通信情報保護學會誌
第8卷 第1號, 1998. 3

디렉토리 서비스에 관한 고찰
-제 1 부 : 디렉토리 구성, 모델 및 서비스-
(X.500, X.501, X.511)

정 권 성*, 김 영 수*, 이 형 규*, 조 동 욱*, 원 동 호*

요 약

분산된 환경에서의 정보의 효율적인 사용을 위하여 이를 체계적으로 구축할 필요가 발생하였으며 더욱이 현재 공개키 기반구조에 대한 필요성이 증대되고 공개키 관리가 중요 문제로 다루어지고 있다. 이를 위한 한가지 방법으로 디렉토리가 제안되었고, 이는 현재 ISO/IEC 9594 시리즈로 표준화되었다. 본 고에서는 디렉토리의 개념과, 정보의 구성방식 및 지원 서비스에 대하여 고찰하였다.

Ⅰ. 서 론

현재 우리는 정보화 사회를 지나 정보사회에 들어서 있다. 수많은 정보들이 발생되고 있으며 이들 정보들은 대부분 한 곳에 집중되어 있는 것이 아니라 각각의 영역에 별도로 분산되어 있다. 다행이 컴퓨터 네트워크의 발달로 인하여 분산된 정보들을 보다 용이하게 다룰 수 있게 되었고 이를 활용하기 위하여 실세계에서 탈생되는 정보들은 디지털 형태로 저장되기 시작하였고 이를 체계적으로 구성할 필요성을 느끼게 되었다. 이에 분산된 환경에서 구축된 정보들을 좀 더 체계적으로 구성하여 사용자가 필요한 정보를 손쉽게 획득할 수 있도록 하기 위하여 디렉토리라는 개념이 대두되었다.

디렉토리는 실세계의 정보를 식별 가능한 명칭을 사용하여 하나의 단위로 구성한 뒤 이를 각각의 영역에서 보유한 후, 사용자의 요구가 있을 시에는 디렉토리 내의 각 영역에서 해당요구에 대해 수행하고, 담당 영역에 대해서 요구를 수행하지 못하는 경우 분산된 다른 영역과의 상호 협조하여 그 결과 또는 오류를 사용자에게 되돌린다. 결과적으로 디렉토리에 분산된 정보는 하나의 단일한 것으로 다루어져 사용된다.

X.500이란 국제 표준화 그룹 ISO/ITU-T에서 제정한 디렉토리에 관련된 표준들의 묶음을 대표하는 명칭으로, 이는 컴퓨터 응용프로그램이나 네트워크 플랫폼에 관계없이 광범위한 디렉토리 서비스를 위한 정보 모델과 프로토콜들을 포함하고 있다. 제 1판이 1988년에 제정되었고 이를 좀 더 보완하여 1993년 11월에 제 2판이 제정되었으며 제 3판이 1997년 6월 정되어 있는 상태이다. 디렉토리 표준안은 워낙 그 응용 범위가 방대하여 현재 세계 각

* 성균관대학교 정보공학과

국에서는 X.500기반의 디렉토리 시스템에 대한 연구가 활발히 진행되고 있으며 표준안을 구현한 구체적인 디렉토리 서비스도 제공하고 있다.

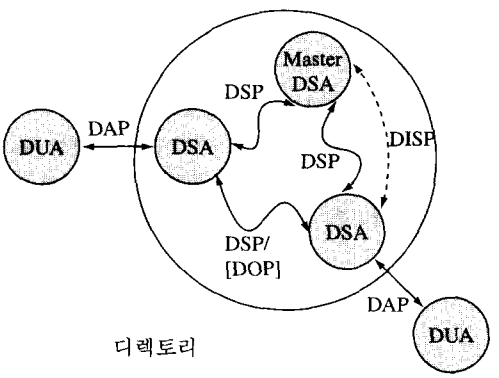
본 고에서는 디렉토리에 관한 고찰 중 제 1부로써 디렉토리 시스템을 구성하는데 있어서 알아야할 일반적인 사항을 살펴보고 이에 대한 예를 다룸으로써 디렉토리에 대해 고찰하고자 한다. 2장에서는 디렉토리가 어떻게 기능 하는 가에 대해 설명을 하고 3장에서는 정보를 디렉토리에서 어떻게 구성, 관리하는 가에 대한 것을 설명하며, 4장에서는 사용자가 디렉토리에게 요구하는 서비스들에 대해서 알아보고, 5장 결론으로 맺는다.

Ⅱ. 디렉토리 기능 구조

디렉토리는 실생활에 존재하는 정보들을 사용자가 사용하는데 용이하도록 체계적으로 구성한 것이다. 따라서 우선적으로 디렉토리에는 디렉토리를 구성하는 정보들이 필요하다. 다음으로 디렉토리의 정보는 결국 사용자가 이용하기 위한 것이므로 사용자를 위한 응용 프로세스 또한 필요하다. 디렉토리에 사용자의 요구를 전달하고, 디렉토리로부터 사용자의 요구에 대한 응답을 받는 응용 프로세스는 디렉토리와 사용자와의 인터페이스 역할을 수행한다.

이 응용 프로세스가 DUA(Directory User Agent)이다. 또한 디렉토리 내에 존재하는 정보들은 각각의 시스템에 분산되어 관리되어 운영되기 때문에 이를 디렉토리 내에서 디렉토리 프로토콜로써 디렉토리 정보를 보유하고 있는 시스템들과 상호 협력하여 사용자의 요구를 수행하는 응용 프로세스가 존재하여 최종적으로 사용자는 분산된 전체 디렉토리를 하나의 지역 디렉토리로 간주하여 사용할 수 있다. 분산된 디렉토리 시스템에서 사용자의 요구를 디렉토리 프로토콜에 따라 수행하는 응용 프로세스들이 DSA(Directory System Agent)이다.

DSA는 사용자가 지시한 명령이나 요구들에 대해서 디렉토리를 통해서 그것을 수행하고 그것의 결과를 사용자에게 되돌리는 역할을 하며 디렉토리에는 여러 개의 DSA가 존재할 수 있다. DUA는 DSA에 있는 접근점(access point)을 통해서 디렉토리에 접근을 하게 되는데 하나의 DSA는 하나 이상의 접근점을 가지고 있다.



〈그림 1〉 디렉토리 구성

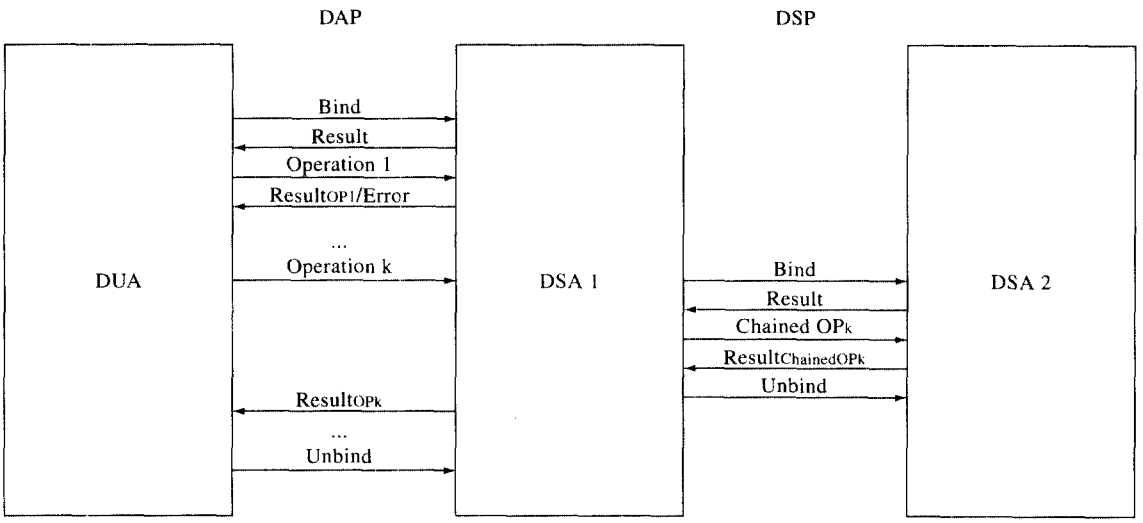
사용되는 프로토콜은 다음과 같다.

- DAP(Directory Access Protocol) : 사용자가 디렉토리에 서비스를 요구하는 DUA와 DSA간의 프로토콜.
- DSP(Directory System Protocol) : 사용자가 요구한 서비스를 수행하기 위한 두 DSA간의 프로토콜.
- DISP(Directory Information Shadow Protocol) : 한 DSA가 보유하고 있는 정보를 다른 DSA에게 복사하여 이를 사용하는 경우 이때 복사를 해준 DSA가 주기적으로 또는 특정한 상황(예를 들면, 복사 정보의 수정) 발생시 복사해 주어야 하는데, 이런 복사된 정보를 두 DSA간에 주고 받기 위해

사용되는 프로토콜.

- DOP(Directory Operational Binding Management Protocol) : 두 DSA간에 관리적인 동작 관계가 설정되어 있는 경우 이들 DSA간의 통신을 위해서 사용되는 프로토콜.

다음 그림은 사용자의 요구가 어떻게 수행되는 가를 간략하게 보여 주고 있는데 여기서 DUA와 DSA 1간의 프로토콜이 DAP이고 DSA 1과 DSA 2간의 프로토콜이 DSP이다.



〈그림 2〉 사용자 요구의 수행 과정

위에서 보듯이 Operation K를 DSA 1이 수행하지 못하는 경우 이를 다시 DSA 2와 연결한 후 DSA 2에서 수행한다. 이처럼 디렉토리 서비스를 지원하기 위해서는 DSA들간의 상호 협력이 필요하다. 자체 내에서 디렉토리 서비스를 수행할 수 없는 경우 다음의 방법을 이용하여 서비스를 수행한다.

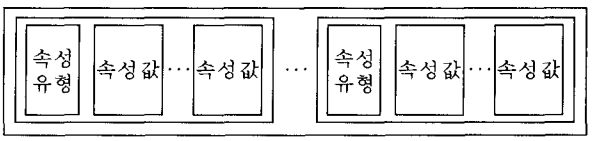
- referral : 요청자에게 요청 서비스를 수행할 만한 다른 DSA 관련 정보를 알려준다.
- chaining : 요청 받은 서비스 수행을 다른 DSA에게 넘겨준다. 수행결과는 연결 차례 역순으로 전달된다.
- multichaining : 하나의 DSA가 동시에 여러 다른 DSA에게 요청 받은 서비스 수행을 넘겨준다.

Ⅲ. 디렉토리 구성

실세계에 있는 사용자 정보를 디렉토리로 구축하기 위해서는 여러 가지 모델들이 필요하다. 가장 중요한 것이 사용자 정보를 디렉토리에 담는 것이고, 다음으로 구축된 사용자 정보를 관리하는 것이며, 마지막으로 이를 실질적으로 보유하고 운용하는 DSA들을 어떻게 구성하는 것이다. DIT의 성질뿐만 아니라 DSA와 관련된 모든 정보를 담아야 하는 것이 DSA 정보 트리이다. 각각의 DSA가 디렉토리 정보 트리를 어떻게 구성할 것인가는 디렉토리 구서의 중요한 한 부분이다. 본 절에서는 이를 구성해 나가는 과정을 살펴본다.

1. 정보의 표현

디렉토리는 디렉토리 사용자들의 정보를 체계화시킨 것으로 사용자들의 정보를 모두 모아 둔 것이 DIB(Directory Information Base)이다. 하지만 단순히 정보를 모아 놓은 것만으로는 정보를 이용하기 힘들다. 따라서 이 DIB에 있는 정보들을 트리의 형태로 구성하여 DIT(Directory Information Tree)를 구성한다. 이는 사용자들의 정보를 계층성을 갖는 트리의 형태로 체계화한 것이므로 정보의 이용 또는 유지 관리하는데 상당히 유용하다. 각 사용자 정보는 엔트리를 통해 나타나며, 이 엔트리는 하나 이상의 속성(Attribute)들로 구성되어 있고, 각각의 속성들은 속성유형(attribute type)과 하나 이상의 속성값들로 구성되어 있다.

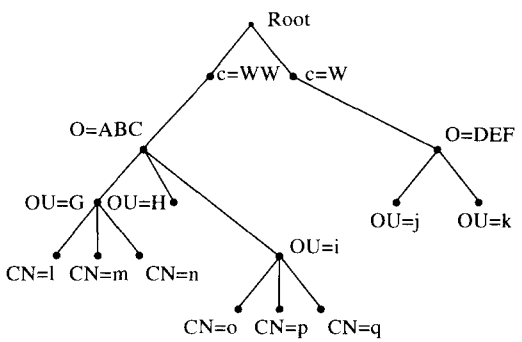


〈그림 3〉 엔트리 구성

또한 엔트리들은 명칭을 통해서 식별되는데 엔트리의 명칭은 DIT의 각 계층에 있는 엔트리들을 식별하도록 하는 RDN(Relative Distinguished Name)이 있고, 루트에서 해당 엔트리까지의 RDN의 순차열로 구성되는, 전체 DIT를 통틀어 식별 가능한 DN(Distinguished Name)이 있다. 별명(alias) 명칭도 존재하는데 이 경우 하나의 경로이외의 다른 경로로 엔트리에 접근하고자 할 때 사용된다. 즉, 하나의 엔트리가 여러 개의 이름을 가지고 싶은 경우({C=VV,O=DEF,OU=K}, {C=WW,O=ABC,OU=I}) 원래의 엔트리 {C=WW,O=ABC,OU=I}에 실질적인 데이터를 저장하고 나머지 별명 엔트리 {C=VV,O=DEF,OU=K}는 원래 엔트리만을 가리키는 포인터만을 가지게 된다.

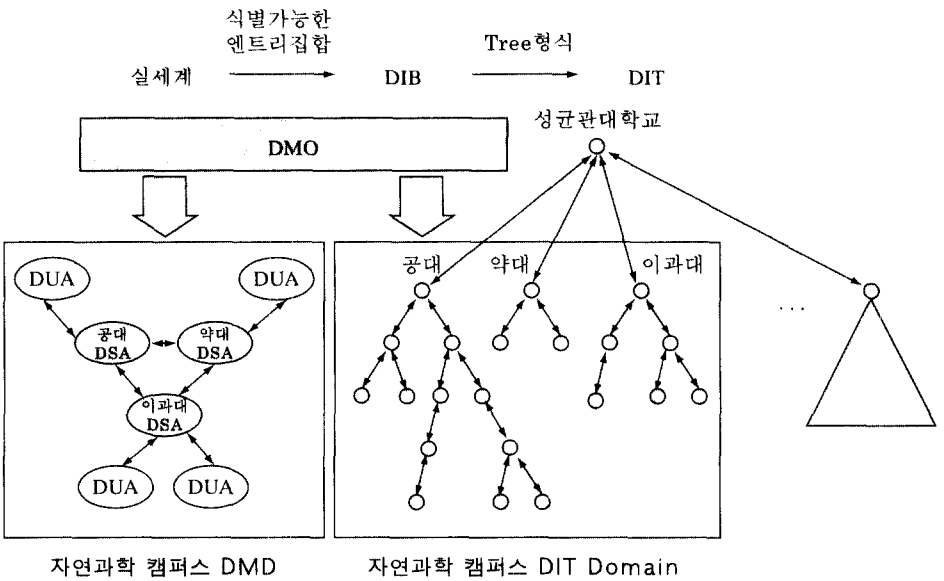
2. 정보의 관리

디렉토리에 보유되는 정보들은 하나 이상의 관리기관들에 의해서 관리되며, 이들 관리 기관들은 디렉토리의 주어진 영역 내에 존재하



〈그림 4〉 디렉토리 정보 트리

는 정보들에 대해서 관리하고 이에 대해 책임을 진다. 관리해야 하는 디렉토리의 일정한 영역을 DMD(Directory Management Domain)이라고 하는데 이는 하나 이상의 DSA와 DUA로 구성된다. 또한 이들 DSA들이 지니고 있는 디렉토리 정보들을 모아 놓은 것이 DIT 도메인(DIT domain)이라 한다. 결국 하나의 DMD와 DIT 도메인은 일대일 대응관계를 갖게 된다. 이들은 하나의 관리기관 DMO(Directory Management Organization)에 의해서 관리되며 관리 기관이 공공 관리 기관인 경우 ADDMO(Administrative DMO)라 하고 사설 관리 기관인 경우 PRDMO(Private DMO)라고 한다.



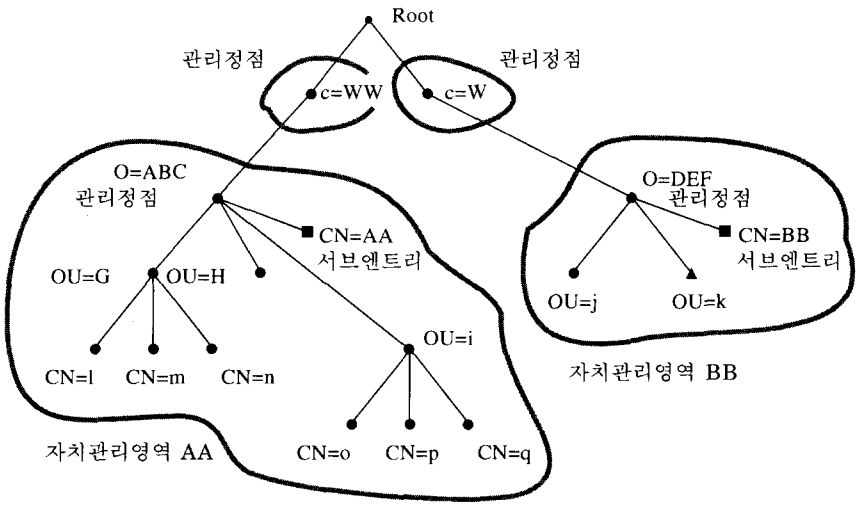
〈그림 5〉 디렉토리 정보의 관리

하나의 DIT 도메인은 여러 개의 서브트리들로 구성이 되어 있는데 이들을 관리 영역(Administrative Area)이라 하고 관리 영역은 하나의 관리 기관으로부터 관리를 받는데 관리기관의 성격에 따라 관리 영역이 특성이 결정된다.

관리 영역은 크게 세 가지로 볼 수 있다.

- 자치 관리 영역(autonomous administrative area)
  이것은 DIT 영역에 포함되어 있는 서브트리로 자치 관리 기관이 서브트리 전체를 관리하게 된다.
- 특정 관리 영역(specific administrative area)
  자치 관리 영역에 있는 엔트리의 특성상 특정 관리 기관에서 관리되어야 하는 경우로 특정 관리 영역은 자치 관리 영역의 일부분에 대해서 관리의 책임을 위임받아 관리한다. 결국 하나의 자치 관리 영역은 하나 이상의 특정 관리 영역으로 분할된다.
- 내부 관리 영역(inner administrative area)
  이것은 자치 관리 영역이나, 특정 관리 영역에 중첩되어 있다. 이 영역의 엔트리들은 내부 관리 영역으로 위임된 기능에 대해서는 내부 관리 기관의 관리를 받고, 그 이외의 사항에 대해서는 자치 관리 기관 또는 특정 관리 기관의 관리를 받는다.

각각의 관리 영역에 해당하는 서브트리의 루트는 관리정점(Administrative Point)라 하고 여기에 위치하는 엔트리가 관리 엔트리(Administrative Entry)이다. 관리 엔트리는 하나 이상의 서브엔트리(Subentry)를 가지는데 서브엔트리는 그 밑에 다른 엔트리를 두지 않는 특수한 엔트리로 관리 엔트리가 해당 영역을 관리하는데 필요한 정보(예를 들어, 영역 내에서의 접근 제어 범위)를 포함한다.



〈그림 6〉 디렉토리 관리 영역

위 그림에서는 4개의 자치관리 영역이 존재하고 각 영역의 루트 {C=WW}, {C=VV}, {C=WW, O=ABC}, 그리고 {C=VV, O=DEF}는 관리 정점이다. 특히 자치관리영역 AA, BB는 그 영역에 대한 정보를 각각의 서브엔트리 {C=WW, O=ABC, CN=AA}, {C=VV, O=DEF, CN=BB}에 담고 있다.

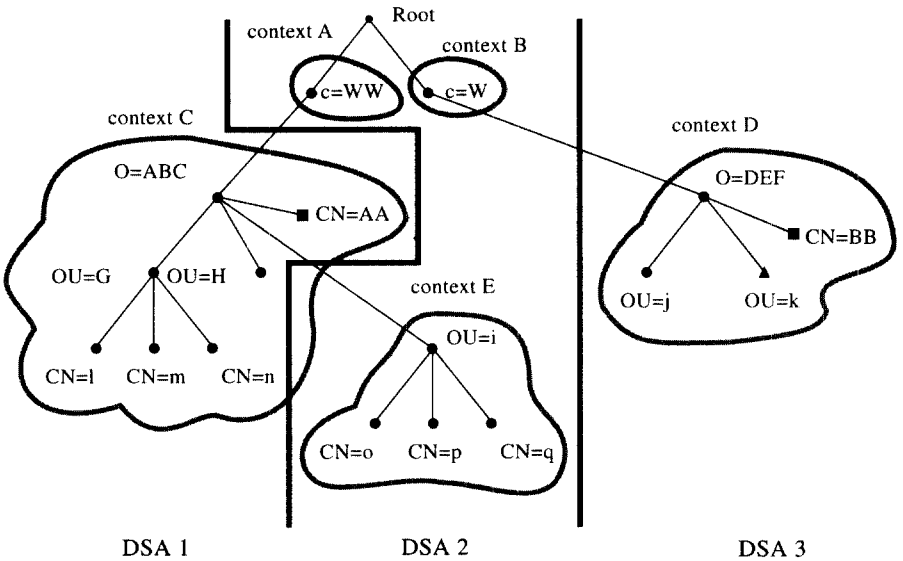
3. 분산 환경에서의 정보 표현

디렉토리 정보 모델이란 디렉토리를 구성하는 정보를 어떻게 표현하는가에 대한 것을 나타낸다. 그러나 디렉토리는 하나 이상의 DSA로 구성되고 있고 정보들을 이들 DSA에 분산되어 있다. 따라서 사용자가 디렉토리를 사용하기 위해서는 분산된 정보를 보유하고 있는 DSA를 분산된 환경에 맞게 구성하여야 한다.

DSA들은 일단 사용자로부터 요구를 받아 이를 수행하는데 자신이 보유하고 있는 정보 내에서 사용자의 요구를 수행하지 못할 경우 분산된 환경에서의 다른 DSA들이 지니고 있는 정보를 참조하게 되는데 DSA 정보 트리에서 이러한 정보들을 표현하고 있다.

<그림 7>에서는 <그림 6>에서 나타내는 DIT를 3개의 DSA가 분산시켜 보유하고 있고 이들 정보들은 논리적으로 5개의 콘텍스트들로 구성하고 있다.

각각의 DSA는 DIT의 엔트리에 대한 정보를 포함한다. DSA는 보유하고 있는 디렉토리 정보뿐만 아니라 DSA 동작에 관련된 정보를 표현하기 위해서 DIT와는 별도로 DSA 자체 내에서 트리의 형태로 정보를 표현하는데 이



〈그림 7〉 DIT와 DSA

DSA 정보 트리(DSA Information Tree)는 DSE(DSA Specification Entry)들로 구성된다. 분산된 환경에서 가장 중요한 것이 하나의 DSA에서 요구를 수행하지 못하는 경우 다른 DSA를 참조 후 사용자의 요구를 수행하는데 DSE에는 DIT 엔트리의 정보와 이러한 참조에 관련된 DSA의 정보를 포함하고 있다. DSA가 사용하는 정보 참조에는 다음과 같은 것들이 있다.

- 자기 참조
  DSA 자신에 대한 정보(예를 들면, DSA의 접근점)를 나타낸다.
- 상위 참조
  참조를 원하는 엔트리가 상위 DSA에 포함되어 있는 경우 이를 참조하기 위하여 상위 DSA에 접근한다.
- 직속 상위 참조
  DSA내의 명칭 문맥에 대해서 직속 상위인 DSA를 참조하는 경우로, 상위 참조에서는 DSA 상/하위 관계로부터의 상위 DSA를 참조하는 것에 반해 직속 상위 참조는 명칭 문맥의 직속 상/하위 관계로부터의 상위 DSA를 참조한다. 또한 직속 상위 참조는 두 DSA간에 계층적 동작 결합이 이루어진 상태에서만 이루어진다.
- 하위 참조
  참조하고자 하는 엔트리/엔트리사본을 하위 DSA가 보유하는 경우 이를 참조하기 위하여 하위 DSA를 참조한다.
- 비특정 하위 참조
  엔트리/엔트리사본을 보유하고 있는 DSA는 알려져 있지만 엔트리/엔트리사본의 RDN이 알려져 있지 않은 경우에 사용되는 참조로 해당 하위 DSA에 대한 정보(DSA의 접근점)를 나타낸다.
- 교차 참조
  두 DSA의 상호간의 참조로서, 참조하는 두 DSA간에는 계층적인 상위/하위 관계가 존재하지 않는다.

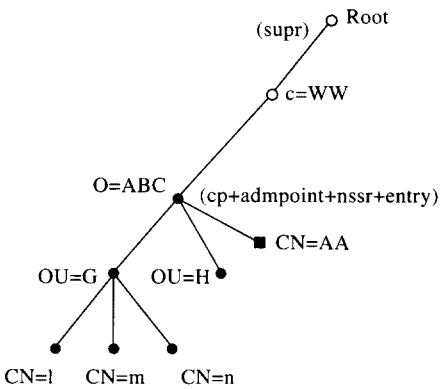
- 공급자 참조
  정보 복사에서 나타나는 참조로써, 복사된 정보를 사용하는 쉐도우 사용 DSA가 정보를 제공하는 마스터 DSA에 대한 정보를 보유한다.
- 소비자 참조
  정보 복사 관계에서 복사 정보를 제공받는 DSA에 대한 정보를 나타내는 것으로 마스터 DSA가 보유한다.

DSE의 유형은 보유하는 엔트리 정보 및 DSA 운용 정보에 따라 결정되는데 이는 DSE의 dseType 속성 중 DSEType에서 이를 나타낸다. 다음은 DSEType이 가질 수 있는 값들로서 이들은 OR비트열로 구성된다.

DSEType															
root	glue	cp	entry	alias	subr	nssr	supr	xr	adm Point	sub entry	shadow	-	imm Supr	rhob	sa

〈표 1〉 DSE 의 유형 - 'DSEType' 의 구성

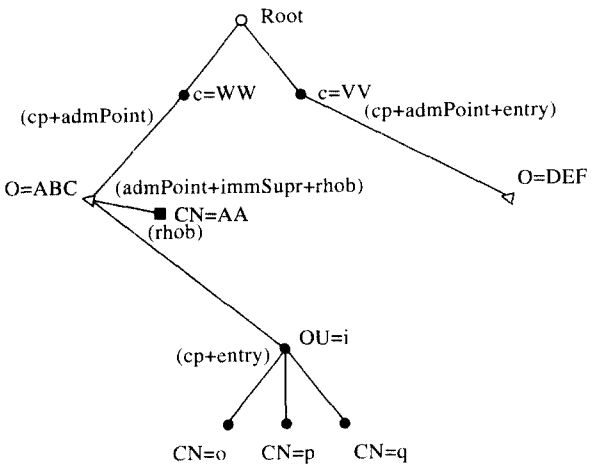
<그림 7> 에서 나타난 DSA의 정보트리를 구체적인 예로 살펴보면 다음과 같다.



〈그림 8〉 DSA 1 정보 트리

- DSA1이 보유하고 있는 정보는 자치관리영역 AA에서 명칭문맥 C만을 포함한다.
- root : DSA2가 DSA1의 상위 DSA이므로 상위 참조를 위해서 supr 유형을 갖는다.
- O=ABC : 자치관리영역 AA의 관리 정점이므로 admPoint 유형을 갖는다.
  명칭문맥 C에 대한 문맥접두사이므로 cp 유형을 갖는다.
  명칭문맥 E에 대해서 비특정하위 참조가 이루어지므로 nssr 유형을 갖는다.
  {C=WW,O=ABC}엔트리에 대한 정보를 포함하여 entry 유형을 갖는다.
- CN=AA : 자치관리영역 AA의 관리 정점 O=ABC 에 대한 서브엔트리이다.

다음 그림은 DSA2에 대한 DSA 정보트리이다.

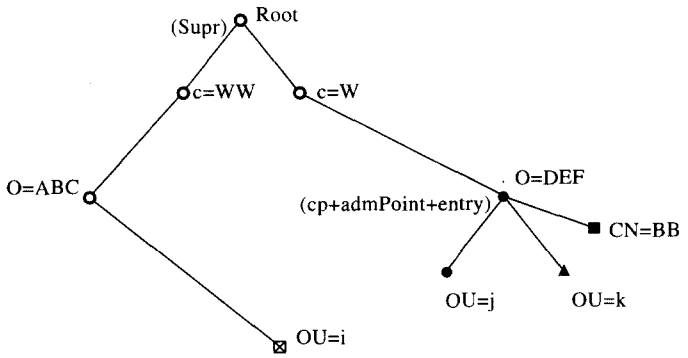


〈그림 9〉 DSA 2 정보 트리

- DSA2가 보유하고 있는 정보는 명칭문맥 A, B와 E를 포함한다.
- root : DSA2가 최상위 DSA이므로 상위 참조는 없다.
- C=WW, C=VV : 자치관리영역 관리 정점이므로 admPoint 유형을 갖는다.
  명칭문맥 C에 대한 문맥접두사이므로 cp 유형을 갖는다.
- O=ABC : DSA1에 대한 하위 참조는 나타낸다.
  자치관리영역 AA에 대한 관리 정점이므로 admPoint 유형을 갖는다.
  명칭문맥 E는 명칭문맥 C에 대해서 직상위 참조를 이루어 immSupr유형을 갖는다.
  명칭문맥 C와 E가 직상위참조를 이루기 위해서 이전에 계층적 동작 관계가 설정되므로 rhob 유형을 갖는다.
- O=DEF : DSA1에 대한 하위 참조는 나타낸다.
- OU=I : 명칭문맥 E에 대한 문맥접두사로써 cp 유형을 갖는다.
- CN=AA : 자치관리영역 AA의 관리 정점 O=ABC 에 대한 서브엔트리이다.

다음은 DSA3에 대한 정보트리이다.

- DSA2가 보유하고 있는 정보는 자치관리영역 BB의 명칭문맥 D를 포함한다.
- root : DSA3이 DSA1의 상위 DSA이므로 상위 참조를 위해서 supr 유형을 갖는다.
- O=DEF : 자치관리영역 BB의 관리 정점이므로 admPoint 유형을 갖는다.
  명칭문맥 D에 대한 문맥접두사이므로 cp 유형을 갖는다.
  {C=VV,O=DEF}엔트리에 대한 정보를 포함하여 entry 유형을 갖는다.
- CN=BB : 자치관리영역 BB의 관리 정점 O=DEF 에 대한 서브엔트리이다.
- OU=I : OU=K는 OU=I의 별명이므로 이를 포함하는 명칭문맥 E와 상호참조 관계가 있다.



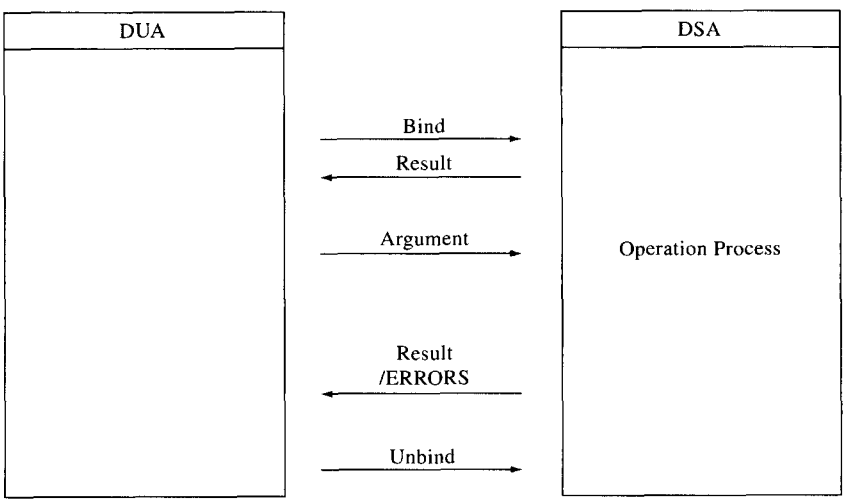
〈그림 10〉 DSA 3 정보 트리

Ⅳ. 디렉토리 서비스

디렉토리가 제공하는 모든 서비스들은 DUA의 요구에 대해 응답하는 형식을 가진다. 즉 DAP에서 지원하는 서비스들이 디렉토리 서비스이며, 사용자가 디렉토리에 요구하는 것들은 크게 질의에 대한 부분과 수정에 대한 부분이다. 이들 질의/수정에 대한 요구는 서비스 제어, 보안 파라미터 또는 필터로 인해 제한되어 질 수 있다. 디렉토리는 항상 요구에 대해 결과를 보고하는데 비정상적인 결과들에 대해서는 오류가 보고된다. 다음 그림에서는 일반적인 디렉토리 서비스의 수행과정을 나타내고 있다.

1. 디렉토리 질의 서비스

디렉토리 질의 서비스에는 디렉토리에서 지정된 엔트리에 대한 정보를 읽어오는 '읽기(Read)' 서비스, 엔트리의 속성유형/속성값을

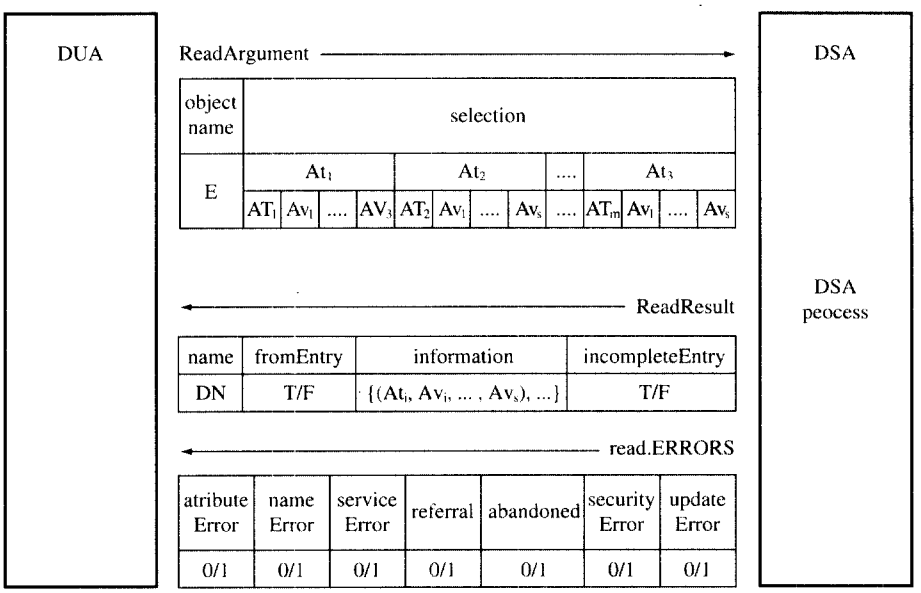


〈그림 11〉 디렉토리 서비스의 수행 절차

비교하는 '비교(Compare)' 서비스, 지정 엔트리의 직속하위 엔트리 목록을 구하는 '목록(List)', 그리고 디렉토리에서 원하는 엔트리 정보를 찾는 '검색(Search)' 서비스 등이 있다.

1.1 읽기(Read)

읽기 서비스는 지정된 엔트리의 속성 또는 속성값들을 결과로 반환한다.



E : 식별명칭 At_i : 속성 AT_i : 속성유형 Av_j : 속성값

< DSA process >

```
if (E.ItemPermition.grantsAndDenials.grantRead = 1)
 then ReadResult.entry.incompliteEntry := T;
    if (ReadArgument.selection = ∅)
     then Return of DN procedure;*1
        return ReadResult;
     else for (At1 to Atm )
          if (Ai = E.Aj)
           then if (E.Aj.ItemPermition.grantsAndDenials.grantRead = 1)
             then if (ReadArgument.selection.infoTypes = 1)
               then for (Av1 to Avx)
                    if (E.Aj.Avk.ItemPermition.
                              grantsAndDenials.grantRead = 1)
                   then ReadResult.entry.information.attribute.type := ATj;
                       ReadResult.entry.information.attribute.values := Avk;
                   else if (E.Aj.Avk.ItemPermition.
```

```
                    grantsAndDenials.grantDisclosureOnError = 1)
                  then ReadResult.entry.incompleteEntry := T;
              else ReadResult.entry.information.attributeType := ATj;
          else ReadResult.entry.incompleteEntry := T;
      NoSuchAttributeOrValue-Read procedure;*2
  Non-disclosure procedure;*3
```

*1 Return of DN procedure
; 식별명칭 또는 alias 명칭을 출력한다.

*2 NoSuchAttributOrValue-Read Procedure
; 정상적인 결과 또는 오류를 출력한다.

*3 Non-disclosure procedure
; 오류를 지정한다.

```
if (E.ItemPermition.grantsAndDenials.grantDisclosureOnError = 1)
 then if (List/Search)
      then return ∅;
      else read.ERRORS := SecurityError;
 else read.ERRORS := NameError;
```

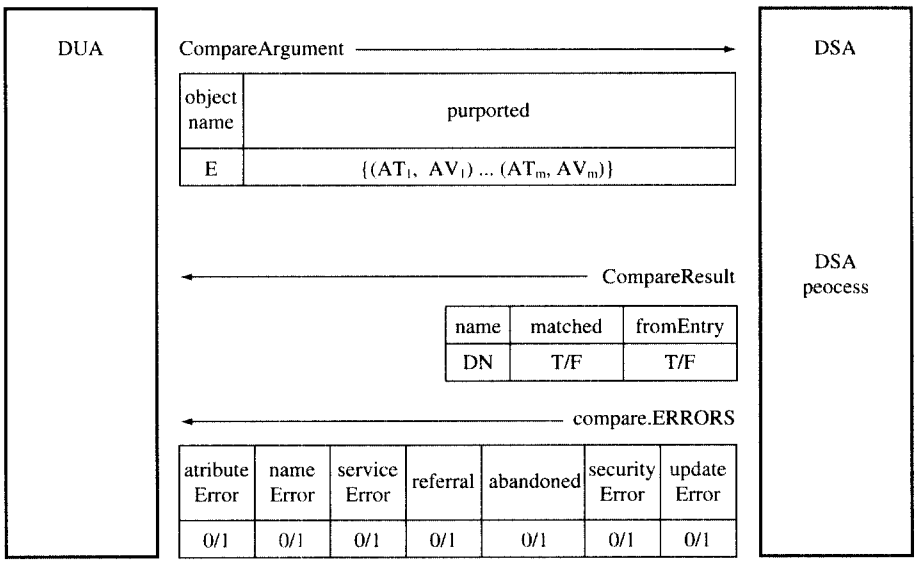
1.2 비교(Compare)

비교 서비스는 특정 엔트리의 특정 속성을 지정하여 해당 엔트리 속성의 전부 또는 일부와 비교 후 그 결과를 반환한다.

비교 서비스가 진행되는 절차는 다음과 같다.

< DSA process >

```
if (E.ItemPermition.grantsAndDenials.grantRead = 1)
 then if (Ati = E.Atj)
      then if (E.Atj.ATj.ItemPermition.grantsAndDenials.grantCompare = 1)
           then for (Avj1 to Avjs)
                if (E.Atj.Avjk.ItemPermition.grantsAndDenials.grantCompare = 1)
                 then if (AVi = Avjk)
                      then CompareResult.matched := T;
                CompareResult.matched := F;
                Return of DN procedure;*1
                return Compare;
           else if (E.Atj.Avjk.ItemPermition.
                          grantsAndDenials.grantDisclosureOnError = 1)
```



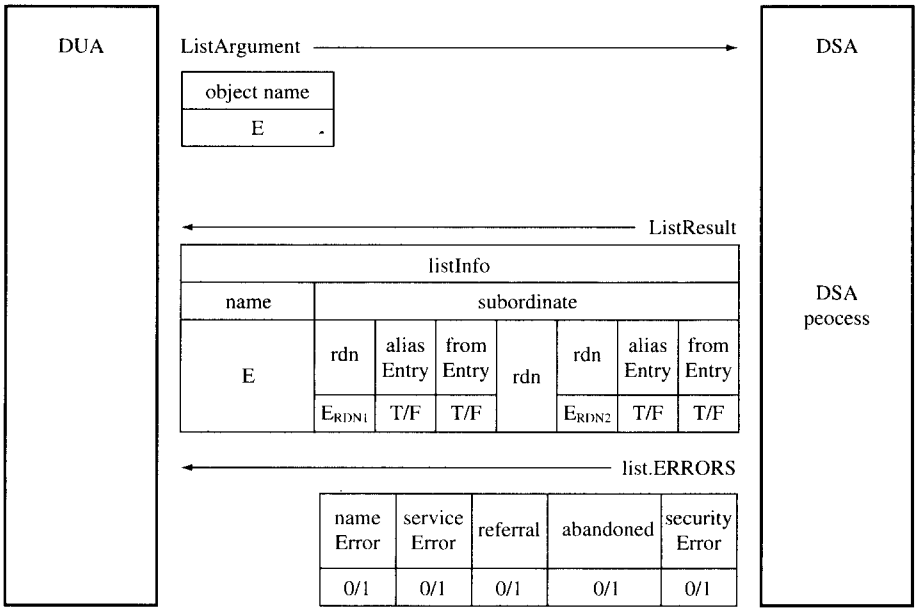
E : 식별명칭 AT_i : 속성유형 AV_j : 속성값

then compare.ERRORS := attributeError;
else compare. ERRORS := securityError;
else compare.ERRORS := attributeError;
else Non-disclosure procedure

1.3 목록(List)

목록 서비스는 특정 엔트리의 직속하위 엔트리들의 목록을 나타낸다. 이때 지정되는 특정 엔트리는 하나의 서브트리의 루트 엔트리이다. 즉, 서브트리의 base 엔트리이다.

< DSA process >
SEi : 직속하위 엔트리
for (SE1 to SEm)
if (E.ItemPermition.grantsAndDenials.grantBrowse = 1 and
E.ItemPermition.grantsAndDenials.grantReturnDN = 1)
then ListResult.listInfo.name := E;
ListResult.listInfo.subordinates.rdn := ERDN;
if (ListResult.listInfo.subordinates)
then return ListResult;
else Non-disclosure procedure;



E : 엔트리의 식별 명칭

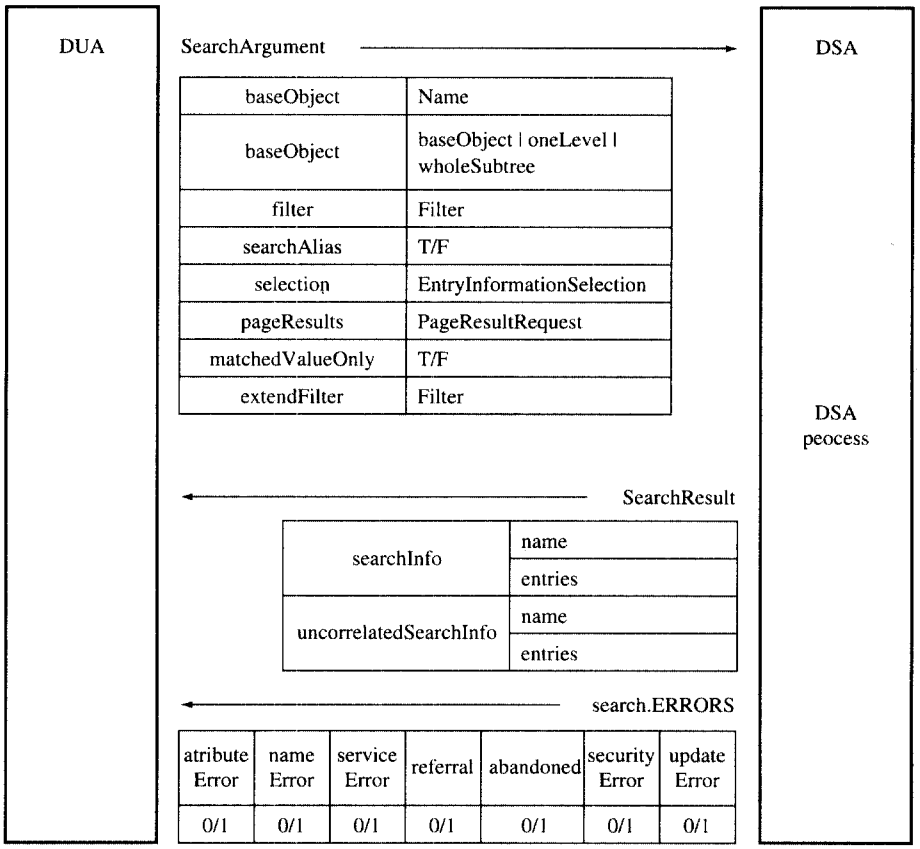
1.4 검색(Search)

검색 서비스는 디렉토리에서 지정한 엔트리의 속성유형 또는 속성값을 찾는다. 이때 검색하기 위한 디렉토리의 범위가 넓은 경우 필터를 이용하여 검색 범위를 줄인 후 검색한다.

< DSA process >

E_i : 하위 엔트리 FI_j : filter item

```
for (E1 to Em)
{
 if (Ei.searchAlias = T)
   then Alias Dereference on Search*1
   else if (Ei.ItemPermition.grantsAndDenials.grantBrowse = 1)
       then for (FI1 to FIn)
           {
            if (Ei.Atk.ItemPermition.grantsAndDenials.grantFilterMatch = 1 and
                Atk present in Ei)
             then if (present match)
  then for (AV1 to AVp)
                    {
```




```
            if(E_i.At_k.AV_r.ItemPermition.grantsAndDenials.
               grantFilterMatch = 1 and
                 AV_r present in E_i)
             then FI := T;
                 break;
           }
           if (not exist matched value)
            then FI := F;
       else FI := T;
   else if (present match)
       then FI := F;
       else FI := UNDEFINED;
evaluate filter;
if (filter = T)
 then Entry Selection in Search Procedure*2;
```

```
}
if (any Entries and Cross References exist)
 then finish;
 else Non-disclosure procedure;
```

*1 Alias Dereference on Search procedure
; 검색범위 안의 별명 엔트리들을 역참조하여 검색범위에 추가한다. 다른 DSA로의 연속참조가 필요한 경우는 이를 검색 결과에 추가한다.

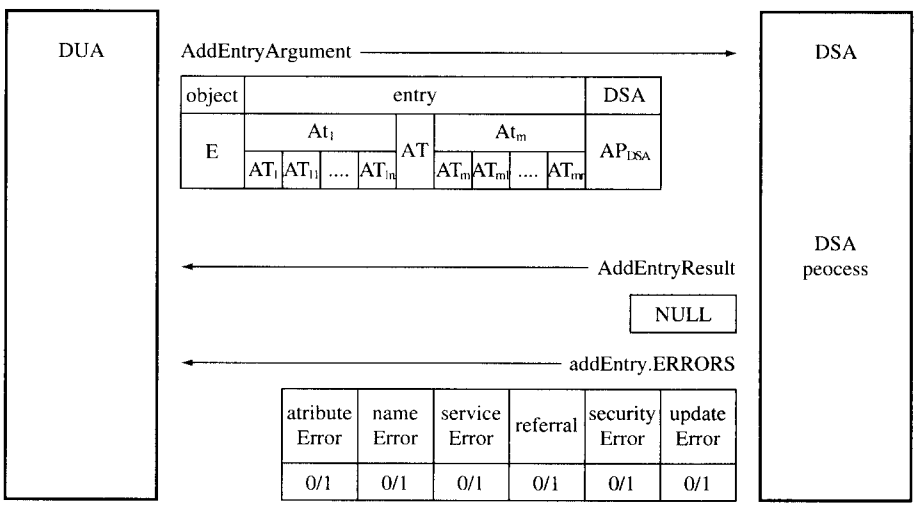
*2 Entry Selection in Search procedure
; 필터를 통과한 엔트리에 대해서 사용자가 요구한 selection에 있는 속성들을 검사하고 해당되는 속성 유형 및 속성값을 결과에 추가한다.

2. 디렉토리 수정 서비스(Directory Modification Service)

디렉토리 수정 서비스는 질의 서비스와는 달리 디렉토리 엔트리에 대한 정보를 직접적으로 수정하는 것이다. 디렉토리에 새로운 엔트리를 추가하거나 디렉토리로부터 엔트리 전체를 삭제하는 서비스와 지정 엔트리의 일부 속성유형 및 속성값을 추가/삭제하는 서비스가 있다.

2.1 엔트리 추가(Add entry)

DIT의 끝 노드에 엔트리를 추가한다. 엔트리를 추가하기 위해서는 최소한 하나의 속성



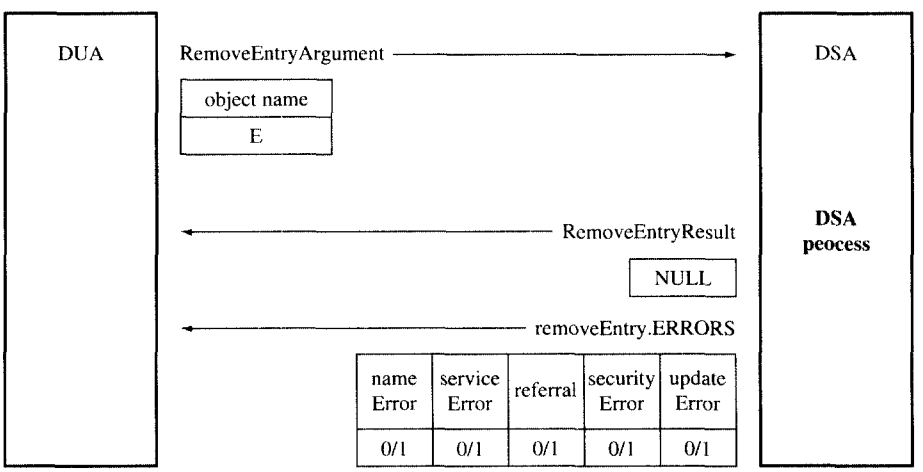
E : 추가할 엔트리 명칭 AT_1 : 엔트리 속성 AT_i : 엔트리 속성 유형
AV_{ij} : 속성값 AP_{DSA} : DSA의 접근점(Access Point)

을 가져야 하고, 각각의 속성값은 선택적이다.

< DSA process >

if (AP_{DSA}.E = E)
 then if (AP_{DSA}.E.ItemPermition.grantsAndDenials.grantDisclosureOnError = 1 or
 AP_{DSA}.E.ItemPermition.grantsAndDenials.grantAdd = 1)
 then add.ERRORS := UpdateError;
 else Non-disclosure procedure
 else if (E.ItemPermition.grantsAndDenials.grantAdd = 1)
 then for (At_1 to At_m)
 if (E.At_i.ItemPermition.grantsAndDenials.grantAdd = 1)
 then for (AV_{i1} to AV_{in})
 if (E.At_i.AV_{ij}.ItemPermition.grantsAndDenials.grantAdd = 0)
 then add.ERRORS := SecurityError;
 else add.ERRORS := SecurityError;
 add entry E to DSA;
 else add.ERRORS := Non-disclosure procedure;

2.2 엔트리 제거(Remove entry)



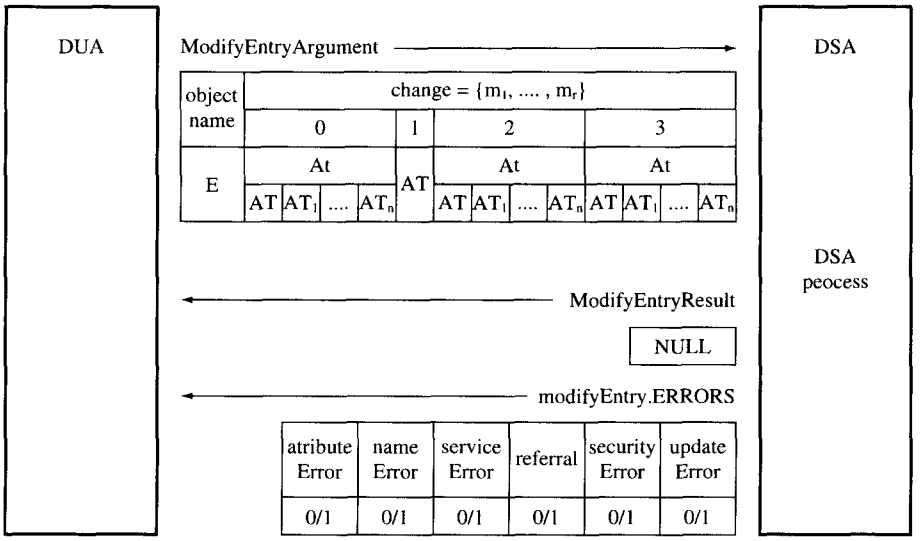
E : 엔트리의 식별명칭

지정된 명칭의 엔트리를 DIT로 부터 제거한다.

< DSA process >

if (E.ItemPermition.grantsAndDenials.grantRemove = 1)

then **remove E from DIT;**
else Non-disclosure procedure;



m_k : 수정 동작 종류
0 = 속성추가, 1 = 속성제거, 2= 속성값 추가, 3 = 속성값 제거

2.3 엔트리 수정(Modify entry)

지정한 엔트리의 속성/속성값을 제거하거나, 지정한 엔트리에 속성/속성값을 추가한다.

< DSA process >

```
if (E.ItemPermition.grantsAndDenials.grantModify = 1)
then for (m1 to mr)
     case of mk = 0
      if (exist At in E)
       then if (E.At.ItemPermition.grantsAndDenials.
                              grantDisclosureOnError = 1)
          then modifyEntry.ERRORS := AttributeError;
          else modifyEntry.ERRORS := SecurityError;
       else if (E.At.ItemPermition.grantsAndDenials.grantAdd = 1)
          then for (AV1 to AVn)
             if (exist AVi in At)
              then if (E.At.AVi.ItemPermition.grantsAndDenials.
                                    grantDisclosureOnError = 1)
                 then modifyEntry.ERRORS := AttributeError;
```

```
                    else if (E.At.ItemPermition.grantsAnd
                                Denials.grantDisclosureOnError = 1)
                        then modifyEntry.ERRORS := AttributeError;
                        else modifyEntry.ERRORS := SecurityError;
                else if (E.At.AV_i.ItemPermition.
                                grantsAndDenials.grantAdd = 0)
                    then if (E.At.AV_i.ItemPermition.grantsAndDenials.
                                grantDisclosureOnError = 1)
                        then modifyEntry.ERRORS := AttributeError;
                        else if (E.At.ItemPermition.grantsAnd
                                Denials.grantDisclosureOnError = 1)
                            then modifyEntry.ERRORS :=
                                        AttributeError;
                            else modifyEntry.ERRORS :=
                                        SecurityError;
case of m_k = 1
 if (exist At in E)
  then if (E.At.ItemPermition.grantsAndDenials.grantRemove = 0)
      then if (E.At.ItemPermition.grantsAndDenials.
                                grantDisclosureOnError = 1)
          then modifyEntry.ERRORS := SecurityError;
          else modifyEntry.ERRORS := AttributeError;
  else modifyEntry.ERRORS := AttributeError;
case of m_k = 2
 if (exist At in E)
  then for (AV_1 to AV_n)
      if (exist AV_i in A_t)
       then if (E.At.AV_i.ItemPermition.grantsAndDenials.
                                grantDisclosureOnError = 1)
           then modifyEntry.ERRORS := AttributeError;
           else if (E.At.ItemPermition.grantsAnd
                                Denials.grantDisclosureOnError = 1)
               then modifyEntry.ERRORS := AttributeError;
               else modifyEntry.ERRORS := SecurityError;
       else if (E.At.AVi.ItemPermition.grantsAndDenials.grantAdd = 0)
           then if (E.At.AVi.ItemPermition.grantsAndDenials.
                                grantDisclosureOnError = 1)
               then modifyEntry.ERRORS := AttributeError;
```

```
                    else if (E.At.ItemPermition.grantsAnd
                                Denials.grantDisclosureOnError = 1)
                        then modifyEntry.ERRORS := AttributeError;
                        else modifyEntry.ERRORS := SecurityError;
        else modifyEntry.ERRORS := AttributeError;
    case of m_k = 3
     if (exist At in E)
      then for (AV_1 to AV_n)
          if (exist AV_i in At)
           then if (E.At.AV_i.ItemPermition.grantsAndDenials.grantRemove = 0)
                then if (E.At.AV_i.ItemPermition.grantsAnd
                                Denials.grantDisclosureOnError = 1)
                    then modifyEntry.ERRORS := SecurityError;
                    else modifyEntry.ERRORS := AttributeError;
           else modifyEntry.ERRORS := AttributeError;
        else modifyEntry.ERRORS := AttributeError;
  modify E;
else Non-disclosure procedure;
```

V. 결 론

네트워크가 발달하면서 분산된 정보를 다루어야만 하는 경우가 빈번해지는데 이에 대한 해결책으로 대두되는 것이 디렉토리이다. 실세계에서의 정보를 디렉토리에 담아 관리함으로써 사용자들은 더욱 편리하게 분산된 정보를 사용할 수 있게 되었다. 더욱이 날이 갈수록 공개키 기반구조에 대한 필요성이 증대되고 공개키 관리가 중요 문제로 다루어진다. 공개키 관리에 대한 방법 중에 하나로 디렉토리가 제안되었고, 이에 대한 연구가 활발히 이루어지고 있다. 또한 디렉토리는 전세계적으로 연구가 활발히 진행되고 있으며 국제 표준 또한 지속적으로 발표/개선되고 있다. 이에 국내에서도 디렉토리 시스템 전반에 관한 연구가 이루어져야 할 것이며 아울러 디렉토리 구축과 운영에 관한 연구도 이루어져야 할 것이다.

참 고 문 헌

[1] ITU-T Recommandation X.500 (1993) | ISO/IEC 9594-1:1993, Information technology - Open Systems Interconnection - The Directory:Overview of Concepts, Models and Services, 1993.

[2] ITU-T Recommandation X.501 (1993) | ISO/IEC 9594-3:1993, Information technology - Open Systems Interconnection - The Directory:Abstract Service Definition, 1993.

[3] ITU-T Recommandation X.511 (1993) | ISO/IEC 9594-3:1993, Information

technology - Open Systems Interconnection - The Directory:Abstract Service Definition

[4] ITU-T Recommandation X.518 (1993) | ISO/IEC 9594-4:1993, Information technology - Open Systems Interconnection - The Directory:Procedures for Distributed Operation, 1993.

[5] ITU-T Recommandation X.519 (1993) | ISO/IEC 9594-5:1993, Information technology - Open Systems Interconnection - The Directory:Protocol Specifications, 1993.

[6] ITU-T Recommandation X.520 (1993) | ISO/IEC 9594-6:1993, Information technology - Open Systems Interconnection - The Directory:Selected Attribute Types, 1993.

[7] ITU-T Recommandation X.521 (1993) | ISO/IEC 9594-7:1993, Information technology - Open Systems Interconnection - The Directory:Selected Object Classes, 1993.

[8] ITU-T Recommandation X.509 (1993) | ISO/IEC 9594-8:1993, Information technology - Open Systems Interconnection - The Directory:Authectication Framework, 1993.

[9] ITU-T Recommandation X.525 (1994) | ISO/IEC 9594-9:1993, Information technology - Open Systems Interconnection - The Directory:Replication, 1993.

[10] 이재광, 이용준, "X.500 디렉토리 정보보호", 통신정보보호학회지, 4권 3호, pp. 22-33, 1994. 9.

[11] 최성민, 이인숙, 장청룡, 원동호, "분산 디렉토리 시스템에서 인증에 관한 연구", 한국통신정보보호학회 학술 발표회 논문집, pp. 41-54, 1992.

□ 著者紹介

정 권 성

1971년 11월 16일생
1996년 8월 성균관대학교 수학과 졸업(이학사)
1997년 3월 ~ 현재 성균관대학교 전기전자컴퓨터공학부 석사과정

김 영 수

1972년 4월 20일생
1998년 2월 성균관대학교 정보공학과 졸업(공학사)
1998년 3월 ~ 현재 성균관대학교 전기전자컴퓨터공학부 석사과정

이 형 규

1969년 1월 18일생
1996년 2월 성균관대학교 산업공학과 졸업(공학사)
1998년 3월 ~ 현재 성균관대학교 전기전자컴퓨터공학부 석사과정

조 동 욱

1972년 3월 9일생
1998년 2월 성균관대학교 정보공학과 졸업(공학사)
1998년 3월 ~ 현재 성균관대학교 전기전자컴퓨터공학부 석사과정

원 동 호

1976년 2월 성균관대학교 전자공학과 졸업(공학사)
1978년 2월 성균관대학교 대학원 졸업(공학석사)
1988년 2월 성균관대학교 대학원 전자공학과(공학박사)
1978년 4월 ~ 1980년 3월 한국전자통신연구소 연구원
1985년 9월 ~ 1986년 8월 일본 동경 공대 객원 연구원
1982년 3월 ~ 현재 성균관대학교 공과대학 정보공학과 교수
1991년 ~ 현재 한국통신정보보호학회 편집이사
1996년 4월 ~ 현재 정보화추진위원회 자문위원

※ 주관심 분야 : 암호이론, 정보이론