2004

Enhanced security architecture for support of credential repository in grid computing.

Hao Chen
University of Windsor

Enhanced Security Architecture for Support of Credential Repository in Grid Computing

By

Hao Chen

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirement for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2004

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisisitons et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-92491-2
Our file Notre référence
ISBN: 0-612-92491-2

ABSTRACT

Grid Computing involves heterogeneous computers and resources, multiple administrative domains and the mechanisms and techniques for establishing and maintaining effective and secure communications between devices and systems. Both authentication and authorization are required. Current authorization models in each domain vary from one system to another, which makes it difficult for users to obtain authorization across multiple domains at one time.

We propose an enhanced security architecture to provide support for decentralized authorization based on attribute certificates which may be accessed via the Internet. This allows the administration of privileges to be widely distributed over the Internet in support of autonomy for resource owners and providers. In addition, it provides a uniform approach for authorization which may be used by resource providers from various domains. We combine authentication with the authorization mechanism by using both MyProxy online credential repository and LDAP directory server.

In our architecture, we use MyProxy server to store identity certificates for authentication, and utilize an LDAP server-based architecture to store attribute certificates for authorization. Using a standard web browser, a user may connect to a grid portal and allow the portal to retrieve those certificates in order to access grid resources on behalf of the user. Thus, our approach can make use of the online credential repository to integrate authentication, delegation and attribute based access control together to provide enhanced, flexible security for grid system.

Keywords: Grid computing, security, authentication, authorization, PKC, SSL, LDAP, GSI, MyProxy, proxy credential, attribute certificate, RBAC, grid portal

DEDICATION

To my husband, WenBo

To my parents

Who inspire me and love me all the time

ACKNOWLEDGEMENTS

My first and foremost thanks go to my supervisor, Dr. Robert D. Kent, for his unwavering encouragement, experienced guidance, and invaluable discussions during the research. Without his assistance and inspiration, this thesis work would not be accomplished at all.

I would also like to thank my committee members, Dr. Xiao Jun Chen and Dr. Kemal E. Tepe for their interest, critical analysis and valuable suggestions on the entire thesis. Thanks to Dr. Akshai Aggarwal for chairing my thesis committee.

Finally, I would like to thank my friends and colleagues in our Computational Grids lab for many pleasant discussions during my thesis research. A special acknowledgement must be made to SHARCnet for research funding received through the Research Fellowships Program during 2003 to 2004.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AA	Attribute Authority
AC	Attribute Certificate
AS	Authentication Server
BER	Basic Encoding Rules
CA	Certificate Authority
CN	Common Name
DAP	Directory Access Protocol
DIT	Directory Information Tree
DN	Distinguished Name
GSI	Grid Security Infrastructure
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
OID	Object Identifier
OSI	Open Systems Interconnection
OU	Organizational Unit
PKI	Public Key Infrastructure
PKC	Public Key Certificate
RBAC	Role Based Access Control
RSA	Rivest Shamir Adelman
SAML	Security Assertion Markup Language
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TGS	Ticket-granting Server
URI	Uniform Resource Identifier

1. INTRODUCTION

Grid Computing involves heterogeneous computers and resources, multiple administrative domains and the mechanisms and techniques for establishing and maintaining effective and secure communications between devices and systems. In grid environments, the requirements for security tend to increase rapidly, due to the complexity of the systems and interconnection models used. Because of the heterogeneity of security mechanisms, grid security requirements may vary from one system to another, but at least they include authentication, authorization, single sign on and delegation. Within security requirements, delegation and distributed authorization are among the most challenging issues in grids.

In this thesis, we address the problems of current credential repositories for grid computing. We propose a new approach to develop an enhanced and flexible security architecture for support of MyProxy using attribute certificates and LDAP server, and show how this architecture can provide both authentication and authorization. In addition, this architecture can also integrate with current existing technology for Role Based Access Control (RBAC) and support certificate delegation. Security in large distributed systems requires access control based on roles rather than individual identities. Role-based management has proved to be a very important technique in reducing administrative costs. Finally, we provide support for localized resource ownership-based autonomy in setting authorization policy within the context of resource sharing.

1.1 An Overview of Grid Computing

"The Grid is a next-generation Internet" [Foster01]. It provides secure and high-performance mechanisms for remote access and resource sharing. It can also help geographically distributed groups work together. The goal of grid computing is to make easy, fast and inexpensive access to computing resources across the world.

[Foster01] defines grid computing to be different from conventional distributed computing because it includes large-scale resource sharing and high performance applications. And Grids are used in dynamic virtual organizations which are a group of people or institutions sharing computer resources to further their objectives. [Foster01] gives a detailed introduction to grid concepts and technologies. It presents requirements and a framework for grid architectures, and protocols and services used among different grid systems. Grid computing offers more powerful applications including using computer instruments, data-intensive computing, multi-supercomputer simulation, collaborative work, and so on. For example, scientists often want to discuss with other people in real time after they have performed experiments, therefore, collaborative work is required. In addition, they also require the real-time analysis of large amounts of data using computer-enhanced instruments. All of these are important applications in grid computing.

There are three characteristics for computational grids [Foster98b]:

(1) **Heterogeneous**. A grid involves many administrative domains and different resources that may exist in any place of the world. The computing resources may be different. The difference may be in physical devices, system software, and various policies.

(2) **Scalable**. The grid size may increase when adding resource. The scalability will make it not affect the performance of whole system.

(3) **Dynamic and unpredictable**. Traditional networking systems are predictable. But in grid-computing environments, there are many resources for sharing, and it is impossible to guarantee each resource will not fail sometimes. So some measures will be taken dynamically in order to provide the best performance.

1.2 Contribution

Although there is a substantial body of published research and development on authentication in Grids, it has not resulted in sufficient standardized mechanisms to control user access to grid resources scattered across large-scale networks. Authorization mechanisms should combine with authentication mechanisms in order to provide strong security service.

Most Grids use Public Key Infrastructure (PKI) for authentication, but users are required to protect their keys by themselves and have to copy certificates to their own systems. It is inconvenient and insecure to access the grid credentials. MyProxy is an online credential repository system that can solve this kind of problems.

There are two problems in current security architectures. First problem is about grid portal. Grid portal provides user interfaces for the grids and allows users to access grid resources using standard web browsers. It is based on standard web server and some grid enabled software. But grid portals cannot integrate with Grid Security Infrastructure (GSI) [Foster98b] because standard web browsers do not support credential delegation while GSI support certificate delegation. Second problem is about MyProxy repository. MyProxy is used for authentication using identity certificates, but there is little work done for authorization. And although it implements delegation, there is no more restriction on the delegated credentials except the lifetime.

We propose to use MyProxy to solve the first problem about grid portal. Using MyProxy, the user can connect to a Grid portal through a web browser and allows the portal to retrieve a proxy credential to access Grid resources on the user's behalf. In this manner, MyProxy can solve the delegation problem and enable grid portals to access GSI protected resources.

We provide a solution for the second problem about MyProxy by using attribute certificate technology. Attribute certificates can extend identity-based public key certificate (PKC) infrastructures towards support for role-based authorization policies [Linn99]. First, it allows decentralized authorities to manage identities, roles and permissions. Second, it can provide explicit representations of the roles. Third, attributes represented in attribute certificates can be separated from identity certificates, which allow authorization attributes to be updated more frequently than the identity certificates. In addition, the large and constantly changing population of entities requires the access policy on roles or groups rather than individual ones. So role-based controls can be accomplished more flexibly when using attribute certificates.

There is another problem with MyProxy. Its protocol does not support uploading of attribute certificates. So we propose to use Lightweight Directory Access Protocol (LDAP) which is a platform-independent network protocol standard. It can be used across a highly heterogeneous network. Moreover, it can also be a gateway to other protocols such as X.500 and HTTP. LDAP supports any type of data and various security technologies. Therefore we propose to use LDAP and build our LDAP server that can support uploading and storing attribute certificates.

The design of the enhanced security architecture for support of online credential repository and the deployment of LDAP server for ACs are both innovative and constitute the main contributions of this thesis.

We design a new and innovative architecture through combining MyProxy and LDAP Server to provide both authentication and authorization service for Grids. In this thesis, we use attribute certificates to convey privileges, which are separated from identity credentials used for authentication. The separation of authentication from authorization allows for flexible delegation. Our novel enhanced security architecture enables fewer overheads than using general access control lists to make access decisions. We set up LDAP directory server and deploy environment for adding attribute certificates onto LDAP server, and also provide a role design for authorization by using attribute certificates. Our enhanced security architecture can support Role Based Access Control (RBAC) model based on the roles in the attribute certificates. In addition, grid users are typically widely dispersed, so our architecture supports distributed authorization.

1.3 Thesis Statement

1.3.1 Motivation – Problem

Grid Computing involves heterogeneous computers and resources, multiple administrative domains and the mechanisms and techniques for establishing and maintaining effective and secure communications between devices and systems. Both authentication and authorization are required. Most research in the security area focuses on authentication techniques, including digital signatures and public key certificates; but there is little work established for building a uniform authorization model for large-scale networks. Each local domain authorization model may be different from one system to another, which makes it difficult for users to obtain authorization across multiple domains at one time.

Some grid security architectures are limited by using access control lists. Having a user account on every resource a subject does not scale well to large grids. It not only limits scalability, but also hinders collaboration. In grids, resource providers and users are both geographically distributed. Without standardized and easily administered authorization mechanisms it is impossible to share distributed resources.

Some questions are related to the issue of authorization: how can resource providers to manage their resource across different domain flexibly? How can users access multiple resources and delegate their rights? How to build a standardized authorization model for grids? This thesis elaborates on solutions that address these questions.

1.3.2 Solution – Design Brief

We propose an enhanced and flexible security architecture to provide decentralized authorization, based on attribute certificates, which may be accessed via the Internet. We seek to provide an authorization mechanism for widely distributed resource providers to control their resources flexibly and for users to delegate their rights. Our security architecture enables users to share grid resources and support distributed collaboration.

Our solution is based on digital certificates that convey identity, attributes or authorization information. This allows the administration of privileges to be widely distributed over the Internet. In addition, it provides a unified approach for authorization which can be used by resource providers from various domains. It addresses the utilization of attribute certificates for support of role based access control and rights delegation.

1.3.3 Technology and Tools

We combine with authentication and authorization mechanism by using both MyProxy online credential repository and LDAP directory server. In addition, the thesis also elaborates on how to deploy LDAP server in order to support using attribute certificates.

MyProxy is a credential repository for the Grid. It allows users to retrieve a credential at any time or any place without needing to manage private keys and certificate files. However, it only enables users to delegate identity certificates to resources. The identity certificate supports only authentication, but not authorization. We propose an approach to develop an enhanced security architecture using attribute certificates. An AC can grant some attributes to its owner, which is well suited for controlling access to grid resources and for implementing role-based authorization.

In grid environment, it is necessary to have a flexible, scalable security infrastructure to store and manage user profiles for user authentication and authorization. LDAP directory server allows both users and resource providers to collaborate and share information. We use LDAP server in our architecture for managing users' ACs for large-scale applications. In our architecture, we use MyProxy server to store identity certificates for authentication, and build LDAP server to store attribute certificates for authorization. Then, using a standard web browser, a user can connect to a grid portal and allow the portal to retrieve those certificates in order to access grid resources on behalf of the user. Section 4.3 describes the three components in our proposed security architecture, and six basic steps for providing both authentication and authorization service. In this thesis, we address the second component of our security architecture, LDAP directory server, and provide details on how to design our schema for support of ACs and the deployment.

1.3.4 Impact

Our proposed architecture improves the security services of computational grids. It supports sharing authorization information of grid users. It also contributes to the scalability and delegation in grid environments. In addition, it can reduce administrative costs to resource providers and enable dynamic collaboration in grid computing.

Our approach significantly simplifies the management of access controls for large numbers of users because access decisions are based on a set of roles rather than a great number of individual users. In addition, role information is in form of digital certificate which can be distributed all over the world. It allows resource providers to share the authorization information of users and create collaborative environments, and may allow users to access multiple resources by using the same roles or different roles.

This architecture enable distributed resource providers to control their resources and specify access requirements flexibly. Resource providers can define their own access policy for various roles or attributes and can create or change any type of roles of a user. With this new approach and architecture, it is more secure and convenient for users to access and for resource providers to manage resources.

1.4 Thesis Structure

This thesis presents both background and solutions to current problems in grid security area. It is structured as follows. Chapter 2 provides a literature review about current grid security requirements, architectures and policies. Chapter 3 introduces the related security technologies including public key infrastructure, SSL, RBAC and their limitations and advantages are discussed. Chapter 4 describes in more detail our proposed enhanced security architecture and technology used including AC and LDAP. Chapter 5 concentrates on the design and deployment of LDAP server for our architecture. Chapter 6 describes testing and evaluation for the proposed mechanism, features and limitation are also discussed. Chapter 7 concludes and discusses research issues for future work.

2. GRID SECURITY

Computational grids demand advanced security mechanisms. This chapter covers the various aspects in the area of grid security. Section 2.1 describes grid security requirements including authentication, authorization, confidentiality, integrity, single sign-on and delegation. Section 2.2 provides an overview of existing grid security architectures that satisfy these requirements. Section 2.3 introduces current representation approaches to security policies on the resources.

2.1 Grid Security Requirements

Security in grid computing is complicated because there are many administrative domains. [Humphrey02] describes grid security requirements including authentication, authorization, integrity, and confidentiality. And there are some other important requirements such as single sign-on and certificate delegation.

2.1.1 Authentication

Authentication is the foundation of the security requirements in grids. It is used to guarantee that entities are who they claim to be. For each communication between two parties, both of them would like to prove the identity of each other and the origin of the information. There are many authentication technologies such as password-based authentication, public key certification, Kerberos, Secure Socket Library (SSL), Secure Shell (SSH) and biometric characteristics such as fingerprint, voice and retinal recognition. These approaches represent different strengths of authentication. People use them depending on the level of security required by their application systems.

User name and password is the traditional authentication method. Although it is very popular, there are some existing problems. Users frequently use simple passwords, so passwords can be easily stolen, sniffed or guessed by dictionary tools or brute-force attacks [Hayes00]. And some passwords never expire; some password files are stored in clear text; and although the files are encrypted, many passwords are transmitted in clear text over the network. An unencrypted password is easily broken. "One-time password scheme" [Hayes00] provides authentication for unsecured networks. Passwords can be kept on a central system and requested by users. Because these passwords are used only once, they are better than conventional passwords.

Kerberos is a network authentication protocol and standard for security. It has been widely used since the mid-1980s [Foster98b]. Most research on it relies on conventional cryptography and on-line Authentication Server (AS)/Ticket-granting Server (TGS) [Kohl93]. Kerberos is a published standard and a single-sign-on technology, which permits users to log on once and access all authorized resources without re-entering password. The most important part of Kerberos is the Key Distribution Centre (KDC) that issues tickets for the user. The client and server should register their keys in advance on the KDC respectively. It includes two servers to perform key distribution. One is AS; the other is TGS. AS is responsible for issuing tickets for TGS, and TGS issues tickets for various servers and generates the shared secret key between client and server. Therefore, before a client communicates with a resource server, it should contact AS first. In addition, Kerberos can also implement cross-realm authentication by designating trustworthy key servers in other organizations. But the Kerberos administrators have to set up agreements with another realm. This is often a lengthy and complex process [Thompson02a]. And it is infeasible for inter-site authentication because of equipment cost. Kerberos meets many security requirements for Grids, such as single sign-on and delegation [Kohl93]. However, Kerberos is more suitable for intra-site authentication because it always requires a centrally maintained key server for authentication [Butler99], while grids require inter-site authentication. Therefore, Kerberos is mostly used within a single administrative domain.

In grid computing, authentication technologies are for larger, dynamic, heterogeneous communities, and trust relationships will span multiple domains. Mutual authentication should be required to ensure that the sender and receiver are both authentic during the entire connection period. PKC and SSL are widely used and will be described in more detail in the following chapters.

2.1.2 Authorization

Authorization is based on authentication. It specifies what kind of operation is allowed for the user after the authentication. Authentication verifies identity, while authorization verifies privileges. The privileges for accessing grid resources may include reading or writing files, accessing database, running a program on a processing node, and so on. There are two authorization models: pull model and push model. Both models include the requestor, a request and a target resource, and they also provide an authorization decision.

(1) **Authorization Pull Model**

In the pull model (see Figure 2-1), a requestor sends a request to the resource, and then the resource sends a query to the authorization service for a decision. Finally, the authorization service evaluates the query against the policy and returns a response.

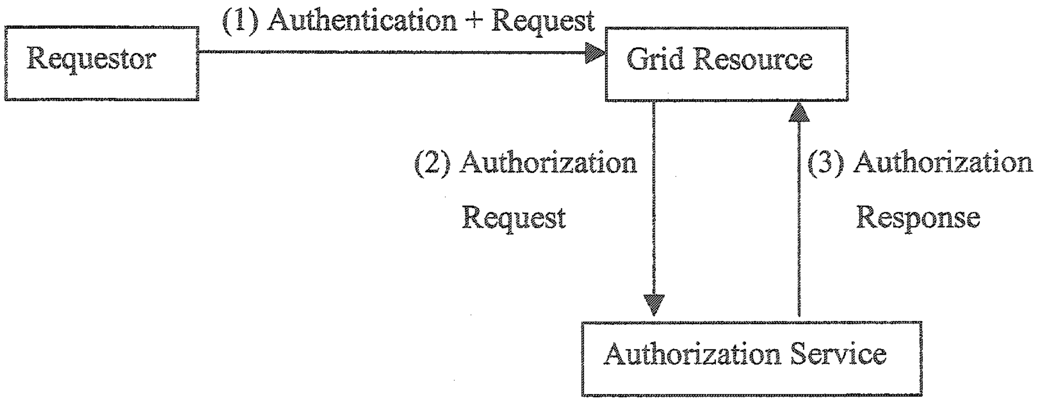

Requestor
(1) Authentication + Request
Grid Resource
(2) Authorization Request
(3) Authorization Response
Authorization Service

Figure 2-1: Authorization Pull Model Showing Sequence of Operation (1) to (3)

(2) **Authorization Push Model**

In push model (see Figure 2-2), the requestor first sends request to the authorization service. And then the authorization service returns a capability; it specifies all the permitted rights given to the requestor. The requestor then presents the capability to the resource. It is also called capability-based model. When the user wishes to access a resource, he needs to send this capability and his identity certificate together to the resource gatekeeper, the gatekeeper should verify that the subject named in the capability is the same as that in the identity certificate.

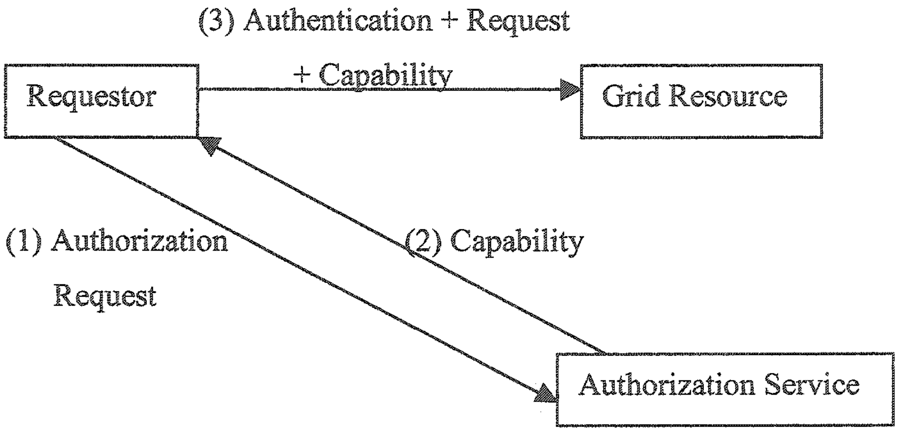


Figure 2-2: Authorization Push Model Showing Sequence of Operation (1) to (3)

2.1.3 Confidentiality

Confidentiality is also called privacy. For a two-party communication, only the sender and receiver know the contents of the information. Encoding technique is used to provide confidentiality. Even if attackers steal the message, they can only see the garbled data and cannot understand it. Current technology includes public key cryptography and conventional cryptography.

2.1.4 Integrity

Integrity ensures that only authorized parties are able to modify the information. This is used to prevent attackers from altering the information. Data alteration includes insertion, deletion, substitution and replaying of transmitted messages. The technology for integrity uses one-way hash functions to protect original data. If the data is modified, the result of the hash function will be changed greatly, and then the receiver knows that the message has been modified.

2.1.5 Other Requirements

Currently grid security architectures support virtual organizations through single sign-on and delegation.

Single sign-on is a service that a user can log on just once and then have access to any resource without repeatedly entering his password. Now many standards consider single sign-on to be necessary for large-scale distributed security architectures [Parker95].

Delegation [Gasser90] [Howell00] is an important operation in grid environments. It means one entity grants the ability to act on its behalf to another entity. A user can submit a program with the ability to run on the user's behalf. Then the program can access the resources instead of the user. Restricted delegation is used to conditionally delegate a subset of the user's rights. It can prevent malicious actions effectively and reduce some security problems in Grids. Delegation can be chained; one can delegate credentials to host A and then the process on host A can delegate credentials to host B.

2.2 Grid Security Architectures

A number of corporations, professional groups, and universities have developed frameworks and software in grid computing. The European Community is sponsoring a grid project for physics, earth and biology applications. In the United States, the National Technology Grid is building a computational grid infrastructure. There are some projects that have implemented the security for grid systems. Globus has been developed by the Argonne National Laboratory and the University of Southern California. GSI is part of this project and provides security services that support grids. Legion is an object-based grid operating system that provides secure infrastructure. CRISIS provides security for Web-OS. CAS is community authorization service, which separates resource administration from community administration and increases scalability.

2.2.1 Grid Security Infrastructure (GSI)

Grid Security Infrastructure (GSI) [Foster98b] was developed as part of the Globus project and provides secure authentication and communication for grids. The Globus Toolkit provides software tools used to build computational grids and grid-based applications. GSI is used for inter-site security. It bridges the different local security solutions of each site. The GSI software is a set of libraries and tools that focuses on cross-domain authentication and message protection [Foster97b], single sign-on, and delegation mechanisms [Gasser90] [Hardjono94] [Howell00]. It is based on public key infrastructure, X.509 certificates, and uses SSL for its mutual authentication protocol. GSI includes several significant features: each entity including user, resource and program has a globally unique name [Butler99]. This name is included in X.509 identity certificate. A certification authority (CA) ties an identity to a public key by signing a certificate. An authentication algorithm is defined by the Secure Socket Layer Version 3 (SSLv3) protocol. An entity can delegate a subset of its rights to a third party by creating a temporary identity called a proxy.

In the Grid environment, a user may be authenticated many times when accessing multiple resources. The user has to repeatedly type his pass phrase. The more times the user's private key is exposed, the more chances it will be compromised. [Novotny01] proposed using proxy credentials to solve this problem. A proxy credential is a short-term credential in order to take place of long-term credential of the user. A proxy includes two parts: a new certificate (having a new public key) and a new private key. The new certificate contains the user's identity, and is indicated that it is a proxy. This proxy is signed by the user or by another proxy. During mutual authentication, the user's public key is used to validate the signature on the proxy certificate. The CA's public key is then used to validate the signature on the user's certificate. This establishes a chain of trust beginning with the CA and first the user, then the user's proxies.

The Globus infrastructure provides a gateway server at each site. The overview of the security in Globus is shown in Figure 2-3. The server on the client site verifies a user's identity certificate and creates a temporary proxy certificate for the user. The server at the resource site authenticates the received certificate and maps the identity into a local user ID. There is one problem about the keys. The private keys are always stored on the local file systems without additional protection. It relies on file system access security provided by the host operating system.

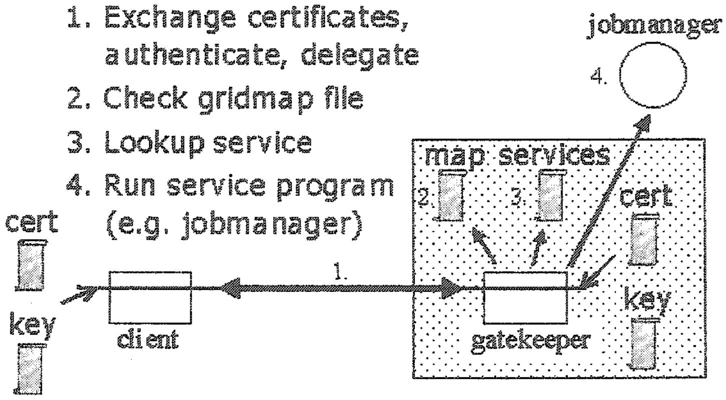


Figure 2-3: Security Model in Globus

Grid computing builds dynamic, inter-domain, distributed computing environments. It is important for a program to be able to act on the user's behalf on the Grid [Novotny01]. Globus has implemented certificate delegation. Single sign-on is also achieved through the use of proxy certificates. An entity may delegate a subset of its rights to a third party by creating a temporary identity proxy credential. It allows the user to delegate a proxy credential to processes or another entity on remote hosts. For example, if a user wants to collect more storage system on the grid by running a program, he is required to authenticate himself to different systems. After the user performs single sign-on using proxy, the program running on the user's behalf can access the permitted resources. A proxy certificate is used to delegate the user's authority for a very limited time period in order to authenticate on the user's behalf to remote resources. The long-lived user credentials is only required at initial log-on when a proxy is created.

2.2.2 Online Credential Repository (MyProxy)

MyProxy is an online credentials repository system. [Novotny01] gives a detailed introduction on MyProxy. It describes the architecture and security of the MyProxy system in details. There are two problems about grid credentials [Novotny01]. One is that the grid credentials must be stored in file system and kept private, when a user wants to use his credentials he must log in a secure manner to the file system. So there is a problem for traveler users who is away from the file system. They must have a copy of their grid credentials in order to access Grid. MyProxy was designed to solve this problem. A user can store his credential on the MyProxy server. When the credential is needed the user connects to the server and allows MyProxy server to use the stored credential on his behalf. Therefore, it provides a more secure environment than the user's home computer, and it also allow the user to use his credential when away from his home computer.

The other problem is that grid portals cannot integrate with GSI because web browser cannot support delegation, while GSI support delegation. MyProxy credential repository system can solve the incompatibility problem between web and grid security. It has a repository server and client tools used to delegate and retrieve credentials from the repository [Novotny01]. First, a user delegates a proxy to the MyProxy repository with user ID and pass phrase [Butler00]. Later, when the user needs his credentials, he uses a program to connect with the MyProxy repository with the same user ID and pass phrase, and can retrieve that user's proxy. Then this proxy will be used to act on behalf of user on the grid. Therefore, MyProxy System permits users to access their credentials from anywhere and allows users to delegate credentials to resources. It also makes Grid Portals to access GSI protected resources [Novotny01]. In addition, using short-term proxy credentials can limit private key exposure, and is better than copying long-term credentials to all the systems that the user may use.

Besides that, it provides many new functions such as management of permanent credentials and automatic selection of credentials from multiple credentials [Novotny01]. It is also called a "credential wallet" [Butler00]. Users can put all their credentials for multiple sites into this "wallet". When credentials are requested, the MyProxy server will automatically select which credential to delegate according to the user's identity [Butler00].

2.2.3 LEGION

The Legion project at the University of Virginia is an object-based operating system for grids. The goal of the project is to create a system using principles from object-oriented programming, parallel processing, distributed computing, and computer security fields. It is a software tool to provide a secure and high-performance infrastructure for grid computing.

Legion is object oriented. It uses object to represent the entities in grid such as hosts, files and programs. In Legion every object is identified by a unique Legion object ID, and each object is responsible for its own security policies. The system makes the resource provider has authority over the use of his resources. Legion provides a basic implementation of a public-key based policy. The security model is aimed at protecting objects and their communication. Data encryption and integrity services are available to messages passed between objects in Legion. The authentication in Legion uses object's identifier and its public key. It also supports X.509-based certification. [Stoker01] Legion has a default security infrastructure [Ferrari99] based on access control lists. An ACL is associated with each object and lists all the operations that the principal can run on the object.

Legion is similar to Globus, but it focus on object-based software technologies for grid computing. It can provide more flexible security mechanisms. But Legion security model does not include actual architecture and protocols [Foster98b]. Legion is above the physical resources of the grid and below the application layer, so Legion is often called middleware.

2.2.4 CRISIS

CRISIS is a security part of Web-OS. It provides wide area security, remote process execution, resource management, and distributed storage services. WebOS was developed at the University of California at Berkeley.

Web-OS is similar to Globus, while CRISIS is similar to GSI. It uses SSL and X.509 certificates. The authorization in the CRISIS architecture uses both access control lists and a capability lists. The entity provides its identity plus its capabilities to the resource, and access decision is made by using the two lists with its identity and capabilities. [Foster98b] lists some difference between GSI and CRISIS. The resources in CRISIS don't include process, while GSI includes process and allows it to request access to other resources. CRISIS is more suitable for Web architecture rather than grid computing.

CRISIS can be used in different administrative domains. Each domain has its own CA. If an entity from one domain wants to authenticate with a member of another domain, a trust path needs to be established between the two domains. [Belani98] describes the design, architecture and implementation of CRISIS. It uses transfer certificates as a simple mechanism to create roles and capabilities. It increased system security and performance across the wide area. The technique includes redundancy, caching, least privilege, and complete accountability. But the current implementation of CRISIS is only for Solaris.

2.2.5 Community Authorization Service (CAS)

The Community Authorization Server (CAS) [Pearlman02] is a new authorization service developed by the Globus Project for Grid environments. [Pearlman02] describes CAS architecture in detail. It provides a scalable mechanism for specifying community policies, and allows resource providers to delegate some of the authority for access control. CAS builds on public key authentication and delegation mechanisms by GSI, it solves three authorization problems in distributed virtual organizations: scalability, flexibility and policy hierarchy [Pearlman02].

The CAS model is used to manage the assignment of privileges to users from several communities. CAS allows a resource provider to assign coarse-grained access rights for its resources. CAS uses three major extensions to GSI [Pearlman02]: restricted proxy credentials for delegation, a policy language for specifying the rights in the restricted proxy, and libraries and APIs. CAS uses the Generic Authorization and Access control API (GAA-API) [Ryutov98] [Ryutov00] and GSS API [Linn93] as its API. In CAS, checking the policy and granting the rights to the user is before contacting the gatekeeper [Thompson02a]. And CAS uses the centralized community policies to govern all the resources [Pearlman02]. A community uses CAS servers as a centralized trusted party within the community. CAS server can specify access policies for each community member. Users make requests to the resources using the CAS proxy credentials, and then resources grant access based on the access policy in resource and the community policy stated in the proxy credentials.

2.3 Security Policy Specification Approaches

Many approaches on expressive languages for making and verifying security assertions [Kornievskaia01] include PolicyMaker [Blaze96], and its successor KeyNote [Blaze99c], PONDER [Damianou00], and SAML [Hallam02].

2.3.1 PolicyMaker

Matt Blaze's PolicyMaker [Blaze96] is the earliest system for distributed trust management [Kagal02a]. It is able to specify and interpret security policies about access rights. But the policy is a little complicated and not easy for non-programmers to use. The goal for PolicyMaker [Blaze96] [Blaze98] is putting all the policy and credential information into signed certificates that can be stored in a distributed way. PolicyMaker certificates are generalized and difficult to apply the access policy for a resource.

2.3.2 KeyNote

KeyNote [Blaze99c] is a trust-management system. It provides a language for describing and implementing security policy, trust relationship and credentials. The KeyNote defines principal including its public key. Authorization policy has some fields. Each field is represented by a keyword and value. A credential includes some attributes about a principal and is signed by a trusted authority. Assertions and credentials are both represented by the keyword policy language. The KeyNote system is on a C-like expression and regular syntax for describing conditions. KeyNote certificates are "assertion monotony" [Thompson99]. Each assertion can only increase the permitted actions and can not specify negatives.

2.3.3 PONDER

Ponder is an object-oriented language [Damianou00] [Damianou01] for specifying security policies and management policies. It uses role-based access control. [Dulay01] presents an overview of the types of policies specified with Ponder, and describes how an object implements a policy. It has four basic policy types: authorizations, obligations, refrains and delegations; and has three composite policy types: roles, relationships and management structures that are used to compose policies. The relationships between the various types are described in [Dulay01]. Ponder also supports abstractions policies definition.

Existing policy work has focused on specification and application-specific policy enforcement. [Dulay01] introduces using policies for the security and management of large-scale distributed networks, and describes a general-purpose deployment model for managing Ponder policies. In this model each policy type is represented by a policy object. The policy object maintains the state of the policy and manages all policy operations. (See Figure 2-4) Then the policy object will invoke corresponding operations on its enforcement agents for the policies.

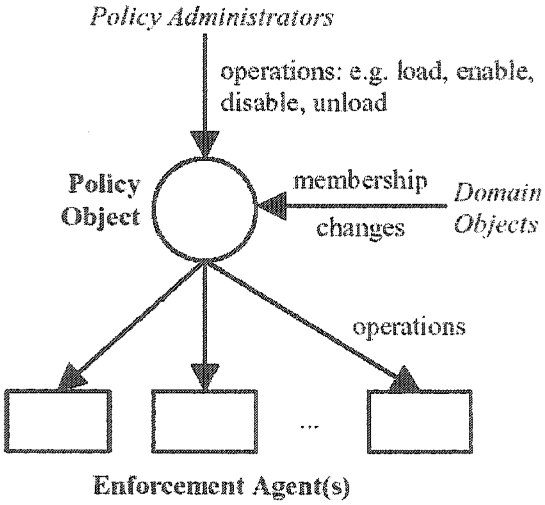


Figure 2-4: Ponder Policy Operations [Dulay01]

The model supports authorization and obligation policies using multiple access control mechanisms. Management policies define what actions can be performed under specific conditions. The Ponder policy specification language uses domains to group objects and the application of policies to domains. It defines objects for policies, for domains, and for policy agent.

2.3.4 Security Assertion Markup Language (SAML)

It is very difficult to express the constraints and trust relationships because there is too much information in relationships. XML has greater flexibility. Security Assertion Markup Language (SAML) [Hallam02] has been published by the Organization for the Advancement of Structured Information Standards (OASIS). It is a XML standard specification for security assertions. This standard defines XML protocols and assertion structures. Assertions have three types: authentication, authorization and attribute. Authentication is the specified subject authenticated; Authorization Decision is permitting or denying a request to access the resource; Attribute specifies some attributes associated with the subject.

The SAML specification [Welch03] defines a number of elements for making assertions and queries regarding authentication, authorization decisions and attributes. The Resource element is used to identify the target where the policy is being asserted. This element is simply a URI. The Subject element contains identity information about the requestor. Assertion element can contain any number of conditions elements. Conditions elements are used to express restrictions on the validity time of the assertion. Each Evidence element can contain any number of assertions elements that affect the policy decision process. SAML defines three kinds of assertions. Authentication assertions require the user's identity. Attribute assertions contain specific details about the user. The authorization decision assertion identifies what the user can do. SAML defines a message exchange between a policy enforcement point (PEP) and a policy decision point (PDP). A user sends an AuthorizationDecisionQuery to a server, and then the authorization server will evaluate the request according to the policy, and return Assertions containing either an AttributeStatement or an AuthorizationDecision. The message flow in SAML is shown in Figure 2-5.

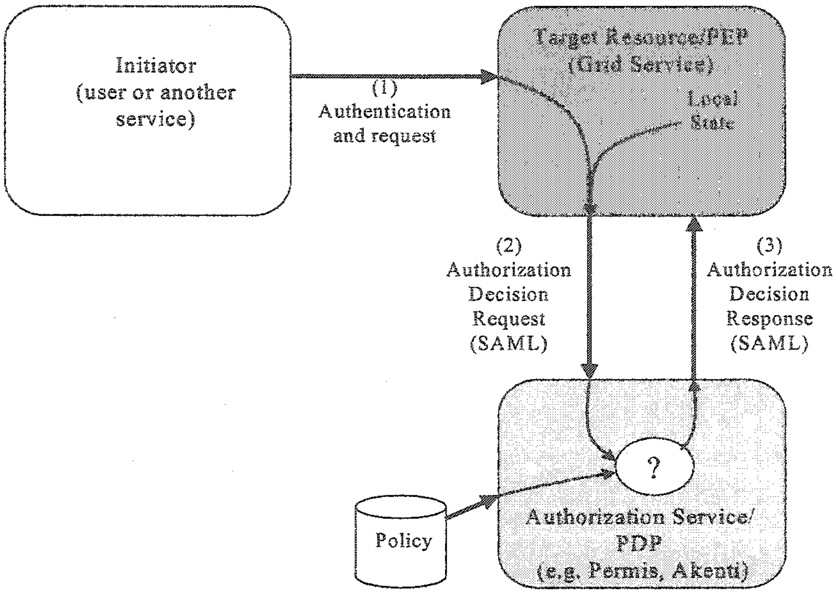


Figure 2-5: SAML Message Flow [Welch03]

SAML becomes the common language to defining how identity and access information exchanged. The SAML specification itself doesn't define any new technology for authentication. It only makes assertions about credentials and doesn't actually authenticate or authorize users. But it establishes assertion and protocol schemas for the structure of the documents.

3. RELATED TECHNOLOGIES

This chapter introduces several critical security technologies which are related to the ideas in this thesis. Section 3.1 explains basic private key infrastructure and section 3.2 provides background information on authentication mechanisms based on x.509 digital certificates. Section 3.3 introduces the features of widely used SSL protocol and explains how it works. Section 3.4 provides an overview of RBAC model.

3.1 Public Key Infrastructure (PKI)

Public-key cryptography is using asymmetric encryption, it makes use of a pair of keys: private key and public key. If using one key for encryption, we must use the other key for decryption. The two keys are linked together by way of a complex mathematical equation. The public key is published and known to anyone. It can be used to check the digital signature, while the private key is known only to its owner who should never reveal it. It can be used to create a digital signature. RSA algorithm is one of the most widely used public key algorithms which makes it impossible to deduce the private key from the public key. It is named after the inventors Ron Rivest, Adi Shamir, and Len Adleman [Housley01].

If a message sender wishes to use public-key technology to encrypt a message to a recipient, the sender needs a copy of the recipient's public key. When a party wishes to verify a digital signature generated by another party, the verifying party needs a copy of the public key of the signing party. The two operations are different: signing and encrypting messages. To encrypt a message with recipient's public key make sure that only the recipient can decrypt it and read the message. To sign a message with signer's private key proves that the message is indeed sent from the signer. People can check a digital signature in order to verify the identity of the person who signed the message. Digital signature is a cryptographic transformation of data. It provides the services of origin authentication and data integrity.

In a PKI [Thompson99], all entities are identified by a globally unique name called a Distinguished Name (DN) in the certificate. Entities prove their identity using grid credentials consisting of a certificate and the private key. According to [Thompson99], PKI identity certificates are more widespread than Kerberos identities.

3.2 X.509 Public Key Certificate (PKC)

From remote operations on grid resources, security remains a major issue. The access control by username and password is not enough because most information is transmitted in clear text on the Internet. To control the access by combining the use of the password and the digital certificate is safer. Many companies become aware of this reality and the digital certificates have been widely used in the Internet. ITU (International Telecommunication Union) and ISO (International Organization for Standardization) published the X.509 standard in 1988, which has been adopted by IETF (International Engineering Task Force). X.509 is the data format for public-key certificates.

A pair of keys is not enough to associate with an identity; the association is done by digital certificate that associates the public key with an identity. A digital certificate makes it possible to check that somebody has the right to use a key, thus helping to prevent that a person uses a fake key to appear as someone else. An X.509 public key certificate (PKC) as defined by [Housley01] includes a public key, a subject name and a validity period. It is signed by a trusted third party called a Certification Authority (CA). A PKC binds a name and a public key. Due to this specific binding PKC can be used for authentication. These certificates are also called "identity certificates" [Thompson02b] which is used to securely identify an entity. The format of X.509 PKC is shown in Figure3-1. Digital Certificates are based on the public key infrastructure. The link between the public key and the individual is certified by CA.

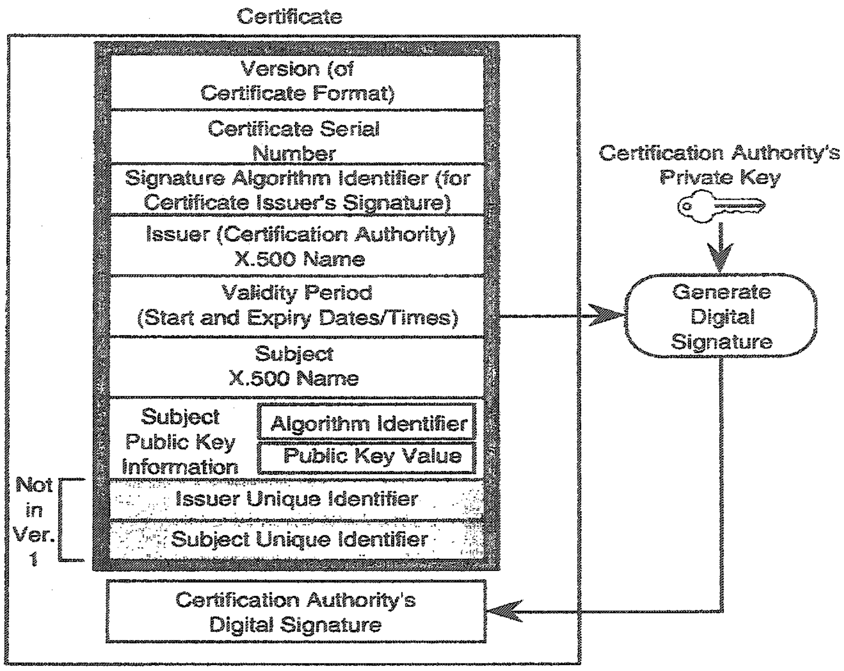
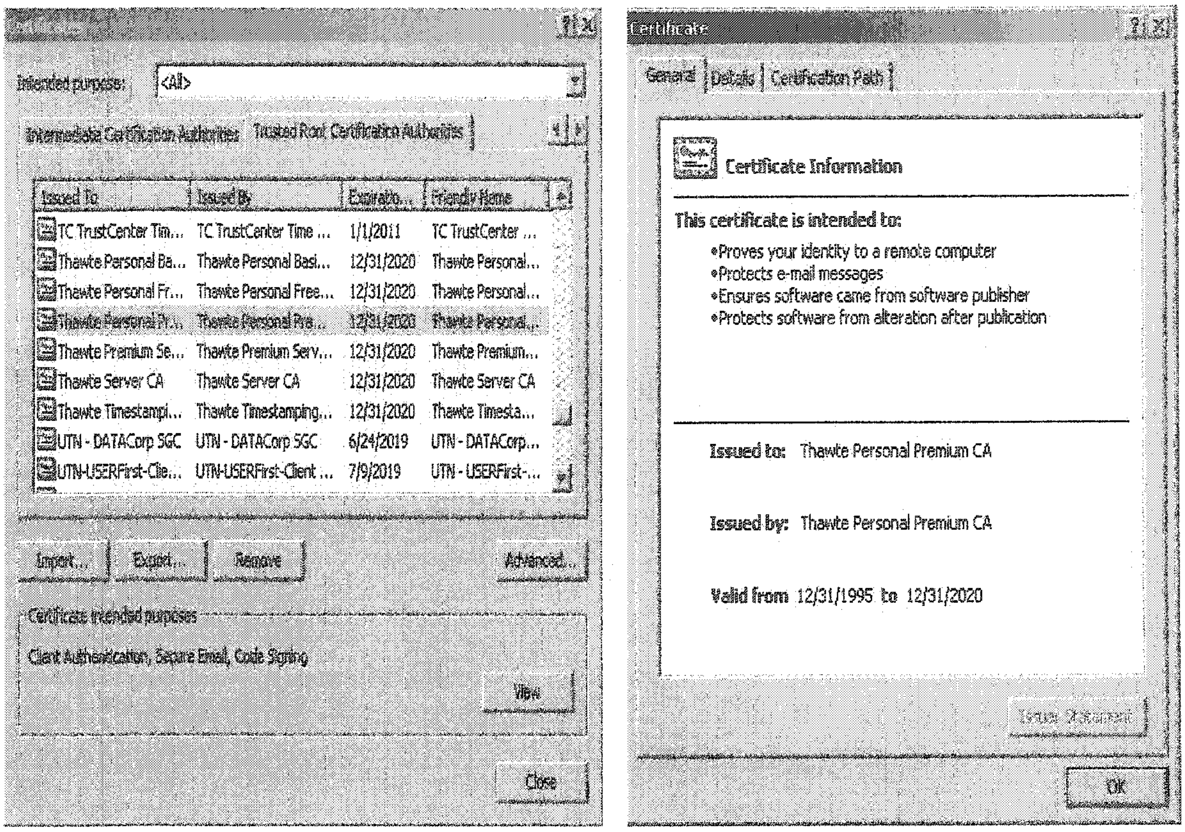

Certificate
Version (of Certificate Format)
Certificate Serial Number
Signature Algorithm Identifier (for Certificate Issuer's Signature)
Issuer (Certification Authority) X.500 Name
Validity Period (Start and Expiry Dates/Times)
Subject X.500 Name
Subject Public Key Information
Algorithm Identifier
Public Key Value
Not in Ver. 1
Issuer Unique Identifier
Subject Unique Identifier
Certification Authority's Digital Signature
Certification Authority's Private Key
Generate Digital Signature

Figure 3-1: X.509 Public Key Certificate Format

Certificate Authority (CA) is a recognized authority trusted by one or more parties. It can create and sign Public Key Certificates, and also revoke certificates it has issued. CA also manages a certificate revocation list (CRL). A grid CA is defined in [Thompson02b], it is independent of any single organization and it is used to sign certificates for individuals who are allowed access to the grid resources. Every node in the network needs to know the public keys of the CAs it trusts because the CAs' public keys are used to verify CAs' signatures. The user interface of trusted CAs and one of CA certificates are shown in Figure 3-2.



Intended purpose: <All>

Issued To	Issued By	Expiratio...	Friendly Name
TC TrustCenter Tim...	TC TrustCenter Time ...	1/1/2011	TC TrustCenter ...
Thawte Personal Ba...	Thawte Personal Basi...	12/31/2020	Thawte Personal...
Thawte Personal Fr...	Thawte Personal Free...	12/31/2020	Thawte Personal...
Thawte Personal Pr...	Thawte Personal Pre...	12/31/2020	Thawte Personal...
Thawte Premium Se...	Thawte Premium Serv...	12/31/2020	Thawte Premium...
Thawte Server CA	Thawte Server CA	12/31/2020	Thawte Server CA
Thawte Timestampi...	Thawte Timestamping...	12/31/2020	Thawte Timesta...
UTN - DATACorp SGC	UTN - DATACorp SGC	6/24/2019	UTN - DATACorp...
UTN-USERFirst-Clie...	UTN-USERFirst-Client ...	7/9/2019	UTN - USERFirst-...

Certificate intended purposes

Client Authentication, Secure Email, Code Signing

Certificate Information

This certificate is intended to:

- Proves your identity to a remote computer
- Protects e-mail messages
- Ensures software came from software publisher
- Protects software from alteration after publication

Issued to: Thawte Personal Premium CA

Issued by: Thawte Personal Premium CA

Valid from 12/31/1995 to 12/31/2020

Figure 3-2: User Interface of Trusted CAs and CA's Certificate

A certificate binds an identity of the holder to the public key. So it is used to verify that a given public key does belong to a given individual. Every user and service on the grid is identified via a certificate. Revocation of certificates becomes an issue if the private key of an entity is compromised. Certificate revocation lists (CRLs) are distributed regularly to all nodes. A node should check the revocation list to make sure the certificate is still valid before accepting it. Therefore, during the certificate verification, first check its expiration dates, revocation list, and if the certificate is signed by trusted CA. If it passes all the checks, then the public keys will be used to build a handshake to prove that the certificate holder has the corresponding private key. Passing all these tests, a secure connection has established between the two parties.

In summary, the certificates can provide confidentiality by encrypting messages. It can also provide integrity, authentication and non-repudiation by digital signature. Certificates are used to prove the identity of the sender and to make sure that the message was not altered by anybody. Most important, it can provide access control to Internet sites, Intranets or other networks. Grid computing requires mutual authentication, so the two parties need send their certificates to the other in order to confirm the identity to each other.

3.3 Secure Socket Layer (SSL)

Secure Sockets Layer (SSL) protocol was originally developed by Netscape, and now has become security protocol and authentication tool used in web browsers and web servers. The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP or LDAP. Since the protocol operates at the transport layer, any program that uses TCP can use SSL connections. SSL protocol provides a secure means for establishing an encrypted communication. It uses PKI credentials for authentication and supports numerous encryptions and digest mechanisms. SSL comes in two strengths, 40-bit and 128-bit, which refer to the length of the "session key" generated by every encrypted transaction. The longer the key, the more difficult it is to break the encryption code.

The SSL protocol includes two sub-protocols: the SSL record protocol and the SSL handshake protocol. The SSL record protocol defines the format of transmitted data. The SSL handshake protocol defines exchanging messages between server and client during establishing an SSL connection. This exchange of messages include the following actions:

- Authenticate the server to the client.
- Allow the client and server to select the cryptographic algorithms that they both support.
- Optionally authenticate the client to the server.
- Use public-key encryption techniques to generate shared secrets
- Establish an encrypted SSL connection.

The SSL protocol uses both public-key and symmetric key encryption to establish encrypted communication. Symmetric key encryption is much faster than public-key encryption, but public-key encryption provides better authentication techniques. An SSL session always begins with an exchange of messages called the SSL handshake. The handshake allows the server to authenticate itself to the client using public-key techniques, and then allows the client and the server to create symmetric keys used for rapid encryption during the following communication. SSL uses X.509 certificates. Server certificates provide a way for users to authenticate the identity of a server.

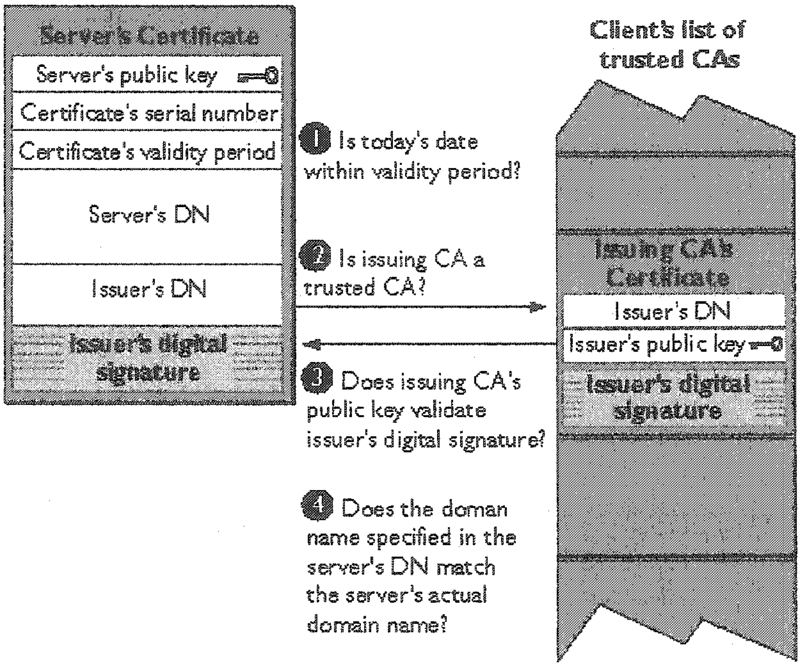

Server's Certificate
Server's public key
Certificate's serial number
Certificate's validity period
Server's DN
Issuer's DN
Issuer's digital signature
1 Is today's date within validity period?
2 Is issuing CA a trusted CA?
3 Does issuing CA's public key validate issuer's digital signature?
4 Does the doman name specified in the server's DN match the server's actual domain name?
Client's list of trusted CAs
Issuing CA's Certificate
Issuer's DN
Issuer's public key
Issuer's digital signature

Figure 3-3: Server Authentication using SSL (Basic Four Steps)

Figure 3-3 illustrates the four basic steps during authenticating a server's identity by a client. First step, the client checks the server certificate's validity period. If the current date is invalid, the authentication process will inform the client. Second step, checks the distinguished name (DN) of the issuing CA matches the DN on the client's trust CA list.

Each client should maintain a list of trusted CA certificates. This list determines which server certificates the client will accept. Third step, the client uses the public key from the CA's certificate in its trusted list to validate the CA's digital signature on the server certificate. If the CA's digital signature can be validated, then the server certificate is valid. Fourth step, check if the domain name in the server's certificate match the domain name of the current server, which confirms that the server is actually located at it own network address. This step can protect against Man-in-the-Middle attack.

SSL protocol provides authentication, message integrity and message confidentiality. It uses digital certificates to encrypt the data to keep it safe from interception. And it can prevent tampering. SSL will generate a warning if the information is changed between the server and the client. It also supports mutual authentication [Kornievskaia01]. If SSL is configured in the server to require client side authorization, then mutual authentication is performed [Thompson99]. The client must send its credentials to the server to confirm the identity of the user. Client authentication can protect against user impersonation.

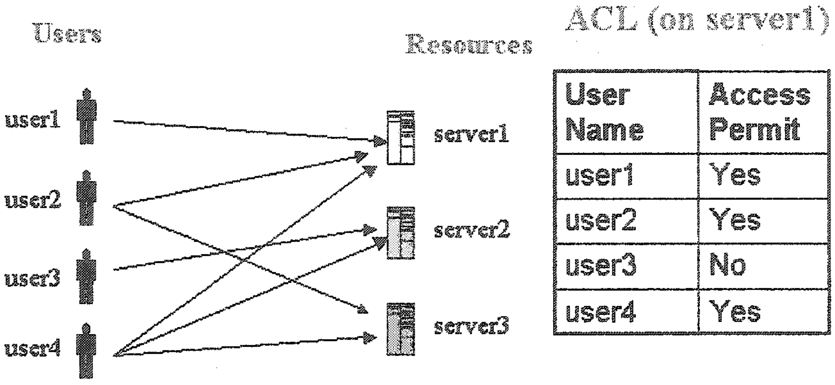
3.4 Role Based Access Control (RBAC)

One of the most challenging problems is managing users in large networks. Access control is needed to control the operations of legitimate users in order to protect the stored information. The effectiveness of access control systems can be measured on two criteria: reliability of security and ease of administration. Role based access control (RBAC) attracts more attention because it reduces the complexity and cost of security administration in large networked applications.

3.4.1 Access Control List (ACL)

Traditional security administration uses access control lists for each user on the system individually. Access control list (ACL) is used to perform authorization. It contains information of both identity of the users and the name of the programs to be run. ACL is based on operating system, where all access rights for each authenticated entity are listed.

It establishes specific permissions for each user for access control. This approach is effective in a static environment, but it is very difficult to manage in dynamic grid environments which may require frequent adding new users and updating of access permissions. The maintenance of those ACLs requires a lot of effort. Additionally, if people wish to grant users access some specific resource for a short period of time the overhead for this is too much.



User Name	Access Permit
user1	Yes
user2	Yes
user3	No
user4	Yes

Figure 3-4: Access Control List (ACL) Schema

There are several other problems about using ACLs. First, a central administrator is required and must be trusted; second, all the users who wish to share the resource must have a local account on the system in advance [Thompson02a]; third, it is difficult to maintain applications and difficult to scale; fourth, users have to require separate ACLs for each system. Figure 3-4 describes the ACL schema, and each server should have an ACL for storing the name of users who can access this server. Therefore, it is not suitable for grid environments.

3.4.2 RBAC Model

With RBAC, each user is assigned one or more roles, and each role is assigned one or more privileges. In RBAC model [Sandhu96], it includes three sets of entities called users, roles and permissions. A user is a human being. A role is the basis of access control policy. It represents the authority and responsibility of a job position in the organization. Roles can be set up based on locations, projects and management level. Permission is authorized actions on particular targets. 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 if a particular user is allowed access to a specific piece of system data. The relationship among them such as user-role and role-permission is shown in Figure 3-5. A user can be assigned many roles, and a role can be given to many users, so it is many-to-many relationship. A role can be assigned many permissions, and a permission can also be given to many roles. This is also a many-to-many relationship.

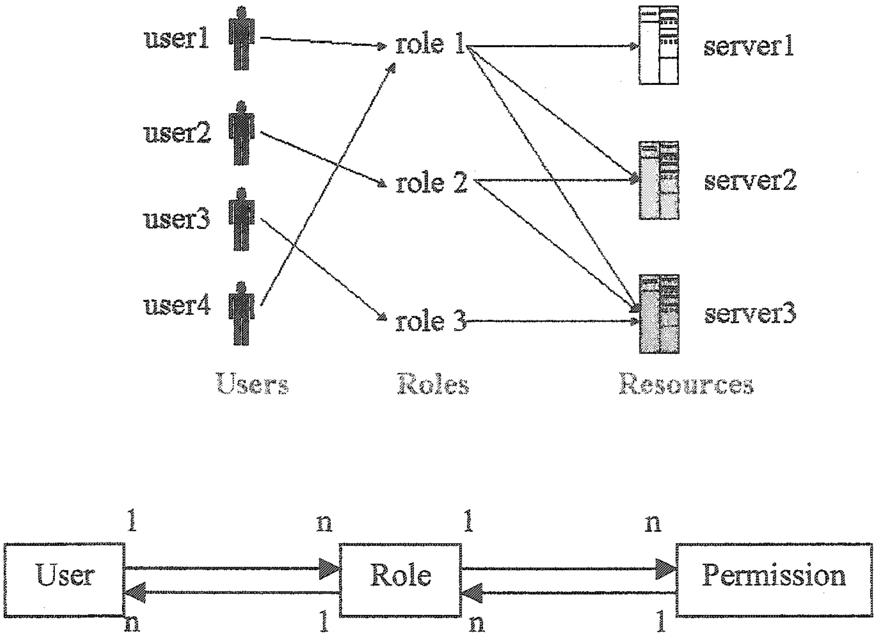


Figure 3-5: Role Based Access Control (RBAC) Model

Access decisions are based on the roles of the users, rather than the individual users. Permissions associated with a role are better than permissions associated with a user because users always change their job, but a role's permissions are seldom changed [Sandhu96]. And users can be easily reassigned to different roles when they change the jobs.

Role-based access control (RBAC) is suitable for managing and enforcing security in large-scale systems. It has proved to be a successful technology in security infrastructures. There are some advantages of RBAC as follows:

1. Flexible access control model. An administrator using RBAC can easily register and revoke the user's role membership based on the user's jobs. In addition, users may be able to use the same role across multiple systems.

2. Simplifies security management. Roles can be updated without updating the rights for each user. Therefore, it enables access control maintenance easier. It can simplify the allocation and removal of privileges. When accessing a target, a user presents his role, and the target reads the policy to see if this role is allowed to perform this action. By removing the role from the original role holder, his privileges are automatically removed. Many people can hold the same role, and have the same set of privileges.

3. Reduces administrative cost. The efficiency and cost savings come from the need for maintaining a RBAC system, which grants users access rights based on their role rather than their individual identity.

4. Scalability. There are typically far fewer roles than participants. The number of participants in each role can run into the thousands or more. RBAC makes it easier to administer the privileges to a limited number of roles, and also simpler to allocate roles to the large numbers of users. Therefore, RBAC supports

scalability in both users and privileges which may dynamically change in grid environments.

5. Greater security. It prevents users from obtaining inconsistent privileges that can enable access violations. All the users who belong to the same role will have the same privileges. In addition, it separates the duties, which can constrain role allocation. An individual cannot occupy mutually exclusive roles.

6. Privilege inheritance. Role hierarchies can be defined. Junior roles appear at the bottom of the hierarchic structure and senior roles are at the top. The superior roles inherit the privileges of the subordinate roles, and also have their own additional privileges. Hierarchical RBAC allows role specifications to be easier, since a superior role does not need to enumerate the privileges it has inherited from its subordinate roles.

4. PROPOSED ENHANCED SECURITY ARCHITECTURE

In order to control access to grid resource, both authentication and authorization are required. Most research in security area always focuses on authentication techniques including digital signatures, Public Key Infrastructures and public key certificates. Our research focuses on authorization technique based on attribute certificates in order to solve the problems of MyProxy online credential repository.

In grid environment, it is necessary to have a flexible, scalable security infrastructure to store and manage user profiles for user authentication and authorization. Directories will allow people to collaborate and share information both internally and externally. With the feature, we use LDAP server in our architecture for managing users' ACs for large-scale applications.

In section 4.1, we introduce attribute certificates contents and its benefits in details, and also give a comparison between PKC and AC. Section 4.2 provides information on LDAP protocol and why we use it in our architecture. Section4.3 describes the three components in our proposed security architecture, and six basic steps for providing both authentication and authorization service.

4.1 Attribute Certificate (AC)

Attribute certificates were introduced by International Standardization Organization in 1997. An attribute certificate (AC) is an X.509 certificate which is similar to a PKC, However, an attribute certificate does not contain a public key, but contains attributes or other authorization information. AC binds attribute information to the certificate's subject.

Attribute certificate is a means for authorization. It uses PKI in an authentication system to support authorization facilities, and is used to make authorization decisions and delegate role-related attributes. It can separate attribute management from identity management, which means can separate authentication and authorization when required. Attribute certificates are issued by an Attribute Authority (AA), while identity certificates are issued by a Certification Authority (CA).

There are two forms of attribute certificates described as follows:

- One form is the attribute certificate which is included in the extensions to identity certificate when the AA and CA of the two certificates are same. There are some benefits for using this method such as requiring fewer entities, simplifying certification path and improving the interoperability with existing systems.

- Another form is separating attribute certificate from identity certificate. So CA is used for authentication and AA is for authorization. Especially when the authorization information is often required to change, using separated attributed certificate is very convenient for updating. And most identity certificate is long-term used while attribute certificate is short valid period.

The format of an attribute certificate is shown in Figure 4-1. An attribute certificate comprises the common information like subject-name (distinguished name of the certificate holder), issuer name (distinguished name of the attribute authority), unique serial number, version number, validity period, and the algorithm used to sign this AC. Besides that, an attribute certificate also includes attribute information like group membership, roles or rights of access in certificates. A user can have multiple attribute certificates for different roles. Delegation is supported through the basic attribute constraints extension. This extension holds an integer that indicates the depth of delegation. Zero indicates no delegation, and one indicates one level of delegation. It is possible to manage access attributes in a flexible and scalable manner by using this approach.

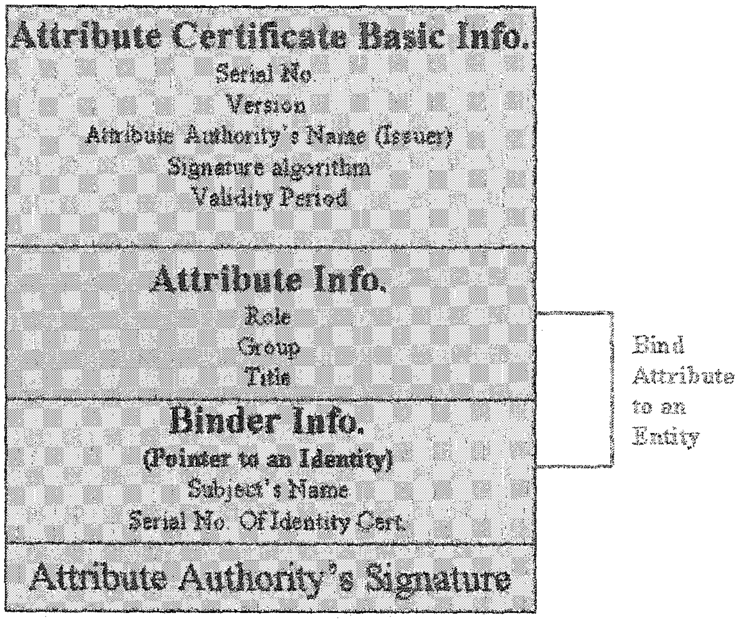


Figure 4-1: Attribute Certificate Format

A public key certificate (PKC), also called identity certificate, is used for authentication and binds a holder's name with his public key, and it is an authentication-oriented credential. While attribute certificate binds a holder's name with one or more privilege attributes, and it is an authorization-oriented credential. The comparison between them is described in Table 1. A PKC is like a passport because it is used to identify the holder and the valid period is long-lived. However, an AC is like an entry visa because it is short-lived and used to get the permission. In addition, an AC is issued by an Attribute Authority (AA), while an PKC is issued by a Certification Authority (CA). So a CA is used for authentication and an AA is for authorization. The requirements for being a CA is very high and the process is complex, but anyone holding privileges can be an AA and be allowed to issue attribute certificates to others. This also facilitates delegation of user's rights. Therefore, when the authorization information is often required to change, using separated certificate is very convenient for updating because the AC is short valid period.

	Authentication	**Authorization**
Certificate Name	Public Key Certificate (PKC)	Attribute Certificate (AC)
Certificate Issuer	Certification Authority (CA)	Attribute Authority (AA)
Certificate Receiver	Subject	Holder
Binding Information	Subject's Name to Public Key	Holder's Name to Privilege Attributes
Validity period	Long-lived	Short-lived
Key Required	With Public Key	Without Public Key

Table 1: A Comparison between PKC and AC

Authorization is always based on authentication. A PKC is used to identify a user while an AC is used to define the role of the user and grant the user access rights and privileges. So before using attribute certificate to do the authorization, identity certificate that includes public key is used to authenticate the subject first. An AC and a PKC can be bound together by associating the AC's holder with PKC's serial number or the subject name. The relationship between them is shown in Figure 4-2.

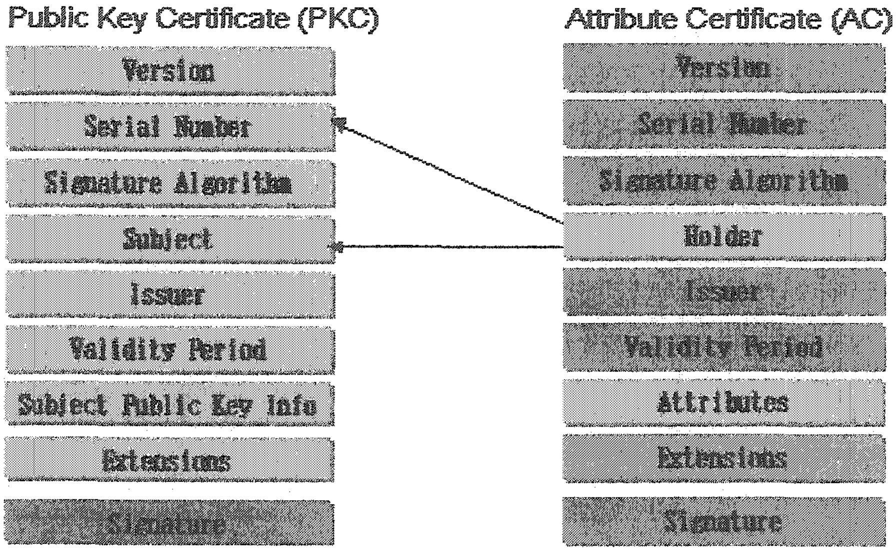


Figure 4-2: Binding PKC and AC

4.2 Lightweight Directory Access Protocol (LDAP)

Lightweight Directory Access Protocol (LDAP) is the Internet standard for searching information in directory, just like the Hypertext Transfer Protocol (HTTP) is the Internet standard for delivering documents. It was developed by the University of Michigan and the Internet Engineering Task Force (IETF). LDAP is defined by RFC 1777 and RFC 2251 and is a standard protocol that operates over TCP/IP. It creates a standard way for applications to request and manage directory information. Directory service plays an important role in providing access to information at any time from anywhere across the world. Many companies store user information in an LDAP server which can be used by Web servers, mail servers, and other applications.

LDAP provides access to complex X.500 directories at a low-cost way. The X.500 standards proved to be cumbersome, LDAP avoids large overhead of the X.500 Directory Access Protocol (DAP). LDAP server simply accessed an X.500 directory service using the OSI model and returned the information to the LDAP client. The relationship between LDAP and X.500 server is shown in the Figure 4-3. LDAP bridges the gap between users and global directories, and provides the means for users on any computer to access X.500 directories. It has been widely used for many different directory applications.

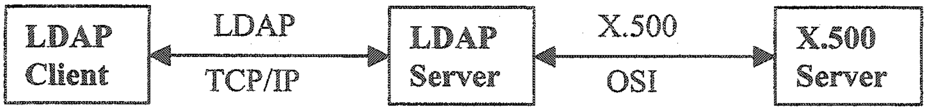


Figure 4-3: Relationship Between LDAP and X.500 Server

LDAP focuses on how information was stored and accessed in a large directory or multiple directories. It not only provides the hierarchal information structure but also allows the directory to be accessed. Directories are set up in a hierarchy structure, it begins with a global name called Distinguished Name (DN) such as a organization name and goes down into more detail such as an Organizational Unit (OU), to a Common Name (CN) and finally an attribute type and attribute value. This facilitates the storage of the data. It is also easy to use access control and policies to each category. In the representation of distinguished names of the X.500 Directory system, several unique keywords may be necessary. [Zeilenga02] standardizes a set of strings for some attribute types that can be used in the LDAP encoding of X.500 distinguished names. Those strings are listed in Table 2.

Keyword	Attribute Type
CN	CommonName
OU	OrganizationalUnitName
O	OrganizationName
C	CountryName
L	LocalityName
ST	StateOrProvinceName
STREET	StreetAddress
DC	domainComponent
UID	userid

Table 2: Directory System Name

Directories make enterprise information available to multiple different systems. The most common information stored in a directory service is about user identity and related information. In order to retrieve the information, directory access protocol is used to send the entries from directory server. When an LDAP client needs a specific entry in an LDAP server, the LDAP client generates an LDAP message containing a request and sends it to the LDAP server. The server retrieves the entry from its database and sends it back to the client. LDAP also allows using universal resource locators (URLs) to conduct directory lookups.

4.3 Proposed Security Architecture

There are many situations requiring both authentication and authorization. For example, there may be an application server that provides all kinds of source code for developers, say students and professors at university. Everyone who wants to use the code must be verified for his identity. This procedure is called authentication. In addition, each user has different permissions or rights. For example, undergraduate students may only access some basic level code for helping their studying programming; graduate students can access high level code for helping their research and permit to add new code into the database of the application server; and professors may not only access any level code but also can execute the code on the server. This procedure is called authorization. Different user may be permitted to do different things after the authentication.

MyProxy is a credential repository for the Grid. It allows users to delegate credentials to resources. But it supports only authentication without authorization. We propose a new architecture for support of MyProxy by concentrating on both authentication and authorization for grid systems. This architecture enables distributed resource providers to control their resources and specify access requirements flexibly.

An attribute certificate (AC) extends authentication-oriented public-key infrastructures (PKI) to support authorization facilities. Attributes in AC include authorization information about AC holder such as role, title or group membership. In particular, it can be used to delegate role-related attributes within distributed computing environments, minimizing unnecessary trust in intermediaries.

The proposed architecture using attribute certificate and LDAP directory server is shown in Figure 4-4. We have three major components:

1. MyProxy online credential repository. Users will delegate their identity proxy certificates into MyProxy repository. In our architecture, we will use it to do the delegation and authentication with their identity credentials. And it can also provide single sign-on without any further intervention by users.

2. LDAP directory server. LDAP provides easier and more secure access to grid resources. It provides a method to manage and distribute information in a low cost, allows information in the LDAP directory can be made available to other computer systems. We use it to store users' attribute certificates which can be retrieved later to do authorization.

3. Grid portal. Grid portal is at the top of the layer for grids. It can facilitate the execution of programs on remote resources through a web-based interface. In our architecture, grid portal will provide both authentication and authorization service using identity certificates and attribute certificates.

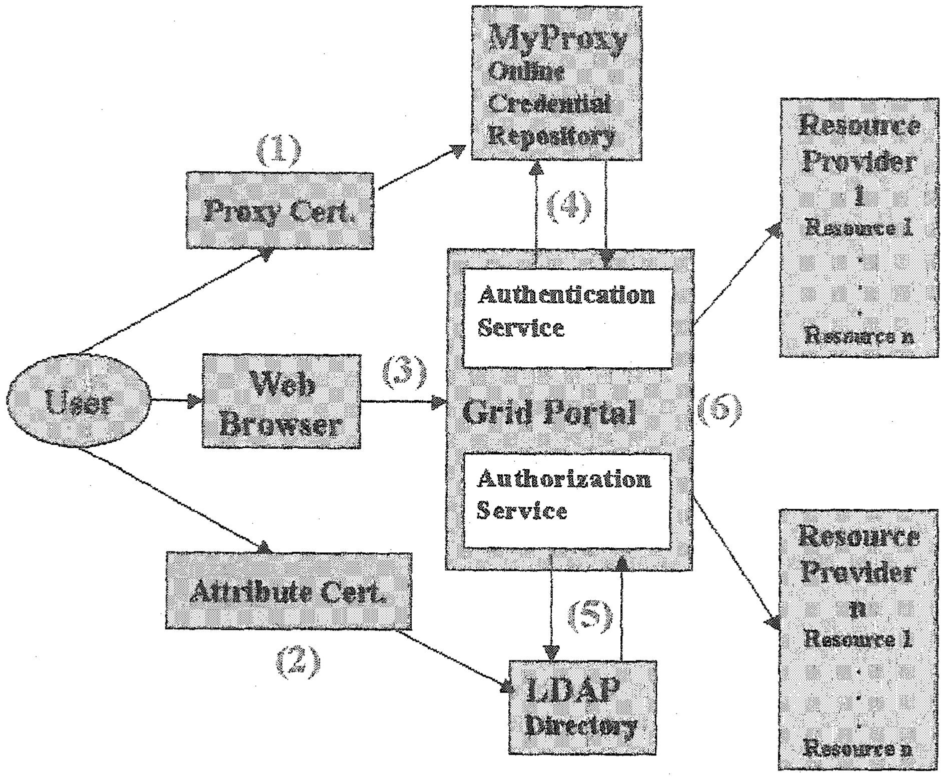


Figure 4-4: Proposed Enhanced Security Architecture

There are six basic steps (see Figure 4-4) for achieving both authentication and authorization service in our architecture as follows:

1. A user delegates a proxy identity certificate to the MyProxy repository with user ID and pass phrase. The user may specify the lifetime of credentials.
2. The user uploads attribute certificates to LDAP server.
3. Using a standard web browser, the user can connect to a Grid portal and allow the portal to contact MyProxy and LDAP server. And he must provide the same username and pass phrase to the portal.
4. MyProxy delegates proxy identity certificates to the portal on behalf of the user.

5. Grid portal will connect with the LDAP server, and send a request to retrieve the user's attribute certificates.
6. Grid portal will use these credentials to access grid resources on the user's behalf. Proxy identity certificate will be used to act on behalf of user for authentication and those attribute certificates will be used for authorization for various local resources in distributed areas. After authentication, the service provider will check user access rights information in the attribute certificates. If the user has rights to access service resources, the service provider will permit user to access them and establish service sessions. And before using attribute certificate to do the authorization, identity certificate that includes public key is used to authenticate the subject. And most important of all, the subject in the two certificates should be same.

Authorization information can be added into attribute certificates. Resource providers define their own access policies for various attributes. Distributed certificates can implement decentralized administration of resources. Attribute certificates are signed by authorities who are responsible for providing attribute information of users. An attribute comprises a type and a value. A role is just one type of attribute. Attribute certificates (ACs) are certified by an Attribute Authority (AA). If anyone trusts the AA, then he should trust the attributes specified in the AC which is issued by this AA. When a user sends request to access a resource, the system first authenticates the user with X.509 identity certificate, then makes access decisions with his attribute certificates. Therefore, our proposed architecture can make use of the online credential repository to build authentication, delegation and attribute based access control together to provide better security for grid system.

The proxy certificates stored in MyProxy repository in their current version cannot be limited except for validity period, so users have to delegate all rights to other entity. In our mechanism attribute certificates (AC) can be used to delegate specific access rights. The resource provider will issue an attribute certificate to that user. The AC certifies that a set of rights has been assigned to the specific user, and then the user can use this attribute certificate and his identity proxy certificate to access the resource.

In our proposed architecture, after a successful authentication for a user with MyProxy, the grid portal gets the user's role information from LDAP server. The LDAP server stores the user's attribute certificates that include his role assignment information. The grid portal contains resources that require particular roles to be accessed. Grid portal will allow or deny users to access the resources by using their ACs which are embedded with roles.

5. DEPLOYMENT

In this thesis, we address the second component of our security architecture, LDAP directory server, and provide details on how to deploy it. In section 5.1, we describe the installation, configuration and design for building our LDAP server, and also introduce how to design our schema and directory tree. Section 5.2 provides information on creating ACs and adding them onto LDAP server.

5.1 Building LDAP Server

5.1.1 Installation and Configuration

We use Sun One Directory Server 5.2 to build our own LDAP server because it is able to deploy extensible and secure global directory services. Moreover, it supports a wide variety of standards such as X.509 v3, LDAP URLs (RFC 2255), LDAP v3 (RFC 2251), LDIF (RFC 2849) and SSL. Sun One Directory Server 5.2 supports a large number of entries which can provide scalability for our system. Users can access the online services whenever they need them. In addition, we can use access management and security tools to manage users and the policies controlling their privileges. Most important, it supports digital certificates.

Sun One Directory Server 5.2 can be downloaded from Sun Website: http://www.sun.com/software/download/. We installed Sun One Server on the Windows 2003 Server, and our host system has a static IP address: 137.207.234.121. First, we obtain a naming service and the domain name for our host, and then we use the information in Table 3 for our installation.

Administration domain	galab.uwindsor.ca
Administration Server port number	3888
Directory Administrator ID	admin
Directory Administrator password	****** (6 characters)
Directory Manager DN	cn=Directory Manager
Directory Manager password	******** (8 characters)
Directory Server port number	389 (default LDAP)
Fully qualified host distinguished name	kent8.galab.uwindsor.ca
Server ID	kent8
Server suffix	dc=galab, dc=uwindsor, dc=ca
ServerRoot (software installation directory)	C:\Program Files\Sun\MPS

Table 3: Installation information for Directory Server

The configuration information for directory server is stored as LDAP entries in the directory itself. Therefore, changing the configuration information is required using the server rather than by changing configuration files. It allows a directory administrator to reconfigure the server when it is running, and need not shut it down. The Configuration Directory stores information about how Directory Server itself is configured. The `dse.ldif` file contains all the configuration information of directory server. The default configuration is stored as LDAP entries under the subtree `cn=config`. When the server is started, the contents of the `cn=config` subtree are read from a file `dse.ldif.` This directory is generally installed first, and every subsequent server registers with it. It is required to set access permissions for the configure files serverRoot\admin-serv\config\adm.conf. This can prevent unauthorized users from modifying

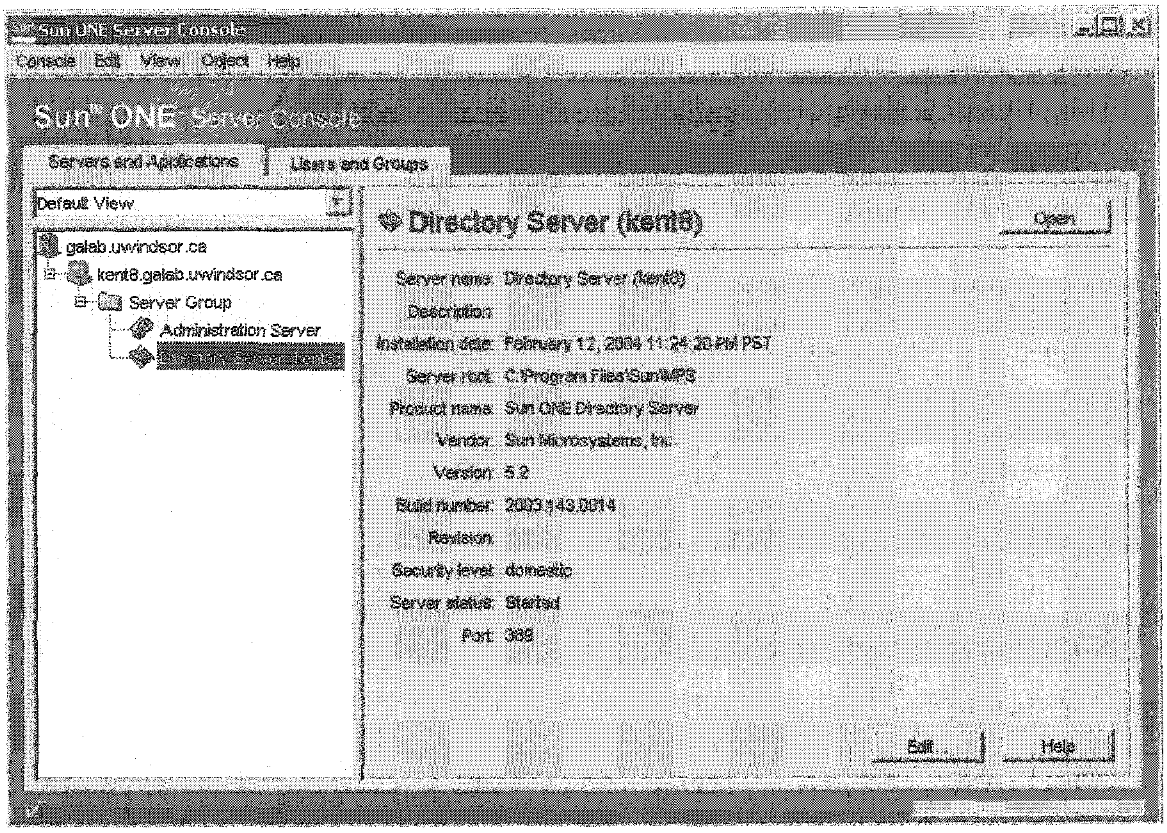
Administration Server configuration data.

Part of `dse.ldif` file for a Directory Server can be shown as follows:

```
dn: cn=config
objectclass: top
objectclass: extensibleObject
objectclass: nsslapdConfig
nsslapd-accesslog-logging-enabled: on
nsslapd-enquote-sup-oc: on
nsslapd-localhost: kent8.galab.uwindsor.ca
nsslapd-errorlog: ServerRoot/slapd-kent8/logs/errors
nsslapd-schemacheck: on
nsslapd-port: 389
```

The `cn=config` entry is an instance of the `nsslapdConfig` object class, which is inherits from `extensibleObject` object class. The value of attribute `nsslapd-schemacheck` is `on`, which makes schema checking turn on. The access logging is also turned on. nsslapd-enquote-sup-oc is used to enable superior object class en-quoting which controls whether quoting in the `objectclasses` attributes. It should conform to the quoting specified by internet draft RFC 2252.

Figure 5-1 shows the main interface of Sun One directory server 5.2 which is configured and started.



Directory Server (kent8)

Server name: Directory Server (kent8)
Description:
Installation date: February 12, 2004 11:24:20 PM PST
Server root: C:\Program Files\Sun\MPS
Product name: Sun ONE Directory Server
Vendor: Sun Microsystems, Inc.
Version: 5.2
Build number: 2003.143.0014
Revision:
Security level: domestic
Server status: Started
Port: 389

Figure 5-1: Sun One Directory Server 5.2 Interface

5.1.2 Schema Design

We design the contents of our directory server according to various types of data stored in it. The user directory stores entries for users who access directory services. The user directory is generally unique to the network domain. The directory schema determines the characteristics of the data stored in the directory.

While the object classes and attributes in the Directory Server do not meet our needs for ACs, a given object class does not permit us to store ACs information. So we extend the schema to support the object classes and attributes we need. We add "attributeCertificateAttribute" attribute and "pmiUser" object class into our LDAP schema in order to support storing and retrieve ACs.

Directory Server bases its schema format on version 3 of the LDAP protocol (LDAPv3). Schema includes two parts: attributes and object classes. When adding new schema elements, all attributes need to be defined before they can be used in an object class. Attributes and object classes can be defined in the same schema file. We can modify the schema with the command line tools. This is because the schema is stored as a regular entry. The schema for Directory Server is in the entry named cn=schema. But we have to authenticate as the Directory Server Manager, and then the schema content can be read or change by setting base to the DN of "cn=schema". `subschemaSubentry` is DN of the entry that contains schema information. This attribute is defined in RFC 2252 and present for every entry in the directory such as `subschemaSubentry: cn=schema`, syntax is DN; Object Identifier (OID) is 2.5.18.10; single-valued. `attributeTypes` specifies the attribute types used within a subschema. Each value describes a single attribute. Syntax is attribute types syntax; OID is 2.5.21.5; multi-valued.

The attributeCertificateAttribute is defined in [ITU01]. It is used to hold the attribute certificates of a user. The LDAP specific encoding for values of this attribute is described as follows:

```
attributeCertificateAttribute  ATTRIBUTE ::= {
WITH SYNTAX          AttributeCertificate
EQUALITY MATCHING RULE     attributeCertificateExactMatch
ID { joint-iso-ccitt(2) ds(5) attributeType(4)
attributeCertificate(58) } }
```

The corresponding LDAP description is

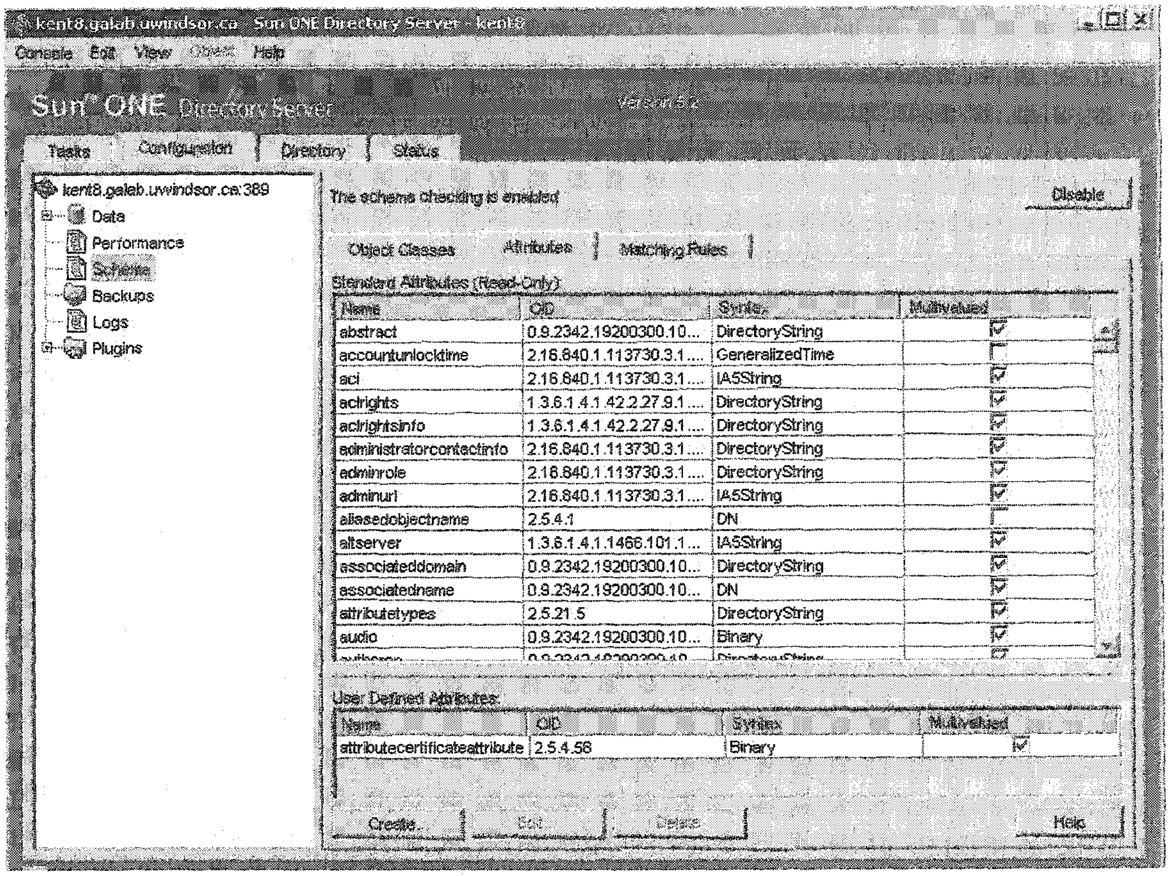
```
( 2.5.4.58 NAME 'attributeCertificateAttribute'
EQUALITY attributeCertificateExactMatch
SYNTAX 1.2.826.0.1.3344810.7.5 )
```

The LDAP uses Basic Encoding Rules (BER) as encoding method for an X.509 attribute certificate. The following string states the OID assigned to this syntax:

```
(1.2.826.0.1.3344810.7.5 DESC 'Attribute Certificate' )
```

We add a new attribute called "attributeCertificateAttribute" into our LDAP schema using the following information. Figure 5-2 shows this new attribute.

```
Attribute Name:     attributeCertificateAttribute
OID:                2.5.4.58
Syntax:             Binary
Multivalued Attribute: Yes
```



The schema checking is enabled

Standard Attributes (Read-Only)

Name	OID	Syntax	Multivalued
abstract	0.9.2342.19200300.10...	DirectoryString	☑
accountunlocktime	2.16.840.1.113730.3.1....	GeneralizedTime	□
aci	2.16.840.1.113730.3.1....	IA5String	☑
aclrights	1.3.6.1.4.1.42.2.27.9.1....	DirectoryString	☑
aclrightsinfo	1.3.6.1.4.1.42.2.27.9.1....	DirectoryString	☑
administratorcontactinfo	2.16.840.1.113730.3.1....	DirectoryString	☑
adminrole	2.16.840.1.113730.3.1....	DirectoryString	☑
adminurl	2.16.840.1.113730.3.1....	IA5String	☑
aliasedobjectname	2.5.4.1	DN	□
altserver	1.3.6.1.4.1.1466.101.1...	IA5String	☑
associateddomain	0.9.2342.19200300.10...	DirectoryString	☑
associatedname	0.9.2342.19200300.10...	DN	☑
attributetypes	2.5.21.5	DirectoryString	☑
audio	0.9.2342.19200300.10...	Binary	☑

User Defined Attributes:

Name	OID	Syntax	Multivalued
attributecertificateattribute	2.5.4.58	Binary	☑

Figure 5-2: Adding Attribute for AC into LDAP Schema

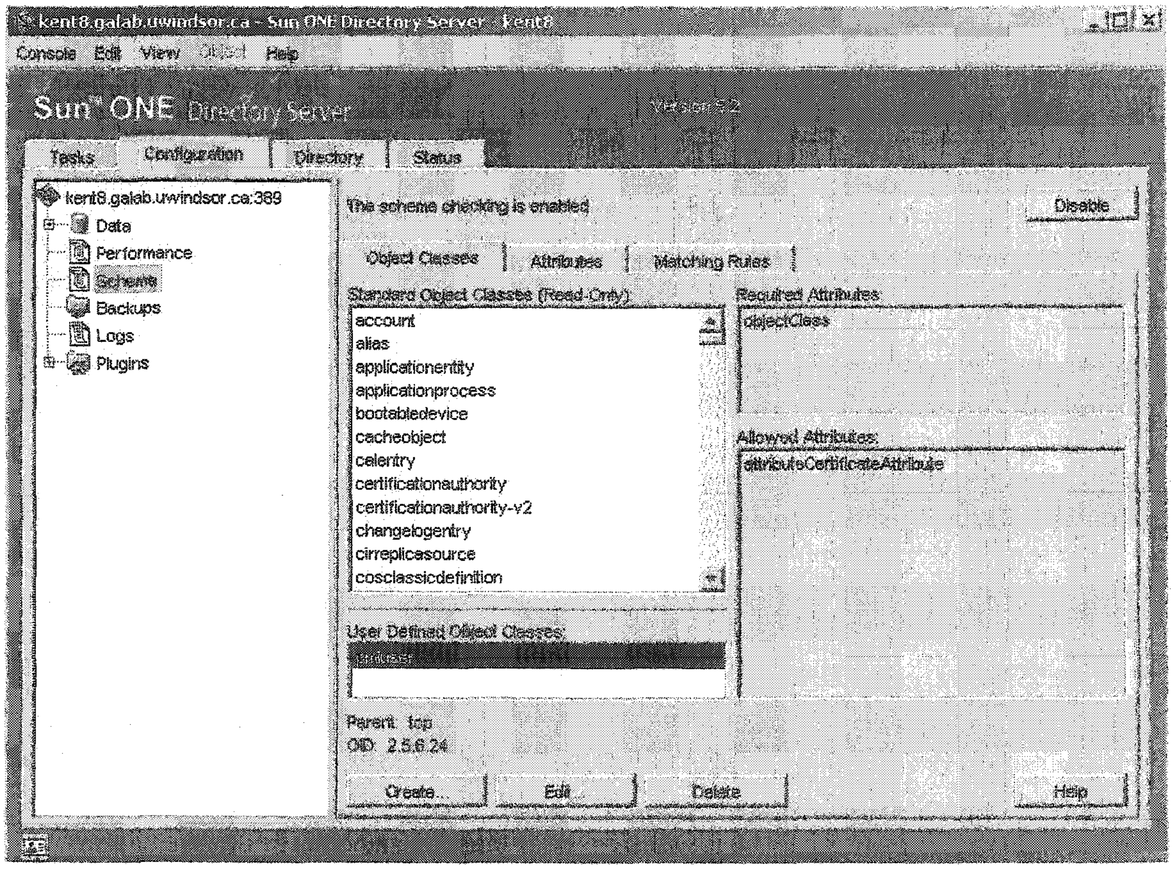
PMI Object Classes

PMI directory object classes [ITU01] is described as follows:

```
pmiUser OBJECT-CLASS ::= { -- a privilege holder
SUBCLASS OF   {top}
KIND          auxiliary
MAY CONTAIN   {attributeCertificateAttribute}
ID { joint-iso-ccitt(2) ds(5) objectClass(6) pmiUser (24) } }
```

We add a new object class called "pmiUser" into our LDAP schema using the following information. Figure 5-3 shows this new object class.

```
Object Class Name:  pmiUser
OID:                2.5.6.24
Superior Class:     top
Required Attributes
Optional Attributes:attributeCertificateAttribute
```



The schema checking is enabled

Standard Object Classes (Read-Only):
account
alias
applicationentity
applicationprocess
bootabledevice
cacheobject
calentry
certificationauthority
certificationauthority-v2
changelogentry
cirreplicasource
cosclassicdefinition

User Defined Object Classes:
pmiUser

Parent: top
OID: 2.5.6.24

Required Attributes:
objectClass

Allowed Attributes:
attributeCertificateAttribute

Figure 5-3: Adding PMI Object Class to LDAP Schema

5.1.3 Directory Tree Design

LDAP uses a hierarchical tree as a data structure. The directory tree is also known as a directory information tree (DIT). There is a single root node that is called root suffix. It has some subordinate nodes called subentries. A node is a directory entry with some attributes.

A LDAP entry is basic record in the LDAP directory. It includes attributes and values. Each entry has a special attribute called the distinguished name, which is the unique name of the entry. A Distinguished Name must be unique in the LDAP namespace. Its components must be part of the LDAP entry. A distinguished name (DN) is always indexed and will always be returned in any search. A DN is the string representation of the name of an entry in an LDAP directory, and it also describes a path to a directory entry. Each DN is made up of a number of components called relative distinguished names (RDNs). Each RDN identifies a specific entry in the directory. A single parent entry cannot have two identical RDNs below it. A DN always contain three types of RDN:

- A user name, user ID, or group name (identified by the `cn` or `uid` keyword)
- An organization name (identified by the `o` keyword)
- One or more domain name components (identified by the `dc` keyword)

The base of the tree can use the DomainClass (dc) as the root attributes. The dc is then based upon the domain name. Our domain name is "galab.uwindsor.ca", so our base dn could be "dc=galab, dc=uwindsor, dc=ca". The directory tree makes our directory data to be named and referred by client applications. The directory tree design involves choosing a suffix to contain our data, determining the hierarchical relationship amongst data entries, and naming the entries in the tree.

Creating a Suffix

The suffix is the name of the entry at the root of the tree. We store all our directory data under it. Our Directory Server deployment contains multiple suffixes, one for storing data and the others for internal directory operations such as configuration information and directory schema. Sub suffix is a branch under the root suffix.

We creating a new root suffix c=CA from Directory Server console. First, using Configuration tab and data field to choose "New suffix", then enter a suffix name c=CA in the Suffix DN field. The name must use the distinguished name format. The location of database files for this suffix is chosen automatically by the server. After the new root suffix was created, we logged in as the directory manager using cn=Directory Manager and the password of it. Then we select the New Root Object to create the new root suffix c=CA on the root node of the directory tree, and select an object class for the root object c=country. Finally we can use the generic editor to edit it.

Creating Directory Tree Structure

The design of data hierarchy can optimize the entry grouping and attribute management. Figure 5-4 shows our directory tree design. It includes four levels. First is country name, second is organization name, the third level can be organization unit or specific person, the fourth level is the end users who belong to various organization units.

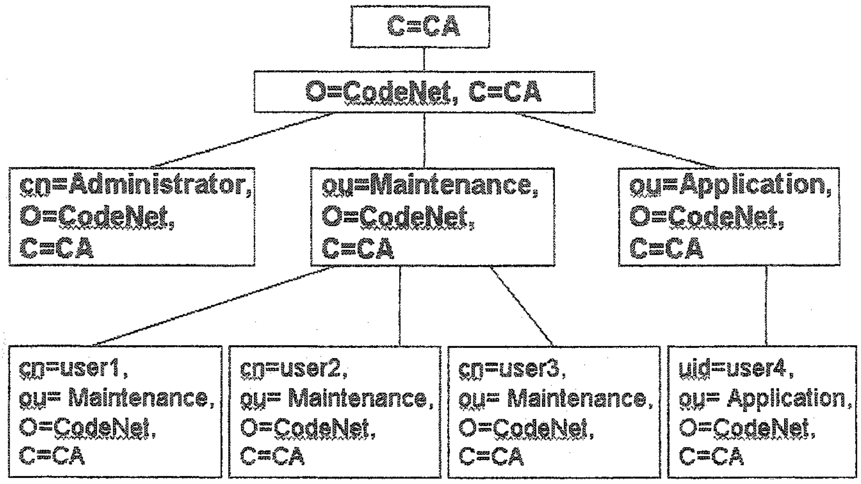

C=CA
O=CodeNet, C=CA
cn=Administrator, O=CodeNet, C=CA
ou=Maintenance, O=CodeNet, C=CA
ou=Application, O=CodeNet, C=CA
cn=user1, ou= Maintenance, O=CodeNet, C=CA
cn=user2, ou= Maintenance, O=CodeNet, C=CA
cn=user3, ou= Maintenance, O=CodeNet, C=CA
uid=user4, ou= Application, O=CodeNet, C=CA

Figure 5-4: Directory Information Tree (DIT) Design

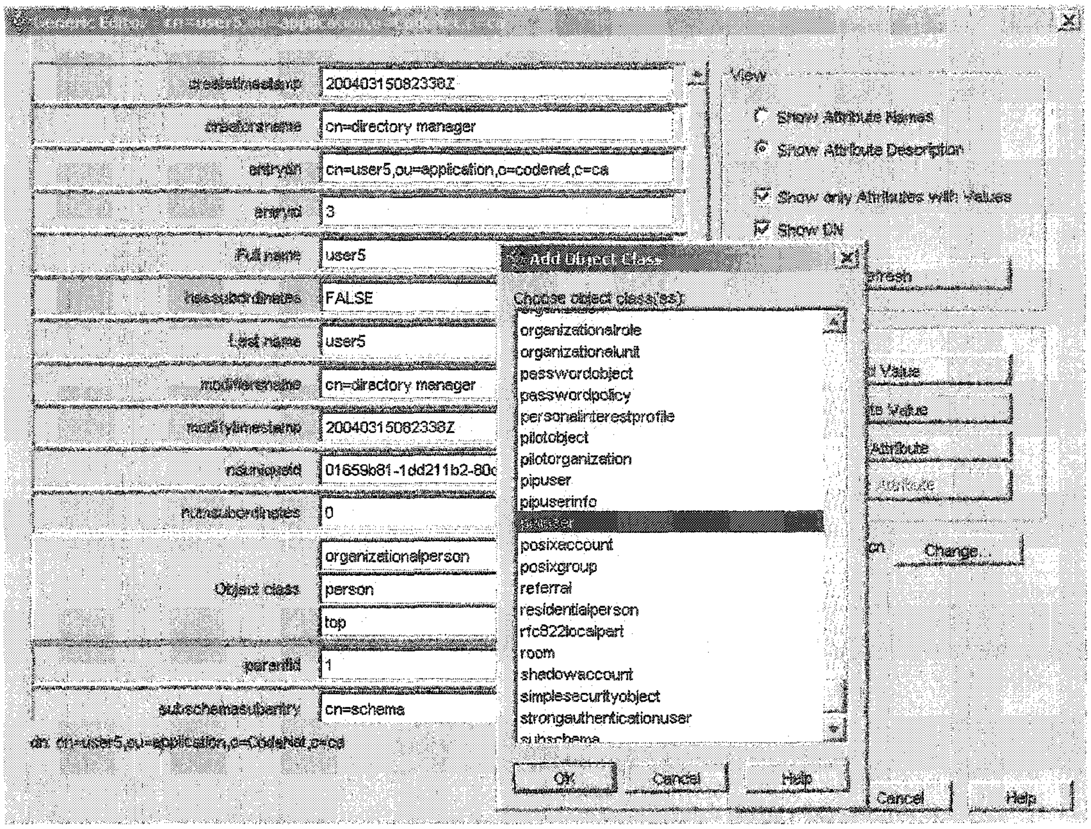
Based on the schema and the DIT design, we create entries as follows:

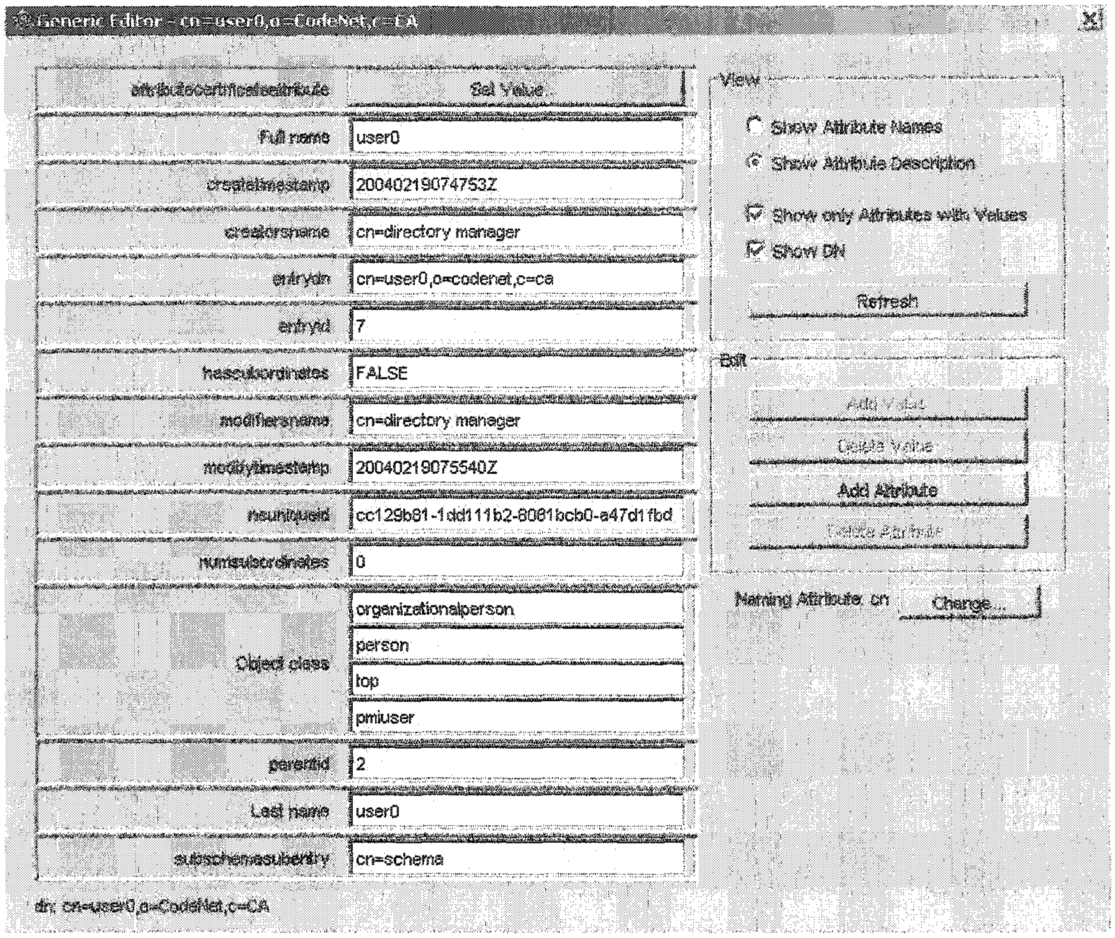
```
dn: o=CodeNet,c=CA
objectclass: top
objectclass: organization
o:CodeNet

dn: ou=Application,o=CodeNet,c=CA
objectclass: top
objectclass: organizationalUnit
ou: Application
     .
     .
     .
     .
dn:cn=user5,ou=Applicaiton,o=CodeNet,c=CA
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: pmiUser
```

Creating Entries with New Object Class and Attribute

We create an entry with new object class we have defined in the directory schema because they are not in the default list. First, use Directory tab and expand the directory tree to find the parent of the new entry, and then select the "New Other item". In the object class list of the New Object dialog, select the object class that defines our new entry. Figure 5-5 shows how to add the pmiUser object class in our entry `cn=user5, ou=Applicaiton, o=CodeNet, c=CA`. When creating a new entry, the generic editor contains a field for each required attribute of the object class we selected. And most important, we must add the new attributes that are involved in the new object class using add attribute dialog and generic editor. Figure 5-6 shows the new attribute has been added into the entry `cn=user0, o=CodeNet, c=CA`. Then the new entry is displayed as a child of that parent entry we have selected in the directory tree.

The generic editor allows us to see the attributes of an entry and edit them such as adding and removing attributes, and manage the object classes of the entry. The object classes of an entry are defined by the multi-valued `objectclass` attribute. We can use generic editor to manage our defined object classes. Figure 5-5 shows how to add an object class to an entry. First, select the `objectclass` attribute, and then using Add Value button to get a list of object classes that we can add to the entry. If this button is not activated, we do not have permission to modify this entry's object class. The object classes we selected appear in the list of values for the `objectclass` attribute.





Generic Editor - cn=user0,o=CodeNet,c=CA

attributecertificateattribute	Set Value...
Full name	user0
createtimestamp	20040219074753Z
creatorsname	cn=directory manager
entrydn	cn=user0,o=codenet,c=ca
entryid	7
hassubordinates	FALSE
modifiersname	cn=directory manager
modifytimestamp	20040219075540Z
nsuniqueid	cc129b81-1dd111b2-8081bcb0-a47d1fbd
numsubordinates	0
Object class	organizationalperson person top pmiuser
parentid	2
Last name	user0
subschemasubentry	cn=schema

dn: cn=user0,o=CodeNet,c=CA

View

○ Show Attribute Names

● Show Attribute Description

☑ Show only Attributes with Values

☑ Show DN

Naming Attribute: cn

Figure 5-6: Creating Entry with Adding AC attribute

We can also add one or more entries to the directory by using the "`ldapmodify`". The ldapmodify utility is used to create each entry as follows:

```
ldapmodify -a -h kent8.galab.uwindsor.ca -p 389
-D "cn=Directory Manager" -w password
dn: ou=Application,o=CodeNet,c=CA
objectclass: top
objectclass: organizationalUnit
ou: Application
description: Container for user entries


dn:cn=user5,ou=Applicaiton,o=CodeNet,c=CA
```

```
objectclass: top

objectclass: person
objectclass: organizationalPerson
objectclass: pmiUser
cn: user5
sn: user5
attributecertificateattribute:
telephoneNumber:
```

We use "cn=Directory Manager" to create these entries because this account has the permission to perform any actions on entries, and `-D` and `-w` options give the bind DN and password. The `-a` option makes all entries will be added. Then each entry is given by its DN and its attribute values, and there is a blank line between each entry. The LDAP Data Interchange Format (LDIF) of an entry lists the attributes in the following order:

- The list of object classes.
- The naming attribute used in the DN (optional).
- The list of required attributes for all object classes.
- The list of optional attributes for the entry.

5.1.4 Access Control for LDAP Server

Access control was one of the important issues of any server. There are several mechanisms to accomplish the authentication using a distinguished name (DN) in the directory. This distinguished name is used to determine the access for the user. Associating a distinguished name with a user is called binding. A user can bind to a server using the DN and a password, or they can bind anonymously.

We use the access control information (ACI) rules to perform access control for our LDAP server. It specifies the LDAP object, the users who are permitted access, and what permission is being allowed. The directory manager has the highest rights to access and manage the directory. If there is no access control information, the new directory can not be read by any user. The basic format of the ACI is as follows:

```
(target="ldap:///dn")(targetattr="attrname")
   [(targetfilter="rfc2254-style filter")]
   ( version 3.0; acl "name"; (allow | deny)
   (read, write, search, compare, selfwrite, add, delete )
   (userdn | groupdn)="ldap:///dn";)
```

The target specifies the entry where ACI rule will be effective. Usually, the target is the same as the entry's DN. `Targetattr` specifies one or more attributes that the access control information applies to. Access control rules can apply to specific attributes. This parameter provides more strict access to attributes such as user password. `Targetfilter` is an optional parameter that can be used to apply to specific entries based on a filter. The filter has the same syntax as a filter provided to the ldapsearch command. The permissions parameter consists of the version number, the ACI name, the operation including allow and deny, the permissions including read, write, search, compare, add, delete and selfwrite which allows a user to add or delete himself from a group.

During our data design, we have to specify who can write data to the directory. First, allow CodeNet administrator to do anything related creating and managing whole entries, and then allow read only access to the directory for everyone. We give the "Administrator" all access to our directory. It will be used to perform any operations in the directory. The ACI for the user cn=Administrator,o=CodeNet,c=ca is as follows:

```
(target="ldap:///o=CodeNet,c=CA")(targetattr=*)
(version 3.0; acl "codenet admin"; allow (all)
userdn="ldap:///cn=Administrator,o=CodeNet,c=CA";)
```

We allow all non-anonymous users to see pmiUser object class and can access our LDAP server to retrieve their attribute certificates. The ACI for all the users who has the account in LDAP server is as follows:

```
(target="ldap:///o=CodeNet,c=CA ")(targetattr=*)
(targetfilter="(objectclass=pmiUser)")(version 3.0;
  acl "AC users"; allow (compare, read, search)
```

```
        userdn="ldap:///all";)
```

The LDAP protocol supports anonymous access, which allows easy access to the directory. It allows any user to access our directory without authentication. We only allow anonymous access for read and search privileges of anonymous access as follows:

```
        (version 3.0; acl "anonymous-read";
       allow (read, search) userdn = "ldap:///anyone";)
```

5.1.5 Creating LDAP Server Certificate for SSL

In order to implement SSL between LDAP Server and its clients, the LDAP server must have a security certificate, and the client must be configured to trust this certificate. The server sends its certificate to the client to perform server authentication using public-key cryptography. The client and server then begin to encrypt all data transmitted over the network.

OpenSSL is a cryptography toolkit implementing the SSL network protocols. It also implements a wide range of cryptographic algorithms used in various Internet standards. OpenSSL toolkit includes three parts: the SSL library, the Crypto library and the command line tool. We use these tools to create self-signed certificate for our LDAP server.

We create our configure file "myopenssl.cnf ", the main content is shown in Figure 5-7.

```
# This is our own OpenSSL configuration file for creating CA
# and certificates.

HOME                    = .
RANDFILE                = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file               = $ENV::HOME/.oid
oid_section             = new_oids

####################################################################
[ ca ]
default_ca      = CodeNetCA
####################################################################
```

Figure 5-7: Our OpenSSL Configure File

```
[ CodeNetCA ]
dir             = ./openssl-0.9.7d  # Where everything is kept
certs           = $dir/certs        # Where the issued certs are kept
database        = $dir/index.txt    # database index file.
new_certs_dir= $dir/newcerts              # default place for new certs.
certificate     = $dir/cacert.pem   # The CA certificate
serial          = $dir/serial             # The current serial number
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand# private random number file
x509_extensions       = usr_cert          # The extentions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt        = ca_default        # Subject Name options
cert_opt        = ca_default        # Certificate field options
default_days    = 365               # how long to certify for
default_crl_days= 30                # how long before next CRL
default_md      = md5               # which md to use.

policy          = policy_match
# For the CA policy
[ policy_match ]
countryName             = optional
stateOrProvinceName     = optional
organizationName        = optional
organizationalUnitName        = optional
commonName              = supplied
emailAddress            = optional

####################################################################
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions         = v3_ca       # The extentions to add to the self signed
cert

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret
# req_extensions = v3_req # The extensions to add to a certificate request
[ req_distinguished_name ]
countryName                   = Country Name (2 letter code)
countryName_default           =
countryName_min                     = 2
countryName_max                     = 2
stateOrProvinceName           = State or Province Name (full name)
stateOrProvinceName_default         =
localityName                  = Locality Name (eg, city)
0.organizationName            = Organization Name (eg, company)
0.organizationName_default = Internet Widgits Pty Ltd
organizationalUnitName              = Organizational Unit Name (eg, section)
commonName                    = Common Name (eg, YOUR name)
commonName_max                      = 64
emailAddress                  = Email Address
emailAddress_max              = 64
```

Figure 5-7: Our OpenSSL Configure File (cont.)

```
[ req_attributes ]
challengePassword          = A challenge password
challengePassword_min            = 4
challengePassword_max            = 20
unstructuredName           = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.
# requires this to avoid interpreting an end user certificate as a CA.
basicConstraints=CA:FALSE

[ v3_req ]

# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]

# Extensions for a typical CA
# PKIX recommendation.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true
```

Figure 5-7: Our OpenSSL Configure File (cont.)

Then we create certificates with OpenSSL on Linux using our configure file "myopenssl.cnf". `x509` is a certificate display and signing utility. `-inform DER|PEM` specifies the input format. Normally the command will expect an X509 certificate but this can change if other options such as `-req` are present. `-req` indicates a certificate request. The DER format is the DER encoding of the certificate and PEM is the base64 encoding of the DER encoding with header and footer lines added. `-in` specifies the input filename to read a certificate from. `-out` specifies the output filename to write to. `-text` prints out the certificate in text form including the public key, signature algorithms, issuer and subject names, serial number any extensions present and any trust settings. `-signkey` makes the input file to be self signed using the supplied private key. If the input file is a certificate it sets the issuer name to the subject name, which makes it self signed. If the input is a certificate request then a self signed certificate is created using the supplied private key using the subject name in the request. `-CA` specifies the CA certificate to be used for signing. The input file is signed by this CA, that is its issuer name is set to the subject name of the CA and it is digitally signed using the CAs private key. `-CAkey` sets the CA private key to sign a certificate with. `-extfile` specifies the file containing certificate extensions to use. `-extensions` adds certificate extensions from a specified file. The `basicConstraints` extension CA flag is used to determine whether the certificate can be used as a CA. If the CA flag is true then it is a CA, if the CA flag is false then it is not a CA. All CAs should have the CA flag set to true.

Figure 5-8 shows how we can create self-signed certificate as a server certificate. We first create a 1024 bit RSA key pair, public key will be included in the certificate request, and private key is stored in our machine. Then we used the self sign function of OpenSSL to sign the request using the private key we have generated.

```
chen12p@0[openssl-0.9.7d]$ openssl req -newkey rsa:1024 -config myopenssl.cnf -
keyout rootkey.pem -out rootreq.pem

Generating a 1024 bit RSA private key
.....................++++++
.......++++++
writing new private key to 'rootkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:CA
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CodeNet
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:rootadmin
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

chen12p@0[openssl-0.9.7d]$ openssl x509 -req -in rootreq.pem -extfile
myopenssl.cnf -extensions v3_ca -signkey rootkey.pem -out rootcert.pem

Signature ok
subject=/C=CA/O=CodeNet/CN=rootadmin
Getting Private key
Enter pass phrase for rootkey.pem:
```

Figure 5-8: Creating Self-Signed Certificate

```
chen12p@0[openssl-0.9.7d]$ cat rootcert.pem rootkey.pem > root.pem
chen12p@0[openssl-0.9.7d]$ openssl x509 -in rootcert.pem -text -noout

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=CA, O=CodeNet, CN=rootadmin
        Validity
            Not Before: Mar 24 05:13:39 2004 GMT
            Not After : Apr 23 05:13:39 2004 GMT
        Subject: C=CA, O=CodeNet, CN=rootadmin
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b0:f3:95:c1:ba:68:86:95:de:2b:2b:40:ed:7f:
                    10:2a:27:7f:17:50:fe:fb:34:d8:00:a5:29:57:b4:
                    8e:4a:48:f1:43:29:d2:5e:1c:0d:3b:19:2a:b8:30:
                    be:e6:7e:b5:72:c6:ca:bb:13:5c:da:f0:9e:fd:20:
                    c6:cb:65:60:9e:fe:15:16:3d:a2:6f:a9:10:18:ba:
                    20:99:3b:22:2a:bf:c9:00:5a:6d:c8:9d:d4:bf:8e:
                    f6:9a:d2:53:1f:b5:07:d6:e0:b4:39:81:ba:30:e3:
                    47:df:75:e5:16:15:2e:00:50:66:63:df:7c:ea:06:
                    0c:28:7f:7b:6b:e8:bc:40:57
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                21:A2:2A:2C:7A:1A:28:03:79:00:09:14:EE:86:1A:2B:DB:E7:21:A6
            X509v3 Authority Key Identifier:

keyid:21:A2:2A:2C:7A:1A:28:03:79:00:09:14:EE:86:1A:2B:DB:E7:21:A6
                DirName:/C=CA/O=CodeNet/CN=rootadmin
                serial:00

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
        12:1f:23:e8:99:e4:b1:89:d5:56:f3:e1:bb:5a:ae:b1:a9:a9:
        3e:2b:79:c6:e7:c6:88:1e:17:a8:ed:13:2a:4a:0c:f1:5c:06:
        94:ba:c7:56:66:a6:7d:d7:f8:0f:84:ee:a4:b5:39:2a:16:e7:
        ae:6f:82:20:54:72:79:1c:25:46:d8:e8:c8:d3:ed:0f:ec:ce:
        83:85:aa:cb:c1:12:8c:e9:c8:90:64:6f:99:a5:14:59:06:98:
        08:82:a0:88:82:b6:c3:e6:17:f2:d5:c3:ea:8c:ad:78:5e:22:
        dd:ca:fe:da:d0:5e:78:cc:42:fc:bf:59:44:a8:5f:8f:81:a8:
        51:2c
```

Figure 5-8: Creating Self-Signed Certificate (cont.)

A secure connection encrypts all data during the communication over the network. Clients may establish secure connections through the secure port using the Secure Socket Layer (SSL).

5.2 Adding ACs into LDAP Server

5.2.1 Obtaining OID

Each LDAP object class or attribute must be assigned a unique name and object identifier (OID). One OID is enough because we can add another level of hierarchy to create new branches for our attributes and object classes. Obtain an OID from the Internet Assigned Numbers Authority (IANA) or a national organization. ANSI also provide this service, but it require to pay, so we choose IANA and apply it through their website http://www.iana.org/cgi-bin/enterprise.pl. And our OID is prefixed with 1.3.6.1.4.1 (the private Internet OID) after we obtain the number from IANA.

For the management of hosts and gateways on the Internet a data structure for the information has been defined. The data structure is the "Structure and Identification of Management Information for TCP/IP-based Internets" (SMI) [RFC1155], and the "Management Information Base for Network Management of TCP/IP-based Internets" (MIB-II) [RFC1213]. The SMI provides parameters or codes to indicate private data structures. The assignment for our OID is listed in Figure 5-9.

```
Prefix: iso.org.dod.internet.private.enterprise (1.3.6.1.4.1)
See the file "enterprise-numbers".
http://www.iana.org/assignments/enterprise-numbers
SMI Network Management Private Enterprise Codes:
19569
  Grid Research Lab
    Hao Chen
      chen12p@uwindsor.ca
our oid: 1.3.6.1.4.1.19569.
```

Figure 5-9: Obtain Private Internet OID from IANA

So our internationally registered Object Identifier (OID) is 1.3.6.1.4.1.19569. It is possible to create a hierarchical structure of subordinate identifiers using this OID. This approach allows us to obtain more OID for various attributes or information.

5.2.2 Creating ACs

ACs are used to perform authorization for users. They associate attributes with users. The certificate includes several fields that are similar to identity certificate. We create an AC which includes version, serial number, user's DN, roles and time validity.

Figure 5-10 shows the object class of Attribute Certificates[Chadwick03].

```
(1.2.826.0.1.3344810.1.0.16
NAME 'x509AC'
SUP x509base
STRUCTURAL
MUST ( x509version $
       x509serialNumber $
       x509validityNotBefore $
       x509validityNotAfter )
MAY ( x509acHolderPKCSerialNumber $
       x509acHolderPKCissuerDN
       x509acHolderRfc822Name $
       x509acHolderDnsName $
       x509acHolderDN $
       x509acHolderURI $
       x509acHolderIpAddress $
       x509acHolderRegisteredID $
       x509IssuerRfc822Name $
       x509IssuerDnsName $
       x509IssuerURI $
       x509IssuerIpAddress $
       x509IssuerRegisteredID $
       x509authorityCertIssuer $
       x509authorityCertSerialNumber ))
```

Figure 5-10: X509 Attribute Certificate Object Class

Figure 5-11 shows attribute types defined in [Klasen02] which are often used in corresponding fields of ACs.

```
x509serialNumber:         used to hold the serial number of the AC
x509version:              used to hold the version of the AC
x509signatureAlgorithm:   used to hold the OID of the algorithm used to sign
the CRL
x509issuer:               used to hold the DN of the AC issuer
x509validityNotBefore:    used to hold the not before validity time of the AC
x509validityNotAfter:     used to hold the not after validity time of the AC
x509authorityCertIssuer:  used in conjunction with
x509authorityCertSerialNumber to identify the PKC of the AC issuer
x509authorityCertSerialNumber: used in conjunction with x509authorityCertIssuer
to identify the PKC of the AC issuer
x509issuerRfc822Name:     used to hold the email address of the AC issuer
x509issuerDnsName:        used to hold the DNS name of the AC issuer
x509issuerURI:            used to hold a URI for the AC issuer
x509issuerIpAddress:      used to hold the IP address of the AC issuer
x509issuerRegisteredID:   used to hold a registered OID of the AC issuer
x509authorityKeyIdentifier: used to hold the identifier of the public key used
to sign the AC
```

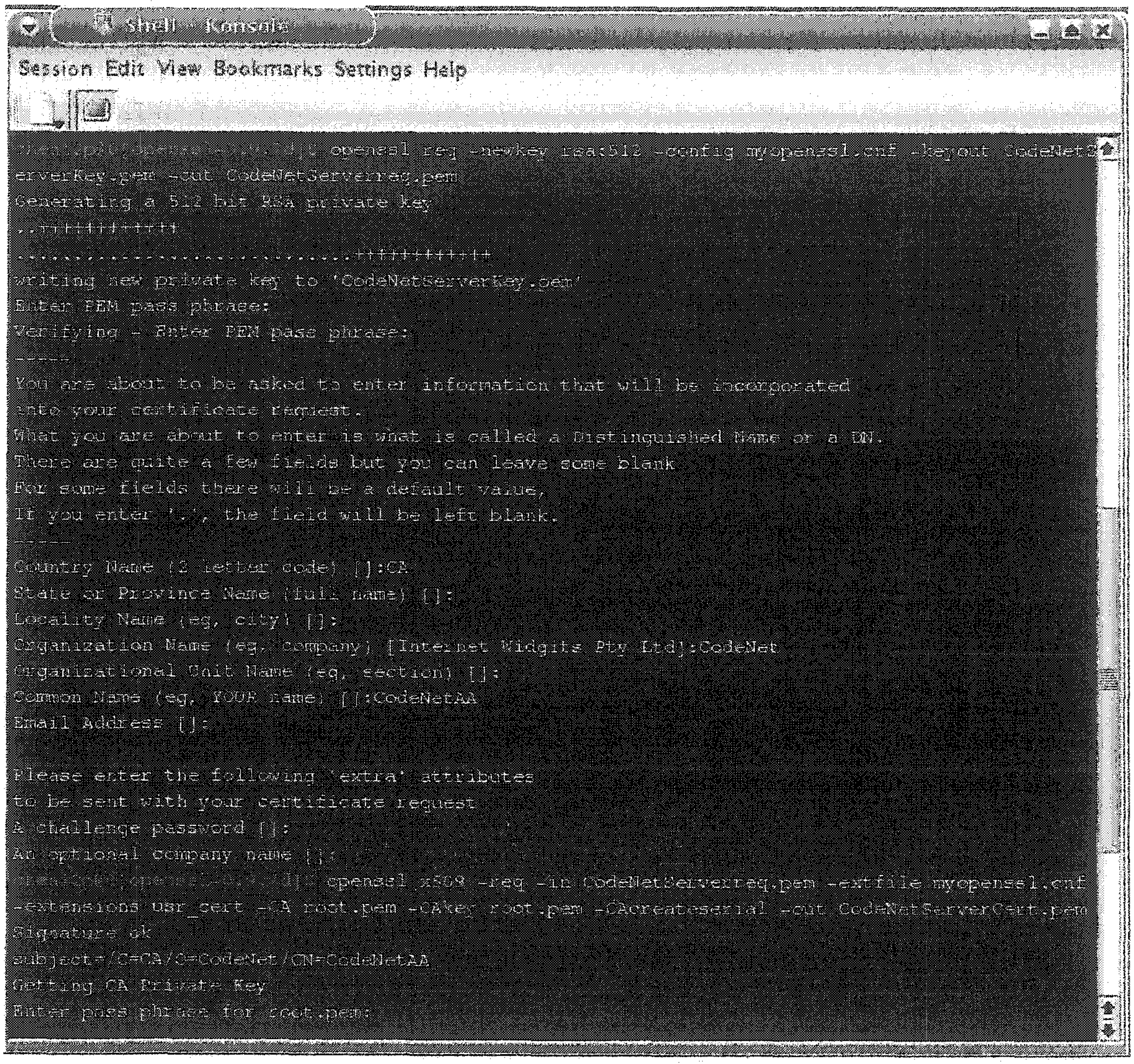
Figure 5-11: Attribute Certificate Attribute Types

The user's privilege is related with roles because each role will be assigned a certain set of access privileges. We provide a role design as in Table 4 for each user who belongs to one of the four groups. And each group has different permission to access the resource. The role information will be added into the attribute field of AC of each user. When performing the authorization later, the information of role will be extracted from the AC of the user.

User	Role	Permission	Resource
user8	General User	Read, search	Limited DB1
user0, user4, user5	Registered User	Read, search, execute	limited DB1, limited DB2
user1, user2, user3	System User	Read, search, add, execute	Full DB1, limited DB2
administrator	Administrator	Read, search, add, delete, execute	Full DB1, Full DB2

Table 4: Roles and Permissions Design

Each AA should have its own Public Key Certificate which is used to sign the ACs. So we use OpenSSL to create AA's certificate. Its subject name or distinguish name is "CN=CodeNetAA, O=CodeNet, C=CA". We use CA's private key which is included in file "root.pem" to sign AA's certificate. Figure 5-12 shows the detailed information on how to generate AA's certificate.



```
chen12p@0[openssl-0.9.7d]$ openssl req -newkey rsa:512 -config myopenssl.cnf -keyout CodeNetS
erverKey.pem -out CodeNetServerreq.pem
Generating a 512 bit RSA private key
..++++++++++++
.................................++++++++++++
writing new private key to 'CodeNetServerKey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:CA
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CodeNet
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:CodeNetAA
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
chen12p@0[openssl-0.9.7d]$ openssl x509 -req -in CodeNetServerreq.pem -extfile myopenssl.cnf
-extensions usr_cert -CA root.pem -CAkey root.pem -CAcreateserial -out CodeNetServerCert.pem
Signature ok
subject=/C=CA/O=CodeNet/CN=CodeNetAA
Getting CA Private Key
Enter pass phrase for root.pem:
```

Figure 5-12: Creating Certificate for Attribute Authority (AA)

We use RSA (Rivest Shamir Adelman) algorithm for digital signature. In order to sign, first compute the message digest, then encrypt the message using AA's private key. After we obtain the AA's certificate which is called "CodeNetServerCert", then we will put it with its private key "CodeNetServerKey" and the CA's certificate "rootcert" which issues AA's certificate together.

```
chen12p@0[openssl-0.9.7d]$ cat CodeNetServerCert.pem CodeNetServerKey.pem
rootcert.pem > CodeNetServer.pem
```

```
chen12p@0[openssl-0.9.7d]$ openssl pkcs12 -export -in CodeNetServer.pem -out
CodeNetServer.p12
Enter pass phrase for CodeNetServer.pem:
Enter Export Password:
Verifying - Enter Export Password:
```

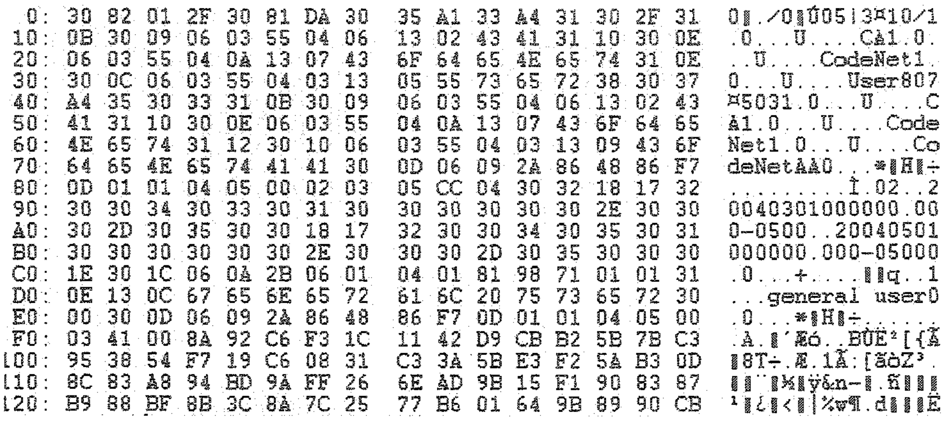
Then we use the following basic information for creating AC including role, and we also use our own OID to represent the role attribute.

```
AC version:  v1 or v2
Holder Name: cn=user8,o=CodeNet,c=ca
Issuer Name: cn=CodeNetAA,o=CodeNet,c=ca
Serial No.:  379908
Validity Period: from 2004/03/01 to 2004/05/01
CodeNetRole: 1.3.6.1.4.1.19569.1.1 (general user)
```

Then we use attribute authority AA's private key which is in the file "CodeNetServer.p12" to sign the AC. After using AA's certificate to sign the AC, it is in binary form as follows:

```
0,´/0 Ú05¡3¤10/10
       ˙˜U¯˙ž^CA1Ł0˙˜U¯ž¨CodeNet10˙˜U¯˜ž˘User807¤50310˙˜U¯˙ž^CA1Ł0˙˜U¯ž¨CodeNet1
Ž0Ł˙˜U¯˜žCodeNetAA0˙*†H†÷´´¯˘ ^˜˘Ì¯02  20040301000000.000-
0500  20040501000000.000-05000-0 ˙+˙´¯´ ˜q´´1žgeneral user0 0˙     *†H†÷
´´¯˘ ˜A Š'Æó łBÙË²[{Ã•8T÷ Æ˚1Ã:[ãòZ³©ƒ¨"½šÿ&n-> ñ ƒ‡¹^¿<<Š|%w¶´d>‰ Ëjº:
```

We use Textpad editor to view the AC we created as in Figure 5-13, in binary form we can still see some information in the certificate such as the pieces of the distinguished names, validity period, and even roles. However, A digital signature is an encrypted digest of the content. If the role or other information is modified by any attacker, the modification will break the signature because the digest of the certificate will not be the same as original.



```
  0: 30 82 01 2F 30 81 DA 30  35 A1 33 A4 31 30 2F 31  0▮./0▮Ú05|3¤10/1
 10: 0B 30 09 06 03 55 04 06  13 02 43 41 31 10 30 0E  .0...U....CA1.0.
 20: 06 03 55 04 0A 13 07 43  6F 64 65 4E 65 74 31 0E  ..U....CodeNet1.
 30: 30 0C 06 03 55 04 03 13  05 55 73 65 72 38 30 37  0...U....User807
 40: A4 35 30 33 31 0B 30 09  06 03 55 04 06 13 02 43  ¤5031.0...U....C
 50: 41 31 10 30 0E 06 03 55  04 0A 13 07 43 6F 64 65  A1.0...U....Code
 60: 4E 65 74 31 12 30 10 06  03 55 04 03 13 09 43 6F  Net1.0...U....Co
 70: 64 65 4E 65 74 41 41 30  0D 06 09 2A 86 48 86 F7  deNetAA0...*▮H▮÷
 80: 0D 01 01 04 05 00 02 03  05 CC 04 30 32 18 17 32  .........Ì.02..2
 90: 30 30 34 30 33 30 31 30  30 30 30 30 30 2E 30 30  0040301000000.00
 A0: 30 2D 30 35 30 30 18 17  32 30 30 34 30 35 30 31  0-0500..20040501
 B0: 30 30 30 30 30 30 2E 30  30 30 2D 30 35 30 30 30  000000.000-05000
 C0: 1E 30 1C 06 0A 2B 06 01  04 01 81 98 71 01 01 31  .0...+....▮▮q..1
 D0: 0E 13 0C 67 65 6E 65 72  61 6C 20 75 73 65 72 30  ...general user0
 E0: 00 30 0D 06 09 2A 86 48  86 F7 0D 01 01 04 05 00  .0...*▮H▮÷......
 F0: 03 41 00 8A 92 C6 F3 1C  11 42 D9 CB B2 5B 7B C3  .A.▮'Æó..BÙË²[{Ã
100: 95 38 54 F7 19 C6 08 31  C3 3A 5B E3 F2 5A B3 0D  ▮8T÷.Æ.1Ã:[ãòZ³.
110: 8C 83 A8 94 BD 9A FF 26  6E AD 9B 15 F1 90 83 87  ▮▮¨▮½▮ÿ&n-▮.ñ▮▮▮
120: B9 88 BF 8B 3C 8A 7C 25  77 B6 01 64 9B 89 90 CB  ¹▮¿▮<▮|%w¶.d▮▮▮Ë
```

Figure 5-13: Viewing Attribute Certificate (AC)

5.2.3 Storing AC on LDAP Server

We have extended the schema to allow ACs to be stored in our LDAP directory server. However, we can only add ACs to existing entries. So before adding an AC to LDAP server, we first create an entry for the AC holder.

After we create the new attribute for ACs in LDAP server, we will set value to them. Attribute for ACs are in binary form. Binary attribute values are marked with the `attribute;binary` subtype which indicates the content of an attribute. Subtypes may be added to attribute names in the LDIF statements used with the `ldapmodify` command as follows:

```
ldapmodify -h kent8.galab.uwindsor.ca -p 389 -D
"cn=Directory Manager" -w password
version: 1
dn: cn=user2,ou=Maintenance,o=CodeNet,c=CA
changetype: modify
add: attributeCertificateAttribute;binary
attributeCertificateAttribute;binary: <
file:///chen12p/user2.ace
```

To add a binary value, we can write it in the LDIF text or read it from the file. In order to use the < syntax to specify a filename, we must begin the LDIF statement with the line `version:1`, and the space should be added before and after `<`. When `ldapmodify` processes this statement, it will set the attribute to the value which is read from the given file after < . When modifying an entry, the attributes must be allowed by the object classes in the entry, and attributes must contain values that match our defined syntax. Directory Server performs schema checking to conform the object class or attribute to our schema.

LDAP Data Interchange Format (LDIF) is used for exchange directory information. LDIF is an ASCII format. It can be used to make changes to the LDAP server. Binary data can be referenced in an external file or included in-line BASE-64 encoded. LDIF files are simply ASCII files that describe a set of changes to be applied to a directory. We can use an LDIF file to add multiple entries, to perform a mix of operations or to import an entire suffix.

LDIF is a textual representation of entries, attributes and their values. Its standard format is described in RFC 2849. When using LDIF, there are some important aspects such as command-line input, special characters, schema checking, and the order of entries. There is a strict order for the entries listed in the LDIF file. Parent entries must be listed before their children. When the server processes the LDIF text, it will create the parent entries before the children entries. We list an `ou=Application` entry container before the entries within the subtree as follows:

```
dn: o=CodeNet, c=CA
dn: ou=Application,o=CodeNet, c=CA
...
Application users subtree entries
...
dn: ou=Maintenance,dc=example,dc=com
...
Maintenance users subtree entries
...
```

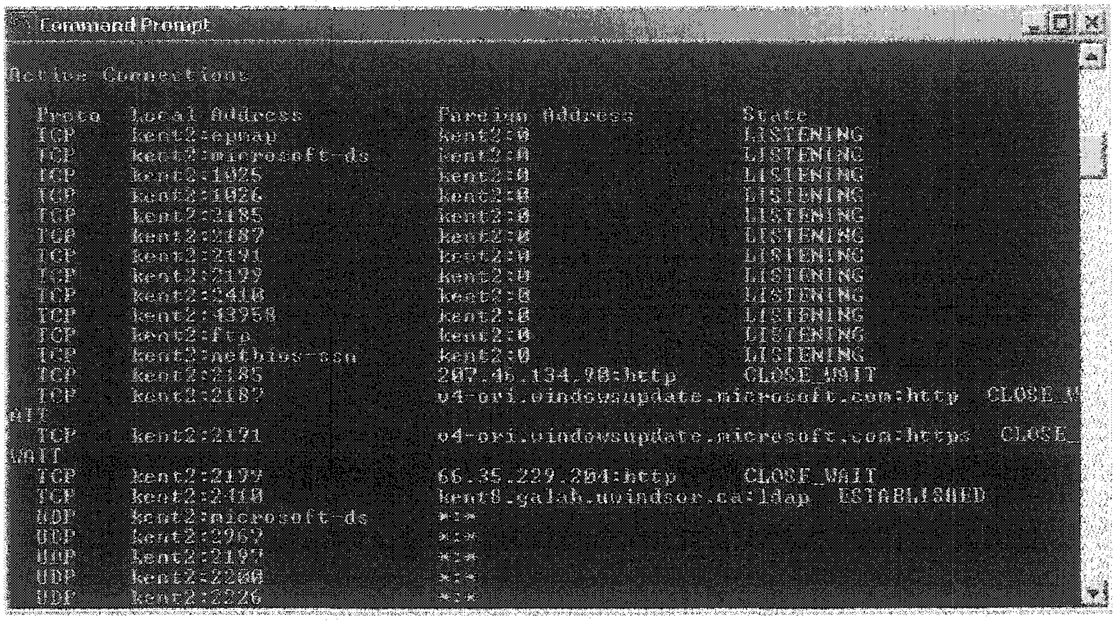
6. TESTING AND EVALUATION

This chapter covers the testing issues using various tools and the evaluations on the whole security architecture. Section 6.1 describes the testing about connections between client and server. Section 6.2 describes how to use various Web browsers to test LDAP URL. Section 6.3 provides details on using LBE to test our LDAP server and operations on attribute certificates. Section 6.4 introduces the features of the architecture. Section 6.5 describes the analysis of search algorithm and performance. Section 6.6 lists the limitation of the design and implementation of the enhanced security architecture.

6.1 Testing Client and Server Connection

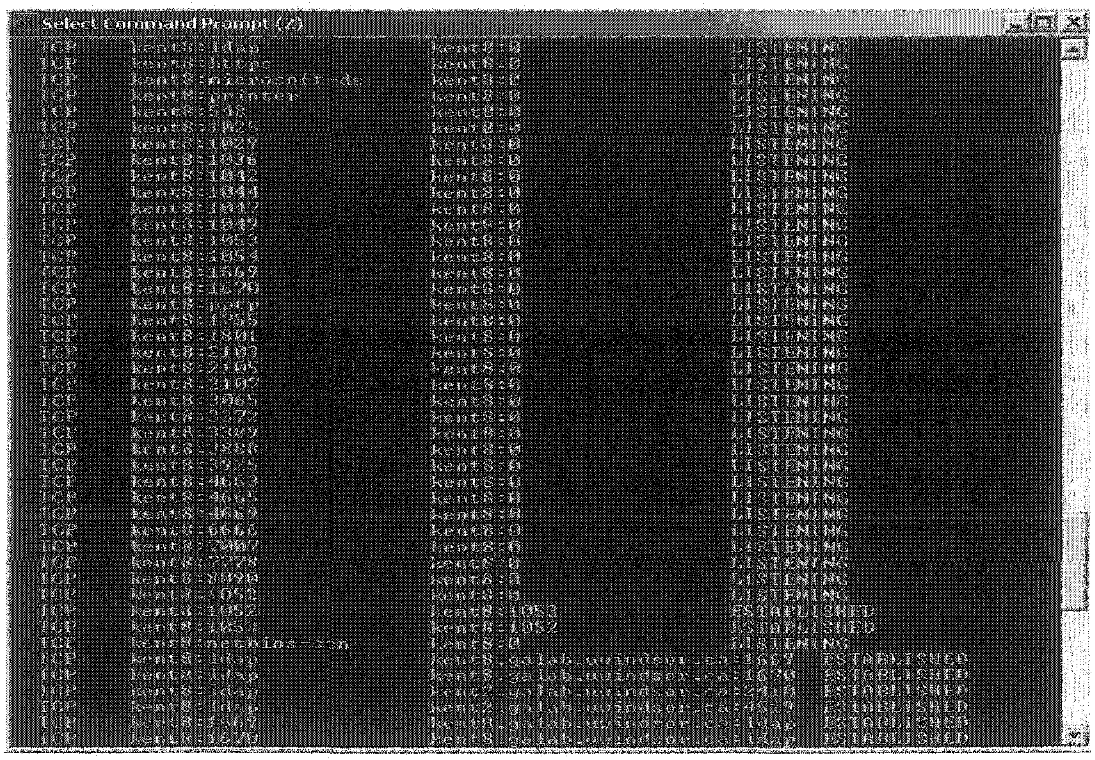
We use the "netstat" tool to examine if the connection between LDAP client and server has established. Figure 6-1 describes the connections on client machine whose host name is "kent2.galab.uwindsor.ca" (137.207.234.171), while Figure 6-2 shows the connections on our LDAP server whose host name is "kent8.galab.uwindsor.ca" (137.207.234.121). From Figure 6-2, we can see LDAP server is always listening on any LDAP request when the server is running. If the connection between LDAP client and server has been established, the state of connections will be "ESTABLISHED" which is shown in the following figures.

From the client in Figure 6-1, local address is "kent2: 2410", foreign address is "kent8.galab.uwindsor.ca: ldap", the state is "ESTABLISHED", which means the client "kent2" has connected with the LDAP server "kent8". From the server in Figure 6-2, local address is "kent8: ldap", foreign address is "kent2.galab.uwindor.ca: 2410", the state is "ESTABLISHED" too. Therefore, client and LDAP server have established the connection with each other.



```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    kent2:epmap            kent2:0                LISTENING
  TCP    kent2:microsoft-ds     kent2:0                LISTENING
  TCP    kent2:1025             kent2:0                LISTENING
  TCP    kent2:1026             kent2:0                LISTENING
  TCP    kent2:2185             kent2:0                LISTENING
  TCP    kent2:2187             kent2:0                LISTENING
  TCP    kent2:2191             kent2:0                LISTENING
  TCP    kent2:2199             kent2:0                LISTENING
  TCP    kent2:2410             kent2:0                LISTENING
  TCP    kent2:43958            kent2:0                LISTENING
  TCP    kent2:ftp              kent2:0                LISTENING
  TCP    kent2:netbios-ssn      kent2:0                LISTENING
  TCP    kent2:2185             207.46.134.98:http     CLOSE_WAIT
  TCP    kent2:2187             v4-ori.windowsupdate.microsoft.com:http  CLOSE_W
AIT
  TCP    kent2:2191             v4-ori.windowsupdate.microsoft.com:https  CLOSE_
WAIT
  TCP    kent2:2199             66.35.229.204:http     CLOSE_WAIT
  TCP    kent2:2410             kent8.galab.uwindsor.ca:ldap  ESTABLISHED
  UDP    kent2:microsoft-ds     *:*
  UDP    kent2:2969             *:*
  UDP    kent2:2197             *:*
  UDP    kent2:2200             *:*
  UDP    kent2:2226             *:*
```

Figure 6-1: LDAP Client Connections State



```
  TCP    kent8:ldap             kent8:0                LISTENING
  TCP    kent8:https            kent8:0                LISTENING
  TCP    kent8:microsoft-ds     kent8:0                LISTENING
  TCP    kent8:printer          kent8:0                LISTENING
  TCP    kent8:548              kent8:0                LISTENING
  TCP    kent8:1025             kent8:0                LISTENING
  TCP    kent8:1029             kent8:0                LISTENING
  TCP    kent8:1036             kent8:0                LISTENING
  TCP    kent8:1042             kent8:0                LISTENING
  TCP    kent8:1044             kent8:0                LISTENING
  TCP    kent8:1047             kent8:0                LISTENING
  TCP    kent8:1049             kent8:0                LISTENING
  TCP    kent8:1053             kent8:0                LISTENING
  TCP    kent8:1054             kent8:0                LISTENING
  TCP    kent8:1669             kent8:0                LISTENING
  TCP    kent8:1670             kent8:0                LISTENING
  TCP    kent8:pptp             kent8:0                LISTENING
  TCP    kent8:1755             kent8:0                LISTENING
  TCP    kent8:1801             kent8:0                LISTENING
  TCP    kent8:2103             kent8:0                LISTENING
  TCP    kent8:2105             kent8:0                LISTENING
  TCP    kent8:2107             kent8:0                LISTENING
  TCP    kent8:2065             kent8:0                LISTENING
  TCP    kent8:3372             kent8:0                LISTENING
  TCP    kent8:3389             kent8:0                LISTENING
  TCP    kent8:3888             kent8:0                LISTENING
  TCP    kent8:3925             kent8:0                LISTENING
  TCP    kent8:4663             kent8:0                LISTENING
  TCP    kent8:4665             kent8:0                LISTENING
  TCP    kent8:4669             kent8:0                LISTENING
  TCP    kent8:6666             kent8:0                LISTENING
  TCP    kent8:7007             kent8:0                LISTENING
  TCP    kent8:7778             kent8:0                LISTENING
  TCP    kent8:8090             kent8:0                LISTENING
  TCP    kent8:1052             kent8:0                LISTENING
  TCP    kent8:1052             kent8:1053             ESTABLISHED
  TCP    kent8:1053             kent8:1052             ESTABLISHED
  TCP    kent8:netbios-ssn      kent8:0                LISTENING
  TCP    kent8:ldap             kent8.galab.uwindsor.ca:1669  ESTABLISHED
  TCP    kent8:ldap             kent8.galab.uwindsor.ca:1670  ESTABLISHED
  TCP    kent8:ldap             kent2.galab.uwindsor.ca:2410  ESTABLISHED
  TCP    kent8:ldap             kent2.galab.uwindsor.ca:4559  ESTABLISHED
  TCP    kent8:1669             kent8.galab.uwindsor.ca:ldap  ESTABLISHED
  TCP    kent8:1670             kent8.galab.uwindsor.ca:ldap  ESTABLISHED
```

Figure 6-2: LDAP Server Connections State

In any case, the following configurations apply for every client:

- LDAP server: kent8.galab.uwindsor.ca
- Search base: c=ca
- Port number: 389

If a firewall is existed in the domain of LDAP server, the firewall should be configured by its system administrator in order to make it be accessed from another domain. Otherwise, the connection can not be established.

Another problem is also need to be considered. The idle timeout specifies the idle time of a connection between client to the server before the server drops the connection. Within a configuration entry "cn=config", there is an attribute "nsslapd-idletimeout" which Specifies the time after which an idle LDAP client connection is closed by the server. This value is given in seconds. During the idle period, the connection remains open, but no operations are requested. For example, if we set its value is 300 seconds, then after five minutes, the connection status is from "ESTABLISHED" to "CLOSE WAIT". But timeout setting will not stop normal search operation to the server. It is only applied to the idle period. We should limit idle time for the client applications to access directory server. Leaving them idle or unused may impact server performance because of opening many connections.

6.2 Using Web Browser

A Uniform Resource Locator (URL) is a short string of characters that identifies a resource through its primary access mechanism such as its network address. It is a standard way to name hypertext link destinations for web browsers. Figure 6-3 describes a format for an LDAP Uniform Resource Locator. The format describes an LDAP search operation to perform to retrieve information from an LDAP directory.

```
                              URL Definition

<ldapurl> ::= "ldap://" [ <hostport> ] "/" <dn> [ "?" <attributes>
                    [ "?" <scope> "?" <filter> ] ]
<hostport> ::= <hostname> [ ":" <portnumber> ]
<dn> ::= a string as defined in RFC 1485
<attributes> ::= NULL | <attributelist>
<attributelist> ::= <attributetype>
                    | <attributetype> [ "," <attributelist> ]
<attributetype> ::= a string as defined in RFC 1777
<scope> ::= "base" | "one" | "sub"
<filter> ::= a string as defined in RFC 1558
```

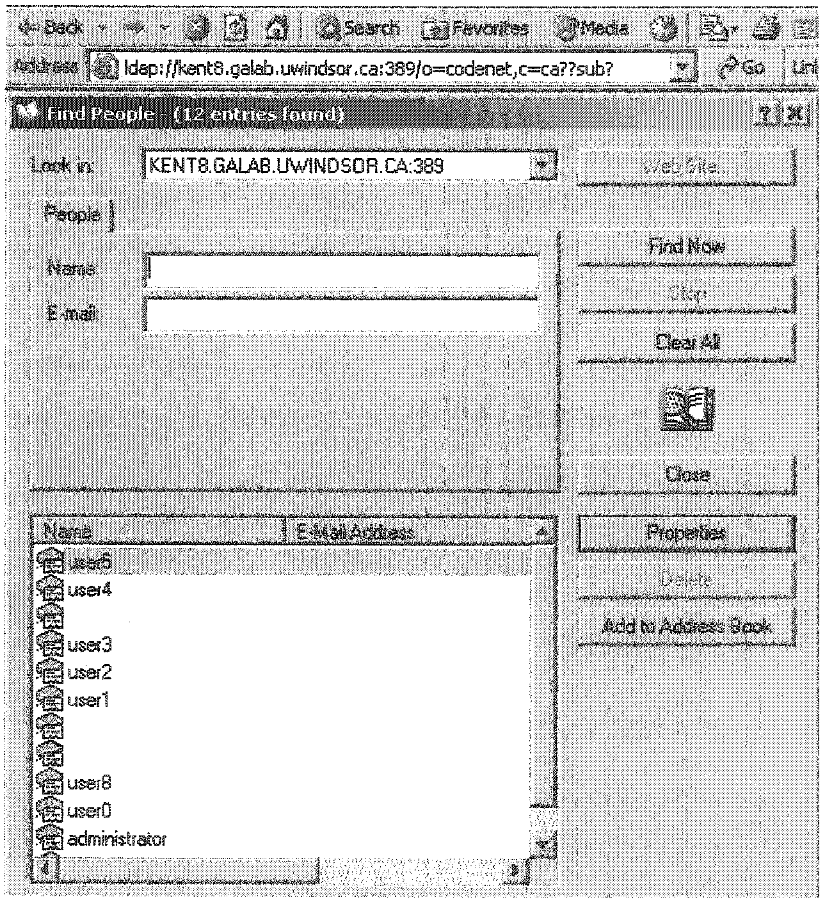
Figure 6-3: LDAP URL Format

This scheme supports queries to the Lightweight Directory Access Protocol (LDAP) for hierarchically organized information including user and resources. It uses `ldap://hostport/dn?attributes?scope?filter` as LDAP URL. The <dn> is an LDAP Distinguished Name using the string format. It identifies the base object of the LDAP search (RFC 2253). The <attributes> construct is used to indicate which attributes should be returned from the entry. Individual <attributetype> names are as defined for AttributeType in RFC 1777. If the <attributes> part is not used, all attributes of the entry will be returned. The <scope> is used to specify the scope of the search to perform in the LDAP server. The allowable scopes are “base” for a base object search, “one” for a one-level search, or “sub” for a sub tree search. If scope is omitted, “base” is assumed. The <filter> is used to specify the search filter which makes it return subset of entries. If omitted, all entries should be returned. A default filter is (objectClass=*). More information on the LDAP URL scheme is available in RFC 2255. We make a query that asks kent8.galab.uwindsor.ca for information about CodeNet in the Canada, the LDAP URL is:

```
ldap://kent8.galab.uwindsor.ca:389/o=codenet,c=ca??sub?
```

This URL corresponds to a base object search of the "o=codenet, c=ca" entry using a filter of (objectclass=*), requesting all subentries from the LDAP server `kent8.galab.uwindsor.ca` on port 389.

We have tested our LDAP server by using LDAP URL from several representative machines with some popular Web browsers including Microsoft^T Internet Explorer (see Figure 6-4), Netscape^T Communicator (see Figure 6-5) in a Microsoft^T Windows environment and Konqueror (see Figure 6-6) which is built in KDE in Linux.



Address: ldap://kent8.galab.uwindsor.ca:389/o=codenet,c=ca??sub?

Find People - (12 entries found)

Look in: KENT8.GALAB.UWINDSOR.CA:389

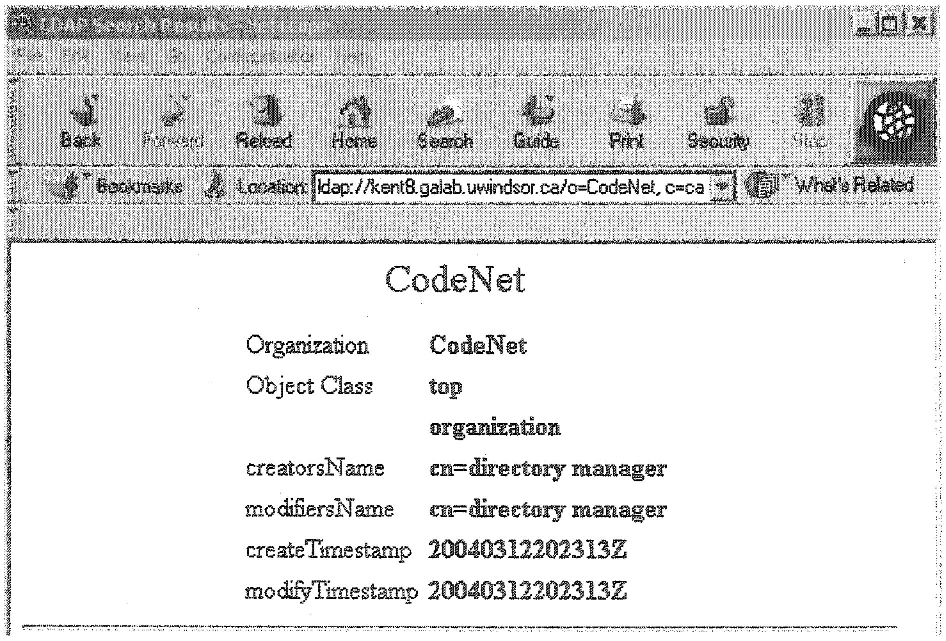
Name	E-Mail Address
user5	
user4	
user3	
user2	
user1	
user8	
user0	
administrator	

Figure 6-4: Connecting LDAP Server using IE

(Microsoft Internet Explorer 6.0)

Figure 6-4 shows the search result after using LDAP URL: "ldap://kent8.galab.uwindsor.ca:389/o=codenet,c=ca??sub?" in IE 6.0. It returns all the information from user0 to user8 within the subentries of Base DN " o=codenet, c=ca" from the LDAP server. Using "sub" performs a sub tree search.

Using Netscape Communicator 4.8 to connect LDAP server, Figure 6-5 shows all the attributes of Base DN "o=codenet, c=ca" without any subentries after using "ldap://kent8.galab.uwindsor.ca/o=codenet, c=ca" which performs only a base object search.



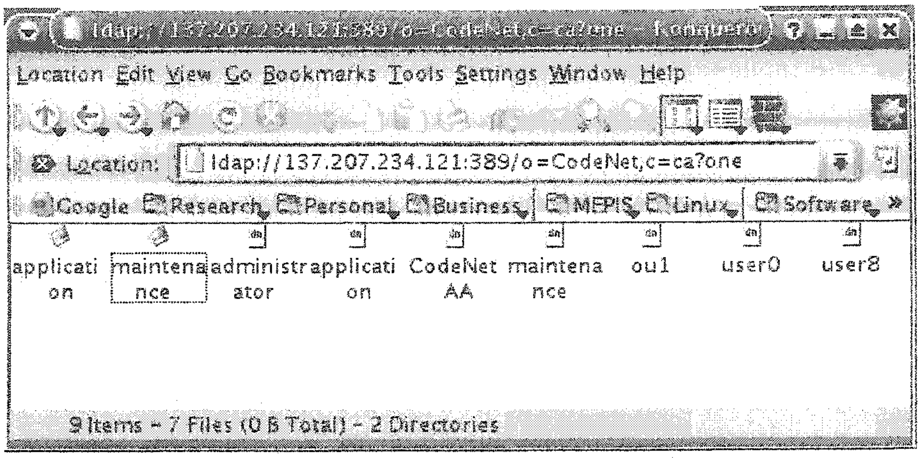
ldap://kent8.galab.uwindsor.ca/o=CodeNet, c=ca

CodeNet

Organization	**CodeNet**
Object Class	**top** **organization**
creatorsName	**cn=directory manager**
modifiersName	**cn=directory manager**
createTimestamp	**20040312202313Z**
modifyTimestamp	**20040312202313Z**

Figure 6-5: Connecting LDAP Server using Netscape

(Netscape Communicator 4.8)



ldap://137.207.234.121:389/o=CodeNet,c=ca?one

9 Items - 7 Files (0 B Total) - 2 Directories

Figure 6-6: Connecting LDAP Server using Konqueror

(Konqueror 3.1.3 on Linux)

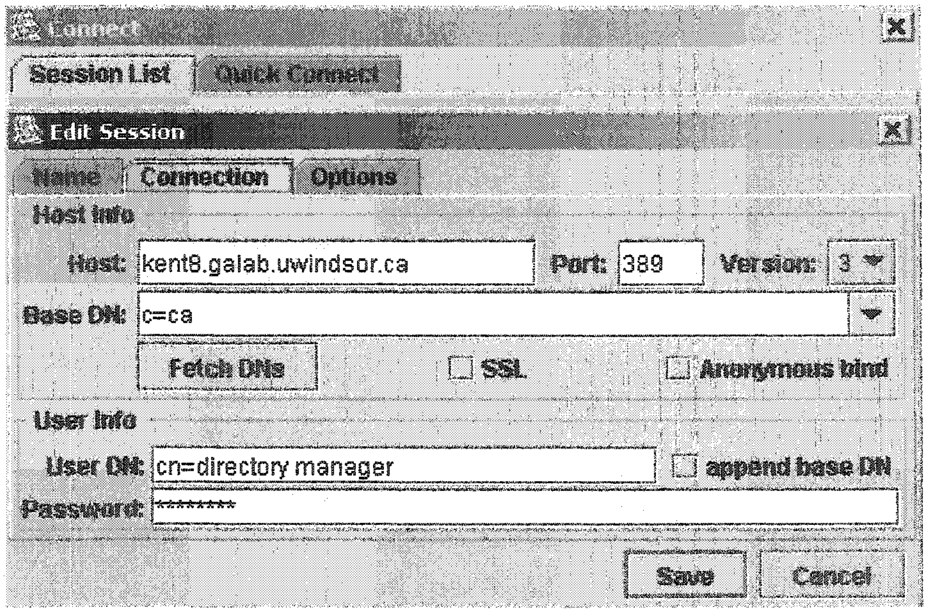
Figure 6-6 shows all the entries of only one level under the Base DN "o=codenet, c=ca" in the Konqueror Web browser in linux. "ldap://kent8.galab.uwindsor.ca/o=codenet, c=ca" performs only a one-level search.

6.3 Using LDAP Browser/Editor

The client of the LDAP can use the LDAP protocol to request information from an LDAP server. The nature of LDAP allows computers on any platform to communicate with LDAP directory server. The LDAP Browser/Editor (LBE) allows users to view the entries stored in a LDAP server in a hierarchical manner. It also allows modifications of the LDAP contents. We use the LBE to test our LDAP server. Figure 6-8 shows the results after using LBE to connect with LDAP server.

The LDAP Browser/Editor provides an interface to LDAP directories server with tightly integrated browsing and editing capabilities. It requires Java version 1.2.2 or greater. The Browser will first try to use the Java interpreter specified by the JAVA_HOME environment variable. We can run the browser with a different Java environment by setting the JAVA_HOME environment variable from the command line before starting the Browser. For Windows, we use C:\ldapbrowser>set JAVA_HOME=c:\jdk1.3; for Linux, we use ~/ldapbrowser>setenv JAVA_HOME /sandbox/jdk1.3. After the connection between LBE and our LDAP server is successfully established, we use it to add, delete and modify entries and their attributes (see following Figures) in the LDAP directory remotely.

From LBE main menu "File", we choose "New Window", and then we enter all the information including the host, port no, base DN, User DN and password for authentication (see Figure6-7). The information is used for connecting with LDAP server.



Host: kent8.galab.uwindsor.ca Port: 389 Version: 3

Base DN: c=ca

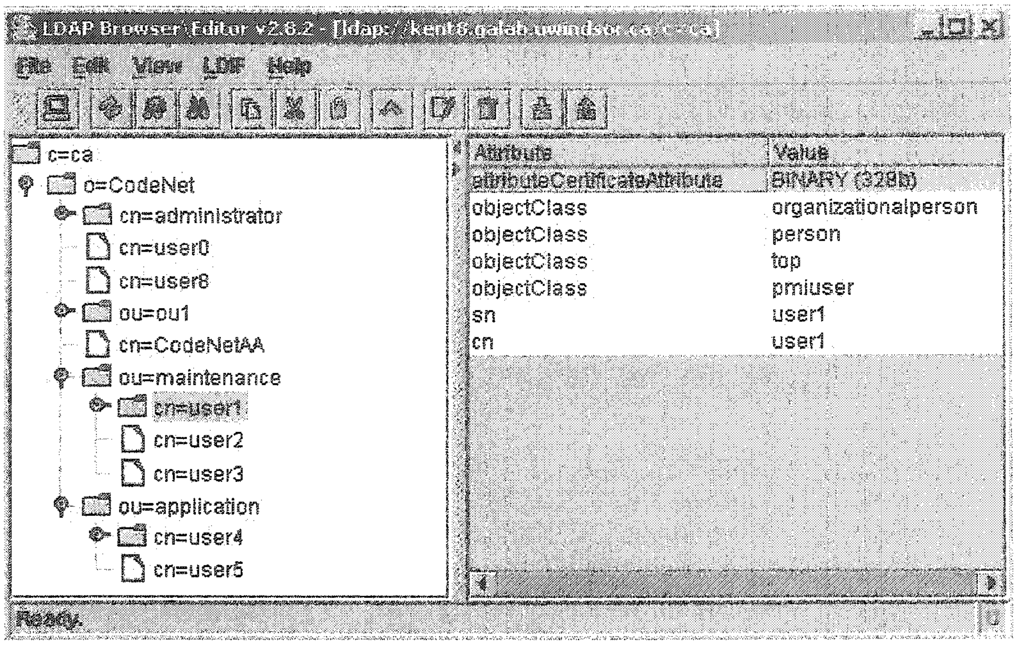
□ SSL □ Anonymous bind

User DN: cn=directory manager □ append base DN

Password: ********

Figure 6-7: Configuration for LBE

If the connection is established, directory information tree of the LDAP server will be shown as Figure 6-8.


LDAP Browser\Editor v2.8.2 - [ldap://kent8.galab.uwindsor.ca/c=ca]
File Edit View LDIF Help
Ready.

c=ca
- o=CodeNet
  - cn=administrator
  - cn=user0
  - cn=user8
  - ou=ou1
  - cn=CodeNetAA
  - ou=maintenance
    - cn=user1
    - cn=user2
    - cn=user3
  - ou=application
    - cn=user4
    - cn=user5

Attribute	Value
attributeCertificateAttribute	BINARY (329b)
objectClass	organizationalperson
objectClass	person
objectClass	top
objectClass	pmiuser
sn	user1
cn	user1

Figure 6-8: Connecting LDAP Server using LBE

Figure 6-9, Figure 6-10 and Figure 6-11 show how to add new entries to the directory. If the entry is successfully created, it will be shown on the tree (see Figure 6-11). And in Figure 6-9, using "Delete Entry" will deletes the selected entry or entries on the tree. This will only work if the entries have no children. In order to remove entries with children, delete tool provided by "Delete Tree" function found in the "Delete Entry" tab.

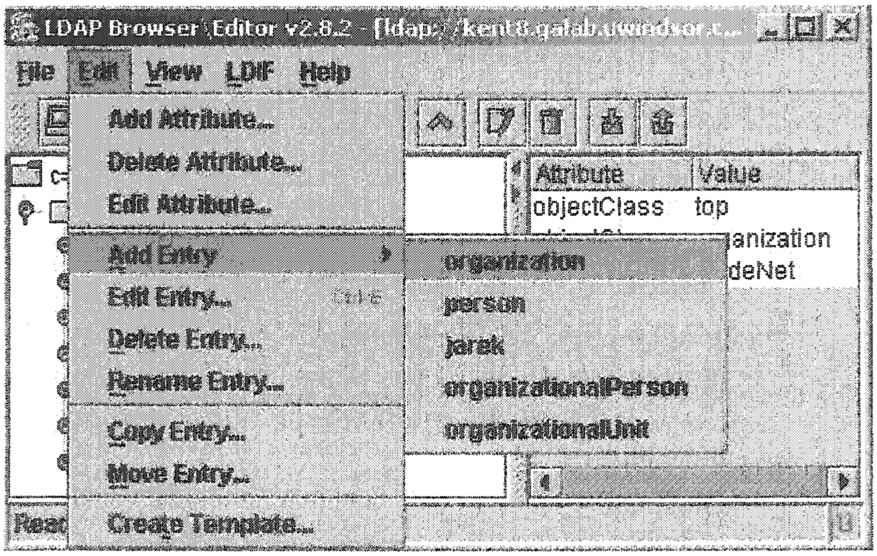
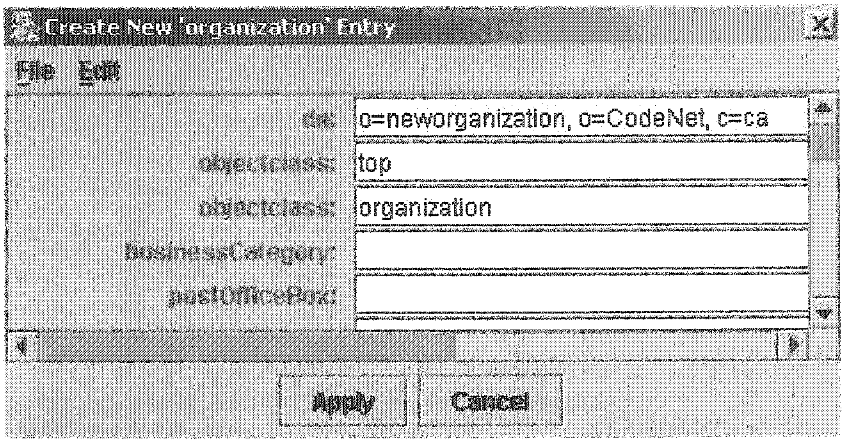


Figure 6-9: Adding an Entry Using LBE (1)

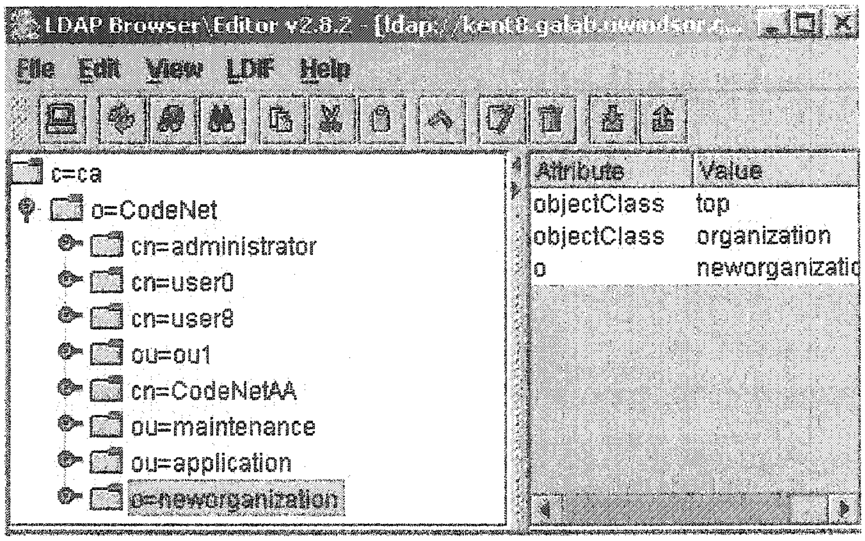
Figure 6-10 shows adding a new organization onto LDAP server. Fill in the required fields such as dn and object class, and then press "Apply" to continue.



dn: o=neworganization, o=CodeNet, c=ca
objectclass: top
objectclass: organization
businessCategory:
postOfficeBox:

Figure 6-10: Adding an Entry Using LBE (2)

Figure 6-11 shows the result after adding a new entry, we can see "o=neworganization" has been added into our LDAP server.

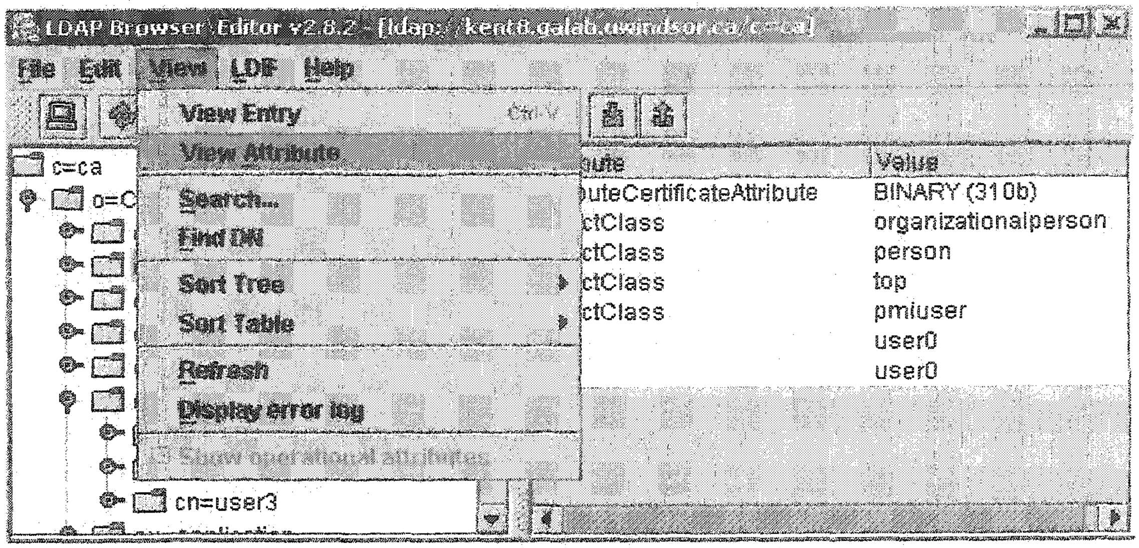

LDAP Browser\Editor v2.8.2 - [ldap://kent8.galab.uwindsor.c...
File
Edit
View
LDIF
Help

```
c=ca
  o=CodeNet
    cn=administrator
    cn=user0
    cn=user8
    ou=ou1
    cn=CodeNetAA
    ou=maintenance
    ou=application
    o=neworganization
```

Attribute	Value
objectClass	top
objectClass	organization
o	neworganizatio

Figure 6-11: Adding an Entry Using LBE (3)

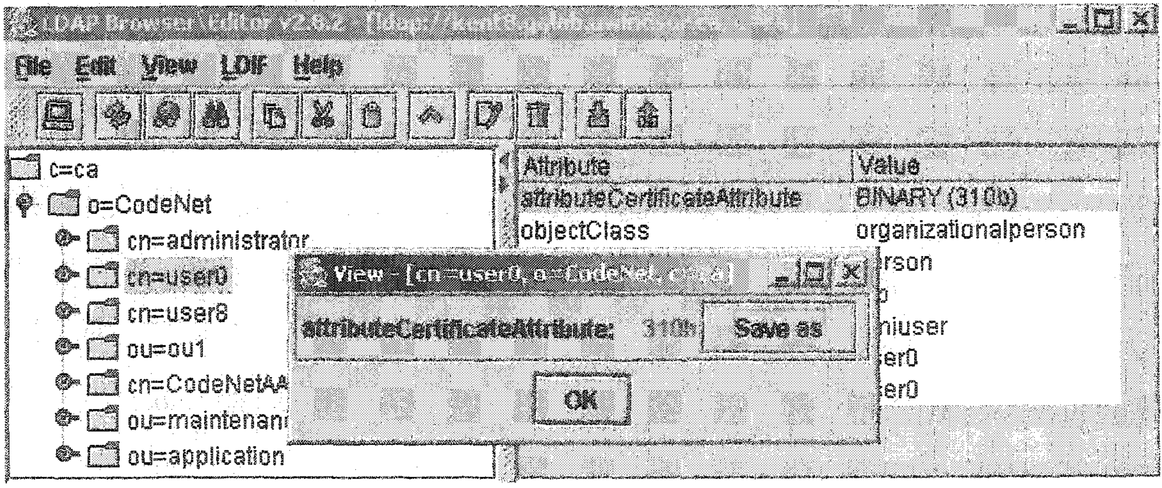
Figure 6-12 and Figure 6-13 shows how to view all the attributes. The attribute names and values are displayed in the attribute table.



...bute	Value
...uteCertificateAttribute	BINARY (310b)
...ctClass	organizationalperson
...ctClass	person
...ctClass	top
...ctClass	pmiuser
	user0
	user0

Figure 6-12: Viewing Attribute Using LBE (1)

Figure 6-13 shows the attribute for ACs. The value of an "attributeCertificateAttribute" is binary and the size is 310 bytes. We can obtain this AC and save it in a file using 'save as' button.

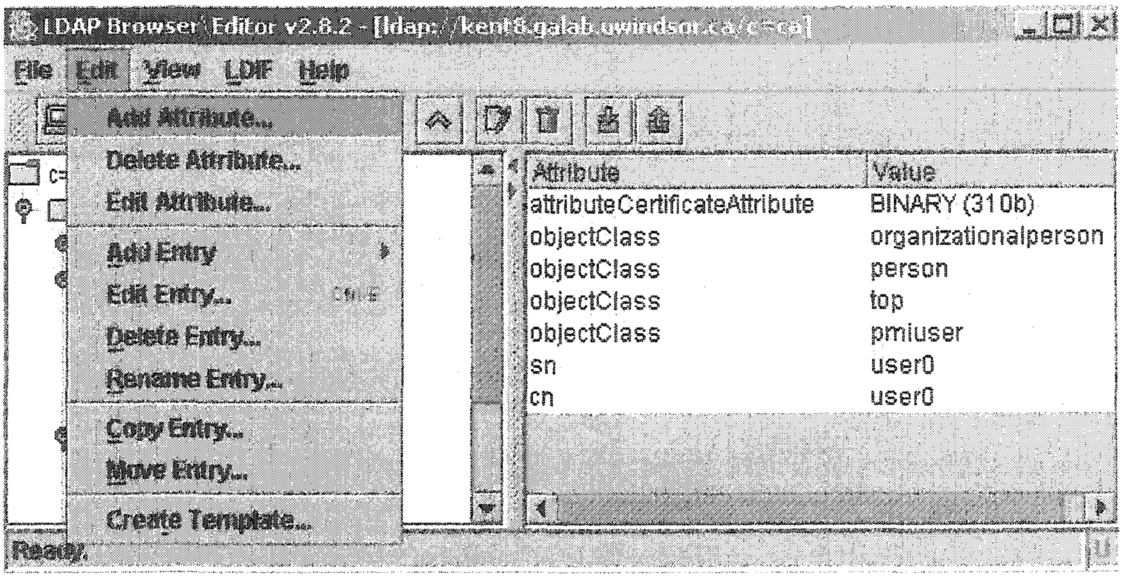


Attribute	Value
attributeCertificateAttribute	BINARY (310b)
objectClass	organizationalperson
	...rson
	...p
	...iuser
	...er0
	...er0

attributeCertificateAttribute: 310b

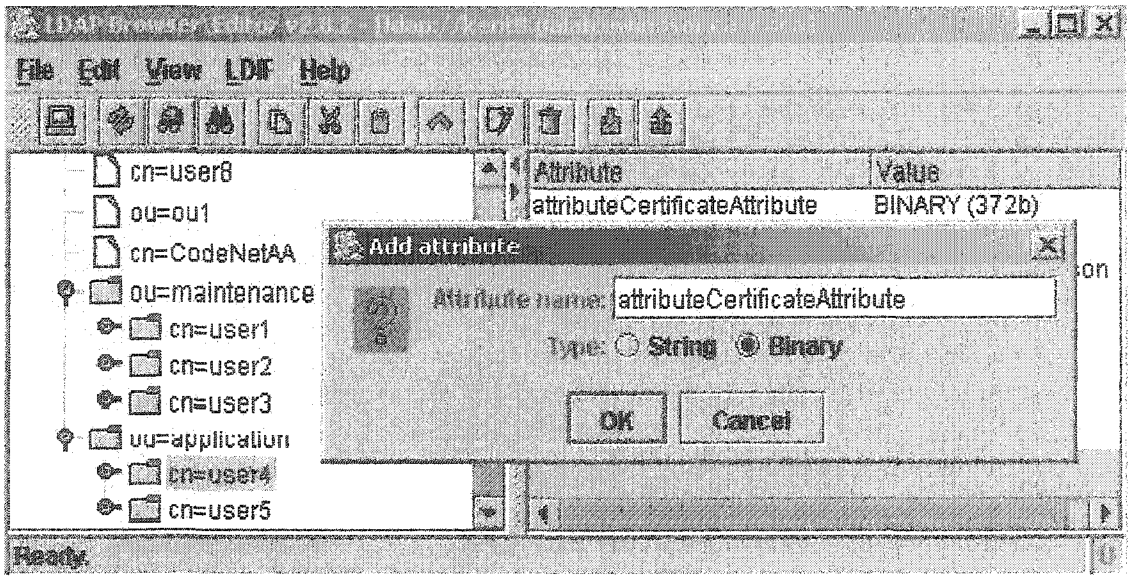
Figure 6-13: Viewing Attribute Using LBE (2)

From Figure 6-14 to Figure 6-17 shows how we can add "attributeCertificateAttribute" to a user entry, and then upload its corresponding AC from files to the entry. Figure 6-14 the functions provided by LBE including "Add Attribute" and "Delete Attribute" which can delete the selected attribute or attributes of the specified entry. In the following figures, there is detailed information about how to add an AC to the selected user entry on the LDAP server.



Attribute	Value
attributeCertificateAttribute	BINARY (310b)
objectClass	organizationalperson
objectClass	person
objectClass	top
objectClass	pmiuser
sn	user0
cn	user0

Figure 6-14: Adding AC onto LDAP Remotely Using LBE (1)



Attribute	Value
attributeCertificateAttribute	BINARY (372b)

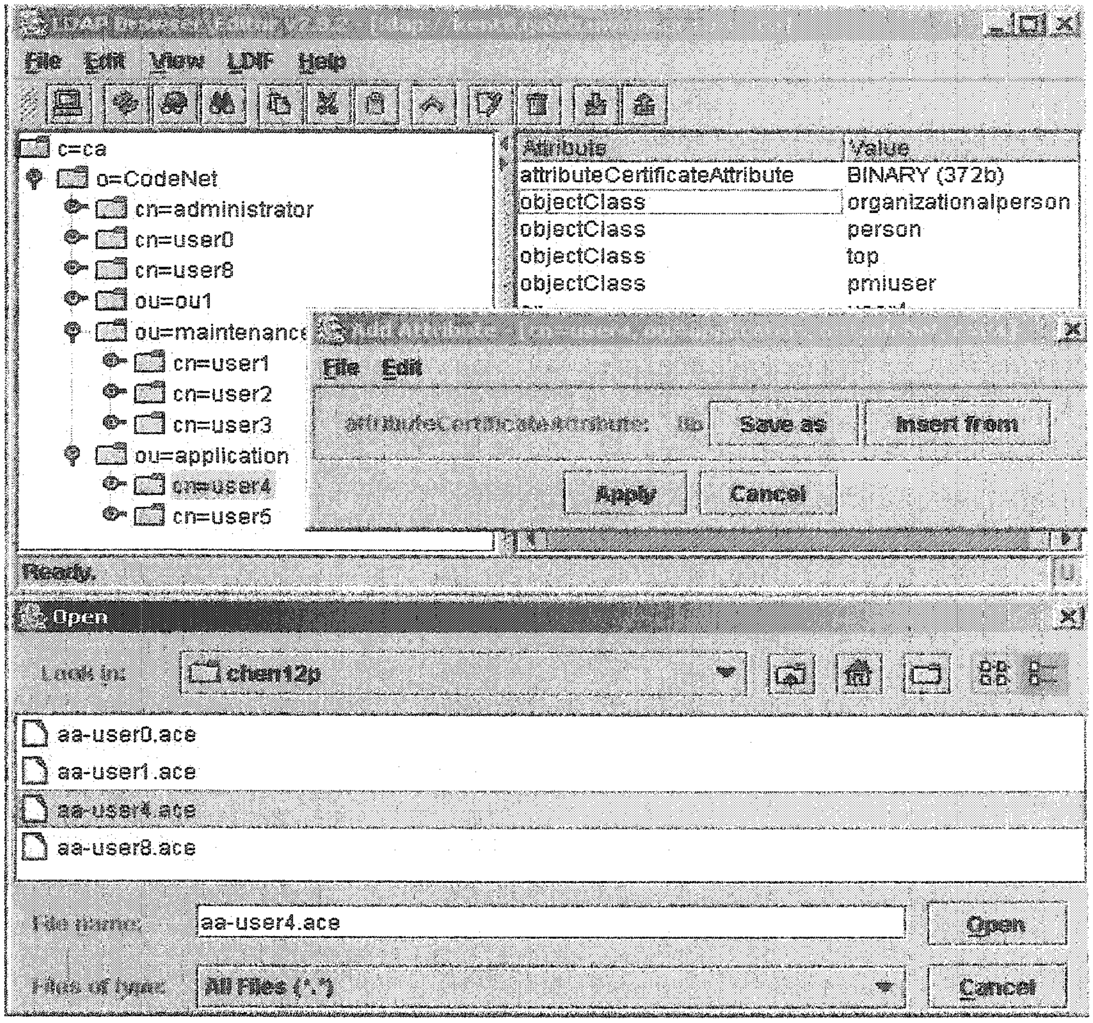
Attribute name: attributeCertificateAttribute

Type: ○ String ● Binary

Figure 6-15: Adding AC onto LDAP Remotely Using LBE (2)

After choosing "Add Attribute" in Figure 6-14, a window prompts for the attribute name and type shown in Figure 6-15. Enter the attribute name "attributeCertificateAttribute" and select a binary for the attribute.

Then an editor window will appear with the attribute name in Figure 6-16. We use the "Insert from" button to load the AC file "aa-user4.ace", and press "Apply" to add it into LDAP server.



c=ca
o=CodeNet
cn=administrator
cn=user0
cn=user8
ou=ou1
ou=maintenance
cn=user1
cn=user2
cn=user3
ou=application
cn=user4
cn=user5

Attribute	Value
attributeCertificateAttribute	BINARY (372b)
objectClass	organizationalperson
objectClass	person
objectClass	top
objectClass	pmiuser

Look in: chen12p

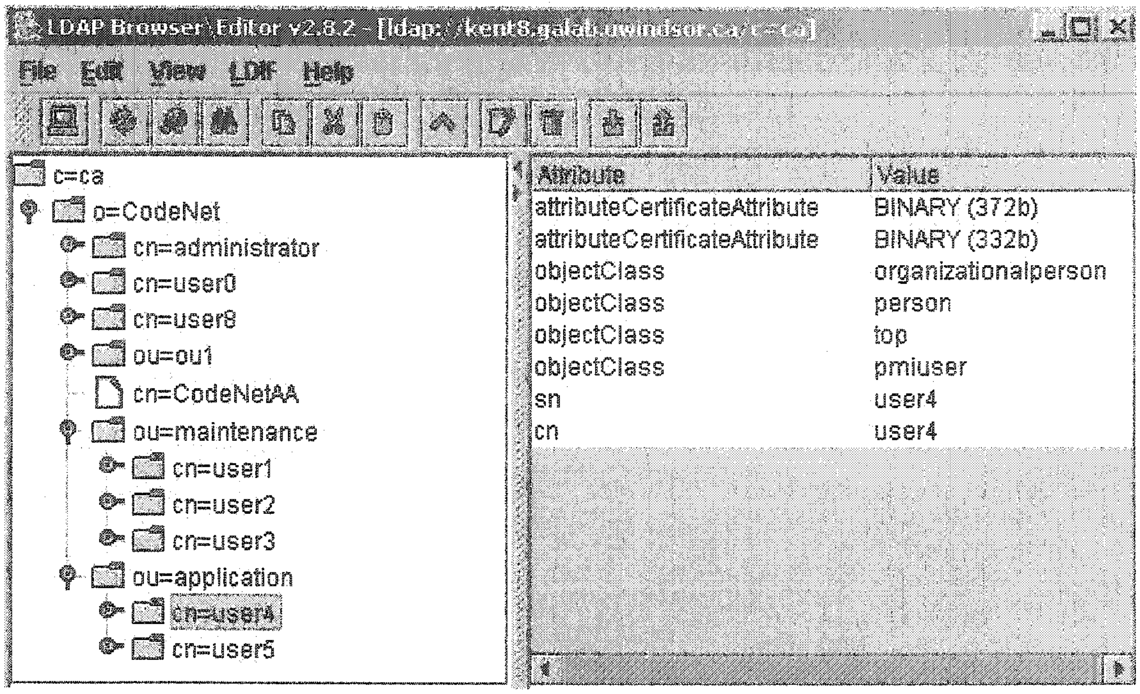
aa-user0.ace
aa-user1.ace
aa-user4.ace
aa-user8.ace

File name: aa-user4.ace

Files of type: All Files (*.*)

Figure 6-16: Adding AC onto LDAP Remotely Using LBE (3)

Figure 6-17 shows the final results about adding AC, a new attribute certificate whose value is Binary and size is 332 bytes has been added in the LDAP Server.



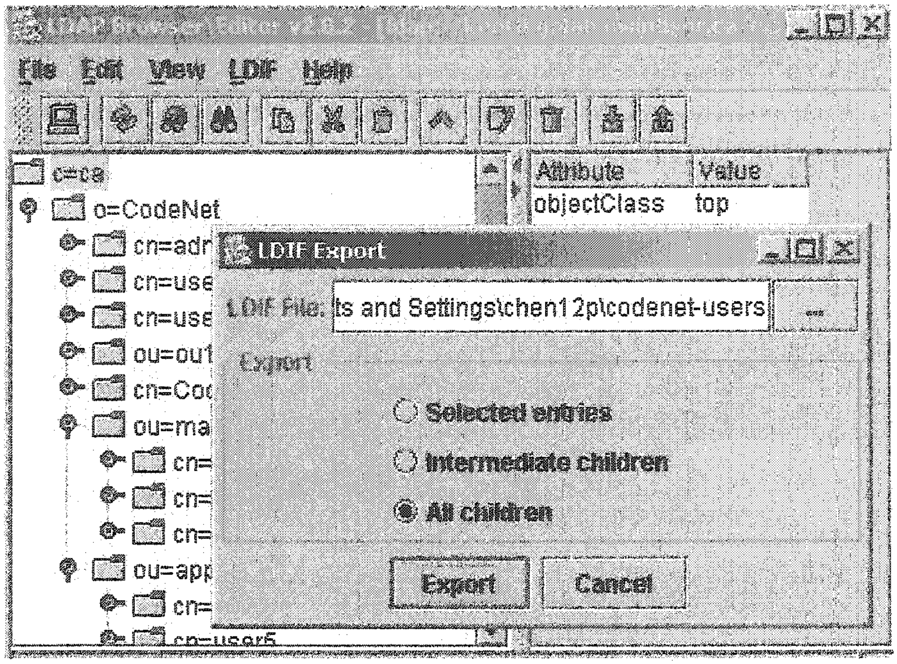
LDAP Browser\Editor v2.8.2 - [ldap://kent8.galab.uwindsor.ca/c=ca]
File Edit View LDIF Help

```
c=ca
  o=CodeNet
    cn=administrator
    cn=user0
    cn=user8
    ou=ou1
    cn=CodeNetAA
    ou=maintenance
      cn=user1
      cn=user2
      cn=user3
    ou=application
      cn=user4
      cn=user5
```

Attribute	Value
attributeCertificateAttribute	BINARY (372b)
attributeCertificateAttribute	BINARY (332b)
objectClass	organizationalperson
objectClass	person
objectClass	top
objectClass	pmiuser
sn	user4
cn	user4

Figure 6-17: Adding AC onto LDAP Remotely Using LBE (4)

LBE supports the LDIF file format. "Export" function can store the selected entry or entries into a LDIF file. We can also import entries from a LDIF file and insert them into the LDAP directory. Figure 6-18 shows how to export an entry "c=ca" with all its children that is the subentries into one LDIF file called "codenet-users.ldif". Figure 6-19 shows the result for the export operation, "Exported 14 entries" means 14 subentries of "c=ca" has been exported successfully.



objectClass top

LDIF File: ts and Settings\chen12p\codenet-users

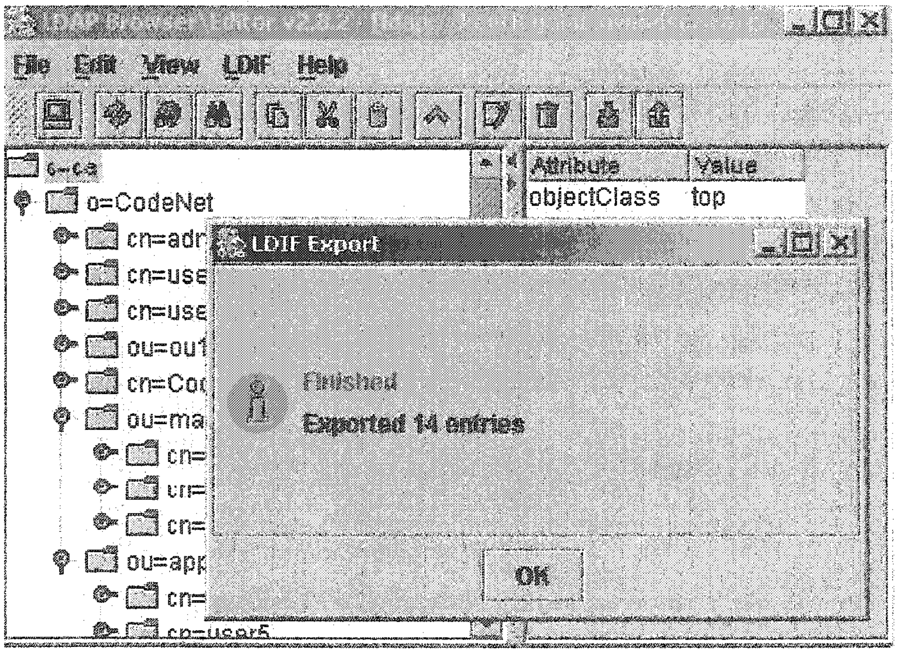
Export

○ Selected entries

○ Intermediate children

● All children

Figure 6-18: Exporting LDIF File Using LBE (1)



objectClass top

Finished

Exported 14 entries

Figure 6-19: Exporting LDIF File Using LBE (2)

6.4 Features

This thesis describes enhanced security architecture by using LDAP server and attribute certificates which can support distributed access control. Our proposed architecture for MyProxy repository is both new and innovative. With this new approach and architecture, it makes both users and resource providers access and manage their resource more securely and conveniently. Each resource provider only needs to define their access policies and need not keep the information of all users. They can manage a large number of users by using both MyProxy and LDAP server. The overhead at the resource providers will reduce because they only make access decision on a set of roles rather than a great number of users.

The features of this innovative architecture are as follows:

- Enhanced Security. Design enhanced security architecture to provide decentralized authorization. The attribute certificates are stored in LDAP directories so that they can be accessed via the Internet. This allows the administration of privileges to be widely distributed over the Internet. It also supports role based access control. We combine with authentication and authorization mechanism based on PKC (public key certificate) and AC (attribute certificate).

- Scalability. Access policy based on roles or groups in attribute certificates supports the scalability. When adding a new user, we only need to create an AC for that user and put it onto the LDAP server, it will not affect the other component of the system. And LDAP server also supports user entries up to one million which can meet the requirements of the increased number of users.

- Accessibility. In our architecture, MyProxy allows users to access their proxy identity certificates from anywhere and LDAP server also allows users to upload or retrieve their attribute certificates from anywhere. Users can access grid resources by using their identity certificates and ACs to perform authentication and authorization at anywhere.

- Flexibility. This architecture enable distributed resource providers to control their resources and specify access requirements flexibly. They can define their own access policy for various roles or attributes. Resource providers can create or change any type of roles of a user, and ACs can also be retrieved on different platforms such as Windows and Linux.

- Integration. Our architecture can integrate with many current technologies. Attribute certificates can integrate with PKC infrastructures (see Chapter 4.1); MyProxy can integrate with GSI (see Chapter 2.2); and LDAP can be easily integrated with different Web browsers (see Chapter 6.2).

6.5 Analysis on Performance of LDAP Server

We used Sun One Directory Server 5.2 as the basis for implementing our LDAP server and authorization schema. The document [Sun03] for Sun ONE Directory Server 5.2 on the Sun Website describes in detail its key features. This LDAP server is designed to deploy extensible, secure, global directory services. We list some of it functionality as follows [Sun03]:

- *64-bit large cache support for high volume deployments*
- *Configurable encryption for all attributes*
- *Directory access through DSML v2 for web services*
- *Replication monitoring and management tools to handle large topologies*
- *Sun Cluster agents for additional high availability services*

This product represents the current state-of-the-art in server architecture and performance and can be used to gauge the achievability of suitable support for authorization handling. We focus on the analysis of search performance of LDAP server. The Sun ONE Directory Server 5.2 can run as a 64-bit application. It can use a cache larger than the 4 GB limit.

Using the large cache allows better performance for high-volume systems. The multiple database architecture of the Sun ONE Directory Server supports distributed naming contexts. It provides large scalability to support millions of users on a single system. The entry, database, and import caches can be larger than 4 GB. With the increased cache size, searches can scale linearly on servers with up to 12 processors. High-performance design supports 64-bit caching on Solaris and HP-UX operating systems. Performance is also improved through reduced memory allocation and improved thread management.

To use one example, Sun One Directory Server has the best performance for over a million entries based on previous testing statistics. Those performance tests cover loading, searching, adding, deleting and modifying the data. Tests were performed on directory trees containing up to 10 million entries [Sun03]. Based on these operating characteristics and performance of an LDAP server, we use it to provide directory services for storing ACs in our security architecture for grid systems.

It must be kept in mind that the size of the search path will be limited by hardware (e.g. disk drives), and by the relative population of the directory entries at each level in the LDAP hierarchy. Hence, smaller machines may retain higher request latency even though the total number of LDAP entries is much smaller.

6.6 Observations of Search Query Performance

When the client is initialized, an IP address of LDAP server and a search base is specified. This information can be specified as a command line argument to the ldapclient command, or be provided by other client tools. For example, if we want to find "cn=user0, o=CodeNet, c=ca" The steps that the ldapclient command perform are:

1. Search the LDAP server from the base DN "c=ca" which is specified by client. Typical number of "country" in the world may be about 200.

2. Search the found base DN for an entry containing an object class such as "organization". The real number may change from hundreds to millions according to different countries.
3. Check the found organization attribute of the entry to verify if its value equals the value "CodeNet" stored in the client's request.
4. Search the cn=* or ou=* directly below the entry "o=CodeNet, c=ca" for an entry that matches the common name or name of organization unite provided by client. In this example, if the value of cn equals "user0", then the entry requested has been found.

We note that the directory structure is a tree and hierarchy supports a "divide and conquer" strategy for searching; this leads immediately to logarithmic complexity in the partitioning of the search spaces. It is to be noted that the search algorithm currently used in LDAP is the linear search technique, but for relatively small search lists this achieves good performance. At the lowest level of the directory hierarchy the search space is typically must larger and linear search may be slow. In order to improve the performance, we would recommend further study on the use of both binary search and hashing techniques. These matters have been left as subjects for possible future work.

6.7 Limitations

In this thesis research, our focus has been on the architectural design based on fundamental notions of authorization. We also provide the deployment and testing of the LDAP server for the proposed architecture. In practice, a production level Grid portal should be capable of handling the increased capabilities of authorization, in addition to the conventional authentication capability exhibited within MyProxy, for example. Such a tool must be quite flexible, robust and afford significant guarantees to users. There are some limitations for our proposed security architecture. These include:

- Before using LDAP directory server in grid environments, some problems such as firewall should be considered and solved. Otherwise the LDAP server may not be accessed outside the firewall. From the outside, it may be inaccessible because our authorization approach will serve the need of grid nodes featuring substantial resources. As such, the installation, including integration within existing environments, might be a complex task involving various IT professionals, particularly, system administrators. Such circumstances are not uncommon in large institutions, however.

- Some versions of LDAP may not support storing attribute certificates; these include examples such as Microsoft Active Directory or the old version of Sun server.

- In section 6.2, testing has been done using some Web browsers. IE and Netscape are both popular in Windows system, and Konqueror is also widely used in Linux. To determine the grade of compatibility to support LDAP URL, future testing on other Web browsers, different versions and operating systems may be needed.

- In a certificate-based architecture, there is the problem of certificate revocation when making certificates invalidate after they have been issued but before their expiration dates. All the revocation methods add complexity and cost to the access control architecture. In large-scale distributed networks, it is impossible to implement real-time revocation as in a centralized system. Therefore, we limit the validity time and authorization information in the certificates rather than using revocation lists. These remaining problems will need further research.

In addition to the problems of limitations listed above, there exist a variety of specific system functionalities that have not been developed for the thesis. Such "bells and whistles" would be added, both internally and externally, as in GUI applications, in a production system version.

7. CONCLUSION AND FUTURE WORK

In this chapter we provide a summary of the work presented in this thesis, and identify what has been achieved. We conclude with an account of future work.

7.1 Summary of Thesis

Computational grids are an emerging technology that allows the combination of widely distributed resources to support large-scale computations. We have presented a comprehensive survey of security technologies, policy specification approaches and various security architectures. We proposed an enhanced grid security architecture for support of MyProxy online credential repository.

Our enhanced security architecture provides decentralized authorization. The attribute certificates can be accessed via the Internet. This allows the administration of privileges to be widely distributed over the Internet. It also supports role based access control and delegation. We combine with authentication and authorization mechanism by using both MyProxy online credential repository and LDAP directory server. It enables end users and resource providers to access resources through X.509 attribute certificates that carry authorization information such as role or rights and policy statements.

A person can be assigned a role and get all the rights applying to the role. It conveniently specifies the rights and duties for a user. Based on these concepts, we have designed attribute certificates with role information, and build an LDAP server for storing and uploading them. These services enable user management by support of RBAC and improve the abilities of existing systems that focus on authorization.

Our security mechanisms, based on a public-key infrastructure, have been added for the secure support of authorization and the integration of existing grid security software. The solution employs attribute certificates to bind rights to users and facilitates managing privileges. In our approach, attribute certificates support short-lived lifetime and contain attributes, multiple attributes can be added in one AC. A PKC is used to authenticate a user, and an AC describing attributes is used for authorization.

We have designed attribute certificates with role information, and build an LDAP server for storing and uploading them. These services enable user management by support of RBAC and improve the abilities of existing systems that focus on authorization. Our approach makes use of the online credential repository to build authentication, delegation and attribute based access control together to provide enhanced security for grid systems. Section 4.3 describes the three components in our proposed security architecture, and six basic steps for providing both authentication and authorization service. In this thesis, we address the second component of our security architecture, LDAP directory server, and provide details on how to design our schema for support of ACs. We also tested our LDAP server in different platforms using Web browsers and LBE. Our enhanced security architecture is able to integrate many well-known techniques, such as X.509, SSL, LDAP, and MyProxy, which provide compatibility with current security technologies. The design of the enhanced security architecture for support of MyProxy and the deployment of LDAP server for storing AC are both innovative and constitute the main contributions of this thesis.

Our architecture provides for distributed management of role based access control over resources. It allows resource providers to share the authorization information of users and control their resources remotely, and may allow users to access multiple resources by using the same roles or different roles. Our proposed architecture improves the security services of computational grids. This research is motivated to support sharing authorization information and facilitate collaborations of grid users. It also contributes to the scalability of security mechanisms in grid environments.

7.2 Future Research Directions

The next stages of the development in this architecture are described as follows:

- For this thesis research we concentrated on the problem of authorization applied to a single resource-provider node. We used the Sun One directory server version 5.2 and deployed it on a single server. In general, however, it is important to use multiple servers across various domains for large-scale networks. To deal with the increased problems of multiple resource-provider nodes 'that might share authorization information additional work is required on the management issues inherent to an extended architecture.

- MindCraft's DirectoryMark 1.2.1 is widely recognised as the industry performance benchmark for LDAP directories. Using this measurement tool for testing query response time and the practical throughput of the LDAP server would be highly useful in detected areas for improvement in design, such as the core search algorithms.

- Security policy needs to be designed for resource providers and the authorization model for executing the policy together with ACs needs to be developed for the proposed security architecture.

- Develop a grid portal which is able to support authentication and authorization service using proxy credentials and attribute certificates.

- Integrating this work with the GSI (Globus) work through an integrated environment including MyProxy for the simulation and viewing the results should be continued.

- Finally, integrating all management tools to provide a uniform interface will be developed further.

And other possible future developments may be includes: design an approach to solve the revocation problem, modifying the protocols of MyProxy to support different forms of credentials and integrate LDAP with more web technologies and protocols.

We believe that our contribution is an important step towards offering strong and enhanced security management for grid computing.

REFERENCES

1. [Adams99] Adams, C., Lloyd, S. (1999). Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations. *Macmillan Technical Publishing*, 1999

2. [Adamson01] Adamson, W.A. and Kornievskaia, O. A Practical Distributed Authorization System for GARA, *Information Technology Integration technical report*, November 2001.

3. [Aura98] Aura, T. On the Structure of Delegation Network. *Proceedings 11th IEEE Computer Security Foundations Workshop*, pp 14-26, June 1998.

4. [Aura99] Aura, T. Distributed access-rights management with delegation certificates. *Secure Internet Programming: Security Issues for Mobile and Distributed Objects* LNCS 1603, pp 213-238, 1999.

5. [Baker02] Baker, M., Buyya, R., Laforenza, D. Grids and Grid technologies for wide-area distributed computing. *Software-Practice & Experience,* 32(15), pp 1437-1466 Dec. 2002.

6. [Belani98] Belani, E., Vahdat, A., Anderson, T., Dahlin, M. The CRISIS wide Area Security Architecture. *8th Usenix Security Symposium*, Jan. 1998. http://citeseer.nj.nec.com/belani98crisi.html

7. [Blaze96] Blaze, M., Feigenbaum, J. and Lacy, J. Decentralized Trust Management, *Proceedings IEEE Symposium on Security and Privacy*, pp 164-173, May 1996.

8. [Blaze98] Blaze, M., Feigenbaum J. and Strauss, M. Compliance Checking in the Policy Maker Trust Management System. *Proceedings of the Financial Cryptography '98*, Lecture Notes in Computer Science, (1465), pp 254-274, 1998.

9. [Blaze99c] Blaze, M., Feigenabaum, J., Ioannidis, J., Keromytis, A. The KeyNote Trust Management System, *Version 2. IETF RFC-2704*, 1999. http://www.crypto.com/papers/rfc2704.txt

10. [Butler99] Butler, R. Engert, D. Foster, I., Kesselman, C., Tuecke, S., Volmer J. and Welch, V. Design and Deployment of a National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12), pp 60-66, 1999.

11. [Butler00] Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J. and Welch, V. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12), pp 60-66, 2000.

12. [Chadwick03] Chadwick, D. W., Sahalayev, M. V. Internet X.509 Public Key Infrastructure LDAP Schema for X.509 Attribute Certificates, *<draft-ietf-pkix-ldap-ac-schema-00.txt>*, Feb. 2003.

13. [Damianou01] Damianou, N., Dulay, N., Lupu, E. and Sloman, M. The Ponder Policy Specification Language. *Proceedings Workshop on Policies for Distributed Systems and Networks*, Springer-Verlag LNCS 1995, pp 18-39, Bristol, UK, Jan.2001.

14. [Dulay01] Dulay, N., Lupu, E., Sloman, M., Damianou, N. A Policy Deployment Model for the Ponder Language. *Proceedings IEEE/IFIP International Symposium on Integrated Network Management* (IM'2001), pp 529-544, Seattle, USA, May 2001.

15. [Farrell98] Farrell, S. TLS Extensions for Attribute Certificate Based authorization. *IETF Transport Layer Security Working Group*, Aug.1998. http://www.ietf.org/proceedings/99jul/I-D/draft-ietf-tls-attr-cert-01.txt

16. [Farrell01] Farrell, S., Housley, R. An Internet Attribute Certificate Profile for Authorization, *draft-ietf-pkixac509prof-09.txt*, June 2001. http://www.ietf.org/internet-drafts/draft-ietf-pkix-ac509prof-09.txt

17. [Farrell02] Farrell, S., Housley, R. An Internet Attribute Certificate Profile for Authorization, *RFC 3281*, April 2002.

18. [Ferraiolo95] Ferraiolo, D., Cugini, J. and Kuhn, R. Role Based Access Control: Features and Motivations. *Proceedings of 11th Annual Computer Security Applications*, 1995. http://hissa.nist.gov/rbac/newpaper/rbac.html

19. [Ferrari99] Ferrari, A., Knabe, F., Humphrey, M. Chapin, S. and Grimshaw, A. A Flexible Security System for Metacomputing Environments. *7th International Conference on High Performance Computing and Networking Europe* (HPCN Europe 99), pp 370-380, Apr. 1999.

20. [Foster97a] Foster, I., Kesselman. C. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications,* 11(2), pp 115-128, 1997.

21. [Foster97b] Foster, I., Karonis, N.T., Kesselman, C., Koenig G. and Tuecke, S. A Secure Communications Infrastructure for High-Performance Distributed Computing. *Proceedings 6th IEEE Symp. on High Performance Distributed Computing*, pp 125-136, 1997.

22. [Foster98a] Foster, I., Karonis, N.T., Kesselman, C., Tuecke, S. Managing Security in High-Performance Distributed Computations. *Cluster Computing* 1(1) pp 95-107, 1998.

23. [Foster98b] Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A Security Architecture for Computational Grids. *Proceedings 5th ACM Conference on Computer and Communication Security*, pp 83-91, Nov.1998.

24. [Foster99a] Foster, I. and Kesselman, C. The Grid: Blueprint for a Future Computing Infrastructure. *Morgan Kaufmann*, 1999.

25. [Foster99b] Foster, I. and Kesselman, C. Globus: A Toolkit-Based Grid Architecture. In Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann*, pp 259-278, 1999.

26. [Foster01] Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling scalable virtual organizations. *International Journal Supercomputer Applications*, 15(3), 2001. http://citeseer.nj.nec.com/foster01anatomy.html

27. [Foster02b] Foster, I. Kesselman, C. Nick, J. Tuecke S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. The latest version can be found at http://www.globus.org/research/papers/ogsa.pdf. June, 2002.

28. [Gasser90] Gasser, M. and E. McDermott. An Architecture for Practical Delegation in a Distributed System. *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pp 20-30, May 1990.

29. [Hallam02] Hallam-Baker, P., Maler, E. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML). *Draft-ssct-core-31*, http://www.oasis-open.org/committees/security/docs/ draft-ssctcore-31.pdf, 2002.

30. [Hardjono94] Hardjono, T. and T. Ohta, Secure End-to-End Delegation in Distrbuted Systems. *Computer Communications,* 17(3), pp 230-238, Mar. 1994.

31. [Hayes00] Hayes, J. IND, A. Policy-based Authentication and Authorization: Secure Access to the Network Infrastructure. *Proceedings 16th Annual Computer Security Application Conference* (ACSAC '00), pp 328-333, Dec 2000.

32. [Housley01] Housely, R. Ford, W., Polk, W., Solo, D. Internet X.509 Public Key Infrastructure Certificate and CRL Profile, *draft-ietf-pkix-new-part1-12.txt,* http://www.ietf.org/internet-drafts/draft-ietf-pkix-new-part1-12.txt

33. [Howell00] Howell, J. and D. Kotz. End-to-end authorization. *Proceedings 2000 Symposium on Operating Systems Design and Implementation*, USENIX Association, 2000. http://citeseer.nj.nec.com/562954.html

34. [Hu01] Hu, Y. J. Some Thoughts on Agent Trust and Delegation. *Proceedings 5th international conference on autonomous agents*, pp 489-496, Montreal, May 2001. http://plum.cs.nccu.edu.tw/~jong/pub/agents01-talk.pdf

35. [Humphrey02] Humphrey, M., Thompson M.R. Security Implications of Typical Grid Computing Usage Scenarios. *Cluster Computing* 5(3), pp 257-264, Jul. 2002.

36. [ITU01] ISO/ITU-T Rec. X.509(2001) The Directory: Authentication Framework

37. [Johnston96] Johnston, W., and Larsen, C. A Use-Condition Centered Approach to Authenticated Global Capabilities: Security Architectures for Large-Scale Distributed Collaboratory Environments. *Imaging and Distributed Computing Group,* Ernest Orlando Lawrence Berkeley National Laboratory, University of California, 1996. http://www-itg.lbl.gov/~johnston/Security.Arch.Global.Cap.html

38. [Johnston98] Johnston, W., Mudumbai, S. and Thompson, M. Authorization and Attribute Certificates for Widely Distributed Access Control. *IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise – WETICE*, Stanford, CA. June 1998. http://www-itg.lbl.gov/security/Akenti/Papers/cert.based.access.control.ieee.pdf

39. [Johnston01] Johnston, W.E., Jackson, K.R. and Talwar, S. Security Considerations for Computational and Data Grids. *Proceedings 10th IEEE Symposium on High Performance Distributed Computing*, pp 439-440, Aug. 2001.

40. [Klasen02] Klasen, N., Gietz, P. An LDAPv3 Schema for X.509 Certificates, *<draft-klasen-ldap-x509certificate-schema-00.txt>*, Feb. 2002

41. [Kohl93] Kohl, J.T., Neuman, B.C. The Kerberos network authentication service: Version 5. *Draft protocol specification.* April 1993

42. [Kornievskaia01] Kornievskaia, O., Honeyman, P., Doster, B., Coffman, K. Kerberized Credential Translation: A Solution to Web Access Control. *10th USENIX Security Symposium.* Washington, DC, August 2001. http://citeseer.nj.nec.com/kornievskaia01kerberized.html

43. [Li99] Li, N.H., Grosof, B.N., Feigenbaum, J. A Logic-based Knowledge Representation for Authorization with Delegation. *PCSFW: Proceedings of The 12th IEEE Computer Security Foundations Workshop*, pp 162-174, Italy, June 1999.

44. [Linn99] Linn, J. and Nystrom, M. Attribute Certification: An Enabling Technology for Delegation and Role-Based Controls in Distributed Environments, *Proceedings of 4th ACM Workshop on Role-Based Access Control*, pp 121-130, 1999.

45. [Neuman93] Neuman, B.C. Proxy-Based Authorization and Accounting for Distributed Systems. *Proceedings 13th International Conference on Distributed Computing Systems*, pp 283-291, 1993.

46. [Novotny01] Novotny, J., Tuecke, S. and Welch, V. An Online Credential Repository for the Grid: MyProxy. *Proceedings 10th International Symposium on High*

Performance Distributed Computing, August 2001. http://globus-mirror.hep.man.ac.uk/research/papers/myproxy.pdf

47. [Parker95] Parker, T.A. Single Sign On Systems – The Technologies and the Products. *Proceedings of European Convention on Security and Detection*, IEEE. May 1995

48. [Pearlman02] Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S. A Community Authorization Service for Group Collaboration. *Proceedings IEEE 3rd International Workshop on Policies for Distributed Systems and Networks,* pp 50-59, Monterey, California, June 2002.

49. [Ryutov98] Ryutov, T. and Neuman, C. Access Control Framework for Distributed Applications. *Internet Draft, Internet Engineering Task Force,* Nov. 1998.

50. [Ryutov00] Ryutov, T.V. and Neuman, C. Representation and evaluation of security policies for distributed system services. *Proceedings of the DARPA Information Survivability Conference and Exposition* (DISCEX), Volume 2, pp 1172-1183, Jan. 2000.

51. [Sandhu96] Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E. Role-Based Access Control Models, *IEEE Computer,* 29(2), pp 38-47, Feb. 1996.

52. [Sun03] Sun ONE Directory Server 5.2-Product Brief, June 2003 http://docs.sun.com/coll/S1_DirectoryServer_52

53. [Stoker01] Stoker, G., White, B.S., Stackpole, E., Highley, T.J. and Humphrey, M. Toward Realizable Restricted Delegation in Computational Grids, *European High Performance Computing and Networking*, June 2001. http://legion.virginia.edu/papers/delegation.pdf

54. [Thompson99] Thompson, M., Johnson, W., Mudumbai, S., Hoo, G., Jackson, K. and Essiari, A. Certificate based access control for widely distributed resources. *Proceeding 8th USENIX Security Symposium*, August 1999. http://www-itg.lbl.gov/security/Akenti/Papers/UsenixSec99.pdf

55. [Thompson02a] Thompson, M, Essiari, A., Mudumbai, S. Certificate-based Authorization Policy in a PKI Environment. *Submitted to a special issue of ACM Transactions on Infomation and System Security*, Aug 2002. http://citeseer.nj.nec.com/546062.html

56. [Thompson02b] Thompson, M.R., Olson, D., Cowles, R., Mullen, S., Helm, M. CA-based Trust Model for Grid Authentication and Identity Delegation, *GGF*, Oct. 2002.http://www.globalgridforum.org/Meetings/ggf6/ggf6_wg_papers/IBM/TrustModel-v6c.pdf

57. [Tuecke02a] Tuecke, S., Engert, D., Foster, I., Thompson, M., Pearlman, L. and Kesselman, C. Internet X.509 Public Key Infrastructure Proxy Certificate Profile.

GGF Information Document, draft-ggf-gsi-proxy.04, August 2002. http://www.gridforum.org/security/gsi/draftpggf-gsi-proxy-04.pdf

58. [Tung00] Tung, B., Neuman, C. and Wray, J. Public Key Cryptography for Initial Authentication in Kerberos. *Internet draft*. Apr. 2000.

59. [Welch03] Welch, V., Siebenlist, F., Meder, S., Pearlman, L. Use of SAML for OGSA Authorization. *Global Grid Forum OGSA Security Working Group*, Feb 15, 2003. http://www.globus.org/ogsa/Security

60. [Zeilenga02] K. Zeilenga. K. LDAP: String Representation of Distinguished Names. <*draft-ietf-ldapbis-dn-07.txt*>, March 2002

APPENDIX

We attach all the user files and entries on our LDAP server based on our schema in LDIF file format in this appendix. This file can be used to deploy our user directory service. Each LDIF entry consists of a required distinguished name, object classes, and attribute definitions. The object class defines attribute types which are allowed and required for the entry. The attribute should be defined either in `slapd.at.conf` or in `slapd.conf`. In the LDIF file, each entry is separated by a blank line, and base 64 encoded attribute values are indicated by a `::` after the attribute name.

1. DN: cn=config

```
dn: cn=config
cn: config
objectClass: top
objectClass: extensibleObject
objectClass: nsslapdConfig
nsslapd-accesslog-logging-enabled: on
nsslapd-accesslog: C:/Program Files/Sun/MPS/slapd-kent8/logs/access
nsslapd-accesslog-maxlogsperdir: 10
nsslapd-accesslog-maxlogsize: 100
nsslapd-accesslog-logrotationtime: 1
nsslapd-accesslog-logrotationtimeunit: day
nsslapd-enquote-sup-oc: off
nsslapd-localhost: kent8.galab.uwindsor.ca
nsslapd-schemacheck: on
nsslapd-rewrite-rfc1274: off
nsslapd-return-exact-case: on
nsslapd-port: 389
nsslapd-errorlog: C:/Program Files/Sun/MPS/slapd-kent8/logs/errors
nsslapd-errorlog-logging-enabled: on
nsslapd-errorlog-maxlogsperdir: 2
nsslapd-errorlog-maxlogsize: 100
nsslapd-errorlog-logrotationtime: 1
nsslapd-errorlog-logrotationtimeunit: week
nsslapd-auditlog: C:/Program Files/Sun/MPS/slapd-kent8/logs/audit
nsslapd-rootdn: cn=Directory Manager
nsslapd-groupevalnestlevel: 5
nsslapd-windows-authentication-enabled: off
aci: (targetattr = "*")(version 3.0; acl "Configuration Administrators Group";
  allow (all) groupdn = "ldap:///cn=Configuration Administrators, ou=Groups,
 ou=TopologyManagement, o=NetscapeRoot";)
aci: (targetattr = "*")(version 3.0; acl "Configuration Administrator"; allow
 (all) userdn = "ldap:///uid=admin,ou=Administrators, ou=TopologyManagement,
 o=NetscapeRoot";)
aci: (targetattr = "*")(version 3.0; acl "Local Directory Administrators Group
 "; allow (all) groupdn = "ldap:///cn=Directory Administrators, dc=galab,dc=u
 windsor,dc=ca";)
```

```
aci: (targetattr = "*")(version 3.0; acl "SIE Group"; allow (all)groupdn = "ld
 ap:///cn=slapd-kent8, cn=Sun ONE Directory Server, cn=Server Group, cn=kent8
 .galab.uwindsor.ca, ou=galab.uwindsor.ca, o=NetscapeRoot";)
ds-verify-plugin-signature: off
ds-require-valid-plugin-signature: on
modifiersName: cn=directory manager
modifyTimestamp: 20040301002351Z
nsslapd-rootpw: {SSHA}XfPKd5ReC+GWw0cY70AZBDMZBQSWHtVZZ22KeA==
numSubordinates: 11
```

2. DN: cn=schema

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
aci: (target="ldap:///cn=schema")(targetattr !="aci")(version 3.0;acl "anonymo
 us, no acis"; allow (read, search, compare) userdn = "ldap:///anyone";)
aci: (targetattr = "*")(version 3.0; acl "Configuration Administrators Group";
  allow (all) groupdn = "ldap:///cn=Configuration Administrators, ou=Groups,
 ou=TopologyManagement, o=NetscapeRoot";)
aci: (targetattr = "*")(version 3.0; acl "Configuration Administrator"; allow
 (all) userdn = "ldap:///uid=admin,ou=Administrators, ou=TopologyManagement,
 o=NetscapeRoot";)
aci: (targetattr = "*")(version 3.0; acl "Local Directory Administrators Group
 "; allow (all) groupdn = "ldap:///cn=Directory Administrators, dc=galab,dc=u
 windsor,dc=ca";)
aci: (targetattr = "*")(version 3.0; acl "SIE Group"; allow (all)groupdn = "ld
 ap:///cn=slapd-kent8, cn=Sun ONE Directory Server, cn=Server Group, cn=kent8
 .galab.uwindsor.ca, ou=galab.uwindsor.ca, o=NetscapeRoot";)
aci: (targetattr = "*") (version 3.0;acl "anonymous access";allow (all)(userdn
  = "ldap:///anyone");)
modifiersName: uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoo
 t
modifyTimestamp: 20040309073809Z
attributeTypes: ( 2.5.4.58 NAME 'attributeCertificateAttribute'  SYNTAX 1.3.6.
 1.4.1.1466.115.121.1.5 X-ORIGIN 'user defined' )
objectClasses: ( 2.5.6.24 NAME 'pmiUser' SUP top STRUCTURAL MAY attributeCerti
 ficateAttribute X-ORIGIN 'user defined' )
nsSchemaCSN: 4047ede0000000000000
```

3. suffix: c=ca

```
version: 1

# entry-id: 1
dn: c=ca
c: ca
objectClass: top
objectClass: country
creatorsName: cn=directory manager
createTimestamp: 20040303091623Z
nsUniqueId: 6bde1981-1dd211b2-805a83d5-d7fb4a61
aci: (targetattr != "userPassword") (version 3.0; acl "Anonymous access"; allo
 w (read, write, search, compare)userdn = "ldap:///anyone";)
aci: (targetattr != "nsroledn||aci")(version 3.0; acl "Allow self entry modifi
 cation except for nsroledn and aci attributes"; allow (write)userdn ="ldap:/
 //self";)
aci: (targetattr = "*")(version 3.0; acl "Configuration Administrator"; allow
 (all) userdn = "ldap:///uid=admin,ou=Administrators, ou=TopologyManagement,
 o=NetscapeRoot";)
aci: (targetattr ="*")(version 3.0;acl "Configuration Administrators Group";al
 low (all) (groupdn = "ldap:///cn=Configuration Administrators, ou=Groups, ou
 =TopologyManagement, o=NetscapeRoot");)
aci: (targetattr = "*")(version 3.0; acl "SIE Group"; allow (all)groupdn = "ld
 ap:///c=ca";)
modifiersName: cn=directory manager
modifyTimestamp: 20040312203709Z
```

4. suffix: o=CodeNet, c=ca

```
version: 1

# entry-id: 1
dn: o=CodeNet,c=ca
o: CodeNet
objectClass: top
objectClass: organization
creatorsName: cn=directory manager
modifiersName: cn=directory manager
createTimestamp: 20040312202313Z
modifyTimestamp: 20040312202313Z
nsUniqueId: f399f582-1dd111b2-80ca83d5-d7fb4a61

# entry-id: 2
dn: cn=administrator,o=CodeNet,c=ca
sn: administrator
cn: administrator
creatorsName: cn=directory manager
createTimestamp: 20040312202448Z
nsUniqueId: 3b208181-1dd211b2-80ca83d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBeTCB4zA9oTukOTA3MQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEWMBQGA1UEAxMNYWRtaW5pc3RyYXRvcjA3pDUwMzELMAkGA1UEBhMCQ0ExE
 DAOBgNVBAoTB0NvZGVOZXQxEjAQBgNVBAMTCXJvb3RhZG1pbjANBgkqhkiG9w0BAQQFAAIDBcv8M
 DIYFzIwMDQwMzAxMDAwMDAwLjAwMC0wNTAwGBcyMDA1MDMwMTAwMDAwMC4wMDAtMDUwMDAfMB0GC
 isGAQQBgZhxAQExDxMNYWRtaW5pc3RyYXRvcjAAMA0GCSqGSIb3DQEBBAUAA4GBAEIC5BMmLvBDi
 ZGM76CXTxHqKvfykxIW3BFqd/noQQfVdmycXsmkhn4vXksfz/k0PHr38bfLozTvlFtcPtbRBFIqi
 bJY/rvqRclS50M7yFnn7EfhA4v+7J10U+MZP9vdnE0cmMINvNn4hLq9ivG7N4LRS6+lp+AYGiHID
 2ZGJ8b3
modifiersName: cn=directory manager
modifyTimestamp: 20040329065459Z

# entry-id: 3
dn: cn=user0,o=CodeNet,c=ca
sn: user0
cn: user0
creatorsName: cn=directory manager
createTimestamp: 20040312205931Z
nsUniqueId: 1ed31381-1dd211b2-80cf83d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBMjCB3TA1oTOkMTAvMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEOMAwGA1UEAxMFVXNlcjAwN6Q1MDMxCzAJBgNVBAYTAkNBMRAwDgYDVQQKE
 wdDb2RlTmV0MRIwEAYDVQQDEwlDb2RlTmV0QUEwDQYJKoZIhvcNAQEEBQACAwXL/DAyGBcyMDA0M
 DMwMTAwMDAwMC4wMDAtMDUwMBgXMjAwNDA2MDEwMDAwMDAuMDAwLTA1MDAwITAfBgorBgEEAYGYc
 QEBMRETD3JlZ2lzdGVyZWQgdXNlcjAAMA0GCSqGSIb3DQEBBAUAA0EAlB8tJk6BWlV2HSNTeEMCs
 nzGajcZNqXTte71jNIWGYaLtvwSkbQRRr1hVc3EM1UfwiXZttU8Gecht/eHqS7zvw==
modifiersName: cn=directory manager
modifyTimestamp: 20040403082119Z

# entry-id: 4
dn: cn=user8,o=CodeNet,c=ca
sn: user8
cn: user8
creatorsName: cn=directory manager
createTimestamp: 20040312210121Z
nsUniqueId: 66599f81-1dd211b2-80cf83d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBLzCB2jA1oTOkMTAvMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEOMAwGA1UEAxMFVXNlcjgwN6Q1MDMxCzAJBgNVBAYTAkNBMRAwDgYDVQQKE
 wdDb2RlTmV0MRIwEAYDVQQDEwlDb2RlTmV0QUEwDQYJKoZIhvcNAQEEBQACAwXMBDAyGBcyMDA0M
 DMwMTAwMDAwMC4wMDAtMDUwMBgXMjAwNDA1MDEwMDAwMDAuMDAwLTA1MDAwHjAcBgorBgEEAYGYc
 QEBMQ4TDGdlbmVyYWwgdXNlcjAAMA0GCSqGSIb3DQEBBAUAA0EAipLG8xwRQtnLslt7w5U4VPcZx
 ggxwzpb4/Jasw2Mg6iUvZr/Jm6tmxXxkIOHuYi/izyKfCV3tgFkm4mQy2q6Og==
modifiersName: cn=directory manager
modifyTimestamp: 20040403082634Z

# entry-id: 5
dn: ou=ou1, o=CodeNet, c=ca
postalCode: N9B 3P4
objectClass: top
objectClass: organizationalUnit
street: 401 sunset Ave.
ou: ou1
creatorsName: cn=directory manager
modifiersName: cn=directory manager
createTimestamp: 20040315091556Z
modifyTimestamp: 20040315091556Z
nsUniqueId: 450fd381-1dd211b2-80c883d5-d7fb4a61

# entry-id: 6
dn: cn=CodeNetAA,o=CodeNet,c=ca
sn: CodeNetAA
cn: CodeNetAA
creatorsName: cn=directory manager
createTimestamp: 20040403091631Z
nsUniqueId: 923fe081-1dd211b2-801383d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIILsDCCC1owOaE3pDUwMzELMAkGA1UEBhMCQ0ExEDAOB
 gNVBAoTB0NvZGVOZXQxEjAQBgNVBAMTCUNvZGVOZXRBQTA3pDUwMzELMAkGA1UEBhMCQ0ExEDAOB
 gNVBAoTB0NvZGVOZXQxEjAQBgNVBAMTCUNvZGVOZXRBQTANBgkqhkiG9w0BAQQFAAIDOfg8MDIYF
 zIwMDQwMzAxMDAwMDAwLjAwMC0wNTAwGBcyMDA1MDMwMTAwMDAwMC4wMDAtMDUwMDCCCpgwggqUB
 gwqhjoAAYHMkyoBAQ0xggqCE4IKfjw/eG1sIHZlcnNpb249IjEuMCIgZW5jb2Rpbmc9IlVURi04I
 j8+DQo8IURPQ1RZUEUgWC41MDlfUE1JX1JCQUNfUG9saWN5Pg0KPCEtLSBIaW50OiBkb24ndCBwd
 XQgU1lTVEVNICJmaWxlbmFtZS5kdGQiIGluIHRoZSAhRE9DVFlQRSB0YWcgLS0+DQoNCjwhLS0gD
 QoNClVzZXIwOiBDTj1Vc2VyMCxPPUNvZGVOZXQsQz1DQQ0KCUNvZGVOZXQgUm9sZTpSb2xlMA0KD
 QpVc2VyODogQ049VXNlcjgsTz1Db2RlTmV0LEM9Q0EgDQoJQ29kZU5ldCBSb2xlOlJvbGUxDQoNC
 lRhcmdldDA6IENOPVRhcmdldDAsTz1Db2RlTmV0LEM9Q0ENClRhcmdldDE6IENOPVRhcmdldDEsT
 z1Db2RlTmV0LEM9Q0ENCg0KU09BMDogQ049Q29kZU5ldEFBLE89Q29kZU5ldCxDPUNBDQoNCkFjd
 GlvbjAgY2FuIGJlIGV4ZWN1dGVkIG9uIGFueSB0YXJnZXQgYnkgUm9sZTAsIA0KQWN0aW9uMSBjY
 W4gYmUgZXhlY3V0ZWQgb24gYW55IHRhcmdldCBieSBSb2xlMSwgDQpBY3Rpb24zIGNhbiBiZSBle
 GVjdXRlZCBvbiBhbnkgdGFyZ2V0IGJ5IGFueSB2YWxpZCB1c2VyIChubyByb2xlcyBhcmUgcmVxd
 WlyZWQ7DQoJdGhlIHVzZXIganVzdCBtdXN0IG1hdGNoIHRoZSBTdWJqZWN0RG9tYWluOiBPPUNvZ
 GVOZXQsQz1DQSkNCg0KQWN0aW9uMiBkb2VzIG5vdCBleGlzdDogeW91IHNob3VsZCBnZXQgYW4gZ
 XJyb3IgbWVzc2FnZSBzYXlpbmcgdGhhdA0KDQotLT4NCg0KPFguNTA5X1BNSV9SQkFDX1BvbGlje
 SBPSUQ9IjEuMy42LjEuNC4xLjE5NTY5LjEuMiI+DQo8IS0tIHRlc3RQb2xpY2llcy4zIC0tPg0KD
 QogIDxTdWJqZWN0UG9saWN5Pg0KICAgIDxTdWJqZWN0RG9tYWluU3BlYyBJRD0iU3ViamVjdERvb
 WFpbiI+DQogICAgICA8SW5jbHVkZSBMREFQRE49Ik89Q29kZU5ldCxDPUNBIi8+DQogICAgPC9Td
 WJqZWN0RG9tYWluU3BlYz4NCiAgPC9TdWJqZWN0UG9saWN5Pg0KDQogIDxSb2xlSGllcmFyY2h5U
 G9saWN5Pg0KPCEtLSAxIHJvbGUsIENvZGVOZXQgUm9sZSAtLT4NCiAgICA8Um9sZVNwZWMgT0lEP
 SIxLjMuNi4xLjQuMS4xOTU2OS4xLjEiIFR5cGU9IkNvZGVOZXQgUm9sZSI+DQogICAgICA8U3VwU
 m9sZSBWYWx1ZT0iUm9sZTAiLz4NCiAgICAgIDxTdXBSb2xlIFZhbHVlPSJSb2xlMSIvPg0KICAgI
 DwvUm9sZVNwZWM+DQogIDwvUm9sZUhpZXJhcmNoeVBvbGljeT4NCg0KICA8U09BUG9saWN5Pg0KI
 CAgIDxTT0FTcGVjIElEPSJUaGVTT0EiIExEQVBETj0iQ049Q29kZU5ldEFBLE89Q29kZU5ldCxDP
 UNBIi8+DQogIDwvU09BUG9saWN5Pg0KDQogIDxSb2xlQXNzaWdubWVudFBvbGljeT4NCiAgICA8U
 m9sZUFzc2lnbm1lbnQ+DQogICAgICA8U3ViamVjdERvbWFpbiBJRD0iU3ViamVjdERvbWFpbiIvP
 g0KICAgICAgPFJvbGVMaXN0Pg0KICAgICAgICA8Um9sZSBUeXBlPSJDb2RlTmV0IFJvbGUiIFZhb
 HVlPSIiLz4NCiAgICAgIDwvUm9sZUxpc3Q+DQogICAgICA8RGVsZWdhdGUgRGVwdGg9IjAiLz4NC
 iAgICAgIDxTT0EgSUQ9IlRoZVNPQSIvPg0KICAgICAgPFZhbGlkaXR5Lz4NCiAgICA8L1JvbGVBc
 3NpZ25tZW50Pg0KICA8L1JvbGVBc3NpZ25tZW50UG9saWN5Pg0KDQogIDxUYXJnZXRQb2xpY3k+D
 QogICAgPFRhcmdldERvbWFpblNwZWMgSUQ9IlRhcmdldERvbWFpbiI+DQogICAgICA8SW5jbHVkZ
 SBMREFQRE49Ik89Q29kZU5ldCxDPUNBIi8+DQogICAgPC9UYXJnZXREb21haW5TcGVjPg0KICA8L
 1RhcmdldFBvbGljeT4NCg0KICA8QWN0aW9uUG9saWN5Pg0KICAgIDxBY3Rpb24gQXJncz0iIiBOY
 W1lPSJBY3Rpb24wIi8+DQogICAgPEFjdGlvbiBBcmdzPSIiIE5hbWU9IkFjdGlvbjEiLz4NCiAgI
 CA8QWN0aW9uIEFyZ3M9IiIgTmFtZT0iQWN0aW9uMyIvPg0KICA8L0FjdGlvblBvbGljeT4NCg0KI
 CA8VGFyZ2V0QWNjZXNzUG9saWN5Pg0KICAgIDxUYXJnZXRBY2Nlc3M+DQogICAgICA8Um9sZUxpc
 3Q+DQogICAgICAgIDxSb2xlIFR5cGU9IkNvZGVOZXQgUm9sZSIgVmFsdWU9IlJvbGUwIi8+DQogI
 CAgICA8L1JvbGVMaXN0Pg0KDQogICAgICA8VGFyZ2V0TGlzdD4NCiAgICAgICAgPFRhcmdldCBBY
 3Rpb25zPSJBY3Rpb24wIj4NCiAgICAgICAgICA8VGFyZ2V0RG9tYWluIElEPSJUYXJnZXREb21ha
 W4iLz4NCiAgICAgICAgPC9UYXJnZXQ+DQogICAgICA8L1RhcmdldExpc3Q+DQogICAgPC9UYXJnZ
 XRBY2Nlc3M+DQoNCiAgICA8VGFyZ2V0QWNjZXNzPg0KICAgICAgPFJvbGVMaXN0Pg0KICAgICAgI
 CA8Um9sZSBUeXBlPSJDb2RlTmV0IFJvbGUiIFZhbHVlPSJSb2xlMSIvPg0KICAgICAgPC9Sb2xlT
 GlzdD4NCg0KICAgICAgPFRhcmdldExpc3Q+DQogICAgICAgIDxUYXJnZXQgQWN0aW9ucz0iQWN0a
 W9uMSI+DQogICAgICAgICAgPFRhcmdldERvbWFpbiBJRD0iVGFyZ2V0RG9tYWluIi8+DQogICAgI
 CAgIDwvVGFyZ2V0Pg0KICAgICAgPC9UYXJnZXRMaXN0Pg0KICAgIDwvVGFyZ2V0QWNjZXNzPg0KD
 QogICAgPFRhcmdldEFjY2Vzcz4NCiAgICAgIDxSb2xlTGlzdCAvPg0KICAgICAgPFRhcmdldExpc
 3Q+DQogICAgICAgIDxUYXJnZXQgQWN0aW9ucz0iQWN0aW9uMyI+DQogICAgICAgICAgPFRhcmdld
 ERvbWFpbiBJRD0iVGFyZ2V0RG9tYWluIi8+DQogICAgICAgIDwvVGFyZ2V0Pg0KICAgICAgPC9UY
 XJnZXRMaXN0Pg0KICAgIDwvVGFyZ2V0QWNjZXNzPg0KICA8L1RhcmdldEFjY2Vzc1BvbGljeT4NC
 g0KPC9YLjUwOV9QTUlfUkJBQ19Qb2xpY3k+DQowADANBgkqhkiG9w0BAQQFAANBABg0bDLqqNstz
 pTeF7/FnPTZ21cgltw9oz91+7vYlCebdGYRJkgUjWdOCmNLbuI8AQ2clH5VCQ06E04AnIMij2I=
```

```
modifiersName: cn=directory manager
modifyTimestamp: 20040403093753Z
```

5. Suffix: ou=maintenance,o=CodeNet, c=ca

```
version: 1

# entry-id: 1
dn: ou=maintenance,o=CodeNet,c=ca
ou: maintenance
objectClass: top
objectClass: organizationalunit
creatorsName: cn=directory manager
modifiersName: cn=directory manager
createTimestamp: 20040312202350Z
modifyTimestamp: 20040312202350Z
nsUniqueId: 175d3b81-1dd211b2-80ca83d5-d7fb4a61

# entry-id: 2
dn: cn=user1,ou=maintenance,o=CodeNet,c=ca
sn: user1
cn: user1
creatorsName: cn=directory manager
```

```
createTimestamp: 20040312203953Z
nsUniqueId: 53919b81-1dd211b2-80cc83d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBRDCB7zBLoUmkRzBFMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEUMBIGA1UECxMLTWFpbnRlbmFuY2UxDjAMBgNVBAMTBVVzZXIxMDekNTAzM
 QswCQYDVQQGEwJDQTEQMA4GA1UEChMHQ29kZU5ldDESMBAGA1UEAxMJQ29kZU5ldEFBMA0GCSqGS
 Ib3DQEBBAUAAgMFy/0wMhgXMjAwNDAzMDEwMDAwMDAuMDAwLTA1MDAYFzIwMDUwMzAxMDAwMDAwL
 jAwMC0wNTAwMB0wGwYKKwYBBAGBmHEBATENEwtzeXN0ZW0gdXNlcjAAMA0GCSqGSIb3DQEBBAUAA
 0EAYD2HafWwWMhQtF4Jh8Ti1SW02KaP3PwQ5I1AYu7nkpsRawP/j27siJ21CfddeKNRAjxUlkAGY
 2r4dcfaI/+Kvw==
modifiersName: cn=directory manager
modifyTimestamp: 20040406075857Z

# entry-id: 3
dn: cn=user2,ou=maintenance,o=CodeNet,c=ca
sn: user2
cn: user2
creatorsName: cn=directory manager
createTimestamp: 20040312204102Z
nsUniqueId: 7754e181-1dd211b2-80cc83d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBRDCB7zBLoUmkRzBFMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEUMBIGA1UECxMLTWFpbnRlbmFuY2UxDjAMBgNVBAMTBVVzZXIyMDekNTAzM
 QswCQYDVQQGEwJDQTEQMA4GA1UEChMHQ29kZU5ldDESMBAGA1UEAxMJQ29kZU5ldEFBMA0GCSqGS
 Ib3DQEBBAUAAgMFy/4wMhgXMjAwNDAzMDEwMDAwMDAuMDAwLTA1MDAYFzIwMDUwMzAxMDAwMDAwL
 jAwMC0wNTAwMB0wGwYKKwYBBAGBmHEBATENEwtzeXN0ZW0gdXNlcjAAMA0GCSqGSIb3DQEBBAUAA
 0EATupy1zHDRlMLieDcKkn43UY5zsmFBU1tM64Md1Tu8Lv6LwnU4oqTEI3+Y7RnSEf1SAdCbrUsE
 6mLeJs9RxwyRA==
modifiersName: cn=directory manager
modifyTimestamp: 20040406075957Z

# entry-id: 4
dn: cn=user3,ou=maintenance,o=CodeNet,c=ca
sn: user3
cn: user3
creatorsName: cn=directory manager
createTimestamp: 20040312204209Z
nsUniqueId: 9b182781-1dd111b2-80cd83d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBRDCB7zBLoUmkRzBFMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEUMBIGA1UECxMLTWFpbnRlbmFuY2UxDjAMBgNVBAMTBVVzZXIzMDekNTAzM
 QswCQYDVQQGEwJDQTEQMA4GA1UEChMHQ29kZU5ldDESMBAGA1UEAxMJQ29kZU5ldEFBMA0GCSqGS
 Ib3DQEBBAUAAgMFy/8wMhgXMjAwNDAzMDEwMDAwMDAuMDAwLTA1MDAYFzIwMDUwMzAxMDAwMDAwL
 jAwMC0wNTAwMB0wGwYKKwYBBAGBmHEBATENEwtzeXN0ZW0gdXNlcjAAMA0GCSqGSIb3DQEBBAUAA
 0EAN9NEWI+Vbrg5dFeTMq43/Comqf/AdXCHiZpqPXfT2ZPp62AZhqT8LPZC7oshSALNvO9uwdExX
 7/jMKzUKC0zoQ==
modifiersName: cn=directory manager
modifyTimestamp: 20040406080036Z
```

6. Suffix: ou=application,o=CodeNet, c=ca

```
version: 1
```

```
# entry-id: 1
dn: ou=application,o=CodeNet,c=ca
ou: application
objectClass: top
objectClass: organizationalunit
creatorsName: cn=directory manager
modifiersName: cn=directory manager
createTimestamp: 20040312202401Z
modifyTimestamp: 20040312202401Z
nsUniqueId: 175d3b82-1dd211b2-80ca83d5-d7fb4a61

# entry-id: 2
dn: cn=user4,ou=application,o=CodeNet,c=ca
sn: user4
cn: user4
creatorsName: cn=directory manager
createTimestamp: 20040312204427Z
nsUniqueId: 0661f981-1dd211b2-80cd83d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBcDCB2jA1oTOkMTAvMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEOMAwGA1UEAxMFVXNlcjQwN6Q1MDMxCzAJBgNVBAYTAkNBMRAwDgYDVQQKE
 wdDb2RlTmV0MRIwEAYDVQQDEwlyb290YWRtaW4wDQYJKoZIhvcNAQEEBQACAwXMADAyGBcyMDA0M
 DMwMTAwMDAwMC4wMDAtMDUwMBgXMjAwNDA2MDEwMDAwMDAuMDAwLTA1MDAwHjAcBgorBgEEAYGYc
 QEBMQ4TDGdlbmVyYWwgdXNlcjAAMA0GCSqGSIb3DQEBBAUAA4GBAGqella15yAhvstMZWul0DkXv
 U6X9YPrravZs6H60WvMukn23Og/dZvjJ7y2NsccK7aZ3tXjFjzEIIIpKUEPiJleMs8Gfgm5UaHIS
 Hw95pNXZRicsjVjT8xnx57EEfI/tD+D00ngH/Fi3+uxc0i68nnCqNQaIEgZ+naHBUYttHR7
attributeCertificateAttribute:: MIIBSDCB8zBLoUmkRzBFMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEUMBIGA1UECxMLQXBwbGljYXRpb24xDjAMBgNVBAMTBVVzZXI0MDekNTAzM
 QswCQYDVQQGEwJDQTEQMA4GA1UEChMHQ29kZU5ldDESMBAGA1UEAxMJQ29kZU5ldEFBMA0GCSqGS
 Ib3DQEBBAUAAgMFzAAwMhgXMjAwNDAzMDEwMDAwMDAuMDAwLTA1MDAYFzIwMDUwMzAxMDAwMDAwL
 jAwMC0wNTAwMCEwHwYKKwYBBAGBmHEBATEREw9yZWdpc3RlcmVkIHVzZXIwADANBgkqhkiG9w0BA
 QQFAANBAIm3/e2QVvsaHWbLHtPGByvq+X1NQ5Lx4Kd0m3Pwipb5j1x7LQIMZHpJTeoPadJfzGpVM
 y/RgZgyJC2NOXeQ5T8=
modifiersName: cn=directory manager
modifyTimestamp: 20040403084854Z

# entry-id: 3
dn: cn=user5,ou=application,o=CodeNet,c=ca
sn: user5
cn: user5
creatorsName: cn=directory manager
createTimestamp: 20040315082338Z
nsUniqueId: 01659b81-1dd211b2-80c183d5-d7fb4a61
objectClass: organizationalperson
objectClass: person
objectClass: top
objectClass: pmiuser
attributeCertificateAttribute:: MIIBSDCB8zBLoUmkRzBFMQswCQYDVQQGEwJDQTEQMA4GA
 1UEChMHQ29kZU5ldDEUMBIGA1UECxMLQXBwbGljYXRpb24xDjAMBgNVBAMTBVVzZXI1MDekNTAzM
 QswCQYDVQQGEwJDQTEQMA4GA1UEChMHQ29kZU5ldDESMBAGA1UEAxMJQ29kZU5ldEFBMA0GCSqGS
 Ib3DQEBBAUAAgMFzAEwMhgXMjAwNDAzMDEwMDAwMDAuMDAwLTA1MDAYFzIwMDQwOTAxMDAwMDAwL
 jAwMC0wNTAwMCEwHwYKKwYBBAGBmHEBATEREw9yZWdpc3RlcmVkIHVzZXIwADANBgkqhkiG9w0BA
 QQFAANBAGOr7U+IT1Z4GpJcHL/yagJPYFAjdWverrwTok9ljhAkwG6D9fUGFmRMYSuaYpkzS8+Se
 GUaIQHjluqzyaugMSM=
modifiersName: cn=directory manager
modifyTimestamp: 20040406080545Z
```

VITA AUCTORIS

Hao Chen was born in P. R. China, 1972. She obtained the Bachelor's degree in Computer and Electronics Engineering from Jilin Institute of Technology, China, in 1993. She is currently a candidate for the Master's degree in Computer Science at the University of Windsor and hopes to graduate in the Winter 2004.