OSGi 서비스 플랫폼 환경에서 서비스 번들 인증 메커니즘의 검증 및 구현

(Verification and Implementation of a Service Bundle Authentication Mechanism in the OSGi Service Platform Environment)

김 영 갑 † 문 창 주 †† 박 대 하 ††† 백 두 권 ††††
(Young-Gab Kim) (Chang-Joo Moon) (Dae-Ha Park) (Doo-Kwon Baik)

요 약 OSGi 서비스 프레임워크는 다음과 같은 몇 가지 특성을 갖는다. 첫째, 서비스가 번들이라는 자체 설치 가능한 컴포넌트 형태로 제공되어 동적으로 배치된다. 둘째, 서비스가 생명주기에 따라 동적이며 다른 서비스와의 상호 작용이 자주 일어난다. 셋째, 홈 게이트웨이의 시스템 자원이 충분하지 않다. 이러한 특성들로 인하여 네트워크 상에서 인증되지 않은 오퍼레이터에 의해 악의적인 서비스가 배치될 수 있거나 서비스가 변질 될 수 있다. 이러한 변경된 서비스 번들은 서비스 게이트웨이는 물론 사용자에게 보안상의 악영향을 준다. 더불어 현재 OSGi 프레임워크 표준안에는 위의 특성을 고려한 서비스 번들 인증 메커니즘이 제시되어 있지 않다.

따라서 본 논문은 서비스 플랫폼 상에서의 특성을 고려한 서비스 번들 인증 메커니즘을 제안하고자 한다. 본 논문에서는 서비스 번들의 안전한 전송을 위해서 장치를 인식하고 초기설정을 하는 초기화 작업인 부트스트래핑 단계에서 키 공유를 위한 메커니즘을 설계하였고, 이 단계에서 생성된 공유 비밀키를 이용한 MAC 기반 서비스 번들 인증 메커니즘을 제시하였다. 또한 BAN Logic을 이용해 키 공유 메커니즘과 서비스 번들 인증 메커니즘의 안전성을 검증하였다. 이와 같은 서비스 번들 인증 메커니즘은 기존의 PKI 기반의 서비스 번들 인증이나 OSGi에서 권고하고 있는 RSH 프로토콜과 비교해 볼 때 저장 공간이나 연산의 제약된 자원을 가지고 있는 서비스 플랫폼 상에서 더 효율적이다.

키워드 : 인증, 키 공유, 번들 , 메시지인증코드(Message Authentication Code, MAC), BAN Logic, Signed JAR, OSGi(Open Service Gateway initiative)

Abstract The OSGi service platform has several characteristics as in the followings. First, the service is deployed in the form of self-installable component called service bundle. Second, the service is dynamic according to its life-cycle and has interactions with other services. Third, the system resources of a home gateway are restricted. Due to these characteristics of a home gateway, there are a lot of rooms for malicious services can be installed, and further, the nature of service can be changed. It is possible for those service bundles to influence badly on service gateways and users. However, there is no service bundle authentication mechanism considering those characteristics for the home gateway

In this paper, we propose a service bundle authentication mechanism considering those characteristics for the home gateway environment. We design the mechanism for sharing a key which transports a service bundle safely in bootstrapping step that recognize and initialize equipments. And we propose the service bundle authentication mechanism based on MAC that use a shared secret created in bootstrapping step. Also we verify the safety of key sharing mechanism and service bundle authentication mechanism using a BAN Logic. This service bundle authentication mechanism is more efficient than PKI-based service bundle authentication mechanism or RSH protocol in the service

· 고려대학교 특별연구비에 의하여 수행 되었음
† 학생회원 : 고려대학교 컴퓨터학과
ygkim@software.korea.ac.kr
†† 비 회 원 : 고려대학교 컴퓨터학과
mcj@software.korea.ac.kr
††† 비 회 원 : (주)시큐리티테크놀로지스 연구원
†††† 종신회원 : 고려대학교 정보통신대학 학장
baik@software.korea.ac.kr
논문접수 : 2003년 4월 17일
심사완료 : 2003년 9월 1일

platform which has restricted resources such as storage spaces and operations.
Key words : Authentication, Key Exchange, Bundle, MAC(Message Authentication Code), BAN Logic, Signed JAR, OSGi(Open Service Gateway initiative)

1. 서 론

현재 OSGi(open service gateway initiative)는 서비스 제공업체, 망 관리업체, 시스템 개발업체 및 정보가전 기기 업체 간의 상호 운용성을 보장하는 프레임워크를 정의하고 있다. OSGi 서비스 프레임워크는 자바 프로그래밍 언어의 플랫폼 독립성과 동적 코드 로딩 능력을 이용하여 소형 메모리 디바이스에 적합한 응용프로그램을 쉽게 개발하고 동적으로 배치할 수 있도록 한다. 또한, 번들(bundle)이라는 자체 설치(self installable)가 가능한 컴포넌트 형태로 응용프로그램을 분할할 수 있도록 생명주기(life-cycle) 관리가 가능하여 전체 시스템에 영향을 주지 않고 새로운 서비스를 추가하거나 갱신할 수 있도록 해준다.

기존의 네트워크 서비스는 서비스를 제공하는 서버와 제공받는 클라이언트 사이의 정적인 환경에서 컴포넌트 형태로 제공된다. 이와는 달리 OSGi 프레임워크 환경에서의 서비스는 게이트웨이 관리자(또는 오퍼레이터, operator, OP)와 번들의 생명주기에 따라서 동적으로 서비스 게이트웨이(Service Gateway, SG)에 배치된다. 또한 다른 서비스 번들과의 상호 작용도 일어난다. 이러한 특성으로 인해 네트워크 상에서 인증되지 않은 오퍼레이터에 의하여 서비스 번들이 불법적으로 변경되거나 서비스 게이트웨이가 공격을 받을 수 있다. 변경된 서비스 번들은 서비스 게이트웨이는 물론 사용자에게 보안상의 악영향을 준다.

현재 OSGi 에서는 PKI(Public Key Infrastructure)[1] 기반 서비스 번들 인증 메커니즘을 이용하고 RSH(Remote communication in a Secure way based on HTTP) 프로토콜[2]을 권고하고 있다. 그러나 PKI 기반 서비스 번들 인증의 경우 RSA 또는 DSA와 같은 공개키 연산뿐만 아니라 외부 인증기관과 연동한 인증서의 유효성 검사도 수행해야 하므로 저장 공간이나 연산이 제한된 자원을 가지고 있는 OSGi 서비스 플랫폼 내에서 작동하는 데는 어려움이 있다. 또한 RSH 프로토콜은 MAC(Message Authentication Code)[3]을 오퍼레이터와 서비스 게이트웨이 사이에 전송되는 모든 데이타를 암호화 또는 인증하는데 사용되므로 번들의 크기에 따라 성능의 저하가 예상된다. 따라서 본 논문에서는 이러한 한계점을 극복하기 위하여 MAC 기반 서비스 번들 인증 메커니즘을 제시한다.

MAC 기반 서비스 번들 인증에 앞서 오퍼레이터와 서비스 게이트 간에 상호 인증[4]이 필요하다. 그리고 이 인증을 통해 생성된 공유 비밀키(shared secret)는 MAC을 생성하는 과정에 필요하다. 따라서 본 논문에서는 서비스 번들의 인증을 위해서 장치를 인식하고 초기 설정을 하는 초기화 작업인 OSGi 부트스트래핑(bootstrapping) 단계에서 키 공유를 위한 메커니즘을 설계, 구현한다. 그리고 이 단계에서 생성된 공유 비밀키를 이용한 MAC 기반 서비스 번들 인증 메커니즘을 설계, 구현한다. 또한 BAN Logic을 사용하여 설계한 키 공유 메커니즘과 서비스 번들 인증 메커니즘의 안전성을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 OSGi 서비스 프레임워크에 대한 설명과 함께, OSGi 보안 구조에 대하여 기술한다. 또한 OSGi에서 제안하고 있는 안전한 데이타 전송을 위한 프로토콜에 소개한다. 3장에서는 서비스 번들 인증을 위한 키 공유 메커니즘을 제시하고 4장에서는 OSGi 프레임워크 환경에서 서비스 번들 인증 메커니즘을 제시한다. 5장에서는 3장, 4장에서 제시한 메커니즘의 정형 명세 및 검증을 하고 6장에서 메커니즘의 구현 및 평가에 대하여 기술한다. 마지막으로 7장에서 본 연구의 결론 및 향후 연구 과제에 대해 서술한다.

2. OSGi 서비스 플랫폼

이 장에서는 OSGi 서비스 프레임 워크의 구조, OSGi 보안 구조 및 OSGi 단체에서 권고하는 안전한 데이타 전송을 위한 RSH 프로토콜에 대하여 기술한다.

2.1 OSGi 서비스 프레임워크의 구조

OSGi는 홈 네트워크 기술과 독립적으로 가정 내의 홈 네트워크나 정보가전 기기에 접근할 수 있는 개방된 공통구조를 정의함으로써 액세스망을 거쳐서 다중의 서비스를 홈 네트워크와 정보 가전기기에 전달할 수 있는 규격을 만들고 있는 표준화 단체이다. 1999년 3월에 창립되어 초기 15개의 업체로 시작하여 2002년 말 현재 50 여개의 소프트웨어, 하드웨어, 서비스 제공 업체가 참여하고 있다. 우리나라에서는 삼성전자, ETRI(Electronics and Telecommunications Research Institute), Connected Systems, 4DHomeNet 등 4개 업체가 참여하고 있다. OSGi는 2001년 10월, 공식으로 릴리즈

(release) 2.0[5]을 발표하였다. 이 스펙은 거의 모든 가정용 네트워킹 표준을 지원하도록 설계되었으며, 서비스들이 홈 게이트웨이나 또는 PC 같은 다른 장치들에서 실행될 수 있게 해준다. 이 표준은 자바 기반으로 Bluetooth, CAL, CEBus, Convergence, emNET, HAVi, HomePNA, HomePlug, HomeRF, 그리고 Jini 기술과 같이 주택에 초점을 둔 여러 유선 및 무선 표준들과 프로토콜들을 지원한다.

OSGi 서비스 플랫폼의 전체적인 구조는 그림 1과 같다.

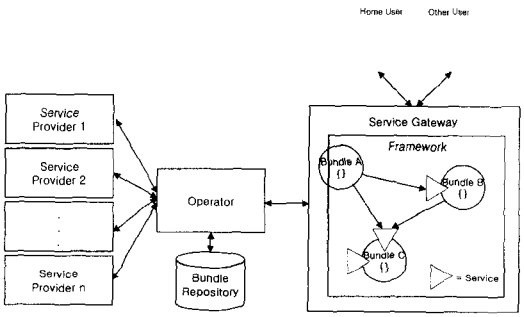


그림 1 OSGi 서비스 플랫폼 구조

서비스 제공자(service provider)는 가정에 에너지 관리 서비스를 제공하는 전력회사와 같은 서비스 개발 전문 업체를 의미하고 번들 서비스를 제작, 제공하는 역할을 한다. 오퍼레이터는 독립적인 서비스 업체의 형태로 존재하면서 각 가정의 서비스 게이트웨이와 서비스를 관리하는 관리 전문 업체를 의미한다. 그리고 서비스 게이트웨이는 각 가정에 설치되어 있는 홈 게이트웨이를 의미한다. 사용자는 서비스 게이트웨이를 사용하는 주체로서 다양한 정보 단말과 유무선 인터넷을 사용하여 집 외부에서 서비스 게이트웨이를 경유하여 집안의 홈네트워크나 정보가전 기기에 접근할 수 있다.

프레임워크의 핵심 개체는 다음과 같다. 특정 기능을 수행하는 자바 인터페이스와 실제 구현 객체인 서비스(service), 서비스를 제공하기 위한 기능적 배치단위인 번들(bundle), 프레임워크 내에서 번들들의 실행 환경인 번들 문맥(bundle context)로 구성되어 있다. 서비스는 미리 정의된 서비스 인터페이스를 통해 접근이 가능한 독립적인 컴포넌트다. 하나의 응용프로그램은 여러 개의 서비스의 협동작업을 통해 구성되고, 실행시(runtime)에 필요한 서비스를 요청할 수도 있다. 프레임워크는 각 서비스와 그 서비스에 해당하는 실제구현에 대한 매핑을 가지고 있고, 각 서비스간의 상호 의존관계를 관리한다. 번들은 여러 서비스의 구현을 하나의 패키지로 묶은 JAR 파일의 형태로 존재하며 프레임워크 내부에서 설치되거나 활성화 될 때 하나의 기능적인 컴포넌트를 나타낸다. 프레임워크에 설치된 각각의 번들에 대해 하나의 번들 객체가 존재하는데 이 객체는 번들의 자바 클래스에 대한 명칭 공간을 관리하는데 사용되며 클래스 로딩과 분석을 지원한다. 번들 문맥은 하나의 번들에 대한 실행 환경이며 번들이 활성화 될 때 프레임워크에 의해 생성된다. 새로운 번들을 프레임워크에 설치하거나, 프레임워크 등록기(registry) 내에 서비스를 등록하기 위해 사용된다.

2.2 OSGi 보안 구조

OSGi가 제안하는 보안 모델[6]은 공개키 기반구조(PKI) 솔루션에 기반을 두고 있다. 인증기관(CA)은 모든 OSGi 개체에 인증서를 전달하는 책임을 지게 되며 인증서와 연관된 키는 인증, 무결성, 기밀성에 사용된다.

부트스트래핑 단계에서 번들을 다운로드하고 설치하기 이전에 오퍼레이터는 특정 서비스 게이트웨이를 인증해야 한다. 그러나 OSGi 프레임워크에서의 부트스트랩 프로세스는 현재 다른 작업그룹에서 개발 중이며 아직 최종적으로 결정되지 않은 상태이다.

위에서 언급하였듯이 OSGi 서비스 플랫폼 릴리즈 2에서는 서비스 프레임워크에 대한 기본적인 보안 모델 방향만 제시되어 있을 뿐 부트스트래핑 단계에서의 오퍼레이터와 서비스 게이트웨이의 상호 인증 및 서비스 번들 인증에 대한 구체적인 방안은 제시되지 않았다. 따라서 본 논문에서는 3장과 4장에서 이러한 메커니즘을 제시한다.

2.3 안전한 데이타 전송 프로토콜

OSGi에서는 서비스 플랫폼에 안전한 설치 데이타 전송(secure provisioning data transport)을 수행하는 원격 통신 기법을 제안하고 있다. 설치 데이타 전송을 위해서 HTTP(HyperText Transfer Protocol)[7], HTTPS (Hypertext Transfer Protocol Secure)[7], RSH (Remote communication in a Secure way based on HTTP) 프로토콜 중에 하나를 사용할 수 있다. RSH 프로토콜의 기본 아이디어는 서비스 게이트웨이와 오퍼레이터 사이에 공유한 대칭키로부터 유도된 MAC을 인증과 암호화의 기반으로 삼자는 것이다. 이 MAC은 서버와 클라이언트 사이에 이동하는 모든 데이타에 대하여 암호화와 인증을 수행한다. 그러나 이러한 과정은 이동되는 데이타의 크기가 클 경우 MAC값을 생성하는 시간이 오래 걸릴 뿐만 아니라 암호화 또는 인증이 필요 없는 데이타에 대해서도 작업이 수행되므로 많은 시스템 자원을 소모하게 된다. 그리고 오퍼레이터와 서비스 게이트웨이 사이에 대칭키를 공유하는 방법에 대해서는 제시되지 않고 있다. 본 논문에서 제시하는 MAC 기반 서비스 번들 인증은 이 프로토콜의 개념을 OSGi 서비스

플랫폼의 환경에 맞추어 RSH 프로토콜이 가지고 있는 단점을 변형, 개선하여 새로운 서비스 번들 인증 메커니즘을 제시한다.

OSGi에서는 제안하는 추상적인 RSH 프로토콜을 나타내면 그림 2와 같다.

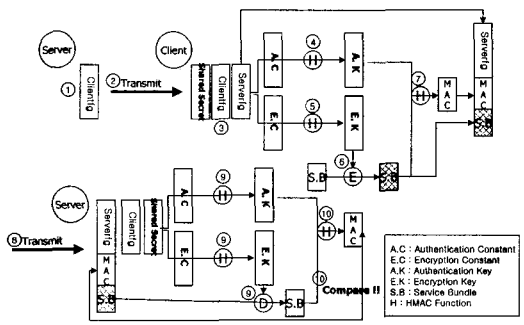


그림 2 RSH 프로토콜

우선 클라이언트는 clientfg라는 하나의 난수(nonce)를 생성하여 임의의 위치에 저장한 후 clientfg를 서버로 전송 한다. 서버는 serverfg라는 하나의 난수를 생성하고 HMAC(hashed message authentication code)[8] 함수와 공유 비밀키, 수신한 clientfg, serverfg 및 인증 상수(Authentication Constant)를 기반으로 인증 키를 계산한다. 또한 암호화 상수(Encryption Constant)를 기반으로 암호화 키를 생성한다. 그리고 앞 단계에서 유도된 암호화 키를 사용하여 응답 데이타를 암호화 하고 HMAC 함수와 암호화된 응답 데이타 및 유도된 인증 키를 사용하여 MAC을 계산한다. 마지막으로 서버는 serverfg와 MAC 및 암호화된 응답 데이타를 포함하는 응답을 클라이언트에 전송한다. 클라이언트는 서버와 동일한 방식으로 암호화 키를 계산하고 암호화된 응답 데이타를 복호화 하는데 사용한다. 계산에는 수신한 응답 내부의 serverfg 값이 사용된다. 서버와 동일한 방식으로 MAC의 계산을 수행하고 수신한 MAC과 일치하는지 검사한다. 만약 일치하지 않으면 더 이상의 처리는 이루어지지 않는다. 계산에는 수신한 응답 내부의 serverfg 값이 사용된다.

이 프로토콜과 서비스 번들 인증 메커니즘의 비교는 6장에서 비교 평가한다.

3. OSGi 플랫폼 환경에서의 키 공유 메커니즘

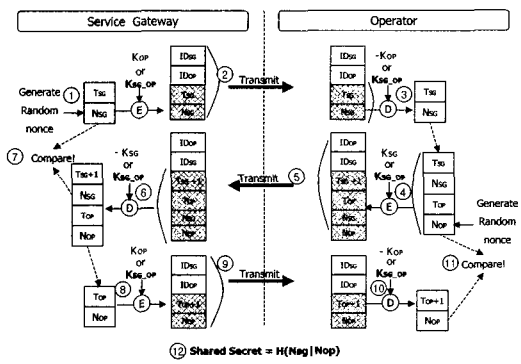
서비스 게이트웨이는 사용자에게 서비스를 제공하기에 전에 오퍼레이터와의 상호 인증을 필요로 한다. 이 장에서는 OSGi 서비스 게이트웨이의 부트스트래핑 단계에서 서비스 게이트웨이와 오퍼레이터의 상호 인증을 통하여 서비스 번들 인증시 필요한 공유 비밀키를 생성하는 키 공유 메커니즘을 제시하였다.

3.1 대칭키를 이용한 키 공유 메커니즘

키 공유 메커니즘은 오퍼레이터와 서비스 게이트 사이에 안전한 공유 비밀키를 공유하여 사용자 인증 티켓의 암호화 또는 서비스 번들의 무결성을 보장하도록 지원하기 위한 방법이다. 키 공유 메커니즘은 서비스 게이트웨이의 부트스트래핑 직후에 공유 비밀키를 생성하고 보안 정보의 요청이 있을 때마다 새롭게 갱신되어 공격자의 키 탐색 공격에 대한 위험을 최소화하도록 한다. 그림 3은 이 논문에서 제시하는 부트스트래핑 단계에서 공유 비밀키를 생성하는 키 공유 메커니즘이다[9]. 서비스 게이트웨이와 오퍼레이터에서 생성한 각각의 난수 값(N_{SG}, N_{OP})을 공개키(K_{SG}, K_{OP}) 또는 대칭키(K_{SG_OP})를 이용하여 암호, 복호화 한다. 이 과정을 통해 상호 인증을 하고 공유 비밀키를 생성함을 보여주고 있다. 대칭키를 이용한 키 공유 메커니즘은 오퍼레이터와 서비스 게이트 사이의 상호 인증과 동시에 공유 비밀키를 생성하기 위하여 오퍼레이터와 게이트웨이에서 생성된 난수 값(N_{SG}, N_{OP})과 타임스탬프를 암호, 복호화 하는데 동일한 키(대칭키)를 이용한 방법이다.

그림 3에서 제시한 대칭키를 이용한 키 공유 메커니즘을 자세히 설명하면 다음과 같다.

① SG의 난수 발생기(Random generator)에 의해 난수 값 N_{SG}와 타임스탬프 T_{SG}를 생성한다. SG에서 생성



ID_{SG} : SG의 id

ID_{OP} : OP의 id

T_{SG}, T_{OP} : SG, OP에서 생성한 타임스탬프(Timestamp)

N_{SG} : SG가 생성한 난수 값(random nonce)

N_{OP} : OP가 생성한 난수 값

K_{OP}, $-K_{OP}$: OP의 공개키, 개인키

K_{SG}, $-K_{SG}$: SG의 공개키, 개인키

K_{SG_OP} : SG, OP가 공유하고 있는 대칭키

그림 3 부트스트래핑 단계에서의 키 교환 메커니즘

한 N_{SG}는 부팅 시마다 생성되며 항상 새롭다.

② 생성된 N_{SG},T_{SG}를 SG와 OP의 대칭키(K_{SG_OP})로 암호화하고 ID_{SG}, ID_{OP}와 함께 OP로 전송한다. 본 논문의 구현에서는 ID_{SG}로서 SG에 등록된 시리얼 넘버(serial number)를 이용하였고 ID_{OP}는 OP의 IP 주소를 사용하였다.

③ OP에서는 대칭키로 N_{SG}, T_{SG}를 복호화 한 후 난수 발생기에 의해 생성된 난수 값 N_{OP}와 타임스탬프 T_{OP}를 생성한다.

④ OP는 복호화한 N_{SG}, T_{SG}와 OP에서 생성한 N_{OP}, T_{OP}를 함께 대칭키로 암호화한다.

⑤ OP는 암호화된 {T_{SG}, T_{OP}, N_{SG}, N_{OP}}를 ID_{OP}, ID_{SG}와 함께 SG로 전송한다.

⑥ SG는 전송 받은 {T_{SG}, T_{OP}, N_{SG}, N_{OP}}를 대칭키로 복호화한다.

⑦ SG는 OP로 전달된 T_{SG}가 자신이 보낸 최근의 타임스탬프인지 확인하고 또한 OP로 전송한 N_{SG}와 전송받은 N_{SG}와 같은지 비교하여 일치하면 SG는 OP를 인증하게 되고 다음 작업(⑧)이 수행된다. 만약 일치하지 않으면 더 이상의 처리는 이루어지지 않는다.

⑧ SG는 전송 받은 T_{OP}, N_{OP}를 대칭키로 암호화한다.

⑨ SG는 암호화한 T_{OP}, N_{OP}와 ID_{SG}, ID_{OP}를 함께 OP로 전송한다.

⑩ OP는 전송 받은 T_{OP}, N_{OP}를 대칭키로 복호화한다.

⑪ OP는 SG로부터 전송 받은 T_{OP}가 최근의 타임스탬프인지 확인하고 SG로 전송한 N_{OP}를 자신의 N_{OP}와 비교한다. 일치하면 OP는 SG를 인증하게 되고 결국 상호 인증이 이루어진다. 만약 일치하지 않으면 더 이상의 처리는 이루어지지 않는다.

⑫ OP, SG가 서로 인증되면 N_{SG}, N_{OP}에 기반한 해쉬 함수에 의해 공유 비밀키(shared secret)가 생성되고 저장된다. 여기서 생성된 공유 비밀키는 서비스 번들 인증시 MAC 값을 생성하는데 사용된다.

⑦, ⑪ 단계를 통하여 오퍼레이터와 서비스 게이트웨이 사이에 상호 인증이 일어남을 알 수 있고 ⑫를 통하여 둘 사이의 공유 비밀키가 생성됨을 알 수 있다.

3.2 공개키를 이용한 키 공유 메커니즘

공개키를 이용한 키 교환은 대칭키를 이용한 키 교환과 거의 비슷하다. 차이점은 SG와 OP에서 생성하는 난수 값(N_{SG}, N_{OP})를 암호, 복호화 할 때 서비스 게이트웨이와 오퍼레이터가 각각의 공개키(K_{SG}, K_{OP}) 개인키($-K_{SG}$, $-K_{OP}$)를 가지고 한다. 대칭키와 공개키를 이용한 키 공유 메커니즘의 비교는 6장 비교 평가에서 한다.

4. OSGi 플랫폼 환경에서의 서비스 번들 인증 메커니즘

서비스 번들 인증 메커니즘의 설명에 앞서 본 논문에서 제안하는 OSGi 플랫폼 환경에서의 서비스 번들의 인증과 보호 도메인(protection domain) 할당에 대한 전체적인 구조를 보면 그림 4와 같다[9]. 서비스 제공자는 제공하는 서명된 서비스 번들(signed JAR 파일)[10]을 오퍼레이터에게 제공(①,②)하며 이 때 오퍼레이터는 PKI 기반으로 받은 서명 파일을 검증(③)하게 된다. 검증을 통해 확인된 서비스 번들은 서명 번들 저장소(signed bundle repository)에 저장을 하게 되며 검증에 실패한 번들은 임시 번들 저장소(temp bundle repository)에 저장(④)하게 된다.

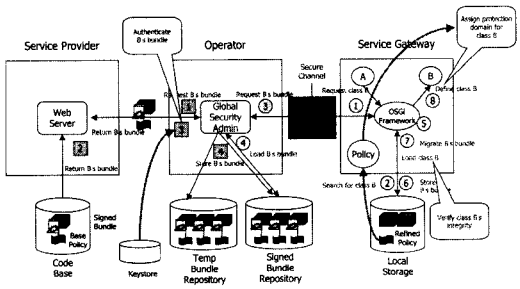


그림 4 OSGi 플랫폼 환경에서의 서비스 번들의 인증과 보호 도메인 할당

사용자가 서비스 요청시(①) 서비스 게이트웨이는 우선 서비스 플랫폼에 설치되어 있는지를 확인(②)한다. 설치되어 있지 않을 경우 사용자의 요청을 오퍼레이터에 요청(③)을 한다. 오퍼레이터의 서명 번들 저장소에 서비스 번들이 있을 시에 동적으로 서비스 번들을 이동하고 서비스 플랫폼에 저장(④~⑥)한다. 이 때 오퍼레이터와 서비스 게이트웨이는 MAC 값을 기반으로 안전한 전송 채널을 형성하고 서비스 번들을 전송하게 된다. 저장된 서비스 번들로부터 서비스(실제적으로 클래스)가 실행될 때 자바 클래스 로더(java class loader)에 의해 클래스의 무결성을 검사한 후 실행(⑥,⑦)된다. 또한 정책 파일에 의하여 보호 도메인이 생성(⑧)된다.

오퍼레이터와 서비스 게이트웨이 사이의 서비스 번들의 제공은 서명된 JAR 파일 형태로 이루어진다. 따라서 서비스 번들의 인증은 서명된 JAR 파일에 대한 서명을 검증하여 이루어진다. 일반적인 JAR 서명 코드는 개발자의 개인키를 이용하여 작성되고 개발자의 공개키 인증서와 PKI를 기반으로 검증하게 된다. 그러나 PKI 기반의 인증은 공개키 연산뿐만 아니라 인증서의 유효성 검증을 위해 인증기관과의 연동을 필요로 하기 때문에 저장 공간이나 연산 등의 제한된 시스템 자원을 가지고 있는 서비스 플랫폼 상에서는 부적합하다. 이러한 점을

고려해 볼 때 서비스 플랫폼에 설치되는 서비스 번들은 오퍼레이터와 각각의 서비스 플랫폼 간에 공유하고 있는 대칭키 형식의 인증키(MAC 키)를 사용하여 MAC 기반의 인증을 수행하는 것이 효율적이다. 따라서 본 논문에서는 서비스 게이트웨이가 오퍼레이터로부터 서비스 번들을 검증한 무결성이 유지된 인증 값 즉, MAC을 받아서 인증을 수행하도록 하였다.

먼저 서비스 플랫폼과 오퍼레이터 사이에 효율적인 인증을 수행하기 위해 사전에 대칭키를 공유해야 한다. 이를 공유하기 위하여 3장에서 제시한 바와 같이 서비스 플랫폼의 부트스트래핑 직후에 서비스 플랫폼과 오퍼레이터 사이에 미리 등록된 시스템 키를 이용하여 상호 인증을 수행하고 공유 대칭키(공유 비밀키)를 공유하는 메커니즘을 이용한다.

그림 5는 MAC 기반 서비스 번들 인증 메커니즘을 나타낸다[9].

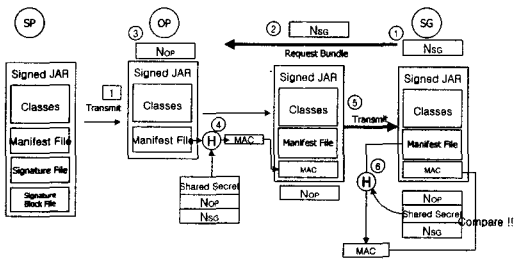


그림 5 MAC 기반 서비스 번들 인증

MAC 기반 서비스 번들 인증은 부트스트래핑 단계에서 생성된 공유 비밀키로부터 유도된 MAC을 인증의 기반으로 삼는다. MAC은 HMAC이라고 불리는 암호화 해쉬 함수에 기반을 두고 있다. SUN에서 제공하는 HMAC는 HmacMD5 알고리즘과 HmacSHA1 알고리즘을 사용할 수 있는데 본 논문의 구현 부분에서는 HmacSHA1 알고리즘을 이용한 HMAC 함수를 사용하였다. HMAC 계산의 입력으로 서명된 JAR 파일과 서비스 게이트웨이와 오퍼레이터에서의 난수 값인 N_{SG}, N_{OP}와 부트스트래핑 단계에서 생성된 공유 비밀키를 사용한다.

그림 5의 ①에서 보듯이 서비스 제공자가 오퍼레이터에 서비스를 등록할 때 오퍼레이터는 인증된 서비스 번들에 대하여 서명파일(signature file)과 서명 블록 파일(signature block file)을 제외시킨다. 그러므로 서비스 번들의 크기가 작아지며 오퍼레이터에서 서비스 게이트웨이로 이동시 전송 속도를 향상 할 수 있다.

그림 5에 표현한 MAC 기반 서비스 번들 인증 과정을 자세히 살펴보면 다음과 같다.

① SG는 N_{SG}라는 하나의 난수 값(nonce)을 생성하여 저장한다. 이 때 생성한 N_{SG}는 서비스를 요청시마다 생성되며 새롭다. 그러므로 보안의 안정성을 높을 수 있다.

② SG는 N_{SG}를 OP에 전송한다.

③ OP는 N_{OP}라는 하나의 난수를 생성하여 저장한다. SG와 마찬가지로 생성할 때마다 새로운 키가 생성된다.

④ OP는 HMAC 함수와 서명된 JAR 파일, 공유 비밀키, N_{SG}, N_{OP}에 기반하여 HMAC을 이용하여 MAC을 계산한다. 해쉬 함수로는 SHA-1이 사용되고 MAC은 다음과 같이 계산된다.

MAC = HMAC-SHA1(signed JAR, shared secret | N_{SG} | N_{OP})

(|는 연결(concatenation)을 뜻한다.)

여기에서 공유 비밀키는 부트스트랩핑 단계에서 키 공유 메커니즘을 통하여 생성된 것이다. MAC 값을 계산한 후 서명된 JAR 파일 안에 MAC 값을 넣게 되며 MAC이 서명파일의 역할을 대체한다.

⑤ OP는 N_{OP}와 MAC을 포함하는 서비스 번들을 SG에 전송한다.

⑥ SG는 OP와 동일한 방식으로 MAC 값을 계산하고 수신한 MAC 값과 일치하는지 검사한다. 일치하면 SG는 OP에서 전달된 서비스 번들을 믿고 사용할 수 있다. 만약 일치하지 않으면 더 이상의 처리는 이루어지지 않는다. MAC 값의 계산에는 수신한 응답 내부의 N_{OP} 값이 사용된다. 실제적인 서비스 번들 인증은 자바 클래스 로더에서 클래스를 로딩할 때 일어난다.

①~⑥의 과정을 통하여 오퍼레이터에서 생성한 MAC과 서비스 게이트웨이에서 생성한 MAC값을 비교함으로써 제공된 서비스 번들이 변경되었는가를 알 수 있다. 이러한 MAC은 서비스 요청시 마다 새롭게 생성 되므로 보안을 높일 수 있다.

5. 키 공유 및 서비스 번들 인증 메커니즘의 정형 명세 및 검증

인증 프로토콜의 기술을 위해 전통적인 보안 명세에서는 주체 사이에 단순히 메시지만을 나열하거나, 상징적으로 보내는 사람, 받는 사람, 메시지의 내용들을 보여주는 것에 치중한다. 그러나 이러한 명세는 로직(loigc)에서 조작하기 힘들 뿐 아니라 메시지에 포함된 내용으로부터 명백한 의미를 얻어낼 수 없다. 따라서 보안 정형 명세를 통해 프로토콜의 정확한 의미뿐만 아니라 주고받는 메시지의 의미를 명확히 나타낼 필요가 있

다. BAN Logic[11-14]은 분산 컴퓨팅 시스템 환경 하에서 주체(principal)들의 인증에 관한 프로토콜의 검증과 분석을 위하여 만들어진 로직이다. 따라서 본 논문에서 제시하는 서비스 번들 인증 메커니즘을 정형 명세하고 검증을 위하여 BAN Logic을 이용한다.

5.1 BAN Logic 기반의 보안 명세

BAN Logic은 크게 네 부분의 명세 사항으로 나누어진다. 일반적인 프로토콜 명세(usual notation), BAN 표기법을 이용한 이상화된 프로토콜(idealized protocol), 이러한 이상화된 프로토콜을 나타내기 위한 가정(assumption), 그리고 프로토콜이 나타내려고 하는 궁극적인 목적(goal)으로 구성된다. BAN Logic을 사용하여 프로토콜 검증하기 위해서는 우선 원래의 프로토콜 명세로부터 BAN 표기법을 이용하여 이상화된 프로토콜을 명세해야 한다. 그 다음에 처음 상태에 관한 가정들을 기술한 다음 프로토콜의 문장에 논리적인 의미를 덧붙인다. 그리고 BAN 규칙들을 이용하여 검증한다.

BAN의 기본적인 표기법은 표 1과 같다.

표 1 BAN 기본 표기법

표 기	설 명
P, Q and R	특정 principals
X and Y	statement(formula)
K	Encryption Key
K_p, K_q, K_r	특정 principal에 대한 공개키
K_{p-1}, K_{q-1}, K_{r-1}	공개키에 대응되는 개인키
N_p, N_q	특정 statement
,(comma)	그리고 (and)
$P \mid\equiv X$	P believes X
$P \triangleleft X$	P Sees X
$P \mid\sim X$	P once said X
#(P)	X is fresh
$P \mid\Rightarrow X$	P has jurisdiction over X
$P \overset{K}{\leftrightarrow} Q$	P and Q may use the shared key K to communicate
$\overset{K}{\mapsto} P$	P has K as a public key
$P \overset{X}{\rightleftharpoons} Q$	X is a secret known only to P and Q
$\{X\}_K$	Statement X encrypted under the key K
$\langle X \rangle_Y$	X combined with (secret) Y

이상화된 프로토콜을 검증하기 위해서는 표 2와 같은 여러 가지 규칙들이 적용된다.

본 논문에서 제시하는 키 공유 메커니즘의 안전성을 검증하기 위해서는 BAN Logic의 기본적인 규칙 이외에 새로운 규칙이 필요하다. 따라서 규칙 17을 새로 생성하였다.

5.2 대칭키를 이용한 키 공유 메커니즘의 정형 명세 및 검증

그림 3의 키 공유 메커니즘을 앞서 설명한 프로토콜 검증 단계에 의해 일반적인 보안 명세 프로토콜을 명세하면 표 3과 같다. 여기에서 메시지 1은 3.1절에서 설명한 ①,② 과정에 해당하며, 메시지 2는 ③~⑤ 과정에 해당한다. 메시지 3은 ⑥~⑨ 과정에 해당한다.

표 3 대칭키를 이용한 키 공유 메커니즘의 일반적인 보안 명세

▫ Usual Notation

메시지 1 $SG \rightarrow OP : ID_{SG}, ID_{OP}, \{T_{SG}, N_{SG}\}K_{SG\_OP}$

메시지 2 $OP \rightarrow SG : ID_{SG}, ID_{OP}, \{T_{SG}+1, T_{OP}, N_{SG}, N_{OP}\}K_{SG\_OP}$

메시지 3 $SG \rightarrow OP : ID_{SG}, ID_{OP}, \{T_{OP}+1, N_{OP}\}K_{SG\_OP}$

일반적인 보안 명세 프로토콜에 대한 설명은 3.1절을 참고하면 된다. BAN Logic을 통하여 각각의 메시지들을 이상화된 프로토콜로 변환하면 표 4와 같다.

표 4 대칭키를 이용한 키 공유 메커니즘의 이상화된 프로토콜 명세

▫ BAN Idealization

메시지 1 $SG \rightarrow OP : \{T_{SG}, N_{SG}\}K_{SG\_OP}$

메시지 2 $OP \rightarrow SG : \{\langle T_{SG}+1, SG \overset{N_{OP}}{\rightleftharpoons} OP \rangle N_{SG}\}K_{SG\_OP}$

메시지 3 $SG \rightarrow OP : \{\langle T_{OP}+1, SG \overset{N_{SG}}{\rightleftharpoons} OP, OP \mid\equiv SG \overset{N_{OP}}{\rightleftharpoons} OP \rangle N_{OP}\}K_{SG\_OP}$

메시지 1에서 SG는 T_{SG}와 N_{SG}를 SG와 OP의 대칭키인 K_{SG_OP}로 암호화하여 OP로 보낸다. 메시지 2에서 OP는 T_{SG}+1과 'N_{OP}는 SG와 OP의 공유 비밀키라는 것'을 N_{SG}로 결합한 후에 대칭키(K_{SG_OP})로 암호화하여 SG로 보낸다. 메시지 3에서 SG는 T_{OP}+1과 'N_{SG}가 SG와 OP의 공유 비밀키라는 것'과 'OP는 N_{OP}가 SG와

표 5 대칭키를 이용한 키 공유 메커니즘의 가정

▫ BAN Assumption

① $SG \mid\equiv SG \overset{K_{SG\_OP}}{\leftrightarrow} OP$, ② $OP \mid\equiv SG \overset{K_{SG\_OP}}{\leftrightarrow} OP$

③ $SG \mid\equiv SG \overset{N_{SG}}{\rightleftharpoons} OP$, ④ $OP \mid\equiv SG \overset{N_{OP}}{\rightleftharpoons} OP$

⑤ $SG \mid\equiv \#(N_{SG})$, ⑥ $OP \mid\equiv \#(N_{OP})$

⑦ $SG \mid\equiv \#(T_{SG})$, ⑧ $OP \mid\equiv \#(T_{SG})$

⑨ $SG \mid\equiv \#(T_{OP})$, ⑩ $OP \mid\equiv \#(T_{OP})$

표 2 BAN Logic 규칙

규 칙		표 기	설 명
Message-meaning rule	1	$\frac{P\mid\equiv \overset{K}{Q\leftrightarrow}P,\ P\triangleleft\{X\}_K}{P\mid\equiv Q\mid\sim X}$ |~ Introduction Rule	P가 'K가 Q와 P의 대칭키라는 것'을 믿고, 키 K로 암호화된 X를 받았다면, P는 'Q가 X를 보냈다는 것'을 믿을 수 있다.
	2	$\frac{P\mid\equiv \overset{K}{\mapsto} Q,\ P\triangleleft\{X\}_{K^{-1}}}{P\mid\equiv Q\mid\sim X}$ |~ Introduction Rule	P는 'K가 Q의 공개키'이고 K의 개인키로 암호화된 X를 받았다면, P는 'Q가 X를 보냈다는 것'을 믿을 수 있다.
	3	$\frac{P\mid\equiv Q\overset{Y}{\rightleftharpoons}P,\ P\triangleleft\langle X\rangle_Y}{P\mid\equiv Q\mid\sim X}$ |~ Introduction Rule	P는 'Y가 Q와 P의 공유 비밀키라는 것'을 믿고, Y로 결합된 X를 받았다면, P는 'Q가 X를 보냈다는 것'을 믿을 수 있다.
Nonce-verification rule	4	$\frac{P\mid\equiv \#(X),\ P\mid\equiv Q\mid\sim X}{P\mid\equiv Q\mid\equiv X}$ |~ Elimination Rule	P가 'X가 새롭다는 것'을 믿고, 'Q가 X를 보냈다는 것'을 믿는다면, P는 'Q가 X를 믿는다는 것'을 믿을 수 있다.
Jurisdiction rule	5	$\frac{P\mid\equiv Q\mid\Rightarrow X,\ P\mid\equiv Q\mid\equiv X}{P\mid\equiv X}$ |⇒ Elimination Rule	P가 'Q가 X를 관할하는 것'을 믿고, 'Q가 X를 믿는 것'을 믿는다면, P는 X를 믿을 수 있다.
Other Rule	6	$\frac{P\mid\equiv X,\ P\mid\equiv Y}{P\mid\equiv Q\mid\equiv (X,Y)}$, Introduction Rule	P가 X를 믿고, 또 P가 Y를 믿는다면, P는 (X,Y) 집합(set)을 믿을 수 있다.
	7	$\frac{P\mid\equiv (X,\ Y)}{P\mid\equiv X}$, Elimination Rule	P가 (x, y) 집합을 믿는다면 P는 x,y 집합의 일부분인 x를 믿을 수 있다.
	8	$\frac{P\mid\equiv Q\mid\equiv (X,Y)}{P\mid\equiv Q\mid\equiv X}$, Elimination Rule	Q가 (x, y) 집합을 믿는 것을 P가 믿는다면 Q가 (x,y) 집합의 일부분인 X를 믿는 것을 P가 믿을 수 있다.
	9	$\frac{P\mid\equiv Q\mid\sim (X,Y)}{P\mid\equiv Q\mid\sim X}$, Elimination Rule	P가 'Q가 (x, y) 집합을 믿는 것' 믿는다면 'Q가 (x,y) 집합의 일부분인 X를 믿는다는 것'을 믿을 수 있다.
	10	$\frac{P\mid\equiv \#(K),\ P\mid\equiv Q\mid\equiv X}{P\mid\equiv \overset{K}{P\leftrightarrow Q}}$ ↔ Introduction Rule	P가 키 'K가 새롭다는 것'을 믿고 'Q가 X를 믿는다는 것'을 믿는다면 P는 'K가 P와 Q 사이의 공유키라는 것'을 믿을 수 있다.
◁ operator usage	11	$\frac{P\triangleleft (X,Y)}{P\triangleleft X}$, Elimination Rule	P가 (x, y)를 받은 것은 'P가 x를 받았다'고 말할 수 있다.
	12	$\frac{P\triangleleft \langle X\rangle_Y}{P\triangleleft X}$ <> Elimination Rule	P가 Y로 결합된 X를 받았다면 'P는 X를 받았다'고 말할 수 있다.
	13	$\frac{P\mid\equiv \overset{K}{Q\leftrightarrow}P,\ P\triangleleft\{X\}_K}{P\triangleleft X}$ ↔ Elimination Rule	P가 'K는 Q와 P사이의 공유키임'을 믿고 K로 암호화된 X를 받았다면 'P는 X를 받았다'고 말할 수 있다.
	14	$\frac{P\mid\equiv \overset{K}{\mapsto} Q,\ P\triangleleft\{X\}_{K^{-1}}}{P\mid\equiv Q\triangleleft X}$ ↦ Elimination Rule	P가 'K는 Q의 공개키'이고 K에 대응하는 개인키로 암호화된 X를 받았다면, 'P는 X를 받았다'고 말할 수 있다.
	15	$\frac{P\mid\equiv \#(X)}{P\mid\equiv \#(X,Y)}$, Introduction Rule	집합의 일부분(x)이 새롭다고 P가 믿는다면, P는 집합 전체(x,y)가 새롭다고 말할 수 있다.
해쉬 함수	16	$\frac{P\mid\equiv Q\mid\sim H(X),\ P\triangleleft X}{P\mid\equiv Q\mid\sim X}$ |~ Introduction Rule	P가 'Q가 X의 해쉬 함수 값을 보낸 것'을 믿고, 메시지 X를 받았다면 P는 'Q가 메시지 X를 보냈다는 것'을 믿을 수 있다.
새로 생성한 규칙	17	$\frac{P\mid\equiv \#(X),\ P\mid\equiv \#(X,\ \overset{K}{A\leftrightarrow B})}{P\mid\equiv \#(\overset{K}{A\leftrightarrow B})}$	P가 'X가 새롭다는 것'을 믿고, X와 'K가 A와 B의 공유키라는 사실'의 집합이 사실이라는 것을 믿는다면 P는 'K가 A와 B의 공유키라는 사실'이 새롭다고 믿을 수 있다.

OP의 공유 비밀키라는 사실'을 믿는 것을 N_{OP}로 결합한 후에 대칭키로 암호화하여 OP로 보낸다.

위에 기술한 이상화된 프로토콜을 검증하기 위해서는 표 5와 같은 가정들을 기술할 수 있다.

SG에서 대칭키의 접근은 기본적으로 내장되어 있는 번들 서비스에서만 접근이 가능하고 OP에서는 외부 방

화벽(firewall)이 존재하고 있다고 전제한다. 그리고 SG와 OP는 랜덤 생성기(random generator)가 있다고 전제한다. 그러므로 ①에서 K_{SG_OP}는 SG와 OP의 대칭키이므로 SG는 'K_{SG_OP}가 SG와 OP의 대칭키라는 것을 믿을 수 있다. 이와 마찬가지로 ②에서 OP는 'K_{SG_OP}가 SG와 OP의 대칭키라는 것'을 믿는다. 또한 SG는 'N_{SG}가 SG와 OP의 공유 비밀키라는 것'을 믿고(③) OP도 'N_{OP}가 SG와 OP의 공유 비밀키라는 것'을 믿는다(④). SG는 자신이 생성한 N_{SG}가 새롭다는 것을 믿고(⑤), OP 역시 자신이 생성한 N_{OP}가 새롭다는 것을 믿는다(⑥). 또한 SG는 T_{SG}, T_{OP}가 새롭다는 것을 믿고(⑦,⑨) OP 또한 T_{SG}, T_{OP}가 새롭다는 것을 믿는다(⑧,⑩).

키 공유 시스템의 궁극적인 목적은 서비스 게이트웨이와 오퍼레이터 사이에 상호 인증을 통해 둘만의 공유 비밀키를 생성, 공유하는 것인데 BAN Logic을 이용하여 나타내면 표 6과 같다.

표 6 대칭키를 이용한 키 공유 메커니즘의 목적

▫ BAN Goal

① $OP \mid\equiv \#(SG \overset{N_{SG}}{\rightleftharpoons} OP)$ ② $SG \mid\equiv \#(SG \overset{N_{OP}}{\rightleftharpoons} OP)$

③ $OP \mid\equiv SG \mid\equiv SG \overset{N_{SG}}{\rightleftharpoons} OP$ ④ $SG \mid\equiv OP \mid\equiv SG \overset{N_{OP}}{\rightleftharpoons} OP$

즉 OP는 'N_{SG}가 SG와 OP의 공유 비밀키라는 것'이 새롭다는 것을 믿어야 하고(①), SG도 'N_{OP}가 SG와 OP의 공유 비밀키라는 것'이 새롭다는 것을 믿어야 한다(②). 또한 OP는 SG가 'N_{SG}가 SG와 OP의 공유 비밀키라는 것'을 믿는다는 사실을 믿어야 하고(③), SG도 OP가 'N_{OP}는 SG와 OP의 공유 비밀키라는 것'을 믿는다는 사실을 믿어야 한다(④).

위의 목적을 5.1에서 기술한 여러 가지 규칙들을 적용하여 검증하면 표 7, 표 8과 같다.

먼저 메시지 1에서는 SG는 OP에게 단순한 데이타만을 보내므로 생략할 수 있다.

표 7에서 보듯이 메시지 2에서는 5단계의 검증 과정이 일어난다. 적용된 규칙은 표 2를 참조한다. 단계 1에서는 규칙 1과 12를 통하여 SG가 $T_{SG}+1$와 'N_{OP}가 SG와 OP의 공유 비밀키라는 사실'을 받았다는 것을 알 수 있다. 단계 2에서는 규칙 15에 의하여 SG는 OP로부터 받은 $T_{SG}+1$과 'N_{OP}는 SG와 OP의 공유 비밀키라는 사실'이 새롭다는 것을 믿을 수 있고, 단계 3에서는 규칙 17에 의하여 SG는 'N_{OP}는 SG와 OP의 공유 비밀키라는 사실'이 새롭다는 것을 믿을 수 있다. 또한 단계 4에서는 규칙 3과 9에 의해서 SG는 OP가 'N_{OP}는 SG와 OP의 공유 비밀키라는 사실'을 보냈다는 것을 믿을 수

표 7 대칭키를 이용한 키 공유 메커니즘의 검증 (메시지 2에서)

메시지 2

단계 1

$$\frac{SG \mid\equiv SG \overset{K_{SG\_OP}}{\leftrightarrow} OP,\ SG \triangleleft \{<T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP>_{N_{SG}}\}_{K_{SG\_OP}}}{SG \triangleleft <T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP>_{N_{SG}}} \vdash \text{Introduction Rule}$$

$$\frac{SG \triangleleft <T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP>_{N_{SG}}}{SG \triangleleft (T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP)} \langle\rangle \text{ Elimination Rule}$$

단계 2

$$\frac{SG \mid\equiv \#(T_{SG}+1)}{SG \mid\equiv \#(T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP)}\ , \text{Introduction Rule}$$

단계 3

$$\frac{SG \mid\equiv \#(T_{SG}+1),\ SG \mid\equiv \#(T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP)}{SG \mid\equiv \#(SG \overset{N_{OP}}{\rightleftharpoons} OP)}\ , \text{Introduction Rule}$$

단계 4

$$\frac{SG \mid\equiv SG \overset{N_{SG}}{\rightleftharpoons} OP\ ,\ SG \triangleleft <T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP>_{N_{SG}}}{SG \mid\equiv OP \vdash (T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP)} \vdash \text{Introduction Rule}$$

$$\frac{SG \mid\equiv OP \vdash (T_{SG}+1,\ SG \overset{N_{OP}}{\rightleftharpoons} OP)}{SG \mid\equiv OP \vdash SG \overset{N_{OP}}{\rightleftharpoons} OP} \vdash \text{Introduction Rule}$$

단계 5

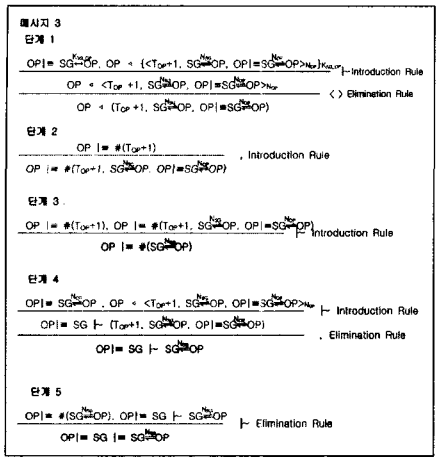
$$\frac{SG \mid\equiv \#(SG \overset{N_{OP}}{\rightleftharpoons} OP),\ SG \mid\equiv OP \vdash SG \overset{N_{OP}}{\rightleftharpoons} OP}{SG \mid\equiv OP \mid\equiv SG \overset{N_{OP}}{\rightleftharpoons} OP} \vdash \text{Elimination Rule}$$

있고 단계 5에서 4번 규칙에 의해 OP가 'N_{OP}는 SG와 OP의 공유 비밀키라는 사실'을 믿는 것을 SG가 믿을 수 있다. 즉, 메지지 2의 1~5의 검증 단계를 통하여 표 6의 목적 ②,④ 가 성립됨을 알 수 있다. 이것은 그림 3의 ⑦에서 보듯이 SG가 자신이 전송한 N_{SG}가 OP를 거쳐 동일한 N_{SG}를 받았다는 것을 나타내듯이 'SG가 OP를 인증함'을 의미한다.

메시지 3에서도 표 8과 같은 5단계의 검증 과정이 일어난다.

단계 1에서는 규칙 1과 12에 의해 OP는 $T_{OP}+1$와 'N_{SG}는 SG와 OP의 공유 비밀키'이고 'OP는 N_{OP}가 SG와 OP의 공유 비밀키라는 것을 믿는 것'을 받았음을 알 수 있고, 단계 2에서 규칙 15에 의해 OP는 SG로부터 받은 $T_{OP}+1$와 'N_{SG}가 SG와 OP의 공유 비밀키'이고, 'OP는 N_{OP}가 SG와 OP의 공유 비밀키라는 것을 믿는 것이 새롭다는 것'을 믿을 수 있다. 단계 3에서는 규칙 17에 의해 OP는 'N_{SG}가 SG와 OP의 공유 비밀키라는 사실'이 새롭다고 믿을 수 있다. 단계 4에는 3과 9번 규칙에 의해서 OP는 'SG가 SG와 OP의 공유 비밀키인 N_{SG}를 보냈음'을 믿을 수 있고, 단계 5에서 규칙 4에 의해 OP는 SG가 'N_{SG}는 SG와 OP의 공유 비밀키라는 사실'을 믿는 것을 믿을 수 있다. 메시지 3에서의 1~5의 검증 단계를 통하여 표 6의 목적 ①,③이 성립됨을 알 수 있다. 이것은 OP가 자신이 전송한 N_{OP}가 SG를 거쳐 동일한 N_{OP}를 받았다는 것을 나타내듯이 'OP가 SG

표 8 대칭키를 이용한 키 공유 메커니즘의 검증 (메시지 3에서)

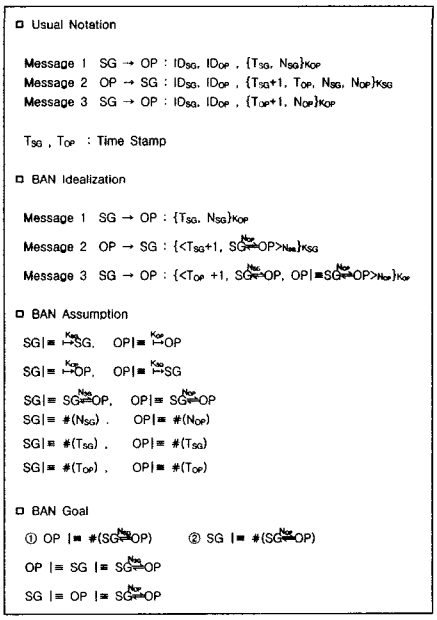


를 인증함'을 의미한다. 따라서 메시지2와 메시지3의 ⑤ 과정으로부터 SG는 OP를 인증하고, OP도 SG를 인증함으로써 상호 인증이 이루어짐을 알 수 있다. 이 과정이 후 SG와 OP가 공유하고 있는 N_{SG}, N_{OP}를 이용하여 MAC 값 생성에 필요한 공유 비밀키를 생성한다.

5.3 공개키를 이용한 키 공유 메커니즘의 정형 명세

BAN Logic에 기반하여 공개키를 이용한 키 공유 메커니즘을 명세하면 표 9와 같다. 이 명세의 검증은 메시지를 암호,복호화 할 때 SG와 OP의 공개키, 개인키를 이용한 다는 것을 제외하고 5.2절의 대칭키를 이용한 키 공유 메커니즘의 검증과 거의 유사하다. 따라서 이 메커니즘의 정형명세 검증은 생략하기로 한다.

표 9 BAN Logic 표기법에 기반한 공개키를 이용한 키 공유 메커니즘을 보안 명세



5.4 MAC 기반 서비스 번들 인증 메커니즘의 정형 명세 및 검증

그림 5의 MAC 기반 서비스 번들 인증 메커니즘의 일반적인 보안 명세는 표 10과 같다. 여기에서 메시지 1은 4장에서 설명한 ①~② 과정에 해당되고, 메시지 2는 ③~⑤ 과정에 해당한다.

표 10 MAC 기반 서비스 번들 인증 메커니즘의 일반적인 명세

□ Usual Notation

메시지 1 SG → OP : N_{SG}

메시지 2 OP → SG : N_{OP}, SJ, MAC_{OP}

SJ : Signed JAR

SS : Shared Secret

MAC_{OP} = H(SJ, SS| N_{OP} | N_{SG})

| 는 연결(concatenation)을 뜻한다.

BAN Logic을 이용하여 각각의 메시지들을 이상화된 프로토콜로 변환하면 표 11과 같다.

표 11 MAC 기반 서비스 번들 인증 메커니즘의 이상화된 프로토콜 명세

□ BAN Idealization

Message 2 OP → SG : N_{OP}, <H(SJ, SS|N_{OP}|N_{SG})>N_{SG}

메시지 1은 SG와 OP 사이의 단순한 메시지의 전달이므로 생략할 수 있다.

메시지 2에서 SG는 OP로부터 SJ와 공유 비밀키(SS), N_{OP}, N_{SG}을 기반으로 하는 해쉬 함수 값을 N_{SG}로 결합된 데이타를 받았다고 나타낼 수 있다.

이러한 이상화된 프로토콜을 검증하기 위해서 표 12와 같은 가정들을 기술할 수 있다.

표 5에서 언급했듯이 SG에서 대칭키의 접근은 기본적으로 내장되어 있는 번들 서비스에서만 접근이 가능하고 OP에서는 외부 방화벽(firewall)이 존재하고 있다고 전제한다. 그리고 SG와 OP는 랜덤 생성기(random

표 12 MAC 기반 서비스 번들 인증 메커니즘의 가정

□ BAN Assumption

① $SG|\equiv N_{SG}$, ② $SG|\equiv \#(N_{SG})$, ③ $SG|\equiv SG \overset{N_{SG}}{\rightleftharpoons} OP$

④ $OP|\equiv N_{OP}$, ⑤ $OP|\equiv \#(N_{OP})$

⑥ $SG|\equiv SG \overset{SS}{\rightleftharpoons} OP$, ⑦ $OP|\equiv SG \overset{SS}{\rightleftharpoons} OP$

generator)가 있다고 전제한다. 따라서 SG는 자신이 생성한 N_{SG}를 믿고(①), 그것이 새롭다는 것을 믿는다(②). 그리고 'N_{SG}는 SG와 OP의 공유 비밀키라는 것'을 믿을 수 있다(③). OP는 N_{OP}를 믿고 또한 그것이 새롭다는 것을 믿는다(④,⑤). 또한 SG, OP 모두 키 공유 메커니즘에 생성된 SS는 'SG와 OP의 공유 비밀키라는 것'을 믿는다(⑥,⑦).

서비스 번들 인증 메커니즘이 궁극적인 목적은 서비스 게이트웨이에서 요청한 서비스를 오퍼레이터로부터 받았을 때 변질되지 않고 정상적으로 받았느냐를 검증하는 것이다. 이것을 BAN Logic을 써서 표현하면 표 13과 같다.

표 13 MAC 기반 서비스 번들 인증 메커니즘의 목적

□ BAN Goal

$SG \mid\equiv MAC_{OP}$, $OP \mid\equiv SG \mid\equiv MAC_{OP}$

$SG|\equiv OP \vdash SJ$

즉 SG는 OP가 SJ를 보냈다는 것을 믿어야 한다.

위의 목적을 검증하기 5.1에서 제시한 여러 가지 규칙들을 적용하여 검증해보면 표 14와 같다.

표 14 MAC 기반 서비스 번들 인증 메커니즘의 검증

In Message2

단계 1

$$\frac{SG|\equiv SG \overset{N_{SG}}{\rightleftharpoons} OP,\ SG \triangleleft \langle N_{OP}, H(SJ, SS|N_{OP}|N_{SG})\rangle_{N_{SG}}}{SG|\equiv OP \vdash (N_{OP}, H(SJ, SS|N_{OP}|N_{SG}))} \vdash \text{Introduction Rule}$$

$$\frac{SG|\equiv OP \vdash (N_{OP}, H(SJ, SS|N_{OP}|N_{SG}))}{SG|\equiv OP \vdash H(SJ, SS|N_{OP}|N_{SG})} \text{, Elimination Rule}$$

단계 2

$$\frac{SG|\equiv OP \vdash H(SJ,SS|N_{OP}|N_{SG}),\ SG \triangleleft (SJ, SS|N_{OP}|N_{SG})}{SG|\equiv OP \vdash (SJ, SS|N_{OP}|N_{SG})} \vdash \text{Introduction Rule}$$

$$\frac{SG|\equiv OP \vdash (SJ, SS|N_{OP}|N_{SG})}{SG|\equiv OP \vdash SJ} \text{, Elimination Rule}$$

메시지 2에서는 2 단계의 과정이 일어난다. 우선 단계 1에서는 BAN logic 규칙 3,9 에 의해 SG는 OP가 해쉬 함수 $H(SJ, SS|N_{OP}|N_{SG})$를 보냈다는 것을 믿을 수 있다. 단계 2에서는 규칙 19, 9에 의하여 SG는 'OP가 SJ를 보냈다'는 믿을 수 있다. 따라서 1~2의 단계를 통하여 표 13의 목적을 만족함을 알 수 있다. 즉 SG는 OP가 보낸 서비스 번들을 신뢰하고 사용할 수 있다.

6. 서비스 번들 인증 메커니즘의 구현 및 평가

6.1 서비스 번들 인증 메커니즘의 설계

서비스 번을 인증 메커니즘은 오퍼레이터에서의 어플리케이션과 서비스 게이트웨이의 서비스 번들 인증 서비스로 구성된다. 그림 6은 오퍼레이터에서의 서비스 번들 인증 메커니즘의 구조를 보여주는 클래스 다이어그램(class diagram)이다.

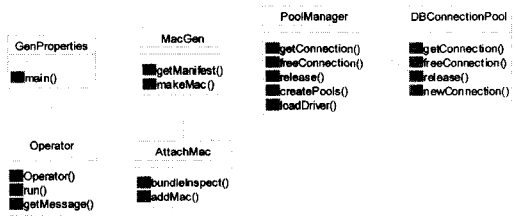


그림 6 서비스 번들 인증 시스템 클래스 다이어그램 (오퍼레이터)

서비스 게이트웨이에서 서비스 요청시 오퍼레이터에서는 MacGen 클래스를 이용하여 MAC 값을 생성하고 AttachMac 클래스를 이용하여 서명된 JAR 파일 내에 MAC 값을 저장하여 전송하게 된다. 또한 생성된 MAC 값은 DB에 저장된다.

그림 7은 서비스 게이트웨이에 있는 서비스 번들 인증을 위한 서비스 번들 인증 서비스 구조를 보여주고 있는 클래스 다이어그램이다.

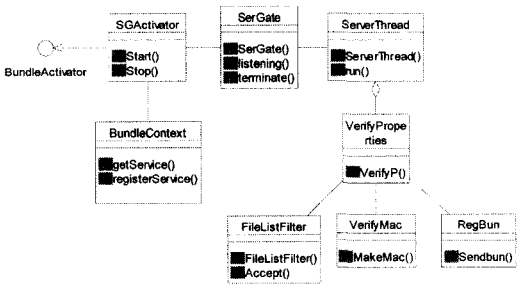


그림 7 서비스 번들 인증 시스템 클래스 다이어 그램 (서비스 게이트웨이)

BundleContext는 OSGi 서비스 플랫폼의 표준 API 클래스로서 각 번들 활성기(bundle activator)에서 서비스를 등록하고 등록된 서비스의 참조를 찾아서 사용하는데 쓰인다. 서비스 번들 인증 서비스는 크게 SGActi-

vator, SerGate, VerifyProperties 클래스로 구성된다. SGActivator은 번들 인증 서비스를 번들 문맥을 이용하여 OSGi 서비스 플랫폼에 등록한다. 등록된 서비스 번들 인증 서비스는 다른 번들이 OSGi 서비스 플래폼에 등록되기 전에 그 번들을 인증하게 된다. 실제적인 서비스 번들의 인증은 VerifyProperties 클래스에 의해 이루어진다.

6.2 구현

키 공유 시스템과 서비스 번들 인증 시스템 실험을 위하여 Windows 2000 Server 운영체제를 기반으로 개발언어는 JAVA1.3을 이용하여 시스템을 구축하였다[15-17]. 오퍼레이터와 서비스 게이트간에 생성한 공유 비밀키 및 서비스 번들 인증 시스템에서 생성된 MAC 값을 저장하기 위한 데이타베이스로는 Oracle 8i를 사용하였다. 홈 게이트웨이의 이용은 OSGi 플랫폼을 구현한 JES(Java Embedded Server 2)[18]를 사용하였으며 MAC 값을 생성하기 위해서 자바 기반의 암호화 라이브러리인 (주)시큐리티 테크놀로지스(STI)의 J/LOCK API[19]를 이용하였다. 홈 게이트웨이 시스템 환경은 현재 제품화 되어 있는 에릭슨의 E-Box[20]과 유사한 환경(200MHz CUP, 64MB RAM, 10/100M Ethernet Adapter)을 구축하여 실험하였다.

그림 8은 대칭키를 이용한 키 공유 메커니즘을 구현한 키 공유 시스템이다.

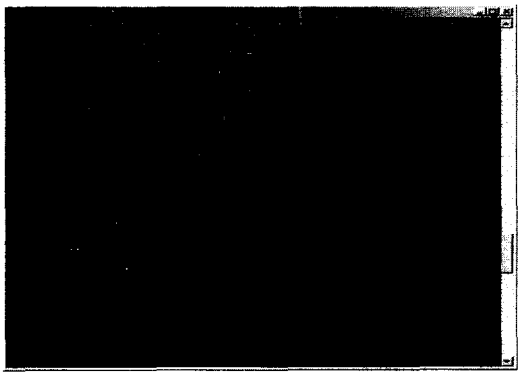


그림 8 대칭키를 이용한 키 공유 시스템

서비스 게이트웨이의 OSGi 서비스 프레임워크에 등록된 후 난수를 생성하고 오퍼레이터에 연결하게 된다. 연결 후 난수를 검사함으로써 인증을 수행하고 공유 비밀키를 생성됨을 보여주고 있다. 그림 9는 MAC 기반 서비스 번들 인증 메커니즘을 구현한 시스템이다.

서비스 번들 인증 서비스가 설치된 후 오퍼레이터로부터 서비스 번들의 전송을 대기하게 된다. 서비스 번들이 도착하였을 때 오퍼레이터에서 생성된 MAC 값

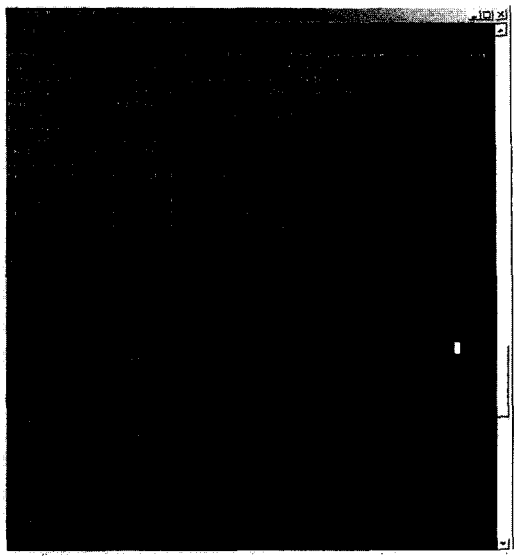


그림 9 서비스 번들 인증 시스템

(op_MAC)과 서비스 게이트웨이에서 생성된 MAC 값(sg_MAC_mv)을 비교함으로써 번들을 인증하게 된다. 인증 후 서비스 번들이 설치됨을 보여주고 있다. 대칭키를 이용한 키 공유 시스템이나 MAC 기반의 서비스 번들 인증 시스템은 그림 8, 9에서 보듯이 하나의 서비스 번들 형태로 제공된다.

6.3 평가

본 논문에서는 서비스 번들을 위한 키 공유 메커니즘과 서비스 번들 인증 메커니즘을 제시하고 구현하였다. 공개키를 이용한 키 공유 메커니즘은 대칭키를 이용한 방식에 비해 키 관리 및 분배, 갱신이 쉽지만 서비스 게이트웨이, 오퍼레이터가 암호 복호를 하는데 각각 다른 키를 가지고 있고 복잡한 연산 및 키 크기(size)가 크기 때문에 암호, 복호 속도가 느리고 전송 속도가 느리다. 또한 인증서 자체 검증과, 인증기관 연동 시에 시간이 많이 걸리며 연산 로드가 많이 걸려 자원이 많이 소모된다. 반면에 대칭키를 이용한 키 공유 메커니즘은 암호, 복호를 하는데 동일한 키를 사용하기 때문에 암호, 복호화 속도가 빠르며, 키 길이가 짧아 전송 속도가 빠르다. 따라서 연산의 속도나 네트워크 상에 전송되어야 하는 데이타의 크기를 고려해 볼 때, 제한적인 자원을 가지고 있는 OSGi 프레임워크 환경에서는 대칭키를 적용한 키 공유가 더 효율적이다. 또한 대칭키를 이용한 키 공유 메커니즘은 공개키를 이용한 키 공유 메커니즘에 비해 구현이 용이할 뿐 아니라 적은 비용으로 구축할 수 있다.

표 15에서는 서비스 번들 인증 방법을 PKI 기반과

MAC 기반으로 나누어 비교하였다.

표 15 서비스 번들 인증 메커니즘의 비교

서비스 번들 인증 방법	PKI 기반	MAC 기반
검증 방법	서비스 제공자로부터 공급받은 것을 인증	오퍼레이터가 검증한 서비스 번들을 인증
인증 속도	저속	고속
키의 최근성	중	대
오퍼레이터의 관여 정도	소	대

PKI 기반의 서비스 번들 인증에 앞서 PKI의 구성을 살펴보면 다음과 같다. 디지털 인증서를 발급하고 검증하는 인증기관, 공개키 또는 공개키에 관한 정보를 포함하고 있는 인증서, 디지털 인증서가 신청자에게 발급되기 전에 인증 기관의 입증을 대행하는 등록기관, 공개키를 가진 인증서들이 보관되고 있는 하나 이상의 디렉토리 그리고 인증서 관리 시스템으로 이루어진다. OSGi에서 제공하는 이러한 기존의 PKI 기반 서비스 번들의 인증은 특정 오퍼레이터에서 서비스 번들에 대한 인증 없이 서비스 게이트웨이에서 이루어지게 된다. 그렇기 때문에 서비스 플랫폼 상에서 서비스 번들의 인증을 위해 RSA 또는 DSA와 같은 공개키 연산뿐만 아니라 외부 인증기관과의 연동한 인증서의 유효성 검사 등 여러 가지 연산이 일어나게 된다. 이러한 연산들은 시스템 자원을 많이 차지하고 서비스 번들의 인증 속도가 느려지게 된다. 반면에 MAC 기반 서비스 번들 인증은 그림 5에서 보듯이 우선 오퍼레이터가 서비스 제공자로부터 서비스 번들을 받아 공개키 기반으로 검증을 한 후 서비스 게이트웨이로 전달시 MAC 값을 붙여 전송하게 되는데 서비스 게이트웨이에서는 단지 보내온 MAC 값과 자신이 생성한 MAC 값을 비교하는 연산만으로 서비스 번들의 무결성을 판단할 수 있다. 따라서 인증의 속도를 빠르게 할 수 있으며 또한, 서비스 번들 요청 시마다 MAC 키가 생성되므로 키의 최근성 및 보안을 높일 수 있다.

OSGi에서 제안하고 있는 RSH 프로토콜과 MAC 기반 서비스 번들 인증 메커니즘은 표 16과 같은 차이점이 있다.

그림 2에서 보듯이 RSH 프로토콜은 서버와 클라이언트 사이에 이동하는 모든 데이타에 대하여 암호화, 인증이 이루어지고 안전한 전송을 위해 여러가지 상수(A.C, E.C, A.K, E.K)를 이용하거나 함수 연산 많이 일어난다. 이러한 경우 제공되는 서비스 번들의 크기가 클 경우에는 MAC 값의 생성에 많은 시간이 걸리며 암호, 복호화가 필요 없는 데이타에 대하여 연산이 일어나므로

표 16 RSH 프로토콜과 MAC 기반 서비스 번들 인증 메커니즘의 비교

프로토콜	RSH 프로토콜	MAC 기반 서비스 번들 인증 메커니즘
MAC 값의 처리 범위	서버와 클라이언트 사이에 이동하는 모든 데이타	번들 안의 매니페스트 파일에만 해당
인증	유	유
암호화	유	무
파일의 크기	동일	축소
키 공유 메커니즘	무	유

시스템 자원을 많이 차지하거나 인증의 속도가 감소하게 된다. 그리고 RSH 프로토콜에서는 공유 비밀키(shared secret)를 만들기 위한 키 공유 메커니즘이 제시되지 않고 있다. 그러나 MAC 기반 서비스 번들 인증 메커니즘의 경우 그림 5의 ①에서 보듯이 서비스 제공자게 오퍼레이터에 서비스를 등록할 때 인증된 서비스 번들(Signed JAR)은 서명 파일과 서명 블록 파일을 제외시킨다. 그리고 서명 파일의 대체로서 MAC 값을 생성하게 되는데 전송되는 서비스 번들 전체에 해당하는 MAC 값을 생성하는 것이 아니라 클래스들의 해쉬 값인 매니페스트(manifest) 파일에 대한 MAC 값을 구하게 된다. 또한 RSH의 불필요한 암호, 복호화 과정을 없앰으로써 전송 및 인증 속도를 향상시킬 수 있다. 이러한 MAC 기반 서비스 번들 인증 메커니즘은 시스템 자원이 열악한 OSGi 서비스 플랫폼 환경에서 유리할 뿐만 아니라 이보다 더 좋은 환경에서도 유리함을 알 수 있다.

7. 결론 및 향후 과제

본 논문에서는 OSGi 플랫폼 환경에서 서비스 번들의 인증 메커니즘을 제시하고 구현, 검증하였다. 서비스 번들을 인증하기 위해서 먼저 오퍼레이터와 서비스 게이트웨이의 인증이 필요하며 이 인증 과정을 통해 둘 사이의 공유 비밀키가 생성된다. 서비스 번들 인증에 필요한 공유 비밀키를 생성하기 위해 부트스트래핑 단계에서 공개키, 대칭키를 이용한 키 공유 방법을 제시, 구현하였으며 BAN Logic을 이용해 검증하였다. 대칭키를 이용한 키 공유 메커니즘은 암호, 복호의 속도가 빠르고 키 길이가 짧기 때문에 전송 속도 또한 빠르다.

MAC 기반 서비스 번들의 인증 메커니즘에서는 키 공유 메커니즘을 통해 생성된 공유 비밀키를 이용하여 MAC을 생성한다. 기존의 PKI 기반은 안정적이지만은 연산이나 저장 공간의 제한된 OSGi 플랫폼 환경을 고려해 볼 때 부적합하다. 또한 RSH 프로토콜의 전체 데

이타에 대한 인증, 암호화는 서비스 번들의 인증 이외의 불필요한 많은 연산을 수행한다. 따라서 본 논문에서는 이러한 문제점을 없애기 위해 부트스트래핑 단계에서 키 공유 메커니즘을 통해 생성된 공유 비밀키를 이용하여 MAC을 기반으로 서비스 번들을 인증함으로써 인증 속도 및 전송 속도를 향상시켰다. 또한 BAN Logic을 이용하여 키 공유 메커니즘 및 서비스 번을 인증 메커니즘의 안정성을 검증하였다.

본 논문의 구현부분에서는 서비스 번들 인증을 위한 키 공유시스템, 서비스 번들 인증 시스템만을 구현하였으나 향후에는 논문 [21]에서 제시하는 사용자 인증 티켓의 암호화와 연동하기 위한 키 공유 시스템의 설계가 필요하고 또한 오퍼레이터의 기능이 강화됨에 따라 오퍼레이터에 대한 보안대책을 마련해야 한다.

참 고 문 헌

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

[2] OSGi, "Secure Provisioning Data Transport using Http," RFC36, http://www.osgi.org/, 2002.

[3] William Stallings, "Cryptography and Network Security," Pearson Education, 2002.

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

[5] OSGi, "OSGi Service Gateway Specification - Release 2.0" http://www.osgi.org, 2001.

[6] OSGi, "RFC 18 - Security Architecture Specification" Draft, http://www.osgi.org/member, 2001.

[7] Fielding, R., et. al., "Hypertext Transfer Protocol - HTTP/1.1, IETF RFC 2616, June 1999.

[8] H. Krawczyk et. al., "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, February 1997.

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

[10] Sun, JAR Feature, http://java.sun.com/j2se/1.4/docs/guide/jar/, 2001.

[11] Michael Burrows, Martin Abadi, Roger Needham, "A Logic of Authentication," Digital Equipment Corporation, 1989.

[12] Jan Wessels, Cmg Finance B.V. "Applications of BAN-Logic," 2001.

[13] George Coluouris, et. al., "Distributed System," Edition 2, 1994.

[14] A.D. Rubin, P. Honeyman, "Formal Methods for the Analysis of Authentication Protocols," CITI Technical Report 93-7, 1993.

[15] Jess Garms, Daniel Somerfield, "Professional Java Security," WROX, 2001.

[16] M. Pistoia, et. al., "Java 2 Network Security," Second edition, Prentice Hall, 1999.

[17] Kirk Chen, Li Gong "Programming Open Service Gateways with Java Embedded Server Technology," Sun, 2001.

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

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

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

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

김 영 갑

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

문 창 주

1997년 고려대학교 컴퓨터학과 학사 1999년 고려대학교 컴퓨터학과 석사 2000년~현재 고려대학교 컴퓨터학과 박사과정. 관심분야는 컴포넌트, 보안공학, RBAC

박 대 하

1992년 고려대학교 컴퓨터학과 학사 1994년 고려대학교 일반대학원 컴퓨터학과 석사. 1996년 고려대학교 일반대학원 컴퓨터학과 박사과정 수료. 1999년~현재 (주)시큐리티테크놀로지스 책임연구원 관심분야는 XML 보안, 보안 프로토콜, 이동코드 보안, 임베디드 시스템 보안. 1974년 고려대학교 수학과(학사). 1977년 고려대학교 대학원 산업공학과(석사) 1983년 Wayne State Univ. 전산학과(석사).

백 두 권

1985년 Wayne State Univ. 전산학과(박사). 1986년~현재 고려대학교 컴퓨터학과 교수. 1989년~현재 한국 정보과학회 평의원/이사. 1991년~현재 한국 시뮬레이션 학회 이사/부회장. 1991년~현재 ISO/IEC JTC1/SC32 국내위원회 위원장. 1999년~현재 정보통신진흥협회 데이타 기술위원회 의장. 2002년~현재 고려대학교 정보통신대학 학장. 관심분야는 데이타베이스, 소프트웨어 공학, 데이타 공학, 컴포넌트 기반 시스템, 메타데이타 레지스트리, 정보 통합