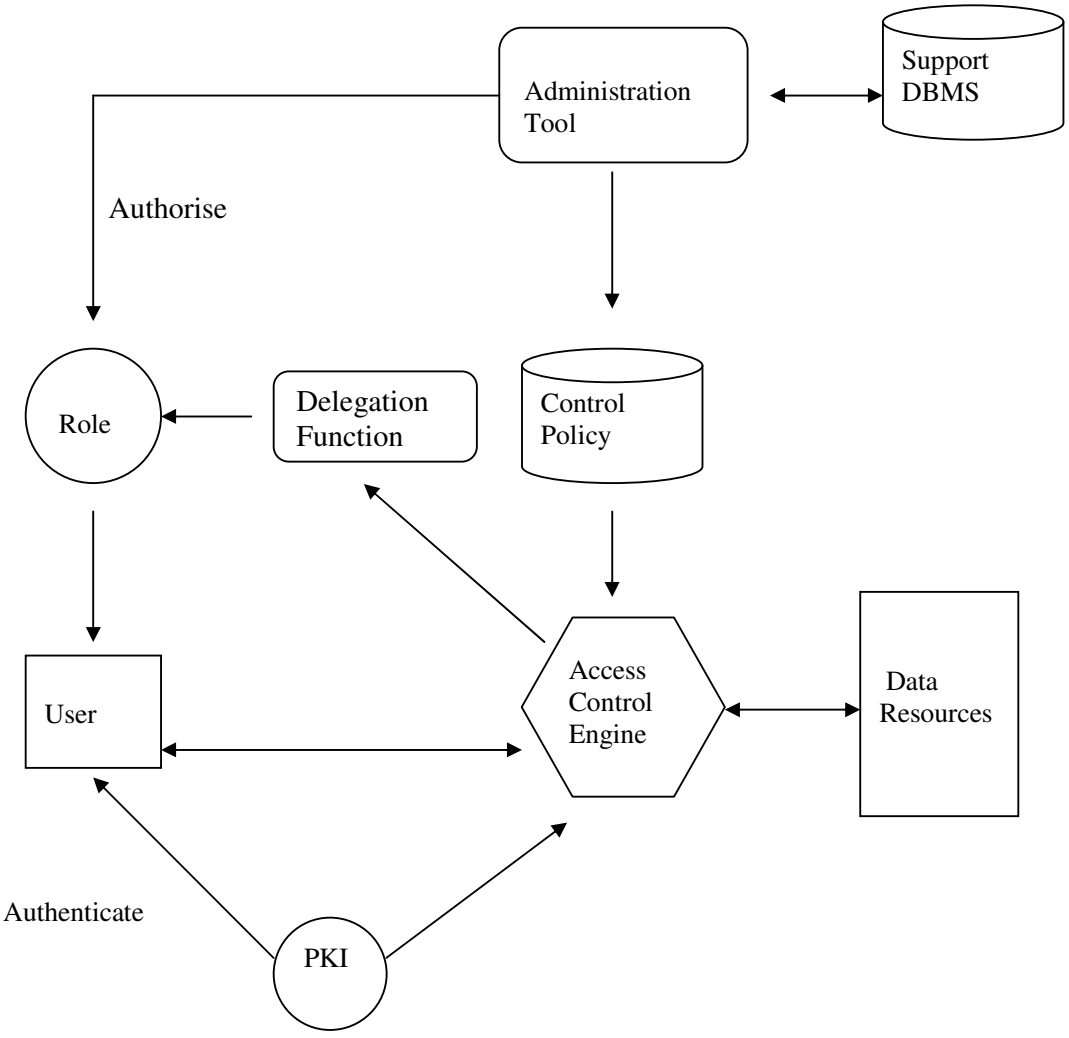

Figure 4. Overview of the access control system

- Support DBMS - storing the various information that are used by the administration tool, for example, the key stores' information, users' information, public key certificates and attribute certificates creating information and so on.
- Control policy - at the moment there are two kinds of polices, the RBAC policy and the delegation policy. The RBAC policy decides which role has which right to visit which data resources. The delegation policy decides which role can delegate which right to which role. Both are written in XML language. Both polices are stored in policy attribute certificates which are signed by the SOA guaranteeing their integrity.
- User side - the user connects to a dedicated web server via his browser, certainly the communication channel should be secure. Otherwise a secure protocol should be used, such as HTTPS. First the user uploads his signature file to server side for authentication. After passing the authentication process, he can visit some data resources, but what he can do there depends on which roles have been assigned to him.
- Delegation Function - it can automatically create role attribute certificates according to users' requests. But before doing it, the users' requests must be strictly checked by the access control engine according to the delegation policy.
- Access Control Engine - during runtime the Access Control Engine is the center of all processes, first it authenticates the user with the user's PKC, second it get the user's ACs, then gets the user's roles from his ACs. It will make the decision if the role has the right to visit the targets according to a policy and in case of permission, it will access the target and return the result to the user.
- Data Resources – on a Web server, data server, file system or some other format data resources.

4.2. Access Control Engine

The access control engine is responsible for authentication and authorization. Its structure is depicted in Fig.5. We adopt ISO 10181-3 Access Control Framework that is defined by the Open Group [11]. This framework separates authentication from authorization, and defines four roles for components participating in an access request: Initiator, Target, Access Control Enforcement Function (AEF) and Access Control Decision Function (ADF). Initiator submits access request that specifies an operation to be performed on a target. AEF mediates access requests, it submits decision request to ADF through the authorization API (aznAPI) [18], a decision request asks whether a particular access request should be granted or denied. AEF uses the aznAPI to presents Access Control Information (ACI) to ADF. ADF decides whether access requests should be granted or denied, it makes access control decisions based on Access Control Decision Information (ADI) that describes security-relevant properties of the initiator, the target, the access request, and the system and its environment. The aznAPI is responsible for deriving ADI from the ACI and presenting the ADI to the

ADF. At last AEF enforces access control decisions made by ADF. Our access control framework conforms to the basic principle of ISO 10181-3 Access Control Framework.

In our model, a user accesses resources via an access control engine. First a user uploads his signature file with browser. The AEF authenticates the user through verifying his signature with his PKC. If he is a valid user, then a session will be established for him, an LDAP DN will also be yielded and saved for him, otherwise stops this request. Secondly, after passing the authentication, the user can submit access request in order to visit some information. The AEF uses aznAPI to present ACI to ADF, the ACI holds the user's information (e.g. user's LDAP DN), action information and target information. The ADF uses User's LDAP DN to retrieve all the roles ACs that belongs to the user. These ACs are checked whether they are issued by a trusted AAs and still valid, they are also checked against the policy. The invalid ACs are discarded. From the valid ACs the user's roles will be extracted. Then these roles are checked whether they are allowed to perform the action to the target according to the access control policy. If the action is allowed, "permit" is returned, otherwise "no permit" is returned. In case of permission, the AEF will access the target on behalf of the user and return the result to user. If no permission user's request is refused. Our access control engine supports multi-policies, so each access request is checked against the specified policy.

At the moment our aznAPI has three methods: initializePolicy, getUserRoles and makeDecision:

- initializePolicy(policyURI, policyOID) - is called only once by AEF when AEF starts up. The AEF passes policy URI and policy Object Identifier (OID) to the ADF. From them the ADF retrieves the policy AC and checks if it is issued by the SOA and still within the valid time. Afterwards the policy, that is described in XML format, will be extracted from the AC. Then the policy is parsed and read into a series of arrays in order to fasten the decision process in runtime.
- getUserRoles(userDN) - AEF passes user's LDAP DN to ADF. ADF retrieves all valid ACs of the user using the LDAP DN, and extracts the roles from these ÁCs. Then it returns a role list to the caller. This function is useful in some scenarios where the AEF needs to know which roles the user owns.
- makeDecision(userDN, userAction, targetDN) - after a user has successfully passed authentication, he will attempt to perform certain actions on the target. At each attempt, the AEF passes the user's DN, Action and Target DN to ADF. ADF retrieves all valid ACs of the user and gets roles from these ACs. Then it checks if these roles have the right to perform the Action to the Target. If this wished action is allowed, then it returns "permit", otherwise "no permit". From the result of makeDecision, the AEF will perform the action to the target on behalf of the user or it refuses the user's request.

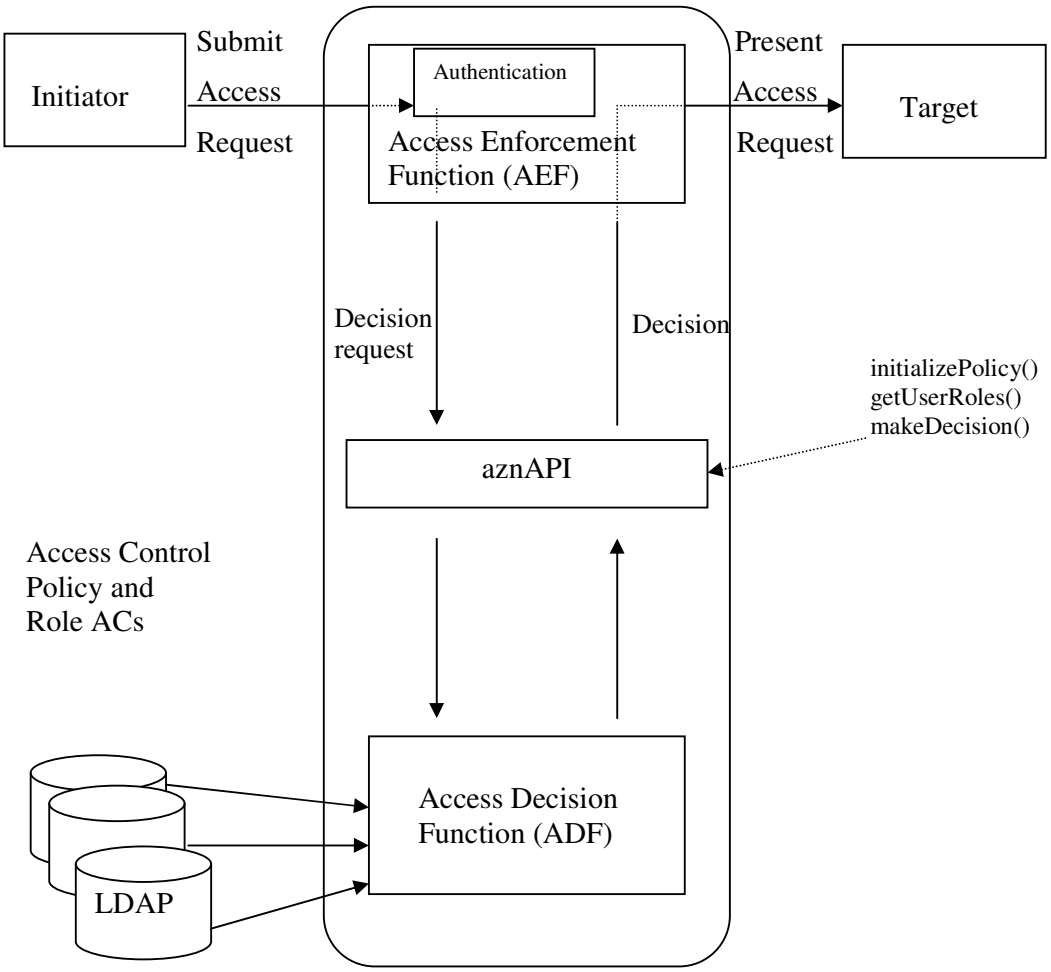

Fig. 5. Privilege verification subsystem

4.3. Access Control Policy

A role-based authorisation policy should specify which permissions are assigned to which roles, which roles are assigned to which users, and on which conditions the decision is taken. The access control engine reads the policy in, parses it, and then uses it to control access to targets within the policy domain. Each access request is checked against the policy: A permitted access request will be executed and the result will be returned to the user, a none permitted access request will be refused. Our access control engine has been designed to support multi-policies. There are two kinds of policies in our prototype, the RBAC policy and the delegation policy.

In our prototype we directly use the PERMIS X.500 PMI RBAC policy [4, 5, 12] as our RBAC policy, which is written in XML. Its Data Type Definition (DTD) has been published at http://www.xml.org, and is also available from the web site at http://sec.sis.salford.ac.uk/permis/Policy.dtd. We choice it as our RBAC policy,

because it was specifically designed for use in a X.509 attribute certificate based PMI, and totally suits to our first stage of research, e.g. constructing a role based access control system using attribute certificates. PERMIS X.500 PMI RBAC policy is composed of a set of sub-policies. These are:

- SubjectPolicy - this specifies the subject domains, i.e. only users from a subject domain may be authorised to access resources covered by the policy.
- RoleHierarchyPolicy - this specifies the different roles and their hierarchical relationships to each other.
- SOAPolicy - this specifies which SOAs are trusted to allocate roles. By including more than one SOA in this policy, the local SOA is effectively a cross-certifying remote authorisation domains.
- RoleAssignmentPolicy - this specifies which roles may be allocated to which subjects by which SOAs, whether delegation of roles may take place or not, and how long the roles may be assigned for.
- TargetPolicy - this specifies the target domains covered by this policy.
- ActionPolicy - this specifies the actions (or methods) supported by the targets, along with the parameters that should be passed along with each action, e.g. action Open with parameter Filename.
- TargetAccessPolicy - this specifies which roles have permission to perform which actions on which targets, and under which conditions.

A full description of the PERMIS X.500 PMI RBAC policy can be found in [4].

4.4. Delegation Mechanism

The basic idea behind delegation is that some active entity in a system delegates authority to another active entity in order to carry out some functions on behalf of the former. Delegation can be permanent or temporary, total or partial, monotonic or non-monotonic [13, 27]. In this part we concentrate on implementing a role based delegation mechanism using public key certificates and attribute certificates, that does not need administrator's interventions. A delegation control policy is also developed. It is responsible for checking if a delegation is conform to various constraints. It collaborates with RBAC policy to complete delegation control.

When a user wants to delegate some right to another user, first he uploads his signature to the access control engine, the AEF authenticates him, and a session will be set for him if he is a valid user. This authentication process is the same as it in normal access control. Secondly the user provides a delegation request to the AEF. The AEF checks if the delegated user is a valid user, for example, if he has a valid PKC. Then AEF passes delegating user's LDAP DN, action and delegated user's LDAP DN to the ADF. The ADF retrieves delegating user's ACs and delegated user's ACs, and checks if this delegation is permitted against the RBAC policy. Then ADF returns the delegating role and delegated role to the AEF. If the request has passed this audit, the AEF will pass delegating role, delegation content and delegated role to ADF again. But this time, this decision request is checked against the delegation control policy, and a permitted or none permitted result will be returned to AEF. If the second check has passed, the AEF will send user's request to a delegation server (e.g.

the Attribute Authority). The delegation server will create an attribute certificate for the delegated user according to the delegation content, and bind it to his public key certificate. Then it publishes it to the user's LDAP entity (in ACs "pull" model). After that the delegated user has the new privilege immediately.

4.5. Delegation Policy

Delegation policy is used by delegation authorisation. It specifies which roles can delegate which privileges to which roles, and what are the constraints. This policy is composed of some sub-policies. These are:

- DelegationPolicy - this specifies which types of delegation can be used for delegating.
- RolePolicy - this specifies which roles can delegate, and which roles can be delegated.
- RoleDelegationPolicy - this specifies which roles have permission to delegate which privileges to which roles, and under which constraints.

4.6. Administration Tool

The administration tool can complement three main functions: Create key pair and its X.509 PKC, manage users' PKCs, and create X.509 policy ACs and role ACs. The key pair and its PKC tool can create key pair and its self-signed PKC for CA or AA, or non self-signed PKC for normal users when the system does not need Trusted Third Party services (TTPs). When TTPs are needed, users must provide their PKCs to the SOA creating role ACs for them. The PKCs management tool can be used for managing users' PKCs, for example, imports or exports users' PKCs from a LDAP server or erases them. The AC tool can create two kinds of ACs, one is policy AC and the other is role AC. The policy AC's attribute value is in printable XML format that is read from a XML file. This kind of AC is bounded to a SOA's PKC and signed by the SOA, so the issuer and holder names are the same in a policy AC, it is similar the self-signed PKC of root CA in a PKI. The role AC's attribute value consists of one or several role names in printable string format, for example, Manager, Clerk, and so on. This kind of AC is bound to a user's PKC and signed by the SOA. In our prototype we adopt AC "pull" model, so the role ACs are not given to users. The policy ACs and role ACs are all stored in a LDAP server. Since they have been signed by the AA, they are tamper-resistant, and therefore there is no modification risk from allowing them to be stored in a publicly accessible repository.

The administration tool also has some other useful functions, for example, manage key store, manage users' information, verify PKCs and ACs, display ACs, generate and verify the digital signature, and so on.

5. Demo System

5.1. Overview of the Demo System

We have developed a demo system in order to illustrate how our approach can meet the requests of real world. We use the context of an e-shopping scenario. On the client side, the customers can view the product list of a company via a browser and order some products. On the server side, the following roles have been defined for the company: Administrator, Manager, and Clerk. An overview of the demo system is depicted in Fig.6. It is comprised with User Browser, Web Server, Support Database Server, LDAP Server, Administration tool, delegation server and Application Database Servers.

The access control engine is implemented by a Java servlet (Tomcat servlet container) [14]. The servlet implements a service that mediates the data between the users and the database or other web servers, which also executes the functions of authentication and authorisation. Application data are stored in an application database server (mySQL) [16]. Support database contains the information that is used by the administration tool, for example the users' information that is used for creating users' LDAP entity. Policy ACs, users' role ACs, and users' PKCs are all stored in a LDAP server (openLDAP) [15]. The user uses a browser to upload his or her signature file to server side for authentication. After that he can access the target resources due to some constraints depending on which roles he or she has. Delegation server can create role ACs for other users according to the delegating user's commands. But before doing it, delegation server must consult RBAC policy and delegation policy to confirm that the delegating user has the right to delegate, and that the delegated user has the right to be delegated. This process does not need administrator's intervention.

In our prototype we defined two types of Attribute Certificates: policy AC and role AC. Policy ACs hold access control decision information. They are read in by the access control engine in the initialize phase. There are two policy ACs in the demo system, one is the RBAC policy, the other is the delegation policy. The SOA creates the authorisation policies for the domain using his favourite XML editing tool, and stores them in local files. Then it uses the administration tool to create the policy ACs. Each policy is given an OID that identifies it globally unambiguously. This OID is passed to the ADF by the AEF in the access control engine, in order to guarantee that the correct policy will be used in the subsequent access control decisions.

The role ACs holds the role information, in this scenario the role names. Every user has one PKC and one or several role ACs. The user can obtain various privileges from owning different roles. These roles can be put in one AC or in several ACs. If these roles are stored in several role ACs, assigning some rights to him or revoking some rights from him only needs create new role ACs that holds new role name to him or simply delete them (in ACs "pull" model) or puts them into attribute certificate revocation list (ACRL). In the runtime, when a user has passed the authentication process that finishes in the AEF, the AEF pass his request to the ADF, the ADF retrieves all the user's valid ACs and extracts roles from them, then a decision will be made based on these roles and privilege polices. The ADF returns either "granted" or "denied" to the AEF. The AEF continues to access the target on behalf of the user and returns the result if the decision is granted, otherwise refuses the user's request.

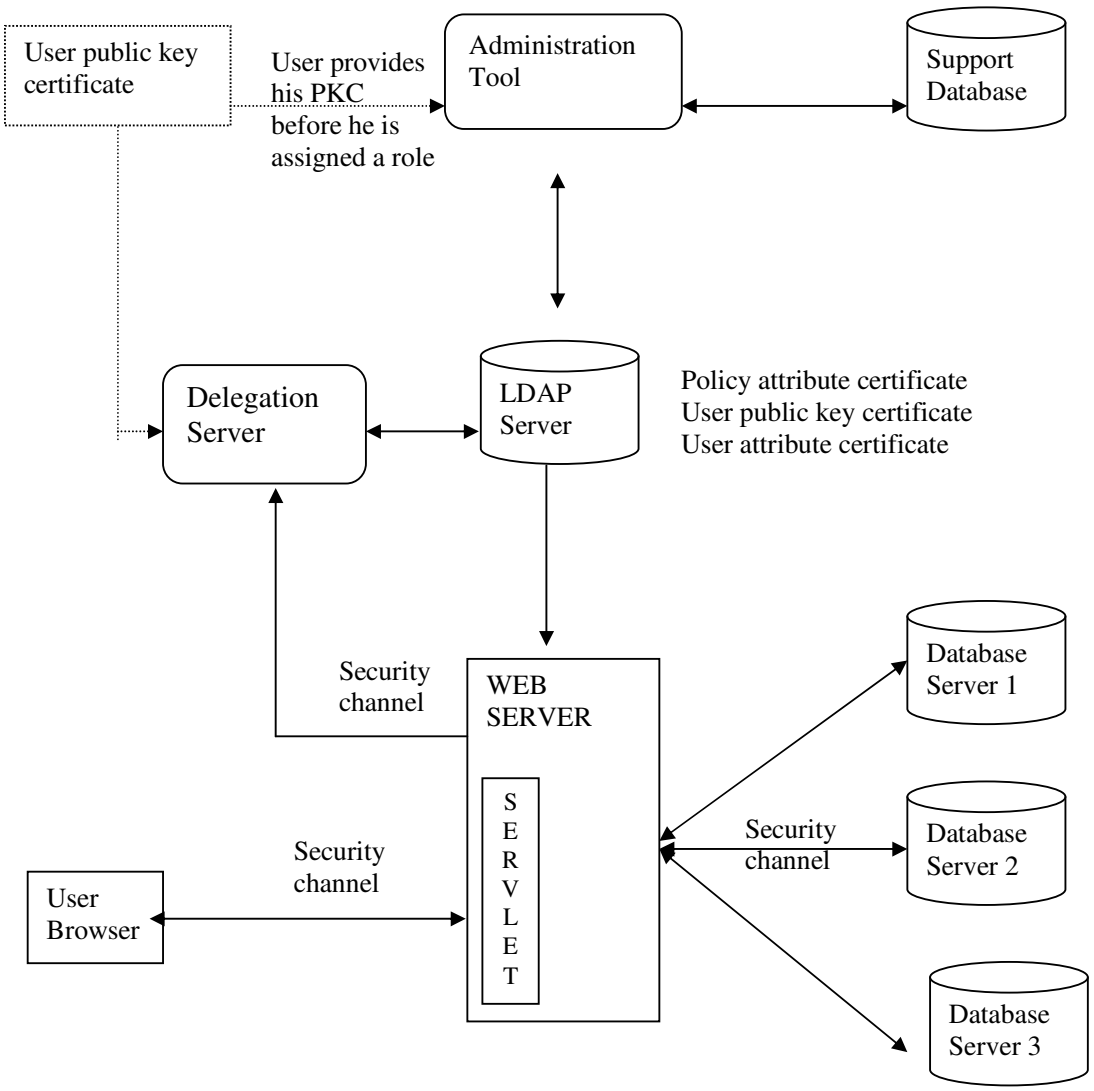

Fig.6. The access control demo system

5.2. Definition of Roles, Actions and Targets in demo system

In our demo system roles, actions and targets are defined as follows:

- Roles: "Administrator", "Manager", "Clerk", and "Customer".
- Actions: "Search", "Modify", "Delete", "Append", "Initialize", and "Display" recorded in some tables.
- Targets: "Product Table" and "Shopping Table" in a database.

The administrator role can initialize the product table and the shopping table. The Clerk role can append new recorders in product table. The Manager role can modify and delete records in the product table, it also has the right to append records through inheriting it from the Clerk role. The Customer role can search, modify, delete, append and display records that belong to him in the shopping table. Any valid role can search and display the records in the product table and shopping table. These assignments and constraints have been described in the PERMIS X.500 PMI RBAC Policy.

5.3. Authentication, Authorization and Service Sequence

The runtime oft authentication, authorisation, and the service sequence is depicted in Fig.7.

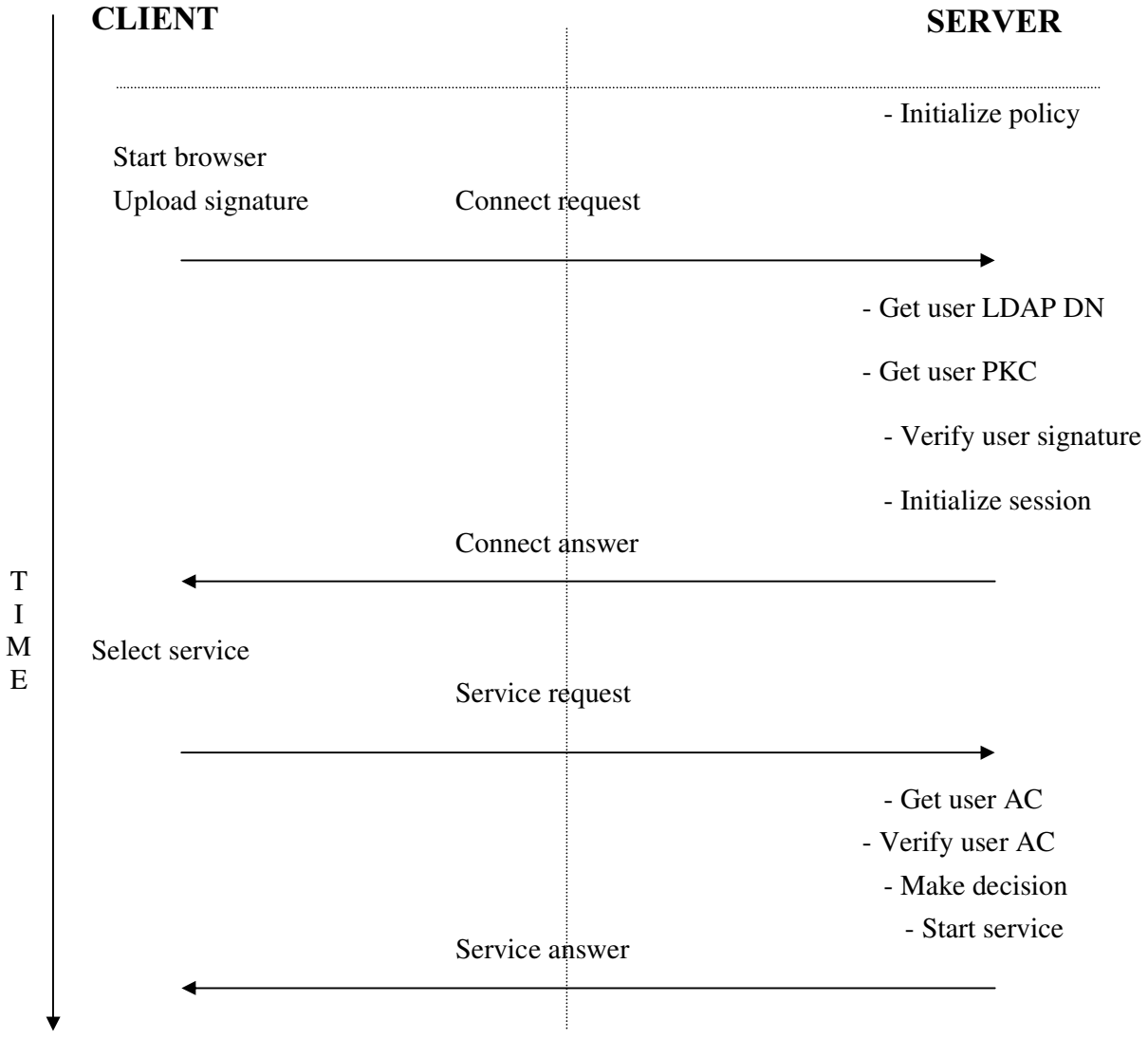

Fig.7. Authentication, authorization and service sequence

First, server reads in the policy ACs and prepares the running environment. Then the client uploads his signature, it begins the connect process. The server gets user's LDAP DN. Using the LDAP DN the server gets user's PKC from a LDAP server. Then it verifies the user's signature.If the verification is successful, a session will be set for the user. Then connect process, e.g. for authentication process is finished.

After passing the authentication, the client can select a service and send his request to the server. The service process starts. The server gets user's valid ACs and extracts the roles from them. Then a decision is made by checking the roles against some policies. If the decision is granted, the server will start the service and return the result to the client. Then the service process, e.g. for the authorization process is finished.

5.4. Signature File Structure

In order to establish a connection to a server, the client needs to upload his digital signature to the server. The signature file structure which is used in our demo system is depicted in Fig.8. Signature describing information includes signature type, algorithm length, provider length, signature length, algorithm, provider, signature and original data. There are two types of signatures in the demo system. One is for "normal" access control. Here the original data is user's LDAP DN. The other is for delegation. Here the original data is delegating user's LDAP DN, delegation information and delegated user's LDAP DN.

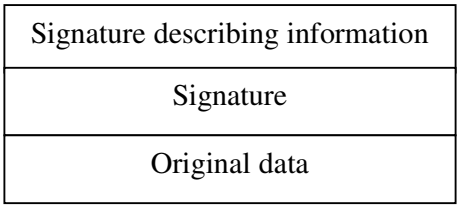

Fig.8. Signature file structure

6. Conclusions and Future Work

We established a framework for RBAC and delegation using PKCs and ACs, where all access decisions are made against some authorisation policies. These policies are written in XML, and stored in policy ACs. Our access control engine supports multi-policies, this is one difference to the PERMIS model that controls one domain by one policy. We think that supporting multi-policies is very important for constructing a flexible application system. We have developed an access control engine that runs in Java servlet. We also developed a demo system that uses this access control engine, and has proved that using PMI is a good solution for access control in a Web-environment.

Role based access control usually provides a sufficient way to establish access control in most information systems. However passive security permission assignment can not support efficiently the dynamic aspects of many modern information systems like the workflow that in an enterprise environment. Although there was done some work in this area [22,23,24,], there is still no perfect solution. So, our future work will concentrate on finding new access control mechanism that can overcome some weaknesses of RBAC. The main idea is to extent RBAC through adding a new form of access control layer on RBAC and integrate PKI and PMI technologies in it. A new access control policy should also be developed.

References

[1] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, Role based access control models, IEEE Comput. 29 (2) (1996).

[2] David F. Ferraiolo, R.S. Sandhu, Serban Gavrila, D. Richard Kuhn and Ramaswamy Chandramouli, Proposed NIST Standard for Role-Based Access Control, ACM Transactions on Information and Systems Security (TISSEC), Volume 4, Number 3, August 2001.

[3] ITU-T Rec. X.509 ISO/IEC 9595-8, The Directory: Public-key and Attribute Certificate Frameworks, May 3, 2001.

[4] D.W. Chadwick, A. Otenko, RBAC Policies in XML for X.509 Based Privilege Management, to be presented at SEC 2002, Egypt, May 2002.

[5] D.W. Chadwick, A. Otenko, The PERMIS X.509 Role Based Privilege Management Infrastructure, SACMAT'2002, Monterey, CA, June 2002, pp. 135-140.

[6] B. Blobel, P. Hoepner, R. Joop, S. Karnouskos, G. Kleinhuis and G. Stassinopoulos, Using a privilege management infrastructure for secure web-based e-health applications, Computer Communications, Volume 26, Issue 16, 15 October 2003, Pages 1863-1872.

[7] Javier Lopez, Antonio Mana, Juan J. Ortega, Jose M. Troya and Manemma I. Yague, Integrating PMI services in CORBA applications, Computer Standards & Interfaces, Volume 25, Issue 4, August 2003, Pages 391-409.

[8] W. Johnston, S. Mudumbai, M. Thompson, Authorization and attribute certificates for widely distributed access control, in: Proceedings of the Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), Stanford, CA, June 1998, pp. 340–345. http://dsd.lbl.gov/security/Akenti/.

[9] An Internet Attribute Certificate Profile for Authorization, http://www.ietf.org/rfc/rfc3281.txt.

[10] Internet X.509 Public key Infrastructure Certificate and CRL Profile, http://www.ietf.org/rfc/rfc2459.txt.

[11] ITU-T Rec X.812 (1995) ISO/IEC 10181-3:1996 "Security Frameworks for open systems: Access control framework, 1995.

[12] D.W. Chadwick, A. Otenko, The PERMIS X.509 role based privilege management infrastructure, Future Generation Computer Systems, Volume 19, Issue 2, February 2003, Pages 277-289.

[13] Ezedin S. Barka, Framework for Role-Based Delegation Models, A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

[14] Jakarta Tomcat servlet container, http://jakarta.apache.org/tomcat/.

[15] OpenLDAP, the Open Source Lightweight Directory Access Protocol (LDAP), http://www.openldap.org/.
[16] MySQL, the Open Source SQL database, http://www.mysql.com/.
[17] J.S. Park, R. Sandhu, G. Ahn, Role-based access control on the web, ACM Transactions on Information and System Security, 4 (1) (2001) 37-71.
[18] The Open Group, Authorization (AZN) API, January 200. ISBN 1-85912-266-3.
[19] Yi-Shiung Yeh, Wei-Shen Lai, Chung-Jaye Cheng, Applying lightweight directory access protocol service on session certification authority, Computer Networks 38 (2002) 675-692.
[20] Jing-Jang Hwang, Kou-Chen Wu, Duen-Ren Liu, Access control with role attribute certificates, Computer Standards & Interfaces 22 (2000) 43-53.
[21] Andrew D. Fernandes, Risking "trust" in a public key infrastructure: old techniques of managing risk applied to new technology, Decision Support Systems 31 (2001) 303-322.
[22] Edward C, Cheng, An object-oriented organizational model to support dynamic role-based access control in electronic commerce, Decision Support Systems 29 (2000) 357-369.
[23] Sejong Oh, Seog Park, Task-role-based access control model, Information Systems 27 (2002) 533-562.
[24] Won Bo Shim, Seog Park, The Work Concept RBAC Model for the Access Control of the Distributed Web Server Environment, Web Intelligence, First Asia-Pacific Conference, Japan, October 2001 262-266.
[25] Gail-Joon Ahn, Role-based access control in DCOM, Journal of Systems Architecture 46 (2000) 1175-1184.
[26] A. Lin, R. Brown, The application of security policy to role-based access control and the common data security architecture, Computer Communications 23 (2000) 1584-1593.
[27] E. Barka, R. Sandhu, Framework for role-based delegation models, Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC/2000), New Orleans, LA, 2000.
[28] T. Becker, T. Engel, Ch. Meinel, IT-Sicherheitszerifikate, Technischer Bericht 01-05 des Instituts für Telematik, Trier, Germany, 2001.