通信情報保護學會誌
第9卷 第1號, 1999. 3

LDAP프로토콜에 대한 고찰

최 진 주*

요 약

본 문서에서는 개방된 네트워크에서는 인증기관의 필요성이 대두되고 있고 인증기관시스템에서는 인증서보관소에 있는 내용에 사용자가 접근을 위해 LDAP프로토콜이 주로 사용되고 있다. 그러므로 LDAP프로토콜과 검색방법에 대해 살펴보고자 한다.

1. 서 론

공개키를 이용한 인증기관(CA Certificate authority)은 PKIX에서 정의한 X.509에 근거하여 인증서를 발행하고 사용자의 공개키와 사용자의 ID 정보를 결합하여 CA의 서명을 통해서 전달되어진다.

인증서를 전자상거래나 문서교환등에 사용할 때는 유효성을 사전에 확인하여야 한다.

인증서를 확인하는 방법은 CA가 운영하고 있는 CRL폐지목록 보관소에 해당하는 인증서의 일련번호가 있는가를 확인하고 있으면 그 인증서는 유효하지 않으며 없으면 유효한 것으로 판단한다.

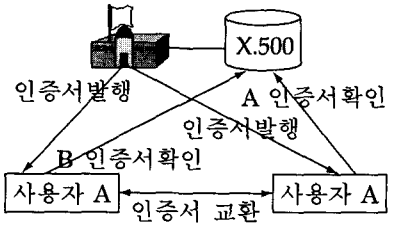


그림1 인증서 발행/확인

* 국방대학원 전자계산학과

인증기관의 보관소는 통상 X.500을 이용하여 구현하는데 디렉토리에 있는 정보를 검색하고 관리하는데는 LDAP 프로토콜을 사용하며 인터넷의 Working group에서 표준화 작업을 수행하고 있다.

본 고에서는 RFC1777과 2251에 있는 내용을 분석하여 연구하였으며 2장에서는 디렉토리에 대해 제 3장에서는 LDAP구성에 대해 기술하고 4장은 LDAP 프로토콜을 설명하며 5장은 결론으로 맺는다.

2. 디렉토리 서비스

2.1 개요

LDAP는 X.500디렉토리에 있는 CRL과 인증서 및 기타 정보를 검색하기 위해 필요한 스펙을 정의한 것으로 보관되어 있는 정보를 검색하고, 관리하기 위한 것이다.

얼마전까지만해도 LDAPv2가 사용되어지다가 현재는 referal 기능이 보강된 LDAPv3가 RFC2251로 공표되어 있는 상태이다.

최초 X.500디렉토리는 OSI환경을 위해 설계

되었고 프로토콜로는 DAP(Directory Access Protocol)이 있으나 많은 기능으로 인해 overload가 많이 발생하였는데 이를 해결하고 TCP/IP에서도 작동할 수 있도록 구현한 것이 LDAP이다.

2.2 디렉토리 서비스란?

디렉토리는 데이터베이스와 같으나 좀더 기술적이고 attribute에 의한 정보표현을 하는 형태이다. 디렉토리의 정보는 쓰기(write)보다는 읽기(read)가 훨씬 많다. 그래서 디렉토리는 복잡한 트랜잭션이나 다량의 복잡한 갱신을 위해 규칙적으로 주기억장치에 쓰기(roll-back scheme)기능을 구현할 수는 없다.

비록 갱신이 허용되더라도 단지 해당 객체의 전부변경이거나 또는 전혀하지 않던가 둘중의 하나가 주형태이다[16]. 디렉토리는 look-up나 search연산에 빠른 응답을 해야하며 응답시간을 줄이고 이용성과 신뢰성을 증진하기 위해 정보를 폭넓게 복사할 수 있어야 한다.

디렉토리 서비스를 제공하는 많은 다른 방법이 존재한다. 이들 각각은 디렉토리에 저장된 각기 다른 정보를 제공한다.

디렉토리 서비스는 local하여서 서비스의 제공이 제한된것과 global하여서 서로 다른 시스템에 자료가 분산되어서 있고 서비스 제공은 상호 협조하여 자료를 제공하는 것으로 구분되며 global한 서비스는 같은 형태의 정보를 사용자에게 제공하기 위해 통일된 namespace로 정의된다.

2.3 디렉토리

인증을 위한 디렉토리는 X.500을 주로 이용하며 필요에 따라 LDAP를 stand-alone씩으 로 구성하여 서비스를 제공할 수 있다.

LDAP(Lightweight directory Access Protocol)란 global한 디렉토리 서비스 모델로 TCP/IP위에서 실행되도록한 프로토콜이다.

상세한 사항은 버전2와 3은 RFC1777, 2251 "THE Lightweight Directory Access protocol"에 잘 명시되어 있다.

디렉토리에 어떠한 정보가 저장되는가 하는 것이 중요한데 LDAP에는 엔트리(entry)에 기초한 서비스 모델이다. 엔트리는 DN (Distinguished name)이라고 불리는 이름을 가진 속성(attributes)의 집합이다. DN은 엔트리를 참조하는데 모호하지 않도록 하기 위해 사용된다.

각 엔트리의 속성은 타입과 하나이상의 값으로 구성된다. 타입이란 기호화된 스트링으로 common name를 위한 "cn"이나 email address를 위한 "mail"과 같은 것이다[1].

값은 속성의 타입에 따라 결정되는데 보기로 mail속성은 값으로 "babs@umich.edu"라는 값을 갖고 jpegphoto속성은 JPEG/JFIF형식의 그림을 값으로 한다.

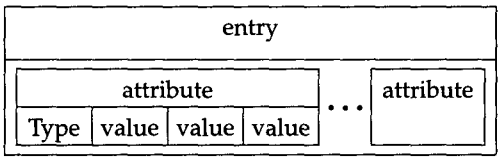


그림2 엔트리의 구조

LDAP에서 디렉토리 엔트리는 계층적인 트리와 같은 구조를 가지며 이는 정치적, 지리적, 조직적 경계를 나타내는 것이다. 국가를 나타내는 엔트리는 트리의 최상위에 오게되며 다음으로 주(states) 또는 국가 조직, 아래에 사람, 조직체, 아래에 문서, 기타 생각나는 것들이 오게된다.

그림3은 트리를 나타내는 것으로 어떤 개체가 명확히 무엇인지를 나타내는 것이다.

LDAP는 트리로 구성되고 각 노드는 디렉토리 오브젝트이라고 하고 또는 디렉토리 엔

트리라고 한다. 그리고 LDAP는 objectclass라는 특별한 속성을 이용하여 엔트리 내에 어떠한 속성이 요구되고 허용되어지나를 통제한다.

objectclass속성의 값에 의해 엔트리가 따라야할 스키마규칙이 결정된다.

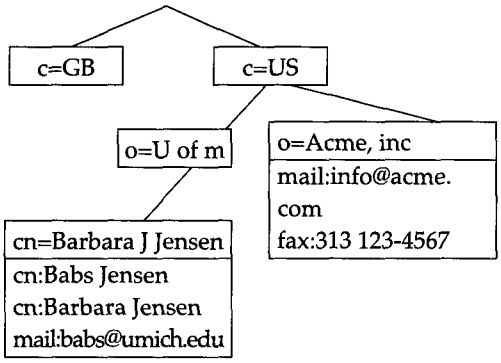


그림3 LDAP디렉토리 트리

정보는 어떻게 참조되는가? 엔트리는 DN에 의해서 참조되며 DN은 RDN이라고 하는 엔트리 그 자체의 이름과 조상의 엔트리의 연결에 의해서 된다.

보기로 Barbara Jensen이라는 엔트리는 RDN으로 "cn=Barbara J Jensen"을 가지며 DN으로는 "cn=Barbara J Jensen, o=U of M, c=US"을 가진다. DN의 완전한 구조는 RFC1779 "A String Representation of Distinguished Names"에 잘나타나 있다.

3. LDAP구성

3.1 모델

LDAP 작동과 관련된 PKI구성요소는 말단 개체(end entity)와 인증기관, 보관소로 구성된다.

말단 개체와 인증기관은 LDAP프로토콜을 이용해서 보관소에 필요한 정보를 검색하며 또한 인증기관은 이 프로토콜을 이용하여 보관소의 PKI정보를 관리하게 된다. 보관소는 인증서, 인증서폐지 목록, 인증기관 정책, 기타 필요한 정보를 보관하는 데이터베이스이다.

구성요소 들간의 관계를 보면 LDAP서버는 말단개체의 요청을 받아서 보관소인 디렉토리를 검색하고 응답을 말단개체에 전달하게 된다. LDAP버전 1과 2에는 referals을 반환하는 기능이 없었으나 버전3에서는 말단개체와 다른서버에게로 반환하는 것을 허용한다.

3.2 LDAP의 구성

Michigan대학은 LDAP프로토콜을 이용한 Slapd(stand-alone LDAP daemon)과 slurpd (stand-alone LDAP update replication daemon)이를 바탕으로 LDAP의 구성방법을 알아보면 다음과 같다[11].

local영역에 LDAP를 사용하여 다른 영역과의 연결에 관심이 없는 경우에는 클라이언트와 서버로만 구성되며 필요시 확장할 수 있다.

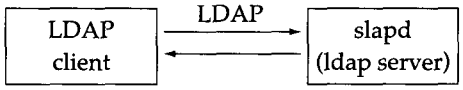


그림4 local 영역의 사용

local영역의 이용과 동시에 기존에 X.500디렉토리 서비스를 이용하고 있을 때는 X.500과의 연결을 위한 기능이 있는 ldapd(LDAP daemon)를 이용하여 연결을 실시한다.

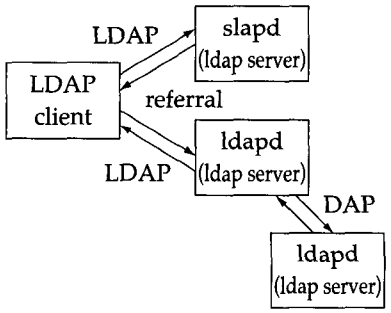


그림5 local서비스와 X.500의 연결

LDAP와 X.500서버간의 front-end 관계의 구성을 보면 아래와 같고 이때 ldapd는 X.500과의 연결을 위한 게이트웨이형 LDAP서버 역할을 수행한다

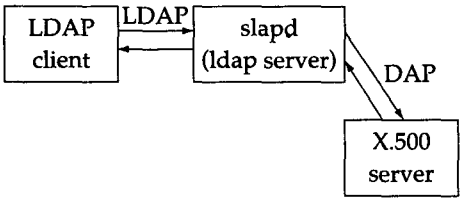


그림6 게이트웨이형 LDAP서버

한 개의 LDAP서버로는 서비스를 제공할 수 없는 경우에는 신뢰성과 확장성을 보장할 수 없다 이를 위해 Slurpd daemon와 마스터 Slapd와 슬래브 Slapd을 이용하여 변화 내용을 갱신하고 서비스 제공이 가능토록 한 구조이다.

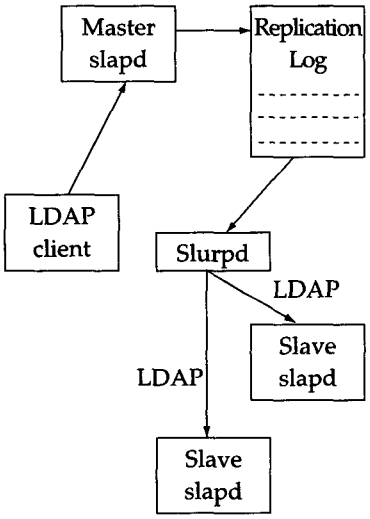


그림7 LDAP의 확장

4. LDAP프로토콜[10]

4.1 용어의 정의

엔트리: 하나 이상의 속성 집합으로 구성된다.

속성 : 속성 타입과 속성 값으로 구성된다.

속성타입(attribute type) : 엔트리에서 속성이 개수, 속성값의 신텍스, 속성 값에서 매핑 방법 및 기타 다른 기능을 정의한다.

속성값(attribute value) : 속성 변수에 기록된 실제의 값

Naming context : 특정 서버가 해당하는 엔트리로부터 그의 하위 서버 및 다른 서버까지 포함하는 엔트리의 모음이다.

스키마 : 모든 속성 타입의 정의, 오브젝클래스(object class)의 정의, 서버에 요청된 검색 정보와 디렉토리의 엔트리를 매핑시키는 정보, 디렉토리의 엔트리를 추가 및 수정가능 여부 에 대한 정의의 모음을 말한다.

4.2 속성의 종류

① 오브젝클래스 속성

엔트리는 오브젝클래스속성을 가져야 하며 오브젝클래스 속성은 엔트리의 오브젝클래스들을 명시한 것으로 시스템과 사용자스키마와 함께 엔트리에 속성의 허용여부를 결정한다. 클라이언트는 속성의 값을 변경할 수 있으나 오브젝클래스 속성은 변경할 수 없어서 기본적인 구조를 변경할 수 없게 하고 있다 (예: 사람은 국가로 변경할 수 없음).

엔트리를 새로 만들거나 오브젝클래스의 값을 추가할때는 모든 상위의 클래스에도 추가되어야 한다.

② 연산 속성(operation attribute)

이 속성은 서버가 디렉토리를 관리 하는데 필요한 속성으로 명시적으로 요청하지 않으면 값이 리턴되지 않는다.

③ 일반 속성

"mail"과 같이 연산 속성에 포함되지 않는 일반적 속성으로 그들의 스키마와 신텍스의 제한을 받는다. 서버는 오브젝클래스, 스키마의 정의에 어긋나거나 , 서버가 모르는 연산 속성, 관리 목적의 속성일 경우에 클라이언트가 엔트리에 속성을 추가하는 것을 용인하지 않는다.

④ 기타 연산 속성

이 연산 속성은 서버에 의해 자동적으로 유지되고 클라이언트에 의해 수정될 수 없다.

- creatorsName : 디렉토리에 엔트리를 추가한 사용자의 DN이다.
- createTimestamp : 디렉토리에 엔트리가 추가된 시간이다.
- modifiersName : 마지막으로 엔트리를 수정한 사용자의 이름이다.
- modifyTimestamp : 엔트리가 마지막으로 수정된 시간
- subschemaSubentry : 엔트리의 스키마를 제어하는 서브스키마의 엔트리 DN

4.3 서브스키마 엔트리와 서브 엔트리

서브스키마 엔트리는 디렉토리 스키마에 대한 정보를 관리하기 위해 사용된다.

특히 디렉토리 서버에 의해 지원되는 오브젝클래스들과 속성타입들에 대한 정보를 관리한다. 하나의 서브스키마 엔트리는 디렉토리 트리의 특정부분에 해당하는 엔트리에 의해 사용된 모든 스키마 정의를 포함하고 있다.

모든 서브스키마 엔트리에 포함되어야 할 속성 4가지는 다음과 같다.

- cn : 이 속성은 서브스키마 엔트리의 RDN을 나타내기 위해 사용되어진다.
- objectClass : 속성은 적어도 "top" and "subschema" 값을 가져야 한다.
- objectClasses : 이 속성의 값은 서버에게 알려진 object class을 설명한다.
- attributeTypes : 이 속성의 값은 서버에게 알려진 속성 타입을 설명한다.

추가적인 기능을 수행하기 위해 다음과 같은 속성도 지원된다.

matchingRules, matchingRuleUse, dITStructureRules,

dITContentRules, nameForms and ldapsyntaxse을 포함하고 있어야 한다.

서버는 createTimestampe와 modifyTimestamp는 클라이언트가 서브스키마 정보를 캐쉬에 보관하기 위해 서브스키마 엔트리에 제공된다.

클라이언트는 엔트리에 대해 기초적인 오브젝트(basic object)의 검색요구로 서브스키마 엔트리로부터 속성을 검색한다. 이때 검색필터는 "objectclass = subschema"가 된다.

4.4 서버의 특정 데이터 요구

LDAP서버는 각 서버에 대한 특정한 정보가 있으면 알려야 한다. 이것은 각 루트 DSE에 있는 속성그룹으로 표현되며 DSE는 길이가(length)가 0인 LDAPDN이다.

이속성은 클라이언트가 기초적 오브젝트 검색으로, 필터에는 "objectclass = *"으로 검색될 수 있다. 다음과 같은 루트 DSE 속성이 있다.

- namingContexts : 서버에 유지된 naming context들, Naming context는 X.501의 17장에서 정의된다.
- subschemaSubentry : 서버에 알려진 서브스키마 엔트리들
- altServer :후에 이용되지 않을 경우의 대체 서버
- supportedExtension : 지원되는 확장연산 리스트
- supportedControl : 지원되는 제어 리스트
- supportedSASLMechanisms : SASL의

보안 특성을 지원하는 리스트
- supportedLDAPVersion : 서버에 의해 구현될 LDAP versions

4.5 프로토콜 요소

버전은 확장 메카니즘을 사용하지 않는 프로토콜의 변경은 다른 버전번호를 사용한다.

클라이언트가 처음 서버에게 bind요구의 일부로 버전정보를 알려야 하고 만일bind요구가 없으면 서버는 클라이언트에서 버전3이 지원됨을 가정한다(버전2는 먼저 클라이언트가 bind요구를 해야 한다).

클라이언트는 루트 DSE의 supported LDAPVersion 속성을 읽어서 서버가 지원하는 프로토콜 버전을 결정할 수 있다.

버전3 이후의 LDAP는 이 속성을 지원한다. 그러나 버전2는 이 속성을 지원하지 않는다.

4.5.1 공통요소

이 장은 LDAPMessage envolope PDU형식과 프로토콜 연산에 사용된 데이터 타입 에 대한 정의를 서술하고 있다.

4.5.2 메시지 묶음(message envolope)

프로토콜이 교환되기 위해서 모든 프로토콜은 하나의 묶음으로 캡슐화되며 정의는 다음과 같다.

```
LDAPMessage ::= SEQUENCE {
    messageID MessageID,
    protocolOp CHOICE {
    bindRequest BindRequest,
    bindResponse BindResponse,
    unbindRequest UnbindRequest,
    searchRequest SearchRequest,
    searchResEntry SearchResultEntry,
    searchResDone SearchResultDone,
    searchResRef
SearchResultReference,
    modifyRequest ModifyRequest,
    modifyResponse ModifyResponse,
    addRequest AddRequest,
    addResponse AddResponse,
    delRequest DelRequest,
    delResponse DelResponse,
    modDNRequest ModifyDNRequest,
    modDNResponse
ModifyDNResponse,
    compareRequest CompareRequest,
    compareResponse
    CompareResponse,
    abandonRequest AbandonRequest,
    extendedReq ExtendedRequest,
    extendedResp ExtendedResponse },
    controls [0] Controls OPTIONAL }

MessageID ::= INTEGER (0 ..
                maxInt)
    maxInt INTEGER ::=
  2147483647 -- (2^^31 - 1) --
```

LDAPMessage는 프로토콜 교환을 위해 필요한 공통의 필드를 하나의 묶음으로 캡슐화한 것이다. 이때 유일한 공통필드는 message ID와 controls이다.

서버는 클라이언트로부터 인식할 수 없는 SEQUENCE 태그를 알 수 없으면 message ID는 분석되지 않으며 protocolOP 태그를 인식할 수 없거나 인코딩 구조나 데이터 필드의 길이가 적절하지 않으면 프로토콜 에러라는 응답을 리턴하고 즉시 연결을 해제한다.

4.5.3 Message ID

message ID값을 가지고 요구를 하면 상응하는 message ID값을 가지고 응답을 하며 message ID값은 다른 message ID 값과는 구별된다. 그리고 클라이언트는 앞의 요구에 대한 최종응답을 받지 못했으면 같은 mess-age ID를 가지고 일정기간동안 요구를 할수 없다.

클라이언트는 abandonRequest는 응답이 필요 없기 때문에 abandonRequest다음에 행한 다른 요구에 대해 응답을 받지 않고서는 abandonRequest을 다시 사용해서는 안된다.

4.6 추가적 프로토콜 연산의 기초적 정의

위에서 정의한 LDAPMessage 에 추가하여 프로토콜 연산을 정의하기 위해서는 다음과 같은 정의가 사용된다.

4.6.1 스트링 타입

LDAPString 타입은 OCTET STRING[12] 타입인 ISO 10646[13]문자세트를 이용 인코드되나 UTF-8[4]알고리즘을 따라서 행해진 다. UTF-8은 한바이트는 ASCII와 같은 형태로 표현되고 나머지 다른 바이트는 임의의 길이에 해당하는 문자로 인코딩된다.

LDAPString : : = OCTET STRING

LDAPOID는 스트링에 대한 허용된 값은 오브젝트 식별자에 대해 십진수와 점(dot)를 이용해서 표현한 것이다.

LDAPOID : : = OCTET STRING
보기로
1.3.6.1.4.1.1466.1.2.3

4.6.2 Distinguished Name와 Relative Distinguished Name

LDAPDN과 RelativeLDAPDN은 각자 DN과 RDN을 나타내기 위한 정의이다.

LDAPDN : : = LDAPString

RelativeLDAPDN : : = LDAPString로
정의에서 〈name〉와 〈name-component〉는
〈distinguished-name〉: : = 〈name〉
〈relative-distinguished-name〉: : = 〈name-component〉로 표현된다.

속성타입은 RDN에 나타날 수 있지만 속성서술(attribute decription)은 DN을 명시하기 위해 사용되지 않는다.

4.6.3 속성타입

속성타입은 값으로 스펙에 정의되어 있는 AttributeType 관련된 텍스트스트링을 가진다.

AttributeType: : = LDAPString

각 속성타입은 유닉한 오브젝트 식별자를 가진다. 이 식별자는 점과 십진수를 이용 표시된다(예:속성타입인 organizationname는 오브젝트 식별자로 "2.5.4.10"을 가진다.).

스펙에서 속성타입에 대해 하나이상의 텍스트를 할당할 수 있으며 반드시 하나의 문자로 시작한다.

속성타입에 대해 텍스트 이름으로 서버에게 요구되면 서버는 반드스 텍스트 이름으로 응답을 하고 오브젝트 식별자는 속성타입에 대해 텍스트 이름이 없을 때만 사용해서 응답한다.

서버는 그들이 지원하고 있는 속성타입에 대한 리스트를 서브스키마 엔트리에 나타내야 한다. 클라이언트는 오브젝트 식별자와 속성타

입에 대한 다른 관련 정보를 파악하기

위해 서브스키마 엔트리로부터 속성타입의 값을 검색을 할 수 있다.

4.6.4 속성서술(attributedescription)

속성서술은 속성타입의 정의에 대한 슈퍼세트 개념이다. 이는 ASN.1[3]의 정의를 이용 표현되나 추가로 option이 명시된다.

AttributeDescription ::= LDAPString이며 AttributeDescription의 값은 다음과 같이 BNF 표기법을 이용한다.

〈AttributeDescription〉 ::=
〈AttributeType〉 [";" 〈options〉]
〈options〉 ::= 〈option〉 | 〈option〉
";" 〈options〉
〈option〉 ::= 〈opt-char〉
〈opt-char〉*
〈opt-char〉 ::= ASCII-equivalent
letters, numbers and hyphen

AttributeDescription 예

cn
userCertificate ; binary

옵션인 binary는 이 문서에서 정의한 것이며 기타 다른 옵션은 IETF나 사용자가 정의할 수 있다. 그러나 서버가 모든 옵션을 다 지원할 수 있는 것은 아니다.

하나이상의 옵션을 가진 속성서술은 옵션이 없는 속성타입의 서브타입이라고 볼 수 있다. 서버는 구현하지 않은 옵션을 사용한 속성서술은 인식되지 않는 속성타입으로 취급한다.

데이터 타입인 "AttributeDescributionList"는 속성타입들의 리스트이다.

4.6.5 binary 옵션

속성서술에 binary옵션이 사용되면 속성에 정의된 스트링에 기초한 표기법을 무시하고 대신에 BER(Basic Encoding Rules)을 이용해서 인코드된 binary값으로 전달된다는 의미이다.

binary옵션의 존재여부는 프로토콜에서 속성값의 전달에 영향을 미친다. 서버가 어떤 속성을 어떤형식을 이용 보관하고 있을때 클라이언트가 속성값을 서버가 리턴 해주기를 어떤 형식으로 요청시 서버가 그 형식으로 응답할 수 없다면 서버는 이 속성타입을 인식할 수 없는 것으로 취급한다.

옵션은 복잡한 ASN.1데이터타입 형태의 속성을 나타내고 클라이언트에 의해 필요한 것을 나타내기 위해 사용되며 그 예로는 인증서와 인증서리스트가 있다.

4.6.6 속성값

속성값 타입의 필드는 그 값으로 Attribute Value데이터타입을 인코딩하는 스트링이다.인코드된 Attributevalue데이터타입은 OCTET STRING을 취한다.

AttributeValue: = OCTET STRING

인코딩에 대한 사이즈의 제한규정은 없어서 사진과 같은 속성에는 메가바이트의 속성값을 포함할 수도 있다.

4.6.7 Attribute value Assertion

AttributeValueAssertion타입의 정의는 X.500 디렉토리 속성과 유사하다.

여기에는 속성서술과 매칭규칙인 assertion Value으로 구성된다.

AttributeValueAssertion ::=
SEQUENCE {
attributeDesc AttributeDescription,
assertionValue AssertionValue }

AssertionValue ::= OCTETSTRING

attributeDesc가 binary옵션을 가지면 assertion Value는 값을 binary로 인코딩된다는 것을 나타낸다.

모든 스트링 값의 유저속성은 [5]에 명시되어 있다. assertionValue의 값은 속성값의 신텍스와 같다. 클라이언트는 요구와 검색필터에 assertionValue값을 속성값으로 사용한다.

4.6.8 속성(attribute)

하나의 타입과 그 타입에 대한 하나이상의 값으로 구성된다. 속성값을 저장할 때는 접근제어규칙에 의해 적어도 하나의 값을 가져야 하나 응답시에는 없을 수도 있다.

```
Attribute ::= SEQUENCE {
    type AttributeDescription,
    vals SET OF AttributeValue }
```

각 속성값은 집합내에서 중복되지 않으며 vals에 해당하는 속성값들의 순서는 정의되지 않았다.

4.6.8 매칭규칙 식별자

매칭규칙은 서버가 검색필터를 통해 요청된 Assertionvalue의 값과 그의 추상데이터 타입과의 비교하는 방법을 나타낸 것이다.

LDAP는 X.501의 매칭규칙과 같으며 오브젝트식별자에 의해 표현된다.

Matching Ruled : : LDAPstring

검색필터의 extensibleMatch을 사용하는 매칭규칙을 지원하는 서버는 매칭규칙을 서브스키마 엔트리에 리스트되어야 한다.

서버는 또한 matchingRuleuse속성을 이용 각 매칭규칙이 사용되어지는 속성타입을 리스트해야 한다.

4.6.9 결과메세지(result message)

LDAPResult는 연산결과가 성공 또는 실패인지를 나타내기 위해 사용된다. 서버는 최 종결과에 응답하기 위해 LDAPResult 타입 을 이용해서 리턴한다.

```
LDAPResult ::= SEQUENCE {
resultCode ENUMERATED {
    success (0),
    operationsError (1),
    protocolError (2),
    timeLimitExceeded (3),
    sizeLimitExceeded (4),
    compareFalse (5),
    compareTrue (6),
    authMethodNotSupported (7),
    strongAuthRequired (8),
        -- 9 reserved --
    referral (10), -- new
    adminLimitExceeded (11),
            -- new
    unavailableCriticalExtension
(12), -- new
    confidentialityRequired (13),
            -- new
    saslBindInProgress (14),
            -- new
    noSuchAttribute (16),
    undefinedAttributeType (17),
    inappropriateMatching (18),
    constraintViolation (19),
    attributeOrValueExists (20),
    invalidAttributeSyntax (21),
        -- 22-31 unused --
    noSuchObject (32),
    aliasProblem (33),
    invalidDNSyntax (34),
 -- 35 reserved for undefined isLeaf --
```

```
        aliasDereferencingProblem
                        (36),
              -- 37-47 unused --
        inappropriateAuthentication
                        (48),
        invalidCredentials (49),
        insufficientAccessRights (50),
        busy (51),
        unavailable (52),
    unwillingToPerform (53),
    loopDetect (54),
              -- 55-63 unused --
    namingViolation (64),
    objectClassViolation (65),
    notAllowedOnNonLeaf (66),
    notAllowedOnRDN (67),
    entryAlreadyExists (68),
    objectClassModsProhibited (69),
    -- 70 reserved for CLDAP --
    affectsMultipleDSAs (71), -- new
              -- 72-79 unused --
    other (80) },
    -- 81-90 reserved for APIs --
    matchedDN LDAPDN,
    errorMessage LDAPString,
    referral [3] Referral OPTIONAL }
```

16에서 21은 속성의 문제, 32,33,34,36은 이름문제, 48,49,50은 보안문제, 51에서 54는 서비스문제, 64에서 69와 71은 갱신문제를 언 급한다. 서비스의 옵션을 이용 사람이 읽을 수 있는 텍스트로 리턴할 수도 있다.

4.6.10 Referral

연결된 서버가 요구된 엔트리의 타겟을 가지고 있지 않으면 referral에러를 표시하고 resultcode값이 referral이면 참조될 엔트리가 있다는 것을 의미한다.

이는 다른 서버의 참조까지도 포함한다. 적어도 Referral에 하나의 URL이 존재해야 한다.

```
Referral ::= SEQUENCE OF
LDAPURL -- one or more
LDAPURL ::= LDAPString
-- limited to characters permitted in URLs
```

서버를 위한 URL의 표현은[9]의 규칙을 따른다.

4.6.11 control

control은 확장정보를 명시하는 방법이다.

```
Controls ::= SEQUENCE OF Control
 Control ::= SEQUENCE {
    controlType LDAPOID,
  criticality BOOLEAN DEFAULT
                    FALSE,
  controlValue OCTET STRING
            OPTIONAL }
```

control타입의 필드는 UTF-8형태로 인코드된 오브젝트 식별자로 control을 유닉하게 한다.

criticality필드는 참 거짓의 불린 값이다.

controlType과 critical필드의 값에 따라 연산의 수행, unsupported, 무시등의 행위가 나타난다. controlValue는 제어와 관련된 정보를 가지고 있다.

4.7 연산의 종류

4.7.1 Bind 연산

Bind연산은 클라이언트와 서버간에 인증정보를 교환하게 한다.

```
 BindRequest ::= [APPLICATION 0]
SEQUENCE {
    version INTEGER (1 .. 127),
    name LDAPDN,
    authentication AuthenticationChoice
```

```
}
AuthenticationChoice ::= CHOICE {
simple [0] OCTET STRING,
        -- 1 and 2 reserved
sasl [3] SaslCredentials }
SaslCredentials ::= SEQUENCE {
mechanism LDAPString,
credentials OCTET STRING
                OPTIONAL }
```

version은 프로토콜에 사용된 버전번호이다. 여기는 버전3을 사용하고 있다.

name : 클라이언트가 연결하고자 하는 디렉토리 오브젝트의 이름이다.

인증이 낮은 등급에서 행해주고 Credential란에 LDAPDN을 포함한 SASL을 사용할 때 익명의 바인딩을 실시한다[7].

인증 : 이름을 인증하기 위해 사용된 정보 바인딩 요구을 받자 말자 서버는 요청한 클라이언트를 인증하고 인증상태를 나타내는 상태를 가지고 클라이언트로 응답한다. 주로 NULL이 사용된다.

4.7.1.1 Bind의 연속적인 요구

SASL인증메카니즘을 위해서는 클라이언트가 여러번 BindRequest를 요청할 수 있다. 클라이언트가 Bind연산을 중지하고 싶으면 unbind하거나 연결상태를 끊으면 된다.

SASL바인딩 협상을 중지하고자 할 때는 SaslCredentials의 메카니즘 필드나 sasl외에 AuthenticationChoice에 각기 다른 값을 가지고 바인딩요구를 하여 할 수도 있다. 만약 클라이언트가 공백문자열로 sasl의 메카니즘 필드를 이용 바인더 요구를 보내면 서버는 결과코드에 authMethodNotSupported를 가지고 응답한다.

bind요구중에 신임을 변경시키고자 여러차례 바인드요구를 변경할 수 있다.

연속적인 바인드요구는 연결중인 bind 프로세스를 중지하게 하며 계속적인 bind요구의 신임이 무시되면 익명의 사용자로 취급한다.

4.7.1.2 인증과 다른 보안 서비스

단순한 인증 옵션은 cleartext 패스워드만을 이용해서 최소의 인증기능만을 제공한다. 그러나 이것은 낮은 수준의 인증이나 암호화도 되지 않을 때 개방된 네트워크에서는 추천할 만한 것이 아니다.

4.7.2 Bind 응답

Bind response은 인증에 관한 서버의 지시상태를 나타내며 다음과 같은 응답이 되돌려 진다.

success는 bind가 성공했음을 나타내고, operationserror은 서버가 내부 에러를 만난 경우, protocolerror은 식별할 수 없는 프로토콜 버전번호, authmethodnotsupported 식별할 수 없는 메카니즘 이름, strongAuth-Required SASL을 메카니즘으로 형성되어진 인증을 필요로 하는 서버, referal은 서버가 bind를 받아들일 수 없고 클라이언트가 다른 시도를 해야함, inappropriateAuthentication이 서버는 익명으로 bind되어지나 invalid-Credentials 잘못된 패스워드로 인해 인증이 진행될 수 없는 것, unavilable 서버가 꺼져있는 상태을 나타내며 만약 서버가 클라이언트가 요청한 프로토콜버전을 지원하지 못하면 resultcode에 protocolerror로 설정한다.

4.7.3 unbind 연산

unbind는 동작을 종료시키는 것을 나타내며 서버는 클라이언트가 모든 작업을 종료한 것으로 가정하고 모든 연결을 해재한다.

4.7.4 Unsolicited Notification

unsolicited notification은 서버에서나 서버와 클라이언트사이에 존재하는 비정상적인 상태를 나타내기 위해 응답이 없는 LDAPmessage를 나타낸다.

4.7.5 notice of Disconnection

notification은 서버가 에러에 의해 연결을 닫기 위해 클라이언트에게 충고하는데 사용되며 고객에 의해 unbind요구를 바라지 않는다. 이는 클라이언트에게 에러조건이나 네트워크상의 실패를 구별하기 위해 사용된다.

4.7.6 search 연산

검색 연산은 서버가 검색해주기를 바라는 클라이언트의 요구이다.

이 연산은 하나의 엔트리, 특정 엔트리의 아래의 엔트리, 전서브트리 엔트리를 읽는데 사용된다.

```
SearchRequest ::= [APPLICATION 3]
SEQUENCE {
  baseObject LDAPDN,
  scope ENUMERATED {
    baseObject (0),
    singleLevel (1),
    wholeSubtree (2) },
    derefAliases ENUMERATED {
    neverDerefAliases (0),
    derefInSearching (1),

derefFindingBaseObj (2),
                derefAlways
    (3) },
        sizeLimit       INTEGER
(0 .. maxInt),
        timeLimit
INTEGER (0 .. maxInt),
        typesOnly
BOOLEAN,
        filter Filter,
        attributes
AttributeDescriptionList }
  Filter ::= CHOICE {
    and [0] SET OF Filter,
    or [1] SET OF Filter,
    not [2] Filter,
    equalityMatch [3]
        AttributeValueAssertion,
    substrings [4] SubstringFilter,
    greaterOrEqual [5]
        AttributeValueAssertion,
    lessOrEqual [6]
        AttributeValueAssertion,
    present [7] AttributeDescription,
    approxMatch [8]
        AttributeValueAssertion,
    extensibleMatch [9]
        MatchingRuleAssertion }

SubstringFilter ::= SEQUENCE {
  type AttributeDescription,
  -- at least one must be present
  substrings SEQUENCE OF
            CHOICE {
        initial [0] LDAPString,
        any [1] LDAPString,
        final [2] LDAPString } }

MatchingRuleAssertion ::=
            SEQUENCE {
        matchingRule [1]
  MatchingRuleId OPTIONAL,
```

type [2] AttributeDescription
OPTIONAL,
matchValue [3]
AssertionValue,
dnAttributes [4] BOOLEAN
DEFAULT FALSE }

base object : 검색과 관련된 기본적인 오브젝트 엔트리인 LDAPDN

scope : 검색 범위 지시자, 이 필드의 가능한값은 X.511검색필드의 scope와 동일하다.

derefaliases : 검색에서 alias의 오브젝트가 어떻게 취급되는지를 나타내는 지시자 이며 자세한 내용은 X.501[6]에 정의되어 있다.

sizelimit : 검색의 결과로 리턴될 엔트리의 최대수에 대한 제한사항

timelimit : 검색에 허용된 최대시간

typeonly : 검색결과에 속성타입과 값을 함께 포함시키는지 아니면 속성타입만 포함시킬지 여부 지시

filter : 명명된 엔트리를 매칭하는 검색에서 수행할 조건에 대한 정의

여기서는 and, or, not연산자가 사용될 수 있다.

서버는 true, false, undefined의 3개의 논리를 가지고 filter를 평가한다.

present : 엔트리에 명시된 속성서술의 서브타입과 속성이 있으면 true이고 아니면 false가 된다.

attribute : 검색 필드와 매칭되는 엔트리로부터 리턴될 속성리스트

4.7.7 검색결과

클라이언트로부터 검색요구를 받고서 서버는 검색을 하면 검색결과는 SearchresultEntry, searchResultEntry, SearchResultReference, ExtendedResponse, SearchResultDone데이타타입을 이용 응답한다.

4.7.8 modify연산

modify연산은 서버가 수행해주기를 바라는 엔트리의 수정을 클라이언트가 요구할 때 사용된다.

ModifyRequest ::= [APPLICATION 6]
SEQUENCE {
object LDAPDN,
modification SEQUENCE OF
SEQUENCE {
operation ENUMERATED {
add (0),
delete (1),
replace (2) },
modification
AttributeTypeAndValues
} }

AttributeTypeAndValues ::=
SEQUENCE {
type AttributeDescription,
vals SET OF AttributeValue }

objecr : 수정되기를 바라는 오브젝트, 이 필드의 값은 수정되기를 바라는 엔트리의 DN을 가진다.

modification : 엔트리에 행해지기를 바라는 수정할 리스트이며 여기는 추가, 제거, 대치가 있다.

수정결과는 ModifyResponse를 통해 응답된다.

4.7.9 Add 연산

이 연산은 클라이언트가 디렉토리에 엔트리를 추가하고자하는 경우에 사용된다.

AddRequest ::= [APPLICATION 8]
SEQUENCE {
entry LDAPDN,
attributes AttributeList }
AttributeList ::= SEQUENCE OF
SEQUENCE {
type AttributeDescription,
vals SET OF AttributeValue
}
엔트리: 추가될 엔트리의 DN
속성: 추가하고자 하는 엔트리의 내용을 구성하는 요소

4.7.10 Delete 연산

이 연산은 디렉토리로부터 제거하고자 하는 엔트리가 있을 때 사용된다.
이 연산의 요구는 제거하고자 하는 엔트리 DN을 전달하여 수행한다.

4.7.11 Modify DN 연산

이 연산은 클라이언트가 디렉토리에 있는 엔트리의 이름요소들 중에서 가장 좌측의 것을 변경하고자 할 때 사용된다.
ModifyDNRequest ::= [APPLICATION 12] SEQUENCE {
entry LDAPDN,
newrdn RelativeLDAPDN,
deleteoldrdn BOOLEAN,
newSuperior [0] LDAPDN
OPTIONAL }
엔트리 : 변경되고자 하는 엔트리의 DN
newrdn : 엔트리의 새로운 이름을 나타내는 RDN
deleteoldrn : 변경되기 전의 엔트리 속성 값들을 그대로 존재시킬지 여부를 나타내는 불린 값
newsuperior : 현 엔트리의 상위엔트리가 될 DN
일반적으로 클라이언트가 임의로 서버들 사이에 엔트리나 서브트리를 이동시키는 것을 허용하지 않는다.
수행결과는 modifyDNResponse로 응답되며 이 연산을 수행중 LDAP와 DAP가 일치하지 않아서 에러가 발생시는 응답으로 affectMultipleDSA이 전달된다.

4.7.12 compare 연산

이 연산은 클라이언트가 디렉토리에 있는 엔트리와 assertion과의 비교를 하고자 할 때 사용된다.
CompareRequest ::= [APPLICATION 14] SEQUENCE {
entry LDAPDN,
ava AttributeValueAssertion }
엔트리: 비교하고자 하는 엔트리의 이름
ava : 엔트리의 속성과 비교될 assertion의 값 비교의 결과는 compareresponse로 응답한다. 각 디렉토리는 비교는 되나 읽힐 수 없는 속 성에 대해서는 접근제어를 해야 한다.

4.7.13 Abandon 연산

이 연산은 서버가 수행중인 연산을 클라이언트가 취소하고자 하는 경우에 이용한다.
AbandonRequest ::= [APPLICATION 16] MessageID
MessageID는 앞에서 요구된 연산의 종류이다.

4.7.14 Extended 연산

이 연산은 LDAP에 추가로 수행하고자 하는 연산으로 이 프로토콜외에는 다른 곳에는

적용되지 않는 연산이다.

ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
requestName [0] LDAPOID,
requestValue [1] OCTETSTRING OPTIONAL }

requestName : 오브젝트 식별자의 십진수와 점을 이용한 표기의 값이다.

requestvalue : 요구에 정의된 폼에 대한 정보

이 연산을 수행후 서버가 응답시는 ExtendedResponse를 포함한 LDAPMessage를 이용하여 응답한다.

ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
COMPONENTS OF LDAPResult,
responseName [10] LDAPOID OPTIONAL,
response [11] OCTET STRING OPTIONAL }

요구된 이름을 서버가 인식할 수 없을 때는 프로토콜 에러로 값이 응답된다.

4.8 구현시 고려사항

4.8.1 오버헤드의 감소

프로토콜의 오버헤드를 줄이기 위해 ASN.1의 BER기법을 이용 인코드되고 추가로 인코딩의 길이 의 제한, OCTET STRING는 원어에만 사용, 불린 타입의 true값은 16진 수로 "FF", 타입의 값은 디폴트값의 유지 등을 통해 오버헤드를 최소화한다.

4.8.2 전달 프로토콜

LDAPMessage의 PDU는 TCP의 바이트스트링과 직접 매핑되어야 하며 서버의 구현은 프로토콜 listener에게 포토 389를 제공하기 를 추천되어지며 필요시는 다른 포토를 사용 할 수 있으며 클라이언트는 TCP와 포토를 일치시켜야 한다.

4.8.3 구현될 속성

서버는 프로토콜에서 의무적인 속성의 타입 이름을 인식할 수 있도록 구현되어야 하며[5]의 신텍스를 이용하여 구현되어야 한다.

4.8.4 구현될 보안서비스

보안 서비스를 위해 인증메가니즘을 제공해야 하며 cleartext와 같이 단순인증과 SASL메카니즘을 이용 무결성과 프라이버시를 협상할 수 있어야 한다.

그리고 서버는 클라이언트의 bind요구에 의해 클라이언트를 인증하면 신용상태를 응답할 수 있어야 한다.

4.9 검색방법

트리의 각노드는 디렉토리의 오브젝트(object) 로 디렉토리 엔트리와 같은 의미이며 LDAP의 오브젝트는 속성의 집합으로 표현되고 이는 objectclass에 의해 결정된다.

어떤 노드로 가는 경로는 DN을 형성하며 DN에 의해 특정 오브젝트로 찾아 갈 수 있다.

LDAP의 검색(search)은 다음의 3가지 페러미트에 의해 결정된다.

- 검색에 사용되는 기본적인 DN검색을 할 수 있는 서브트리의 탑 노드를 명시한 DN이다.
- 검색범위(search scope)

DN과 검색범위는 어느 노드를 검색해야 하는지를 검색하게 된다.

· base DN(단지 DN만)
· base DN과 DN아래의 모든 노드

· DN아래의 있는 모든 노드

- 검색 필터

검색 필터는 디렉토리 오브젝트의 속성 값에 기초한 검색 정보표현이다. 즉 검색 필터는 검색된 오브젝트 중 어느 디렉토리 오브젝트가 검색과 실제적으로 일치하는 지를 결정하게 된다. 검색 영역내의 디렉토리 오브젝트의 속성값이 검색 필터와 일치할 때 검색은 성공하게 된다.

4.9.1 LDAP의 검색 보기 예

LDAP의 검색에서 기본적인 DN은 "OU=Accounting, O=Acme Widget Company, C=US"이고 검색 범위는 subtree scope이며 검색 필터는 "N=John Doe"을 사용해서 search를 하면은 조직내의 common name의 속성이 일치하는 모든 디렉토리 엔트리가 응답되어 진다.

4.9.2 X.509와 관계

같은 개체의 LDAP엔트리를 찾기 위해서 mapping되는 오브젝트는 인증서에서 Subject DN으로 명시된다. 이는 Subject DN의 값을 그림8과 같이 LDAP의 search로 전환하는 알고리즘이 필요로 한다[9].

4.9.3 보관소 정보 검색

보관소에 있는 정보를 검색하고, 관리하기 위한 기능은 크게 4가지로 구분되어진다[2].

repository read : 검색 개체인 말단 개체나 인증기관이 보관소에 있는 정보에 대해 이름 엔트리(entry)를 알고있는 경우에 그 엔트리(entry)를 이용하여 필요한 정보를 검색하는 것이다. 이때 검색되는 정보는 인증서, CRL (Certificate Revocation List), 기타 다른 정보이며 size와 time limit는 사용되지 않는다.

① bindRequest() / BindResponse()

② SearchRequest() / SearchResponse()

③ UnbindRequest()

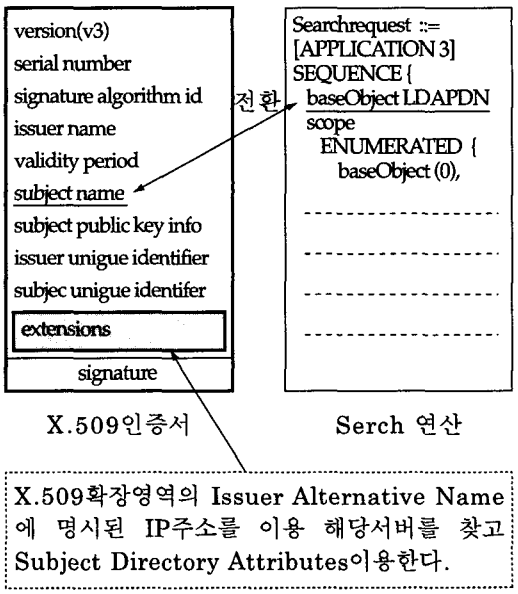


그림8 X.509와 Search 연산

repository search : 이름 엔트리를 잘 모르면서 다른 관련 정보를 알고 있을 때 보관소의 정보를 필터링하는 기능이다. 검색되는 정보는 명시된 필터 정보와 일치하는 엔트리로부터의 인증서가 된다. 이때는 raed에서와는 달리 필터의 기능이 강화되고 size와 time limit는 사용되어 진다[8].

① bindRequest() / BindResponse()

② SearchRequest() / SearchResponse()

③ UnbindRequest()

repository modify : 보관소의 정보를 추가하고, 제거, 수정하는 연산으로 인증서 관리와 관련된 디렉토리에서는 클라이언트와는 관계가 없고 인증기관의 관리 기능과 관련된 요구기능이다.

① Modify Operation

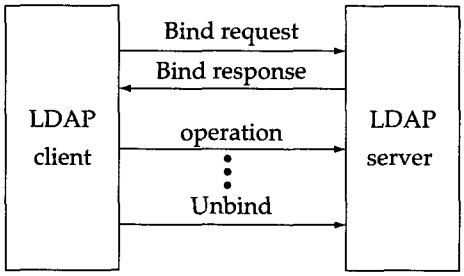
② Add operation

③ Delete Operation
④ Modify DN operation
⑤ Compare operation
⑥ Abond Operation
⑦ Extended Operation

referal : referal은 다른 디렉토를 참조하라는 값으로 이 값을 클라이언트에 전달함으로 서버의 오버헤드를 감소시킨다.

4.9.4 프로토콜 수행절차

referal을 제외한 연산에서 상호연결을 위해 Bind request, Bind response를 기본적으로 수행되며 연결 해제를 위해서는 Unbind가 수행된다.



[그림8] 사용자의 요구수행과정

5. 결 론

인터넷이 발달하면서 이를 이용하여 전자상거래, 전자우편, 메시지 교환 등의 업무를 수행하고 있으며 그 응용영역이 더욱 확장되고 있다. 이와 더불어 인터넷의 이용은 많은 위험요소가 존재하는데 이를 극복하는 방법의 하나로 인증기관을 통한 전자서명기술을 응용하여 메시지의 무결성과 부인방지, 기밀성을 달성하고자 하고 있으며 인증기관을 구축하는데에는 디렉토리 시스템과 디렉토리에 접근하는 프로토콜의 이용은 필수적이다.

디렉토리에 접근하는 프로토콜로는 FTP프로토콜, OCSP프로토콜 등이 있으나 인증기관과 관련해서는 다른 인증기관과의 상호운영성을 확보하기 용이하고 X.500의 DAP프로토콜에 비해 오버헤드가 감소된 LDAP프로토콜이 많이 이용되고 있다.

그러므로 본 논문에서는 LDAPv3버전의 RFC2251을 바탕으로 프로토콜에 대해 알아보았다. 국내에서도 인증기관을 구축하는데 LDAP프로토콜을 이용하므로 국제적인 전자상거래를 대비해야 하고 또한 LDAP디렉토리 시스템과 관련해서 심층깊은 연구가 병행되어야 한다.

참고문헌

[1] ITU-T Rec. X.500, "The Directory: Overview of Concepts, Models and Service", 1993.

[2] Yeong, W., Howes, T., and S. Kille, "Lightweight Directory AccessProtocol", RFC 1777, March 1995.

[3] ITU-T Rec. X.680, "Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation", 1994.

[4] Kille, S., Wahl, M., and T. Howes, Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

[5] Wahl, M., Coulbeck, A., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[6] ITU-T Rec. X.501, "The Directory: Models", 1993.

[7] Meyers, J., "Simple Authentication and Security Layer", RFC 2222, October 1997.

[8] Sharon Boeyen, Russell Housley, Tim Howes, Michael Myer, Patrick Richard, "Optional Protocol(draft-ietf-pkix-ipk2opp - 00.txt)," http://www.ietf.org/proceedings/97apr/sec/pkix-20, 1997

[9] Structured Arts Computing Corporation, "Certificate to LDAP Mapping in Netscape Servers," http://www.structuredarts.com/ edu/cert2ldap.html, 1997

[10] T.howes, M.Wahl, S.Kille, "Light-weight Directory Access Protocol(v3) RFC2251," ftp://ds.internic.net/rfc/rfc2251. txt, 1997

[11] university of Michigan, "Introduction to SLAPD and SLURPD administrator's guide," http://www.umich.edu/~dirsvcs/ldap/doc/guides/slapd, 1996

[12] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1 : 1993.

[13] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2044, October 1996.

□ 著者紹介

최 진 주

1989년 대구대학교 영문학과 졸업(학사)
1999년 2월 국방대학원 전자계산학과 졸업

※ 주관심분야 : 전자상거래 보안, 전자서명, 암호프로토콜