OSGi 서비스 플랫폼 환경에서의 사용자 인증 메커니즘

(An User Authentication Mechanism in the OSGi Service Platform Environment)

전 경 석 † 문 창 주 †† 박 대 하 ††† 백 두 권 ††††

(Gyoung-Seok Jeon) (Chang-Joo Moon) (Dae-Ha Park) (Doo-Kwon Baik)

요 약 홈 게이트웨이의 환경에서 사용자 인증 메커니즘은 기존의 분산 서비스 환경과 다른 몇 가지 특성들이 고려되어야 한다. 첫째로 서비스가 동적으로 설치되는 컴포넌트 환경이라는 것, 둘째로 사용자는 한번의 인증으로 여러 서비스를 사용할 수 있는 편의성이 제공되어져야 한다는 것, 마지막으로 홈 게이트웨이의 시스템 자원이 충분하지 않다는 것이 고려되어야 한다. 하지만, 현재 홈 게이트웨이의 대표적인 표준화 단체인 OSGi에서 제시한 표준안의 사용자 인증 부분에는 위와 같은 특성을 고려한 사용자 인증 메커니즘이 제시되어 있지 않다. 또한 기존의 인증 프로토콜로는 위와 같은 특성을 동시에 만족시킬 수 없다. 따라서 본 논문은 홈 게이트웨이의 위와 같은 특성을 고려한 사용자 인증 메커니즘을 제안하고자 한다. 본 논문에서는 OSGi 서비스 플랫폼을 바탕으로 각 번들 서비스의 사용자 인증 작업을 대리 수행할 수 있도록 독립된 번들 서비스를 설계하였고, 사용자의 편의성을 제공하기 위해 Kerberos 프로토콜을 확장하여 Single Sign-On의 개념이 적용될 수 있도록 하였으며, 열악한 시스템 자원을 고려하여 인증 티켓의 생성 및 메시지의 암호화에 있어 대칭키를 사용하였다. 이와 같은 사용자 인증 메커니즘은 홈 게이트웨이 환경에서 사용자의 각 번들 서비스 사용에 있어 안전성과 Single Sign-On의 적용을 통한 편의성을 제공한다. 또한 서비스 제공자의 각 서비스 제작에 있어 사용자 인증 모듈을 별도로 구현하지 않아도 되는 편의성을 제공한다.

키워드 : 인증, 승인, Kerberos 프로토콜, 인증 티켓, 역할 집합, OSGi(Open Service Gateway initiative), Single Sign-On

Abstract In the home gateway environment, several characteristics for the user authentication mechanism should be reflected separately from the existing distributed service environment. First, the platform of a home gateway is a component based system that its services are installed dynamically. Second, the convenience that user can use several services by authentication of once should be offered. Finally, the system resources of a home gateway are restricted. However, a user authentication mechanism that reflected these characteristics is not shown at the user admin service specification of the OSGi service platform.(OSGi is the representative standardization organization of home gateway.) Also, there is no existing authentication protocol that satisfies these qualities at the same time.

In this paper, we propose a new user authentication mechanism considering those characteristics for the home gateway environment. We also design and implement an independent authentication service bundle based on the OSGi service platform so that it can perform user authentication operations for each bundle service. We supplement and extend the Kerberos protocol that can apply

† 학생회원 : 고려대학교 컴퓨터학과
hoi2040@swsys2.korea.ac.kr
†† 비 회 원 : 고려대학교 컴퓨터학과
mcj@swsys2.korea.ac.kr
††† 비 회 원 : 시큐리티테크놀로지스 연구원
dhpark@stitec.com
†††† 종신회원 : 고려대학교 컴퓨터학과 교수
baikdk@korea.ac.kr
논문접수 : 2002년 8월 9일
심사완료 : 2002년 11월 19일

the concept of Single Sign-On using authentication ticket. Considering restricted system resources of home gateways, we use symmetric key cryptography for generating authentication tickets and for encrypting message. This authentication mechanism offers both safety to user in using each bundle service and convenience through realizing the concept of Single Sign-On. In addition, it offers convenience to the service providers in that they do not need to consider user authentication mechanisms in time of developing each bundle services.

Key words : Authentication, Authorization, Kerberos Protocol, Authentication Ticket, Role Set, OSGi(Open Service Gateway initiative), Single Sign-On

1. 서 론

현재 가정에 있는 다양한 정보 가전을 원격제어 등의 서비스를 목적으로 인터넷에 연결하려는 시도가 진행되고 있다. 이러한 다양한 정보 가전을 외부 네트워크와 연결하기 위해 홈 게이트웨이가 필요하다. 현재, 홈 게이트웨이에 대한 표준화 작업이 TIA TR41.5, ISO/IEC JTC21 SC25 WG1, DOCSIS 등 몇몇 기관 및 단체에서 이루어지고 있다. 그 중에서 가장 활성화되어 있는 단체는 OSGi(Open Service Gateway Initiative)이다. 현재 OSGi에서 OSGi Service Platform 릴리즈 2[1]까지 배포하였다. OSGi Service Platform 릴리즈 2는 번들 서비스를 위한 프레임워크의 표준에 대한 내용을 다루고 있으며, 프레임워크의 각 파트별로 보안에 관련된 표준안이 제시되어 있다. 사용자 인증에 대해서는 User Admin Service Specification 부분에서 패스워드, 일회용 토큰 카드, 생체 인식, 인증서 등의 여러 방식을 통해 수행할 수 있다는 기본 방향만 제시되어 있다.

홈 게이트웨이는 가정의 정보 가전이 연결되어, 가정 내부 또는 외부의 사용자들이 접속하여 다양한 서비스를 사용한다. 따라서, 보안의 관점에서 홈 게이트웨이의 안전성 여부는 기밀성, 승인, 인증 등 여러 가지 측면을 충족시켜야 한다. 그 중에서 보안의 시작이라고 할 수 있는 것이 홈 게이트웨이에 접속한 사용자가 어떤 사용자이며, 어느 정도 믿을 수 있는가 하는 사용자를 인증하는 부분이라 할 수 있다. 이는 모든 보안의 시작이며, 홈 게이트웨이에서도 마찬가지이다.

사용자 인증에 사용하는 방법으로 기존의 아이디/패스워드, PKI(Public Key Infrastructure)[2], Kerberos 프로토콜[3], EKE(Encrypted Key Exchange) 프로토콜[4] 등 여러 가지 방법이 있다. 하지만, OSGi 서비스 플랫폼 환경(이하 OSGi 환경)에서 사용자 인증은 기존의 분산 서비스 환경과 다른 몇 가지 상황들이 고려되어야 한다. 첫째, 컴포넌트 환경이 고려되어야 한다. 즉, 홈 게이트웨이에서 제공되는 각 서비스는 번들 단위의 컴포넌트로 동적으로 설치 및 제거된다. 따라서, 각 서비스 별로 인증 모듈을 갖는 것이 아니라 홈 게이트웨이 단위로 서비스 전체에 대한 공통된 사용자 인증 메커니즘이 필요하다. 둘째, 서비스 사용자에 대한 편의성이 고려되어야 한다. 사용자는 하나의 홈 게이트웨이에서 여러 서비스를 사용할 뿐만 아니라 여러 홈 게이트웨이에 있는 특정 서비스를 사용하기도 한다. 따라서 사용자는 가능한 한 하나의 인증정보(예 : 패스워드) 또는 인증서를 사용하고 한번의 인증을 통해 여러 서비스들을 사용할 수 있어야 한다. 셋째, 시스템 자원의 제약이 따른다. 홈 게이트웨이는 분산 서비스 환경의 서비스 워크스테이션처럼 빠른 중앙처리장치와 충분한 메모리, 저장매체 등을 사용할 수 없다. 따라서 공개키 암호화 알고리즘과 같은 복잡한 연산의 암호화 알고리즘을 적용하기 어려우며, 여러 사용자에 대한 인증정보를 저장하여 관리하기가 어렵다.

따라서, OSGi 환경에서 위와 같은 고려사항들을 만족시키려면 Single Sign-On의 개념을 적용한 사용자 인증 메커니즘이 필요하다. 본 논문은 Kerberos 프로토콜을 확장하여, 위와 같은 고려사항들을 만족하는 새로운 사용자 인증 메커니즘을 제시하고 실제 적용 가능한 사용자 인증 서비스를 설계 및 구현함으로써, OSGi 환경에 맞는 사용자 인증 메커니즘을 제안하고자 한다[5].

본 논문의 구성은 다음과 같다. 2장은 OSGi 서비스 플랫폼과 사용자 인증 프로토콜에 대해 기술하고, 3장에서는 OSGi 환경에서 제기되는 문제들을 해결한 사용자 인증 메커니즘을 제안한다. 4장은 제안된 사용자 인증 시스템의 소프트웨어 구성요소의 설계에 대한 내용을 설명하고 5장은 구현결과 및 평가에 대해 기술한다. 마지막으로 6장은 본 연구의 결론 및 향후 연구과제에 대해 서술한다.

2. 관련 연구

2.1 OSGi 서비스 플랫폼

OSGi는 홈 게이트웨이 표준화 단체 중의 하나로 업계표준을 정하는 단체이다. 현재 OSGi에는 썬 마이크로

시스템, 소니, 삼성전자 등 세계 유수의 기업이 참여하고 있다. 2001년 10월 OSGi Service Platform 릴리즈 2[1]가 발표되었으며, 홈 게이트웨이에서 각 번들 서비스에 대한 프레임워크의 표준안에 대해 다루고 있다. OSGi 서비스 플랫폼의 전체적인 구조는 그림 1과 같다.

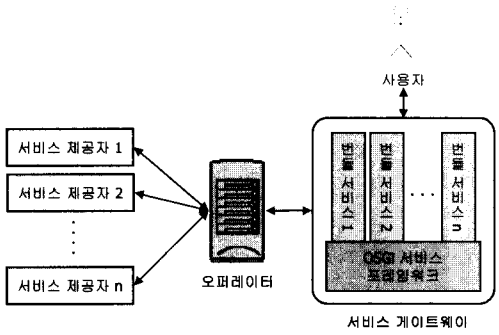

그림 1 OSGi 서비스 플랫폼의 구조

OSGi 서비스 플랫폼은 번들 서비스를 제작하여 배포하는 서비스 제공자(service provider)와 각 게이트웨이에 배포된 번들 서비스를 관리하는 오퍼레이터(operator), 그리고 각 가정에 설치되어 있는 홈 게이트웨이인 서비스 게이트웨이(service gateway), 마지막으로 서비스 게이트웨이를 사용하는 사용자(user)로 구성된다.

OSGi 서비스 플랫폼은 번들 단위의 컴포넌트 환경으로 번들의 기본 구조 및 OSGi 서비스 프레임워크와의 관계는 그림 2와 같다.

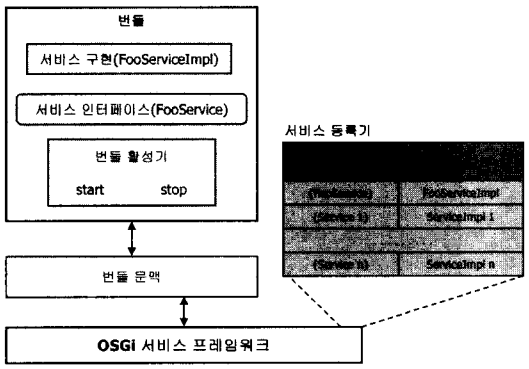

그림 2 번들의 구조 및 OSGi 서비스 프레임워크와의 관계

각 번들 서비스는 OSGi 서비스 프레임워크의 서비스 등록기(service registry)에 등록되어 다른 서비스가 등록된 서비스의 서비스 인터페이스(service interface)를 통해 사용할 수 있다. 각 번들은 번들의 생명 주기를 관리하는 번들 활성기(bundle activator)와 다른 서비스로부터의 참조를 위한 서비스 인터페이스(service interface) 그리고 인터페이스를 구현한 서비스 구현(service implementation)으로 구성된다. 마지막으로 번들 문맥(bundle context)은 각 번들과 OSGi 서비스 프레임워크 간의 인터페이스 역할을 담당한다. 즉 번들 문맥은 각 번들을 OSGi 서비스 프레임워크에 등록하고 다른 서비스를 찾아서 사용할 수 있도록 한다.

OSGi Service Platform 릴리즈 2의 User Admin Service Specification 부분에서 제시하고 있는 사용자 인증 과정은 그림 3과 같다.

사용자가 서비스 게이트웨이의 서비스를 사용하기 위해서는 먼저 각 번들 서비스에 인증 요청을 하고(①) 각 번들 서비스는 사용자가 제시한 인증정보를 바탕으로 인증서버를 통해(②) 사용자에 대한 인증을 수행한다. 이 때 사용자 인증에 사용되는 정보는 아이디/패스워드, 일회용 토큰 카드, 생체 인식, 인증서 등의 여러 방식을 사용할 수 있다. 하지만, 사용자가 서비스 게이트웨이의 다른 서비스를 사용하고자 할 때, ① ②의 과정을 반복적으로 수행해야 한다. 따라서, 사용자가 하나의 홈 게이트웨이에서 여러 서비스를 사용하거나 또는 여러 홈 게이트웨이에 있는 특정 서비스를 사용하고자 할 때 매번 인증 작업을 수행해야 한다.

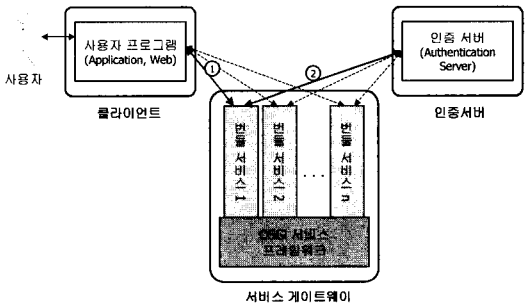

그림 3 기존 OSGi 서비스 플랫폼에서 사용자 인증 과정

2.2 사용자 인증 프로토콜

네트워크에 연결되어 있는 기존의 분산서비스 환경에서, 서버로부터 제공되는 서비스에 접속하기 위한 다양한 사용자 인증 프로토콜이 존재한다. 이러한 인증 프로토콜들은 메시지 암호화의 관점에서 대칭키 또는 비대칭키의 사용 유무에 따라 그리고 당사자(principal)간의

안전한 통신을 위해 미리 정의된 일을 수행하는 TTP(trusted third party)의 존재 유무에 따라 다음과 같은 네 가지로 분류된다[6].

① Symmetric Key Without Trusted Third Party
② Symmetric Key with Trusted Third Party
③ Public Key
④ Hybrid Protocols

첫 번째 인증 프로토콜은 당사자간에 안전한 통신을 위한 별도의 TTP를 사용하지 않고, 암호화된 메시지를 생성하기 위해 대칭키를 사용한다. 이 부류에 속하는 프로토콜의 예로 ISO의 One-pass Symmetric Key Unilateral Authentication Protocol[7]이 있다.

두 번째 인증 프로토콜은 당사자간에 안전한 통신을 위한 별도의 TTP를 사용하고, 암호화된 메시지를 생성하기 위해 대칭키를 사용한다. TTP는 당사자간에 암호화된 통신을 위해 세션키와 이 세션키의 전달을 위해 티켓을 생성 배포한다. 즉 당사자 A는 당사자 B와의 통신을 위해 TTP로부터 세션키와 이 세션키가 들어있고 당사자 B만 복호화 할 수 있는 티켓을 발부 받는다. 그런 후 당사자 B에게 티켓을 전달 해 줌으로서 세션키를 공유하게 되고 이를 사용하여 안전한 통신을 할 수 있게 된다. 여기서 이 세션키를 재사용 할 수 있도록 설계된 프로토콜을 반복 인증 프로토콜(repeated authentication protocol)이라 하며, 대표적인 프로토콜로는 Neuman과 Stubblebine의 인증 프로토콜[8]과 MIT에서 개발한 Kerberos 프로토콜[3]이 있다. 그림 4는 Kerberos 프로토콜의 간략한 구조이다.

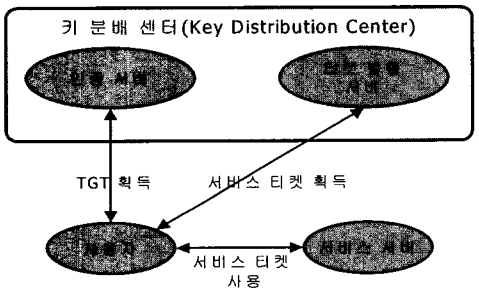

그림 4 Kerberos 프로토콜의 구조

Kerberos 프로토콜은 인증 서버(authentication server), 티켓 발행 서버(ticket granting server)로 분리되어, 사용자 인증을 한 후, TGT(ticket granting ticket)를 받고, 이것을 사용하여 티켓 발생 서버에서 서비스 티켓을 받는다. 그리고 난 후 사용자는 이 서비스 티켓을 사용하여 원하는 서비스를 제공하는 서버(service server)에 접속하여 서비스를 사용할 수 있는 권한을 얻는다. Kerberos 프로토콜은 사용자 인증을 위한 티켓의 생성에 있어 대칭키를 사용할 수 있고, 다수의 서비스를 사용하기 위해 한번의 인증 과정을 통해 받은 TGT를 일정 기간(티켓의 생명주기)동안 재사용할 수 있다. 또한, 사전에 인증 서버와 티켓 발행 서버, 티켓 발행 서버와 서비스 서버 사이에 공유 대칭키를 가지고 있어야 한다

세 번째 인증 프로토콜은 메시지의 암호화를 위해 공개키 기반 구조(PKI)를 사용한다. 즉 메시지를 암호화하는데 있어 공개키를 사용하고, 인증서를 기반으로 사용자 인증을 수행한다. 대표적인 프로토콜로는 CCITT의 X.509에서 정의한 인증 프로토콜[9]이 있다.

네 번째 인증 프로토콜은 메시지의 암호화를 위해 공개키 및 대칭키를 동시에 사용한다. 일반적이지는 않지만, 메시지는 공개키를 사용하여 암호화하고 공개키를 배포하는데 대칭키를 사용한다. 대표적인 프로토콜로는 EKE(Encrypted Key Exchange) 프로토콜[4]이 있다.

3. OSGi 환경에서 사용자 인증 메커니즘

본 논문에서 제안하는 사용자 인증 메커니즘은 OSGi의 컴포넌트 환경으로 고려하여 각 서비스가 세부적인 로그인 과정을 수행하지 않도록 로그인 과정을 전담하는 로그인 서비스를 두었고, 기존의 인증 프로토콜 중 인증 티켓을 이용하여 Single Sign-On의 개념을 적용할 수 있는 Kerberos 프로토콜[10]을 OSGi 환경에 맞게 적용하였다. 또한 시스템 자원이 적은 서비스 게이트웨이 환경을 고려하여 인증 티켓의 생성 및 메시지의 암호화에 있어 대칭키를 사용하였다. 마지막으로 인증 티켓에 사용자의 역할 집합을 포함시켜 인증 작업 및 향후 승인 작업의 수행에 있어 역할(role) 기반 모델[11]이 적용될 수 있도록 하였다.

3.1 사용자 인증 메커니즘의 구조

본 논문에서 제안하는 사용자 인증 메커니즘의 전체적인 구조는 그림 5와 같다.

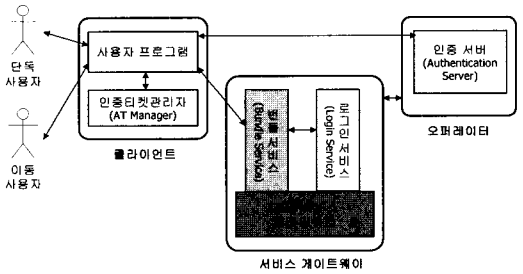

그림 5 사용자 인증 메커니즘의 구조

사용자 인증 메커니즘에서 시스템 개체(entity)는 오퍼레이터, 서비스 게이트웨이, 클라이언트의 세 가지가 있다.

- 오퍼레이터 : OSGi 환경에서 오퍼레이터의 수는 하나 또는 극히 적은 수가 된다. 또한 오퍼레이터가 서비스 게이트웨이들을 관리하는 역할을 수행하기 때문에 인증서버가 오퍼레이터에 존재한다.
- 서비스 게이트웨이 : 서비스 게이트웨이는 사용자에 대한 서비스를 제공하는 각 번들 서비스들을 오퍼레이터로부터 다운로드하여 설치한다. 또한 서비스 게이트웨이에는 각 번들 서비스의 사용자 인증을 위한 로그인 서비스(login service)가 존재한다.
- 클라이언트 : 클라이언트는 서비스 게이트웨이 및 오퍼레이터와 연결되어 있는 일종의 단말기로서, 사용자와의 인터페이스를 제공하며, 사용자의 인증정보가 들어있는 인증 티켓(authentication ticket)을 관리한다.

그림 5에서 사용자는 여러 서비스 게이트웨이의 사용 여부에 따라 단독 사용자(standalone user)와 이동 사용자(travel user)로 구분되어 진다. 단독 사용자는 하나의 서비스 게이트웨이에 접속하여 여러 서비스를 사용하는 사람이고, 이동 사용자는 여러 서비스 게이트웨이에 접속하여 특정 서비스를 사용하는 사람 또는 시스템이다. 예를 들면, 단독 사용자는 자신의 가정에 있는 서비스 게이트웨이에 접근하여 전등을 켜고 끄는 서비스를 사용한다. 하지만 이동 사용자는 각 가정의 서비스 게이트웨이마다 접근하여 전기 사용량을 점검하는 서비스를 사용한다.

3.2 소프트웨어 구성요소의 역할

그림 5에서 소프트웨어적인 구성요소는 오퍼레이터의 인증 서버, 서비스 게이트웨이의 로그인 서비스와 사용자에게 특정 서비스를 제공하는 번들 서비스, 그리고 클라이언트의 사용자 프로그램이 있다. 각 구성요소의 역할은 아래와 같다.

- 인증 서버 : 그림 5의 인증 서버는 Kerberos 프로토콜의 인증 서버와 같은 역할을 한다. 즉 사용자의 인증 요청에 인증 작업을 수행하며 그에 대한 결과로 티켓을 발행한다. Kerberos 프로토콜에서는 이 티켓을 TGT(ticket granting ticket)라고 하지만 여기서는 인증 티켓이라고 한다. 인증 티켓은 TGT와 다르게 사용자에 대한 역할정보를 가지고 있다. 이 역할은 사용자의 권한 범위를 나타내는 것으로서 서비스 게이트웨이에서 서비스 및 시스템 자원의 사용 가능 여부를 결정하는데 사용한다[11].
- 로그인 서비스 : 로그인 서비스는 서비스 게이트웨이의 부트스트래핑(bootstrapping) 과정에서 오퍼레이터로부터 다운로드하여 설치된다. 로그인 서비스는 Kerberos 프로토콜의 TGS와 유사하다 할 수 있다. 하지만 TGS는 사용자의 서비스 요청에 대해 서비스 티켓(service ticket)을 사용자에게 발행하지만 여기서 로그인 서비스는 그와 다른 작업을 수행한다. 로그인 서비스는 시스템 전체의 서비스에 대한 정책 파일(policy file)에 따라 작업을 수행한다. 이 정책 파일은 각 서비스를 사용할 수 있는 사용자의 역할에 대한 정보를 가지고 있다. 따라서 사용자가 제시한 인증 티켓에 대해 복호화를 수행하여 그 안에 있는 역할이 사용자가 요청한 서비스에 대한 역할과 일치하는지 검사한 후 해당 서비스를 사용할 수 있는지 여부를 결정한다. 그리고 로그인 서비스는 인증 티켓을 복호화 할 수 있는 특정 대칭키를 가지고 있어야 한다.
- 번들 서비스 : 번들 서비스는 사용자에게 특정 서비스를 제공한다. 이러한 번들 서비스는 서비스 제공자별로 각기 다르게 제작 배포된다. 그리고 오퍼레이터에 의해 번들 단위로 관리되며, 사용자의 요청에 따라 서비스 게이트웨이에 다운로드되어 설치 또는 제거된다. 각 번들 서비스는 위에서 서술한 로그인 서비스에 사용자로부터 얻은 인증 티켓을 넘겨주고 그에 대한 결과에 따라 사용자의 인증 여부를 결정하게 된다.
- 사용자 프로그램 : 사용자 프로그램은 사용자와 인증 서버, 각 번들 서비스간의 인터페이스를 제공하며, 사용자에 대한 인증정보를 가지고 있는 인증 티켓을 관리한다. 사용자는 처음 번들 서비스를 사용하고자 할 때 오퍼레이터의 인증 서버로부터 인증을 받는다. 그 결과로 인증 티켓을 받아서 저장 관리한다. 그런 다음 이 인증 티켓을 사용하고자 하는 번들 서비스에 전달하여 로그인 서비스를 통해 서비스 사용 가능 여부를 결정 받게 된다. 또 다른 번들 서비스를 사용하고 할 때는 인증 서버로부터 다시 인증을 받지 않고 가지고 있는 인증 티켓을 사용한다.

3.3 인증 프로토콜

사용자 인증 메커니즘의 인증 프로토콜에 대한 전체적인 과정은 그림 6과 같다.

- 표기법

U	사용자 명
AS	인증 서버 명
BS	번들 서비스 명
SG	서비스 게이트웨이 명
Ka_b	a 와 b 사이의 공유 대칭키

{M}Ka_b	*Ka_b*를 사용하여 암호화된 메시지
Ts#	타임스탬프
Tlt	최초 인증 시간(인증 티켓의 생명주기를 계산하는데 사용)
Tauth	인증 티켓
Auth	인증 확인 정보
Subject	서비스 사용 승인 정보(사용자의 아이디와 역할 정보)
RS	인증된 사용자의 역할 집합

• 각 단계별 프로토콜

① AS_REQ : *U, SG, Ts1*

② AS_REP : *{Ku_ls, SG, Ts1}Ku_as, Tauth*

③ BS_REQ : *U, BS, Ts2, Tauth, {Auth}Ku_ls*

④ LS_REP : *{U, Ts3}Ku_ls, Subject*

⑤ BS_REP : *{U, Ts3}Ku_ls*

• 인증 정보

Tauth : *SG, {Ku_ls, U, RS, Tlt}Kls_as*

Auth : *U, BS, Ts2*

• 프로토콜 시나리오 구성도

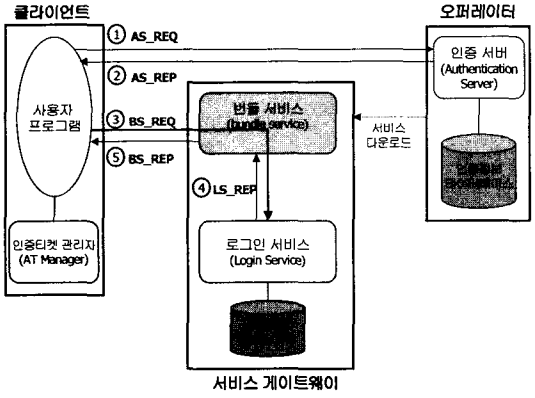

그림 6 사용자 인증 프로토콜

그림 6의 사용자 인증 프로토콜은 5개의 단계로 이루어져 있으며, 각 단계별 프로토콜의 상세 내용은 다음과 같다.

① 사용자 인증 요청(AS_REQ) : 사용자가 특정 번들 서비스를 사용하고자 할 때 먼저 오퍼레이터의 인증 서버로부터 인증을 받아야 한다. 사용자는 개인의 아이디, 패스워드를 가지고 있고, 인증 서버에 사용자의 인증 요청 시 사용자의 아이디, 사용하고자 하는 서비스 게이트웨이의 아이디(본 논문의 구현에서는 서비스 게이트웨이의 IP 주소를 사용), 그리고 요청을 보내는 시간을 보낸다. 여기서 사용자의 패스워드는 보내지 않으며, 이미 인증 서버의 데이타베이스에 등록되어 저장된 것으로 간주한다.

② 사용자에 대한 인증 결과 전송(AS_REP) : 인증 서버는 데이타베이스에 저장되어 있는 사용자의 인증정보를 사용해서 결과 메시지를 작성한 후 사용자에게 전송한다. 이때, 인증 티켓과 사용자와 인증 서버간의 공유 대칭키(*Ku_as*)를 사용하여 암호화한 인증내용을 보낸다. *Ku_as*는 사용자의 패스워드를 사용한다. 따라서, 인증내용은 패스워드를 알고있는 사용자만이 복호화할 수 있다. 사용자는 복호화된 내용과 처음 사용자가 보낸 내용을 비교 확인함으로써 인증이 성공적으로 수행되었는지를 알 수 있다. 인증 티켓은 로그인 서비스와 인증 서버간의 공유 대칭키(*Kls_as*)를 사용해서 암호화했기 때문에 사용자는 복호화할 수 없다. 인증 서버가 인증 티켓을 암호화할 때는 인증 요청 시 입력된 아이디에 따라 단독 사용자용과 이동 사용자용 대칭키 중 선택하여 암호화한다. 단독 사용자용 대칭키는 로그인 서비스별로 다르며, 이동 사용자용 대칭키는 로그인 서비스별로 같다.

③ 번들 서비스에 대한 승인 요청(BS_REQ) : 사용자는 인증 서버로부터 인증 받은 후 사용하고자 하는 특정 번들 서비스에 인증 티켓을 전송한다. 인증 티켓은 자체가 암호화되어 있기 때문에 네트워크상에서 안전하다는 것을 보장한다. 또한 인증 티켓을 보내는 사용자가 인증 서버로부터 인증 받은 사용자라는 것을 보장하기 위해, 인증 서버가 생성하여 사용자에게 보낸 사용자와 로그인 서비스간의 공유 대칭키(*Ku_ls*)를 사용하여 암호화된 인증 확인 정보(*Auth*)를 보낸다. 이는 사용자와 로그인 서비스간의 공유 대칭키(*Ku_ls*)를 인증 받은 사용자만이 알 수 있기 때문에 인증 받지 않은 사용자의 인증 티켓 사용을 방지한다.

해당 번들 서비스는 요청 받은 인증 티켓을 로그인 서비스에 전달한다. 이 때 인증 티켓은 로그인 서비스와 인증 서버간의 대칭키(*Kls_as*)로 암호화되어 있기 때문에 번들 서비스에 의해 변질되지 않는다는 것을 보장한다. 또한, 인증 확인 정보(*Auth*)도 사용자와 로그인 서비스간의 공유 대칭키(*Ku_ls*)에 의해 암호화되어 있기 때문에 번들 서비스에 의해 변질되지 않는다는 것을 보장한다.

④ 번들 서비스에 사용자의 승인 결과 전송(LS_REP) : 먼저, 로그인 서비스는 요청된 인증 정보 중에서 번들 서비스의 종류에 따라 인증 티켓을 복호화 할 수 있는 로그

인 서비스와 인증 서버간의 공유 대칭키(Kls_as)를 단독 사용자용과 이동 사용자용 중에서 하나를 선택하게 된다. 그런 다음 선택된 Kls_as를 사용하여 인증 티켓을 복호화 한다.

인증 티켓 안에 있는 사용자와 로그인 서비스간의 공유 대칭키(Ku_ls)로 사용자가 보낸 인증 확인 정보 ($\{Auth\}Ku\_ls$)를 복호화 하여 인증 티켓의 정당한 사용자인지를 확인한다. 사용자의 확인 작업이 성공적으로 끝나면, 인증 티켓 안에 있는 번들 서비스의 사용을 요청한 사용자의 역할 집합을 검사한다. 인증 티켓에 있는 역할 집합이 시스템 정책 파일(policy file)에 있는 해당 번들 서비스의 역할 집합과 같은지를 검사한 후 번들 서비스의 사용가능 여부를 결정한다.

이런 로그인 과정이 성공적으로 수행되면, 로그인 서비스는 승인 결과를 Ku_ls를 사용하여 암호하고 ($\{U, TS\}Ku\_ls$), 인증 티켓에 있는 역할집합을 포함하는 JAAS API 의 Subject[12] 객체를 같이 전송한다. 여기서 JAAS API의 Subject 객체는 번들 서비스에서 시스템 자원의 사용에 대한 승인(Authorization)을 수행하기 위한 것이다.

⑤ 사용자에 요청한 번들 서비스의 사용 승인 결과 전송 (BS_REP) : 번들 서비스는 로그인 서비스로부터 전달받은 내용 중 Subject 객체를 제외한 나머지 서비스 사용 승인 결과 부분($\{U, TS\}Ku\_ls$)을 사용자에게 전송한다. 사용자는 전송 받은 내용을 인증 서버로부터 받은 인증 정보(AS_REP)에 있는 사용자와 로그인 서비스간의 공유 대칭키(Ku_ls)를 사용하여 복호화 함으로서 요청한 번들 서비스의 사용 승인 여부를 알 수 있다.

이와 같은 사용자 인증 프로토콜을 통해 사용자는 다수의 번들 서비스에 대해 매번 인증을 받는 것이 아니라, 한번 인증을 통해 받은 인증 티켓을 사용함으로써 Single Sign-On의 개념을 만족할 수 있다. 물론 인증 티켓에는 인증시간이 있어 특정 기간(생명주기)동안만 사용할 수 있다. 이것은 보안상 안전을 위해 인증 티켓의 신선도(freshness)[13]를 제공하기 위함이다.

3.4 공유 대칭키 공유 방법

그림 6의 사용자 인증 프로토콜에서 필요로 하는 공유 대칭키는 표 1과 같이 세 가지가 있다.

각 키의 공유관계에 따른 공유 방법은 아래와 같다. Ku_as는 사용자의 패스워드로 사용자가 처음 가입할 때 인증 서버에 등록이 된다. 또한 사용자의 인증 시 Ku_as가 네트워크 상에서 이동하는 것이 아니기 때문에 별도의 공유 방법이 필요하지 않다.

표 1 공유 대칭키 간의 비교

키 종류 / 항 목	Ku_as	Ku_ls	Kls_as
공유관계	사용자와 인증 서버	사용자와 로그인 서비스	로그인 서비스와 인증서버
목 적	Ku_ls의암호화	사용자의 인증 확인 정보 암호화	인증 티켓의 암호화
생 성	사용자	인증 서버	오퍼레이터의 키 생성기
유효기간	사용자가 변경할 때까지	인증 티켓의 생명주기 까지	인증 서버가 변경 할 때까지

Ku_ls는 사용자의 인증이 성공적으로 이루어 졌을 때, 인증정보에 Ku_as에 의해 암호화되어 사용자에게 전달된다. 또한 인증 티켓에 의해 보호되어 로그인 서비스에 전달된다.

Kls_as는 단독 사용자용과 이동 사용자용 인증 티켓에 사용하는 두 가지가 있다. 먼저 인증 서버가 있는 오퍼레이터는 키 생성기(key generator)에 의해 단독 사용자용 인증 티켓에 사용하는 Kls_as를 서비스 게이트웨이별로 다르게 생성하고, 이동 사용자용 인증 티켓에 사용하는 Kls_as를 생성하여 데이타베이스에 저장한다. 로그인 서비스는 서비스 게이트웨이가 부트스트래핑 과정에서 오퍼레이터에 의해 서비스 게이트웨이로 다운 설치되어 실행된다. 이 때 오퍼레이터가 로그인 서비스 번들에 데이타베이스에 저장되어 있는 Kls_as를 번들 속성 파일(bundle property file)에 포함시켜 서비스 게이트웨이에 다운되도록 한다.

여기서 로그인 서비스 번들이 네트워크를 통해 서비스 게이트웨이에 다운 설치될 때 Kls_as가 저장되어 있는 번들 속성 파일이 보안상 안전하지 않을 수 있다. 하지만 여기서는 OSGi 환경에서 사용자 인증 메커니즘과 별도로 서비스 게이트웨이의 부트스트래핑 과정에서 다수의 작업을 수행하는데, 이 때 특정 보안 방법을 통해 서비스 게이트웨이와 오퍼레이터간의 안전한 채널이 생성되었다는 것을 가정으로 한다[14].

4. 사용자 인증 메커니즘의 소프트웨어 구성요소 설계 및 구현

사용자 인증 메커니즘의 핵심적인 소프트웨어 구성요소는 3.2에서 기술한 바와 같이 인증 서버, 로그인 서비스, 번들 서비스, 사용자 프로그램이 있다.

4.1 인증 서버

그림 7은 오퍼레이터에 있는 인증 서버의 구조를 보여주는 클래스 다이어그램(class diagram)이다.15].

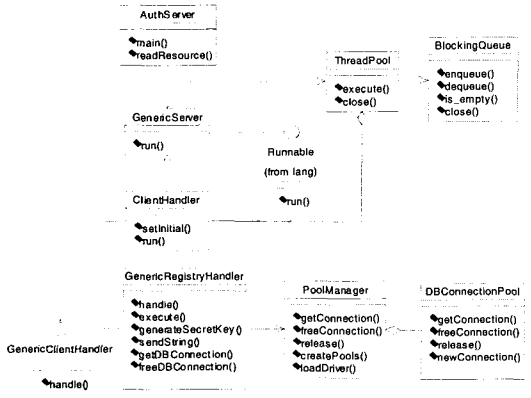

그림 7 인증 서버의 클래스 다이어그램

사용자의 인증 요청에 대한 결과 처리는 GenericRegistryHandler의 execute()메소드에서 수행한다. 그림 8은 사용자 인증 작업을 수행하는 execute() 메소드의 개략적인 코드이다. 사용자와 로그인 서비스간의 공유 대칭키 (*Ku_ls*)를 생성하고, 인증 티켓(*Tauth*)과 사용자 인증 결과 정보를 암호화한다. 이러한 과정이 성공적으로 끝나면 인증 결과 메시지를 전송한다.

```
protected void execute() {

 Read user's request[user id, service gateway id,
 timestamp];
 try{
   Retrieve the type of user[single user or travel user]
   from user table;
   if(the type is single user){
     Ku_as := Retrieve the user's password from user
     table;
     Ku_ls := Generate a new secret key[between user
     and login service];

     /* generate AT */
     Kls_as := Retrieve the Kls_as[between login service
               and authentication server] from gateway
               table;
     Tlt := Generate timestamp;
     RS := Retrieve the user's Role Set from user_role
           table;
     Encrypt AT with Kls_as;
       /* AT = service gateway id || (Ku_ls || user id
                || RS || Tlt) */

     Encrypt ARD(authentication result data) with
     Ku_as;
```

```
       /* ARD = Ku_ls || service gateway id || time-
              stamp of user's request */

     AS_REP := ARD || AT;
     Send AS_REP;

    }else(the type is travel user){
     . . .
   }
  }catch(Exception e){
   Send Error message;
  }
}
```

그림 8 인증 서버에서 사용자 인증 코드

4.2 로그인 서비스와 번들 서비스

그림 9는 서비스 게이트웨이에 있는 번들 서비스인 Light Service와 로그인 서비스의 개략적인 클래스 다이어그램이다.

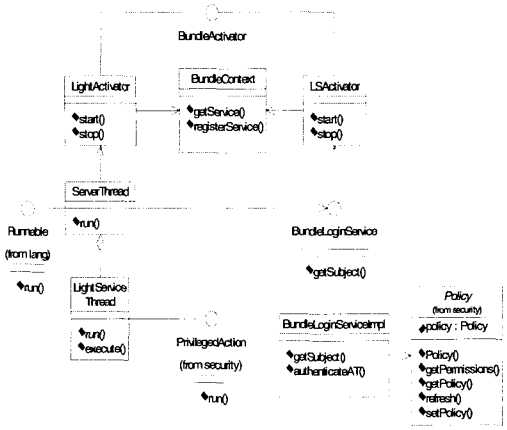

그림 9 번들 서비스와 로그인 서비스의 클래스 다이어그램

BundleContext는 OSGi 서비스 플랫폼의 표준 API 클래스이다. 이것은 각 번들 활성기(bundle activator)에서 서비스를 등록하고 등록된 서비스의 레퍼런스를 찾아서 사용하는데 쓰인다.

그림 9에서 로그인 서비스[16]는 LSActivator, BundleLoginServiceImpl, Policy 클래스와 BundleLogin Service 인터페이스로 구성된다. LSActivator는 로그인 서비스를 번들 문맥을 사용하여 OSGi 서비스 프레임워크에 등록한다. 이렇게 등록된 로그인 서비스는 다른 번들 서비스에 의해 참조 사용되어 진다.

사용자에 대한 로그인 작업은 BundleLoginService

Impl 클래스의 getSubject() 메소드에서 수행된다. 그림 10은 getSubject() 메소드의 개략적인 코드이다.

```
public Subject getSubject(Socket sock){
  Read user's request[AT, authentication confirmation
  data, ...];
  try{
    UAD := Retrieve UAD[user's authorization data]
    from policy file;
    Decrypt AT;
    Verify AT; /* verify AT's lifetime */
    if(AT is valid){
      Decrypt user's authentication verification
      data(Auth in the protocol);
      if(user has right AT){ /* by user's authentication
      verification data[user id, bundle service id,
      timestamp] */
        RS := Get user's role set from AT;
        /* compare RS with UAD */
        if(RS imply roles that are required to use the
        requested service){
          Subject := Generate a new Subject[user id,
          RS];
          return Subject;
        }else{
          Send role_error message;
        }
      }else{
        Send user_invalid_error message;
      }
    }else{
      Send AT_invalid_error message;
    }
  }catch(Exception e){
    Send except_error message;
  }
}
```

그림 10 로그인 서비스에서 사용자의 서비스 사용에 대한 승인 코드

getSubject() 메소드는 먼저 사용자의 서비스 사용에 대한 승인정보를 정책 파일[12][17]로부터 읽어 들인다. 그런 다음 인증 티켓을 복호화 한 후 그 안에 있는 인증 티켓의 생명주기를 확인하여 인증 티켓의 유효성 여부를 검사한다. 인증 티켓이 유효하면 인증 티켓 안에 있는 Ku_ls를 사용하여 사용자가 암호화하여 보낸 인증 확인 정보(Auth)를 복호화 하여 사용자가 정당하게 인증 받았는지 여부를 확인한다. 사용자에 대한 확인이 끝나면 인증 티켓 안에 있는 사용자의 역할 집합을 정책 파일에 있는 내용과 비교하여 사용자의 서비스 요청에 대한 승인작업을 수행한다. 이러한 모든 과정이 정상적으로 수행되면 마지막으로 Subject 객체를 생성하여 반환한다.

getSubject() 메소드에서 사용하는 정책 파일의 내용은 그림 11과 같다. 정책 파일은 어떠한 역할이 각 서비스에 대해 어떤 권한을 가지고 있는지를 나타내는 파일이다.

```
//JAAS 역할기반 정책
grant Codebase "http://software.korea.ac.kr", Signedby
"operator",
      Principal ac.korea.Role "normal_user"{
        Permission org.osgi.framework.ServicePermission
                   "LightService", "use";
      Permission org.osgi.framework.ServicePermission
                   "BundleLoginService", "get";
}
```

그림 11 정책 파일

그림 9에서 번들 서비스는 서비스 제공자로부터 제공 가능한 여러 종류의 번들 서비스 중 한 예로 LightActivator, ServerThread, LightServiceThread 클래스로 구성되며, 전기를 점등하거나, 소등하는 서비스를 제공한다. LightActivator는 서비스 쓰레드인 ServerThread 클래스를 시작시키거나 멈추게 하는 클래스이다. 또한 LightActivator는 BundleContext를 통해 등록된 로그인 서비스의 레퍼런스를 찾는 작업을 수행한다. ServerThread는 각 사용자의 접속을 받아들이며, 로그인 서비스의 getSubject() 메소드 호출을 통해 사용자의 로그인 작업을 수행한다. 로그인 작업이 성공적으로 수행되면 각 사용자에게 서비스를 제공하는 LightServiceThread 클래스를 시작시킨다.

```
/* find the ls[Login Service] from ctxt[BundleContext] */
LoginService ls := ctxt.getService(reference of login service);

public void run(){
  while(this.running){
    try{
      /* accept a user's request */
      Socket sock := this.sv_sock.accept();
      if(!this.running)
        break;
      /* request authorization using bundle service to ls[
         Login Service] */
      Subject u_subject := ls.getSubject(sock);
      LightServiceThread lst := Create a LightService
      Thread;
      Subject.doAsPrivileged(u_subject, lst, null);
    }catch(IOException e){
      Send error message;
    }
  }
}
```

그림 12 번들 서비스에서 로그인 서비스의 사용 코드

그림 12는 번들 서비스에서 로그인 서비스의 getSubject() 메소스를 호출하는 ServerThread 클래스의 코드 일부분이다. 그림 12에서와 같이 서비스 제공자는 번들 서비스 제작 시 getSubject()의 호출을 통해 사용자의 서비스 사용 승인 요청에 관한 부분을 특별한 어려움 없이 쉽게 작성할 수 있다.

4.3 사용자 프로그램

그림 13은 사용자 프로그램의 개략적인 클래스 다이어그램이다.

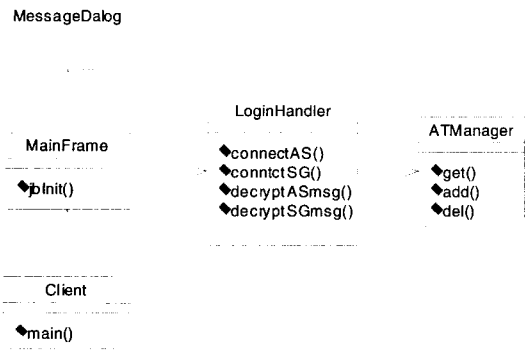

그림 13 사용자 프로그램의 클래스 다이어그램

사용자 프로그램은 사용자의 서비스 게이트웨이에 대한 인터페이스이며 인증 서버로부터 받은 인증 티켓을 관리한다. Client 클래스가 메인 프로그램이고, MainFrame 클래스는 사용자와의 인터페이스 역할을 하는 클래스이다. ATManager 클래스는 인증 서버로부터 받은 인증 티켓을 관리하기 위한 클래스이고 Login Handler 클래스가 인증 서버와 서비스 게이트웨이에 접속하여 사용자의 서비스에 대한 로그인 과정을 수행한다.

5. 사용자 인증 메커니즘의 구현 결과 및 평가

5.1 구현 환경

본 논문에서 제시하는 사용자 인증 메커니즘의 구현 및 평가를 위해 그림 14와 같은 실험 환경을 구성하였다. 인증 서버, 홈 게이트웨이, 사용자 프로그램 모두 운영체제는 Window2000, 개발언어는 JDK1.3을 사용하였으며, 인증 서버의 인증 정보(패스워드, 역할 집합 등)를 저장하는 데이타베이스는 Oracle 8i를 사용하였다. 홈 게이트웨이의 경우 OSGi 서비스 플랫폼을 구현한 Java Embedded Server 2[18]를 사용하였으며, 홈 게이트웨이 장비는 현재 제품화되어 있는 에릭슨의 E-Box[19]를 바탕으로 유사한 환경(200MHz CPU, 64MB RAM, 10/100M Ethernet Adapter)을 구축하여 실험하였다. 그리고, 인증된 결과 및 승인 정보에 대한 내용은 JAAS1.0 API[12]를 사용하였다. 인증 티켓 및 인증정보를 암호화하는데는 한국 정보보호 진흥원(KISA)에서 개발한 SEED[20] 128 비트 대칭키 알고리즘을 사용하였으며, 자바 기반의 암호화 라이브러리는 (주) 시큐리티 테크놀로지스의 J/LOCK API[21]를 사용하였다.

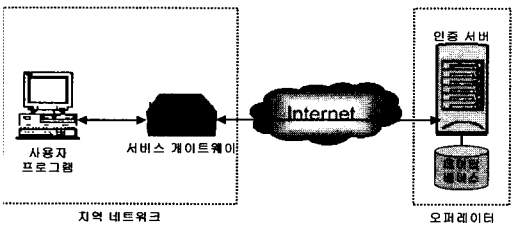

그림 14 실험 환경 구성도

5.2 구현 결과

사용자 인증 메커니즘의 구현은 "4.소프트웨어 구성요소의 설계"에 의해 인증 서버, 로그인 서비스, 번들 서비스, 사용자 프로그램을 구현하였다. 번들 서비스의 경우 Single Sign-On의 테스트를 위해 Light Service, Heater Service, FuseBox Service를 구현하였다. Light Service는 집안의 전등을 켜고 끄는 서비스이며, Heater Service는 집안의 전열기를 켜고 끄는 서비스이고, FuseBox Service는 집안의 전기사용료를 측정하는 이동 사용자용 서비스이다.

그림 15는 사용자가 Light Service를 사용하기 위한 사용자 프로그램 화면이다.

사용자가 Light Service를 사용하기 위해 처음 로그인을 하면, 인증 티켓 관리자에 인증 티켓이 없기 때문

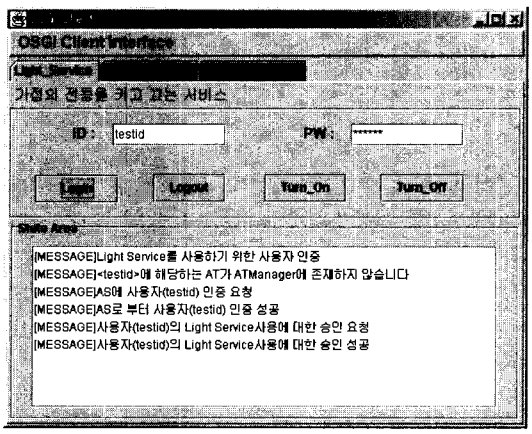

그림 15 Light Service를 사용하기 위한 사용자 인증

에 인증서버로부터 사용자 인증을 받은 후 인증 티켓을 인증 티켓 관리자에 저장한다. 그런 다음 인증 티켓을 사용하여 Light Service에 서비스 사용 가능 여부를 요청하여 인증 받는 작업을 수행한다. 그림 15의 상태 영역(State Area)에 진행 상황이 출력되는 것을 볼 수 있다.

그림 16은 Light Service를 사용한 후 Heater Service를 사용하기 위한 사용자 프로그램 화면이다.

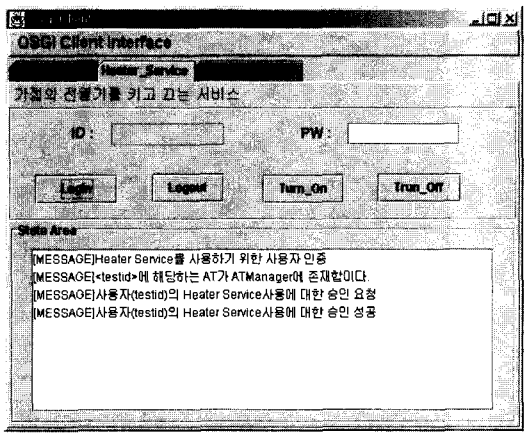

그림 16 Heater Service를 사용하기 위한 사용자 인증

그림 16은 그림 15와 다르게 인증 서버에 대한 사용자 인증과정을 거치지 않는다. Light Service를 사용하기 위해 로그인하는 과정에서 인증 서버로부터 인증 티켓을 획득 인증 티켓 관리자에 저장되어 있기 때문에, Heater Service를 사용하기 위해 로그인하는 과정에서는 저장된 인증 티켓을 사용한다. 따라서 인증 서버로부터 사용자에 대한 인증을 받는 과정을 거치지 않게 되므로 Single Sign-On의 기능을 제공하게 된다. 그림 16에서 아이디와 패스워드 입력필드가 비 활성화되어 있는 것을 볼 수 있다. 만약 인증 티켓 관리자에 인증 티켓이 없다면, 아이디와 패스워드 입력 필드가 활성화되어 사용자로부터 입력을 받는다.

그림 17은 이동 사용자의 인증 과정을 보여주는 사용자 프로그램이다.

FuseBox Service는 이동 사용자용 서비스로, 여러 서비스 게이트웨이에서 실행된다. 따라서 이동 사용자는 사용하고자 하는 서비스 게이트웨이의 아이디(본 논문의 구현에서는 서비스 게이트웨이의 IP 주소를 사용)를 입력한다. 그림 17의 첫 번째 화면은 처음 로그인하는 화면으로 인증 티켓이 없기 때문에 인증 서버로부터 인증을 받아 인증 티켓을 얻는다. 두 번째 화면은 다른 서비스 게이트웨이에 있는 FuseBox Service에 로그인하는 화면으로 첫 번째 로그인하는 과정에서 얻은 인증 티켓을 사용함으로 인증 서버로부터 별도의 사용자 인증 과정을 거치지 않는다.

사용자 프로그램에서 사용자가 로그아웃하면, 인증 티켓 관리자에 저장되어 있는 인증 티켓을 제거함으로써 다른 사용자에 의한 인증 티켓 사용을 방지한다.

5.3 평가

본 논문의 사용자 인증 프로토콜은 기존의 Kerberos 프로토콜을 OSGi의 환경에 맞게 보완한 것으로, 본 논문의 사용자 인증 프로토콜과 Kerberos 프로토콜간의 차이점은 표 2와 같다.

본 논문의 사용자 인증 프로토콜은 Kerberos 프로토콜처럼 각 서비스의 사용을 위해 서비스 티켓을 발행하지 않고, 인증 티켓 안에 사용자의 역할 집합을 포함하

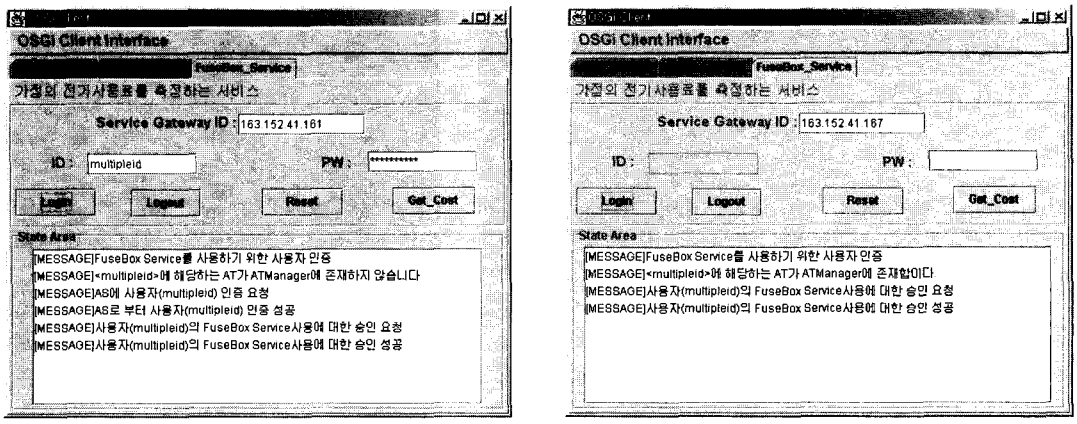

그림 17 FuseBox Service를 사용하기 위한 사용자 인증

표 2 본 논문의 사용자 인증 프로토콜과 Kerberos 프로토콜의 비교

비교대상 / 비교항목	본 논문의 사용자 인증 프로토콜	Kerberos 프로토콜
사용자 인증 정보	인증 서버의 인증 티켓	인증 서버의 TGT
사용자의 서비스 사용 승인 방법	로그인 서비스에서 인증 티켓 안에 있는 역할집합 해석	TGS에서 TGT해석을 통한 서비스 티켓 발행
사용자의 서비스 사용 승인 정보	서비스 게이트웨이의 정책 파일	TGS의 사용자 승인 정보
영역간의 지원	지원하지 않는다.	지원한다.

여 이 역할집합의 해석을 통해 서비스의 사용 승인 작업을 수행한다. 또한 사용자의 서비스 사용에 대한 승인 정보는 Kerberos 프로토콜에서 티켓발행 서버가 관리하는 것과 다르게, 각 서비스 게이트웨이 별로 오퍼레이터가 제작하여 배포하는 정책 파일로 관리한다. 이와 같은 방법을 통해 본 논문에서 로그인 서비스의 설계가 가능하도록 하였다. 그리고 본 논문에서는 각 홈 게이트웨이를 관리하는 오퍼레이터가 하나라고 가정한다. 따라서 다른 영역간(inter-realm)의 지원(예 : 다른 오퍼레이터의 인증 서버가 발행한 인증 티켓의 지원)은 고려되지 않았다.

그림 18은 본 논문에서 제안하는 사용자 인증 프로토콜(A)과 Kerberos 프로토콜(B)에 대한 실험결과 그래프이다.

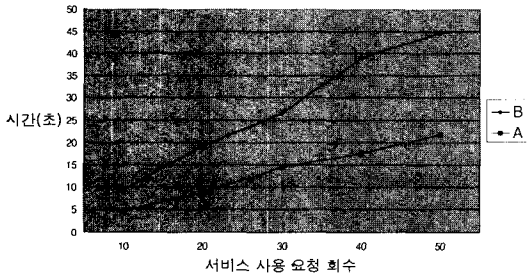

그림 18 인증 프로토콜의 성능실험 결과

위 실험은 인증 서버, 서비스 게이트웨이, 사용자 프로그램을 각각 별도의 컴퓨터에 네트워크로 연결하였고, Kerberos 프로토콜의 경우 인증 서버와 티켓발행 서버가 오퍼레이터에 있는 것을 전제로 실험하였다. 가로축은 각 서비스의 사용 요청 회수를 10회 단위로 시간을 측정한 수치이고, 세로축은 이 때 걸린 시간을 초단위로 측정한 수치이다.

Kerberos 프로토콜의 경우 사용자가 서비스 사용승인을 요청할 때마다 TGT를 사용하여 서비스 티켓을 획득하기 위해 매번 티켓발행 서버가 있는 오퍼레이터에 연결하게 된다. 반면에 본 논문의 인증 프로토콜은 오퍼레이터에 매번 연결을 수행하지 않고 인증 티켓 안에 있는 역할집합의 해석을 통해 사용자의 서비스 승인 요청에 대한 작업을 수행한다. 따라서 그림 18의 실험 결과 그래프에서 보는 바와 같이 사용자의 서비스 사용 승인에 걸리는 시간이 단축된다.

본 논문의 사용자 인증 프로토콜은 Kerberos 프로토콜을 기반으로 하여 인증 프로토콜에서 생길 수 있는 여러 위험요소들을 제거하였다[22]. 먼저 타임스탬프와 암호화된 인증 확인 정보(*Auth*)를 사용하여 인증 티켓의 재활용 공격(replay attack)과 같은 중재자 공격(main-in-the-middle attack)으로부터 안전 할 수 있다. 그리고 인증 티켓 및 메시지 암호화에 있어 SEED 대칭키 알고리즘을 사용함으로써 암호화 메시지의 차분해독법(differential cryptanalysis) 및 선형해독법(linear cryptanalysis) 그리고 기타 연관 키 공격(related key attack) 등으로부터 안전할 수 있다. 하지만 인증 티켓의 생성에 있어 사용자의 패스워드를 사용하기 때문에 패스워드 추측 공격(password guessing attack)과 로그인 속이기 공격(login spoofing attack)에 의한 위험은 존재한다.

본 논문에서 제안하는 인증 메커니즘의 프로토콜에서 인증 티켓 및 메시지 암호에 있어 대칭키를 사용한다. 이는 현재 홈 게이트웨이의 환경이 분산서비스 환경과 다르게 시스템 자원이 열악하기 때문이다. 차후 충분한 시스템 자원이 갖추어진 홈 게이트웨이 환경이 형성된다면, 본 논문에서 제안한 사용자 인증 프로토콜에서 PKI[2]를 기반으로 하는 사용자 인증 프로토콜로 변형이 가능하다.

6. 결론 및 향후 연구과제

본 논문은 OSGi 서비스 플랫폼을 기반으로 홈 게이트웨이의 특성을 고려한 사용자 인증 메커니즘을 제안하였다. 홈 게이트웨이의 컴포넌트 환경을 고려하여 각 서비스가 세부적인 로그인 과정을 수행하지 않도록 로그인 과정을 전담하는 로그인 서비스를 두었고, 역할 집합과 정책 파일을 사용하여 Kerberos 프로토콜을 OSGi 환경에 맞게 보완 확장함으로써 사용자 인증에 있어 Single Sign-On의 개념이 적용될 수 있도록 하였다. 또한 인증 티켓 및 메시지의 암호화에 있어 복잡한 연산을 수행하는 공개키 대신 대칭키를 사용함으로써 사용자 인증작업의 수행에 있어 좀더 빠르고 적은 시스템

자원을 사용하도록 하였다.

본 논문의 사용자 인증 메커니즘은 각 번들 서비스의 로그인 과정을 전담하는 로그인 서비스를 둠으로써 서비스 제공자의 각 번들 서비스 제작에 있어 편의성을 제공한다. 또한 인증 티켓의 역할집합과 정책 파일의 사용으로 Kerberos 프로토콜의 서비스 티켓 발행의 필요성을 제거함으로써 좀 더 빠른 인증 과정의 수행과 정책 파일을 통한 오퍼레이터의 보안 관리 편의성을 제공한다. 마지막으로 사용자는 인증 티켓을 사용하여 한번의 인증을 통해 여러 번들 서비스를 이용할 수 있는 편의성을 제공한다.

하지만 본 연구의 사용자 인증 메커니즘은 다른 영역간의 지원을 고려하지 않았다. 따라서 향후 이에 대한 연구와 홈 게이트웨이에서 서비스 사용자가 각 서비스를 사용하는데 있어, 사용자의 등급별 시스템 자원 및 특정 서비스의 사용에 대한 RBAC(Role-Based Access Control) 모델[11] 기반의 승인 메커니즘에 대한 연구가 필요하다.

참 고 문 헌

[1] OSGi Service Platform Release 2 Specification, http://www.osgi.org/resources/docs/spr2book.pdf, October 2001.

[2] Marc Branchaud, "A Survey of Public Key Infrastructures," Department of Computer Science, McGill University, Montreal, 1997.

[3] B.Clifford Neuman, Theodore Ts'o, "Kerberos : An Authentication Service for Computer Network," IEEE, Computer Magazine 32:9:33-38, September 1994.

[4] S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password based protocols secure against dictionary attacks," In Proceedings 1992 IEEE Symposium on Research in Security and Privacy, 72-84. IEEE Computer Society, May 1992.

[5] 전경석, 문창주, 박대하, 백두권 "OSGi Service Framework 환경에서 사용자 인증 방법," 정보과학회지, 제29권, 제1호, page 865-867, 2002.

[6] John Clark and Jeremy Jacob, "A Survey of Authentication Protocol Literature: Version 1.0," University of York, Department of Computer Science, November 1997.

[7] ISO/IEC. Information Technology - Security techniques — Entity Authentication Mechanisms Part 2: Entity authentication using symmetric techniques, 1993.

[8] B. Clifford Neuman and Stuart G. Stubblebine, "A Note on the Use of Timestamps as Nonces. Operating Systems Review," 27(2):10-4, April 1993.

[9] CCITT Recommendation X.509. The Directory-Authentication Framework. CCITT, December 1988.

[10] Mayank Upadhyay, Ram Marti, "Single Sign-On Using Kerberos in Java," http://java.sun.com/j2se/1.4/docs/guide/security/jgss/single-signon.html.

[11] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, Charles E. Youman, "Role-Based Access Control Models," IEEE Computer, Volume 29, Number 2, February 1996, page 38-47.

[12] Charlie Lai, Li Gong, "User Authentication and Authorization in the Java Platform," Computer Security Applications Conference, December 1999.

[13] Li Gong, "Variations on the Themes of Message Freshness and Replay or the Difficulty in Devising Formal Methods to Analyze Cryptographic Protocols," the Computer Security Foundations Workshop VI, 1993.

[14] 김영갑, 문창주, 박대하, 백두권 "OSGi 서비스 프레임워크 환경에서의 서비스 번들 인증 메커니즘," 정보과학회지, 제29권, 제1호, page 868-870, 2002.

[15] Fowler, M., UML distilled: Applying the standard object modeling language, Addison Wesley, May, 1997.

[16] V. Samar and C. Lai. "Making Login Services Independent from Authentication Technologies," In Proceedings of the SunSoft Developer's Conference, March 1996.

[17] Default Policy Implementation and Policy File Syntax, http://java.sun.com/products/jdk/1.2/docs/guide/security/PolicyFiles.html.

[18] Java Embedded Server 2.0, http://wwws.sun.com/software/embeddedserver/index.html.

[19] Ericsson's e-box system-An electronic services enabler. http://www.ericsson.com/about/publications/review/1999_01/files/1999015.pdf.

[20] 128비트 블록 암호화알고리즘(SEED) 개발 및 분석 보고서, http://www.kisa.or.kr/technology/sub1/report.pdf.

[21] Java Cryptography Library, J/LOCK, http://www.stitec.com/product/ejlock.html.

[22] Steven M. Bellovin, Michael Merritt, "Limitations of the Kerberos Authentication System," Proceedings of the Winter 1991 USENIX Conference, January 1991.

전 경 석

2001 고려대학교 컴퓨터학과 학사. 2001 ~현재 고려대학교 컴퓨터학과 석사과정 관심분야는 보안 프로토콜, 임베디드 시스템 보안, 메타데이타

문 창 주

1997년 고려대학교 컴퓨터학과(학사) 1999년 고려대학교 컴퓨터학과(석사) 1999년~현재 고려대학교 컴퓨터학과 박사과정. 관심분야는 보안 공학, 컴포넌트 기반, 소프트웨어 공학

박 대 하

1992년 고려대학교 컴퓨터학과 학사 1994년 고려대학교 일반대학원 컴퓨터학과 석사. 1996년 고려대학교 일반대학원 컴퓨터학과 박사과정 수료. 1999~현재 (주)시큐리티테크놀로지스 책임연구원. 관심분야는 XML 보안, 보안 프로토콜, 이동코드 보안, 임베디드 시스템 보안

백 두 권

1974년 고려대학교 수학과 학사. 1976년 고려대학교 대학원 산업공학과 석사 1983년 Wayne State Univ. 전산학 석사. 1986년 Wayne State Univ. 전산학 박사. 1986년~현재 고려대학교 컴퓨터학과 교수. 1989년~현재 한국정보과학회 이사/평의원. 1991년~현재 ISO/IEC JTC1/SC32 국내위원회 위원장. 1992년~현재 한국시뮬레이션학회 이사/부회장/회장. 2002년~현재 고려대학교 정보통신대학 학장. 2002년~현재 과학기술정보표준위원회 위원장. 관심분야는 정보보호, 메타데이타, 소프트웨어공학, 시뮬레이션