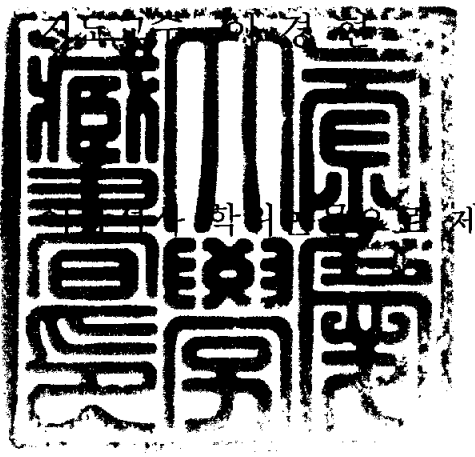
이학석사 학위논문

신뢰성 있는 이동 에이전트 수행에 관한 연구



이 논문을 [illegible] 학위 논문으로 제출함

2003년 2월

부경대학교 대학원

전자계산학과

김 희 연

김희연의 이학석사 학위논문을 인준함

2002년 12 월 26일

주 심 이학박사 박 만 곤 (인)

위 원 공학박사 여 정 모 (인)

위 원 공학박사 정 순 호

<차 례>

<표차례> ii
<그림 차례> iii
Abstract iv
1. 서론 1
2. 이동 에이전트 시스템 개요 4
2.1 이동 에이전트의 개요 4
2.2 이동 에이전트 시스템의 개요 7
2.3 Java 기반 이동 에이전트 시스템의 동향 9
3. 이동 에이전트 시스템 보호를 위한 기존 연구 12
3.1 이동 에이전트 시스템 위협 요소 12
3.1.1 에이전트로부터의 호스트 공격 12
3.1.2 호스트로부터의 에이전트 공격 13
3.1.3 에이전트로부터의 에이전트 공격 13
3.2 이동 에이전트 시스템 보호를 위한 기반 기술 14
3.2.1 암호학 기반 기술 14
3.2.2 시스템 기반 기술 16
3.3 이동 에이전트 보호를 위한 기존 연구 18
4. 이동 에이전트 보호 기법의 제안 21
4.1 에이전트 이전(Migration) 보호 기법의 제안 21
4.1.1 이전 보호 기법 22
4.1.2 제안 방안의 분석 24
4.2 에이전트 수행 결과 보호 방안의 제안 25
4.2.1 수행 결과 보호 기법 25
4.2.2 제안 기법의 분석 33
4.2.3 부정 검출 및 안전성 분석 34
5. 제안 기법의 응용 37
5.1 능동 보안(Active Security) 분야의 응용 37
5.2 전자 상거래 분야의 응용 39
6. 결론 41
참고 문헌 43

<표차례>

[표 1] 이동 에이전트 시스템의 특징 비교 11
[표 2] 이동 에이전트 시스템의 시큐리티 비교 11
[표 3] 이동 에이전트 보호 방안의 분류 20

<그림 차례>

[그림 1] 이동 에이전트의 동작 5
[그림 2] 이동 에이전트 시스템 구성 요소 7
[그림 3] 전자서명의 생성 15
[그림 4] 해쉬 값의 생성 15
[그림 5] Java 시큐리티 모델의 동작 17
[그림 6] 이동 에이전트의 수행 절차 21
[그림 7] PSM의 동작 방식 22
[그림 8] Active Security 구성 예 38

Study of Reliable Mobile Agent Execution

Hee-Yeon Kim

Dept. of Computer Science, Graduate School.

Pukyong National University

Abstract

Mobile agent systems offer a new paradigm for distributed computation and a one of solution for limitation of existent Client-Server model. Mobile agent systems provide interface that can migrate from host to host in a heterogeneous network. For secure execution, it must solve security problem of mobile code before.

In this paper, we propose the **PSM**(Protective Scheme for Migration) and the **PSER**(Protective Scheme for Execution Results). With this view, we describe an idea of mobile agent system based on mobile code and its behavior. Also we are compare with characterizations of existing mobile agent systems based on Java, and we make an analysis of it in viewpoint of security. Next, we describe the vulnerabilities against agent system and the protection technology of applicable to mobile agent. And we propose schemes that PSM for provide mutual authentication between the hosts that exchange with mobile agent, integrity and privacy of data, and that PPEM for protect the executed results by mobile agent. PPEM provide privacy of executed results, forward integrity, and non-repudiation of participate in scheme. Finally, we also present an active security for application of mobile agent technology.

The scheme of proposed in the paper is considered to be able to adapt the study of protection of mobile agent.

1. 서론

이동 코드(Mobile Code)와 이동 에이전트(Mobile Agent) 기술은 최근 분산 처리 환경 및 이동 컴퓨팅의 발전으로 주목받고 있는 기술이다. 이동 에이전트 시스템은 호스트 사이에서 에이전트가 직접 이동함을 전제로 하며, 기존의 클라이언트/서버 시스템의 문제점인 지속적인 상호 작용에 의한 과부하, 높은 대역폭 소모량, 전송 지연 등을 해결할 것으로 기대되고 있다. 또한, 인터넷과 같은 다양하고 이질적인 시스템에서의 유연한 대처가 가능하며, 독립적인 여러 에이전트가 협력하여 하나의 작업을 수행할 수 있어, 분산 처리의 효율성을 높일 수 있다는 장점을 지닌다[1].

현재 인터넷 기술과 활동에 대한 동향을 살펴보면 에이전트 시스템의 도입에 대하여 몇 가지 긍정적인 사실을 알 수 있다.

첫째, 대역폭(Bandwidth)의 분배 문제이다. 폭발적으로 증가하는 인터넷 트래픽을 수용하기 위한 노력이 이루어지고 있으나, 대부분의 사용자는 여러 가지 기술적인 요인으로 인해 사용할 수 있는 대역폭이 여전히 제한되어 있다. 둘째, 모바일 인터넷(Mobile Internet) 분야의 발전과 모바일 디바이스(Mobile Devices) 시장의 성장이다. 인터넷과 무선통신 분야를 결합한 새로운 정보 통신 서비스인 모바일 인터넷 분야가 주목받게 되면서, 휴대 가능한 컴퓨팅 디바이스 시장의 규모와 모바일 유저의 수가 증가하고 있다. 셋째, 개별화(Customization) 요구의 증가이다. 인터넷 상에서 사용자가 이용할 수 있는 서비스와 정보의 양이 엄청나게 늘어나면서, 사용자의 특성에 부합하는 서비스와 정보를 손쉽게 제공받고자 하는 요구가 증가하고 있다.

이동 에이전트 시스템은 기술한 인터넷 환경의 동향과 이동 에이전트 시스템에서 제공하는 분산 프레임워크, 이동 코드를 사용한 정보 지향 애플리케이션 구현 등의 특징을 토대로, 지적 재산 보호(Intellectual Property Protection), 네트워크 제어(Network Management), 액티브 네트워크(Active Network), 전자 상거래(Electronic Commerce) 등의 분야에서 응용되고 있다[2].

그러나, 이동 에이전트 시스템의 이점과 응용 분야의 개발에도 불구하고, 이동 에

이전트 시스템 기반의 컴퓨팅 환경으로의 이행이 지체되는 근본적인 이유는 이동 에이전트 시스템의 안전성이 완벽하게 보장되고 있지 않기 때문이다. 이동 에이전트는 네트워크 상의 여러 실행 환경에서 작업을 수행할 때, 많은 보안상의 문제점이 발생하고 있으므로, 이동 에이전트 시스템을 구현하고 실제 응용 분야에 적용하기 위해서는 보안 문제가 우선적으로 해결되어야 한다[3].

이동 에이전트 시스템의 보안은 크게 에이전트 실행 시스템의 보호와 에이전트 자체의 보호로 나뉜다. 에이전트 실행 시스템 보호는 이동 에이전트를 "외부로부터 유입되는 신뢰할 수 없는 프로그램"으로 간주하여, 에이전트의 신원 확인 및 지속적인 감시와 제재를 통해 신뢰되는 상태에서 수행하는 것에 초점을 맞추고 있다. 이는 기존의 호스트 보호 방안에서 가정하는 "정보 요새 모델(Information Fortress Model)"의 의도와 맥락을 같이 하고 있으므로, 완전히 새로운 보호 방안을 개발하기보다는 알려져 있는 기법들을 적절히 응용하는 것이 효과적이다[3].

반면에, 에이전트 자체 보호는 에이전트 실행 시스템 보호에 비해 비교적 새로운 분야이고, 에이전트 실행 시스템을 보호하기 위한 방안과 대립되는 부분이 있어 이에 대한 연구가 계속되고 있으나 완벽한 보호 방안은 알려져 있지 않다. 이동 에이전트는 호스트의 내부에서 수행되기 때문에 호스트에 의한 위험으로부터 보호될 수 없으므로, 에이전트 보호를 위한 새로운 방안이 강구되어야 한다. 본 논문에서는 이동 에이전트를 보호하기 위해 네트워크 상에서 이동 에이전트를 각 호스트로 안전하게 이전하는 기법과 에이전트의 수행 종료 후 에이전트 수행 결과의 변경을 검출할 수 있는 기법을 제안하고, 안전성을 평가한다.

본 논문의 구성은 2장에서 이동 에이전트의 정의와 특징, 기존에 소개된 이동 에이전트 시스템의 특징, 이동 에이전트 시스템에 대한 위협 요소 및 보호 기술에 대하여 기술한다. 3장에서 이동 에이전트를 보호하기 위한 기존 연구를 살펴보고, 4장에서 이동 에이전트를 보호하기 위한 안전한 이동 에이전트 이전 기법과 이동 에이전트 수행 결과를 신뢰하기 위한 부정 검출 기법을 제안하고 분석한다. 5장에서 제안 방안을 응용할 수 있는 분야를 살펴보고, 6장에서 결론을 맺는다.

2. 이동 에이전트 시스템 개요

이동 에이전트는 사용자가 처리해야 할 복잡하고 반복적인 작업, 시간이 많이 소모되는 작업을 사용자를 대행하여 처리하기 위해 네트워크로 연결된 여러 시스템 사이를 스스로의 판단 하에 이동하는 지능적인 프로그램이다.

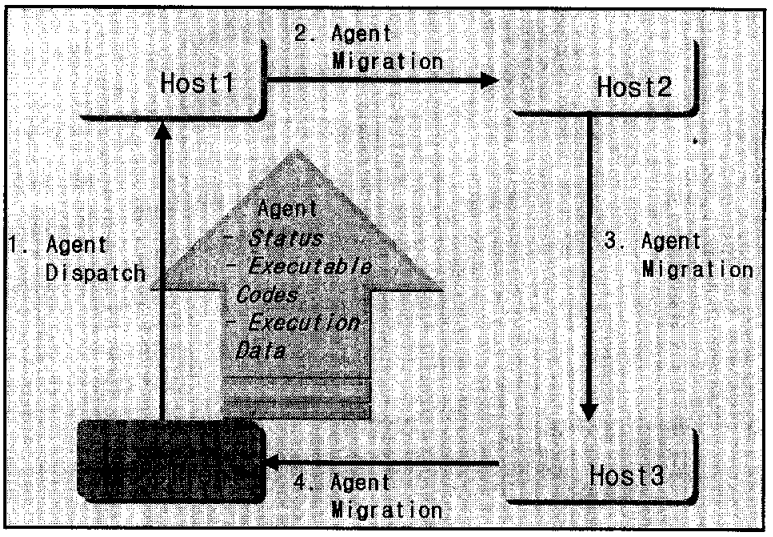
이동 에이전트가 가지는 특성 중에서 가장 두드러진 특징인 이동성은 에이전트를 구현하는 이동 코드에 기반을 두고 있으므로, 이동 에이전트의 보안은 여러 시스템에서 동작하는 이동 코드 자체에 대한 안전성에 기반 한다고 볼 수 있다.

본 장에서는 이동 에이전트의 정의와 특징, 기본 동작 방식을 설명하고, 기존에 개발된 이동 에이전트 시스템을 이동 에이전트 보호 관점에서 간략하게 소개한다.

2.1 이동 에이전트의 개요

에이전트의 정의는 어떤 특성을 기준으로 분류하는가에 따라 달라진다[4]. 본 논문에서 다루고 있는 이동 에이전트는 "이동성" 특징에 초점을 맞추고 있으므로, 이동 에이전트는 "객체(Object)의 현재 상태(Status)와 실행 가능한 코드(Executable Code)를 포함한 객체로써, 분산 환경 하에서 특정 서비스 제공을 목적으로 호스트 간 직접 이동을 통해 실행되는 객체"로 정의하고, 이동 에이전트 시스템은 "네트워크로 연결된 독립적인 다수의 에이전트가 네트워크 상의 이질적인 호스트 환경에서 원활히 동작할 수 있는 인터페이스를 제공하는 시스템"으로 정의할 수 있다.

이동 에이전트는 에이전트가 생성된 로컬 호스트 및 원격 호스트를 이동하여 독자적으로 또는, 다른 에이전트와 작업 처리를 위한 협상 및 협약을 맺으면서 수행된다. 이동 에이전트의 동작은 [그림 1]과 같다. 이동 에이전트는 에이전트의 소유자이자 이동 에이전트에게 작업을 수행하게 하는 사용자 Client를 출발하여 네트워크 상의 호스트 Host1, Host2, … 을 방문한다. 이 때, 이동 에이전트는 상태, 실행



[그림 1] 이동 에이전트의 동작

[Figure 1] Behavior of Mobile Agent

가능한 코드, 실행 데이터와 함께 한 호스트에서 다른 호스트로 완전히 이전(Migration)하고, 각 호스트의 자원과 서비스를 이용하여 작업을 수행한다. 작업 완료 후 이동 에이전트는 수행 결과를 저장하여 출발지인 Client에게로 되돌아온다.

이동 에이전트가 [그림 1]과 같은 일련의 작업 수행을 하기 위해 가져야 할 몇 가지 특징은 다음과 같다.

① 자율성(Autonomous)

인공지능 분야와 분산 처리 패러다임이 결합하여 발전된 개념으로, 이동 에이전트가 독자적인 작업 처리, 주체적인 이전 결정, 신원 증명, 비동기 적인 통신 수행, 정보 수집 및 처리, 오류 처리 등의 기능을 보유하고 있음을 의미한다.

② 적응성/ 학습성(Adaptive/ Learning)

적응성은 주위 실행 환경을 감지하고 변경에 대해 자동적으로 반응하는 능력이고, 학습성은 주어진 작업을 처리하는 방식을 미리 계획하고, 사용자의 요구

패턴과 성향을 파악하여 작업 수행 후 수집한 데이터를 적절히 변형시키는 능력이다.

③ 이동성(Mobility)

에이전트가 한 디바이스에서 다른 디바이스로 전송되도록 설계되었을 때 이동성을 가진다. 에이전트는 수행 중이던 작업을 중지하고 최종 상태를 저장한 다음, 다른 호스트로 이동한 후 작업을 재개한다. 이동성을 통해 사용자와 원격 호스트 사이의 상호 작용을 최소화하고, 수행의 효율성 증가, 네트워크 부하의 감소 등을 제공한다.

④ 영속성(Persistent)

이동 에이전트는 원격 호스트에서 한 번 수행된 후 종료되는 것이 아니라 사용자의 요청에 대한 적절한 결과를 얻을 때까지 상태와 절차를 변화시키면서 지속적으로 실행된다. 각 호스트에서 수행된 작업 결과는 에이전트가 다른 호스트로 이전할 때 삭제되지 않고 갱신(Modify), 저장(Store)을 통해 유지된다.

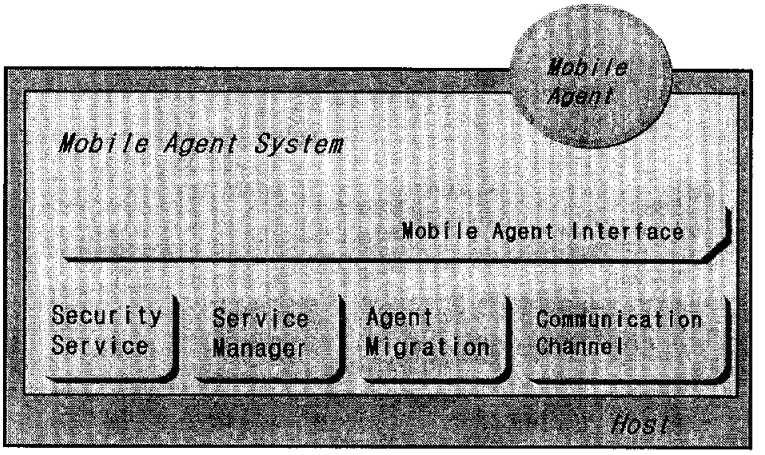
⑤ 통신성/ 협력성(Communicative/ Collaborative)

주어진 작업을 수행할 때 비슷한 작업을 수행하는 다른 에이전트들과 통신하고 협력하는 능력으로, 에이전트 단독으로 작업을 수행하는 것도 가능하지만, 주어진 하나의 작업을 협력하여 처리함으로써 분산 처리 효율성, 네트워크 트래픽 감소 효과 등을 얻을 수 있다.

2.2 이동 에이전트 시스템의 개요

이동 에이전트는 독립적이고 주체적으로 수행이 가능한 프로세서이지만, 제대로 동작하기 위해서는 이동 에이전트가 수행될 수 있는 환경이 먼저 갖추어져 있어야 한다. 즉, 이동 에이전트와 호스트 사이에서 인터페이스 역할을 수행하고 이동 에이전트가 효율적으로 작업을 완수할 수 있도록 돕는 역할을 하는 매개체인 이동 에이

진트 시스템이 필요하다[5]. 이동 에이전트 시스템의 구성 요소는 [그림 2]와 같다.



[그림 2] 이동 에이전트 시스템 구성 요소

[Figure 2] Components of Mobile Agent System

① 이동 에이전트 인터페이스(Mobile Agent Interface)

방문한 이동 에이전트와 호스트 사이를 인터페이스 하는 객체이다. 이동 에이전트의 도착, 인증, 권한 검사, 분류, 등록 등 이동 에이전트 시스템의 모든 구성 요소에서 일어나는 처리에 대해서 관여를 하게 된다.

② 시큐리티 서버(Security Server)

에이전트의 인증, 권한 검증, 암호화 등 시큐리티와 관련된 작업을 수행한다.

③ 서비스 매니저(Service Manager)

이동 에이전트 수행에 필요한 호스트 자원을 안전하게 사용하도록 지원한다.

④ 에이전트 이전(Agent Migration)

이동 에이전트의 이전은 단순한 데이터의 이동이 아닌 상태, 수행 데이터, 프로세스의 동시 이동이므로, 이를 지원하기 위한 방법이 필요하다. 에이전트 이전 구성 요소는 다른 호스트로의 이전 요청이 있을 때, 이전될 호스트와 상호 협력하여 네트워크를 통해 이동 에이전트가 이전될 수 있도록 돕는다.

⑤ 통신 채널(Communication Channel)

이동 에이전트 시스템과 이동 에이전트 시스템, 이동 에이전트와 이동 에이전트 시스템, 이동 에이전트 사이에서 통신 채널을 연결하고 해제하는 기능이다.

2.3 Java 기반 이동 에이전트 시스템의 동향

이동 에이전트 시스템의 구현은 C, Perl, Python, Tcl 등의 프로그래밍 언어를 통해서도 가능하지만, 네트워크 상에서 수행되는 이동 에이전트의 특징을 효율적으로 구현해내기 어렵다. 즉, 구현된 이동 에이전트는 네트워크를 통해 이질적인 시스템에서도 동일한 형태로 원활히 동작할 수 있어야 한다.

객체 지향 언어이자 인터프리터 언어인 Java 언어는 재 컴파일이 필요하지 않고, 가상 기계 위에서 동작하여 플랫폼 독립적인 특성을 가진다. 또한, 네트워크와 보안에 중점을 두고 개발되었기 때문에, 이동 에이전트 시스템 개발에 적절한 언어로 평가받고 있다. 따라서, 본 절에서는 Java 언어로 구현한 대표적인 이동 에이전트 시스템의 특징과 보안상의 특징을 살펴본다.

① Aglets

IBM에서 개발된 이동 에이전트 시스템인 Aglets는 Agent와 Applet의 합성어로 aglet이라고 부르는 Java 이동 객체가 네트워크 상을 이전하면서 실행된다. Aglets는 이동 Java 객체인 aglet, aglet의 작업 공간인 aglet context, 타 시스템이나 다른 aglet 객체로부터 aglet 객체를 보호하고 위치 투명성을 제공하기 위한 aglet proxy로 구성되고, 콜백(Callback) 기반 프로그래밍 모델을 도입하고 있다. 에이전트 이전을 위해 목적지 호스트에 대한 위치 기반 URL과 Java의 객체 직렬화를 이용한다. 또한, 프록시 개념을 도입하여 에이전트가 다른 객체와 직접 통신하는 것이 아니라 프록시를 거치도록 하여 외부로부터의 위

치 투명성을 제공한다. Aglets는 상호 메소드를 호출하여 동작하는 것이 아니라 메시지 전달을 통해 상호 통신이 가능하다[6].

② Voyager

Voyager는 ObjectSpace사에서 개발되었으며, Java로 만들어진 에이전트 기반의 ORB(Object Request Broker)이다. ORB는 객체 위치의 투명성, 데이터 교환, 객체 활성화를 맡고 있으며 원격 시스템 상에서 객체를 생성하고, 생성된 객체로부터 메소드를 호출하는 능력을 제공한다. Voyager 에이전트는 "Agent"라는 기본 클래스에서 제공되는 이동성과 자율성을 가지고 있으며, 지능적인 에이전트는 아니다. Voyager는 동기식(Synchronous), 비동기식(Asynchronous), 후속 호출식(Future) 메소드 호출을 지원하여 상당한 융통성을 제공하며, 메시지 전송을 위한 전문화된 경량 에이전트인 메신저(Messenger)를 통해 메소드 호출을 전달한다. 에이전트를 전송하기 위해 객체 직렬화 기법을 사용하며, 성능이 개선된 시큐리티 매니저를 탑재하고 있는데, 이는 에이전트가 수행할 수 있는 연산을 제한하기 위해 사용된다[7].

③ Ajanta

Minnesota 대학에서 개발되었으며, Java 객체 직렬화를 통해 이동성을 제공한다. 에이전트는 원격 호스트로부터의 요청에 의하여 로드되고, 공개키 시스템을 적용하여 에이전트의 이전을 암호화하고 인증한다. Ajanta에서는 다른 에이전트로부터의 간섭을 방지하기 위한 독립된 보호 구역이 설정되고, 이 보호 구역 내에서 각 에이전트의 수행이 일어난다. 호스트는 프록시 개념을 도입하여 에이전트로부터 호스트의 위치를 숨겨 리소스를 보호하고, 각 에이전트간의 안전한 통신을 위해 직접 통신이 아닌 메소드 호출을 통해 통신이 이루어지도록 한다. Ajanta에서 제안하고 있는 이동 에이전트 보호 방안은 에이전트에 암호 기법을 적용하여 에이전트 상태를 보호하고 부정에 대한 검출을 가능하게 하고 있다. 또한, 이 기법을 통해 특정 호스트에게 에이전트 객체를 선택적으로 보여줄 수 있다[8].

④ Mole

Stuttgart 대학에서 개발되었으며, Mole 에이전트는 변경되지 않고 전역적으로 유일한 이름을 통해 서비스의 제공 및 요청 작업을 수행한다. 이동성을 제공하고 있으며, 단순한 메시지, RPC 유사 통신, 지역 또는 전역적인 세션 통신 메커니즘 등을 제공한다. 보안을 위하여 Java의 Sandbox 시큐리티 모델을 도입하고, JNI로 서비스 에이전트를 작성하여 에이전트 시스템 내부의 리소스, 제어를 담당하도록 한다[9]. 다음 [표 1], [표 2]에서 각 이동 에이전트 시스템을 비교한다.

[표 1] 이동 에이전트 시스템의 특징 비교

[Table 1] Comparison of Mobile Agent Systems

에이전트 시스템	통신 방식	이동성 지원	이름 서비스	에이전트 제어
Aglets	Message의 전달	Aglet Transfer Protocol	DNS 기반의 URL	retract를 사용한 에이전트 강제 리턴
Voyager	Messenger의 지원	Java 객체 직렬화, 리플렉션	전역 ID	지원 안 함
Ajanta	Proxy를 통한 RMI	객체 직렬화	URN 형태	액세스 제어
Mole	Messaging 객체, Java RMI 지원	객체 직렬화	전역 ID 디렉토리 서비스	지원 안 함

[표 2] 이동 에이전트 시스템의 시큐리티 비교

[Table 2] Comparison of Mobile Agent System Security

에이전트 시스템	안전한 통신	호스트 리소스 보호	에이전트 보호
Aglets	지원 안 함	Sandbox 모델	지원 안 함
Voyager	지원 안 함	Sandbox 모델, 보안 채널, 확장된 Security Manager	지원 안 함
Ajanta	ElGamal, DSA를 적용하여 인증 및 암호화 전송	소유자 기반 허가	에이전트 상태, 코드 위조 감지 메커니즘
Mole	지원 안 함	Sandbox 모델	지원 안 함

3. 이동 에이전트 시스템 보호를 위한 기존 연구

본 장에서는 이동 에이전트 시스템 및 이동 에이전트를 보호하기 위한 기존 연구를 기술한다. 먼저, 에이전트 시스템에 대한 위협 요소 및 보호 기술을 살펴보고, 에이전트를 보호하기 위해 제안된 기존 연구에 대하여 알아본다.

3.1 이동 에이전트 시스템 위협 요소

본 절에서는 이동 에이전트 시스템에 대한 위협 요소[3]을 분류하고, 이동 에이전트 시스템을 보호하기 위해 적용 가능한 암호 기술[10]에 대하여 살펴본다.

3.1.1 에이전트로부터의 호스트 공격

이동 에이전트는 외부로부터 들어오는 실행코드이므로, 에이전트가 호스트의 메모리, 정보, 소프트웨어, 하드웨어, 리소스 등에 임의로 접근할 경우 다음과 같은 문제를 초래할 가능성이 있다.

① 시스템 손상(System Damage)

호스트의 파일, 컴퓨터 구성(Computer Configuration), 하드웨어, 메모리를 재구성(Reconfiguration), 변경(Modifying), 삭제(Erasing)하는 공격이다.

② 서비스 거부(Denial of Service, DoS)

컴퓨터 시스템 및 서비스의 동작을 마비시키는 공격으로, 호스트의 리소스와 서비스를 지속적으로 소비하거나 지정된 버퍼 용량을 초과시켜 시스템을 교착상태에 이르게 한다.

③ 비밀성 침해(Breach of Privacy)

호스트의 비밀 정보와 데이터에 허가되지 않은 접근을 하는 공격이다.

④ 위장(Masquerade)

에이전트가 다른 에이전트의 신분으로 인증 받아서, 정당한 에이전트의 권한으로 서비스, 리소스를 사용하는 공격이다.

3.1.2 호스트로부터의 에이전트 공격

이동 에이전트는 일반적으로 안전하지 않은 것으로 가정되는 네트워크 상의 임의의 호스트를 방문하여 작업을 수행하기 때문에, 호스트 내부에서 직접 수행되는 이동 에이전트를 보호하는 것은 어려운 일이다. 발생 가능한 공격은 다음과 같다.

① 수정(Modification)

변경, 삭제, 삽입 공격을 포함하며 이동 에이전트 내부의 상태, 이전 수행 데이터, 실행 코드에 대하여 공격이 가능하다.

② 서비스 거부(Denial of Service, DoS)

에이전트의 수행을 방해하는 것으로, 수행에 필요한 리소스 및 서비스에 접근할 수 없게 함으로써 에이전트는 무한정 대기 상태에 빠지게 된다.

③ 비밀성 침해(Breach of Privacy)

에이전트가 작업 수행을 위하여 코드 및 데이터에 대한 보호 상태를 해제할 때, 악의적인 목적을 가진 호스트가 이에 접근하고, 복제(Copy)하는 공격이다.

④ 위장(Masquerade)

악의적인 목적을 가진 호스트가 다른 정당한 호스트의 신분을 주장하여 에이전트의 실행을 허가하고, 에이전트로부터 데이터와 상태를 전달받는 공격이다.

3.1.3 에이전트로부터의 에이전트 공격

이 공격은 이동 에이전트 사이에서 정보를 수집, 교환하기 위해 서로 통신하고 접촉함으로써 야기될 수 있다. 발생할 수 있는 공격은 다음과 같다.

① 서비스 거부(Denial of Service, DoS)

악의적인 에이전트가 스팸성 메시지를 반복해서 보내거나 에이전트 자원을 점유하여 다른 에이전트의 정상적인 동작을 방해하는 것이다.

② 비밀성 침해(Breach of Privacy)

다른 에이전트의 코드나 데이터에 불법적으로 접근하거나 수정하는 공격이다.

③ 위장(Masquerade)

이동 에이전트 사이에서 통신, 정보 교환이 발생할 경우 서로의 신원 확인 과정에서 다른 에이전트의 신원을 제시하고 인증 받는 경우이다.

3.2 이동 에이전트 시스템 보호를 위한 기반 기술

이동 에이전트 시스템을 보호하기 위해 적용할 수 있는 기술은 크게 암호, 전자서명, 암호 프로토콜 등 암호학을 기반으로 한 보호 기술과 시스템 구현 측면에 있어서 프로그래밍 언어적으로 보호하기 위한 기술로 나눌 수 있다.

3.2.1 암호학 기반 기술

네트워크 및 컴퓨팅 기술의 발달로 여러 분야의 업무들이 전자화되어 네트워크상에서 구현되었고, 도청, 수정, 사용자 위장 등의 보안 위협이 발생함에 따라 교환 정보의 비밀성과 무결성을 보장하기 위한 방안이 발전하게 되었다.

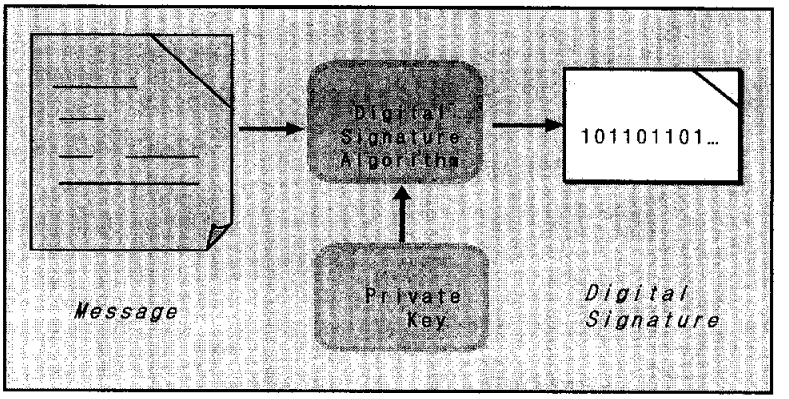
① 암 · 복호화(Encryption · Decryption)

네트워크를 통해 전송되는 데이터 및 시스템 내부에 저장되어 있는 데이터에 대하여 공격자의 도청을 막기 위해, 전달하고자 하는 평문(Plaintext)을 암호화키를 적용하여 식별할 수 없는 암호문(Cipertext)으로 변환하는 방법이다. 암호화키와 복호화키가 동일할 경우 비밀키 암호 시스템이라 하며, 암호화키와 복

호화 키가 각각 수신자의 공개키, 수신자의 개인 키로 다르게 사용될 경우 공개키 암호 시스템이라 한다.

② 전자서명(Digital Signature)

전송하는 데이터가 변경되지 않았고 본인이 생성한 것임을 증명하기 위해 사용한다. 전자서명은 송신자의 비밀키로 생성되며 수신 측에서 송신자의 공개키로 검증한다. [그림 3]은 전자서명을 생성하는 과정을 나타낸다.

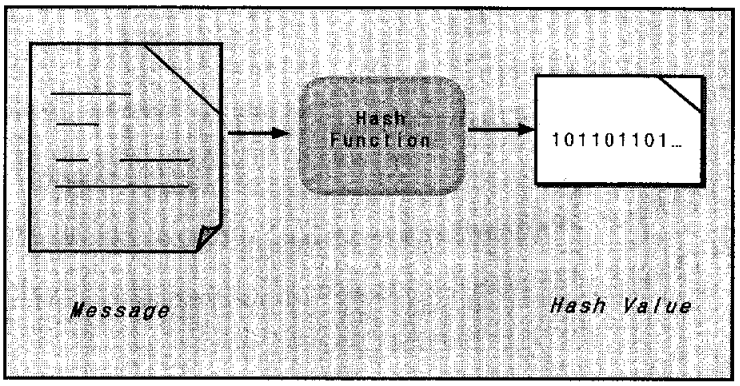


[그림 3] 전자서명의 생성

[Figure 3] Creation of Digital Signature

③ 암호학적 해쉬 함수(Cryptographic Hash Function)

메시지의 무결성과 메시지 인증을 보장하기 위해 사용되며, 해쉬 함수 $h()$는 임의의 길이의 메시지 m을 입력으로 받아 고정된 길이의 비트열 $h(m)$으로 압축, 출력하는 다대일(Many-to-One) 함수이다. 또한 "$h()$, m이 주어졌을 때 $h(m)$의 계산은 쉬운 반면, 주어진 $h(m)$으로부터 메시지 m을 구하는 것은 계산적으로 불가능하다"는 일 방향 성질을 만족해야 한다. [그림 4]는 암호학적 해쉬 값을 생성하는 과정을 나타낸다.



[그림 4] 해쉬 값의 생성

[Figure 4] Creation of Hash Value

④ 암호 프로토콜(Cryptographic Protocol)

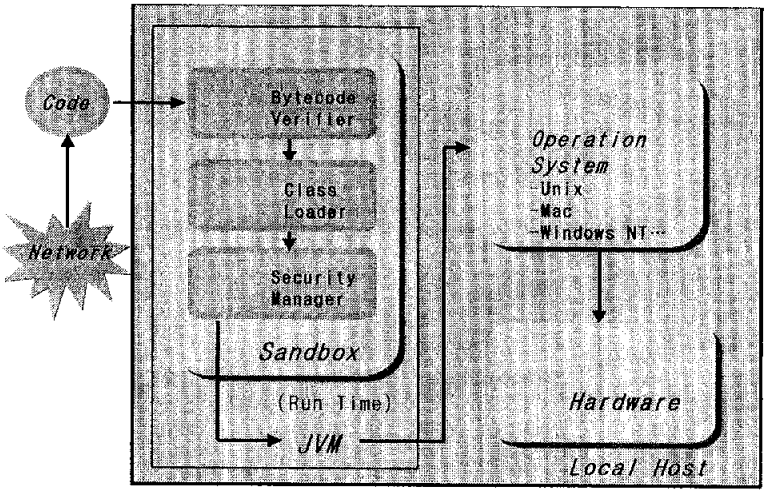
암호 프로토콜은 제공하고자 하는 서비스를 구현하기 위하여 유한한 회수의 절차에 암호 기술을 적용하여 구성한 프로토콜이다. 신용카드 정보, 개인 신상 정보 등의 비밀 정보를 보호하여, 보다 안전하고 편리하게 서비스를 제공할 수 있다. 여기에 포함되는 프로토콜로는 인증 및 상호 인증, 교환되는 문서 및 메시지의 무결성, 비밀성, 부인 방지, 가용성 서비스 등이 있다.

3.2.2 시스템 기반 기술

Java 언어는 크게 시스템 보안과 데이터 보안을 제공하며, 분산 컴퓨팅 환경에 적합한 언어로 여러 가지 분야에서 활용되고 있는 언어이다. 시스템 보안은 운영체제에서의 보안으로 시스템의 안정적인 프로세스 실행, 메모리 사용, 디바이스 사용 등의 권한 설정, 오류 검사를 통해 불법적인 접근과 사용을 막는다. 데이터 보안은 시스템 내의 모든 데이터와 디바이스를 통해 교환되는 데이터를 보호하는 측면으로 데이터 암호화, 접근 제한이 포함된다[11].

① 시큐리티 모델(Security Model)

JVM(Java Virtual Machine) 내에 Sandbox를 만들어 시스템 자원에 대한 접근 제한을 실현한다. 로컬 코드와 원격 코드에 동일한 보안 정책을 적용하며, 클래스 로더(Class Loader)에 의해, 파일 시스템에 저장되어 있는 자바 클래스가 메모리에 로드될 때, 설정된 보안 정책에 따라 시스템 자원 사용 권한이 할당된다. 로드된 클래스는 바이트 코드 검사기(Bytecode Verifier)를 통해 올바른 바이트 코드 포맷인지 검증된다. 시큐리티 매니저(Security Manager)는 신뢰성 없는 클래스가 위험한 메소드를 실행시켜 시스템 자원에 접근할 수 없도록 막는다. [그림 5]는 시큐리티 모델의 동작 상관도를 나타낸다.



[그림 5] Java 시큐리티 모델의 동작

[Figure 5] Coordination of Java Security Model

② 객체 직렬화(Object Serialization)

네트워크를 통해 객체를 내보내기 위해 사용되는 기법으로, 파일에 객체의 상태를 저장해 두었다가 나중에 메모리로 읽어 들이거나 네트워크를 통해 전송할 수 있게 한다. 객체 직렬화를 이용하여 Java 코드로 작성된 에이전트 객체를 원하는 목적지로 이전시키고 받은 객체와 상태 정보를 이용하여 사용자가

원하는 작업을 계속 수행할 수 있다.

③ RMI(Remote Method Invocation)

네트워크 상으로 다른 Java 애플리케이션과 통신할 수 있는 Java 애플리케이션을 만들기 위해 사용되며, 원격 메소드의 호출, 원격 애플리케이션의 변수에 접근, 네트워크를 통해 객체를 송·수신 가능하게 한다. RMI를 이용하면 이질적인(Heterogeneous) 시스템 및 다른 Java 환경에서도 동작 가능하며, 어떤 프로토콜을 통해 통신할지 미리 설정하지 않아도 통신이 가능하다.

3.3 이동 에이전트 보호를 위한 기존 연구

본 절에서는 기존에 제안된 이동 에이전트를 보호하기 위한 방안을 분류하고 대표적인 보호 방안에 대하여 기술한다. 이동 에이전트는 실행 가능한 프로그램이기 때문에 외부로부터 간섭받기 쉬우므로, 에이전트의 독립적인 실행과 에이전트 내부의 상태, 수행 데이터의 보호가 보장되어야 한다.

3.3.1 이동 에이전트 보호 방안의 분류

이동 에이전트 보안은 크게 예방 메커니즘(Prevention Mechanism)과 검출 메커니즘(Detection Mechanism)으로 나눌 수 있다.

예방 메커니즘은 이동 에이전트가 특정 호스트로부터 공격받는 것을 막는 것이 목적이며, 이동 에이전트의 코드와 수행 상태에 접근하거나 변경하는 것이 불가능하도록 만들고자 한다. 물리적으로 봉인된 환경에서 에이전트를 수행하고자 하는 "Tamper-proof Devices" 방법, 에이전트의 코드를 재배열(Rearrange)하여 에이전트를 실행하는 원격 호스트가 코드의 의미를 알 수 없도록 하는 "Code Scrambling" 방법, 외부 입력을 이용하여 암호화된 값을 계산하는 알고리즘을 이용하는

"Computing with Encrypted Functions" 등의 방안이 있다.

검출 메커니즘은 이동 에이전트의 코드, 상태, 실행 흐름(execution flow)에 대해 발생한 불법적인 변경을 검출하는 것이 목적이며, "Mutual Itinerary Recording", "State Appraisal Mechanism" "Cryptographic Traces" 등의 방안이 있다.

① Cryptographic Traces

G. Vigna가 이동 에이전트의 코드, 상태, 실행 흐름 등 모든 가능한 불법적인 변경을 검출하기 위해 제안한 방안으로, 이전에 방문했던 호스트에서 에이전트를 실행하면서 생성된 Trace의 서명된 해쉬 값의 목록을 에이전트의 메시지 내에 포함한다. 에이전트 소유자는 생성된 Trace에 근거하여 변경을 검출하게 되며, Trace의 재실행을 통해 실행 결과 값이 변경되었는지 확인한다. 모든 호스트는 키 분배, 개체인가(principal authentication) 등 특정 협의 사항을 만족해야 한다. 또한, 공개키 기반 구조를 사용함으로써 신뢰되지 않는 개체 사이에서의 인증, 부인 방지를 제공하고 있으며, 실행하는 동안 수집되는 Trace의 크기는 호스트의 수에 비례하여 커진다[12].

② State Appraisal Mechanism

W. M. Farmer 등에 의해 제안되었으며, 이동 에이전트는 state appraisal function과 함께 동작한다. 에이전트가 새로운 실행 환경에 도착했을 때, 에이전트의 현재 상태를 매개 변수로 하여 appraisal function을 통해 상태에 변함이 없는지 평가하고, 변경하고자 하는 시도가 있었는지 검출할 수 있다[13].

③ CEF(Computing with Encrypted Function)

T. Sander 등은 암호화되었으나 실행 가능한 프로그램 형태로 만들어진 에이전트의 함수를 원격 호스트에 보내고, 원격 호스트의 값 x를 함수에 입력 값으로 받아들여 프로그램을 실행하는 방안을 제안하였다. 이 때, 에이전트의 함수는 준동형 암호화 스킴에 기반을 두고 있으며, 에이전트 소유자와 이동 에이전트 사이의 지속적인 상호작용이 불필요하고, 함수 합성 기법을 적용하여 신뢰되지 않는 네트워크 환경에서 에이전트가 안전하게 수행되는 것을 보장한다.

그러나, 이 방안을 적용하기 위해 함수의 형태는 다르지만 같은 결과를 얻을 수 있는 준동형(homomorphism) 관계에 있는 함수를 찾는 방법이 어려워 실제 적용에 있어서는 다소 한계가 있다[14][15].

④ Protection of the Results

G. Karjoth 등에 의해 제안되었으며, 이동 에이전트가 여행하는 호스트로부터 얻는 수행 결과에 해쉬 함수를 적용함으로써, 결과 값 사이에 연관성을 부여한다. 이동 에이전트가 수집한 결과에 대한 무결성을 보장하기 위하여 암호학적 해쉬 함수와 전자서명 기법을 사용하고 있으며, 정당한 호스트라 할지라도 체인 형태로 생성된 자신의 입력 값을 재 입력할 수 없도록 하고 있다. 또한, 에이전트 소유자에 의해 검출되지 않도록 숨기면서 체인 형태로 구성되어 있는 에이전트 결과 값을 절단할 수 없도록 한다. 이동 에이전트의 코드와 데이터에 대한 비밀성은 제공하지 않는 단점이 있다[16].

다음의 [표 3]은 이동 에이전트 보호 방안을 분류한 것이다.

[표 3] 이동 에이전트 보호 방안의 분류

[Table 3] Taxonomy of Mobile Agent Security Scheme

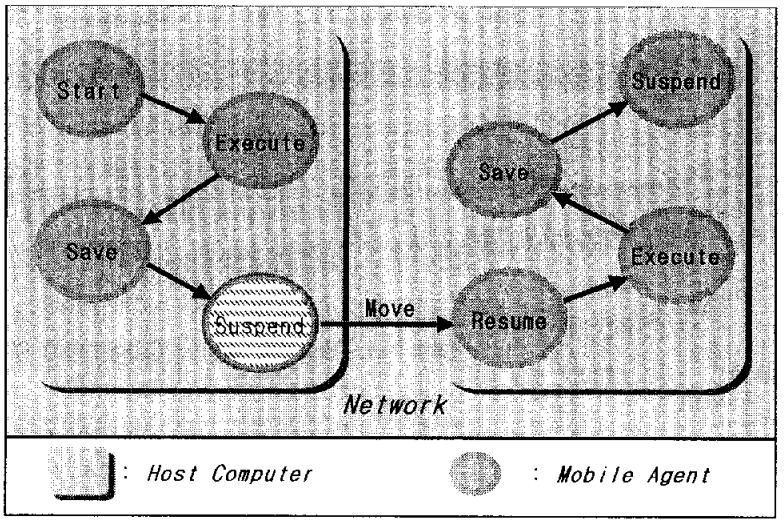
분류	Prevention Mechanism	Detection Mechanism
안전한 수행 방안	Tamper-proof Devices	-
	Code Scrambling	
	Computing with Encrypted Functions	
	Blackbox Security	
수행 결과 보호 방안	-	Sliding Encryption
		Mutual Itinerary Recording
		Itinerary Recording with Replication and Voting
		Path History
수행 후 부정 검출 방안	-	Cryptographic Traces
		State Appraisal Mechanism
		Protection of the Results

4. 이동 에이전트 보호 기법의 제안

이동 에이전트는 일종의 프로그램이므로, 실행 호스트는 에이전트의 코드와 데이터에 대한 접근이 가능하다. 따라서, 본 논문에서는 이동 에이전트를 안전하게 수행하는 것은 불가능하다는 전제 하에, 네트워크를 통해 호스트 사이를 이전하는 에이전트의 이전 보호 기법과 에이전트의 수행 종료 후 에이전트가 수집한 결과에 대한 부정을 검출하고, 전송 결과에 대한 무결성과 비밀성을 보장하는 기법을 제안한다.

4.1 에이전트 이전(Migration) 보호 기법의 제안

에이전트는 로컬 호스트에서 수행 중이던 프로세스를 중단(Suspend)하고 저장(Save)한 후, 원격 호스트로 이동(Move)하여 중단했던 프로세스를 재개(Resume)하는 일련의 절차를 통해 작업을 수행한다. [그림 6]은 이동 에이전트의 수행 절차를 나타낸다.



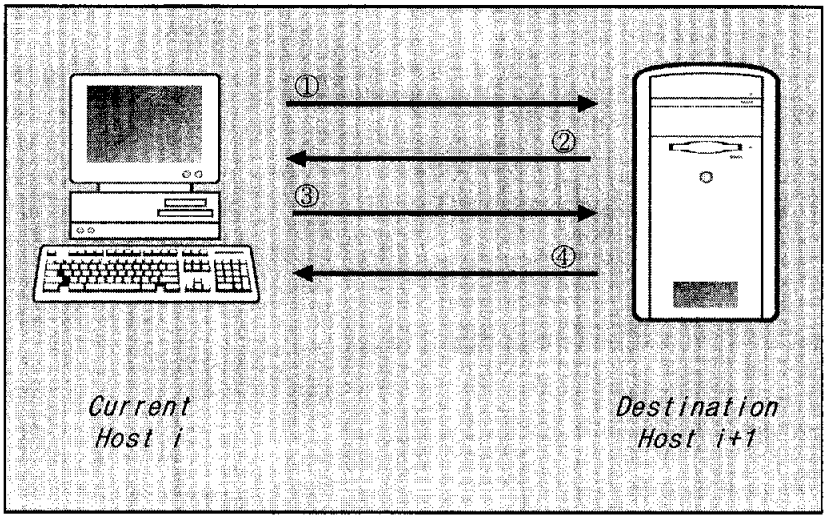
[그림 6] 이동 에이전트의 수행 절차

[Figure 6] Procedure of Mobile Agent Execution

이전(Migration)은 에이전트의 수행 절차 중에서 에이전트가 원격 호스트로 이동하는 작업과 관련이 있다. 에이전트의 이전은 안전하지 않은 여러 네트워크 상에서 이루어지므로, 네트워크 전송 중에 발생할 수 있는 에이전트의 실행 코드, 데이터, 실행 상태에 대한 도청, 수정, 위조 등과 같은 위협에 대비해야 한다. 또한, 에이전트의 이전을 정확하게 수행하기 위해 에이전트 전송 호스트와 수신 호스트 사이의 상호 인증이 필요하다.

4.1.1 이전 보호 기법

이동 에이전트는 현재 작업을 수행하고 있는 호스트 H_i 에서 작업을 종료하고 다른 호스트 H_{i+1}로 이전을 하고자 할 경우, 이전 보호 기법 "PSM(Protection Scheme for Migration)"을 현재 호스트 H_i에게 요청한다. PSM의 동작 방식은 [그림 7]과 같다.



[그림 7] PSM의 동작 방식

[Figure 7] Behavior of PSM

① Request

$$PSM_{Request} = (agent_i, H_0, Addr_{H_i}, agentSize, Cert_{H_i}, Timestamp_{H_i}) \text{(식 1)}$$

현재 이동 에이전트가 수행되고 있는 호스트 H_i는 이동 에이전트 $agent_i$의 이전을 수행하기 위해 원격 호스트 H_{i+1}로의 연결을 설정하고 $PSM_{Request}$ 메시지를 전송한다. 이 메시지는 순서대로 에이전트의 ID, 에이전트 소유자의 ID, 에이전트가 수행 중인 호스트의 주소, 에이전트의 크기, 현재 호스트의 인증서, 타임 스탬프 값을 포함한다. 원격 호스트 H_{i+1}는 $PSM_{Request}$ 메시지 정보를 기반으로 신분 확인 및 액세스 제어를 수행한다.

② Reply

$$PSM_{Reply} = (agent_i, Addr_{H_{i+1}}, Cert_{H_{i+1}}, Sign(Timestamp_{H_i}), Timestamp_{H_{i+1}}, ACK_{i+1}) \text{(식 2)}$$

원격 호스트 H_{i+1}은 PSM_{Reply} 메시지를 H_i에게 전송한다. 이 메시지는 에이전트의 ID, 원격 호스트 H_{i+1}의 주소, 호스트 H_{i+1}의 인증서, H_i로부터 받은 타임 스탬프 값에 대한 서명, H_{i+1}의 타임 스탬프 값, 허가 또는 거절의 응답을 포함하는 ACK_{i+1}을 포함한다. 호스트 H_i는 메시지 내에 포함되어 있는 인증서를 이용하여 서명 값을 검증한다. ACK_{i+1} 메시지가 거절의 응답을 포함하고 있으면, 프로토콜은 더 이상 진행되지 않고 종료된다.

③ Migration

$$PSM_{Migration} = agent_i, AgentCode, Sign(Timestemp_{H_{i+1}}) \text{(식 3)}$$

전달받은 ACK_{i+1} 메시지가 허가의 의미를 담고 있으면, 호스트 H_i는 에이전트 $PSM_{Migration}$ 메시지를 전송한다. 이 메시지는 에이전트의 ID, 이동 에이전트 코드, 호스트 H_{i+1}로부터 전달받은 타임 스탬프 값에 대한 서명을 포함한다. 호스트 H_{i+1}은 $RSM_{Request}$ 메시지 내의 인증서를 이용하여 서명 값을 검증한다.

④ Agreement

$$PSM_{Agreenent} = (agent_i, ACK_{i+1}) \quad \text{(식 4)}$$

원격 호스트 H_{i+1}은 이동 에이전트를 전송 받은 $PSM_{Migration}$ 메시지의 검증이 끝난 후 모든 프로토콜 진행이 정상적으로 동작할 경우, $PSM_{Agreement}$ 메시지를 전송한다. 이 메시지는 에이전트 ID, 에이전트 수행 허가 메시지를 포함한다. 전송이 끝난 후 에이전트를 수행한다. 예외 사항 발생시 예외 사항에 대한 정보를 ACK_{i+1} 메시지에 포함하여 전송한다.

4.1.2 제안 방안의 분석

제안 방안은 에이전트를 수행하고 있는 호스트와 다음 수행지인 원격 호스트 사이에서 동작한다. (식 1)와 (식 3)의 단계에서 교환된 인증서를 사용하여 (식 2)와 (식 3)의 메시지에서 타임 스탬프 값에 대한 전자서명을 검증한다. 전자서명의 검증을 통하여 호스트간의 상호 인증을 수행하고 프로토콜의 진행 의사를 전달한다.

현재 에이전트를 수행 중인 호스트 H_i는 (식 2)의 내부에 포함된 H_{i+1}의 전자서명을 검증하여, (식 1)에서 자신이 전송했던 타임 스탬프 값이 확실하면 에이전트를 $PSM_{Migration}$ 메시지를 전송한다. 전자서명이 안전할 경우, 각 호스트는 자신이 수행한 전자서명에 의해 프로토콜의 부인 방지를 제공할 수 있다.

4.2 에이전트 수행 결과 보호 방안의 제안

이동 에이전트에게 실행 환경을 제공하는 호스트는 악의적인 목적을 가지고, 에이전트의 데이터와 실행 코드에 대해 접근하고 수정 공격을 가할 수 있지만, 이동 에이전트는 실행 프로그램이기 때문에 공격에 대한 방어를 하기가 어렵다.

본 절에서는 에이전트의 수행 종료 후 에이전트 소유자에게 리턴된 에이전트 수행 결과에 대한 부정 검출 기법인 "PSER(Protection Scheme for Execution Results)"을 제안하고, 안전성에 대하여 논의한다.

4.2.1 수행 결과 보호 기법

제안 기법에서 이동 에이전트 *agent*는 에이전트 소유자 H_0를 출발하여 호스트 H_i $(1 \le i \le n)$를 방문한 후 다시 에이전트 소유자에게 돌아오는 것을 가정한다. 이동 에이전트 *agent*는 에이전트 소유자 H_0가 수행하고자 하는 작업에 대한 요청을 저장하고 있으며, 방문하는 각 호스트로부터 소유자 H_0의 요청에 대한 응답인 메시지 m_i를 제공받는다. PSER에 의해 이동 에이전트 *agent*는 메시지 m_i를 안전하게 저장하고 전달하기 위해 해쉬 값을 생성한다. 계산된 해쉬 값을 이용하여 호스트 H_i의 서명 값을 생성하고, 에이전트의 모든 수행이 끝난 후, 이동 에이전트의 수행 결과에 대한 부정 검출에 사용될 SP(Secure Protocol) 값을 생성한다.

에이전트를 수행하는 네트워크 상의 호스트들은 서명을 생성하기 위하여 일반적인 서명 기법을 사용한다. 각 호스트는 서명을 검증하기 위해 다른 호스트의 공개키를 알고 있고, 서명에 사용할 개인키는 안전하게 보관되는 것으로 가정한다. 서비스 거부 공격은 없으나, 각 호스트는 이동 에이전트의 수행 결과 또는 코드 변경을 위한 공격이 가능한 것으로 가정한다. 본 절에서 사용하는 표기법은 다음과 같다.

$agent_i$	이동 에이전트의 *ID*
H_0	이동 에이전트 소유자
H_i	네트워크 상의 이동 에이전트가 방문하는 호스트 (단, $1 \le i \le n$)
SH_i	메시지와 GH_i를 암호화한 값, $SH_i = E_{k_{H_i}}(m_i \| GH_i)$, (단, $0 \le i \le n$)

GH_i 메시지에 대한 해쉬 값, $GH_i = h(m_i)$, (단, $0 \le i \le n$)

Sig_{H_i} 호스트 H_i의 전자서명, $Sig_i = PR_{H_i}(Data)$, (단, $0 \le i \le n$)

SP_i 이동 에이전트 수행 결과의 부정 검출을 위한 프로토콜

$$SP_i = GH_i^{PR_{H_i}} \bmod n_{H_0} \qquad i = 0 \qquad \text{(식 5)}$$

$$SP_i = \begin{cases} SP_0^{GH_i} \bmod n_{H_0} & i = 1 \\ SP_{i-1} \times SP_0^{GH_i} \bmod n_{H_0} & i > 1 \end{cases} \qquad \text{(식 6)}$$

n_{H_0}는 SP를 생성하기 위해 사용되는 법, $n_{H_0} = p \cdot q$(단, $p \neq q$; p, q는 임의의 큰 소수)

k_{H_i} 호스트 H_i가 생성하는 임의의 비밀키, GH_i 값의 암호화에 사용, (단, $0 \le i \le n$)

PR_{H_i} 호스트 H_i의 개인키, 서명 생성에 사용

PU_{H_i} 호스트 H_i의 공개키, 서명 검증에 사용

m_i 이동 에이전트가 호스트 H_i로부터 수집한 데이터

ACK_i 에이전트 수행에 대한 응답, 수락 또는 거부 의사를 담고 있다.

제안 기법의 수행은 다음의 순서를 따른다.

〈1.1〉 이동 에이전트 *agent*는 에이전트 소유자 H_0를 출발하기 전에 위에 필요한 값을 계산하여 호스트 H_1에게 전송한다.

$$SH_0 = E_{k_{H_0}}(H_0 || m_0 || GH_0) \qquad \text{(식 7)}$$

(식 7)에서 에이전트 소유자 H_0는 임의의 비밀키 k_{H_0}을 생성하여 해쉬 값 GH_0를 계산하고, 서명하고자 하는 자신의 *ID* H_0, 이동 에이전트에 의해 수행

될 작업을 명시한 메시지 m_0, GH_0를 임의의 비밀키 k_{H_0}를 사용하여 암호화한 안전한 해쉬 SH_0을 계산한다. 이 때, SH_0를 생성하기 위해 사용된 암호화 비밀키 k_{H_0}는 에이전트 소유자 H_0에 의해 단 한 번만 사용되고, 에이전트가 이동하는 바로 다음 호스트에게만 안전하게 전달되고 제안 방안의 나머지 과정에서는 비밀로 유지된다. SH 값을 통해 전달되는 메시지의 무결성과 비밀성을 보장한다.

$$SP_0 = GH_0^{PR_{H_0}} \bmod n_{H_0} \quad \text{(식 8)}$$

(식 8)에서 에이전트 소유자 H_0는 에이전트의 수행 결과에 대한 부정 검출 기법인 안전한 프로토콜(Secure Protocol) SP를 생성한다.

SP는 GH_0 ($GH = h(m_0)$)에 에이전트 소유자의 개인키 PR_{H_0}를 누승하여 법 n_{H_0}을 취한 값이다. SP_0는 부정 검출 프로토콜 및 에이전트 소유자 H_0의 서명 값으로 사용된다.

$$H_0 \xrightarrow{(SH_0, SP_0)} H_1 \quad \text{(식 9)}$$

(식 9)에서 이동 에이전트 *agent*는 계산한 (SH_0, SP_0) 값을 가지고 에이전트 소유자 H_0를 출발하여 다음 목적지인 호스트 H_1에 도착한다.

〈1.2〉 이동 에이전트 *agent*가 도착하면, 호스트 H_1은 *agent*의 실행에 앞서 수행 의사를 포함하는 ACK 메시지를 서명하여 이전 호스트 H_0에게 서명 값을 전송한다.

$$Sig_{H_1} = PR_{H_1}(ACK_1) \quad \text{(식 10)}$$

(식 10)에서 호스트 H_1은 이동 에이전트 실행을 결정하는 메시지 ACK_1을 서명하여 이전 호스트 H_0에게 전송한다.

$$H_1 \xrightarrow{(Sig_{H_1})} H_0 \qquad \text{(식 11)}$$

(식 11)에서 이전 호스트인 H_0에게 서명 값을 전송한다.

〈1.3〉 에이전트 소유자 H_0는 호스트 H_1의 공개키로 서명 값 Sig_{H_1}을 검증한다. 만약 검증된 ACK_1 값이 수행 거부 의사를 포함하고 있다면 프로토콜은 종료된다.

$$H_0 \xrightarrow{PU_{H_1}(k_{H_0})} H_1 \qquad \text{(식 12)}$$

(식 12)에서 서명이 올바르면, SH_0에 사용된 임의의 비밀키 k_{H_0}를 호스트 H_1의 공개키로 암호화하여 전송한다.

〈1.4〉 호스트 H_1은 전송 받은 k_{H_0}를 이용하여 SH_0를 복호하고 에이전트 *agent*가 가지고 온 에이전트 소유자 H_0의 서명 값을 검증한다.

$$\begin{cases} SP_0^{PU_{H_0}} \bmod n_{H_0} = GH_0' \\ SH_0^{k_{H_0}} = m_0 \| GH_0 \\ h(m_0) = GH_0'' \end{cases} \qquad \text{(식 13)}$$

(식 13)의 SP_0 값은 부정 검출 프로토콜과 에이전트 소유자의 서명 값으로 동시에 사용된다. GH_0', GH_0, GH_0'' 사이의 관계가 $GH_0' = GH_0 = GH_0''$인지 검증한다. 세 가지 값이 모두 동일하다면 서명 값과 SH_0 값이 정당한 것으로 판단한다.

$$SH_1 = E_{k_{H_1}}(H_1 \| H_0 \| m_1 \| GH_1) \qquad \text{(식 14)}$$

(식 14)에서 호스트 H_1은 이동 에이전트 *agent*를 실행하고 요청에 대한 결과 값인 m_1을 생성한다. 자신의 ID H_1, 에이전트 소유자의 ID H_0, 호스트 H_1에서

생성한 m_1, GH_1을 SH의 입력으로 사용하여 암호화된 해쉬 값 SH_1을 계산한다. 이 때, GH_1을 계산하는 과정을 살펴보면, 에이전트 수행에서 공통으로 사용되는 일반적인 해쉬 함수를 적용하여 해쉬 값 $h(m_1)$을 계산하고 임의의 비밀 키 k_{H_1}을 생성하여 $(H_1||H_0||m_1||GH_1)$을 암호화한다.

$$Sig_1 = PR_{H_1}(GH_1) \quad \text{(식 15)}$$

(식 15)에서 개인키 PR_{H_1}을 적용하여 GH_1에 대한 호스트 H_1의 서명 값을 생성한다.

$$SP_1 = SP_0^{GH_1} \bmod n_{H_0} \quad \text{(식 16)}$$

(식 16)에서 호스트 H_1은 이전 호스트 H_0로부터 전송 받은 프로토콜 SP_0를 생성하기 위해 자신의 해쉬 값 GH_1을 적용한다.

$$H_1 \xrightarrow{\{(SH_0, SP_0), (SH_1, Sig_1, SP_1)\}} H_2 \quad \text{(식 17)}$$

(식 17)에서 모든 계산을 마친 호스트 H_1은 이동 에이전트 *agent*가 이전하고자 하는 다음 호스트인 H_2에게 위의 값을 전달한다.

〈2.1〉 이동 에이전트 *agent*가 호스트 H_2에 도착한다.

〈2.2〉 호스트 H_2은 *agent*의 실행 여부를 결정하여 H_1에게 응답한다.

$$Sig_{H_2} = PR_{H_2}(ACK_2) \quad \text{(식 18)}$$

(식 18)에서 호스트 H_2는 전달받은 이동 에이전트에 대한 수행을 결정하는 ACK_2 메시지를 서명하여 이전 호스트인 H_1에게 전송한다.

$$H_2 \xrightarrow{(Sig_{H_2})} H_1 \qquad \text{(식 19)}$$

(식 19)에서 이전 호스트인 H_1에게 서명 값을 전송한다.

〈2.3〉 호스트 H_1은 전달받은 호스트 H_2의 서명 Sig_{H_2}을 검증한다.

$$H_1 \xrightarrow{PU_{H_2}(k_{H_1})} H_2 \qquad \text{(식 20)}$$

(식 20)에서 올바른 서명임을 검증한 후, SH_1에 사용된 임의의 비밀키 k_{H_1}를 호스트 H_2의 공개키로 암호화하여 전송한다.

〈2.4〉 호스트 H_2는 전송 받은 k_{H_1} 값을 이용하여 SH_1을 복호하고 호스트 H_1의 서명 값을 검증한다.

$$\begin{cases} SH_1^{k_{H_1}} & = m_1 \| GH_1 \\ Sig_{H_1}^{PU_{H_1}} & = GH_1' \end{cases} \qquad \text{(식 21)}$$

(식 21)에서 호스트 H_2는 H_1으로부터 전달받은 비밀키 k_{H_1}을 적용하여 얻은 결과 값 GH_1, 호스트 H_1의 서명을 검증하여 나온 GH_1' 사이의 관계가 $GH_1 = GH_1'$인지 확인한다. 이 두 값이 동일할 경우 서명 값 Sig_{H_1}과 SH_1 값을 정당한 것으로 판단한다.

$$SH_2 = E_{k_{H_2}}(H_2 \| H_0 \| m_2 \| GH_2) \qquad \text{(식 22)}$$

(식 22)에서 호스트 H_2는 이동 에이전트 $agent$를 실행하고 요청에 대한 결과 값인 m_2를 생성한다. 자신의 ID H_2, 에이전트 소유자의 ID H_0, 생성한 결과 값 m_2, GH_2를 SH에 적용하여 암호화된 해쉬 값 SH_2을 계산한다.

$$Sig_2 = PR_{H_2}(GH_2) \quad \text{(식 23)}$$

(식 23)에서 호스트 H_2는 개인키 PR_{H_2}을 적용하여 GH_2에 대한 서명 값을 생성한다.

$$SP_2 = SP_1 \times SP_0^{GH_2} \bmod n_{H_0} \quad \text{(식 24)}$$

(식 24)에서 호스트 H_2은 (식 5), (식 6)의 조건에 따라 SP_2를 생성한다.

$$H_2 \xrightarrow{\{(SH_0, SP_0), (SH_1, Sig_1, SP_1), (SH_2, Sig_2, SP_2)\}} H_3 \quad \text{(식 25)}$$

(식 25)에서 모든 계산을 마친 호스트 H_2는 이전 호스트로부터 전송 받은 데이터와 자신이 생성한 데이터를 이동 에이전트 *agent*가 이전하고자 하는 다음 호스트인 H_3에게 위의 값을 전달한다.

⋮

〈n.1〉 에이전트는 $n-1$번 째 호스트 H_{n-1}로부터 n번 째 호스트 H_n으로 이동한다.

$$H_{n-1} \xrightarrow{\{(SH_0, SP_0), (SH_1, Sig_1, SP_1), \ldots, (SH_{n-1}, Sig_{n-1}, SP_{n-1})\}} H_n \quad \text{(식 26)}$$

〈n.2〉 호스트 H_n이 에이전트의 수행을 허용하는 메시지를 호스트 H_{n-1}에게 전송한다.

$$H_n \xrightarrow{(Sig_{H_n})} H_{n-1} \quad \text{(식 27)}$$

〈n.3〉 호스트 H_{n-1}은 서명 검정 후 에이전트 수행에 필요한 암호화 비밀키를 호스트 H_n에게 전송한다.

$$H_{n-1} \xrightarrow{PU_{H_n}(k_{H_{n-1}})} H_n \quad \text{(식 28)}$$

〈$n.4$〉 호스트 H_n은 비밀키 $k_{H_{n-1}}$을 이용하여 해쉬 값 SH과 호스트 H_{n-1}의 서명 $Sig_{H_{n-1}}$을 검증하고 부정 검출 프로토콜 SP_n을 생성하고 자신의 서명값 Sig_n을 생성한다.

$$\begin{cases} SH_{n-1}^{k_{H_{n-1}}} & = m_{n-1} \| GH_{n-1} \\ Sig_{H_{n-1}^{PU_{H_{n-1}}}} & = GH_{n-1}{}' \\ SP_n & = SP_{n-1} \times SP_0^{GH_n} \bmod n_{H_0} \end{cases} \quad \text{(식 29)}$$

〈$n.5$〉 에이전트는 호스트 H_n으로부터 에이전트 소유자 H_0에게 이전한다.

$$H_n \xrightarrow{\{(SH_0, SP_0), (SH_1, Sig_1, SP_1), \ldots ((SH_n, Sig_n, SP_n)\}} H_0 \quad \text{(식 30)}$$

에이전트가 에이전트 소유자 H_0에게 되돌아오면 제안 기법의 수행이 종료된다.

4.2.2 제안 기법의 분석

PSER은 이동 에이전트의 실행 결과에 대한 부정 검출 및 실행 추적 기능을 제공한다. 이동 에이전트의 수행에 따른 PSER의 동작 중 부정 검출 및 실행 추적 기능을 제공하는 핵심 부분인 SP의 동작은 다음과 같다.

에이전트 소유자 H_0가 생성한 SP_0 값은 메시지에 대한 해쉬 값 GH_0에 H_0의 비밀키 PR_{H_0}를 적용한 값이다. 이 값은 일반적인 서명 값 생성 형식과 동일한 형태를 가지므로, 프로토콜 생성과 서명 값을 계산하는 이중 작업을 피하기 위하여 에이전트 소유자의 경우, 즉, $i=0$인 경우만 SP_0 값을 서명 값으로 간주한다. 호스트 H_1은 전송 받은 SP_0 값과 생성한 GH_1 값을 입력으로 하여 H_1의 새로운 SP_1 값을 생성한다. $i>1$ 일 때, SP 값 사이에 연관성이 부여된다.

다음의 (식 31) ~ (식 36)은 모든 수행이 끝난 후 에이전트가 가지고 온 SP의 값을 에이전트 소유자 H_0가 계산하는 과정이다.

$$SP_i = SP_{i-1} \times SP_0^{GH_i} \bmod n_{H_0} \quad \text{(식 31)}$$

$$= SP_{i-2} \times S_0^{GH_{i-1}} \times SP_0^{GH_i} \bmod n_{H_0} \quad \text{(식 32)}$$

$$= SP_0^{GH_1} \times \cdots \times SP_0^{GH_{i-1}} \times SP_0^{GH_i} \bmod n_{H_0} \quad \text{(식 33)}$$

$$= (SP_0)^{GH_1 + \cdots + GH_i} \bmod n_{H_0} \quad \text{(식 34)}$$

$$= (GH_0^{PR_{H_0}})^{GH_1 + \cdots + GH_i} \bmod n_{H_0} \quad \text{(식 35)}$$

$$= (GH_0^{GH_1 + \cdots + GH_i})^{PR_{H_0}} \bmod n_{H_0} \quad \text{(식 36)}$$

SP_i 값은 (식 31)과 같이 정의된다. 이 값은 (식 32)에서 보는 바와 같이 에이전트 소유자 H_0의 SP_0 값에 대한 해쉬 값 GH_0, 에이전트가 이동한 첫 번째 호스트의 H_1 값에 대한 해쉬 값 GH_1, …, 에이전트가 이동한 마지막 호스트의 H_n 값에 대한 해쉬 값 GH_n 들의 합으로 바꿀 수 있다. 각 계산 과정을 통해 SP_i 값은 최종적으로 (식 36)의 식으로 변형되고 모든 해쉬 값의 합에 대해 에이전트 소유자 H_0의 개인키로 서명한 형태가 된다. H_0의 개인키 PR_{H_0}가 안전하다면, SP_i 값은 최종적으로 H_0만 계산할 수 있다.

4.2.3 부정 검출 및 안전성 분석

① 부정 검출

에이전트 소유자는 이동 에이전트 도착 후 에이전트를 수행하여 얻은 결과가 신뢰할 수 있는 것인지 검증할 필요가 있다고 판단되면 부정 검출 프로토콜을 시작한다. 부정 검출 프로토콜의 수행은 다음과 같다.

H_0는 (식 11), (식 19), (식 27)와 같은 각 호스트에 의해 서명된 ACK_i 값을 보

고 프로토콜 참여 여부를 확인 할 수 있으며, 서명 기법이 안전하다면 해당 호스트는 에이전트 수행을 수락한 사실을 부인할 수 없다. 또한 (식 15), (식 23), (식 29)의 서명 값 생성에 의해 자신이 GH_i 값을 생성한 사실을 부인 할 수 없다. 에이전트 수행 과정에서 부정한 호스트의 수행은 정직한 호스트의 도움으로 검출할 수 있다. 예를 들어, 호스트 H_j가 부정한 호스트이고 정직한 이전 호스트 H_{j-1}의 GH_{j-1} 값을 임의로 변경하여 부정을 저지르고자 할 때, 호스트 H_{j-1}로부터 SH_{j-1}에 사용된 암호화 비밀키 $k_{H_{j-1}}$를 얻기 위하여 에이전트를 수행할 것이라는 메시지가 들어있는 서명 값을 생성하여 전달한다. 암호화 비밀키 $k_{H_{j-1}}$을 사용하여 SH_{j-1}로부터 $(H_{j-1}||H_0||m_{j-1}||GH_{j-1})$을 복호하고 메시지 m_{j-1}을 변경한 후 새로운 GH'_{j-1}을 생성하여 암호화 비밀키 $k_{H_{j-1}}$를 적용한 SH'_{j-1}을 생성할 수 있다. 또한 이전 호스트 H_{j-2}가 계산한 SP_{j-2}의 값에 새로 계산한 GH'_{j-1}를 적용하여 호스트 H_{j-1}의 위조된 SP'_{j-1}을 생성할 수 있다. 그러나 호스트 H_{j-1}의 개인 키가 안전하게 보관될 경우, 서명 값은 위조할 수 없기 때문에 H_j의 다음 호스트인 정직한 호스트 H_{j+1}에 의해 H_j가 부정 호스트임이 검출된다. 만약 에이전트 소유자가 에이전트가 여행한 모든 경로 상의 호스트에 대한 부정 검출을 확인하고자 한다면, (식 30)의 에이전트 최종 수행 결과를 차례대로 시뮬레이션 함으로써 하나 이상의 부정에 대한 검출이 가능하다.

② 안전성 분석

제안 기법은 다음과 같은 안전성을 제공한다.

첫째, 프로토콜 참여에 대한 부인 방지를 제공한다. (식 10), (식 18), (식 26)에서 각 호스트 H_i는 이동 에이전트의 수행을 허가하는 전자서명된 ACK_i 메시지를 호스트 H_{i-1}에게 전송하기 때문에 전자서명에 의한 부인 방지가 제공된다.

둘째, 에이전트의 각 수행에 대한 전방향 무결성(Forward Integrity)을 제공한다. 제안 기법의 각 호스트가 생성하는 값은 해당 호스트의 비밀키에 의해 전자서명되기 때문에 서명 방안이 안전할 경우, 악의적인 호스트는 이전에 생성된 결과 값들에 대하여 수정 공격이 불가능하므로 전방향 무결성이 제공된다. 다음은 전방향 무결성의 정의이다.

Definition 1. 전방향 무결성(Forward Integrity)

이동 에이전트가 n개의 호스트 $H_1, H_2, \cdots, H_n$을 방문하고, 최초로 나타나는 악의적인 호스트가 H_c 일 때, 악의적인 호스트 H_c는 이동 에이전트가 이전에 방문했던 임의의 호스트 H_i에서 생성된 결과 값을 변경할 수 없다 (단, $i<c$).

예를 들어, 악의적인 호스트 H_4가 이전 호스트 H_3으로부터 받은 이동 에이전트의 수행 결과를 변경하고자 할 경우, H_4는 H_3의 암호화 비밀키 PR_{H_3}를 위조하여 메시지 값과 해쉬 값 GH_3 값을 변경하고, 새로운 프로토콜 SP'_3 ($SP'_3 = SP_2 \times SP_0^{GH_3} \bmod n_{H_3}$)을 계산할 수 있다. 그러나 서명에 사용되는 개인키가 안전할 경우 호스트 H_3의 서명 값은 위조할 수 없다.

셋째, 각 호스트는 이전 호스트의 수행이 올바른지 검증할 수 있다. 서명에 사용되는 개인키가 안전할 경우, 이전 호스트의 서명 값은 위조될 수 없으므로, 메시지가 변조된 경우 다음 호스트의 검증 과정에서 변조되었음이 드러난다.

넷째, 전송 메시지의 비밀성과 무결성을 제공한다. 메시지를 보호하기 위해 암호화된 해쉬 값 SH를 사용하기 때문에 메시지가 네트워크 상에서 노출되지 않는다.

다섯째, 에이전트 소유자는 모든 에이전트 수행 결과에 대한 부정 검출이 가능하다. SP의 검증을 통하여 이동 에이전트 수행 결과에 대한 변경을 알아낼 수 있다.

5. 제안 기법의 응용

제안한 에이전트의 수행 결과 보호 기법은 능동 보안과 전자 상거래 등의 분야에서 에이전트의 수행 결과를 검증하고, 안전한 에이전트의 수행을 보장하기 위해 적용될 수 있다.

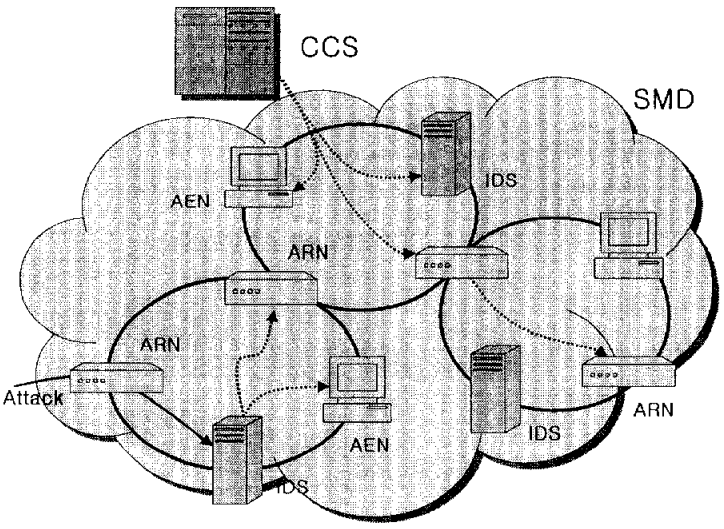
5.1 능동 보안(Active Security) 분야의 응용

네트워크와 인터넷의 폭넓은 보급으로 수많은 애플리케이션이 등장하고 있으며, 특히 민감한 정보를 취급하는 애플리케이션에서는 이를 보호하기 위해 다양한 정보 보호 기술들이 적용되고 있다. 그러나, 네트워크 상에서의 공격은 점차 조직화, 분산화, 자동화되고 있어 모든 공격 기법에 대하여 안전한 애플리케이션 구축은 사실상 불가능하다. 따라서, 그러한 공격 기법에 대응할 수 있는 보다 능동적이고 통합적인 관점의 보안 시스템이 요구된다. 능동 보안(Active Security)은 시스템의 능동적·통합적 보안을 목적으로 제안된 하나의 보안 기반 구조로써, 기반 구조의 프레임워크, 각 보안 시스템, 시스템 사이의 통신 방법, 네트워크 차원에서 대응책을 조율·지시할 수 있는 체계, 구현을 위한 프로그래밍 언어, 통신 프로토콜, 각 시스템의 실행 구조 등으로 구성되며, 다음과 같은 보안 요구 사항을 만족해야 한다[17].

- 시스템 공격에 대한 수동적·능동적 대응이 가능한 보안
- 시스템과 내부 망을 포함한 전체 네트워크에 대한 보안
- 보안 시스템의 독립적인 운용에서 상호 결합적인 운용
- 공격 및 시스템 대응 방법에 대하여 구체적인 기술과 수용이 가능한 유연한 보안 시스템 구조를 포함
- 새로운 보안 정책 및 기술의 수용이 용이한 개방적 실행 구조를 가지는 보안 시스템

능동 보안은 하나의 중앙 관제 시스템(Central Coordinator System, 이하 CCS)과 여러 개의 보안 관리 영역(Security Management Domain, 이하 SMD)으로 구성된다. 하나의 SMD는 하나의 능동 대응 노드(Active Response Node, 이하 ARN)를 포함하는 기초 영역(Elementary Domain, 이하 ED)들로 나누어진다.

ARN의 주 기능은 탐지된 공격, 능동 노드에서 수행 가능한 대응 방안, 현재 진행 중인 공격에 대해 수행 중인 대응 방안 등과 관련된 정보를 인접 ARN과 교환하고 CCS에 보고하는 것이다. 능동 실행 노드(Active Execution Node, 이하 AEN)는 SMD 내에 존재하며, 새로운 공격 유형에 대한 서술 정보 및 탐지 프로그램을 CCS로부터 능동적으로 다운로드 하여 이를 실행할 수 있도록 환경을 조성한다. SMD는 ARN으로부터 보고되는 공격 유형과 같은 정보를 통합하여 CCS에 보고하고, 자신의 영역 안에서 통합된 대응 방안을 결정·적용하는 기능을 담당한다. CCS는 ARN으로부터 보고된 공격 정보를 분석하고 관리 영역 차원에서 더욱 효과적인 대응 방안을 찾아서 시스템 내의 ARN에 수행을 지시한다. 다음의 [그림 8]은 능동 보안의 예를 보여준다.



[그림 8] Active Security 구성 예

[Figure 8] Configuration of Active Security

[그림 8]에서 최초로 공격을 감지한 ED의 ARN은 주위의 ARN 및 AEN에 공격 사실 및 대응 상황을 보고한다. 공격을 탐지한 ARN은 대응 방안을 독자적으로 결정할 수도 있고, CCS에 그 사실을 통보하여 더 나은 대응 방안을 요청할 수도 있다. ARN으로부터 보고 및 대응 방안 요청이 들어오면, CCS는 공격에 대한 분석 후 전체 SMD 내의 ARN에게 변화된 보안 환경, 새로운 공격 기법에 대한 대응 방안을 배포하고 AEN을 통하여 수행할 수 있도록 한다.

이때, ARN 사이에서는 공격 및 대응 방안에 대한 정보를 교환하기 위한 통신이 필수적이고, 각 AEN은 변경된 보안 정책과 대응 방안을 적용하기 위한 실행 환경의 구축되어야 한다. 이러한 작업은 이동 에이전트의 도입을 통하여 더 효율적으로 수행될 수 있다. 이동 에이전트는 ARN 사이를 이동하면서 능동 보안에 대한 여러 자료를 배포하거나 공격 사실을 이웃 ARN에 알리는 역할을 수행하는 것이 가능하며, AEN 상에서 이동 코드를 직접 수행하여 공격 패턴의 탐지, 탐지된 공격에 대한 대응 절차, 수집된 정보의 보고 등을 직접 수행할 수 있다. 따라서, 이동 에이전트의 도입에 따른 에이전트 인증, 에이전트 정보 보호, 실행 환경 보호, 에이전트의 안전한 이전 등이 지원되어야 한다. 이 밖에도 보안 관련 연산의 성능, 각 구성 요소들 간의 호환성, 보안 기능의 확장성, 유연성, 공격 대응의 확실성, 효율성, 보안 기반 구조 구성의 단순화 및 관리의 조건이 충족되어야 한다.

능동 보안 시스템은 새로운 공격 기법의 대응 방법을 기술한 에이전트를 다운로드하고 이를 실행하여 침입 탐지 및 차단을 수행할 수 있다. 즉, 이동 에이전트의 도입을 통하여 능동 보안 시스템의 보안 수준 향상과 새로운 기능의 수용으로 인한 시스템 확장성을 보장할 수 있으며, 결과적으로 동적인 시스템을 구성할 수 있게 된다.

5.2 전자 상거래 분야의 응용

전자 상거래는 인터넷의 도입으로 급속하게 발전한 분야이다. 인터넷 사용자들은 컴퓨터를 통해 보다 많은 상품과 서비스에 대한 선택권을 제공받게 되었다. 그러나, 선택권이 늘어남에 따라 필요한 상품과 서비스에 대해 만족할 만한 선택을 하기 위한 검색 시간의 증가, 선택 기준의 모호함에 따른 구매 만족도의 저하 등의 문제가 발생하게 되었다. 이와 같은 문제점을 해결하고, 보다 사용자 중심적이고 효율적인 구매를 위해 에이전트 기술이 도입될 수 있다.

현재 전자 상거래 분야에 적용된 대표적인 에이전트 시스템으로 BargainFinder, Jango[18]과 같은 시스템을 예로 들 수 있다. 이 시스템들은 비교 검색 시스템으로 구매하고자 하는 특정 상품을 각 상점별로 비교하게 된다. 이러한 구매 에이전트를 가동시킴으로써 사용자는 직접 상품을 검색해야 하는 불편함을 해소할 수 있으며, 보다 효율적인 검색 결과를 얻을 수 있다.

그러나, 에이전트가 검색해 온 정보에 대한 신뢰를 확립하기 위해서는 에이전트를 보호하기 위한 방안이 적용되어야 한다. 예를 들어, 사용자가 항공권을 구매하고자 하여 에이전트를 통해 시간, 목적지, 가격에 대한 조건을 지시한 후 정보를 수집해 오도록 할 경우, 에이전트는 인터넷상의 웹사이트를 돌아다니며 사용자가 제시한 조건에 부합하는 항공권에 대한 정보를 수집하게 된다. 그러나, 에이전트가 수집한 결과에 대하여 사용자가 신뢰하고 결과에 따른 항공권 구매를 수행하기 전에 몇 가지 고려해야 할 사항이 있다. 에이전트의 수행 결과가 다른 웹사이트에 의해 조작되지 않았는지, 에이전트가 가장 적절한 것으로 결정한 결과보다 더 나은 결과가 존재하지 않는지에 대해 확신할 수 있어야 한다. 제안 방안 중 수행 결과 보호 방안을 적용할 경우 에이전트가 수행한 결과에 대한 변경이 있었는지 검증하는 것이 가능하다.

6. 결론

현재 대부분의 네트워크 상에서 사용되고 있는 클라이언트/서버 시스템은 클라이언트와 서버 사이에서 지속적인 상호 작용을 요구하기 때문에 대역폭 소모가 많고, 전송이 지연되거나 통신비용이 많이 요구되는 등 여러 가지 문제점을 가지고 있어 차세대 분산 컴퓨팅 및 무선 인터넷 분야에서 이 패러다임을 수용하기에는 무리가 있다.

이동 에이전트 시스템은 에이전트의 직접적인 이동을 전제로 하고 있으며, 기존 인터넷 환경에서 사용하고 있는 클라이언트/서버 시스템의 취약점을 해결하기 위한 방안으로 제안되었다. 특히, 다양하고 이질적인 시스템에서의 유연한 대처, 독립적인 여러 에이전트 사이에서의 통신 및 협력이 가능하여 분산 처리의 효율성을 높일 수 있을 것으로 기대되고 있다.

그러나, 이동 에이전트 시스템은 보안성 측면에서 해결해야 할 문제를 가지고 있기 때문에 폭넓은 분야에서 응용되기 위한 충분한 신뢰감을 조성하지 못하고 있다.

본 논문에서는 그러한 보안 문제를 해결하기 위한 방안으로 이동 에이전트의 안전한 이전과 수행한 결과를 보호하기 위한 기법을 제안하였다. 이를 위하여 첫째, 이동 코드 기반 에이전트 시스템의 개념과 동작에 대하여 설명하고, 기존에 제시된 Java 기반 이동 에이전트 시스템의 특징을 비교하고, 안전성 측면에서 분석하였다. 둘째, 에이전트 시스템에 가해질 수 있는 위협을 세분화하여 살펴보고, 에이전트 시스템에 적용 가능한 보호 기술로써 암호학 기반 기술과 시스템 기반 기술로 나누어 살펴보았다. 또한, 에이전트를 보호하기 위해 제시된 보호 기법을 예방 메커니즘과 검출 메커니즘으로 분류하여 설명하였다. 셋째, 에이전트를 주고받는 호스트 사이에서의 상호 인증, 통신 데이터의 무결성과 비밀성을 제공하는 안전한 이전을 위한 에이전트 이전 보호 기법과 정직한 호스트와 이동 에이전트의 소유자가 이동 에이전트의 수행 결과의 변경 유무를 판단할 수 있도록 설계된 수행 결과 보호 기법을 제안하였다. 또한, 수행 결과 보호 프로토콜은 전송되는 수행 결과의 비밀성과 전방

향 무결성, 프로토콜 참여에 대한 부인 방지를 제공한다. 마지막으로, 이동 에이전트 시스템의 응용 분야로 능동 보안에 대한 세부 사항을 알아본 후 적용 방안을 제안하였다.

무선 네트워크 기술의 발전과 상용화와 맞물려 이동 에이전트 시스템의 활용 영역도 넓어질 것으로 기대되고 있으며, 현재 많은 이동 에이전트 시스템이 개발되고 여러 응용 분야에서 적용되고 있기는 하지만, 보다 폭넓은 발전을 위해서는 에이전트 시스템의 안전성 문제 해결이 필수적이다.

에이전트 시스템의 수행 특성 상, 에이전트 사용자는 에이전트의 수행 결과에 대하여 전적으로 신뢰해야 하기 때문에, 에이전트 자체에 대한 보호 방안 연구가 시급하다. 특히, 에이전트 시스템을 보호하기 위한 방안은 기존의 호스트 또는 네트워크 상에서의 공격에 대한 보호 방안을 적용할 경우 대부분 해결이 가능한 반면, 이동 에이전트 자체를 보호하기 위한 방안에 대한 연구는 아직 미흡한 실정이므로, 본 논문의 제안 방안이 에이전트를 보호하기 위한 연구에 도움이 될 것으로 기대한다. 보안 시스템의 전체 보안 강도를 결정짓는 것은 그 시스템의 가장 취약한 부분이므로, 이동 에이전트 시스템에 있어서는 에이전트를 보호하기 위한 방안에 대한 연구가 계속되어야 할 것으로 판단된다.

참고 문헌

[1] J. Canavan, "Fundamentals of Network Security," Artech House. INC, 2001

[2] D. Kotz and R. S. Gray, "Mobile Agents and the Future of the Internet," In ACM Operating Systems Review, p.7-13, August 1999

[3] M. S. Greenverg, J. C. Byington and D. G. Harper, "Mobile Agents and Security," IEEE Communications Magazine, Volume: 36 Issue: 7, P.76-85, July 1998

[4] A. Lingnau and O. Drobnik, "AN INFRASTRUCTURE FOR MOBILE AGENTS: REQUIREMENTS AND ARCHITECTURE," Proceedings 13th DIS workshop, S. 295-303, September 1995

[5] 신 원, "안전한 이동 에이전트 시스템의 설계와 응용," 부경대학교 대학원 박사 학위논문, 2001

[6] http://www.trl.ibm.com/aglets/index_e.htm

[7] http://www.recursionsw.com

[8] http://www.cs.umn.edu/Ajanta/

[9] http://mole.informatik.uni-stuttgart.de

[10] 박창섭, "암호이론과 보안," 대영사, 1999

[11] S. Oaks, "Java Security." O'reilly, May 2001

[12] G. Vigna, "Cryptographic Traces for Mobile Agents," In: G. Vigna(Ed.), Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science 1419, pp.137-153, 1998

[13] W. M. Farmer, J. D. Guttmann and V. Swarup, "Security for Mobile Agents: Authentication and State Appraisal," Proceedings of the Fourth European Symposium on Research in Computer Security, 1996

[14] T. Sander and C. Tschudin, "Protecting Mobile Agents Against Malicious Hosts," IN: G. Vigna(Ed.), Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science 1419, pp.44-60, 1998

[15] T. Sander and C. Tschudin, "Towards Mobile Cryptography," International Computer Security Institute(ICSI), TR-97-049, 1997

[16] G. Karjoth, N. Asokan and C. Gulcu, "Protecting the Computation Results of Free-roaming Agents," Proceedings of the Second International Workshop, MA'98, pp.195-207, 1998

[17] DALPA, DALPA Information Survivability Program, http://www.darpa.mil/ito/research/is

[18] R. B. Doorenbos, O. Etzioni and D. S. Weld, "A Scalable Comparison-Shopping Agent for the World-Wide Web," Proceedings of the first international conference on Autonomous agents, pp. 39-48, February 1997

감사의 글

이 글을 적고 있자니 여러 가지 생각을 품고 시작했던 석사 2년 간의 생활이 정말로 끝났구나 하는 기분이 듭니다. 시작과 끝의 자리에서 늘 많은 생각을 하게 되지만, 지금은 그 어느 때보다 복잡한 심정입니다. 여기까지 오면서 너무 부족한 것이 많아서 교수님과 선배들에게 아직 배울 것이 남았는데...하는 아쉬움이 남습니다.

그리고, 감사하다고 이름 불러보고 싶은 분이 정말 많이 떠오르네요. 우선 정말 배울 점도 많고, 멋진 지도 교수님... **이경현 교수님**께 제일 먼저 감사 드립니다. 얼굴 뵐 시간이 길지 않아 좋은 모습 많이 보여 드리지 못한 점이 늘 아쉽습니다. 다음으로, 부족한 논문을 심사해주신 박만곤 교수님, 여정모 교수님, 정순호 교수님께 감사 드립니다. 석사 2년 동안 수업을 지도해주신 김창수 교수님, 박승섭 교수님, 박홍복 교수님, 김영봉 교수님, 윤성대 교수님, 박지환 교수님께도 감사 드립니다.

연구실에서 고마운 분들 이름 부를 차례네요^^ **신원 선배**, 공부든 생활이든 너무 배울 점이 많은 훌륭한 선배라고 생각합니다. 졸업하기까지 정말 많은 도움을 주셨죠. **정화 선배**, 한 배를 타고서 부족한 후배 데리고 마음 고생 많으셨죠? 공부하는 것도 생활하는 것도 선배한테 참 많이 배운 거 같아요. **준석 선배**, **종필 선배**, 늘 활기찬 연구실 분위기를 만들어 주셨죠? 세미나 시간에도 좋은 말씀 많이 해 주셔서 정말 도움이 많이 됐었어요. 그리고, 연구실을 이끌어 나아가시느라 수고하셨다고도 말씀드리고 싶어요. **수정 선배**, 늘 좋은 말씀 많이 해주셔서 참 도움이 많이 됐습니다. **영호 선배**, 지낼수록 정말 재미있고 좋은 분이라고 생각해요. **현호 선배**, 말이 필요 없을 만큼 좋은 선배^^ 서철 선배, 얼굴 좀 자주 봐요... **지철 오빠**랑 **명진 오빠**, 동기로서 2년을 함께 했는데 헤어지려니 섭섭하네요. **성렬씨**, 귀염등이죠? 정이 많은 거 같아요. 항상 농담하면 재미있었는데... **영경이**, 싹싹한 성격 때문에 많이 친해진 거 같아. 너 좋아하는 거 알지? 나 없어도 부디 행복해야 돼. **미선이**, 늘 남의 말을 귀담아 들어주는 게 젤로 맘에 들었달까? 하지만 그것말고도 장점이 많은 후배. **지원씨**, **인경이**, 시작하는 공부 열심히 하고, 원하는 만큼 배워갈 수 있

길 빌어요. 막내 **화경이**, 활달한 웃음과 동그란 얼굴로 늘 열심히 하는 모습이 보기 좋았어. 모두 모두 감사합니다. 그리고, 우리 연구실 산·교대 선생님들께도 정말 감사하단 말 전합니다. 비록 제가 졸업하고 연구실을 떠나더라도 모두 슬퍼하지 마세요. 잘 되어서 종종 들를게요.

사랑하는 내 친구들, **경수, 은화, 수란, 윤희, 혜은, 성진, 창우, 창환, 영찬**, 그리고, 지금은 곁에 없지만, 언젠가 만나게 될 **종원**... 고마워, 나랑 친구 해줘서^^ 우리가 함께 한 10 여 년의 시간은 이제 시작이라고 생각해. 모두들 앞으로도 늘 내 곁에 있어주길 바래.

무엇보다 소중하고 감사한 존재인 내 가족, 여기까지 올 수 있도록 허락해 주시고 이끌어 주신 세상에서 가장 존경하는 **아빠 김병칠씨, 엄마 류숙희씨**, 늘 건강하시고 부디 오래 사세요. 앞으로 많이많이 효도할 수 있게... 항상 아빠, 엄마 반만큼만 노력해도 성공하는 삶 일거라고 생각합니다. 여기까지 봐 주셔서 너무 감사하고, 앞으로도 많이 지도해주세요. **인환 오빠, 희옥 동생**, 티격태격 많이 해도 정말 소중한 형제, 자매. 나보다 더 많이 성공하기 바래. 나중에 나 먹여 살려야 하니까. **종길 선배**, 항상 곁에서 챙겨주는 고마운 선배, 내가 정말 좋아하는 사람...

여기에서 이름 부른 모든 분들과 미처 이름 부르지 못한 내가 아는 모든 분들에게 정말 감사하다는 말 전하고 싶습니다. 앞으로도 희연이가 잘 해 나아가는지 지켜봐주세요.

한 페이지면 될 줄 알았는데, 이렇게 길어졌네요 ^^* 저는 이제 사회 생활을 시작하고자 합니다. 물론 취직됐을 때 얘기지만요. 지금 이 시점이 너무나 큰 변화를 부르는 시간이란 생각이 듭니다. 내가 그 변화를 따라갈 수 있을까, 따라잡을 수 있을까 걱정도 많이 됩니다. 하지만, 제가 감사하게 생각하는 모든 분들이 가끔 제 생각해주시고, 힘내라고 마음속으로 한 마디 해주시면, 정말 잘 해 나 갈 수 있을 거 같은 생각이 드네요. 한 가지 확실한 건 우리 연구실에 들어와서 지낸 2년의 시간은 후회하지 않을 만큼 소중한 시간이었고, 좋은 사람들과 함께 지낼 수 있어서 무척 행복했다는 겁니다. 모두들 행복하세요. *2003년 1월 9일 새벽에...*