

OSGi 서비스 플랫폼 환경을 위한 보안 아키텍처 (Security Architecture for OSGi Service Platform Environment)

박 대 하 * 김 영 갑 ** 문 창 주 *** 백 두 권 ****

(Dae-Ha Park) (Young-Gab Kim) (Chang-Ju Moon) (Doo-Kwon Baik)

요약 본 논문에서는 안전한 OSGi 서비스 플랫폼 환경의 구축에 필요한 새로운 보안 아키텍처를 제시한다. 이 보안 아키텍처에는 1) 사용자 인증 메커니즘, 2) 번들 인증 메커니즘, 3) 키 공유 메커니즘, 4) 권한부여 메커니즘이 포함된다. 사용자 인증 메커니즘에서는 안전하고 편리한 사용자 인증을 위해 SSO(single sign-on) 기능을 제공한다. 번들 인증 메커니즘에서는 효율적인 서비스 번들의 인증을 위해 PKI 기반의 전자서명과 대칭키 기반의 MAC을 함께 이용한다. 서비스 게이트웨이의 부트스트래핑 단계에서 수행되는 키 공유 메커니즘은 인증 메커니즘에서 사용되는 비밀키를 안전하게 공유할 수 있는 방법을 제공한다. 마지막으로 권한부여 메커니즘을 통해 서비스 번들의 제작자와 게이트웨이 오퍼레이터가 분산된 보안 정책을 수립할 수 있는 방안을 제시한다. 본 논문은 기존의 OSGi 스펙에서 추상적으로 명시한 보안 기능을 실제로 OSGi 환경에서 적용할 때 필요한 요구사항을 살펴보고 구체적인 보안 아키텍처의 설계와 구현을 통해 해결 방안을 기술한 점에서 중요한 의미를 갖는다.

키워드 : OSGi 서비스 플랫폼, 사용자 인증, 번들 인증, 키 공유, 권한부여, 보안 정책

Abstract This paper suggests a new security architecture for facilitating secure OSGi service platform environment. The security architecture includes 1) user authentication mechanism, 2) bundle authentication mechanism, 3) key sharing mechanism, and 4) authorization mechanism. The user authentication mechanism supplies SSO(single sign-on) functions which are useful for safe and easy user authentications. The bundle authentication mechanism utilizes both PKI-based and MAC-based digital signatures for efficiently authenticating service bundles. The key sharing mechanism, which is performed during bootstrapping phase of a service gateway, supplies a safe way for sharing secret keys that are required for authentication mechanisms. Finally, the authorization mechanism suggests distributed authorization among service providers and an operator by establishing their own security policies. The main contributions of the paper are twofold. First, we examine several security requirements of current OSGi specification when its abstract security functions can be applied in real OSGi environments. Second, we describe the ways to resolve the problems by means of designing and implementing concrete security mechanisms.

Key words : OSGi service platform, User authentication, Bundle authentication, Key sharing, Authorization, and Security policy

1. 서론

OSGi(Open Services Gateway Initiative)는 다양한 서비스를 외부망(WAN)을 거쳐 내부망(LAN)으로 전달

하기 위한 개방형 스펙을 개발하고, 이 스펙을 기반으로 다양한 서비스를 정보 가전(home appliance) 또는 텔레매틱스(telematics)와 같은 내장형 시스템(embedded system) 시장에 확산시키기 위한 목적으로 조직된 그룹이다. OSGi 서비스 플랫폼 스펙(이하 OSGi 스펙)[1]에서는 서비스 프레임워크(service framework; 이하 프레임워크)를 명시하여 이에 따르는 개방형 서비스 게이트웨이(service gateway; 이하 게이트웨이)와 게이트웨이에서 수행되는 서비스를 개발하는데 필요한 참조 아키텍처 및 표준 서비스 인터페이스를 정의하고 있다.

그림 1은 OSGi 서비스 플랫폼이 동작하는 전체적인

* 비 회 원 : 한국디지털대학교 디지털정보학과 교수

summer69@kdu.edu

** 학생회원 : 고려대학교 컴퓨터학과

ygkim@software.korea.ac.kr

*** 비 회 원 : 고려대학교 컴퓨터학과

mcj@software.korea.ac.kr

**** 종신회원 : 고려대학교 컴퓨터학과 교수

baik@software.korea.ac.kr

논문접수 : 2003년 10월 9일

심사완료 : 2004년 1월 9일

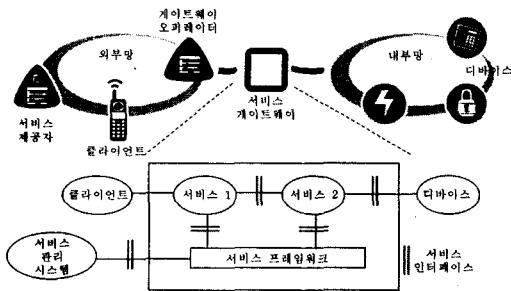


그림 1 OSGi 서비스 플랫폼 환경

환경(이하 OSGi 환경)을 나타낸다. OSGi 환경에서 개방된 네트워크를 따라 게이트웨이에 접근하는 사용자와 게이트웨이에 설치되는 서비스 번들(service bundle)에 대한 인증(authentication) 및 권한부여(authorization)와 같은 기능을 제공하는 보안 아키텍처의 구축이 필수적이다.

OSGi 환경에 적합한 보안 아키텍처를 구축하려면 일반적인 개방형 분산 컴포넌트 서비스 환경(예: CORBA, EJB, DCOM 등)과 달리 다음과 같은 제약조건을 고려해야 한다.

- 이동성(mobility) - OSGi 환경은 서비스를 제공하는 컴포넌트(즉, 번들)를 동적으로 게이트웨이에 설치하여 사용한다. 따라서 보안 아키텍처에서는 컴포넌트의 이동성에 따른 문제(예: 인증, 무결성)를 고려해야 한다.
- 효율성(efficiency) - OSGi 환경은 제한된 시스템 자원을 갖는 내장형 게이트웨이(예: 셋톱박스)에서 작동하는 프레임워크를 제공한다. 따라서 일반적인 워크스테이션처럼 빠른 CPU와 충분한 메모리 또는 저장매체 등을 사용하기 힘든 환경에서도 효율적으로 작동할 수 있는 보안 아키텍처의 구성이 필요하다.
- 다양성(diversity) - OSGi 환경은 정보 가전과 같은 매우 다양한 벤더(vendor)의 장비에 접속하는 서비스를 연결해준다. 보안 아키텍처에서는 서비스에 대한 자세한 구현이나 서비스 제공자를 인식하지 못한 상황에서 다른 컴포넌트와 안전하게 연동할 수 있는 최대한의 가용성(availability)을 제공할 수 있어야 한다.
- 유용성(usability) - OSGi 환경은 사용자와 서비스 제공자가 보안 기능에 대한 충분한 지식을 가지고 있음을 전제하기 어렵다. 따라서 오퍼레이터와 게이트웨이에서는 사용자와 서비스 제공자가 편리하게 이용할 수 있는 보안 메커니즘을 제공해야 한다.

현재 OSGi 스펙에서는 사용자 인증을 위하여 게이트웨이에 설치되는 번들마다 별도의 인증 메커니즘을 구현해야 하고 사용자는 각각의 서비스에 대한 인증을 수

행해야 하므로 유용성 제약사항을 만족시키지 못하는 단점이 있다. 번들의 권한부여를 위하여 오퍼레이터에 의한 중앙 집중형 보안 정책을 강조하고 있어서 온디맨드(on-demand)로 네트워크를 통해 이동하여 설치되는 번들의 구현에 대한 세부적인 지식이 없이는 권한부여가 어려우므로 다양성 제약사항을 만족시키기 어렵다. OSGi 보안 전문가 그룹(SEG)에서는 사용자와 번들의 인증에 공개키 기반(PKI) 기술의 사용을 권고하고 있지만 게이트웨이 자원의 제약으로 인해 효율성 제약사항을 만족하지 못한다. 특히 OSGi 스펙은 추상적인 서비스 인터페이스의 표준화에 중점을 두고 있어서 보안 기능에 대한 구체적인 구현 메커니즘은 명시하지 않고 있으며, 최근 Gamespace 또는 ProSyst와 같은 OSGi 플랫폼의 구현에서도 고가의 전용선(예: PSTN)이나 가상 사설망(VPN)의 연동으로 보안 기능을 대체하고 있는 실정이다.

본 논문의 연구에서는 OSGi 환경에 대한 제약조건을 고려할 때 등장하는 새로운 보안 요구사항을 만족하도록 보안 아키텍처를 제안하고 개별적인 보안 메커니즘을 설계 및 구현하였다. 이 보안 아키텍처에는 사용자 인증 메커니즘(user authentication mechanism), 번들 인증 메커니즘(bundle authentication mechanism), 키 공유 메커니즘(key sharing mechanism), 권한부여 메커니즘(authorization mechanism)이 포함된다.

본 논문은 기존의 OSGi 스펙에서 추상적으로 명세한 보안 기능을 실제로 OSGi 환경에서 적용할 때 필요한 요구사항을 살펴보고 구체적인 보안 아키텍처의 설계와 구현을 통해 해결 방안을 기술한 점에서 중요한 의미를 갖는다.

본 논문의 구성은 다음과 같다. 2장에서는 OSGi 환경의 보안에 필요한 요구사항과 관련 연구를 살펴보고 제안하는 보안 아키텍처의 정당성을 부여한다. 3장에서는 보안 아키텍처를 구성하는 각각의 보안 메커니즘에 대한 정형화된 명세를 제공한다. 4장에서는 보안 아키텍처에 따라 게이트웨이와 오퍼레이터에 설치될 보안 시스템을 설계하고 구현한 결과를 기술한다. 5장에서는 연구 결과를 정리하고 추후 연구방향을 제시한다.

2. OSGi 보안 요구사항 및 관련연구

이 장에서는 OSGi 환경의 보안에 필요한 요구사항과 기존의 관련된 연구를 살펴봄으로써 본 논문에서 제안하는 OSGi 보안 아키텍처의 정당성을 부여하고자 한다.

2.1 사용자의 인증

OSGi 환경에서 외부망으로부터 게이트웨이에 접근해서 서비스를 이용하려는 사용자에게 대한 인증은 다음과 같은 요구사항을 만족해야 한다.

- <요구사항 1-1> 사용자는 하나의 게이트웨이에서 여러 서비스를 사용할 뿐만 아니라 여러 게이트웨이에 있는 특정한 서비스를 사용하기도 한다. 따라서 하나의 인증 정보(예: 패스워드)를 사용하고 한 번의 인증을 통해 여러 서비스를 사용할 수 있는 SSO 기능이 필요하다.
- <요구사항 1-2> 동적으로 게이트웨이에 설치되는 컴포넌트의 이동성과 서비스 제공자에 대한 유용성을 고려해야 한다. 따라서 각각의 번들 별로 인증 메커니즘을 갖는 것이 아니라 게이트웨이 단위로 서비스 전체에 대한 공통된 사용자 인증 메커니즘이 필요하다.
- <요구사항 1-3> 게이트웨이의 시스템 자원이 갖는 제약에 따른 효율성을 고려해야 한다. 따라서 공개키 암호화와 같은 복잡한 연산보다는 대칭키 암호화를 사용하며, 가능한 게이트웨이의 메모리에 저장하고 관리하는 사용자 인증 정보의 크기를 감소시키는 방법이 필요하다.

개방형 분산 서비스 환경에서 외부 개체가 서비스에 접속하기 위한 다양한 사용자 인증 프로토콜이 존재한다[2]. 본 논문의 보안 아키텍처에서는 사용자 인증 메커니즘으로 Kerberos 프로토콜[3]을 OSGi 환경에 맞게 변형하여 적용한다. 이 프로토콜은 인증 티켓의 재사용을 통한 반복 인증이 가능하므로 <요구사항 1-1>을 만족하는 SSO 개념을 구현하기에 적합하다. 또한 게이트웨이에 로그인 서비스를 두어 번들의 각 서비스가 세부적인 로그인 과정을 수행하지 않도록 하여 <요구사항 1-2>에 따른 컴포넌트의 이동성을 고려한다. 인증 티켓의 생성과 메시지의 기밀성 제공에 대칭키 암호화를 사용하므로 공개키 암호화를 사용할 경우보다 시스템 자원의 제약에 따른 영향이 감소하여 <요구사항 1-3>을 만족시킨다.

2.2 서비스 번들의 인증

OSGi 환경에서 게이트웨이에 설치되는 서비스 번들에 대한 인증은 다음과 같은 요구사항을 만족해야 한다.

- <요구사항 2-1> 특정한 오퍼레이터에 종속적인 메커니즘으로 번들의 인증 정보를 생성하려면 서비스 제공자의 비용이 증가한다. 따라서 서비스 제공자는 기존에 널리 사용되는 인증 메커니즘을 이용하여 번들의 인증 정보를 생성할 필요가 있다.
- <요구사항 2-2> 오퍼레이터의 시스템은 게이트웨이와 달리 충분한 시스템 자원을 가진 대형 워크스테이션으로 구성되므로 PKI와 같은 안정적인 인증 메커니즘을 이용하여 번들의 인증 정보를 검증하는 것이 바람직하다.
- <요구사항 2-3> 오퍼레이터의 인증을 거친 번들은 대칭키를 사용한 비교적 간단한 연산으로 인증 정보

를 생성하고 게이트웨이에서 효율적으로 검증을 수행할 수 있어야 한다.

현재 OSGi 스펙에서는 번들에 대한 인증을 위해 서명된 JAR(signed JAR)[4]와 RSH 프로토콜[5]을 권고하고 있다. 본 논문의 보안 아키텍처에서 서비스 제공자와 오퍼레이터 간의 인증은 서명된 JAR를 이용하여 <요구사항 2-1>에 따른 인증 정보 생성에 대한 서비스 제공자의 독립성을 제공한다. 오퍼레이터는 <요구사항 2-2>에 따른 안정적인 기술의 지원이 가능하도록 인증기관(CA)과 연계한 인증서 유효성 검사 기술(예: CRL 또는 OCSP)을 적용할 수 있다. 오퍼레이터와 게이트웨이 간의 인증은 비밀키와 HMAC(Hashed MAC)[6]을 이용하여 RSH 프로토콜보다 효율적인 메커니즘을 제공함으로써 <요구사항 2-3>을 만족시키기로 한다.

2.3 게이트웨이와 오퍼레이터 간의 비밀키 공유

OSGi 환경에서 게이트웨이와 오퍼레이터 간의 안전한 비밀키의 공유는 다음과 같은 요구사항을 만족해야 한다.

- <요구사항 3-1> 공유 비밀키는 사용자 인증 티켓의 암호화에 사용되며, 게이트웨이에 설치되는 번들의 인증 정보를 생성하고 검증하는 연산에 사용된다. 따라서 사용자 인증 메커니즘이나 번들 인증 메커니즘에서 쉽게 이용할 수 있는 서비스 형태로 제공될 필요가 있다.
- <요구사항 3-2> 비밀키의 공유 과정에서 게이트웨이와 오퍼레이터는 쌍방향 인증이 가능해야 하며, 비밀키는 장시간 사용되면 암호해독법(cryptanalysis) 공격에 취약해질 수 있으므로 어느 시점에 변경이 가능해야 한다.
- <요구사항 3-3> 비밀키 공유에 사용되는 고정된 시스템 키의 안전한 관리를 고려해야 한다. 오퍼레이터는 게이트웨이 별로 고유한 시스템 키를 부여하고 사전에 오프라인으로 제공한다. 시스템 키는 공개키/개인키 쌍이 될 수도 있지만 게이트웨이의 자원을 고려할 때 대칭키로 제공되는 것이 바람직하다.

개방된 네트워크로 연결된 시스템 간에 비밀키(또는 세션키)를 공유하기 위한 프로토콜은 매우 다양하다[7-9]. 본 논문의 보안 아키텍처에서는 키 공유 메커니즘이 교환한 비밀키를 사용자 인증 메커니즘과 번들 메커니즘에서 이용할 수 있도록 독립적인 서비스로 제공하여 <요구사항 3-1>을 만족시킨다. 키 공유는 게이트웨이의 부트스트래핑(bootstrapping) 단계에 수행된다. 비밀키 공유 방법으로 대칭키 기반의 Needham-Schroeder 프로토콜[7]을 사용한다. 따라서 <요구사항 3-2>에 따라 쌍방향 인증을 제공하면서 동시에 필요한 시점에 비밀키의 변경이 가능하다. 비밀키의 생성에 사

용되는 시스템 키는 대칭키를 이용하며, 키 공유 메커니즘을 구현한 서비스에서만 접근할 수 있는 권한이 부여되므로 <요구사항 3-3>을 만족시킨다.

2.4 게이트웨이 권한부여

OSGi 환경에서 게이트웨이에 설치되는 서비스와 자원에 대한 권한부여는 다음과 같은 요구사항을 만족해야 한다.

- <요구사항 4-1> 사용자 또는 번들의 인증 메커니즘과 연계를 고려해야 한다. 오퍼레이터는 접근을 요청하는 주체(subject)에 대한 인증 정보를 관리하고 게이트웨이에 안전하게 전달하여 인증 메커니즘에서 접근통제 요청을 가능하게 해야 한다.
- <요구사항 4-2> 권한부여의 주체가 되는 사용자와 번들(또는 서비스 제공자)을 그룹화하거나 대상(target)이 되는 서비스와 자원에 대한 추상화로 권한의 설정을 간단하게 할 수 있어야 한다.
- <요구사항 4-3> 서비스 제공자는 자신이 제공하는 번들에 대한 권한부여를 책임짐으로써 온디맨드로 설치되는 번들을 모두 오퍼레이터가 인식하고 있어야 하는 부담을 줄여줄 수 있다. 하지만 권한부여 과정에서 오퍼레이터의 보안 정책과 서비스 제공자의 보안 정책을 결합하여 안전하고 유연한 접근통제의 결정이 가능해야 한다.

현재 OSGi 스펙에서 제안하는 Java 2 보안 아키텍처 [10]의 중앙 집중형 권한부여는 유연성 있는 고수준의 보안 관리가 어렵고, 서로 다른 위치에서 전송된 이동 코드 간의 상호작용에 필요한 보안 관계의 설정이 불가능한 단점을 가지고 있다. 이를 극복하기 위하여 Java 환경에서 비집중형 권한부여에 대한 다양한 연구가 이루어졌다[11-13]. 본 논문의 보안 아키텍처에서는 동적으로 설치되는 번들의 접근 권한을 위임, 허가 또는 거부할 수 있는 보안 정책을 제공하여 관리의 유연성과 확장성을 높이는 한편, 구체적인 권한부여 모델을 통하여 안전한 권한부여 메커니즘의 구현이 가능하도록 하였다. <요구사항 4-1>에 따라 사용자 인증 메커니즘과 번들 인증 메커니즘을 거친 인증 정보를 이용하여 접근의 주체를 식별한다. 또한 주체와 대상에 대한 그룹화, 권한의 위임, 예외 설정 등이 가능하도록 하여 <요구사항 4-2>를 만족시킨다. 권한부여 모델은 <요구사항 4-3>에 따라 서비스 제공자가 자신의 번들에 대한 보안 정책을 설정하여 전송하고 오퍼레이터의 보안 정책과 충돌이 발생하는 부분을 해결하는 방안을 포함하고 있다.

3. OSGi 보안 아키텍처의 명세

이 장에서는 OSGi 보안 아키텍처를 구성하는 개별적

인 보안 메커니즘을 설명하고, 구체적인 시스템으로 구현하는데 필요한 정형화된 보안 프로토콜과 보안 모델을 명세한다.

3.1 사용자 인증 메커니즘

사용자 인증 메커니즘의 구성요소에는 클라이언트의 사용자 프로그램, 사용자에게 특정한 서비스를 제공하는 번들 서비스(BS), 게이트웨이의 프레임워크에 설치되는 로그인 서비스(LS)와 오퍼레이터가 관리하는 사용자 인증 서버(AS)가 있다. 로그인 서비스는 번들 서비스를 대신해서 인증 서버가 발급한 인증 티켓을 확인해주는 기능을 하는 점에서 Kerberos 시스템의 티켓 발급 서버(TGS)와 유사하지만 TGS와 달리 서비스 티켓(service ticket)은 발급하지 않으며 번들 서비스에 인증의 결과로 권한부여 정보(즉, 사용자 식별자와 역할 정보)를 전송한다.

사용자 인증 메커니즘은 이러한 구성요소를 기반으로 표 1과 같은 인증 프로토콜을 수행한다.

표 1 사용자 인증 프로토콜

[표기법]	
ID _U , ID _{BS} , ID _{SG}	사용자 U, 번들 서비스 BS, 게이트웨이 SG의 식별자
K _{a,b}	A와 B 간의 공유 비밀키(대칭키)
{M}K _{a,b}	K _{a,b} 를 사용하여 암호화된 메시지 M
TS _{it}	타임스탬프
TS _{init}	최초 인증 시간
Ticket _{auth}	인증 티켓
Auth	권한부여 요청 정보
Subject	서비스 사용 권한부여 정보
Role	인증된 사용자의 역할
	연결(concatenation)
[단계별 프로토콜]	
① U → AS:	ID _U ID _{SG} N _U
② AS → U:	{K _{a,ls} ID _{SG} N _U TS _{it} }K _{a,as} Ticket _{auth}
③ U → LS:	ID _U ID _{BS} Ticket _{auth} Auth
④ LS → BS:	{ID _U TS ₂ }K _{a,ls} Subject
⑤ BS → U:	{ID _U TS ₂ }K _{a,ls}
[인증 티켓과 권한부여 요청 정보]	
Ticket _{auth} = ID _{SG} {K _{a,ls} ID _U Role TS _{init} }K _{ls,as}	
Auth = {ID _U ID _{BS} TS ₂ }K _{a,ls}	

사용자 인증 프로토콜은 5 단계로 이루어져 있으며, 각 단계별 프로토콜의 전송 내용과 처리 방법은 다음과 같다.

① 사용자 인증 요청(U → AS): 사용자가 특정 번들 서비스를 사용하고자 할 때 먼저 오퍼레이터의 인증 서버로부터 인증을 받아야 한다. 사용자는 자신의 식별자인 ID_U와 속하려는 게이트웨이의 식별자인 ID_{SG}, 그리고 새로운 난수 임시값 N_U을 인증 서버에 전송한다.

② 인증 티켓 발급(AS → U): 인증 서버는 세션키인

$K_{u,ls}$ 와 인증 티켓 $Ticket_{auth}$ 을 생성하여 사용자에게 전송한다. 세션키는 사용자가 미리 인증 서버에 등록한 패스워드로부터 유도된 비밀키 $K_{u,as}$ 로 암호화된다. 인증 티켓은 로그인 서비스와 인증 서버 간에 공유한 비밀키 $K_{ls,as}$ 로 암호화된다. $K_{ls,as}$ 는 게이트웨이의 부트스트래핑 과정에서 수행되는 키 공유 메커니즘을 통해 생성한다.

- ③ 서비스 권한부여 요청($U \rightarrow LS$): 사용자는 접속하고자 하는 게이트웨이의 특정한 번들 서비스에 인증 티켓을 전송한다. 인증 티켓은 암호화되어 있기 때문에 네트워크 상에서의 기밀성과 무결성을 보장한다. 또한 사용자는 세션키 $K_{u,ls}$ 를 사용하여 암호화한 권한부여 요청 정보 Auth를 함께 전송한다. 번들 서비스는 인증 티켓과 권한부여 요청 정보를 그대로 로그인 서비스에 전달한다. 이때 인증 티켓과 요청 정보는 각각 $K_{ls,as}$ 와 $K_{u,as}$ 로 암호화되어 있으므로 신뢰할 수 없는 번들 서비스에 의한 변조를 방지할 수 있다.

- ④ 권한부여 정보 전송($LS \rightarrow BS$): 로그인 서비스는 비밀키 $K_{ls,as}$ 를 사용하여 인증 티켓을 복호화한다. 또한 인증 티켓에 포함된 세션키 $K_{u,as}$ 로 사용자가 전송한 요청 정보를 복호화하고 인증 티켓의 정당한 사용자인지 확인한다. 인증이 성공하면 인증 티켓에 포함된 역할 Role이 게이트웨이 보안 정책에 있는 해당 번들 서비스의 역할 집합에 속하는지 검사한 후 권한부여 여부를 결정한다. 로그인 서비스는 권한부여 결과를 세션키 $K_{u,ls}$ 로 암호화하고 시스템 자원의 사용에 대한 접근 권한을 포함한 Subject 객체를 함께 번들 서비스에 전송한다. Subject 객체는 JAAS (Java Authentication and Authorization Service) [14]에서 주체(즉, 사용자)의 신원과 보안 속성을 표현하는데 사용되며, 게이트웨이의 자원에 대한 접근 권한을 결정하는데 사용된다.

- ⑤ 권한부여 결과 전송($BS \rightarrow U$): 번들 서비스는 로그인 서비스로부터 전달받은 내용 중에서 Subject 객체를 제외한 나머지 권한부여 결과를 그대로 사용자에게 전송한다. 사용자는 세션키 $K_{u,ls}$ 로 권한부여 결과를 복호화하여 요청한 번들 서비스의 승인 여부를 확인한다.

사용자는 다수의 번들 서비스에 인증하기 위하여 매번 패스워드를 입력할 필요 없이 한 번의 패스워드 입력으로 수신한 인증 티켓을 재사용하므로 SSO의 개념을 만족한다. 또한 인증 티켓과 전송 메시지의 암호화에 복잡한 공개키 연산 대신 대칭키 연산을 사용함으로써 사용자 인증 작업의 수행에 있어 좀 더 빠르고 적은 시스템 자원을 사용하도록 하고 있다.

3.2 번들 인증 메커니즘

번들 인증 메커니즘은 서비스 제공자(SP)와 오퍼레이터(OP) 간에 공개키 연산을 이용한 인증과 오퍼레이터와 게이트웨이(SG) 간에 MAC 연산을 이용한 인증으로 구분된다. 번들 인증 프로토콜에 대한 명세는 표 2와 같다.

표 2 번들 인증 프로토콜

[표기법]	
ID_{SP} , ID_{SG}	서비스 제공자 SP, 게이트웨이 SG의 식별자
ID_{SB}	서비스 번들 SB의 식별자
SB_{SP}	SP가 제공하는 서비스 번들
$Cert_{SP}$	인증기관(CA)이 발급한 SP의 공개키 인증서
Mac_{SG}	오퍼레이터가 SG를 위해 생성한 메시지 인증 코드(MAC)
KU_{SP} , KR_{SP}	SP의 공개키와 개인키
$\{M\}KR_{SP}$	KR_{SP} 를 사용하여 암호화된 메시지 M
$K_{op,sg}$	OP와 SG 간의 공유 비밀키(MAC 키)
$C[M]K_{op,sg}$	$K_{op,sg}$ 를 사용하여 메시지 M에 대해 생성한 MAC
$H(M)$	메시지 M에 대한 해쉬 코드
N_{OP} , N_{SG}	OP와 SG가 각각 생성한 난수 임시값(nonce)
[단계별 프로토콜]	
① $SP \rightarrow OP$: $ID_{SB} \parallel SB_{SP} \parallel \{H(ID_{SB} \parallel SB_{SP})\}KR_{SP} \parallel Cert_{SP}$	
② $SG \rightarrow OP$: $ID_{SG} \parallel ID_{SB} \parallel N_{SG}$	
③ $OP \rightarrow SG$: $ID_{SP} \parallel ID_{SB} \parallel SB_{SP} \parallel N_{OP} \parallel Mac_{SG}$	
[공개키 인증서와 MAC 정보]	
$Cert_{SP} = \{ID_{SP} \parallel KU_{SP}\}KR_{CA}$	
$Mac_{SG} = C[ID_{SP} \parallel H(ID_{SB} \parallel SB_{SP}) \parallel N_{OP} \parallel N_{SG}]K_{op,sg}$	

번들 인증 프로토콜은 3 단계로 이루어져 있으며, 각 단계별 프로토콜의 전송 내용과 처리 방법은 다음과 같다.

- ① 초기 등록 번들 인증($SP \rightarrow OP$): 이 단계는 ②, ③과 비연속적인 별도의 과정으로 분리되며, 오퍼레이터는 서비스 제공자의 번들 서버(예: 웹 서버)에 접속하여 서명된 JAR 형식으로 서비스 번들 SB_{SP} 을 전송받는다. 서명된 JAR는 번들의 해쉬 코드 $H(ID_{SB} \parallel SB_{SP})$ 를 갖는 매니페스트 파일(manifest file)과 서비스 제공자의 개인키 KR_{SP} 로 매니페스트 파일을 암호화한 서명 파일(signature file)을 포함한다. 또한 오퍼레이터가 서명을 검증할 수 있도록 서비스 제공자의 공개키 KU_{SP} 가 포함된 인증서를 제공한다. 오퍼레이터는 인증서의 유효성을 검사하고 KU_{SP} 를 이용하여 서명 파일을 복호화한 후 매니페스트 파일의 해쉬 코드와 동일인지 검사함으로써 번들에 대한 인증을 수행하게 된다.

- ② 게이트웨이 번들 요청($SG \rightarrow OP$): 게이트웨이는 번들을 요청하기 위하여 임시값 N_{SG} 를 생성하여 게이트웨이 식별자 ID_{SG} , 번들 식별자 ID_{SB} 와 함께 오퍼레이터에 전송한다. 임시값은 번들을 요청할 때마다

새로 생성되는 안전한 난수이며, 번들의 신선도(freshness)를 보장하는 역할로 사용된다.

- ③ 게이트웨이 번들 인증(OP → SG): 오퍼레이터는 ①에서 인증된 번들에 게이트웨이가 인증할 수 있는 인증 정보를 추가하여 전송한다. 인증 정보는 게이트웨이와 오퍼레이터가 부트스트래핑 과정에 공유한 비밀키 $K_{op,sg}$ 를 사용하는 HMAC으로 계산된다. 이 계산에는 서비스 제공자의 식별자 ID_{SB} 와 번들의 매니페스트 파일인 해쉬 코드 $H(ID_{SB} || SB_{SP})$, 그리고 게이트웨이의 임시값 N_{SG} 및 오퍼레이터가 새로 생성한 임시값 N_{OP} 가 포함된다. 게이트웨이는 전송된 번들에 대해서 동일한 HMAC을 계산하고 수신한 Mac_{SG} 과 일치하는지 비교하여 인증을 수행한다.

게이트웨이에서 수행되는 번들의 인증은 해쉬 함수를 이용한 MAC의 계산이므로 공개키 연산을 이용하는 방식보다 매우 효율적이다. 또한 오퍼레이터는 서명된 JAR에 포함된 서명 파일과 서비스 제공자의 공개키 인증서를 제외하고 게이트웨이에 전달할 수 있으므로 전송 속도를 향상시킬 수 있다. 모든 번들의 내용 대신에 서명된 JAR에 포함된 매니페스트 파일을 이용하여 MAC을 생성하므로 모든 번들에 대한 MAC을 계산하는 현재 OSGi 스펙의 RSH 프로토콜에 비해 효율적인 성능을 제공한다. 실제 번들의 내용과 매니페스트 파일의 해쉬 코드를 비교하는 작업은 프레임워크의 Java 클래스 로더에서 새로운 클래스를 로딩하는 시점에 수행된다.

3.3 키 공유 메커니즘

키 공유 메커니즘은 부트스트래핑 단계에서 오퍼레이터와 게이트웨이 간에 쌍방향 인증을 거쳐 안전한 공유 비밀키를 제공하기 위한 방법이다. 공유된 비밀키는 사용자 인증 프로토콜에서 인증 서버와 로그인 서비스 간의 인증 티켓 암호화에 사용되며, 오퍼레이터와 게이트웨이 간의 번들 인증 프로토콜에서 MAC 키로 사용된다. 키 공유 프로토콜에 대한 명세는 표 3과 같다.

표 3 키 공유 프로토콜

[표기법]	
ID_{OP} , ID_{SG}	오퍼레이터 OP, 게이트웨이 SG의 식별자
N_A	A가 생성한 난수 임시값
K_{sg}	SG의 시스템 키(대칭키)
$(M)K_{sg}$	K_{sg} 를 사용하여 암호화된 메시지 M
KDF	키 유도 함수
[단계별 프로토콜]	
① SG → OP: $ID_{SG} \{ID_{SG} N_{SG}\}K_{sg}$	
② OP → SG: $ID_{OP} \{ID_{OP} N_{SG} N_{OP}\}K_{sg}$	
③ SG → OP: $ID_{SG} \{ID_{SG} N_{OP}\}K_{sg}$	
[공유 비밀키 계산]	
$K_{op,sg} = KDF(K_{sg}, N_{OP} N_{SG})$	

키 공유 프로토콜은 3 단계로 이루어져 있으며, 각 단계별 프로토콜의 전송 내용과 처리 방법은 다음과 같다.

- ① 키 공유 요청(SG → OP): 게이트웨이는 안전한 난수 임시값 N_{SG} 를 생성한 후 자신의 식별자 ID_{SG} 와 함께 시스템 키 K_{sg} 로 암호화하여 오퍼레이터에 전송한다. 오퍼레이터는 미리 등록한 게이트웨이의 시스템 키 K_{sg} 로 메시지를 복호화하여 게이트웨이의 임시값 N_{SG} 를 추출한다. 평문 형식의 식별자 ID_{SG} 는 제대로 복호화된 것인지 판별하게 해준다.
- ② 오퍼레이터 인증 및 키 공유(OP → SG): 오퍼레이터는 새로운 임시값 N_{OP} 를 생성한 후 게이트웨이가 전송한 임시값 N_{SG} 와 함께 시스템 키 K_{sg} 로 암호화하여 게이트웨이에 전송한다. 게이트웨이는 메시지를 복호화하여 N_{SG} 와 N_{OP} 를 추출한다. 자신이 전송한 임시값과 N_{SG} 가 동일한지 검사하여 오퍼레이터에 대한 인증을 수행한다. 평문 형식의 식별자 ID_{OP} 는 제대로 복호화된 것인지 판별하게 해준다. 게이트웨이는 키 유도 함수(key derivation function)인 KDF()를 사용하여 비밀키를 얻는다. 키 유도 함수는 시스템 키인 K_{sg} 를 MAC 키로 공유한 임시값 N_{SG} 와 N_{OP} 에 대한 HMAC/SHA-1을 반복적으로 적용하는 IKE[15]와 유사한 방법을 사용한다.
- ③ 게이트웨이 인증 및 키 공유(SG → OP): 게이트웨이는 오퍼레이터가 전송한 임시값 N_{OP} 를 시스템 키 K_{sg} 로 암호화하여 다시 오퍼레이터에 전송한다. 오퍼레이터는 메시지를 복호화하여 N_{OP} 를 추출한 후 자신이 전송한 임시값과 동일한지 검사하여 게이트웨이에 대한 인증을 수행한다. 오퍼레이터는 게이트웨이와 동일한 키 유도 함수를 적용하여 비밀키를 얻는다.

게이트웨이와 오퍼레이터 간의 시스템 키로 PKI 기반의 공개키 인증서를 이용할 경우에 키 관리나 분배와 갱신이 쉬울 수 있지만 암호화 속도와 전송 속도가 느려진다. 또한 인증서의 검증을 위한 인증기관과 연동하는데 시간이 많이 걸리며 연산에 따른 시스템 자원이 많이 소모된다. 따라서 제한적 자원을 가지고 있는 OSGi 환경에서는 대칭키를 적용한 키 공유가 더욱 효율적이다. 또한 대칭키를 이용한 키 공유 메커니즘은 공개키를 이용한 메커니즘에 비해 구현이 용이하므로 적은 비용으로 구축할 수 있다.

3.4 권한부여 메커니즘

권한부여 메커니즘은 게이트웨이에 배치되는 번들에 대한 적절한 권한을 부여하는 방법이다. 여기서는 어플리케이션의 설치 요청에 의해 외부망을 통하여 동적으로 설치된 번들의 서비스를 이용하여 다른 번들의 서비스나 게이트웨이의 자원에 접근할 때 발생할 수 있는

보안 문제를 해결할 수 있도록 표 4와 같은 정형화된 권한부여 모델(authorization model)을 명시한다. 이 모델에서는 정형화된 의미론(semantics)을 정의하기 위하여 Jajodia 등이 개발한 권한부여 명세 언어인 ASL (Authorization Specification Language)[16]의 변형된 버전을 사용한다.

권한부여 모델은 4 단계로 이루어져 있으며, 각 단계별 규칙의 의미는 다음과 같다.

- ① 권한부여 규칙(authorization rule): 서비스 제공자가 작성한 번들 보안 정책에 명시된 모든 엔트리를 정의하며, 모델에서는 `cando()`로 표현된다. ASL의 규칙은 우측의 조건이 모두 true이면 좌측의 규칙이 생성됨을 의미한다. [규칙 1.1]에서 우측에 오는 조건은 없다.
- ② 유도 규칙(derivation rule): 게이트웨이 내에 설치된 번들이 자신의 도메인 내에서 확장 보안 정책(extended security policy)의 엔트리를 생성하며, 모델에서는 `dercando()`로 표현된다. 새로 설치된 번들의 보안 정책은 설치를 요청한 번들의 확장 보안 정

책과 결합하여 새로운 확장 보안 정책의 엔트리를 생성하게 된다. [규칙 2.1]은 위임 엔트리(delegate entry)가 생성되는 조건을 명시한다. [규칙 2.2]와 [규칙 2.3]은 허가 엔트리(grant entry)가 생성되는 조건을 명시한다. [규칙 2.4]와 [규칙 2.5]는 거부 엔트리(deny entry)가 생성되는 조건을 명시한다.

- ③ 해결 규칙(resolution rule): 번들의 확장 보안 정책 내에서 발생할 수 있는 허가 권한과 거부 권한의 충돌을 해결하여 접근통제 결정에 사용될 엔트리를 정의하며, 모델에서는 `do()`로 표현된다. [규칙 3.1]과 [규칙 3.2]는 충돌 해결을 위해 사용된 거부 우선순위(denials take precedence) 규칙을 의미한다.
- ④ 접근통제 규칙(access control rule): 번들의 사용자 또는 서비스로부터 도메인으로 요청된 접근에 대한 허가 여부를 결정하며, 모델에서는 `grant()`로 표현된다. [규칙 4.1]의 `implies()`는 요청에 포함된 주체, 대상, 액션과 도메인에 포함된 주체, 대상, 액션 간의 의미적인 내포 관계를 나타낸다. 따라서 주체와 대상에 대한 그룹화와 액션에 대한 추상화가 가능하다.

표 4 권한부여 모델

[표기법]	
<i>s</i>	게이트웨이 자원에 접근을 요청하는 주체
<i>t</i>	접근 요청의 대상이 되는 자원
<i>d</i>	번들 설치 체인에 따라서 접근통제의 요청이 발생하는 도메인
<i>a</i>	위임('#'), 허가('+'), 거부('-')가 명시될 액션
<code>cando()</code>	번들의 보안 정책에 명시된 엔트리
<code>dercando()</code>	번들의 확장 보안 정책에 명시된 엔트리
<code>do()</code>	권한의 충돌(conflict)이 해결된 엔트리
<code>grant()</code>	게이트웨이에서 실제로 번들에 할당되는 퍼미션
<code>in()</code>	번들 설치 체인의 계층 구조
<code>implies()</code>	주체, 대상, 액션 간의 의미적 내포 관계
[단계별 규칙]	
① 권한부여 규칙	
[규칙 1.1] $\text{cando}(s, t, d, \langle \# + - \rangle a) \leftarrow .$	
② 유도 규칙	
[규칙 2.1] $\text{dercando}(s, t, d_1, \#a) \leftarrow \text{cando}(s, t, d_1, \#a) \ \& \ \text{dercando}(s, t, d_2, \#a) \ \& \ \text{in}(d_1, d_2).$	
[규칙 2.2] $\text{dercando}(s, t, d, +a) \leftarrow \text{cando}(s, t, d, \langle \# \ + \rangle a).$	
[규칙 2.3] $\text{dercando}(s, t, d_1, +a) \leftarrow \text{cando}(s, t, d_1, \langle \# \ + \rangle a) \ \& \ \text{dercando}(s, t, d_2, \#a) \ \& \ \text{in}(d_1, d_2).$	
[규칙 2.4] $\text{dercando}(s, t, d, -a) \leftarrow \text{cando}(s, t, d, -a).$	
[규칙 2.5] $\text{dercando}(s, t, d_1, -a) \leftarrow \sim \text{cando}(s, t, d_1, -a) \ \& \ \text{dercando}(s, t, d_2, -a) \ \& \ \text{in}(d_1, d_2).$	
③ 해결 규칙	
[규칙 3.1] $\text{do}(s, t, d, +a) \leftarrow \text{dercando}(s, t, d, +a) \ \& \ \sim \text{dercando}(s, t, d, -a).$	
[규칙 3.2] $\text{do}(s, t, d, -a) \leftarrow \text{dercando}(s, t, d, -a).$	
④ 접근통제 규칙	
[규칙 4.1] $\text{grant}(s', t', d, +a') \leftarrow \text{do}(s, t, d, +a) \ \& \ \sim \text{do}(s, t, d, -a) \ \& \ \text{implies}(s, s') \ \& \ \text{implies}(t, t') \ \& \ \text{implies}(a, a')$	

서비스 제공자는 자신이 제공하는 번들이 등록한 서비스에 접근할 수 있는 권한을 다른 번들의 서비스에 부여하도록 독자적인 보안 정책을 가질 수 있다. 게이트웨이에서 작동하는 모든 번들에 대한 보안 정책을 오퍼레이터가 책임을 지고 설정하는 것은 번들에 대한 세부적인 구현 지식을 요구하며 필요에 따라 오퍼레이터의 인식 밖에 있는 다른 번들을 설치하여 사용할 수 있는 가용성을 해치게 되므로 바람직하지 않다. 권한부여 모델에서 주체와 대상은 게이트웨이에 등록된 서비스 또는 자원에 대한 접근을 요청하는 번들의 서비스 제공자 명칭과 번들을 다운로드할 URL 위치로 표현될 수 있다. 주체와 대상의 식별 정보를 그룹화하여 오퍼레이터가 서비스 제공자와 번들에 대한 세부적인 지식이 없이도 보안 정책을 수립할 수 있도록 정의하고 있다. 오퍼레이터와 서비스 제공자는 번들 인증 메커니즘을 거쳐 인증된 번들의 신뢰도에 따라 다양한 보안 정책을 수립할 수 있다. 일반적으로 완전히 신뢰할 수 있는 번들에 대해 위임 엔트리를 설정하고, 비교적 믿을 수 있는 번들에 대해서는 허가 엔트리를 설정하며, 신뢰할 수 없는 번들(예: 블랙리스트)에 대해서는 거부 엔트리를 설정하는 방식을 사용한다.

4. OSGi 보안 아키텍처의 설계 및 구현

이 장에서는 OSGi 보안 아키텍처에 따라 게이트웨이와 오퍼레이터에 설치될 보안 시스템을 설계한 UML 클래스 다이어그램(class diagram)과 프로토타입으로 구현한 결과를 보여준다.

4.1 전체 구성도 및 구현 환경

보안 아키텍처를 구현한 시스템의 전체적인 구성도는 그림 2와 같다. 게이트웨이에서 수행되는 보안 메커니즘은 프레임워크에서 작동하는 보안 서비스 형태로 제공된다. 서비스 제공자로부터 제공되는 모든 서비스 번들은 번들 저장소(bundle repository)에 저장되며, 게이트웨이의 요청에 따라 프레임워크로 이동한다. 보안 서비스인 로그인 서비스, 번들 인증 서비스, 키 공유 서비스, 접근통제 서비스는 게이트웨이에 미리 설치된 고유(built-in) 서비스로 제공된다. 보안 메커니즘을 수행하기 위해 각각의 서비스는 오퍼레이터의 사용자 인증 서버, 번들 인증 서버, 키 공유 서버, 보안정책 관리기와 연동하며, 개념적으로 사용자 인증 시스템, 번들 인증 시스템, 키 공유 시스템, 접근통제 시스템을 구성한다.

시스템의 프로토타입 구현을 위하여 오퍼레이터 및 서비스 제공자는 모두 Window2000 서버를 운영체제로 하고 개발언어는 JDK 1.3을 사용하였다. 사용자에 대한 인증 정보, 번들의 인증 정보 및 게이트웨이 정보 등을 포함하고 있는 번들 저장소로는 Oracle 8i를 사용하였

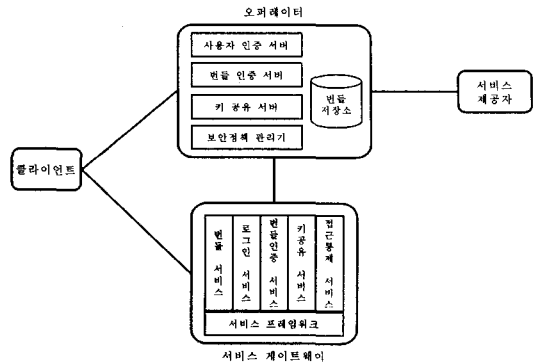


그림 2 OSGi 보안 아키텍처의 전체 시스템 구성도

다. 게이트웨이의 미들웨어로는 OSGi 프레임워크와 호환되는 Sun Microsystems의 JES(Java Embedded Server 2)[17]를 사용하였으며, 하드웨어는 Ericsson의 E-Box[18]와 유사한 환경(200 MHz CPU, 64MB RAM, 10/100M Ethernet Adapter)을 구축하였다. 암호화 연산에 필요한 암호화 라이브러리는 (주)시큐리티테크놀로지의 J/LOCK[19]을 사용하였다.

4.2 사용자 인증 시스템

사용자 인증 시스템은 오퍼레이터에 설치되는 사용자 인증 서버, 게이트웨이에 설치되는 로그인 서비스 및 번들 서비스, 클라이언트에서 수행되는 사용자 프로그램으로 구성된다.

그림 3은 사용자 인증 서버의 구조를 보여주는 클래스 다이어그램이다. 사용자의 인증 요청에 대한 처리와 인증 티켓의 생성은 GenericRegistryHandler 클래스의 execute() 메소드에서 수행하며, generateSecretKey() 메소드에서는 사용자와 로그인 서비스 간의 공유 대칭키를 생성한다.

그림 4는 게이트웨이에 등록된 번들 서비스 LightService와 로그인 서비스의 연동을 보여주는 클래스 다이어그램이다. LSActivator 클래스는 로그인 서비스를 프레임워크에 등록하는데 사용되며, 사용자에 대한 로그인은 BundleLoginServiceImpl 클래스의 getSubject() 메소드에서 수행된다. getSubject() 메소드는 인증 티켓의 유효성을 검사하고 사용자에 대한 권한부여 정보를 Java 2의 Policy 클래스[20]에서 얻은 후 Subject 객체를 반환한다.

그림 5는 사용자가 클라이언트 프로그램을 통해서 인증 서버로부터 인증 티켓을 얻고 게이트웨이에 설치된 서비스 LightService에 접근하는 화면이며, 그림 6은 LightService에서 얻은 인증 티켓을 재사용하여 다른 서비스 HeaterService에 접근하는 SSO 기능을 보여준다.

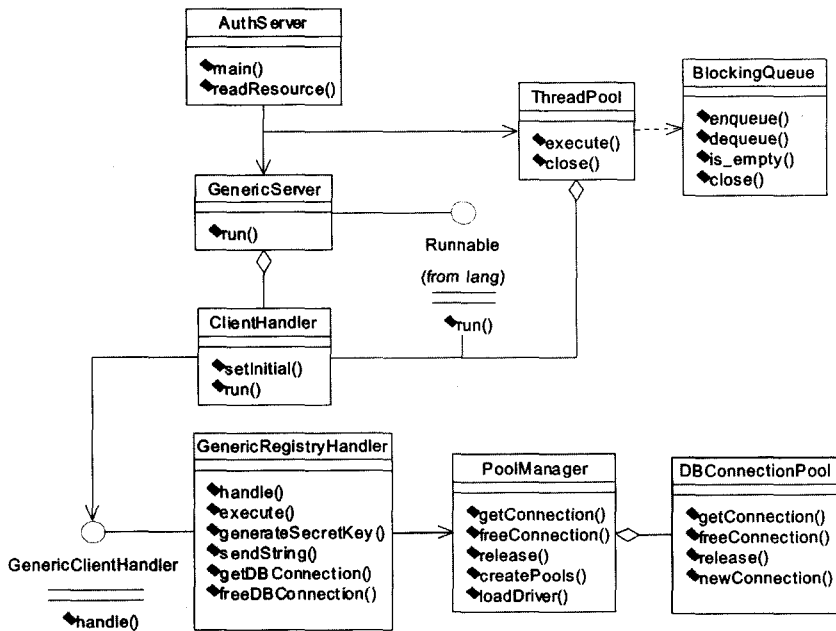


그림 3 사용자 인증 서버의 클래스 다이어그램

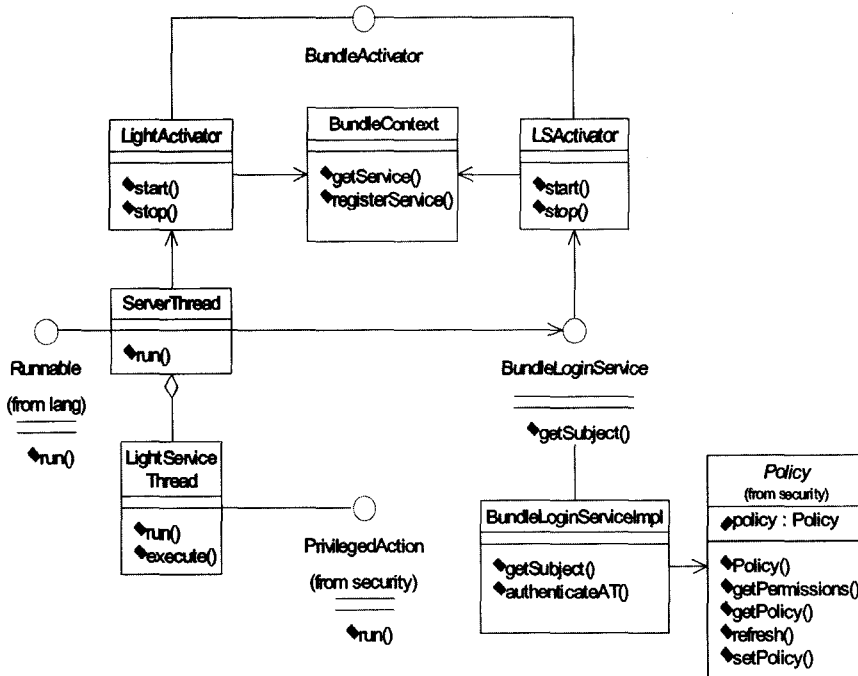


그림 4 번들 서비스와 로그인 서비스의 클래스 다이어그램

4.3 번들 인증 시스템

번들 인증 시스템은 오퍼레이터에 설치되는 번들 인증 서버와 게이트웨이에 설치되는 번들 인증 서비스로 구성된다.

그림 7은 번들 인증 서버의 구조를 보여주는 클래스 다이어그램이다. **BundleAuthSvr** 클래스에서는 게이트웨이에 요청한 서비스 번들에 대해 **MacGenerator** 클래스를 이용하여 서명된 JAR에 포함된 매니페스트 파

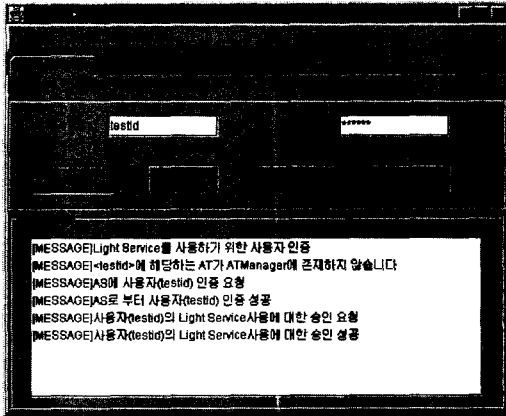


그림 5 LightService에 대한 사용자 인증

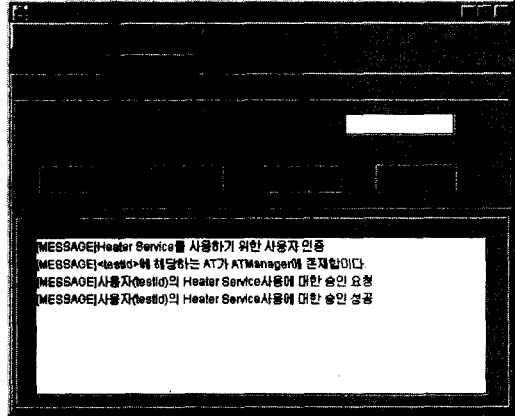


그림 6 HeaterService에 대한 사용자 인증(SSO)

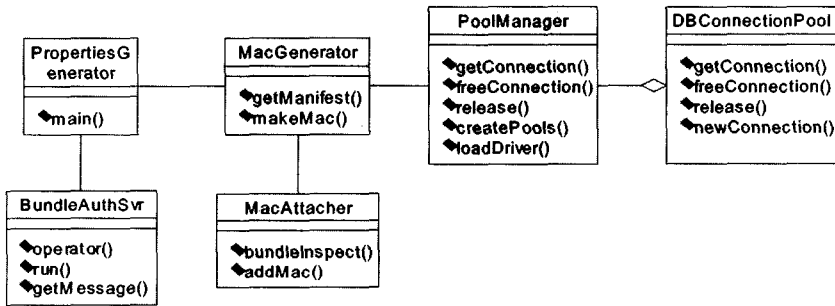


그림 7 번들 인증 서버의 클래스 다이어그램

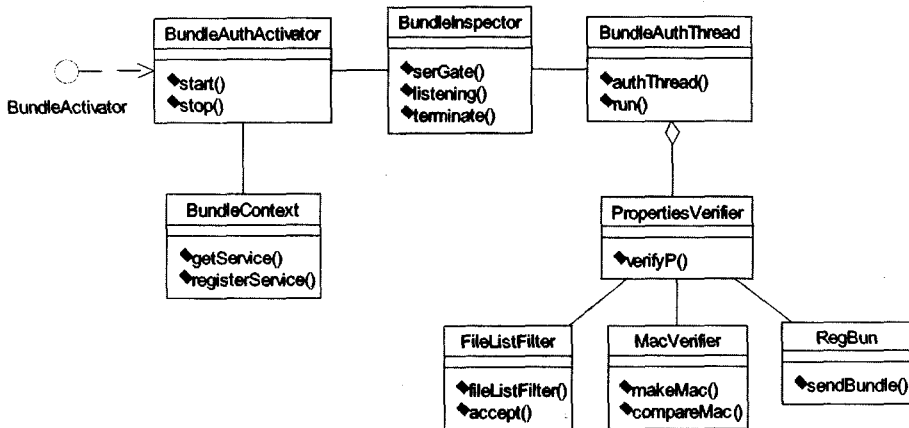


그림 8 번들 인증 서비스의 클래스 다이어그램

일에 대한 MAC을 생성하고, MacAttacher 클래스를 이용하여 PKI 전자서명 부분을 MAC으로 대치한 JAR 파일 형식으로 변형한다. 생성된 MAC과 번들 인증 정보는 PropertiesGenerator 클래스를 통해 번들 저장소에 유지된다.

그림 8은 번들 인증 서비스의 구조를 보여주는 클래스 다이어그램이다. 번들 인증 서비스는 BundleAuthActivator 클래스를 통해 프레임워크에 등록된다. BundleInspector 클래스는 오퍼레이터로부터 전송된 번들을 받아서 PropertiesVerifier 클래스를 이용하여 인



그림 9 번들 인증 서비스의 실행 화면

증을 수행한다. MacVerifier 클래스는 MAC을 생성하고 번들에 포함된 번들 인증 서버의 MAC과 비교하는 역할을 수행한다.

그림 9는 오퍼레이터가 전송한 번들 LogMonitor.jar에 대해 게이트웨이의 번들 인증 서비스에서 MAC을 검증하는 실행 화면이다.

4.4 키 공유 시스템

키 공유 시스템은 오퍼레이터에 설치되는 키 공유 서버와 게이트웨이에 설치되는 키 공유 서비스로 구성된다.

그림 10은 키 공유 서비스의 구조를 보여주는 클래스 다이어그램이다. 키 공유 서버도 유사한 클래스로 구성된다. KeyExchangeActivator 클래스는 오퍼레이터의 응답을 대기하여 처리하며, KeyExThread 클래스를 통해 난수 임의값의 교환을 수행한다. NonceComparision 클래스

의 compareNonce() 메소드를 통해 오퍼레이터와 게이트웨이 간에 쌍방향 인증이 이루어지면 KeyGenerator 클래스의 getSharedSecret() 메소드에서 키 유도 함수를 수행하여 안전한 공유 비밀키를 생성한다.

4.5 접근통제 시스템

접근통제 시스템은 오퍼레이터에 설치되는 보안 정책 관리기와 게이트웨이에 설치되는 접근통제 서비스로 구성된다.

그림 11은 보안 정책 관리기의 구조를 보여주는 클래스 다이어그램이다. PolicyManager 클래스의 요청에 의해 PolicyLoader 클래스는 번들 저장소에서 번들과 함께 저장된 보안 정책 파일과 번들 인증 시스템에 의해 기록된 인증 정보를 가져온다. PolicyExtender 클래스는 번들의 설치를 요청한 상위 번들의 확장 보안 정책과 요청된 번들의 보안 정책 파일을 결합하여 권한부여 모델의 유도 규칙에 따라 새로운 확장 보안 정책을 생성한다. PolicyResolver 클래스는 해결 규칙과 접근통제 규칙을 적용하여 게이트웨이의 접근통제 서비스에서 실제로 허가될 권한을 생성한다.

게이트웨이의 접근통제 서비스는 단순히 프레임워크의 번들 로더(bundle loader)에서 보안 정책 관리기에 안전하게 접속(즉, 공유 비밀키로 암호화된 채널을 사용)하여 번들에 대한 권한을 얻는데 사용된다. 번들 로더는 접근통제 서비스가 가져온 권한에 따라 번들의 클래스에 대한 보호 도메인(protection domain)을 할당하며, Java 2의 AccessController 클래스[20]에서 자원에 대한 접근통제를 수행한다.

그림 12는 오퍼레이터가 보안 정책 관리기에 접속하여 번들에 대한 퍼미션을 부여하는 화면이며, 그림 13은

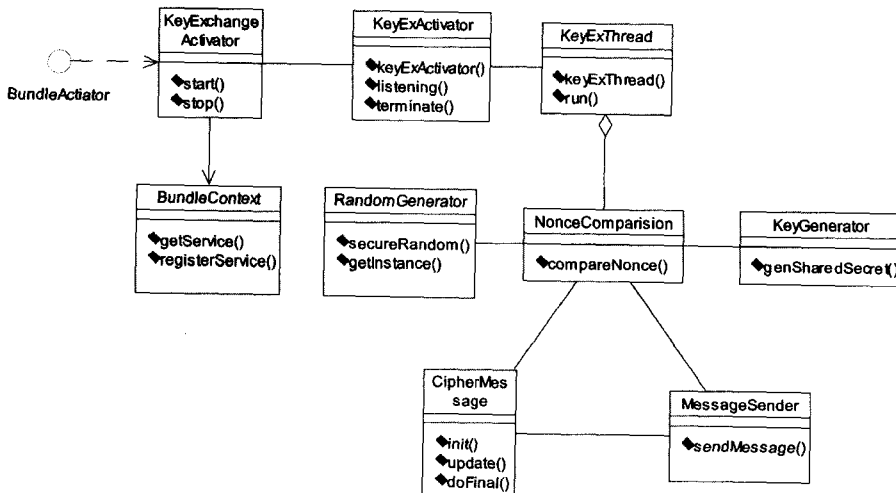


그림 10 키 공유 서비스의 클래스 다이어그램

비스(web service)[21]의 보안에도 응용이 가능하다. 현재 PKI를 기반으로 하는 이동 컴포넌트 환경에 대한 보안 아키텍처로 일반화하기 위한 연구를 진행 중이다. 또한 RBAC 모델[22]을 기반으로 하는 사용자의 권한부여에 대한 연구와 SPKI를 기반으로 하는 분산형 권한부여[12]에 대한 연구도 필요하다. 궁극적으로는 구현된 보안 시스템을 오퍼레이터의 OSGi 서비스 관리 시스템과 통합하여 게이트웨이에서 수행되는 서비스 번들의 신뢰성(reliability)을 높이기 위한 방향으로 연구가 진행될 것이다.

참 고 문 헌

- [1] OSGi, "OSGi Service Platform - Release 3," <http://www.osgi.org>, 2003.3.
- [2] J. Clark and J. Jacob, "A Survey of Authentication Protocol Literature: Version 1.0," University of York, Department of Computer Science, 1997.11.
- [3] C. Neuman and T. Ts'o, "Kerberos: An Authentication Service for Computer Network," IEEE, Computer Magazine, 32(9), pp.33-38, 1994.9.
- [4] Sun Microsystems, "The Java Tutorial - Signing JAR Files," <http://java.sun.com/docs/books/tutorial/jar/sign/signing.html>, 2002.
- [5] OSGi, "RFC 36 - Secure Provisioning Data Transport using HTTP," <http://www.osgi.org/>, 2002.
- [6] H. Krawczyk et al., "IETF RFC 2104 - HMAC: Keyed-Hashing for Message Authentication," <http://www.apps.ietf.org/rfc/rfc2104.html>, 1997.2.
- [7] W. Diffie and M. Hellman, "New Directions in Cryptography," Proc. of the AFIPS National Computer Conference, 1976.6.
- [8] R. Merkle, "Secrecy, Authentication, and Public Key Systems. Ph.D. Thesis," Stanford University, 1979.6.
- [9] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", Communications of the ACM, 1978.12.
- [10] L. Kassab et al., "Towards Formalizing the Java Security Architecture of JDK 1.2," Proc. of the ERORICS'98, Leuven-la-Neuve, Belgium, 1998.9.
- [11] M. Hauswirth et al., "A Secure Execution Framework for Java", Proc. of the 7th ACM conference on computer and communications security (CCS 2000), pp. 43-52, Athens, Greece, 2000.11.
- [12] P. Nikander et al., "Distributed Policy Management for JDK 1.2," Proc. of the 1999 Network and Distributed Systems Security Symposium, pp. 91-102, San Diego, CA, 1999.2.
- [13] G. Karjoth et al., "A Security Model for Aglets," IEEE Internet Computing, 1(4), 1997.7.
- [14] C. Lai and L. Gong, "User Authentication and Authorization in the Java Platform," Proc. of the Computer Security Applications Conference, 1999.12.
- [15] D. Harkins and D. Carrel, "RFC 2409-The Internet Key Exchange (IKE)," 1998.11. <http://www.faqs.org/rfcs/rfc2409.html>
- [16] S. Jajodia et al., "A Logical Language for Expressing Authorization," Proc. of the IEEE Symposium on Security and Privacy, Oakland, CA, 1997.5.
- [17] Sun Microsystems, "Java Embedded Server 2.0," <http://www.sun.com/software/embeddedserver/index.html>
- [18] Ericsson, "Ericsson's E-box System - An Electronic Services Enabler", http://www.ericsson.com/about/publications/review/1999_01/files/1999015.pdf
- [19] Security Technologies Inc., "Java Cryptography Library - J/LOCK", <http://www.stitec.com/product/ejlock.html>
- [20] M. Pistoia, et al., "Java 2 Network Security," Second edition, Prentice Hall, 1999.
- [21] B. Galbraith, et al., "Professional Web Services Security," Wrox Press, 2002.
- [22] R. Sandhu, et al., "Role-Based Access Control Model," IEEE Computer, 29(2), pp.38-47, 1996.2.



박 대 하

1992년 고려대학교 컴퓨터학과 학사
1994년 고려대학교 일반대학원 컴퓨터학과 석사. 1996년 고려대학교 일반대학원 컴퓨터학과 박사과정 수료. 1999년~2003년 (주)시큐리티테크놀로지스 연구소장. 2003년~현재 한국디지털대학교 디지털정보학과 교수. 관심분야는 XML 보안, 보안 프로토콜, 이동코드 보안, 임베디드 시스템 보안



김 영 갑

2001년 고려대학교 식량자원학과 학사 (컴퓨터학과 부전공). 2003년 고려대학교 일반대학원 컴퓨터학과 석사. 2003년~현재 고려대학교 일반대학원 컴퓨터학과 박사과정. 관심분야는 보안 프로토콜, 이동코드 보안, 보안 명세

문 창 주

1997년 고려대학교 컴퓨터학과 학사. 1999년 고려대학교 일반대학원 컴퓨터학과 석사. 2004년 고려대학교 일반대학원 컴퓨터학과 박사. 2004년~현재 고려대학교 정보보호대학원 연구교수. 관심분야는 컴포넌트, 보안공학, 분산보안, RBAC

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 한국시물레이션학회 이사/부회장/회장. 2002년~2003년 고려대학교 정보통신대학 학장. 2002년~현재 과학기술정보표준위원회 위원장. 관심분야는 정보보호, 데이터베이스, 소프트웨어공학, 데이터공학, 컴포넌트 기반 시스템, 메타데이터 레지스트리, 정보통합