분산 컴퓨팅 환경을 위한 통합 인증 시스템
(Single Sign-On for Distributed Computing Environment)

정 연 식 † 김 인 겸 ††
(Yeonsik Jeong) (Inkyeom Kim)

요 약 본 논문에서는 공개키 기반구조에 기반한 통합 인증 시스템을 설계, 구현하였다. 통합 인증 시스템에서는 사용자가 하나의 패스워드를 이용해서 분산 컴퓨팅 환경하의 모든 응용 서버에 접속할 수 있으며, 사용자가 입력한 패스워드가 네트워크를 통해서 전송되지 않으므로 패스워드 도청 및 재사용 공격으로부터 안전하다. 제안한 시스템은 전자서명과 암호학적 인증 프로토콜을 이용하여 구현되었으며, 안정적인 시스템 설계를 위하여 메시지 시퀀스 차트와 상태 천이 다이어그램을 이용하였다.
키워드 : 인증, 전자서명, 공개키 기반구조, 통합 인증 시스템

Abstract In this paper, a single sign-on system has been designed and implemented. The purpose of the proposed system is that a user logs in once using a single password and gets authenticated access to all the servers that he is authorized to use without sending any passwords over the network. The proposed system involves the use of digital signature and cryptographic authentication protocols. For its implementation, a public-key infrastructure is also designed and implemented. To view the behavior of the proposed system, message sequence charts and state transition diagrams have been

1. 서 론

최근 정보통신 및 인터넷의 발달로 네트워크를 통한 서비스 제공의 기회가 확대됨에 따라 전송되는 정보에 대한 보호와 적법한 사용자에 대한 인증 문제가 중요한 연구 분야로 대두되고 있다. 현재 사용자를 인증하기 위한 방법으로는 패스워드 방식이 널리 사용되고 있지만, 네트워크 상에서 패스워드가 도청될 수 있다는 점 때문에 안전한 인증 방식으로 사용되기에는 문제점이 있다. 또한 다중 서버가 운영되는 인트라넷 환경에서 각 서버당 다른 패스워드를 사용해야 하는 요구사항은 사용자 및 관리자 모두에게 부담으로 작용한다. 암호 이론에 기반한 인증 프로토콜이 사용된다 하더라도 다중 서버 환경에서는 구현 및 관리가 용이하지 않다[1,2].

이러한 구현상의 여러 단점들을 극복하기 위해 통합 인증 시스템(Single Sign-On: SSO)의 개념이 도입되었다[3,4]. SSO 시스템에서는 사용자가 하나의 패스워드를 이용해서 인트라넷 환경하의 모든 응용 서버에 접속할 수 있다. 또한 사용자가 입력한 패스워드가 네트워크를 통해서 전송되지 않으므로 패스워드 재사용 공격으로부터 안전할 수 있다.

SSO를 구현하기 위해 다양한 시스템들이 개발되어져 왔으며, 그 중에서 비밀키 암호시스템에 기반한 Kerberos 시스템[5]과 공개키 암호시스템에 기반한 Netscape SuiteSpot 서버[6]가 널리 사용되고 있다. Kerberos 시스템은 중앙 인증 서버를 이용하여 사용자를 인증하고, 티켓을 발급하여 응용 서버를 사용할 수 있도록 한다. 그러나 Kerberos 시스템은 비밀키 암호시스템에 기반하므로 키관리 문제가 존재하고, 티켓에 대한 재사용 공격이 가능하다. Netscape SuiteSpot 서버는 공개키 암호시스템과 SSL(Secure Socket Layer)을 이용하여 사용자를 인증한다. 즉 사용자는 응용 서버에 접속하기 전에 중앙 인증 서버에 접속하여 컴퓨터의 인증서를 자신의 인증서로 대체한 후, SSL 핸드쉐이크 프로토콜을 수행함으로써 사용자 인증을 수행한다. 따라서 SuiteSpot 서버를 사용하기 위해서는 반드시 SSL 프로토콜이 클라이언트에 구현되어 있어야 한다.

본 논문에서는 전자서명과 암호학적 인증 프로토콜에 기반한 SSO 시스템이 설계, 구현되었다. 전자서명의 사용을 위해 공개키 기반구조(Public-Key Infrastructure:

† 정 회 원 : 성공회대학교 컴퓨터정보공학부 교수
ysjeong@mail.skhu.ac.kr
†† 비 회 원 : 성결대학교 정보통신공학부 교수
kik@sungkyul.edu
논문접수 : 2002년 9월 9일
심사완료 : 2003년 5월 19일

PKI)[7] 역시 설계, 구현되었다. 제안한 PKI는 인증기관(Certification Authority: CA) 서버, 디렉토리 서버, 클라이언트 모듈로 구성되어 있으며, SSO 시스템의 인증 프로토콜의 구현을 위해 인트라넷 각 사용자에게 인증서를 생성, 발급, 관리하는 역할을 수행한다.

제안한 SSO 시스템은 PKI, 인증 서버, 응용 서버와 클라이언트들로 구성되어 있으며, 강한 인증을 필요로 하는 인트라넷으로의 구현을 목표로 한다. 제안한 시스템은 PKI에 기반하고 있다는 점에서 Netscape의 시스템과 접근방식이 유사하다. 그러나 Netscape의 시스템은 공개키 암호시스템을 사용한다는 점을 제외하고는 중앙 서버를 통해 모든 서버로의 접근이 시작되므로 오히려 Kerberos의 접근방식과 유사하다. 제안한 시스템은 사용자가 원하는 응용 서버에 접속하면 그 응용 서버가 자동적으로 인증 서버에게 인증 서비스를 요청하는 구조로 설계되었으며, 네트워크 프로토콜 상의 응용 계층에서 구현되므로 Netscape 시스템과 달리 네트워크 및 트랜스포트 계층 프로토콜에 독립적이다.

2. 시스템 설계

2.1 시스템 개요

기존의 대부분의 시스템은 패스워드에 기반한 사용자 인증 방식을 사용하고 있으며, 패스워드 정보는 평문 또는 SSL을 통해 전송된다.

제안한 SSO 시스템은 인증 서버와 응용 서버, 클라이언트, 그리고 PKI로 구성되어 있다. 그림 1은 제안한 SSO 시스템에 대한 구성도를 보여주며, 다음과 같은 단계를 거쳐 인증을 수행한다.

Step 1. 사용자는 원하는 응용 서버에 접속한 후, 자신의 ID로 이메일 주소를 입력한다.

Step 2. 응용 서버는 사용자의 이메일 주소와 자신의 도메인 네임, 그리고 랜덤 스트링을 인증 서버로 전달한다.

Step 3. 인증 서버는 이메일 주소의 소유자가 현재 인증된 상태인지를 체크한다. 만약 인증된 상태라면 Step 5로 간다. 처음으로 인증을 요청한 상태라면 Step 4로 간다.

Step 4. 인증 서버는 클라이언트와 challenge-response 인증 프로토콜을 수행한다. 인증 프로토콜이 성공적으로 끝나면 다음 Step으로 간다. 인증이 실패한 경우에는 인증 프로토콜을 재수행한다.

Step 5. 인증 서버는 Step 2에서 받은 랜덤 스트링을 암호화하여 ticket을 생성한다. 암호화를 위한 키는 인증 서버와 응용 서버 사이에 미리 공유되어 있는 비밀키를 이용한다. 인증 서버는 ticket을 이메일 주소와 함께 응용 서버에게 전송한다.

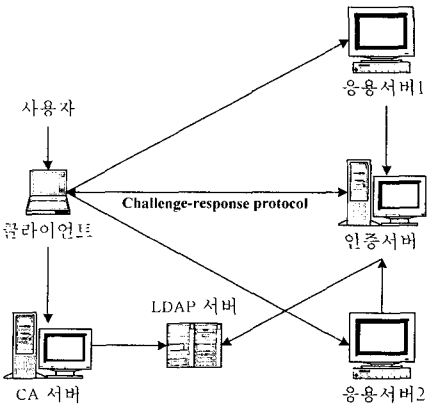

그림 1 제안한 SSO 시스템

Step 6. 응용 서버는 받은 ticket을 복호화한 후 원래 Step 2에서 생성했던 랜덤 스트링과 비교한다. 두 스트링이 일치하면 이메일 주소의 소유자를 적법한 사용자로 인증한 후 서비스를 시작한다.

2.2 설계 기준

이 절에서는 제안한 SSO 시스템의 안정적인 동작을 확인하기 위하여 메시지 시퀀스 차트(Message Sequence Chart: MSC)를 도입하였다[8]. 제안한 시스템을 구성하는 구성요소들간의 동작을 기술하기 위해 10가지의 메시지를 정의하였으며(표 1 참조), 실제 SSO의 운용에서 발생할 수 있는 다양한 시나리오와 그에 대한 MSC들을 제시하였다.

2.2.1 시나리오 1

적법한 사용자가 응용 서버에 접속하는 정상적인 경우를 나타내며, 이에 대한 MSC가 그림 2에 표현되어 있다.

메시지가 교환되는 과정은 다음과 같다. 클라이언트가 사용자의 이메일 주소(userEmail)와 함께 service_req 메시지를 응용 서버에게 보낸다. 응용 서버는 자신의 도메인 네임(serverName)과 userEmail, 랜덤 스트링(randStr)을 파라미터로 하여 auth_req 메시지를 인증 서버에게 보냄으로써 인증 서비스를 요청한다. randStr은 공격자가 인증 프로토콜을 수행하지 않고 직접 응용 서버에 접속할 경우를 방지하기 위해 사용된다.

인증 서버는 password_req 메시지를 보냄으로써 사용자에게 개인키 데이타베이스에 대한 패스워드를 요구한다. 사용자의 개인키 복호화가 성공하면 클라이언트는 S flag와 함께 password_resp 메시지를 인증 서버에게 보낸다. 인증 서버는 lightweight directory access

표 1 시스템 구성요소 간에 교환되는 메시지 정의

메시지	파라미터	주체
service_req	URI, userEmail	클라이언트(사용자) ↔ 응용 서버
service_resp	flag	
auth_req	serverName, userEmail, randStr	응용 서버 ↔ 인증 서버
auth_resp	userEmail, ticket	
password_req		인증 서버 ↔ 클라이언트(사용자)
password_resp	flag	
cert_req		인증 서버 ↔ LDAP 서버
cert_resp	certificate	
signature_req	randNum	인증 서버 ↔ 클라이언트(사용자)
signature_resp	signedData	

protocol(LDAP)[9] 서버로부터 사용자의 인증서를 요청한다. 사용자의 엔트리와 인증서가 LDAP 디렉토리에 존재하면 인증서를 cert_resp 메시지의 파라미터로 전송한다. 인증서를 전송받은 인증 서버는 클라이언트와 signature_req/signature_resp 메시지를 이용하여 인증 프로토콜을 수행한다.

인증 프로토콜이 성공적으로 수행되면 인증 서버는 응용 서버에게 userEmail과 ticket을 파라미터로 하여 auth_resp 메시지를 보낸다. ticket과 randStr은 쌍으로 동작하며, randStr은 응용 서버와 인증 서버 사이에 공유된 비밀키를 이용하여 암호화되어 ticket으로 전송된다. 응용 서버가 ticket을 성공적으로 복호화하면 클라이언트에게 S flag와 함께 service_resp 메시지를 전송함으로써 인증 과정을 마친다.

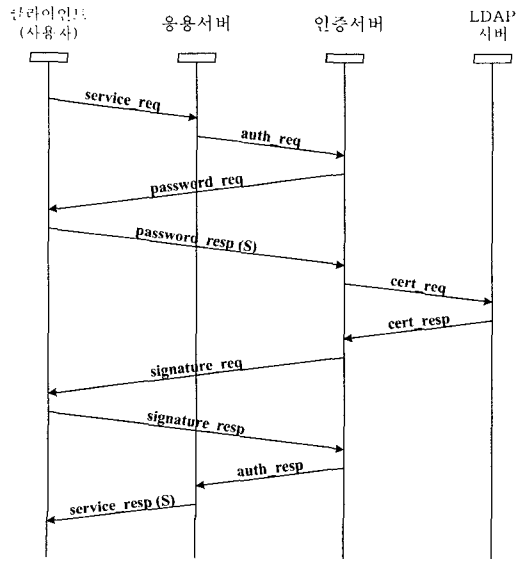

그림 2 시나리오 1: 정상적인 경우

2.2.2 시나리오 2

시나리오 2는 시나리오 1의 경우에서 사용자가 잘못된 password를 입력했을 경우를 나타낸다. 클라이언트는 사용자의 개인키를 복호화할 수 없으므로 인증 서버에게 F flag와 함께 password_resp 메시지를 인증 서버에게 보낸다. 인증 서버는 계속해서 정확한 패스워드의 입력을 요구하는 password_req 메시지를 전송한다. 시나리오 2에 대한 MSC가 그림 3에 표현되어 있다.

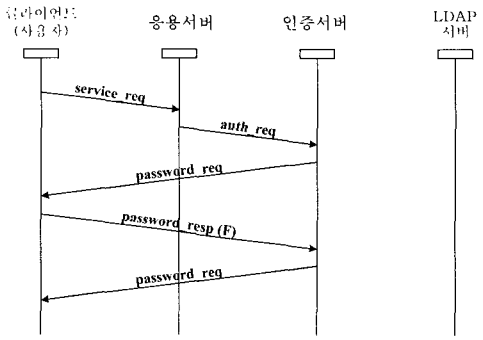

그림 3 시나리오 2: 사용자가 패스워드를 잘못 입력한 경우

2.2.3 시나리오 3

시스템 관리자는 LDAP 서버를 이용하여 모든 사용자의 접근 권한을 제어할 수 있다. CA 서버는 발급한 인증서를 LDAP 디렉토리에 공표한다. 특정 사용자의 접속을 제한하기 위해서는 관리자는 LDAP 디렉토리로부터 그 사용자의 인증서(또는 엔트리)를 제거하기만 하면 된다. 이 시나리오는 LDAP 디렉토리 룩업이 실패한 경우를 나타내며, signature_req/signature_resp 메시지는 생성되지 않는다. LDAP 서버는 더미(제로 스트림) 인증서와 함께 cert_resp 메시지를 인증 서버에게 전송하며, 인증 서버 역시 더미 ticket을 파라미터로 하는

auth_resp 메시지를 응용 서버에게 전달한다. ticket 복호화에 실패한 응용 서버는 F flag와 함께 service_resp 메시지를 클라이언트에게 전송한다. 시나리오 3에 대한 MSC가 그림 4에 표현되어 있다.

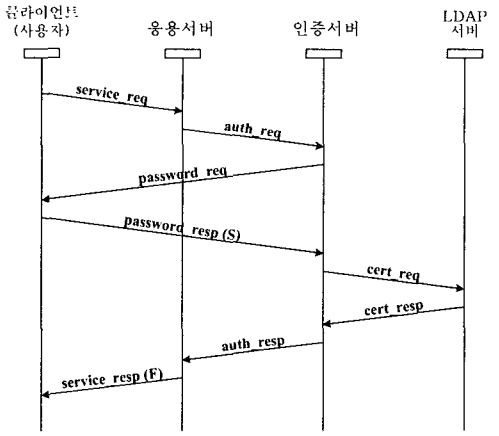

그림 4 시나리오 3: LDAP 디렉토리 룩업이 실패한 경우

2.2.4 시나리오 4

이 시나리오는 인증된 사용자가 같은 클라이언트로부터 같은 응용 서버에 재접속하는 경우와 SSO 내의 다른 응용 서버에 접속하는 경우를 가정한다. 이를 위해 시스템에서는 인증 서버 타이머(TAuth)를 설정한다. 사용자가 TAuth가 만료되기 전에 같은 응용 서버나 다른 응용 서버에 접속한다면 자신의 이메일 주소(user-Email)만 입력함으로써 부가적인 인증 프로토콜을 수행하지 않고 응용 서버를 사용할 수 있다. 이 경우 TAuth의 만료기간 설정이 중요한 설계 기준이 된다. 만료기간이 짧을 경우에는 인증된 사용자의 경우에도 계속해서 재인증을 받아야 할 문제점이 생기며, 만료기간이 길 경우에는 공격자가 인증된 사용자의 부재시 응용 서버에 접속할 수 있는 가능성을 제공한다. 따라서 적절한 만료기간의 설정과 함께 로그아웃 메커니즘이 제공되어야 한다. 시나리오 4에 대한 MSC가 그림 5에 표현되어 있다.

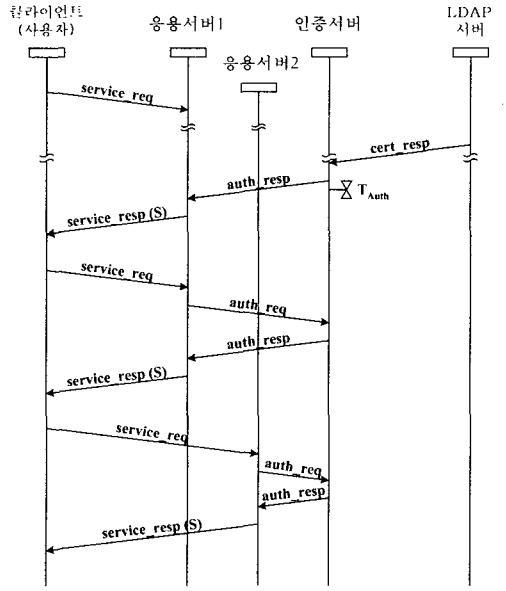

그림 5 시나리오 4: TAuth 만료기간 내에 응용 서버 1, 2에 접속한 경우

2.2.5 시나리오 5

이 시나리오에서는 사용자가 시나리오 1에 의해 인증되어 있으며 인증 서버가 auth_resp 메시지를 응용 서버에게 전송하면서 T_{Auth}를 설정했다고 가정한다. 이 경우 공격자가 인증받은 사용자의 이메일 주소(user-Email)를 이용하여 다른 클라이언트로부터 응용 서버에 접속할 가능성이 있다. 이런 종류의 공격을 막기 위해 인증 서버는 클라이언트의 네트워크 주소를 체크한다.

인증 서버는 auth_resp 메시지를 응용 서버에게 전달하면서 자신의 데이타베이스에 클라이언트의 네트워크 주소를 저장한다. 이후에 auth_req 메시지를 받을 경우, 새로 접속한 클라이언트의 네트워크 주소와 이전에 인증받은 클라이언트의 네트워크 주소가 다르다면, T_{Auth}가 만료되기 전이라 할지라도 공격자는 인증 프로토콜을 새로이 수행해야 한다. 공격자는 적법한 사용자의 password를 알지 못하므로 password_req 메시지에 응답할 수 없으며, 따라서 응용 서버를 사용할 수 없다. 시나리오 5에 대한 MSC가 그림 6에 표현되어 있다.

2.2.6 시나리오 6

이 시나리오에서는 공격자가 인증 프로토콜을 수행하지 않고 응용 서버에 접속할 수 있는 가능성에 대해 언급한다. 이런 종류의 공격을 막기 위해 randStr과 ticket의 개념이 사용된다.

응용 서버는 2가지의 uniform resource indicator (URI)를 통해서 접속될 수 있다. 하나는 service_ req 메시지의 URI 파라미터를 통한 접속이고, 다른 하나는 auth_resp 메시지를 통한 접속이다. 공격자는 후자의 URI를 통해 응용 서버에 접속하려고 하나, ticket을 생성할 수 없으므로 불법적인 접속이 거부될 수 있다.

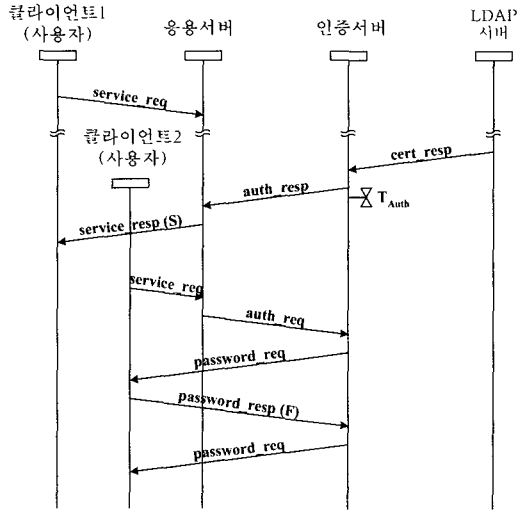

그림 6 시나리오 5: TAuth 만료기간 내에 다른 클라이언트로부터 접속한 경우

공격자가 이전의 전송에서 사용된 ticket을 도청하여 사용한다 할지라도 매번 인증 과정에서 바뀌는 랜덤 스트링(randStr)에 대해 ticket이 생성되어야 하므로 응용 서버에서 받아들여지지 않는다. 시나리오 6에 대한 MSC가 그림 7에 표현되어 있다.

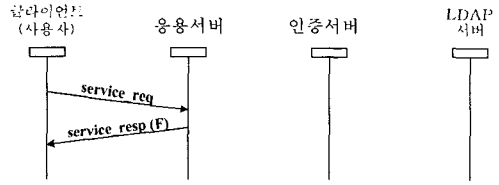

그림 7 시나리오 6: 공격자가 인증 프로토콜을 수행하지 않고 직접 접속하는 경우

2.2.7 시나리오 7

제안한 시스템은 TCP와 같은 안정적인 트랜스포트 계층 프로토콜을 사용하지만, 인증 과정의 수행시 전송된 패킷들이 유실될 수 있다. 사용자는 일정 시간내에 응답이 오지 않으면 데이타들을 재전송하는 경향이 있다. 이 경우 많은 service_req 메시지가 응용 서버에게 전송될 수 있다.

응용 서버는 인증 서버에게 *auth_req* 메시지를 보내면서 응용 서버 타이머(T_{App})를 설정함으로써 이 문제점을 해결할 수 있다. T_{App} 만료 이전에 service_req 메시지가 전송된다면 응용 서버는 그 메시지를 무시하며, T_{App}가 만료된 이후의 새로운 service_req 메시지는 처리된다. 이 경우 이전의 auth_req 메시지에 대한 auth_resp 메시지가 도착한다 할지라도 그 응답 메시지는 무시되고, 새로이 시작된 service_req 메시지에 대한 메시지 전송만이 유효화된다. 시나리오 7에 대한 MSC가 그림 8에 표현되어 있다.

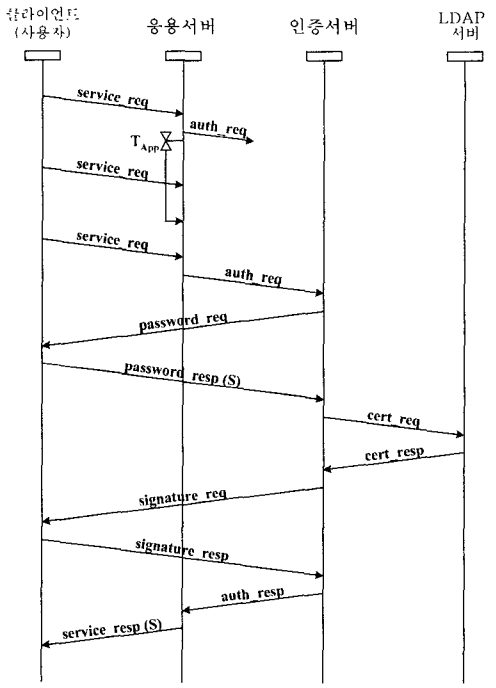

그림 8 시나리오 7: 사용자가 여러 개의 service_req 메시지를 보내는 경우

3. 시스템 구현 및 검증

이 장에서는 제안한 SSO 시스템을 구현 및 검증하는 과정에 대해 설명한다. 제안한 시스템은 PKI의 구현과 전자서명의 사용을 기반으로 하므로, 먼저 PKI를 구현한다. PKI의 주 역할은 사용자들에게 인증서를 생성, 발급하는 일이며 그림 9에 본 절에서 설계 및 구현된 PKI 시스템이 표현되어 있다. 또한 CA 서버에 의해 발급된 인증서를 저장하기 위한 LDAP 서버를 구현하기 위해 OpenLDAP[10] 패키지를 이용하였다. RSA 암호 시스템[11]을 위한 인증서를 생성하는 과정은 다음과 같으며, 상세한 데이타 포맷 및 과정을 위해 public key cryptography standards(PKCS)를 참조하였다.

Step 1. 클라이언트는 PKCS #1[12]에 따라 RSA 키 쌍을 생성한 후, PKCS #5[13]와 PKCS #8[14]에서 제시한 대로 사용자가 입력한 패스워드를 이용하여 개인키를 암호화하여 클라이언트 컴퓨터에 저장한다.

Step 2. 클라이언트는 PKCS #10[15] 인증서 요청서를 생성한 후, CA 서버에게 전송한다.

Step 3. CA 서버는 인증서 요청서를 이용하여 공개키 인증서를 생성한 후, LDAP 서버에 저장한다. 선택적으로 CA 서버는 PKCS #7[16] signedData content type을 이용하여 공개키 인증서를 사용자에게 전송할 수 있다.

제안한 SSO 시스템은 2.2절에서 제시한 설계 기준에 따라 hypertext transfer protocol(HTTP)을 기반으로 하는 웹서비스 환경으로 구현하였다. 시스템의 동작을 살펴보기 위해 상태 천이 다이어그램(State Transition Diagram: STD)을 이용하였다. MSC는 시스템의 구성

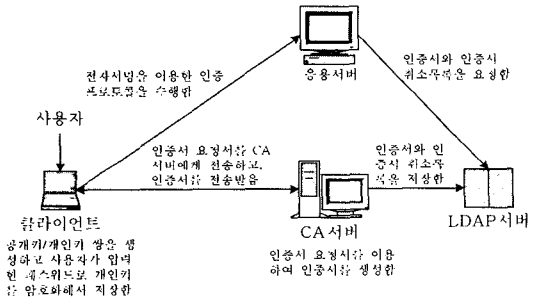

그림 9 PKI 시스템

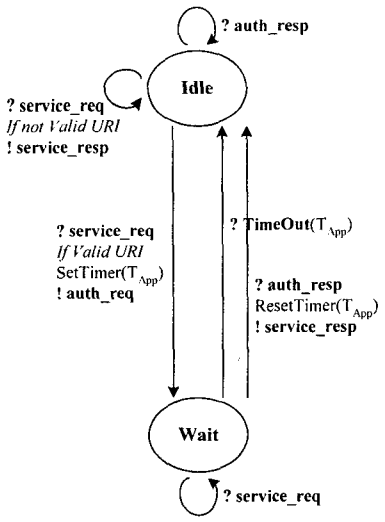

그림 10 응용 서버의 STD

요소간의 메시지 교환을 설명하기 위해 사용되는 반면, STD는 하나의 구성요소(instance)에서의 메시지 입출력에 주로 초점을 둔다. 또한 시스템의 안정적인 동작을 검증하기 위해 specification and description language (SDL)을 이용하여 시스템을 기술하고, 시스템의 동작과 MSC들을 검증하였다[17].

응용 서버는 클라이언트들에 대한 첫 번째 접속 포인트이며, 인증된 사용자에게 다양한 서비스를 제공한다. 그림 10은 응용 서버의 STD를 나타낸다. 응용 서버는 Idle과 Wait의 2가지 상태와 6개의 간선들로 구성되어 있다. 먼저 응용 서버는 Idle 상태에서 클라이언트의 service_req 메시지의 입력을 기다린다. 메시지가 도착하면 응용 서버는 응용 서버 타이머(T_{App})를 설정하고 인증 서버에게 auth_req 메시지를 보낸 후 Wait 상태로 천이한다. Wait 상태에서 인증 서버로부터 auth_resp 메시지를 받으면 ticket 파라미터의 복호화를 시도하여 원래 전송했던 randStr과 일치하면 Idle 상태로 천이하면서 클라이언트로 service_resp 메시지를 전송한다. ticket 암/복호화를 위해 DES-CBC 알고리즘을 사용했으며, 초기 벡터로 64비트 제로 스트림을 사용하였다. Wait 상태에서 Idle 상태로 천이하는 다른 경우는 T_{App}가 만료하는 경우이다. T_{App}의 만료기간은 maximum segment lifetime(MSL)과 같은 30초로 설정하였다. 클라이언트의 접속시 인증 서버에게 인증 서비스를 요청하기 위해 HTTP의 location header를 이용한 redirection 기법을 사용하였다.

인증 서버는 응용 서버들에게 인증 서비스를 제공한다. 그림 11은 인증 서버의 STD를 나타내며, 자기 자신의 상태로 천이하는 사소한 간선들은 생략되어 있다. 인증 서버의 Idle/Wait 상태는 응용 서버의 Idle/Wait 상태와 유사하며, Authenticated 상태는 사용자가 현재 인증되어 있는 상태임을 나타낸다. 먼저 인증 서버는 Idle 상태에서 응용 서버로부터의 auth_req 메시지를 기다린다. 메시지가 도착하면 인증 서버는 응용 서버 접속을 요구한 클라이언트에게 password_req 메시지를 전송한 후 Wait 상태로 천이한다. Wait 상태에서는 어느 상태로도 천이가 가능하다. Wait 상태에서 Idle 상태로의 천이는 인증 과정이 실패했음을 의미하며, Wait 상태에서 Wait 상태로의 천이는 인증 과정이 계속 진행되고 있음을 의미한다. 인증 서버는 유효한 signedData 파라미터를 갖는 signature_resp 메시지가 입력되면 로컬 데이타베이스에 인증된 사용자의 userEmail, 클라이언트의 IP 주소, 인증 프로토콜이 성공한 시각들을 기록한 후, Authenticated 상태로 천이한다. 인증 서버 타이

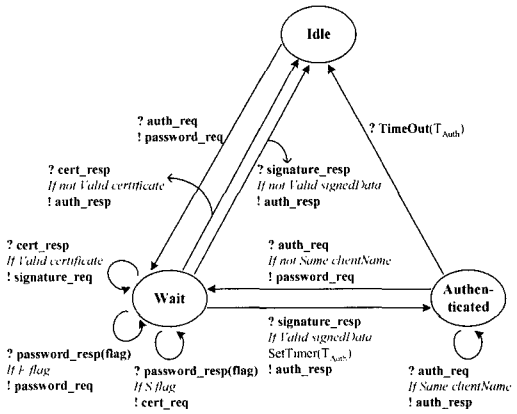

그림 11 인증 서버의 STD

머(T_{Auth})의 제어는 데이타베이스의 로그인 시각을 체크함으로써 구현된다. 제안한 시스템에서 로컬 데이타베이스의 구현을 위해 MySQL 데이타베이스 시스템을 사용하였으며, T_{Auth}의 만료기간은 1시간으로 설정하였다. 따라서 1시간 후에는 인증된 사용자라 할지라도 응용 서버의 재접속을 위해서는 인증 프로토콜의 수행을 위해 패스워드를 입력해야 한다.

또한 Authenticated 상태에 있는 인증 서버가 auth_req 메시지의 입력을 받는다면 접속을 요청한 클라이언트의 IP 주소와 로컬 데이타베이스에 저장되어 있는 클라이언트의 IP 주소를 비교한다. 두 주소가 일치한다면 인증 서버는 계속해서 Authenticated 상태에 존재하게 되며, 다를 경우에는 Wait 상태로의 천이가 발생한다.

클라이언트는 사용자와 응용/인증 서버와의 인터페이스 역할을 담당한다. 클라이언트는 개인키 데이타베이스로부터 개인키를 복호화하고, 인증 프로토콜의 수행을 위해 전자서명을 생성하는 역할을 수행한다. 개인키 암/복호화를 위해서는 PKCS #5에서 제시한 password-based encryption 알고리즘을 사용하였으며, 전자서명의 생성을 위해서는 1024 비트 RSA-MD5 전자서명 알고리즘을 사용하였다. 클라이언트 모듈은 인증 서버에 저장되어 있다가 응용 서버가 인증 서비스를 요청할 때 클라이언트로 다운로드되어 동작한다. 이의 구현을 위해 ActiveX 컨트롤 프로그래밍 기법을 사용하였으며, 클라이언트 스크립트를 위해 Visual Basic Script 언어를 사용하였다.

4. 결 론

사용자 인증은 정보보호 분야에서 점점 더 중요한 위치를 차지하고 있다. 특히 전자상거래가 활발히 이루어지고 있는 현 상황에서 적법한 사용자에 대한 인증은 매우 중요하다. 본 논문에서는 하나의 패스워드로 분산 컴퓨팅 환경하의 모든 서버들을 안전하게 사용할 수 있는 SSO 시스템이 설계, 구현되었다. 제안한 SSO 시스템은 PKI에 기반하고 있으므로 현재 구축되고 있는 전자서명 기반구조를 그대로 이용할 수 있는 장점이 있으며, 본 논문에서 구현된 프로토타입은 웹서비스를 기반으로 하는 환경을 설정하여 구현되었으므로 인터넷 포털(허브) 사이트와 같은 환경에서 바로 이용할 수 있다. 또한 제안한 시스템은 각 사용자별 권한 제어 및 정책 관리를 통합한 privilege management infrastructure (PMI)와의 통합을 통해 extranet access management (EAM)으로 확장, 발전될 수 있다.

참 고 문 헌

[1] H. P. Königs, "Cryptographic identification methods for smart cards in the process of standardization," *IEEE Communications Magazine*, vol. 29, no. 6, pp. 42 48, Jun. 1991.

[2] L. Huovinen, "Modern encryption methods in user authentication," *in Proceedings of Helsinki University of Technology*, Seminar on Network Security: Security of Corporate Networks, Chap. 4, 1997.

[3] J. Hursti, "Single sign-on," *in Proceedings of Helsinki University of Technology*, Seminar on Network Security: Security of Corporate Networks, Chap. 2, 1997.

[4] T. Tervo, "Single sign-on solutions in a mixed computing environment," *in Proceedings of Helsinki University of Technology*, Seminar on Network Security: Authorization and Access Control in Open Network Environment, Chap. 7, 1998.

[5] B. C. Neuman and T. Ts'o, "Kerberos: an authentication service for computer networks," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33 38, Sep., 1994.

[6] Netscape Communications Corporation, Single Sign-On Deployment Guide: Security, Netscape Communications Corporation, 1997.

[7] S. Chokhani, "Toward a national public key infrastructure," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 70 74, Sep. 1994.

[8] E. Rudolph, P. Graubmann, and J. Grabowski, "Tutorial on message sequence charts," *Computer Networks and ISDN Systems*, vol. 28, no. 12, pp. 1629-1641, Dec. 1996.

[9] W. Yeong, T. A. Howes, and S. Kille, Lightweight Directory Access Protocol, RFC 1777, Mar.

1995.
[10] OpenLDAP, http://www.openldap.org.
[11] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signature and public key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.
[12] RSA Laboratories, *PKCS #1: RSA Cryptography Standard*, RSA Laboratories, Sept. 1998, Version 2.0
[13] RSA Laboratories, *PKCS #5: Password-Based Cryptography Standard*, RSA Laboratories, Mar. 1999, Version 2.0
[14] RSA Laboratories, *PKCS #8: Private-Key Information Syntax Standard*, RSA Laboratories, Nov. 1993, Version 1.2
[15] RSA Laboratories, *PKCS #10: Certification Request Syntax Standard*, RSA Laboratories, May 2000, Version 1.7
[16] RSA Laboratories, *PKCS #7: Cryptographic Message Syntax Standard*, RSA Laboratories, Nov. 1993, Version 1.5
[17] D. Hogrefe, "Validation of SDL systems," *Computer Networks and ISDN Systems*, vol. 28, no. 12, pp. 1659-1667, Dec. 1996.

정 연 식

1995년 2월 연세대학교 전자공학과 학사 1997년 2월 연세대학교 대학원 전자공학과 석사. 2001년 2월 연세대학교 대학원 전기전자공학과 박사. 2000년 1월~2001년 3월 인젠 컨설팅본부 선임연구원 2001년 4월~2002년 2월 삼성전자 통신연구소 선임연구원. 2002년 3월~현재 성공회대학교 컴퓨터정보공학부 전임강사. 관심분야는 영상통신, 정보보호

김 인 겸

1990년 2월 연세대학교 전자공학과 졸업 1992년 8월 연세대학교 대학원 전자공학과 공학석사. 1997년 8월 연세대학교 대학원 전자공학과 공학박사. 1996년~1998년 2월 안양과학대학 전임강사. 1998년~현재 성결대학교 정보통신공학부 조교수. 관심분야는 영상통신, 정보보안, 영상처리