情報保護學會論文誌
第12券 第6號, 2002. 12

역할기반 접근제어 및 비밀통신을 지원하는 SPKI/SDSI HTTP 보안 서버

이 영 록*, 김 민 수*, 김 용 민*, 노 봉 남*, 이 형 효**

SPKI/SDSI HTTP Secure Server to support Role-based Access Control & Confidential Communication

Yong-lok Lee*, Min-soo Kim*, Yong-min Kim*, Bong-nam Noh*, Hyeong-hyo Lee**

요 약

인증을 지원하고 비밀통신 설정을 위한 안전한 수단을 제공하기 위해 보편적으로 이용하고 있는 것이 X.509 v3 인증서를 이용하는 SSL/TLS 프로토콜이다. X.509 PKI는 아주 복잡하고도 불완전하기 때문에 이를 보완하기 위한 연구의 결과로 SPKI/SDSI가 탄생하였다. 본 논문은 기존의 Geronimo 프로젝트로 구현된 SPKI/SDSI 서버가 지니지 못한 비밀통신과 역할기반 접근제어 기능을 수행하는 보안모듈들을 추가하고, 온라인으로 이름&권한-인증서를 전담해 발행하는 이름&권한 위임 모듈을 탑재한 보안서버를 설계하며 그의 프로토타입을 구현함이 목적이다. 그리고 이 프로토타입의 구현이 웹상에서 어떻게 적용되는지를 실제 예를 통해 보였다.

ABSTRACT

We generally use SSL/TLS protocol utilizing X.509 v3 certificates so as to provide a secure means in establishment an confidential communication and the support of the authentication service. SPKI/SDSI was motivated by the perception that X.509 is too complex and incomplete. This thesis focuses on designing a secure server and an implementation of the prototype which has two main modules, one is to support secure communication and RBAC, not being remained in the SPKI/SDSI server which was developed by the existing Geronimo project and the other is to wholly issue name-certificate and authorization-certificate. And the demonstration embodied for our server is outlined hereafter.

Keyword : *SPKI(Simple Public Key Infrastructure), SDSI(Simple Distributed Security Infrastructure), SPKI/SDSI 인증서(Certificate)*

Ⅰ. 서 론

서버가 관리하는 비밀 정보들은 정당한 권한의 소유자에게만 전달되어야 한다. 이를 위해 서버는 먼저 클라이언트의 신원을 확인한 후 정당한 사용자이면 정보를 암호화해 보내는 비밀통신을 이용한다. 비밀통신에는 시간과 경제성면에서 대칭키 암호시스템을 이용한다. 이를 위한 세션키 생성에 가장 보편적으로 이용되고 있는 것은 X.509 공개키 기반구조(Public Key Infrastructure : PKI)이다. 하지만 X.509 기반 PKI는 루트에서부터 시작하는 계층적 구조의 전역이름을 이용하고 있어 사용자가 소속되어 있는 회사나 부서의 이동에 따른 전역이름 변경이 자주 일어날 수 있을 뿐만 아니라 루트가 어느국

* 전남대학교 컴퓨터정보학부(yrlee@chonnam.ac.kr)
** 원광대학교 정보 · 전자상거래학부

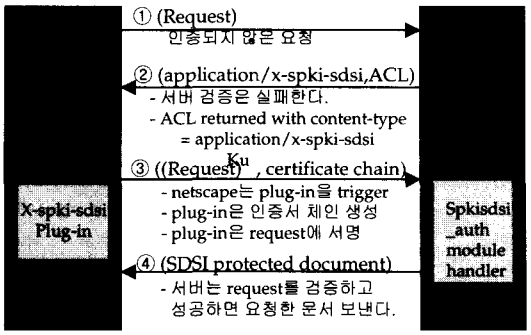
가에 소속되어야하는가 등에 따른 국제적인 이해관계에 얽힐 수 있는 복잡성을 내재하고 있으며, 또한 권한이나 이름을 하나의 인증서에 나타내야 하므로 둘 중 하나라도 변경사항이 생기면 새로 갱신해야하는 불완전성을 지니고 있다.[2,7] 이의 단점을 보완하고자 PKIX 작업그룹은 권한관리 기반구조(PMI: Privilege Management Infrastructure)를 두어 속성-인증서(AC: Attribute Certificate)를 정의해 사용자에 권한을 부여하고 있지만 여전히 전역적인 이름을 사용하고 있고, 구성요소 또한 복잡하다.

이러한 PKI 구조를 보다 단순화시키고 융통성 있게 보완하기 위한 연구의 결과가 1998년에 탄생한 SPKI/SDSI 기반구조이다.[1] SPKI/SDSI는 두 개의 인증서인 이름-인증서와 권한-인증서를 명세하도록 허용하고, 전역이름 대신 계층적인 지역 이름 공간(local name space hierarchy)를 이용하는 등 아주 단순하고 융통성있는 정책모델을 제공한다. [표 1]은 현재 광범위하게 적용되고 있는 X.509 PKI와 SPKI /SDSI를 비교해놓은 것이다.

MIT의 교수 Rivest와 그의 제자들은 SPKI/SDSI에 대한 명세를 구체화하기 위해 그 명세를 바탕으로 SPKI/SDSI 인증서를 실제 정의하고 이를 컴퓨터에 구현하기 위한 노력을 기울여왔다[2,3,4,5,6,7]. 2000년 말에 Andrew와 Dwaine이 SPKI/SDSI 인증서들을 이용하여 안전한 웹 클라이언트와 웹서버간 접

[표 1] X.509 PKI와 SPKI/SDSI의 비교

	이름공간	전역이름
X.509 PKI	인증서 유형	이름-인증서
	이름-키 연결	단일 값 함수 (한개의 이름에 한개의 키 연결)
	인증기관 특징	전역적인 계층구조
	신뢰모델	계층적인 신뢰모델
	서명	발행자의 개인키로 서명
	인증서 폐기	인증서 폐기리스트 이용
SPKI/ SDSI	이름공간	지역이름
	인증서 유형	이름-인증서, 권한-인증서
	이름-키 연결	다중 값 함수(한개의 이름에 한개 이상의 키 연결)
	인증기관 특징	평등적인 구조
	신뢰모델	검증자로부터 의뢰자에 이르는 권한-연결 검사
	서명	발행자의 개인키로 서명
	인증서 폐기	짧은 생명주기의 인증서 옹호, 온라인 체크



[그림 1] "Geronimo 접근제어 프로토콜

근제어에 사용한 접근제어 프로토콜인 "Geronimo Project"를 간략하게 기술하면 [그림 1]과 같다.

하지만 Geronimo Project는 권한-인증서 발행을 위해 클라이언트는 서버에게 자신의 공개키를 디스켓에 담아서 서버관리자에게 직접 전달해주거나, 아니면 SSL/TLS와 같은 안전한 채널을 통해 주고받는다는 가정하에서 이루어져왔다. 또한 클라이언트가 서버의 검증과정을 통과하여 비밀정보를 접근할 수 있다하더라도 웹 상에서 보통 HTML 문서와 같이 평문 형태로 전달한다. 이는 해커에 의해 얼마든지 노출될 수 있다. 비밀통신을 위해서는 기존의 SSL/TLS를 이용할 수는 있는데, SSL/TLS는 X.509 PKI 기반 인증서를 이용하므로 SPKI/SDSI 기본 정신에 어긋난다.

따라서 본 논문은 "Project Geronimo"에 대한 분석으로 드러난 문제점을 보완하기 위해 보안기능을 추가한 보안서버를 설계하고 그 프로토타입을 구현함을 목적으로 한다. 이를 위해 본 논문은 다음과 같은 일을 한다.

(1) SPKI/SDSI 인증서의 일종인 자기소개 인증서와 역할 인증서, 사망신고서 등을 S-expression으로 정의한다.
(2) 순수한 SPKI/SDSI 인증서만을 이용하여 안전한 비밀통신을 지원하는 임의적 접근제어 프로토콜을 제안한다.
(3) SPKI/SDSI 권한-인증서의 단점을 보완하기 위해 XML로 정의된 제약사항을 정의하여 임의적 접근제어에 반영한다.
(4) 임의적 접근제어 반영시 권한-인증서 발행을 위한 웹서버와 클라이언트의 상호인증을 온라인상에서 가능하도록 계층적인 시스템을 구성한다.
(5) 역할기반 접근제어가 이루어지도록 역할서버와

웹서버, 그리고 클라이언트로 구성된 시스템을 설계한다.

(6) SPKI/SDSI 인증서들을 이용하여 안전한 비밀통신을 지원하는 역할기반 접근제어 프로토콜을 제안한다.

(7) 제안한 SPKI/SDSI 보안서버를 웹상에서 적용한 예를 보인다.

본 논문의 구성은 다음과 같다. SPKI/SDSI 기반구조와 역할기반 접근제어모델에 대해 2장에서 설명하고, 3장에서는 역할기반 접근제어 및 비밀통신을 지원하는 SPKI/SDSI 보안서버에 이용하기 위해 제안한 보안프로토콜에 대해 기술한다. 4장은 이 보안서버 프로토타입을 어떻게 웹에 적용할 수 있는지 응용 예에 대해 기술하고 5장에서 본 서버와 기존서버를 비교분석한다. 그리고 마지막으로 결론과 향후계획에 대해 기술한다.

II. SPKI/SDSI와 역할기반 접근제어 모델

이번 장은 본 논문에서 제안하는 보안서버에 추가되는 보안모듈의 기초가 되는 SPKI/SDSI와 역할기반 접근제어 모델에 대해 설명한다.

2.1 SPKI/SDSI 기반구조

SPKI/SDSI(Simple Public Key Inftastructure/Simple Distrubuted Security Infrastructure)는 기존의 PKI 기반의 융통성 없고 복잡한 문제를 단순화하기 위해 두 부류에 의해 연구되었다. 한 부류는 1996년 MIT 교수인 Lampson과 Rivest가 제안한 SDSI이고, 같은 해 Ellison과 Frantz, Thomas 등이 연구하기 시작한 SPKI가 나머지 부류이다.

SDSI의 특징 중 두드러진 것은 "지역이름공간"라는 개념을 도입해서 공개키의 소유자는 각각 그 공개키에 기반을 둔 지역이름공간을 생성할 수 있다는 것이다. 즉, 누구든지 자신이 보유한 대상주체들의 공개키들을 자신의 공개키에 기반을 둔 지역이름으로 바인딩하는 이름-인증서를 발행할 수 있다는 것이다. 또한 공개키 대신 이들 지역이름들을 이용하여 새로운 이름-인증서를 발행할 수도 있다.

한편 SPKI의 특징은 이름-인증서와 권한-인증서를 각각 구별하여 명세할 수 있는 융통성을 제공하였다. 또한 SPKI는 주체(principal)의 구별을 이름이 아닌 공개키로 함으로써 수시로 바뀔 수 있는 전역이름을 주체의 구별 대상으로 하는 X.509 v3 인증서를 이용하는데 따른 문제점을 해결할 수 있었다.

이렇게 각각 연구되기 시작한 SPKI와 SDSI는 1998년에 통합되어 SPKI/SDSI라는 명칭으로 명명되었다[8]. 이 절에서는 SPKI/SDSI의 가장 핵심인 이름-인증서, 권한-인증서, S-expression에 대해 간단히 기술하고[9,10], 클라이언트가 서버에 제공하는 권한-인증서들을 찾는 "Certificate chain discovery" 알고리즘에 대해 설명한다.

2.1.1 이름-인증서

이름-인증서(name-certificate)는 아래와 같이 4-튜플로 구성되어 있다.

〈발행자, 지역이름, 대상주체, 유효기간〉

발행자(issuer)는 인증서를 발행하는 기관이나 사람(principal)이고, 지역이름(local-name)은 대상주체(subject)에 붙이려는 이름, 그리고 대상주체(subject)는 이름에 바인딩되는 대상임과 동시에 그 인증서를 받는 대상이고, 유효기간 명세(validity)는 이 인증서가 유효한 기간이다. 〔그림 2〕는 이름-인증서를 나타낸 것이다.

클라이언트의 공개키를 K_{young}이라고 하고 서버관리자의 공개키를 K_{chul}이라고 하면, 서버는 클라이언트의 공개키를 디스켓으로 받아와 "young"이라는 이름으로 정의한 이름-인증서를 발행할 수 있다. 위의 대상주체(subject) 항목에는 "K_{young} girlfriend brother"와 같이 좀더 복잡한 이름이 올 수 있는데 이는 young의 여자친구의 오빠를 지칭한다.

< K_{chul} , "young", K_{young}, 01/30/02 - 10/21/02 >

발행자	지역이름
객체	
유효성 명세	

K_{chul}	"young"
K_{young}	
01/30/02 - 10/21/02	

(발행자의 비밀키로 서명) (K_{chul}의 비밀키로 서명)

〔그림 2〕 이름-인증서

2.1.2 권한-인증서

권한-인증서(authorization certificate)는 아래와 같이 5-튜플로 구성되어 있다.

〈발행자, 주체, 위임-비트, 권한 태그, 유효기간〉

앞 튜플의 발행자와 대상주체, 그리고 유효기간 명세는 2.1.1절과 같다. 태그는 발행자가 대상주체에게 허가하는 권한을 명세하는 것이며, 위임-비트는 참 또는 거짓 값이 채워질 수 있는데 참이면 대상주체 자신이 받은 권한의 전부 또는 일부를 또 다른 대상주체에게 위임할 수 있음을 말한다.

권한-인증서는 [그림 3]과 같이 명세하면 된다. 서버 관리자 "K_{chul}"은 K_{young}에게 자신이 보호하고 있는 user/project 디렉토리에 있는 파일들을 읽고 수행할 수 있는 권한을 주는 예이다.

< K_{chul} , K_{young} , true, tag, (10/11/02 - 05/30/02) >

Kchul	Kyoung	true
read and execute permissions for files under "user/project"		
10/11/01 - 08/30/02		

(K_{chul} 와 대응되는 비밀키로 서명)

[그림 3] 권한-인증서

2.1.3 S-expression

이름-인증서나 권한-인증서 등은 S-expression이라 불리는 문법구조로 기술될 수 있다.[11] 서버인 K_{chul}이 클라이언트 공개키 K_{young}을 이름 "YoungLok"으로 바인딩하는 이름-인증서와 그 서명을 S-expression으로 나타내면 [그림 4]와 같다.

```
(cert
  (issuer
    (name
      (public-key
        (rsa-pkcs1-md5
          (e #45#)
          (n
            |AKUNgKMptK17p..... | ) ) )
      YoungLok ))
  (subject
      (public-key
        (rsa-pkcs1-md5
          (e #32#)
          (n
            |Uala&*AKUNgKMptKcvc....|))))
(Signature
    (hash md5 | JJR1tuwoieu&8 ... |)
      (public-key
        (rsa-pkcs1-md5
          (e #78#)
          (n
            |PkY45KTa&*AKU6OptKpt..|)))
  (rsa-pkcs1-md5
            |sdfdewrGJj..... | ))
```

[그림 4] 이름-인증서와 서명

2.1.4 Certificate Chain Discovery 알고리즘

"Certificate Chain Discovery" 알고리즘은 클라이언트가 수행하는 것으로 서버의 객체에 대해 정해진 연산을 수행할 수 있는 권한이 있음을 증명해 보이기 위해 제공하는 권한-인증서와 이름-인증서들을 인증서 캐쉬에서 찾아내는 알고리즘이다.[5]

이 알고리즘은 서버의 ACL내의 발행자인 "SELF"로부터 클라이언트의 공개키에 이르는 인증서의 권한 연결이 존재하는 것들을 찾는 것이 목적인데, 몇 가지 정의와 정리를 바탕으로 이루어진다. 이 알고리즘은 다음과 같이 동작한다.

① 쓸모없는 권한-인증서들을 인증서 캐쉬에서 제거한 후 남아있는 이름-인증서와 권한-인증서를 "rewrite rule" 형태로 변환시킨다. 〈K_A, "Bob", K_B, 07/31/02-07/29/03〉인 이름-인증서의 "rewrite rule" 형태는 K_A Bob → K_b이다.

② "name reduction closure"를 계산한다. "name reduction closure"는 남아있는 ①의 결과들이 서로 "→"의 좌측과 우측에 대체되어 변할 수 있는 생성물을 말한다.

③ ②의 결과물 중에서 인증서 발행자가 "SELF"인 것으로부터 유도될 수 있는 것을 뽑아낸다.

④ ③의 결과물로부터 그래프를 그린 후, "SELF"로부터 클라이언트의 공개키에 이르는 패스가 있는지를 결정하기 위해 "depth first search"방법을 이용한다. 성공하면 SELF□ to K2□ to KA ■를 리턴한다. 여기서 □과 ■은 티켓으로서 권한-인증서 위임과 관련있는 것으로 □의 의미는 다른 대상주체(subject)에게 권한을 위임할 수 있다는 것이고, ■은 다른 대상주체에게 더 이상 위임을 허용하지 않는다는 것을 의미이다.

위 알고리즘의 입력은 서버로부터 전달 받은 ACL 리스트와 사용자가 인증서캐시에 보관하고 있는 이름-인증서와 권한-인증서들로 다음과 같다.

- 서버로부터 전달받은 ACL 리스트들
  〈SELF, K_0 engineering, T_1, 1, (07/28/02, 07/30/02)〉
  〈SELF, K_0 finance, T_1, 1, (07/28/02, 07/30/02)〉
  〈SELF,K_0human_resources,T_1,0,(10/09/02, 10/11/02)〉
- 인증서 캐시에 있는 "rewrite rule" 형태로 표현된 이름-인증서와 권한-인증서들
  K_0 finance ■ K_1 accounting

K_1 accounting ■ K_1 Bob
K_1 Bob ■ K_2
K_3 Alice ■ K_A
K_5 Alice_Brown ■ K_A
〈K_3, "K_3 Alice", T_2, 0, (07/28/02, 07/30/02)〉

위의 입력을 예로하여 이 알고리즘을 적용하여 나온 출력은 아래와 같으며 클라이언트는 자신의 증명을 위해 이 묶음을 순서열로하여 서버에게 보낸다

(〈K_0, "finance", K_1 accounting, (07/28/02, 07/30/02)〉,
〈K_1, "accounting", K_1 Bob, (07/28/02, 07/30/02)〉,
〈K_1, "Bob", K_2, (07/28/02, 07/30/02)〉,
〈K_2, "K_3 Alice", T_1, 0, (07/28/02, 07/30/02)〉,
〈K_3, "Alice", K_A, (07/28/02, 07/30/02)〉)

2.2 역할기반 접근제어 모델

RBAC 모델은 특정정보에 대한 연산을 수행할 수 있는 권한(permission)을 사용자에게 직접 할당하지 않고, 기업이나 조직에서 정의된 역할에 배정한다. 그러므로 사용자는 우선 해당정보에 대한 연산을 수행할 수 있는 권한을 지닌 역할의 구성원이 되어야만 한다. RBAC 모델은 또한 최소권한 원칙(least privilege principle), 임무분리(SOD : separation of duty), 자료 추상화(data abstraction)와 같은 주요 보안 원칙들 역시 지원하고 있다[13].

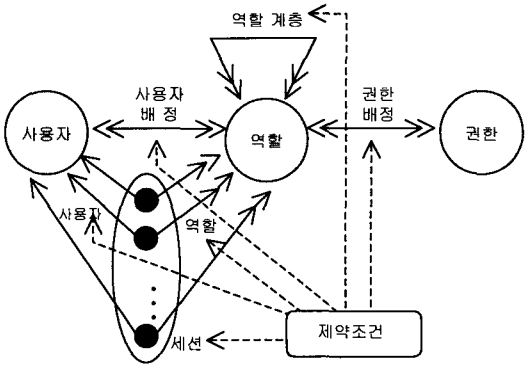
〔그림 5〕는 RBAC 모델의 주요 구성요소와 구성요소간의 관계를 나타내고 있다[13].

가) 역할

역할(role)은 조직이나 기업에서 정의된 업무의 기능을 의미하는 것으로 이 역할에 배정된 사용자에게는 그 업무의 기능에 따른 권한과 책임이 주어진다.

나) 역할계층

역할계층은 역할에 배정된 권한들 사이에 포함관계가 있는 역할들간의 부분순서(patial order) 관계로서 기업의 권한과 책임 체계와 매우 유사하여 기업의 권한체계를 모델링하는데 매우 유용하다. 역할계층의 관계는 반사적(reflexive), 비대칭적(asym-metric), 이행적(transitive) 관계이며 도식화된 역할 계층에서는 사이클이 존재하지 않는다.



〔그림 5〕 RBAC 모델의 구성요소와 관계

다) 사용자

사용자(user)는 기계(machines)나 네트웍, 또는 지능을 가진 자율적인 대리인(intelligent autonomous agent)이나 사람(human) 등이 될 수 있으며, 컴퓨터 시스템내의 정보를 사용하는 주체이다.

라) 권한

권한(permission)은 하나 또는 그 이상의 보호된 정보객체(objects)에 연산(operation)을 수행할 수 있도록 하는 승인을 의미하는 것으로 객체와 그 객체에 수행할 수 있는 연산의 집합으로 나타낸다. 연산(operations:OPS)은 RBAC 시스템이 보호하는 정보객체(objects:OBS)에 대해 수행 가능한 접근모드로 파일시스템에서의 연산은 읽기(read), 쓰기(write), 수행(execute)으로 정의될 수 있다. 객체(objects: OBS)는 RBAC을 구현하려는 시스템에 따라 여러 가지 측면으로 간주될 수 있다. 운영체제 시스템은 객체를 파일이나 디렉토리로 정의한다.

마) 세션

세션(session)은 한 사용자와 여러 개의 역할들로 구성된 집합으로 사용자는 세션을 통해 자신에게 배정된 역할들 중 일부 또는 전부를 수행할 수 있다.

바) 사용자 배정

사용자 배정(user assignment)는 사용자가 수행할 수 있는 역할들을 지정하는 것이다. 사용자와 역할간에는 다대다 관계가 성립한다.

사) 권한 배정

권한배정(permission assignment)은 역할이

수행할 수 있는 권한을 지정한 것으로 역시 서로간에는 다대다의 관계이다.

아) 제약조건

제약조건(constraints)은 위에서 정의된 모든 구성요소들에 대해 적용될 수 있으며, 각 구성요소가 지니는 특성에 제약을 가할 있다. 제약조건의 예로는 한 역할에 할당될 수 있는 최대 사용자의수나 시간제약사항 등이 있다.

Ⅲ. RBAC 및 비밀통신을 지원하는 SPKI/SDSI HTTP 보안 프로토콜

SPKI/SDSI 인증서를 이용해 웹에서 서비스를 제공하기 위한 시스템을 구현하려는 시도는 계속되어왔으나 아직은 연구의 초기단계로 많은 문제점을 안고 있으며 많은 가정들을 전제로 구현되고 있다. 이런 연구들 가운데 SPKI/SDSI 인증서를 이용한 접근제어를 웹에서 수행하기 위한 노력으로 구현된 "Geronimo 프로젝트"도 기본적인 방향만 제시할 뿐 적지않은 문제점을 안고 있다.

본 논문은 Geronimo 프로젝트가 지니지 못한 비밀통신과 온라인으로 키를 등록하는 문제를 해결하기 위해 자기소개 인증서와 키-사망신고서를 정의하여 이용하고 Diffie/Hellman 인수를 이용함으로써 비밀통신 서비스를 제공할 수 있도록 하였다. 또한 연속적인 권한위임에 제약을 가하기 위해 XML로 정의한 제약사항을 접근제어에 이용하도록 하였으며, 임의적 접근제어 외에도 RABC을 수행하도록 하기 위해 역할서버로부터 역할인증서를 발행받도록 함으로써 기업환경에 적합한 웹 보안서버로서의 기능을 수행하도록 하였다.

3.1 SPKI/SDSI 인증서의 정의

본 논문에서 제안한 보안 프로토콜에 이용될 용어들은 다음과 같이 정의된다.

[정의1] 자기소개 인증서

어느 누구나 각각 자신을 소개하는 인증서를 발행할 수 있다. "자기소개 인증서"는 아래와 같이 4-튜플로 구성된다.

〈자신의 공개키, 지역이름, 자신의 공개키, 유효기간〉

온라인으로 유효기간을 체크하기 위해서는 유효기간 항목을 아래와 같이 대체하면 된다.[1]

("online" "one-time" "Key-Health Center URI 주소")

[정의2] 키-사망신고서

공개키 소유자의 개인키가 노출 또는 위·변조되었을 때 그 공개키를 키-건강보증센터로부터 제거하기 위해 공개키 소유자(클라이언트 또는 서버)가 보내는 것으로 아래와 같은 튜플로 구성된다.

〈발행자, 지역이름, 대상주체, 사망일자〉

키-사망신고서는 서명과 함께 보내는 순서열로 이를 S-expression으로 표기하면 [그림 6]과 같다.

```
(death-certificate
  ( issuer
    ( name
      ( public-key
          ( 발행자의 공개키의 공개키 )
        객체에 바인딩하는 로컬네임 ) )
  ( subject ( public-key
          ( 발행받는자의 공개키 ) ) )
  ( death-date ( 년도-월-일_시:분:초 ) )
)
( signature
  ( hash md5
          |9/qwereJk094tBaYakit$Q==|)
  ( public-key
          ( 발행자의 공개키 )
  ( rsa-pkcs1-md5
    | 키-사망신고서의 메시지 다이제스트에
          발행자의 개인키로 서명한 값 | )
)
```

[그림 6] 키-사망신고서의 S-expression

[정의3] 권한-인증서 서버는 자신의 정보에 접근할 수 있는 권한을 나타내는 권한-인증서를 다른 클라이언트에게 발행할 수 있다. 권한-인증서는 아래와 같이 5-튜플로 구성되어 있다.

〈서버 공개키, 클라이언트 공개키 또는 지역이름, "true", 권한을 명세한 태그, 유효기간〉

[정의4] 역할 인증서

이름-인증서의 한 종류로 역할을 정의할 수 있다. 이름-인증서의 지역이름 항목에 역할이름을 부여하고, 대상주체항목에 그 역할에 해당하는 사용자의 공개

키나 지역이름을 지정한다. 아래와 같이 4-튜플로 구성된다.

〈역할서버의 공개키, 역할이름, 대상주체의 공개키나 지역이름, 유효기간〉

[정의5] 클라이언트 도메인 관리자

클라이언트들이 소속되어 있는 도메인을 관리하는 관리자로서 클라이언트들에게 이름-인증서나 권한-인증서를 발행할 수 있다.

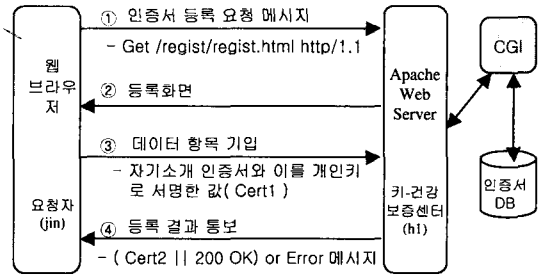
[정의6] 키-건강보증센터

클라이언트나 서버가 상대방으로부터 받은 이름-인증서나 역할-인증서의 유효성을 온라인으로 검증을 요청할 경우 이에 대한 검증을 수행하는 기관으로 검증결과를 의뢰자에 통보한다.

3.2 자기소개 인증서 생성 및 등록

통신하는 당사자의 상호인증과 메시지의 안전한 전송을 보장하는 수단으로 클라이언트와 서버는 각각 아래와 같이 공개키-개인키 쌍을 생성하여 자기소개 인증서를 만든 다음 자신들이 지정하거나 상호 합의하에 미리 지정된 키-건강보증센터에 등록한다. 이 자기소개 인증서는 이후에 제안되는 보안프로토콜에서 이용된다. 자기소개 인증서 생성 및 등록과정은 아래와 같다.

(1) 클라이언트와 서버는 각각 공개키-개인키 쌍을 생성하고 개인키는 자신만의 패스워드를 이용하여 암호화한 후 자신의 비밀장소에 저장한다.
(2) 클라이언트와 서버는 각각 자신의 공개키가 더 이상 유효하지 않음을 나타내는 키-사망신고서를 작성하고 (1)에서 생성한 자신의 개인키로 서명한 후 자신의 비밀장소에 저장한다. 이 키-사망신고서는 차후 각각의 개인키가 위·변조되거나 기간이 만료되었을 시 키-건강보증센터로 보내 자신의 인증서를 폐기하도록 한다.
(3) 클라이언트와 서버는 자신들이 생성한 자기소개 인증서의 유효기간 항목 타입을 결정한다. SPKI/SDSI에서는 유효기간을 아주 짧게 명세하므로 온라인 체크를 하지 않지만, 만일 온라인 체크를 허용하려면 잘 알려진 키-건강보증센터의 URI를 유효기간 항목에 기재하고 〔그림 7〕의 순서대로 등록한다. 간단히 설명하면 아래와 같다.



M1 = $< K_{jin}$, "jin", K_{jin}, (08/20/02 - 11/25/2002) >
M2 = $< K_{h1}$, "jin", K_{jin} (08/20/02 - 11/25/2002) >
Cert1 : (M1 || E_{KRjin} (M1))
Cert2 : (M2 || E_{KRh1} (M2))
E_{KRjin} (M) : 메시지 M의 다이제스트를 개인키 KRjin으로 서명

〔그림 7〕 키-건강보증센터에 공개키 등록

① URI에 위치한 키-건강보증센터로 각각 자기소개 인증서를 등록하기 위한 웹 사이트에 접속한다.
② 서버의 웹서버는 클라이언트가 자기소개인증서의 빈 항목을 채울 수 있는 등록화면을 보낸다.
③ 클라이언트 웹 브라우저는 4-튜플로 구성된 자기소개 인증서의 각 항목을 기입하고, 이의 메시지 다이제스트를 구한 값에 서명한 내용을 키-건강보증센터에 보낸다.
④ 키-건강보증센터는 CGI에 클라이언트로부터 온 내용을 보내 인증서 DB에 등록하고 이의 결과를 클라이언트에게 통보한다.

이름-인증서에 기재되는 주체(subject)는 결국 공개키이고, 이름-인증서가 그 사용자의 개인키로 서명되어 키-건강보증센터에 보내지게 되므로 키-건강보증센터는 사용자로부터 받은 인증서의 무결성을 검증할 수 있다.

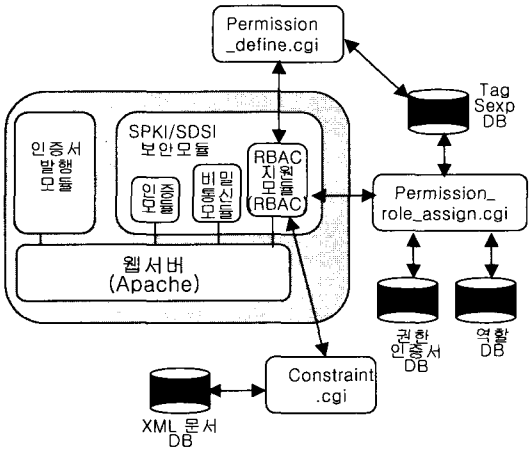
3.3 제안한 보안 프로토콜

이 절은 RBAC(RBAC)과 비밀통신을 지원하기 위해 설계된 보안프로토콜에 대해 기술한다. 〔그림 8〕은 현재 널리 공개되어 이용되고 있는 웹 서버인 아파치에 인증서 발행모듈과 비밀통신 및 RBAC 수행 보안모듈을 추가하여 만든 본 논문에서 제안한 보안서버의 전체그림이다.

3.3.1 시스템 구성

가. 임의적 접근제어를 위한 시스템 구성

서버는 클라이언트에게 권한-인증서를 직접 발행

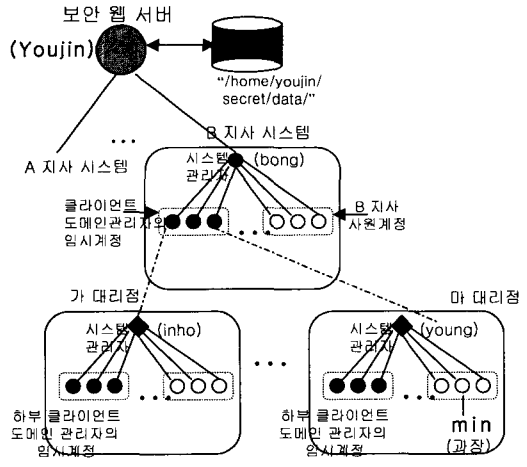


〔그림 8〕 보안서버의 구성요소

해줄 수 있다. 본 논문에서는 온라인상에서 요청자의 신원을 확인한 후 이름-인증서나 권한-인증서를 서버가 발행할 수 있는 방법을 기술하기 위해 〔그림 9〕와 같은 구조를 정의하여 이용한다.

클라이언트가 인증서를 발행받기 위해서는 클라이언트들이 소속되어 있는 최소단위인 도메인을 구성하고 각 도메인마다 도메인 관리자를 두어 각 도메인의 구성원과 도메인 관리자와는 상호인증이 이루어지도록 한다. 클라이언트 도메인들은 서버를 루트(root)로 하는 계층적인 구조를 이루며 서버는 자신과 연결된 바로 아래 계층의 클라이언트 도메인과 상호인증이 이루어진다.

사전에 상위 계층의 도메인 관리자는 아래 계층의 도메인관리자에게 이름-인증서를 발행한다. 지사 B는 본사로부터 이름-인증서를 발행받고, 가대리점은 지사 B로부터 이름-인증서를 발행받는다. 이름-인증서는 인증서를 발행하는 기관의 공개키에 기반을 둔 지역이름을 인증서를 발행받는 지사나 대리점 또는 사원의 공개키에 바인딩(증명번호, 공개키)한 것이다. 따라서 가대리점은 서버로부터 발행받은 지사 B의 이름-인증서와 지사 B로부터 발행받은 이름-인증서를 차례로 자신의 인증서-캐쉬에 보관하고 있다.



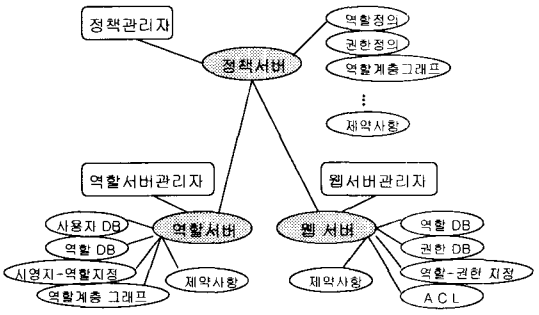
〔그림 9〕 클라이언트와 보안 웹 서버의 계층적 구조

본사(서버)는 본사직원의 계정과 임시적인 지사의 시스템 관리자 계정을 가지고 있고 지사 직원의 신상정보는 지사에서만 관리한다. 또한 지사는 지사직원의 계정과 대리점 시스템 관리자의 임시계정을 지니는 계층적인 구조를 이루고 있다.

나. 역할기반 접근제어를 위한 시스템 구성

본 논문에서의 웹 서버가 RABC을 수행하기 위해서는 사전에 〔그림 10〕과 같이 정책관리자는 역할과 권한을 정의하고, 역할서버 관리자에게 정의된 역할과 그 역할에 배정될 수 있는 사용자에 대한 제약사항, 그리고 역할계층 그래프를 전달하고, 웹 서버 관리자에게는 역할과 권한, 그리고 제약사항을 전달한다.

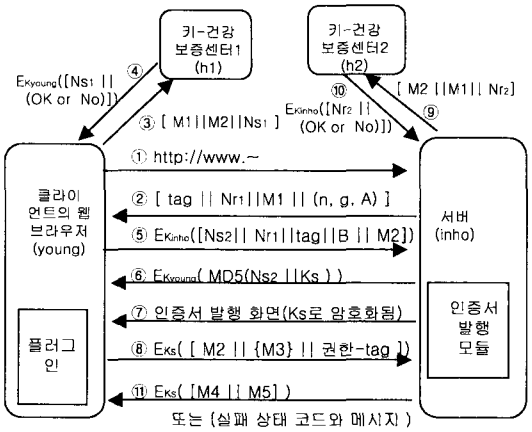
역할서버 관리자는 전달받은 역할과 사용자들을 제약사항에 맞게 배정하여 역할-인증서를 클라이언트에게 발행하고, 그 정보를 데이터베이스에 저장한다. 한편 웹서버 관리자는 역할의 업무에 해당하는 권한을 배정하여 해당하는 디렉토리의 .htaccess 파일안의 ACL 파일에 반영하는 역할-권한 배정을 수행한다.



〔그림 10〕 역할서버와 웹서버, 그리고 정책서버와의 관계

3.3.2 SPKI/SDSI 이름/권한 인증서 발행 프로토콜

이름/권한 인증서 발행모듈은 〔그림 11〕과 같은 절차를 밟아 SPKI/SDSI 이름-인증서나 권한-인증서를 클라이언트나 클라이언트 도메인 관리자에게 발행해주는 일을 한다. 개별적인 클라이언트도 서버로부터 이름-인증서나 권한-인증서를 발행받을 수



M1 = < Kinho , "inho", Kinho , ("online" "onetime" " h1의 URI") >
M2 = < Kyoung, "young", Kyoung ,("online" "onetime" " h1의 URI") >
M3 : 서버에 제출할 클라이언트 도메인으로부터 받은 이름-인증서들로 다음과 같이 구성되어 있다. m1|| m2 ||...|| mn
{M3} : 이름-인증서 뒤에 인증서 발행자의 개인키로 이 이름-인증서의 메시지 다이제스트를 전자서명한 값이 각각 붙어있는 것
M4 = < Kinho , "young", Kyoung , ("online" " h1의 URI") >
M5 = < Kinho, Kinho young, true, 권한-tag, (03/20/02 - 12/25/02) >
[L||M] : L || M, 그리고 (L||M)의 메시지다이제스트를 송신자의 개인키로 서명한 서명값으로 구성된 sequence.
Nr1, Nr2: 서버가 발생시킨 임의의 난수
Ns1, Ns2 : 클라이언트가 발생시킨 임의의 난수
MD5(M || N) : M과 N의 메시지 다이제스트 값
권한-tag : 클라이언트가 허가받고 싶은 권한을 기입한 tag-Sexp
n: 임의의 큰 수 g : n보다는 작고 1보다는 큰 수
A = g^x mod n B= g^y mod n
Ks = $(B)^x$ mod n = g^{xy} mod n
EKyoung(M) : 공개키 Kyoung으로 메시지 M을 암호화

〔그림 11〕 SPKI/SDSI 이름/권한-인증서 발행 프로토콜

있는데 이때 클라이언트는 증명자료로 상위 클라이언트 도메인 관리자로부터 발행받은 이름-인증서들도 함께 증명자료로서 서버에게 보낸다.

서버는 클라이언트의 자기소개 인증서와 그 클라이언트가 소속되어 있는 클라이언트 도메인 관리자로부터 발행받은 이름-인증서들을 검증함으로써 클라이언트를 인증할 수 있는 것이다. 이름/권한 인증서를 발행받는 절차는 아래와 같다.

① 먼저 클라이언트는 서버에 이름-인증서와 권한-인증서를 발행해주는 홈페이지에 접속한다.

② 서버는 클라이언트 도메인 관리자나 클라이언트, 또는 역할서버 관리자로부터 온 요청 메시지가 .htaccess로 보호되어 있음을 알고 .htaccess의 내용을 분석한다. 웹서버는 Sethandler에 "spki_name_auth_delegator"로 지정되어 있으므로 클라이언트로부터 받은 요청메시지를 이 인증서 발행모듈로 넘긴다. 인증서 발행모듈은 클라이언트 도메인 관리자나 클라이언트, 또는 역할서버 관리자에게 서버자신의 자기소개-인증서, Content-Type이 "application/x-spki-sdsi-ii"라는 것과 클라이언트의 요청 메시지중 추출한 tag와 임의의 난수 Nr_1, 그리고 Diffie-Hellman 인수(n, g, A)를 자신의 개인키로 서명한 순서열을 맞춤형 에러메시지와 함께 보낸다.

③,④ 클라이언트는 서버로부터 온 응답의 Content-Type "application/x -spki-sdsi-ii"에 맞는 웹 브라우저 플러그-인을 구동시켜 초기화한다. 이때 플러그-인은 사용자에게 패스워드 창을 띄워 사용자 인증을 수행한 후 사용자의 개인키를 메모리에 로드하고, 이 세션이 끝날 때까지 유지한다. 그리고 이 플러그-인은 서버의 자기소개-인증서를 온라인 체크하기 위한 과정을 밟는다. 먼저 플러그-인은 서버로부터 온 자기소개-인증서와 클라이언트의 자기소개-인증서, 그리고 클라이언트가 발생시킨 임의의 난수 Ns_1을 키-건강보증센터1로 보낸다. 키-건강보증센터1은 자신의 데이터베이스를 검색하여 검사하는 인증서가 유효한지를 검사하고, 그 결과를 클라이언트로부터 받은 임의의 난수와 함께 클라이언트의 공개키로 암호화하여 보낸다.

⑤ 클라이언트 플러그-인은 서버와의 비밀통신을 수행하기 위한 세션키 Ks를 설정하기 위해 Diffie-Hellman 인수 B와 자신이 새로 만든 임의의 난수 Ns_2, 서버로부터 받은 tag와 임의의 난수 Nr_1, 그리고 클라이언트 자신의 자기소개-인증서를 묶어 서명한 후 서버의 공개키로 암호화하여 보낸다.

⑥ 서버는 클라이언트로부터 온 난수 Ns_2와 세션키 Ks를 입력으로 메시지 다이제스트를 구해 클라이언트의 공개키로 암호화해 클라이언트에게 보낸다.

⑦ 서버는 방금 설정된 Ks키로 인증서 발행 화면을 암호화하여 클라이언트의 플러그-인에게 보낸다.

⑧ 클라이언트 플러그-인은 클라이언트 자신의 자기소개-인증서와 클라이언트 자신의 사용자 인증에 도움을 주는 클라이언트 도메인 관리자로부터 받았던 관련 이름-인증서, 그리고 권한허가를 받고자 하는 권한-tag Sexp을 세션키 Ks로 암호화하여 서버의 인증서 발행모듈로 보낸다.

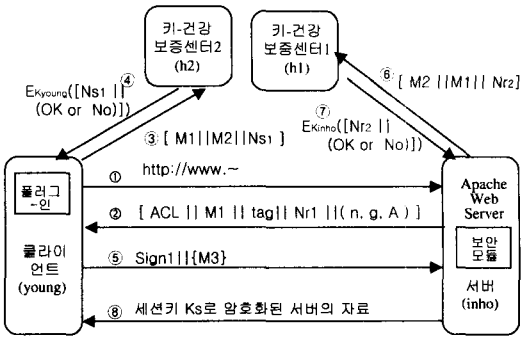
⑨⑩ 서버는 클라이언트로부터 온 메시지를 복호화한 후 필요에 따라 온라인 검증을 수행한다.

⑪ 인증서 발행모듈은 클라이언트에게 이름-인증서와 권한-인증서를 발행하여 세션키 Ks로 암호화하여 클라이언트에게 보낸다.

3.3.3 비밀통신을 지원하는 SPKI/SDSI 임의적 접근 제어 프로토콜

클라이언트가 서버에게 보내는 정보로는 "tag Sexp", "signature Sexp", 그리고 "certificate Sexp"가 있다. "tag Sexp"는 클라이언트가 서버의 서비스에 접속할 URL 주소를 의미하고 "signature Sexp"는 전자서명이며 "certificate sequence Sexp"는 이름-인증서와 권한-인증서들의 묶음을 의미한다. 그리고 서버가 유지하는 정보로는 "ACL Sexp"와 클라이언트가 보내온 메시지헤더에서 추출해 서버 자신이 만든 "tag Sexp"가 있다. 이들의 표기는 참고문헌 [7]에 나와있다. 확장된 SPKI/SDSI 접근제어 프로토콜은 아래와 같은 절차를 밟는다.[14]

① 클라이언트는 표준 HTTP 요청을 서버에 보낸다.
② 서버는 다음중의 하나에 해당하는 반응을 한다.
ⓐ 클라이언트의 요청이 공개가능한 파일에 대한 것이라면 상태코드 "200 OK"와 함께 요청된 파일을 클라이언트로 보낸다.
ⓑ 클라이언트가 요청한 파일이 SPKI/SDSI ACL에 의해 보호받는 파일이라면 서버는 이 요청메시지를 spki/sdsi 보안 모듈인 "sdsi_auth_rbac_conficom"으로 보낸다. 이제부터는 이 보안모듈과 클라이언트간에 통신이 이루어진다. 이 보안모듈은 6가지의 정보가 들어있는 메시지를 [그림 12]의 ②와 같이 클라이언트로 보낸다.
클라이언트에 보내지는 메시지로는 메지지 헤더인 컨텐츠 타입(Content-Type)이 "application/x-spki-sdsi-i"과, 메시지 바디에 있는 태그(tag) Sexp, SDSI_ACL Sexp, 비밀통신을 위한 Diffie-Hellman 키교환 인수들(n, g, A)과 서버자신의 자기소개-인증서, 그리고 서버가 발생한 임의의 난수 Nr_1을 상태코드 "403 Forbidden"과 이에 따르는 맞춤형 error를 클라이언트에게 보낸다.
③,④ 클라이언트는 서버로부터 온 응답 메시지의 내용을 분석하고 곧 Content-Type이 "application/x-spki-sdsi-i에 맞는 웹 브라우저 플러그-인을 구동시켜 초기화한다. 이때 플러그-인은 사용자에게 패스워드 창을 띄워 사용자 인증을 수행한 후 사용자의 개인키를 메모리에 로드한다. 그리고 이 플러그-인은 서버의 자기소개-인증서를 온라인 체크하기 위한 과정을 밟는다. 먼저 플러그-인은 서버로부터 온 자기소개-인증서와 클라이언트의 자기소개-인증서, 그리고 클라이언트가 발생시킨 임의의 난수 Ns_1을 키-건강보증센터1로 보낸다. 키-건강보증센터1은 자신의 데이터베이스를 검색하여 검사하는 인증서가 유효한지를 검사하고, 그 결과를 클라이언트로부터 받은 임의의 난수와 함께 클라이언트의 공개키로 암호화하여 보낸다.



M1 : < K_{inho}, "inho", K_{inho}, ("online" "onetime" "h2의 URI") >
M2 = < K_{young}, "young", K_{young} , ("online" "onetime" "h1의 URI") >
n : 임의의 큰 수 g : n보다는 작고 1보다는 큰 수
$A = g^x \bmod n$
Nr_1, Nr_2: 서버가 발생시킨 임의의 난수
Ns_1 : 클라이언트가 발생시킨 임의의 난수
Sign1 : E_{Kinho} ([tag || Nr_1 || B || M2])
M3 : ACL의 한 항목으로부터 시작해서 결국 클라이언트의 공개키로 위임된 권한 인증서들과 여기에 관여한 이름-인증서들의 묶음
{M3} : 권한-인증서와 권한-인증서 발행자의 전자서명이 함께한 묶음
$B = g^y \bmod n$
[A||B] : A와 B, 그리고 이들 메시지다이제스트를 송신자의 개인키로 전자서명한 값으로 구성된 sequence
$Ks = (B)^x \bmod n = g^{xy} \bmod n$
E_{Kyoung}(M) : 공개키 Kyoung으로 메시지 M을 암호화

[그림 12] 비밀통신 지원하는 SPKI/SDSI 임의적 접근제어 프로토콜

⑤ 플러그-인은 키-건강보증센터2로부터 온 응답내용을 분석하고 다음과 같은 일을 한다.
ⓐ 플러그-인은 "Certificate Chain Discovery" 알고리즘과 자신의 인증서 캐시에 있는 이름-인증서와 권한-인증서들을 이용하여 서버로부터 온 ACL의 한 항목으로부터 자신의 공개키에 이르는 권한-인증서들의 묶음을 찾아내 묶는다.
ⓑ 플러그-인은 서버로부터 온 태그와 자신이 생성한 임의의 난수 Nr_1, 서버와의 비밀통신을 위한 키 교환인수 B를 생성하여 자신의 자기소개-인증서와 묶은 순서열을 생성하고 이를 자신의 개인키로 서명한다.
ⓒ 이제 위에서 생성한 tag Sexp, signature를 포함한 순서열 Sexp, certificate sequence Sexp 들을 [그림 12]의 ⑤와 같이 서버에게 되

돌린다. 이때 인증서 순서열은 널(null)값이 될 수도 있다. 왜냐면 클라이언트가 인증서를 지니고 있지 않거나, 서버의 ACL의 주체항목에 클라이언트의 공개키가 직접 지정되어 있어 서버가 자신에게 직접 발행해준 권한-인증서만 제출하여도 되기 때문이다.

⑥,⑦ 서버의 spki/sdsi 보안모듈인 "sdsi_auth_rbac_conficom"은 클라이언트 플러그-인으로부터 온 tag와 자신이 생성한 tag, 그리고 이름-인증서들을 인증모듈로 보내 ⑥,⑦을 거친 인증을 마치고, 클라이언트의 권한 허용여부를 검사한다.

⑧ 권한을 지니고 있으면 요청한 데이터를 암호화해서 [그림 12]의 ⑧과 같이 클라이언트로 보낸다. 만일 실패하면 "403 forbidden" 상태코드와 맞춤형 에러페이지를 되돌려준다.

3.3.4 비밀통신과 RABC을 지원하는 SPKI/SDSI 프로토콜

권한-인증서에 포함되어있는 위임비트의 참·거짓에 따라 연속적인 위임여부가 결정되는데, 이는 위임자의 의사에 따라 임의대로 위임이 이루어지는 임의적 접근제어(DAC)를 가능하게 한다. 임의적 접근제어는 한 클라이언트가 위임에 의해 여러 개의 역할을 수행하는 경우가 발생할 수도 있기 때문에 임무분리나 최소권한위임 등에 위배되는 행위를 할 수 있는 부작용이 따를 수 있다. 이를 방지하기 위해서는 역할에 기반을 둔 접근제어가 이루어져야 하는데, 이 절은 RABC을 가능하게하는 SPKI/SDSI RBAC 프로토콜을 제안한다.

이 프로토콜은 웹서버의 ACL의 권한을 받는 대상인 대상주체가 역할서버를 기반으로 하는 지역이름으로만 되어 있는 경우로 클라이언트에게 직접 권한-인증서를 발행하지 않고 클라이언트의 모든 역할을 역할서버가 배정하도록 한 것이다.

비밀통신과 RBAC을 지원하는 SPKI /SDSI 접근제어 프로토콜은 위의 2)절 가)에 있는 임의적 접근제어 프로토콜의 일부를 수정하여 클라이언트가 자신의 역할-인증서를 역할서버로부터 얻어오는 절차를 추가한 것으로 다음과 같다.

①,②,③,④는 3.3.3절의 비밀통신을 지원하는 SPKI /SDSI 임의적 접근제어 프로토콜의 ①,②,③,④와 같다.

⑤ 클라이언트 플러그-인은 자신의 인증서 캐시에서 "Certificate Chain Discovery" 알고리즘을 이용하여 ACL의 한 항목으로부터 자신의 공개키에 이르는 권한-인증서들을 찾는다. 성공하면 ⑦로 제어가 넘어간다. 만일 생성하지 못하면 다시 ACL의 주체항목이 역할서버의 공개키에 기반을 둔 지역이름인지를 검사하고 그렇다면 역할서버에 자신의 역할-인증서를 요청한다. 그것도 아니면 인증서가 없는 경우이므로 처리를 마친다.

⑥ 역할서버는 클라이언트의 사용자-역할 배정으로부터 역할-인증서를 클라이언트에게 발행한다. 클라이언트는 역할서버로부터 온 역할-인증서는 유효한 기간동안에는 인증서 캐시에 저장하여 이후 이 역할-인증서를 이용한다.

⑦ 클라이언트는 다시 ①과 같이 평범한 요청메시지를 웹 서버에 보낸다.

⑧ 앞의 3.3.3에서 설명한 비밀통신을 지원하는 SPKI/SDSI 임의적 접근제어 프로토콜의 ②의 ⓐ,ⓑ와 같은 일을 반복한다.

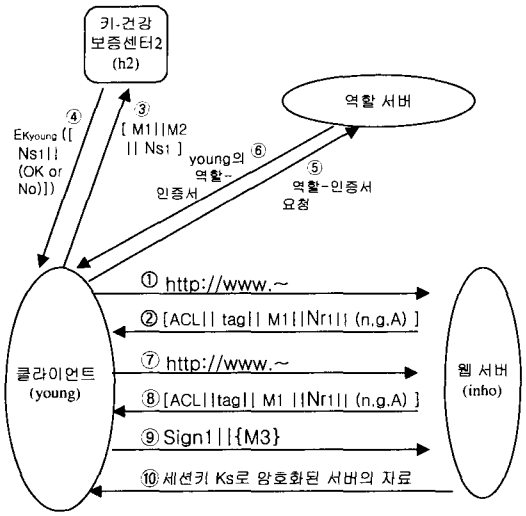
⑨ 클라이언트는 서버로부터 온 응답 메시지의 내용을 분석하고 곧 Content-Type이 "application/x-spki-sdsi-i임을 알고, 다음과 같은 일을 한다.

ⓐ 플러그-인은 "Certificate Chain Discovery" 알고리즘과 자신의 인증서 캐시에 있는 이름-인증서와 권한-인증서들을 이용하여 서버로부터 온 ACL의 한 항목으로부터 자신의 공개키에 이르는 권한-인증서들의 묶음을 찾아내 묶는다. 이제는 역할서버로부터 발행받은 역할-인증서가 있으므로 찾아낸다.

ⓑ 플러그-인은 서버로부터 온 태그와 자신이 생성한 임의의 난수 Nr_1, 서버와의 비밀통신을 위한 키 교환인수 B를 생성하여 자신의 자기소개-인증서와 묶은 순서열을 생성하고 이를 자신의 개인키로 서명한다.

ⓒ 이제 위에서 생성한 tag Sexp, signature를 포함한 순서열 Sexp, certificate sequence Sexp 들을 [그림 13]의 ⑨와 같이 서버에게 되돌린다. 이때 인증서 순서열은 널(null)값이 될수도 있다. 왜냐면 클라이언트가 인증서를 지니고 있지 않거나, 서버의 ACL의 주체항목에 클라이언트의 공개키가 직접 지정되어 있어 서버가 자신에게 직접 발행해준 권한-인증서만 제출하여도 되기 때문이다.

⑩ 권한을 지니고 있으면 요청한 데이터를 암호화해서



M1 = < K inho "inho", K inho, ("online" "onetime" "h2의 URI") >
M2 = <Kyoung, "young", Kyoung, ("online" "onetime" "h1의 URI") >
Nr1, Nr2: 서버가 발생시킨 임의의 난수
Ns1 : 클라이언트가 발생시킨 임의의 난수
n = 임의의 큰 수 g = n보다는 작고 1보다는 큰 수
A = $g^x \bmod n$ B = $g^y \bmod n$
M3 = 클라이언트의 역할-인증서 또는 권한-인증서들, 그리고 이름-인증서들의 묶음
{M3} : 권한-인증서와 권한-인증서 발행자의 전자서명이 함께한 S-exp sequence
[L||M] : L과 M, 그리고 이들 메시지다이제스트를 송신자의 개인키로 전자서명한 값으로 구성된 sequence.
Sign1 : EKinho ([tag || Nr1 || B ||M2])
Ks = (B) $^{x}\bmod n = g^{\tilde{m}bd\ n}$
EKyoung(M) : 공개키 Kyoung으로 메시지 M을 암호화

〔그림 13〕 비밀통신과 RBAC을 지원하는 SPKI/ SDSI 프로토콜

〔그림 13〕의 ⑧과 같이 클라이언트로 보낸다. 만일 실패하면 "403 forbidden" 상태코드와 맞춤형 에러페이지를 되돌려준다.

3.3.4 제약사항 정의

임의적 접근제어에 SPKI/SDSI 권한-인증서를 이용하는 경우 위임비트에 true 또는 false를 세트할 수 있는 관계로 처음 발행자가 클라이언트에게 권한을 위임한 이후 얼마든지 권한위임이 일어날 수 있다. 권한위임은 상호 신뢰를 바탕으로 이루어지는데 현실에서는 맞지 않는 경우도 있다.

따라서 본 논문의 보안모듈은 SPKI/SDSI 권한-인증서의 단점을 줄이기 위해 제약사항을 〔그림 14〕와 같이 XML로 정의하여 이를 접근제어에 반영될 수 있도록 한다. 이 XML은 .htaccess 파일속에 지시항으로 나타낸다. 제약사항은 통신 당사자들에게 전달되는 것이 아니고 접근제어를 수행하는 서버가 접근제어시 참조하는 수단이고 또한 제약사항은 확장

```
<?xml version = "1.0"?>
<constraint>
   <name ID="해당 subject 이름">
        <depth> 5 </depth>
        <bad_subject>
             <file_name>/sdsi/data/name1.txt
             </file_name>
             <file_name>/sdsi/data/group.txt
             </file_name>
        </bad_subject>
   <name>
   <group ID="해당 subject 이름">
        <depth> 3 </depth>
   </group>
</constraint>
```

〔그림 14〕 XML로 정의된 제약사항

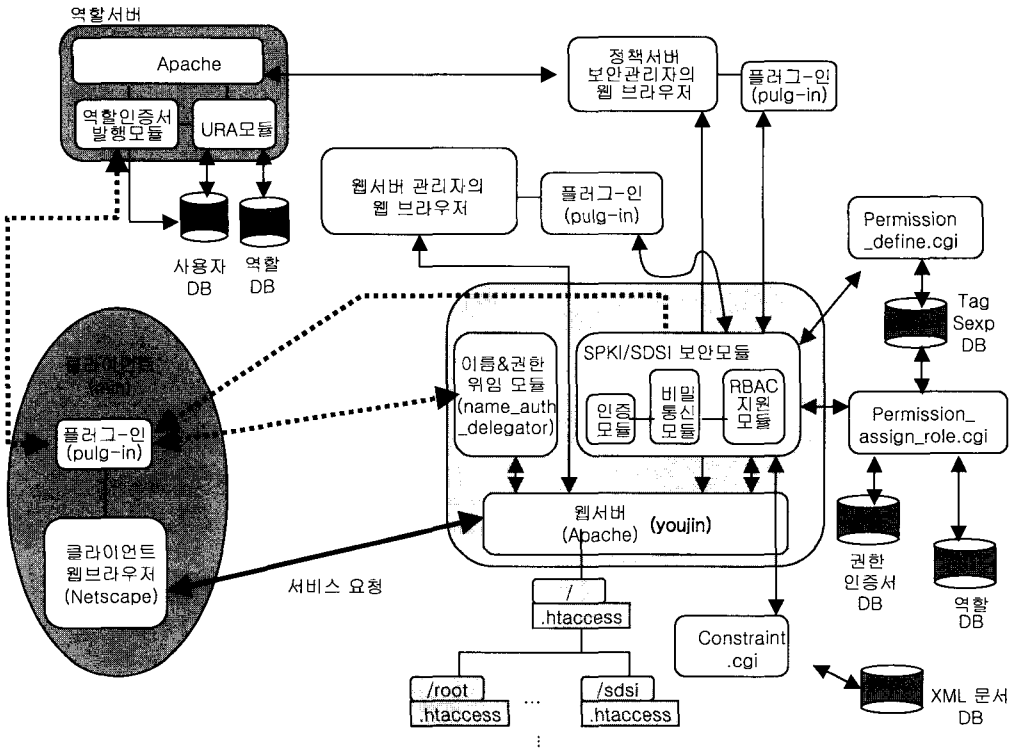
될 수 있으므로 구조와 의미 정의에 확장성이 뛰어난 XML을 이용한다.

이 XML로 정의된 내용이 의미하는 바는 .htaccess 내에 지정된 spki-sdsi ACL 파일속의 각 항목들에 제약을 가할 수 있음을 나타낸다. 즉 depth가 5인 경우 ACL 항목들 중 해당 subject 이름으로부터 시작한 권한위임은 자신을 기점으로 다섯 번의 권한위임을 허용한다는 의미로 그 이상의 권한위임으로 인한 권한-인증서를 지닌 클라이언트는 접근이 허용되지 않음을 나타낸다.

Ⅳ. 웹에서 적용 예

이 절은 본 논문에서 설계한 보안 웹 서버의 프로토타입이 실제 웹에서 어떻게 수행되는지를 예로써 설명한다. 앞의 〔그림 9〕는 보안 웹 서버 프로토타입을 위해 구체적인 시나리오를 설정한 클라이언트와 보안 웹 서버간의 관계를 계층적 구조를 나타낸 것이다. 본사의 모든 직원과 지사의 직원 중 직위가 과장 이상인 사람들만 볼 수 있는 정보를 비밀장소에 저장해놓고 SPKI/SDSI ACL로 관리한다고 가정한다. C지사의 직원 중 min이라는 계정을 지닌 자의 직급은 과장이라고 가정한다.

보안 웹 서버는 본사에서 관리하고 있으며, 마대리점의 직원 min은 이미 young으로부터 이름-인증서를 발행받았고, 역할서버로부터 역할-인증서를 발행받을 예정이며 웹 서버 youjin은 과장이라는 역할에 자신의 웹 페이지를 접근할 수 있는 권한을 부여한 ACL을 유지하고 있다. 〔그림 15〕는 클라이언트와 보안 웹 서버, 그리고 역할서버의 통합구조를 나타낸 것이다. 동작되는 과정을 설명하면 다음과 같다.



〔그림 15〕 클라이언트와 웹서버의 통합된 구조

1. min은 보안 웹 서버의 관리자 youjin의 홈페이지를 방문하여 "2003 vision"이라는 아이콘을 클릭한다.
2. youjin의 웹서버는 이 아이콘에 연결된 정보가 위치하고있는 디렉토리에 〔그림 16〕과 같은 내용이 들어있는 .htaccess가 있음을 알고 이를 분석한다. SetHandler가 기본적인 인증모듈이 아님을 알고 웹서버는 sdsi_auth_rbac_confidencom 모듈에게 min으로부터 온 요구(request) 정보를 곧바로 넘긴다.

```
SetHandler spki/sdsi_auth_rbac_confidencom
AclFile /home/youjin/secret/secdata_acl
ErrorFile /home/youjin/Error/error.html
ConstraintFile /home/constraint/aclconstraint.xml
```

〔그림 16〕 .htaccess 파일의 예제

```
HTTP/1.1 403 Forbidden
Date: Wed, 3 Jul 2002 03:07:02 GMT
Server: Apache/1.3.19 (Unix) PHP/4.0.6
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/x-spki-sdsi

Host: athena.chonnam.ac.kr
Port: 8089
URI: /home/youjin/secret/data/vision2003.html
URL: http://athena.chonnam.ac.kr:8089/home/
        youjin/secret/data/vision2003.html
ACL:(acl (entry (subject (name (public-key (rsa -
    pkcs1-md5( e #23#) (n |GV9v77A2VRLR
    -jhh+fi w==|))) section-chief)
    (tag(http(* set GET) (* prefix
    http://athena.connam.ac.kr:8089/home/
      youjin/secret/data/)))))
SPKI-SDSI-sequence : (sequence
      ( 서버의 자기소개 인증서 || (n, g, A, T )
      || 앞 항목들에 대한 메시지다이젯트에
                서버 개인키로 서명한 값 ) )
```

그림 17 서버가 클라이언트에게 보내는 단서

3. sdsi_auth_rbac_confidencom 모듈은 클라이언트로부터 온 요구메시지가 아무런 인증서도 가지고 있지 않음을 알고 클라이언트에게 그림 17과 같이 ACL과 Diffie-Hellman 인수들을 메시지 몸체부분에 추가하여 보낸다.
4. 클라이언트 웹브라우저는 서버로부터 온 메시지 타입이 x-spki-sdsi임을 알고 관련 플러그인을 메모리에 적재한다. 서버로부터 온 메시지들을 인증한 후, 자신이 인증서 캐쉬에 보관하고 있는 이름-인증서와 권한-인증서들 중 서버의 ACL과

관련있는 인증서들을 "certificate chain discovery" 알고리즘을 이용하여 찾는다.

그리고 이 관련있는 후보 인증서들 중 ACL의 각 항목으로부터 시작해서 크라이언트의 공개키에 이르는 권한 연결이 있는 것을 묶고, 또한 서버와 한 세션동안 공유할 비밀키를 생성하기 위해 키교환을 위한 인수값들을 결정하여 [그림 18]과 같은 HTTP 메시지를 만들어 서버에게 준다.

만일 인증서들을 찾지 못하면, ACL의 대상주체(subject)가 역할서버의 공개키에 기반을 둔 지역이름인가를 살펴본다. 지역이름이 역할서버로부터 지정한 역할이름임을 알고, 플러그인은 역할서버에 역할인증서를 발행해줄 것을 요청한다. 역할서버로부터 역할-인증서를 발행받은 클라이언트는 다시 서버로 접속을 시도하기 위해 과정 1로 되돌아간다.

그것도 아니면 클라이언트에게는 권한이 없음을 통보하고 권한인증서를 발행받아야만 가능하다는 맞춤형 에러메시지를 화면에 나타내고 끝낸다.

```
GET /home/youjin/secret/data/vision2003.html HTTp/1.1
User-Agent: SPKI/SDSI2 dogu-arcon Plug-in 1.0
Host: athena.chonnam.ac.kr:8089
Connection: Keep-Alive

SPKI-SDSI-sequence : ( sequence ( ( tag-Sexp ) ||( B )
                              (signature (( tag) || (B)의 해쉬값) )
SPKI-SDSI-sequence : (sequence ( 권한-인증서들의 모음)
                                    ( 역할-인증서) )
```

[그림 18] 서버로 향하는 클라이언트의 입증자료들

5. 서버는 클라이언트로부터 온 인증서들을 검증해보고 정당한 사용자이면 클라이언트와 동시에 설정한 비밀키를 이용하여 관련정보를 암호화하여 보내고 인증에 실패하면 맞춤형 에러메시지를 클라이언트에게 보낸다.

V. 결과 및 분석

5.1 제안된 보안 프로토콜의 안전성 분석

가. SPKI/SDSI 인증서 발행 프로토콜의 안전성

[정리 1]

임의적 접근제어를 위한 SPKI/SDSI 이름/권한-인증서 발행 프로토콜은 메시지 변형공격, 메시지 인증과 사용자 인증, 그리고 메시지 재전송, 송·수신 부인 공격에 대해 안전하다.

(증명)

먼저 클라이언트의 전송경우 [그림 11]의 ⑤와 ⑧에 변형을 생각하면, ⑤에서 보내지는 메시지는 서버의 공개키로 암호화되어 전송되므로 서버만이 복호화할 수 있다. 또한 보내지는 메시지는 클라이언트의 개인키로 서명되므로 중간자에 의해 변형될 수 없다. ⑧의 경우도 클라이언트의 개인키로 서명되고 이를 세션키로 암호화하여 보내므로 중간자로부터 안전하다.

반대로 서버측의 전송 경우 [그림 11]의 ②와 ⑥, 그리고 ⑪을 고려하면, ②의 경우는 중간자가 서버의 개인키를 모르고는 메시지 변형을 할 수 없고, ⑥의 경우 클라이언트의 개인키를 아는자만이 복호화할 수 있다. 그리고 ⑪은 ②와 ⑤가 중간자에 의해 공격당했을 경우만 가능하므로 안전하다.

중간자가 서버인체 흉내내는 경우 서버의 중간자는 전자서명된 부분의 일부만을 수정할 수는 없고, 또한 제안한 프로토콜의 기능을 파괴하지 않으므로 무시할 수 있다.

중간자가 클라이언트를 흉내내는 경우 [그림 11]의 ⑤와 ⑧을 고려해보자. ⑤의 전송되는 메시지는 임의대로 변형되어 중간자의 개인키로 암호화되어 보내질 수가 있다하더라도, ⑧의 단계 메시지에 포함되어야하는 이름-인증서들에 덧붙여지는 인증서 발행자의 서명은 위조할 수 없으므로 서버가 감지한다.

클라이언트와 서버는 한 세션동안 Ks를 이용해 인증서들을 암호화해서 보낸다. 클라이언트는 Ns_1과 Ns_2, 서버는 Nr_1과 Nr_2를 생성해 각각 순서열에 포함시켜 서명해 전송한다. 따라서 새로운 세션에 메시지 재전송되면 이를 각각은 감지한다.

[그림 11]의 ⑧과 ⑪ 이외의 요소들은 프로토콜의 기능상 중간과정에 불과하므로 의미가 없다. ⑧은 클라이언트의 개인키로 서명되어 보내지게 되고, ⑪은 서버의 개인키로 암호화되어 보내지게 되므로 전자서명의 성질에 의해 양자는 송·수신을 부인할 수 없다. □

[정리 2]

임의적 접근제어를 위한 SPKI/SDSI 이름/권한-인증서 발행 프로토콜은 사용자 메시지 변형 공격, 사용자 인증, 그리고 메시지 재전송 공격에 대해 안전하다. 에 대해 안전하다.

(증명)

[그림 12]의 ⑤에서 메시지가 변형될 경우, 메시

지의 앞부분은 서버의 공개키로 암호화되어 있으므로 서버만이 복호화할 수 있고, 뒷부분은 발행자의 전자서명이 들어있는 권한-인증서의 묶음이므로 변형되면 검증시 발견된다. 반대로 ②단계의 메시지는 서버의 개인키로 전자서명되어 있고, ⑧의 경우는 ②와 ⑤가 공격당하였을 때만이 비로소 세션키가 발각된다.

중간 공격자가 서버를 행세하는 것은 프로토콜의 기능을 파괴하지 않으므로 무시하고, 클라이언트를 흉내낼 경우인 [그림 12]의 ⑤를 고려해보자. 전송되는 메시지는 서버의 공개키로 암호화되고, 또 설령 서버의 개인키가 발각되었더라도, 클라이언트가 서버에게 제공하는 권한-인증서들이 발행자의 전자서명이 붙어 전달되므로, 중간자는 이를 수정할 수 없다. 따라서 서버는 정당한 클라이언트가 아님을 금방 알 수 있다.

클라이언트와 서버는 한 세션동안 Ks를 이용해 인증서들을 암호화해서 보낸다. 클라이언트는 Ns_1과 Ns_2, 서버는 Nr_1과 Nr_2를 생성해 각각 순서열에 포함시켜 서명해 전송한다. 따라서 새로운 세션에 메시지 재전송되면 이를 각각은 감지한다. □

[정리 3]

비밀통신과 RBAC을 지원하는 SPKI/SDSI 프로토콜은 메시지 인증과 사용자 인증, 그리고 메시지 재전송 공격에 대해 안전하다.

(증명)

[그림 13]의 ③, ④단계를 제외하고는 비밀통신을 지원하는 SPKI/SDSI 임의적 접근제어 프로토콜과 같다. 따라서 정리2에 증명된바와 같이 비밀통신과 RBAC 을 지원하는 SPKI/SDSI 프로토콜은 메시지 인증, 사용자 인증, 그리고 메시지 재전송에 대해 안전하다. □

5.2 Geronimo와 제안된 보안 프로토콜의 비교

이름/권한-인증서 발행모듈은 위의 프로토콜 중 서버측에서 행해야하는 일을 수행하는 프로시쥬어들의 작은 모임으로 구성되어 있다.

이름/권한 인증서 발행을 위한 보안 프로토콜은 tag와 자기소개 인증서, 그리고 Diffie-Hellman 파라미터들을 입력으로한 메시지 다이제스트에 서버의 개인키로 서명한 서명값이 순서열(sequence)의 S-expression으로 표현되어 클라이언트에게 보내지게 되므로, 클라이언트가 무결성을 검증할 수 있어 중간자 공격으로부터 안전하게 보호받을 수 있다.

인증서를 이용해 웹상에서 접근제어하려는 시도는 아직 초기단계이다. X.509 v3 인증서를 이용하는 것으로는 기존의 X.509 v3의 틀은 유지하면서 확장영역에 권한을 부여하여 부여자의 개인키로 서명한 Smart 인증서를 이용하는 예가 있다.

X.509 공개키 기반구조는 비밀통신을 수행하기 위해 v3 인증서를 이용하는 SSL/TLS를 이용한다. 하지만 본 서버의 프로토타입은 본 논문에서 제안하는 프로토콜을 사용하여 비밀통신을 수행한다. 출발부터의 철학이 다르기 때문에 X.509 PKI와 SPKI/SDSI 기반구조를 단순히 비교하는 것은 의미가 없다.

Geronimo 프로젝트는 비밀통신을 수행하는 것을 가정하고 과정이 진행된다. 즉, SSL/TLS 같은 프로토콜이 곧 존재할 것이라는 가정을 하고 있다. 본 서버는 비밀통신 모듈이 이 일을 담당한다. 메시지 보안을 위한 비밀통신 기능이 들어감으로써 통신의 부하는 늘어난다. 하지만 정보의 가치에 비할 수 없으므로 이를 감수하고 보안시스템에 구현되는 추세이다.

기존의 Geronimo 프로젝트와 본 서버의 프로토타입을 비교한 결과는 [표 2]와 같다. 전자는 클라이언트의 공개키를 디스켓으로 담아오거나 아니면 SSL/TLS를 이용한다는 가정하에 진행하였다. 본 서버는 본 논문에서 제안한 비밀통신 프로토콜에 의해 온라인에서 안전하게 공개키를 등록할 수 있다.

[표 2] Geronimo 서버와 본서버의 비교

Geronimo	컨텐츠 제공	평문으로 제공
	신뢰성	클라이언트를 신뢰
	공개키 전달	공개키를 디스켓에 담아 직접 전달
	접근제어	DAC
	비밀성	X.509 v3 인증서 이용한 SSL/TLS 사용
본 서버	컨텐츠 제공	암호화된 문서 제공
	신뢰성	정의된 제약사항을 접근제어에 반영할 수 있음
	공개키 전달	온라인으로 전달
	접근제어	DAC, RBAC
	비밀성	순수한 SPKI/SDSI 인증서만을 이용

전자는 접근제어시 SPKI/SDSI 인증서가 지닌 특성으로 인한 단점을 보강할 수 있는 방법을 생각하지 않았다. 본 서버는 XML로 제약사항을 정의하여 .htaccess 파일에 지시항으로 넣어둠으로써 필요시 접근제어에 이용할 수 있다.

Ⅵ. 결론 및 향후과제

본 논문에서는 순수한 SPKI/SDSI 인증서들만을 이용하여 클라이언트가 서버와 비밀통신을 할 수 있는 프로토콜을 제안하였다. 이를 위해 자기소개 인증서와 키-사망신고서를 S-expression으로 정의하였고, 또한 원격 도메인간의 클라이언트에 권한위임을 위해 이름&권한 위임모듈을 서버에 추가하였다. 또한 접근제어에 제약사항을 반영할 수 있도록 XML로 정의하였다.

그리고 RABC을 수행하기 위해 세 가지의 CGI를 두어 정의된 역할에 권한을 배정할 수 있도록 하였다. 마지막으로는 웹상에서 적용해보기 위해 온라인에서 SPKI/SDSI 인증서를 발행받기 위한 시스템 구성을 새로 하고, 클라이언트와 서버간에 인증서를 발행받는 방법도 논의하였다. 그리고 본사아래 여러 지사가 있는 회사를 시나리오로 실제 예를 들어 본 논문에서 제시한 보안 웹 서버의 프로토타입이 웹상에서 어떻게 동작하는가를 보였다.

본 논문에서 제안한 보안 서버는 접근제어를 위해 필요한 클라이언트의 인증과 권한 검사를 거친 후 클라이언트에게 세션키 Ks로 암호화된 정보를 보내게 된다. 또한 클라이언트의 권한을 검사하는데 있어 서버차원에서 한 번 더 여과장치를 거칠 수 있도록 하였다. 본 서버는 자기소개-인증서와 Diffie-Hellman 인수를 이용하여 안전하게 온라인으로 인증서를 신청하고 발행받을 수 있으며, DAC과 RBAC에 기반한 접근제어를 수행할 수 있다. 그리고 SSL/TLS를 이용하여 비밀통신을 수행하는 대신 순수한 SPKI/SDSI 인증서만을 이용하여 비밀통신을 가능하게 하고 접근제어를 수행할 수 있다.

하지만 SPKI/SDSI 인증서를 이용한 연구가 초기단계이고, 또한 본 논문과 비교대상인 Geronimo 프로젝트의 소스가 공개되어있지 않아 Geronomo 프로젝트와 본 서버간에 실제적인 비교를 위해 교환되는 라운드 수, 교환되는 데이터의 양, 추가적으로 수행되어야 하는 연산량, 그리고 추가적으로 수행되어야 할 시간을 비교하지 못했다. 향후 본 논문에서 제안한 비밀통신 모듈을 TCP/IP 계층위에서 독립적으로 구현될 수 있도록 발전시킬 계획이며, 건강보증센터를 통해 온라인상에서 인증서를 검증할 수 있도록 이름-인증서를 안전하게 등록하는 프로토콜로 확장할 것이며, 완전한 RBAC을 수행하기 위한 세분화된 기능을 지닌 역할서버를 구축할 계획이다.

참 고 문 헌

[1] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, T. Ylonen, "Simple Public Key Certificate", Internet Draft, July 1999.

[2] M. H. Fredette, "An implementation of SDSI - the simple distributed security infrastructure", Master of thesis, M.I.T, EECS, May 1997.

[3] Gillian Elcock, "Web-based user interface for a simple distribued security infrastructure(sdsi)", Master of thesis, M.I.T, EECS, June 1997.

[4] Alexander Morcos, "A Java Implementation of Simple Distributed Security Infrastructure", Master of engineering thesis, M.I.T, EECS, May 1998.

[5] D. Clarke, J. Elien, C. Ellison, M. Fredette, A. Morcos, R. L. Rivest, "Certificate Chain Discovery in SPKI/SDSI", December 2000.

[6] Andrew J. Maywah, "An Implementation of a Secure Web Client Using SPKI/SDSI Certificates", Master of thesis, M.I.T, EECS, May 2000.

[7] Dwaine E.Clarke, "SPKI/SDSI HTTP Server/ Certificate Chain Discovery in SPKI/SDSI", Master of engineering thesis, M.I.T, EECS, May 2000.

[8] R, L. Rivest, B, Lampson, "SDSI - A Simple Distributed Security Infrastructure", September 1996.

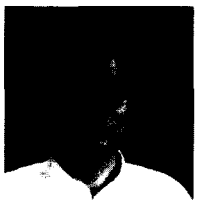
[9] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen, "SPKI Certificate Theory", RFC2693, September 1999.

[10] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, T. Ylonen, "SPKI Examples", Internet-Draft, March 1998.
[11] R. Rivest. "S-Expressions", Internet Draft, May, 1997.
[12] John Barkley, "Compating Simple Role Based Access Control Models and Access Control Lists", Proceedings of 2nd Workshop on Role-Based Access Control, August, 1997, pp. 127~132.
[13] Ravi S. Sandhu, Pierangela Samarati, "Access Control: Principle and Practice", IEEE Computer, February 1996, pp. 38~47.
[14] 이영록, 김민수, 노봉남, "SPKI/SDSI 인증서를 이용한 웹-기반 비밀키 분배 프로토콜", 한국통신학회 2002년도 하계종합학술발표회 논문집, 2002, CD Title-18E 인터넷[18-79]

〈著者紹介〉

이 영 록 (Young-lok Lee) 정회원
1986년 2월 : 전남대학교 계산통계학과 졸업
1990년 2월 : 전남대학교 전산통계학과 한국대학교 전자공학과 석사
〈관심분야〉 전자상거래 보안, 보안모델, 네트워크 보안, 정보보호 시스템

김 용 민 (Yong-min Kim)
1989년 : 전남대학교 전산통계학과 졸업
1991년 : 전남대학교 전산통계학과(이학석사)
2002년 8월 : 전남대학교 전산통계학과(이학박사)
〈관심분야〉 시스템 및 네트워크 보안, 정보보안, 네트워크 관리 등



이 형 효 (Hyung-hyo Lee) 정회원
1987년 2월 : 전남대학교 계산통계학과 졸업(학사)
1989년 2월 : 한국과학기술원 전산학과 졸업(석사)
2000년 2월 : 전남대학교 전산학과 졸업(박사)
1990년 ~ 1997년 : 삼보컴퓨터 기술연구소, 한국통신 연구개발원
2001년 3월 ~ 현재 : 원광대학교 정보·전자상거래학부 전임강사
〈관심분야〉 보안모델, 통신망 보안관리, 전자상거래보안, 침입탐지시스템

김 민 수 (Min-soo Kim) 정회원
1993년 전남대학교 전산통계학과 졸업(학사)
1995년 전남대학교 대학원 전산통계학과(이학석사)
2000년 전남대학교 대학원 전산통계학과(이학박사)
2000년 - 2001년 한국정보보호진흥원 선임연구원
2001년 - 현재 전남대학교 리눅스시스템보안연구센터 객원교수
관심분야 : 시스템 보안, 네트워크 보안, 정보보안, 신경망 등

노 봉 남(Bong-nam Nho) 정회원
1978년 전남대학교 수학교육과 졸업
1982년 KAIST 대학원 전산학과 졸업
1994년 전북대학교 대학원 전산과 졸업
1983년 - 현재 전남대학교 컴퓨터정보학부 교수
2000년 - 리눅스 보안 연구센터 소장
관심분야 : 컴퓨터와 네트워크 보안, 정보보호시스템, 전자상거래 보안, 사이버사회와 윤리