ELSEVIER

Computer Networks 38 (2002) 675–692

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Applying lightweight directory access protocol service on session certification authority

Yi-Shiung Yeh*, Wei-Shen Lai, Chung-Jaye Cheng

Institute of Computer Science and Information Engineering, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC

Received 25 January 2001; received in revised form 22 August 2001; accepted 28 August 2001

Responsible Editor: M. Hamdi

Abstract

Lightweight Directory Access Protocol (LDAP) service is a new technology being applied on the Internet. On large-scale network systems using Transmission control protocol (TCP)/Internet protocol (IP), there is no standard suggested for single directory—certainly without one to be routinely used on the scale of intranets. LDAP service has many great features, such as providing quick and advanced search, quick response and hierarchy view of data. It also can be utilized to many different applications.

Certification Authority (CA) is a trusted system, and it plays an important role just like a notary bridging between end-entities and helps end-entities to establish a secure environment. If someone wants to trade or communicate with others, he or she needs the certificate issued by the CA to help him or her get the trust from others. When a number of end-entities need this service, the load of CA may become huge. Using distributed CAs may sound like a good idea, but it costs too much. In this paper, we have designed a Session CA using a directory system to share its load without the necessity to maintain the Certificate Revocation List (CRL) because the lifetime of the attribute certificate is very short.

With these great features of LDAP service mentioned above, it becomes desirable that we can apply them to design a new CA system. By using LDAP service, we can reduce the load of certification significantly between CA and end-entity. In addition, this new technology can reduce the maintenance work of administration and improve the efficiency of our new proposed CA. Furthermore, combining with Role-Based Access Control (RBAC) and attribute certificate, the security of our system is greatly improved.

Keywords: Lightweight directory access protocol; Certification authority; Certificate revocation list; Role-based access control; Attribute certificate

1. Introduction

Due to World Wide Web's popularity, it is important to provide good security measures to protect end-entity's interest in terms of preventing hackers from impersonating others or forging important data. Public-key infrastructure (PKI) is

* Corresponding author. Tel.: +886-3-5731813; fax: +886-3-5724176.

E-mail addresses: ysyeh@csie.nctu.edu.tw (Y.-S. Yeh), wslai@csie.nctu.edu.tw (W.-S. Lai), zjchen@csie.nctu.edu.tw (C.-J. Cheng).

1389-1286/01/$ - see front matter
PII: S1389-1286(01)00282-1

a crucial technology for enabling security on the Internet. To support PKI, a lot of technologies have been used. The most important one has been X.509 certificate. The purpose of this certificate is to bind the end-entity over to its public key. Using X.509 certificate issued by trusted Certification Authority (CA), the other party can be trusted after the certificate is verified to be valid. New technologies, such as Secure Socket Layer (SSL), is also using certificate to make the connection between the Web server and browser more secure.

Lightweight Directory Access Protocol (LDAP) service is a new technology on the Internet. But the original idea is that LDAP server can store data and retrieve them later on, much the same as a database. In addition, it has many other advantages such as quick and advanced search, quick response, easy maintenance and a hierarchy view of data. Besides, LDAP server can be used in a large-scale network system over Transmission control protocol (TCP)/Internet protocol (IP) as well as in a distributed system. Nowadays, CA is capable of dealing with many things (such as certificate revocation list (CRL) maintenance, certificate issuing, certificate verifying, etc.), but it needs a good method to help handle hundreds of requests with the hope that the database can share some of the CA's responsibilities. From the advantages described above, the LDAP server meets all the requirements needed for supporting CA. In this paper, we shall discuss the implementation of LDAP service and use all its advantages to design an efficient CA system. Furthermore, we shall explore other good applications of LDAP service.

2. Related technologies

There are some technologies related to this paper [10,11]. The following is a brief introduction to them.

2.1. Public-key cryptosystem

The basic idea of the public-key cryptosystem is that it might be possible to find a cryptosystem where it is computationally infeasible to determine the private key d_k if a public key e_k is given [1]. And we can publish the public key e_k on the directory. Diffie and Hellman proposed this idea in 1976. In fact, Rivest, Shamir, and Adleman realized the first public-key cryptosystem in 1977. They designed the well-known RSA cryptosystem. However, session key is still used in this paper (some random key, and it is used for just one time for symmetric cipher) instead of public key. Public-key cryptosystem can also be applied to digital signature, digital envelope, and other related technologies in establishing secure environment. Now it is suggested that 1024 bits be used because 512 bits is considered not secure enough.

2.2. Public-key certificate (X.509)

A public-key certificate is issued by a CA. It is generally used to verify the identity of the end-entity because it is certificated by a trusted CA.

Whenever we want to communicate with others, a copy of public key of recipient is needed. Then, we need to verify the identity of the other party. A notary is what we need to help both ends to verify the identity for each other.

The X.509 standard was published in 1988, and International Engineering Task Force (IETF) [2] adopted it in a short time. Now, it becomes the most widely used data format of public-key certificate. The following is a general data format of X.509 certificate, as shown in Fig. 1.

In X.509 version 3, we can use extension fields for some special purposes because it is truly an open standard. These extension fields can be used to define our extension types to meet our particular needs.

With this powerful concept of certificate, we can enable additional information, such as attribute information, to be kept in the extension field of a certificate. And this is what we call an attribute certificate. We shall discuss it later in this paper.

2.3. Secure socket layer

The Secure socket layer (SSL) protocol is a desirable technology that did a great improvement in the network security [3]. In 1994, Netscape originally developed SSL and it has been universally accepted on the World Wide Web for authenticated

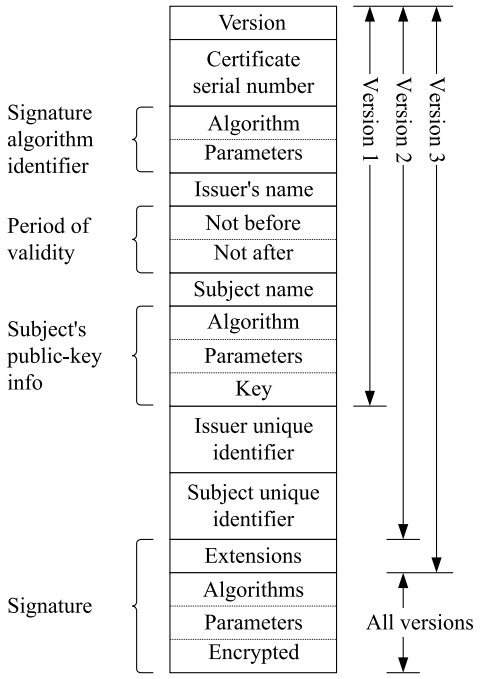


Fig. 1. A data format of X.509 certificate.

and encrypted communication between clients and servers. The SSL protocol operates at the transport layer. Any program that operates at the TCP can work well for SSL. The relationship between server and browser is shown in Fig. 2.

The SSL technology can provide these services as shown in Fig. 2 safely because of SSL's using X.509 certificates. The certificate provides end-

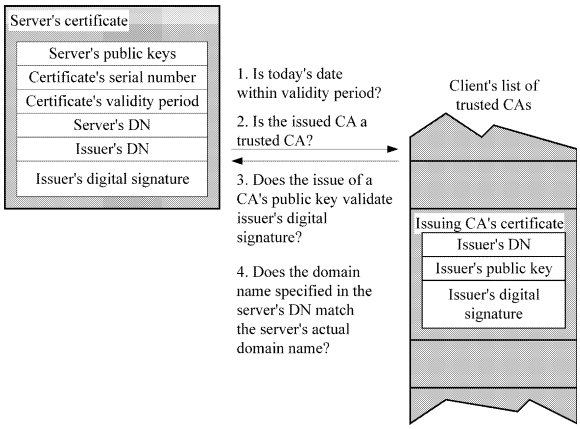


Fig. 2. Illustrations of how a server authenticates a client's certificate.

entities a good way to solve the problem regarding how to authenticate the identity of the end-entity for the Web server.

Although SSL protocol can establish a secure channel between Web server and browsers, it cannot protect the end system from threats if end system does not have enough secure protection. For example, if an end-entity starts using SSL to receive the attributes from the server, this communication is not absolutely secure because the server has no idea of knowing the identity of the end-entity. And if someone intended to impersonate the other end-entity, he could access the server by using forged attributes. However, it is still a good tool that can help us improve the security of the whole network environment whenever it is to be used carefully.

2.4. Certification authority

Certificates work much the same way as any of these familiar forms of identification. And they must have a notary that can help both parties to identify each other. CAs help the end-entities to validate their identities and issue the certificates to their end-entities. But each CA system adopts different policies to validate an identity. For example, the problems that who is issuing the identity and for which purpose it is used may vary with different CA system.

Most importantly, a certificate always includes digital signature of the issued CA. And the CA's digital signature would function as a "letter of introduction" telling end-entities to trust the CA system to exchange their certificates without the necessity of knowing each other.

2.5. Attribute certificate

Attribute certificate is constructed using X.509 format. It is developed by the US financial industry through the ANSI X9 committee [4]. Now, there are a lot of standards concerning attribute certificate and the most advanced one has been used to apply to Transport Layer Security (TLS)—the successor to SSL.

Attribute certificate binds attribute information with the subject name in the certificate. Anyone

who wants to use this certificate can define and register attribute types for his own purpose. An attribute certificate is issued by an attribute certificate authority and managed just the same as X.509 certificate. However, an attribute certificate does not contain a public key. Hence, an attribute certificate needs to be used in conjunction with authentication services in verifying the subject of the attributes.

There are several different categories of information conveyed by using attribute certificate:

(1) *Roles*: Define what an end-entity can do according to what roles it gets to play in this system. Usually, different kinds of end-entities are bound over to different roles. It is up to the system to determine what role the end-entity gets to play. As a result, the end-entity can access to different servers according to his or her privilege given by the system.

(2) *Groups*: A group defines which division the end-entity should belong to. And these groups should be bound over to some rules or limitation such as Role-Based Access Control (RBAC), organizational rules, or other technologies. This helps the system manage its administration affairs.

(3) *Restrictions*: Provide a mechanism for restricting some actions from some attributes. When an end-entity's certificate has these attributes, this end-entity is restricted not to exceed its restrictions. This category is also used to punish end-entities or to lower end-entity's level.

(4) *Access identity*: Provide end-entities who have these attributes in its attribute certificate with a tool to identify themselves for particular server. For instance, we can securely hold our identity by showing the attributes in the certificate to the system after typing the required password in the system that we want to access. If verification is successful, we can access to this system. This increases the security of the system.

2.6. Role-based access control by attribute certificate

RBAC was invented in 1990 [5]. It is a technology designed for managing and enforcing security in large-scale enterprise-wide network system. RBAC uses appropriate roles to combine with end-entities. By using RBAC, large-scale network management becomes easier and clearer.

A role is a management and access control policies for semantic construction. System manager creates the roles. Every end-entity has a role and is bound by this role to follow the rules of the role. Whenever it wants to access the resources in this system, it must authenticate and identify himself. Then the system checks if the role of this end-entity has the permission to access this resource. This approach can simplify the management of the system and increase the security of this system.

RBAC also supports three well-known security policies: data abstraction, least-privilege assignment, and separation of duties. And these are also important to both personal security and management of large-scale enterprise network system.

3. Overview of architecture in lightweight directory access protocol server and session certification authority

3.1. Introduction to directory service

Generally speaking, the information we need to read is far more than we need to write. Similarly, we usually need to query our data more often than we need to modify them. This is why we use directory services. However, different databases cannot share their data because the protocol required to exchange data among one another is too complex and the cost of maintenance is too expensive. So the concept of global directory service is invented.

A service of global directory is to provide a single and centralized repository that supports directory service, which allows any application program to access this service. X.500 is an important method to fulfill this global directory concept by offering an open communication protocol called Directory Access Protocol (DAP) and an extensible information framework to allow the directory to store any kind of information virtu-

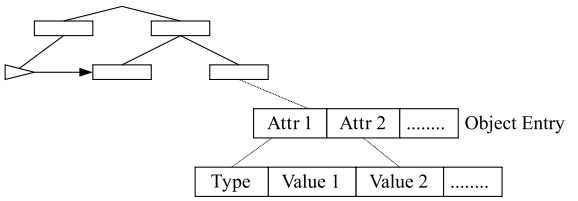


Fig. 3. An example of information model of X.500.

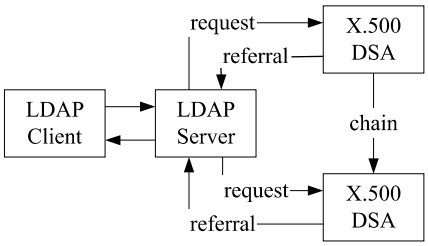


Fig. 4. The relationship between LDAP server and X.500 DSA.

ally. The following is an example of X.500's information model, as shown in Fig. 3.

The naming model defined in X.500 is concerned mainly with the structure of the entries in the namespace, not the way the information is presented to the end-entity. Every entry in a X.500 Directory Information Tree (DIT) is a collection of attributes with each attribute composed of a type element and one or more value elements. Being extensible, X.500 directories may include other objects defined by the end-entities who want to implement them.

3.2. Introduction to lightweight directory access protocol service

Although X.500 provides such a satisfying solution that can be scaled up to millions of end-entities, it still has some disadvantages. One of them is that X.500 depends on a communication layer but itself is not the Internet standard on TCP/IP. The biggest problem is that X.500 is far from ideal to be implemented at the desktop computer, even with today's relatively high-powered hardware. Besides, it also has a big problem regarding directory-naming conventions of which it needs more complicated requirements. Although X.500 offers scalability and robustness, it suffers from more expensive administrative cost.

Because of these drawbacks, a new technology called Lightweight Directory Access Protocol (LDAP) is invented [6,7]. It improves over these disadvantages and become more feasible. Also, it preserves the best quality offered by X.500 standard, while the administrative costs are reduced and the tree of directory is maintained. LDAP service provides an open DAP over TCP/IP and retains X.500's data model. It uses the strings having a minimal subset of Basic Encoding Rules (BER) to simplify an application's requirement for processing directory entries, so it is capable of scaling up to a huge size and millions of entries. For all system administrators, it is the best investment in terms of hardware and network infrastructure. The relationship between LDAP server and X.500 Directory System Agent (DSA) is shown in Fig. 4.

Nevertheless, the information in LDAP server is arranged like X.500 information model. In other words, directory entries are arranged in hierarchical tree-like structure such that political, geographic and/or organizational boundaries are reflected. The entries representing for countries appear on top of the tree. While other entries that are below countries represent states or national organizations. At the bottom of entries as mentioned above there might be other entries representing people, organizational units, printers, documents, or just about anything else you can think of. Fig. 5 shows an example of LDAP

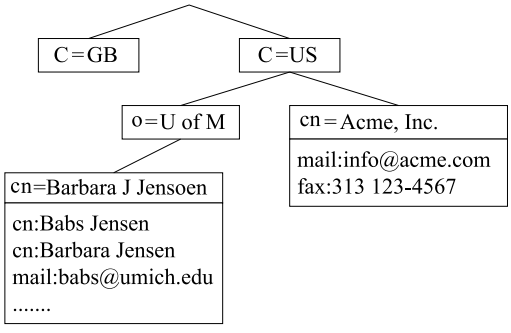


Fig. 5. An example of directory tree of LDAP.

directory tree, which can help us understand LDAP's structure better.

There are many important function models offered by LDAP service.

(1) *Search and read operation*: This function contains reading, searching and listing operation. Reading operation is to be able to read the data that you specify. Searching operation can help end-entity obtain the desirable data by searching the whole database according to the criteria defined by end-entity. Restricting search operation can scope the specific search area to help LDAP server not only reducing the server load but also preventing any ambiguous search.

(2) *Modification operation*: This function contains modification, addition, deletion and modifying Relative Distinguished Name (RDN) operations. Modification operation allows end-entity to modify its secure data. Addition operation can add a new end-entity to the database. Deletion operation can delete a cause after end-entity's privilege being expired or revoke the cancellation resulted from cheating or something else. Modifying RDN operation helps end-entity to modify the associated distinguished name (DN) components.

(3) *Authentication operation*: This helps initiate a session and prove its identity to the directory. Authentication contains bind, unbind and abandon operations, and these operations support several authentication methods, such as simple clear-text password and public-key based authentication. And LDAPv3 can support stronger authentication service.

The built-in extensibility of the LDAP architecture makes it easier to modify the protocol for new situations and to meet end-entities' needs. Therefore, it is convenient to add what we want. In regard to RBAC, we can create attributes like access control lists (ACL) to manage the end-entities and improve the security of the system.

The following is a comparison between DAP and LDAP with respect to query sizes, as shown in Table 1.

It can be seen from Table 1 that query sizes of LDAP are significantly smaller than those of DAP, which is the most important reason for using LDAP server instead of DAP in this work.

Table 1
Comparison between DAP and LDAP with respect to query sizes

Query	DAP (byte)	LDAP (byte)
Unauthenticated bind	192	14
Authenticated bind	409	138
Simple search request	237	105
Simple request	547	355

3.3. End-entity's view of the infrastructure of Session certification authority

There are three main parts divided in our proposed CA system; the first one is Registration Authority (RA), the second one is CA and the last one is LDAP server. Each of them has its own duties and they all have to follow the protocol to communicate. In this section, we shall discuss their duties individually in a rough manner. And then we shall explain the operational protocols in the next section.

The following is the end-entity's view of the infrastructure of our Session CA, as shown in Fig. 6:

(1) *Registration authority*: RA deals with the data from the end-entities. End-entities can use any methods available by delivering their data to RA. Without considering the security measures to protect the data, the information obtained from the end-entities is most likely to be revealed. The most secure method to register is for end-entities to take their individual data and identification to the registration center or to the operator of the

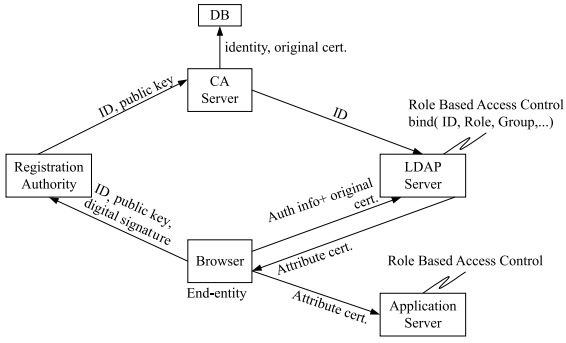


Fig. 6. End-entity's view of infrastructure of Session CA.

RA. After completing the authentication and identification at registration center, the RA will send the registration data to CA. Then CA makes the certificate public. Technically, the administrator will have to take both security and convenience into consideration. For example, regular end-entities' visiting the registration center and delivering a disk is inconvenient and inefficient, but it is more secure. On the other hand, it is much easier if common end-entities can register their data online—but it is not secure. It is really a tradeoff between convenience and security for designers to decide. And the designers should adjust the balance for both factors and make decisions depending on what their purposes are.

(2) *Certification authority*: The role of CA is like a notary. It should be responsible for issuing certificates to end-entities. After receiving registration data from RA (which must be checked by RA successfully), CA uses CA's private key to sign end-entity's registration data and issue a certificate of X.509 format. And then CA sends back this certificate to RA or makes this certificate public. CA must also maintain CRLs. CRL keeps the record of the certificates that are already revoked. The record can be kept for decades if there are arguments related to record. When receiving someone else's certificate, the business owner should check the CRL to see if the certificate is revoked or not. In our proposed Session CA, we do not have to maintain CRLs for attribute certificates, because the lifetime of our attribute certificate is very short.

(3) *Lightweight directory access protocol server*: LDAP server can help CA sharing its load. For example, issuing attribute certificate can be done in a LDAP server. A business owner can use LDAP server to check if the clients who intend to do business with him are on the CRLs or not. And LDAP server can do RBAC with attribute certificate. That is, LDAP server will bind the roles over to the end-entities according to their attributes, and the end-entities should follow their own roles to trade with the business owner. By using attribute certificate and simple authentication, the business owner finds out what privileges the end-entities possess according to the attributes. And there is no need in connecting to the CA or database to check the CRL to see if this certificate is revoked, because the attribute certificate is too short to be maintained.

3.4. Operation protocol of session certification authority

The operation protocol of our proposed Session CA can be divided into two parts. The first one is a registration protocol and the second one is an access protocol. The following are detailed protocols [8]:

3.4.1. Registration protocol

Registration data flow of Session CA is shown in Fig. 7. End-entities must follow this protocol to complete registration. Here are sequential steps of registration protocol.

Step 1: End-entity generates its key pair and saves it in a secured place, and the end-entity sends registration data to RA. The registration data is composed of the identity information (e.g. name or ID), the public key, and digital signature (to be signed on the identity information and public key by using end-entity's private key). In contrast, if the key pair is to be generated by CA or RA, then they (CA or RA) are responsible for storing the key pair and the identity information in database and for managing them. In addition, CA or RA must keep the information securely and safely so it would not be stolen by a scheming person. Most importantly, CA or RA must transmit the key pair to end-entity in a secure fashion. Basically, the

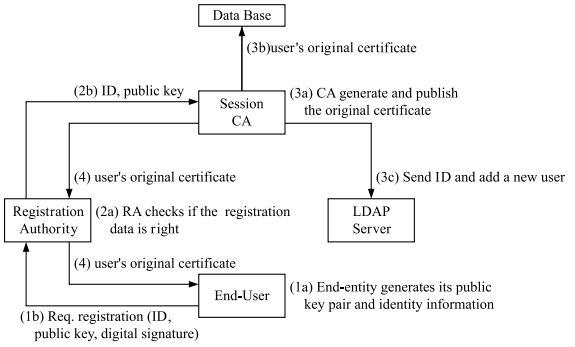


Fig. 7. The flow chart of registration protocol.

end-entity must have confidence in CA or RA, i.e., they will not betray end-entity and reveal the information. So, the generation of key pair by end-entity can prevent these problems just mentioned above from happening and reduce the load of management for CA or RA.

Step 2: After having received the registration data from end-entity, RA has to make sure the following two course of actions are accurate: (1) the claimed identity in registration data is the genuine identity of end-entity, and (2) the digital signature is valid. For course of action number one, there are two methods available as to how the claimed identity can be confirmed by RA. Firstly, end-entity uses face-to-face method to claim the identity. Although this approach is the best way, it is an off-line method. Secondly, if end-entity wants to register by on-line method, the registration procedure must be assisted with other protocols. The Session CA system proposed by this work is to use SSL to help perform on-line registration. However, this method can only guarantee that the information of identity will not be revealed when conducting the transfer of this registration. For course of action number two, if the digital signature is valid, it means that the key pair generated by end-entity is valid (not being counterfeited or mistaken). In other words, the digital signature is signed by end-entity using private key with respect to both identity and public key. After these two course of actions mentioned above are all confirmed, RA will send both identity and public key to CA. And CA will send a name certificate to end-entity.

Step 3: After CA receives the registration data sent securely by RA including identity and public key, CA generates a name certificate for the end-entity. In addition, CA archives this certificate into database, and also sends a copy of end-entity's identity information to LDAP server. Finally, this Session CA system will give authority to end-entity with respect to relative attributes (role or group) in LDAP server.

Step 4: CA must make the name certificate public or send name certificate back to RA. And then RA delivers end-entity's name certificate back to end-entity. After the end-entity receives this certificate, it should be saved into the browser.

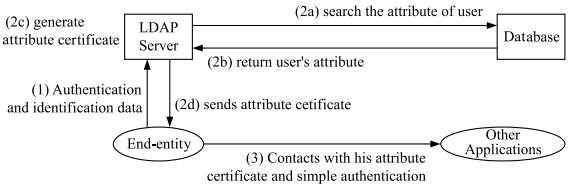


Fig. 8. Accessing protocol of Session CA.

3.4.2. Access protocol

Access protocol is shown in Fig. 8. The following steps are consecutive actions regarding access protocol

Step 1: End-entity uses browser to connect to LDAP server. First, LDAP server checks whether the identity and the password are valid or not, and then it verifies whether the name certificate was issued by a trusted CA. Finally, LDAP server will check to see whether the identity has the same name as the subject name in name certificate.

Step 2: After end-entity's being successfully authenticated by LDAP server, LDAP server will search the database and generate the attribute certificate for end-entity according to the identity from end-entity. The attributes of the end-entity are bound with a proper role (and group). And then LDAP server transfers the attribute certificate back to the end-entity.

Step 3: When an end-entity wants to access an application server, an attribute certificate must be presented to the server. The application server will verify the attribute certificate, and it may ask the end-entity to present additional authentication as well. If the identity of end-entity is confirmed, then application server can offer the access right to end-entity in accordance with end-entity's role in attribute certificate. The role of the end-entity provides end-entity with specific privilege to access to certain resource. In other systems, the application server provides the access right to end-entity based on identity only.

4. Implementation

In this chapter, the implementation of our protocol and the description of data flow are dis-

cussed in detail. We shall describe the on-line version. If an off-line version has to be applied by certain systems, then step 1 will be a face-to-face authentication.

4.1. Data flow of registration protocol

Because this paper is arranged to focus on LDAP's application, we shall discuss the function of LDAP in detail. The detailed data flow is described as below (Fig. 9):

1. *End-entity sends* E_{RA}(*identity*, *public key*, *digital signature*) *to RA*: First, end-entity must generate the key pair (including public key and private key). And then, end-entity may use either SSL or other strategies (e.g., face-to-face) to transmit information to RA. In addition, end-entity has to generate all the necessary data when applying the name certificate and then sends the registration data to RA using SSL—E_{RA}(*identity*, *public key*, *digital signature*). The *digital signature* included in the data is generated by end-entity—applying the digital signature algorithm in conjunction with end-entity's *private key* to process the *identity* and *public key*.

2. *RA does verify*(*identity*, *public key*, *digital signature*): After RA receives the encrypted registration data by SSL, RA uses its private key to decrypt it and get end-entity's *identity*, *public key*, and *digital signature.* Then, RA uses the function *Verify* to check whether the *digital signature* is valid. If the function returns true, it means that the *digital signature* is valid in response to both *identity* information and *public key*. And if the function returns false, it means that either *identity* or *public key* is incorrect.

Additionally, RA must have a proper methodology to check whether the claimed *identity* received from end-entity is really the end-entity (e.g., a face-to-face check).

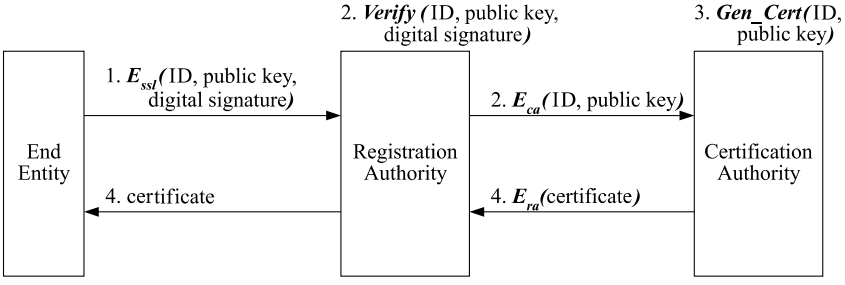
If one of the verifications described above is invalid, the end-entity must repeat the step 1.

3. *RA sends* E_{CA}(*identity*, *public key*) *to CA*, *CA calls Gen_Cert*(*identity*, *public key*): After the verification is completed by RA, RA uses CA's public key to generate the digital envelope to enclose end-entity's *identity* and *public key* and then sends the result to CA. After CA receives the digital envelope, CA uses its private key to open the digital envelope and then gets the end-entity's *identity* and *public key*. Then CA generates the name certificate by using function *Gen_Cert*. Later, the CA makes the certificate public.

4. CA uses RA's public key to generate the digital envelope to enclose end-entity's name certificate—E_{RA}(*certificate*)—and then sends the result to RA.

5. RA sends name certificate of end-entity to end-entity.

6. CA then adds a new end-entity while sending end-entity's identity to LDAP server. In other



$E_{ssl}()$: To encrypt data following the SSL protocol.
$E_{ca}()$: To encrypt data using CA's public key.
$E_{ra}()$: To encrypt data using RA's public key.
Verify(): To verify the user data in accordance with the policy and the key pair.
Gen_Cert(): To generate the certificate in accordance with the user's data and key pair.

Fig. 9. The detailed function of registration protocol.

words, CA can call *add* function to add a new end-entity to database. The following is the data flow of adding a new end-entity to LDAP server:

(i) *Requester calls ldap_init*(*ldap_host*, *ldap_port*): This function needs two parameters, i.e., *ldap_host* and *ldap_port*, which can help the requester initialize a socket and connect with *ldap_host* to establish a link with socket through *ldap_port* of *ldap_host*. In LDAP, *ldap_port* is defined as port 389. After completing the initialization successfully, LDAP server will return back one handle *ld*.

(ii) *Requester calls ldap_simple_bind*(*ld*, *MGR_DN*, *MGR_PW*): The design of this function uses three parameters, *ld*, *MGR_DN*, and *MGR_PW* to do authentication for directory server. *ld* is the handle number of which it was returned by directory server after successfully launching an initialization connected by using *ldap_init*. Since adding new end-entity is part of an administration's work, general end-entities are restricted. Only the manager or the administrator of directory server can add or modify the database. The manager uses the distinguished name of identity and password to authenticate from directory server and get the relevant access right. After successfully completing the authentication, directory server will return back integer of message id called *msgid*.

(iii) *Requester calls ldap_add*(*ld*, *dn*, *mods*): This function uses three parameters, *ld*, *dn*, and *mods* to add a new end-entity to the directory server. The *ld* is the handle number returned by directory server after a successful initialization. The *dn* is the distinguished name of the end-entity, which is an unique name in the database of directory server. The *mods* is the attribute of the data structure filled up by the administrator of directory server according to the policy defined by the management of organization. And then directory server will return back an integer of message id called *msgid*.

(iv) *Requester calls ldap_result*(*ld*, *msgid*, *all*, **timeval*, **result*): This function uses five parameters, *ld*, *msgid*, *all*, *timeval* and *result* to parse the outcome returned previously from the directory server. The *ld* is the handle number that directory server returns after a successful connection. The *msgid* is the message id returned from directory server, representing the LDAP message number. The *all* specifies how the results of the operation return. The *timeval* is a maximum time interval waiting for the selection to complete. The *result* is the outcome of operation. This function will return with an integer to show if there is an error.

(v) *Requester calls ldap_unbind*(*ld*): This function should be used to end our request as we finish our operation. LDAP server can release the system's resources to have been occupied through servicing the connection operation to reduce the load of directory server and to increase the efficiency of resource usage plus memory allocation.

The following is a successful add procedure of adding a new end-entity, as shown in Fig. 10.

4.2. Data flow of access protocol

1. *End-entity sends* E_{ldap}(*identity*, *password*), *name certificate to LDAP server* (Fig. 11): End-

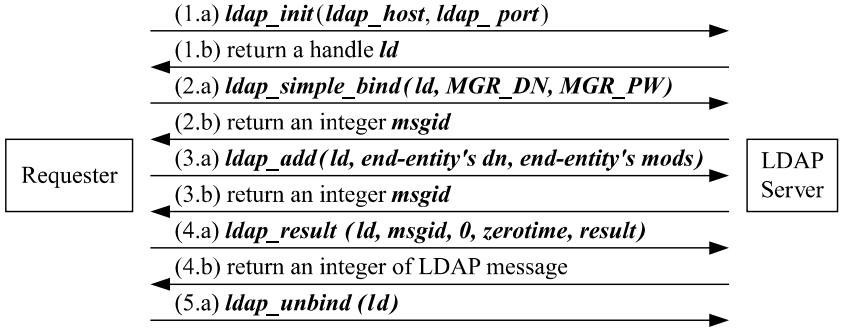
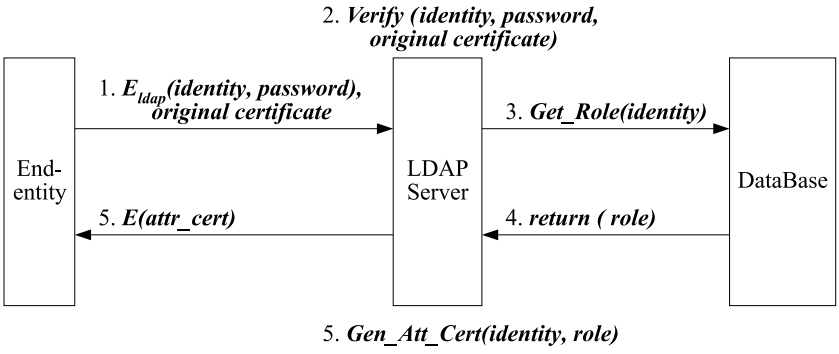


Fig. 10. Procedure of adding a new end-entity.



$E_{ldap}()$: To encrypt data using the LDAP's public key.
Verify(): To verify the user data according to the policy and the key pair.
Gen_Att_Cert(): To generate the attribute certificate according to identity and role

Fig. 11. Detail functions of access protocol.

entity uses the public key of LDAP server to establish a secure channel to connect with LDAP server, and encrypts its *identity* and *password*. And then end-entity transmits the encrypted data and *name certificate* to LDAP server.

2. *LDAP server calls the function Verify*(*identity*, *password*, *name certificate*) *to check the identity of the end-entity*: LDAP server first decrypts the encrypted data using its private key to get *identity*, *password*. LDAP server calls *Verify* (*identity*, *password*, *name certificate*) to check whether the *identity* and the *password* are valid and verifies whether the subject name of the *name certificate* is the same as the *identity*. If it returns true, LDAP server will get its role later. If not, LDAP server will reject the request of accessing. This step is to complete the authentication.

3. *LDAP server calls Get_Role*(*identity*) *to get role from database*: After a successful authentication, LDAP server will call *Get_Role*(*identity*) to connect with database. And then it searches the database to get the role of the end-entity (and other relative information of RBAC).

4. Database returns the role of the end-entity to LDAP server.

5. LDAP server calls *Gen_Att_Cert*(*identity*, *role*) to get attribute certificate and then sends back it to end-entity. LDAP server applies the *identity* and *role* of end-entity to generate an attribute certificate. The lifetime of this attribute certificate is shorter than that of name certificate.

In step (iii), we will use the function *search* to search the whole database and acquire the role of the end-entity [9].

The following is an example of searching an end-entity in Fig. 12:

(i) Starting be similar to the add function, we first call *ldap_init*(*ldap_host*, *ldap_port*) to initialize a connection to LDAP server, and gets back a handle *ld*.

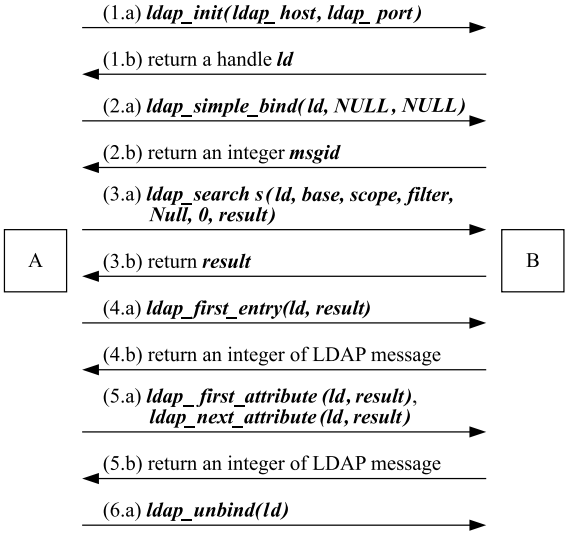


Fig. 12. A complete searching procedure with respect to access protocol.

(ii) Call *ldap_simple_bind*(*ld*, *MGR_DN*, *MGR_PW*). Then, we will retrieve the end-entity's private data by binding over to LDAP server as a manager.
(iii) Call *ldap_search_s*(*ld*, *base*, *scope*, *filter*, *attribute*, *attrsonly*, *result*). This function is used to search the directory synchronously. The *base* is a distinguished name of the entry that serves as the starting point for the search. The *scope* defines which scope to search. The *filter* is a string of words representing the condition of search, for example, such as (attribute type = attribute value). The *attribute* is defined as the attributes that the returned entries must have. The *attrsonly* specifies whether or not the attribute values can return along with the attribute types. The *result* is the result of search.
(iv) Call *ldap_ first_entry*(*ld*, *result*). This helps to retrieve step by step the entries *e* from the returned result. Then call *ldap_get_dn*(*ld*, *e*) to get the *dn* (distinguished name) of entry *e*. We need to compare *dn* with names that we want to find. Whenever the result matches, it means that we find the right *dn*.
(v) Call *ldap_ first_attribute*(*ld*, *e*, *ber*) and *ldap_next_attribute*(*ld*, *e*, *ber*). As we find the right *dn*, we start retrieving the attributes step by step. After finding the *dn* with success, there will be a pointer returned to indicate the position where the name of the first attribute in the entry is located. Therefore, the relevant attributes of RBAC can be found in access protocol step by step. And we can generate attribute certificate for end-entity and return it to end-entity. The attribute certificate must contain the subject name of end-entity. By doing so, we can bind the attribute certificate and name certificate more tightly.
(vi) Call *ldap_unbind*(*ld*). This function is to release the resource that we have used previously.

4.3. Other important function

Whenever regular end-entity wants to request the service from application server, it sends attribute certificate (and name certificate) to application server. And the application server checks the attribute certificate and releases end-entity's privileges according to its role. Often times when application server wants to acquire more information from end-entity, it can be done through calling the search function based on role's service access privilege. Although the application server is not in charge of the LDAP server, it needs to search the database for some reasons. Therefore, LDAP must provide service and allow anonymous search to meet the requests just mentioned above. General end-entities including the application server can get regular information about others through search (not including sensitive information about others). This can be done by first looking up for end-entity's attribute, and then by determining the access right based on end-entity's attribute. In case of general search, we can use *ldap_simple_bind*(*ld*, *NULL*, *NULL*). The last two parameters are *NULL* of which they are used to allow access to anonymous search and to increase the convenience of communication.

The administrator of the LDAP server can modify or delete end-entity's attribute. Since Session CA system does not need to maintain CRLs because of attribute certificate, therefore, the management of the directory becomes more flexible. And, the punishment for end-entity who deceives or violates the protocol can be done by deleting or modifying its attribute.

The functions of deleting and modifying are the same as the function of adding. The following are explanations for deleting function (for simplicity):

(1) Call *ldap_init*(*ldap_host*, *ldap_port*) to initialize connection.
(2) Call *ldap_simple_bind*(*ld*, *MGR_DN*, *MGR_PW*) to do authentication.
(3) Call *ldap_delete*(*ld*, *dn*) to delete the entry.
(4) Call *ldap_result*(*ld*, *msgid*, *all*, **timeval*, **result*) to get result.
(5) Call *ldap_unbind*(*ld*,*result*) to release resource.

In contrast, modifying function is the same as deleting function, except that step 3 is changed as calling *ldap_modify*(*ld*, *dn*, *mod_attr*) to modify the specified attributes.

5. Comparison and discussion

5.1. Discussion of designed protocols

In this section, we shall discuss the security of our proposed protocol. We shall discuss it with two parts, i.e., registration protocol and access protocol. And we shall analyze the security of these two protocols.

5.1.1. Registration protocol

In registration protocol, there will be a problem if digital envelope is used to deliver information. That is, we have no idea of knowing the identity of end-entity (including hacker) whenever SSL is used. There are two things we can do regarding this problem: (1) we would like to encourage the end-entity using face-to-face method to register at registration center, because this is the most secure way to prevent the problem, (2) our system will provide different roles for different end-entities to reduce the loss of being attacked should it happen.

In Section 4.1, we mention that symmetric cipher can be used instead of public key to encrypt at steps 2 and 4. The public key is used to transmit the session key for digital envelope. This improvement can increase the speed of encryption and reduce the load for both CA and RA. The session key between the end-entity and server must be changed after a period of time or it will be compromised.

5.1.2. Access protocol

In access protocol, end-entity needs to have LDAP server authenticate its identity, so the end-entity encrypts its identity and password first and sends the encrypted data and its name certificate to LDAP server to do verification. This is to make sure that the identity claimed by the end-entity is the same as the subject name of name certificate. And we use SSL connection to search the database. Basically, administrator and regular end-entity use different port to conduct the search and only administrator uses SSL to protect the transmitted data. In order to increase the security, we may use firewall to filter the transmission by allowing only permissive IP address and blocking up illegal IP address.

5.2. Comparisons with other certification authorities

When we are planning to implement CA, we are concerned not only the following important elements but also how to improve these elements.

5.2.1. Load

In a regular CA system, if the number of end-entity is huge, then CA system will become the bottleneck of load. The shorter the lifetime of the certificate, the heavier is the load of CA.

- *Reduce the load of CA*: In our proposed Session CA system, we adopt a LDAP server to issue the attribute certificate. As a result, the number of name certificate issued by CA will be greatly reduced by lengthening the lifetime of certificate. End-entity communicates with the LDAP server to get the attribute certificate, and use the attribute certificate to communicate with application server. In regular CA, if the short-lived attribute certificate is implemented, the load of CA will increase rapidly. Hence, we adopt a LDAP server to share the load of CA and the concept of attribute certificate to reduce the probability of issuing name certificate on CA.
- *Reduce the number of checking CRL on CA*: We do our best to apply the functionality of attribute certificate to replace that of name certificate in reducing the probability of using the name certificate. So the number of checking CRL of name certificate on CA will be reduced.

5.2.2. Security

- *Between end-entity and LDAP server*: In our Session CA system, LDAP server uses name certificate, identity, and password to identify the end-entity. If the identification is valid, LDAP server will issue the attribute certificate to end-entity. So the identification method using name certificate on the LDAP server is more secure than traditional method (using identity and password). CA adds new end-entity to directory system by using the same connection as the general end-entity. However, CA must use the identity of manager on LDAP server to request the service because CA is one of the managers on LDAP server.

• *Between end-entity and RA*: The connection between end-entity and RA is established with two methods, on-line (for convenience) or off-line (for more secure).

In the on-line method, we apply the SSL to protect the transmitted data. The confirmation of identity must use other strategy or policy.

In the off-line method, the RA applies face-to-face method to confirm the claimed identity is real identity.

• *Between RA and CA*: The connection between RA and CA is established using digital envelope—based on name certificate—to transmit information.

• *Reduce the possibility of setting flaw of access control*: In our proposed Session CA system, we apply the concept of RBAC and directory to organize the network system such that it matches with the structure of a real organization. In a LDAP server, we can apply RBAC easily. We can reduce the security flaw of an entire network system due to management flaw.

In our Session CA system, we apply RBAC to entire network system, thus the possibility of setting flaw of access control is reduced.

• *Harder to impersonate*: In our Session CA system, if an adversary wants to impersonate a given end-entity, this person needs to have the private key (for digital signature) or the password (for login requirement) of the end-entity (including RA, CA, end LDAP server). If an adversary wants to impersonate a given end-entity while establishing the connection with LDAP server, this person needs to have the password of this given end-entity. If an adversary had all the information (password, and private key) mentioned above, the LDAP server would issue the attribute certificate to this adversary.

5.2.3. Convenience

Generally speaking, it is expected that end-entities can register on-line just for convenience without having to take their data to RA. Our system meets these demands, and we have designed on-line registration protocol. But for security reasons, we use SSL to protect the end-entities as they send their registration data. And we can give different roles to different end-entities. For example, if an end-entity registers on-line, it may have fewer privileges than those who register at registration center. End-entity cannot get higher privileges until the identity that stands for end-entity is presented to RA or this end-entity has followed the protocol for a period of time without any violations. Often times, people find it troublesome to register at RA for name certificate using face-to-face method. So we need to provide convenient measures in order not to discourage end-entities. But this convenient method (on-line registration) has caused the system to take some risks with respect to its security.

5.2.4. Efficiency

• *Preventing the CA from being the bottleneck*: In a general network system, when end-entity requests the network service each time from application server, the application server needs to connect with CA to check CRL. If the scale of network system is huge, the checking of CRL may cause CA to become the bottleneck of the entire network system and then this procedure can increase the setup time for connecting lines. In our Session CA, we reduce the network traffic by providing the short-lived attribute certificate instead of checking the CRL of name certificate. In our Session CA system, there is no need to maintain the CRL of attribute certificate. The end-entity uses the attribute certificate to request the network service from application server. The application server does not need to check the CRL of attribute certificate and name certificate.

• *Increase the setup time of each connection between end-entity and application server*: In our Session CA system, we need an additional setup time to get the attribute certificate for each connection linked between end-entity and application server. In other CA system, the time of checking CRL increases depending on the scale of network system. In other words, the additional setup time increases very slowly in our system when the scale of network system becomes very large.

Since the lifetime of attribute certificate is so short, we do not have to revoke the attribute certificate and to maintain the CRL of the attribute certificate. The attribute certificate need not contain a list of X.509 certificates from root CA to

issue CA, because it contains the attribute already. It can be processed faster than the name certificate.

5.2.5. Scalability

The scalability of Session CA is much better. If we organize the network system properly (each organization unit contains with one LDAP server), the LDAP server is not likely to become the bottleneck. When the scale of a network system becomes very large, the general CA system has to maintain a larger CRL and meet the request of servicing a much larger network. So, CA may become the bottleneck. If CA distributes the services of checking CRL to other sites, the CRLs between CA and other sites must be kept consistent. The correctness of checking CRL cannot be confirmed unless CA and other sites are synchronized periodically. The network traffic of synchronized CRL depends inversely on the length of period. That is, the shorter the period of synchronization, the higher is the accuracy of the information, and the larger is the network traffic.

5.2.6. Capability

The capacity of our CA system is quite flexible. We can use many LDAP servers to distribute the load of CA system. Therefore, CA system only needs to deal with jobs such as certificate issuing and revoking, etc. Unlike other CA systems that need a powerful computing server to perform large amount of network service of checking CRL, our server just needs a regular PC and several distributed LDAP servers. With this technology, we are capable of providing a quality as good as the expensive CA system. And we can add more LDAP servers to accommodate more end-entities. Besides, we can use one LDAP server to attend only one or several domains and use RBAC to enhance the security of our system. Therefore, every LDAP server can handle each end-entity's request very well. As a result, our CA system would not cause too much of load in terms of frequent maintenance.

5.2.7. Policies

- *The policy of our Session CA system is the same as other CA systems*: Policy adopted in our Session CA system is the same as the name certificate issued by other CA systems. However, our LDAP server uses RBAC to issue the attribute certificate and control the privilege of end-entity.
- *The flaw of management will be reduced*: In our LDAP server, we can map the organizational hierarchy to the hierarchy of X.500 and apply RBAC to arrange and manage the access control. We can separate the management between the name certificate and the access control of network resource and service. These procedures will reduce the security flaw of management.

5.3. Session certification authority vs. Kerberos

5.3.1. The relation between server and end-entity

In Kerberos system, each end-entity must share a secret key with Authentication Server (AS). This shared secret key is derived conversely from the password of end-entity. It is possible to derive the password from the ticket.

In our Session CA system, RA is responsible for verifying the identity of end-entity and checking the accuracy of end-entity's key pair, and then it requests CA to issue the name certificate for end-entity.

5.3.2. Certificate vs. ticket

The functionalities between certificate and ticket are similar. Both are used as credentials to access resource and service. They can bind some access control information with them.

5.3.3. Session CA vs. authentication server

In our Session CA system, we use Session CA and LDAP server to issue certificate. The role of Session CA is similar to AS in Kerberos.

The Session CA issues the name certificate to end-entity, and end-entity uses the name certificate to get certain attribute certificate for a given application server from LDAP server.

In Kerberos system, AS issues a ticket-granting ticket (TGT), and the end-entity uses this TGT to get the ticket to a given application server from ticket-granting server (TGS).

5.3.4. LDAP server vs. ticket-granting ticket

The role of LDAP server is similar to that of TGS in Kerberos.

LDAP server applies RBAC to give end-entity the proper right to access resource and service.

The TGS in Kerberos can only use the ACL to restrict the privilege of accessing resource and service for end-entity. So, the application server finally controls the access right.

5.3.5. Access control over resource and network service

In Session CA system, LDAP server is responsible for managing most of the access control.

In Kerberos system, the application server is responsible for managing most of the access control.

5.3.6. Cooperation with operating system

Kerberos tightly cooperates with operating system. Therefore, Kerberos can control more system resource.

So far, Session CA system has a relatively inferior capability in combining with operating system because the design of the Session CA system focuses mainly on network service. Therefore, Session CA system controls system resource poorly. But if the functionalities of certificate is integrated into the operation system, then Session CA system will combine better with operation system to get more access control of system resources.

Regarding the access control of system resource, the capability of Kerberos system is better than that of Session CA system.

As far as the complexity of management is concerned, Kerberos system is more complex than Session CA system.

5.4. The future of lightweight directory access protocol

The motivation behind LDAP v2 was to provide lightweight access to directory services by using X.500 data and information model. This approach having been endorsed in the drafts of LDAP v3 was ratified during the summer of 1997 to be used to solve the very same problem—providing lightweight access to directory services.

As far as we know, LDAP is the best program to be used to support X.500 and this will continue to dominate in the foreseeable future. The following two aspects can be used to understand our policies.

(1) *From the client perspective*: The availability of LDAP-enabled applications, such as mail clients and Web browsers on most desktops, is enormously helpful in legitimizing the deployment of X.500 servers.
(2) *From a server perspective*: In the absence of LDAP servers' proven capability; in other words, whether LDAP is robust enough to be able to provide a high quality server-to-server connections or to replicate between two or more disparate standalone LDAP servers, it appears that the requirement for directory synchronization between directories still remains.

The importance of LDAP v3 is that it will become the protocol with an accepted form to be widely implemented in the industry for the next two to three years. Private vendors may decide to make and implement their own extensions using LDAP v3—some already have—but these will be proprietary in terms of their intents and purposes. Although these programs create competition for us, it will make great improvement in LDAP per se. The existence of competition between vendors and our proposed system will benefit both developers and customers in the future.

5.5. Other applications

In electronic commerce (EC), a convenient and secure environment is needed to provide a good access for communication. After authentication and identification, the merchant and the end-entity can trade with the other. But in EC, the consideration of security should be emphasized even more. If someone wants to deceive, it could cause loss of a great deal of money. So the most important thing in terms of EC is to reduce the possibility of fraud. In general, people want to buy merchandise on the Internet just for convenience. They do not want to go shopping when the weather is bad. All they want to do is buy whatever they want using on-line service and pay for it with credit card. It is supposed to be easy; even a small child can do this. But if we use

a complex protocol, the convenience of the EC may disappear. People would rather go shopping instead of using on-line service, because registration is too complex for them to apply. So this is a big problem.

Our Session CA is a good strategy to provide good on-line service. We do not need to check CRLs; hence we can reduce the load having a lot of queries. We can bind role to restrict the role of the end-entity. For example, an end-entity cannot buy something exceeding the limit of his credit because the merchant will check the attribute of credit. Although this is not a good way to prevent fraud, it can reduce the loss should it happen.

Another example of using directory service is Active Directory of Windows 2000. This is the same as LDAP, it emphasizes the importance of centralized management of vast network database and can reduce the cost of administration of this system.

Using Active Directory, Windows 2000 has the following two features. The first one is the laws of the forest and the other is the fault tolerance provided.

(1) *The laws of the forest*: This is the same as LDAP service. Active Directory uses a hierarchical model described by metaphors. A ''forest'' denotes parts of a network, whereby a ''tree'' can share information with other trees if it is a member of the same forest.

At the root of each tree is a domain. In each domain an administrator can add more domains, such as organizational units (OUs) and objects, the most granular items in Active Directory. Each object is given a global unique identifier to be used as a permanent reference with respect to that object; this identifier allows the object to be renamed or moved without causing any problems.

(2) *Fault tolerance provided*: To provide fault tolerance, Active Directory uses domain controllers. Unlike NT's domain controllers, the domain controllers are not grouped into primary or backup categories.

Replication between sites is controlled by AD replication services, which can be scheduled and also limited to a certain transfer rate to ensure that replication does not flood or slow down network links.

6. Conclusions

In this paper, we have described as how we implement LDAP service and what role the LDAP service plays in the Session CA system. We know that the built-in extensibility of the LDAP architecture makes it easier to modify the protocol for new situations and our own needs. Here are the key advantages of LDAP service.

Key advantages of LDAP service:

- Run directly over TCP, eliminating overhead of the OSI session and presentation layers required by DAP.
- Simplify the X.500 functional model.
- Use string encoding for distinguished names and data elements (RFC 1778).
- Lessen clients from the burden of chasing referrals.

With these advantages, we can take advantage of these useful facts to implement our CA system. We can reduce the load of CA system by using LDAP service. This helps to allocate the load of CA to LDAP server. On the other hand, when the number of end-entities' query is to olarge for LDAP server to afford, we can use distributed LDAP servers to share the load of the system.

In a general CA system, checking CRL may become a serious problem. In order to solve this problem, our Session CA system does not maintain the CRLs of attribute certificate. However, this approach may cause another terrible problem ''security''. Concurrently, our CA system uses this technology called RBAC in conjunction with attribute certificate. When an unidentified end-entity wants to access other applications, it must show its identity, password, and the name certificate first, and then LDAP server will issue an attribute certificate of which it contains its authorized role. And the end-entity can access the resource according to its role. This will improve the security of our CA system because we do access control through RBAC.

Although LDAP is not an ultimate directory protocol, it has grown to include both standalone and replicated servers. LDAP should be recognized as a good method for accessing

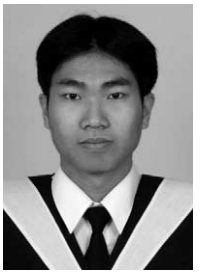
information from a variety of directories. With the implementation of our Session CA system, we can know how much more applications that LDAP service can apply to. Furthermore, we can construct a better CA system in conjunction with these technologies mentioned above.

References

[1] D.R. Stinson, Cryptography Theory and Practice, CRC Press, Boca Raton, 1995.
[2] Internet Public Key Infrastructure-Part I: X.509 Certificate and CRL Profile.
[3] A.O. Freier, P. Karlton, P.C. Kocher, The SSL Protocol version 3.0., Netscape Communication Corp, March 1996.
[4] S. Farrell. TLS extensions for Attribute Certificate based Authorization, February 1998. draft-ietf-tls-attr-cert-00.txt.
[5] D.E. Ferraiolo, J.A. Cugini, D.R. Kuhn, Role-based access control (RBAC): features and motivations, Proceedings of the 11th Annual Computer Security Applications Conference, IEEE Comput. Soc. Press, Los Alamitos, CA, 1995, pp. 241–248.
[6] W. Yeong, T. Howes, S. Kille, Lightweight Directory Access Protocol, RFC 1777, Performance Systems International, University of Michigan, ISODE Consortium, March 1995.
[7] M. Wahl, A Summary of the X.500(96) User Schema for use with LDAPv3, RFC 2256, December 1997.
[8] Internet Public Key Infrastructure-Part II: Operational Protocols.
[9] T. Howes, A String Representation of LDAP Search Filters, RFC 2254, December 1997.
[10] J.S. Park, R. Sandhu, Smart Certificates: Extending X.509 for Secure Attribute Services on the Web.
[11] B. Schneier, Applied Cryptography, second ed., Wiley, New York, 1996.

Yi-Shiung Yeh did his Ph.D. in Computer Science, (September 1981–December 1985) MS in Computer Science, (September 1978–June 1980) from the Department of EE and CS, University of Wisconsin-Milwaukee. Since August 1988 he is working as a Associate Professor in the Institute of CS and IE, National Chiao-Tung University. From July 1986 to August 1988 he worked as a Assistant Professor in the Department of Computer and Information Science, Fardham University. From July to December 1984 he was a doctorate intern at Johson Controls, Inc. From August 1980 to October 1981 he worked as a System Programmer in System Support Division, Milwaukee Country Government. His research interests are Cryptography and Information Security, Reliability and Performance, DNA Computation.

Wei-Shen Lai did his MS in Computer Science and Information Engineering (September 1993–June 1995) from the Department of CSIE, National Chiao-Tung University. Since September 1995 he is doing his Ph.D. in Computer Science and Information Engineering from the Department of CSIE, National Chiao-Tung University. His research interests are Cryptography and Information Security, Network Security, Electronic Commerce and Computer Architecture.



Chung-Jaye Cheng did his MS in Computer Science and Information Engineering (September 1998–June 2000) from the Department of CSIE, National Chiao-Tung University. Since October 2000 he is working as a System Analyst in Computer and Communication Research Labs, Industrial Technology Research Institute of Taiwan, ROC. His research interests are Cryptography and Information Security and Network Security.