이동 에이전트 환경에서의 연속된 위임을 위한 내포된 토큰 기반 위임 기법

정회원 권 혁 만*, 김 문 정*, 정회원 엄 영 익*

A Nested Token-Based Delegation Scheme for Cascaded Delegation in Mobile Agent Environments

Hyeog-Man Kwon*, Moon-Jeong Kim*, Young Ik Eom* ***Regular Members***

요 약

이동 에이전트 환경에서는 에이전트의 이동성으로 인하여 플레이스간에 연속된 위임(cascaded delegation)이 빈번하게 발생한다. 이동 에이전트 환경에서의 연속된 위임은 에이전트가 셋 이상의 플레이스를 이주하면서 플레이스간에 권한을 위임하는 과정으로서 정의된다. 이동 에이전트 환경에서 위임에 대한 대표적인 연구로는 Berkovits의 연구가 있다. 이 연구에서는 에이전트의 안전한 이주를 위하여 에이전트를 실행하고 있는 플레이스와 에이전트가 이주하게 될 플레이스간에 주고받아야 할 메시지만을 정의하고 있다. 그러나 이 연구는 에이전트의 이주와 관련된 두 플레이스만을 위임의 대상으로 고려하기 때문에, 셋 이상의 플레이스를 대상으로 하는 연속된 위임을 위한 연구로는 부적절하다. 즉, 연속된 위임을 안전하게 수행하기 위해서는 연속된 위임에 참여하는 모든 플레이스들이 주고받는 메시지간의 관계 형성이 필수적이나, 이러한 관계 형성 과정이 존재하지 않는다. 본 논문에서는 이동 에이전트 환경에서 연속된 위임을 안전하게 수행하는 위임 기법을 제안한다. 제안 기법은 각 위임토큰(delegation token)을 다음에 생성되는 위임토큰에 내포시킨 후 서명하는 방법을 사용한다. 또한, 제안 기법이 재연(replay)에 의한 공격 및 위임토큰의 치환에 의한 공격으로부터 안전함을 증명한다.

Key Words : delegation; mobile agents; distributed systems; security

ABSTRACT

In mobile agent environments, cascaded delegations among places occur frequently due to the mobility of agents. Cascaded delegation in mobile agent environments can be defined as the process whereby the delegated place delegates the rights of the delegating place further. The representative study for delegation in mobile agent environments is Berkovits et al.'s study. Their study only defines the messages that is sent between the place executing the agent and the place where the agent migrates. Because their study considers only the delegation between two places which participate in migration of an agent, it is inadequate in the situation that the cascaded delegation is necessary. In other words, the relationships among the messages sent from and to places is necessary. However, their study does not exist the relationships. In this paper, we propose a delegation scheme that provides agents with secure cascaded delegation. The proposed scheme achieves the goal by nesting each delegation token within the signed part of the next immediate delegation token. We prove that the proposed scheme is secure against the attack of replaying a message and of substituting a delegation token.

* 성균관대학교 정보통신공학부 분산컴퓨팅 연구실({bestis, top, yieom}@ece.skku.ac.kr),
논문번호 : 020359-0816, 접수일자 : 2002년 8월 16일
※본 논문은 과학기술부 프론티어사업의 유비쿼터스컴퓨팅 및 네트워크원천기반기술개발 과제로 수행된 결과입니다.

Ⅰ. 서 론

최근 이동 에이전트는 분산 시스템의 RPC (Remote Procedure Call)나 메시지 전달 (message passing)과 같은 정적인 프로세스간의 원격 통신 방식을 확장시킬 수 있는 새로운 대안으로 주목받고 있다. 이동 에이전트는 여러 호스트들을 자율적으로 이동하면서, 다른 에이전트들과 상호작용을 하거나 또는 호스트에 의해 제공되는 여러 자원들을 이용하면서 사용자의 작업을 수행할 수 있는 프로세스로서 정의된다. 이동 에이전트는 기존의 클라이언트/서버 구조에 비할 때, 네트워크의 트래픽 감소, 클라이언트와 서버간의 비동기 연산 지원, 서비스의 분산 및 병렬 처리, 동적인 서버 인터페이스의 변경 지원 등의 많은 장점들을 가진다[1][2][3].

그러나 이동 에이전트의 많은 장점들에도 불구하고, 에이전트의 이동성으로 인해 야기되는 심각한 보안 취약점들은 이동 에이전트를 실제 응용에 적용하는데 있어 큰 장애 요소들로 작용하고 있다. 현재 보안 취약점들을 정의하고 해결하기 위한 연구들이 진행되고 있으나, 대부분 에이전트 및 호스트의 인증 문제 또는 에이전트 및 호스트 보호 문제에 초점을 두고 있다[2][3][4][5][6].

본 논문에서는 이동 에이전트 환경에서 빈번하게 발생되는 연속된 위임을 안전하게 수행할 수 있는 위임 기법을 제시한다. 위임(delegation)은 한 주체(principal)가 다른 주체에게 자신을 대신할 수 있도록 전부 또는 일부의 권한을 부여하는 과정으로 정의되며, 연속된 위임(cascaded delegation)은 셋 이상의 주체간에 위임이 연속적으로 수행되는 과정으로 정의된다[7][8]. 이동 에이전트 환경에서의 위임에 대한 대표적인 연구로는 Berkovits의 연구를 들 수 있으나, 이 연구는 두 플레이스간의 위임만을 고려하고 있다[2]. 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 이동 에이전트의 시스템 구조와 이동 에이전트 환경에서의 두 플레이스간의 위임에 대해 알아본다. 또한, 분산 환경에서의 연속된 위임에 대해 설명한다. 3장에서는 본 논문에서 제안하는 이동 에이전트 환경을 위한 내포된 토큰 기반의 위임 기법에 대해 설명하고, 시나리오를 통해 제안 기법의 구체적인 동작 과정에 대해 알아본다. 4장에서는 제안 기법의 안전성을 증명한다. 마지막으로, 5장에서는 결론 및 문제점들을 지적하고 향후 연구 과제에 대해서 기술한다.

Ⅱ. 관련 연구

이동 에이전트 환경에서의 위임을 위한 대표적인 연구로는 Berkovits의 연구가 있다[2]. 이 연구에서는 이동 에이전트 환경에서 에이전트의 실행 주체(executing principal)를 정의하기 위하여, 이주와 관련된 주체간의 신뢰 관계를 분류한 후, 각 주체간에 주고받아야 할 메시지들을 정의한다. 그러나 이 연구는 이주와 관련된 두 플레이스간의 위임만을 대상으로 하기 때문에, 셋 이상의 플레이스를 대상으로 하는 연속된 위임에는 부적절하다. 본 장에서는 먼저 이동 에이전트의 시스템 구조에 대해 소개하고, 이동 에이전트 환경에서의 두 플레이스간의 위임 기법에 대해서 알아본 후, 분산 환경에서의 연속된 위임에 대해 설명한다.

2.1 이동 에이전트 시스템

그림 1은 일반적인 이동 에이전트 시스템 구조를 보인다.

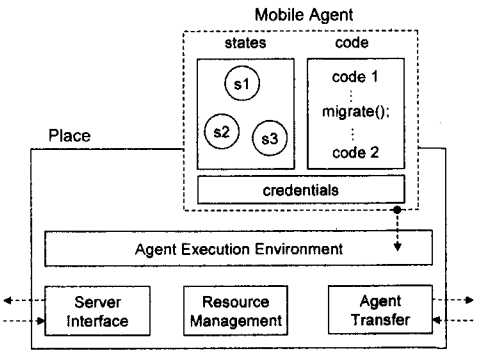


그림 1. 이동 에이전트 시스템 구조

그림 1과 같이, 이동 에이전트 시스템은 이동 에이전트와 플레이스로 구성된다. 에이전트는 실행 코드(code), 실행 결과들을 저장하는 상태값(states), 에이전트의 생성자 및 소유자의 정보를 담는 크리덴셜(credentials) 등으로 구성된다. 크리덴셜은 위조를 방지하기 위하여 소유자의 비밀키로 서명된다. 또한, 플레이스는 에이전트에게 호스트의 자원 및 통신 서비스들을 제공하는 실행 환경(execution

environment)으로서, 플레이스간의 인터페이스 역할을 담당하는 인터페이스(Server Interface) 모듈, 호스트의 자원을 관리하는 리소스 관리(Resource Management) 모듈, 에이전트의 실행을 담당하는 에이전트 실행 환경(Agent Execution Environment) 모듈, 에이전트의 전송을 담당하는 에이전트 전송(Agent Transfer) 모듈 등으로 구성된다.

2.2 이동 에이전트 환경에서 두 플레이스간 위임

1998년, Berkovits는 이동 에이전트 환경을 위한 인증 기법을 제시하고, Lampson의 형식 이론(formal theory)[9][10]을 기반으로 하여 제시한 인증 기법에 대한 안전성을 증명하였다[2]. 이 연구에서는 에이전트의 실행 주체가 에이전트의 이주로 인하여 자주 변경된다는 것에 주목하고, 이주와 관련된 주체간의 신뢰 관계에 따라 주체간에 주고받아야 할 메시지들을 정의하였다. 플레이스 I_1 상에서 주체 P_1으로서 실행 중인 에이전트 A가 새로운 주체 P_2로서 자신을 실행시키기 위해 플레이스 I_2로 이주한다고 할 때, 에이전트의 이주와 관련된 주체간의 신뢰 관계는 다음과 같이 4가지로 분류된다.

1) 플레이스 핸드오프(place handoff): 플레이스 I_1이 플레이스 I_2로 에이전트를 핸드오프하는 관계이다. 플레이스 핸드오프는 이주와 관련된 두 플레이스간의 신뢰도가 가장 높은 신뢰 관계로서, 에이전트를 실행 중인 플레이스가 에이전트를 다음 플레이스로 핸드오프하는 것이다. 즉, 이주가 일어난 직후의 플레이스 I_2는 이주 직전의 에이전트 실행 주체 P_1을 대신하여 에이전트를 실행하게 된다.
2) 플레이스 위임(place delegation): 플레이스 I_1이 플레이스 I_2로 에이전트를 위임하는 관계이다. 플레이스 위임은 에이전트를 실행 중인 플레이스가 에이전트를 다음 플레이스로 위임하는 것이다. 즉, 이주가 일어난 직후의 플레이스 I_2는 이주 직전의 에이전트 실행 주체 P_1의 실행 권한에 자신의 권한을 추가하여, 에이전트를 실행하게 된다.
3) 에이전트 핸드오프(agent handoff): 에이전트가 직접 플레이스 I_2로 자신을 핸드오프하는 관계이다. 에이전트 핸드오프는 에이전트가 실행 중에 직접 자신을 다음 플레이스로 핸드오프하는 것을 의미한다. 즉, 이주가 일어난 직후의 플레이스 I_2는 이주 직전의 에이전트 실행 주체 P_1에 상관없이, 에이전트 A를 대신하여 에이전트를 실행하게 된다.
4) 에이전트 위임(agent delegation): 에이전트가 직접 플레이스 I_2로 자신을 위임하는 것이다. 에이전트 위임은 에이전트가 실행 중에 직접 자신을 다음 플레이스로 위임하는 것을 의미한다. 즉, 이주가 일어난 직후의 플레이스 I_2는 이주 직전의 에이전트 실행 주체 P_1과 상관없이, 에이전트 자체의 권한에 자신의 권한을 추가하여 에이전트를 실행하게 된다.

표 1은 각 신뢰 관계에 따라 주체간에 주고받아야 할 메시지들을 보인다.

표 1. 신뢰 관계에 따라 요구되는 메시지

신뢰관계	요구되는 메시지
플레이스 핸드오프	$Cert((I_1 \mid P_1) \rightarrow P_1)$ $Migrate(I_1, A, P_1, I_2, HO, t)$
플레이스 위임	$Cert((I_1 \mid P_1) \rightarrow P_1)$ $Migrate(I_1, A, P_1, I_2, \nabla, t)$ $Response(I_2, A, P_1, I_1, t_2)$
에이전트 핸드오프	the sender-signed *PPL* or *PPC*
에이전트 위임	the sender-signed *PPL* or *PPC* $Response(I_2, A, P_1, I_1, t_2)$

표 1에서 $Cert((I_1|P_1) \rightarrow P_1)$는 플레이스 I_1이 주체 P_1로서 에이전트를 실행하고 있음을 나타낸다. *Migrate* 메시지는 플레이스가 에이전트의 이주를 요청할 때 사용하며, 이주를 요청한 플레이스의 식별자, 에이전트의 식별자, 이주 직전의 에이전트 실행 주체의 식별자, 이주할 플레이스의 식별자, 신뢰 관계 타입, 타임스탬프 등으로 구성된다. *Response* 메시지는 플레이스가 플레이스 위임 또는 에이전트 위임에서의 *Migrate* 메시지에 대해 응답할 때 사용하며, 이주 직후의 플레이스의 식별자, 에이전트의 식별자, 이주 직전의 에이전트 실행 주체의 식별자, 이주 직전의 플레이스의 식별자, 타임스탬프 등으로 구성된다. *PPL*(Place Permission List)과 *PPC*(Place Permission Certificates)는 에이전트의 송신자에 의해 이주가 허용되는 플레이스들의 목록 및 인증서들을 나타내며, 송신자에 의해 서명된 후, 에이전트에 포함되어 전송된다.

위에서 살펴본 바와 같이, Berkovits의 연구는 신뢰 관계의 정의에 있어서 에이전트를 실행하고 있는 플레이스와 에이전트가 이주하게 될 플레이스간에 주고받아야 할 메시지만을 정의하고 있다. 그러나 이와 같이 두 플레이스간의 위임만을 고려한 방식은 셋 이상의 플레이간의 위임을 대상으로 하는 연속된 위임을 수행하는 과정에 있어서 메시지 치환에 의한 공격에 약점을 가지게 된다[11][12][13][14]. 즉, 연속된 위임을 안전하게 수행하기 위해서는 연속된 위임에 참여하는 모든 플레이스들이 주고받는 메시지간의 관계 형성이 필수적이나, Berkovits의 연구에서는 이러한 관계 형성 과정이 존재하지 않는다. 따라서 Berkovits의 연구는 이동 에이전트 환경에서의 연속된 위임을 위한 연구로는 부적절하다.

2.3 연속된 위임(cascaded delegation)

분산 환경에서의 연속된 위임은 셋 이상의 주체간에 위임이 연속적으로 수행되는 과정으로 정의된다[7][8]. 그림 2는 수여자(Grantor) *A*에 의해 위임이 시작되어 피수여자(Grantee) *B*, *C*로 연속된 위임이 일어난 후, 종단 서버 *S*로 서비스를 요청하는 과정을 보인다.

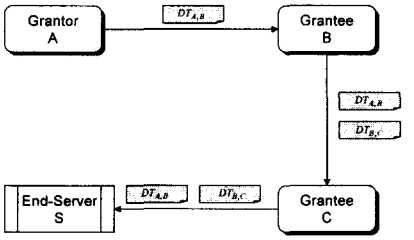


그림 2. 연속된 위임의 동작 과정

그림 2는 *ABCS*를 위임경로(delegation path)로서 가정하고 있다. 위임경로는 위임을 시작한 주체부터 종단 서버까지 위임과 관련된 모든 주체들을 시간순으로 적어놓은 시퀀스(sequence)이다. *A*는 *S*로부터 서비스를 제공받기 위하여, 먼저 위임토큰 $DT_{A,B}$를 생성하여 *B*로 전송한다. 위임토큰 $DT_{A,B}$는 *A*가 *B*로 위임할 권한 및 타임스탬프 등을 내용으로 한다. *A*로부터 $DT_{A,B}$를 받은 *B*는 *C*로 위임할 권한을 담은 $DT_{B,C}$를 생성하여 $DT_{A,B}$와 함께 *C*로 전송한다. 마지막으로, $DT_{A,B}$와 $DT_{B,C}$를 받은 *C*는 제공받을 서비스와 함께 두 토큰들을 종단 서버 *S*로 전송함으로써 *S*로 서비스를 요청하게 된다. 이 때, 만약 위임토큰간의 어떠한 관계 형성 없이 위임토큰들을 네트워크 상으로 전송한다면, 위임토큰 치환 등의 공격으로 인하여 위임경로가 변경될 수 있다. 따라서 안전하게 연속된 위임을 수행하기 위해서는, 위임토큰간에 관계를 맺은 후 전송해야 한다[11][12][13][14]. 본 연구에서는 이동 에이전트 환경에서의 연속된 위임을 안전하게 수행하기 위한 위임 기법을 제시한다.

Ⅲ. 제안 기법

본 연구에서는 이동 에이전트 환경에서 안전한 연속된 위임을 지원하기 위한 위임 기법을 제안한다. 먼저 본 제안 기법의 시스템 구조와 자료 구조에 대해서 설명하고, 연속된 위임을 지원하는 위임 기법을 제시한 후, 시나리오를 통해 제안 기법의 동작과정을 보인다.

3.1. 시스템 구조

이동 에이전트 환경에서 연속된 위임은 작업의 수행을 위해 에이전트가 셋 이상의 플레이스들을 이주할 때 발생한다. 그림 3은 플레이스 P_1에서 플레이스 P_n까지 이주하면서 각 플레이스들의 권한을 위임받은 에이전트 MA가 위임토큰 DTp_{n-1}을 기반으로 종단 서버 *S*에게 서비스를 요청하는 과정을 보인다.

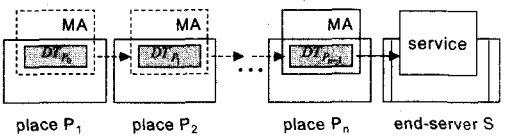


그림 3. 이동 에이전트 환경에서의 연속된 위임

그림 3에서 P_1에서 실행 중인 MA는 P_2로 위임할 권한을 담은 DTp_1을 생성하여 MA에 포함시킨 후, MA를 P_2로 전송한다. MA를 받은 P_2는 P_3로 위임할 권한을 담은 DTp_2를 생성하여 MA에 포함시킨 후, MA를 P_3로 전송한다. 이 때, DTp_2는 DTp_1을 내포한 형식으로 구성되며, P_2의 비밀키에 의해 서명된다. 이와 같은 과정이 반복된 후, MA는 P_n에 도착하게 되고, P_n에 도착한 MA는 DTp_{n-1}과 제공받을 서비스를 *S*로 전송함으로써, *S*로 서비스를 요청하게 된다.

3.2. 자료 구조

본 절에서는 제안 기법에서 사용하게 될 플레이스간의 통신 메시지 구조와 위임토큰의 구조에 대해서 알아본다. 그림 4는 제안 기법에서 사용하는 통신 메시지의 구조를 보인다.

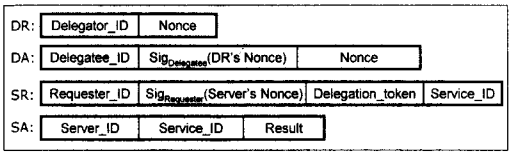


그림 4. 통신 메시지 구조

그림 4에서 *DR* 메시지와 *DA* 메시지는 플레이스간에 권한을 위임할 때 사용된다. *DR* 메시지는 위임자가 피위임자로 에이전트를 이주하고자 할 때 사용되는 메시지로서, 위임자의 식별자(Delegator_ID), 위임자에 의해 발행된 난수(Nonce)로 구성된다. *DA* 메시지는 *DR* 메시지에 대한 응답 메시지로서, 피위임자의 식별자(Delegatee_ID), *DR* 메시지의 난수에 대한 피위임자의 서명($Sig_{Delegatee}$(DR's Nonce)), 그리고 피위임자에 의해 발행된 난수(Nonce)로 구성된다. 한편, *SR* 메시지와 *SA* 메시지는 요청자와 종단 서버간의 서비스 요청 및 응답에 사용된다. *SR* 메시지는 요청자가 종단 서버에 서비스를 요청할 때 사용되는 메시지로서, 요청자의 식별자(Requester_ID), 종단 서버에 의해 발행된 난수에 대한 요청자의 서명($Sig_{Requester}$(Server's Nonce)), 위임토큰, 서비스의 식별자로 구성된다. *SA* 메시지는 *SR* 메시지에 대한 응답 메시지로서, 서버의 식별자(Server_ID), 서비스의 식별자(Service_ID), 서비스의 수행 결과(Result)로 구성된다.

주체간의 위임에 대한 상세한 정보를 기술하는 위임토큰을 위한 자료 구조는 그림 5에서 보인다.

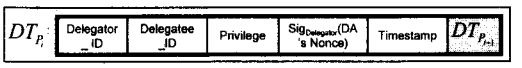


그림 5. 위임토큰(DTp_i)의 구조

그림 5에서 DTp_i는 위임자의 식별자(Delegator_ID), 피위임자의 식별자(Delegatee_ID), 위임자가 피위임자에게 부여하는 권한(Privilege), *DA* 메시지의 난수에 대한 위임자의 서명($Sig_{Delegator}$(DA's nonce)), 위임의 유효 시간을 나타내는 타임스탬프(Timestamp), 이전의 위임토큰(DTp_{i-1})으로 구성된다. 이전의 위임토큰을 내포시키는 방식은 악의적인 플레이스에 의한 위임토큰 치환 공격으로부터 위임토큰을 보호한다. 또한, 위임토큰에 대한 위조를 방지하기 위하여, 위임토큰은 위임자의 비밀키로 서명된다.

3.3. 알고리즘

본 논문에서는 $P_1P_2P_3\cdots P_nS$를 위임경로로서 가정하고 있으며, 이들간의 연속된 위임을 안전하게 수행하기 위하여 내포된 토큰 기반의 위임 기법을 제안한다. 즉, 플레이스 P_1에서 작업을 수행 중인 에이전트 *MA*는 종단 서버 *S*로부터 서비스를 제공받기 위하여 플레이스 P_2, P_3, …, P_{n-1}, P_n을 순서대로 거치면서 필요한 권한을 내포된 토큰 형태로 위임받게 되고, 플레이스 P_n에 도착한 후 위임받은 토큰을 기반으로 *S*에게 서비스를 요청한다. 제안 기법은 연속된 위임을 시작하는 P_1이 수행하는 알고리즘, 연속된 위임을 중재하는 $P_i(2 \le i \le n-1)$가 수행하는 알고리즘, 종단 서버 *S*로 서비스를 요청하는 P_n이 수행하는 알고리즘으로 나뉜다.

알고리즘 1은 연속된 위임을 시작하는 P_1이 수행하는 작업절차를 보인다.

알고리즘 1. 플레이스 P_1의 연속된 위임 초기화

```
dispatch and execute MA with DT_{F_0};
// DT_{F_0} is the token of MA's creator
if migrate(P_2) is executed in MA
  then
 { send DP(P_1, N_{F_1}) message to P_2;
   wait until DA(P_2, Sig_{F_2}(N_{P_1}), N_{F_2})
                    message is sent from P_2;
   auth_result ← verify(Sig_{F_2}(N_{F_1}));
                    // Sig_{F_2}(N_{F_1}) is in DA
   if auth_result is false
     then discard MA;
          // terminate cascaded delegation
   DT_{F_1} ← createDT(P_1, P_2, Priv_{F_1}(P_2),
                    Sig_{F_1}(N_{F_2}), t_{F_1}, DT_{F_1});
   send MA with DT_{F_1} to P_2;
 }
```

알고리즘 1에서 P_1은 DTp_0를 포함한 *MA*를 디스패취한 후 실행한다. 만약 *MA*의 실행 중에 P_2로의 이주가 요구되면, P_1은 자신의 식별자와 난수 Np_1로 구성된 *DR* 메시지를 P_2로 송신한 후, P_2로

부터의 *DA* 메시지 수신을 기다린다. 이 때, P_2로부터의 *DA* 메시지 수신이 정상적으로 이루어지면, P_1은 *DA* 메시지의 $Sigp_2(Np_1)$를 검증하게 된다. 검증 결과, *DA* 메시지의 $Sigp_2(Np_1)$가 옳지 않은 경우는 P_2에 대한 인증이 실패했음을 나타낸다. 따라서 이 경우에는 P_1은 *MA*의 수행을 중단하고, 연속된 위임을 종료하게 된다. 반면, *DA* 메시지의 $Sigp_2(Np_1)$가 옳은 경우는 P_2에 대한 인증이 성공했음을 나타낸다. 인증이 성공한 경우, P_1은 P_2로 부여할 위임토큰 DTp_1을 생성한 후, 자신의 비밀키로 서명한다. DTp_1은 P_1의 식별자, P_2의 식별자, P_1이 P_2로 부여할 권한, Np_2에 대한 P_1의 서명, 위임토큰의 타임스탬프, *MA* 생성자의 위임토큰 DTp_0로 구성된다. P_1은 P_2로 서명된 DTp_1을 포함한 *MA*를 전송함으로써, 연속된 위임이 시작된다.

알고리즘 2는 연속된 위임을 중재하는 $P_i(2 \le i \le n-1)$가 수행하는 작업절차를 보인다.

알고리즘 2는 위임경로 $P_1P_2P_3 \cdots P_nS$상의 위임을 중재하는 $P_i(2 \le i \le n-1)$들을 대상으로 하고 있으며, P_{i-1}로부터 *DR* 메시지를 수신할 때 수행된다. 알고리즘 2의 구체적인 동작 과정은 다음과 같다. P_i가 P_{i-1}로부터 P_{i-1}의 식별자 및 난수 Np_{i-1}로 구성된 *DR* 메시지를 받은 경우, P_i는 자신의 식별자, Np_{i-1}에 대한 P_i의 서명, 난수 Np_i로 구성된 *DA* 메시지를 P_{i-1}로 전송한다. 만약, P_{i-1}의 P_i에 대한 인증이 수락된 경우, P_i는 P_{i-1}로부터 DTp_{i-1}을 포함한 *MA*를 받게 된다. 이 때, P_i는 자신이 보낸 *DA* 메시지의 Np_i에 대한 P_{i-1}의 서명이 정상적으로 이루어졌는지를 알기 위해 *MA*의 위임토큰 DTp_{i-1}으로부터 $Sigp_{i-1}(Np_i)$를 추출하여 검증한다. 검증 결과, 서명이 정상적으로 이루어진 경우, P_i는 *MA*의 실행을 재개한다. 만약 *MA*의 수행 중에 P_{i+1}로의 이주가 요청된 경우에는, 알고리즘 1과 유사한 방식으로 연속된 위임을 수행한 후, P_{i+1}로 *MA*를 이주시키게 된다. 이와 같은 방식으로 P_2부터 P_{n-1}까지의 위임 중재 과정이 성공적으로 수행된 경우, *MA*는 P_n에 도착된다.

알고리즘 3은 위임경로상의 마지막 플레이스인 P_n이 *S*로부터 서비스를 제공받기 위해 수행하는 작업절차를 보인다.

알고리즘 3에서 P_n은 먼저 알고리즘 2의 P_{i-1}를 다

알고리즘 2. 플레이스 $P_i(2 \le i \le n-1)$의 연속된 위임 중재

```
//P_i receives DR(P_{i-1}, N_{P_{i-1}}) message from P_{i-1}

// handling P_{i-1}
send DA(P_i, Sig_{P_i}(N_{P_{i-1}}), N_{P_i}) message to P_{i-1};
if MA with DT_{P_{i-1}} arrived from P_{i-1}
  then
  { auth_result ← verify(Sig_{P_{i-1}}(N_{P_i}));
                 // Sig_{P_{i-1}}(N_{P_i}) is in DT_{P_{i-1}}
    if auth_result is false
      then discard MA;
           // terminate cascaded delegation
  }

resume the execution of MA;

// handling P_{i+1}
if migrate(P_{i+1}) is executed in MA
  then
  { send DR(P_i, N_{P_i}) message to P_{i+1};
    wait until DA(P_{i+1}, Sig_{P_{i+1}}(N_{P_i}), N_{P_{i+1}})
                 message is sent from P_{i+1};
    auth_result ← verify(Sig_{P_{i+1}}(N_{P_i}));
                 // Sig_{P_{i+1}}(N_{P_i}) is in DA
    if auth_result is false
      then discard MA;
           //terminate cascaded delegation
    DT_{P_i} ← createDT(P_i, P_{i+1}, Priv_{P_i}(P_{i+1}),
                 Sig_{P_i}(N_{P_{i+1}}), t_{P_i}, DT_{P_{i-1}});
    send MA with DT_{P_i} to P_{i+1};
  }
```

루는 부분과 유사한 과정을 거쳐서, P_{n-1}로부터 위임토큰 DTp_{n-1}이 포함된 *MA*를 수신한 후, *MA*의 실행을 재개한다. 만약 *MA*의 수행 중에 *S*로의 서비스 요청이 발생된 경우, P_n은 자신의 식별자 및 난수 Np_n으로 구성된 *DR* 메시지를 *S*로 전송한 후, *S*로부터 *DA* 메시지의 수신을 기다린다. 이 때, *S*로부터 *DA* 메시지가 수신된 경우, P_n은 *DA* 메시지의 $Sigs(Np_n)$을 검증한다. 검증 결과, 인증이 성공한 경우, P_n은 *S*로 서비스를 요청하기 위해 *SR* 메시지를 생성하여 *S*로 송신한다. *SR* 메시지는 P_n의 식별자, N_s에 대한 P_n의 서명, 위임토큰 DTp_{n-1}, 서비스의 식별자로 구성된다. *SR* 메시지를 수신한 *S*는 *SR* 메시지의 $Sigp_n(N_s)$와 연속된 위임의 내용이 포함된 DTp_{n-1}의 내용을 검증한다. 검증이 성공하게 되면, *S*는 요청한 서비스를 수행한 후, 수행결과를 *SA* 메시지에 담아 P_n으로 전송하게 된다.

3.4. 동작 시나리오

본 절에서는 제안 기법의 동작 과정을 시나리오

알고리즘 3. 플레이스 P_n의 서비스 요청

```
/* P_n receives DR(P_{n-1}, N_{P_{n-1}}) message
                                  from P_{n-1} */

// handling P_{n-1}
send DA(P_n, Sig_{P_n}(N_{P_{n-1}}), N_{P_n}) message
                                  to P_{n-1};
if MA with DT_{P_{n-1}} arrived from P_{n-1}
  then
  { auth_result ← verify(Sig_{P_{n-1}}(N_{P_n}));
                // Sig_{P_{n-1}}(N_{P_n}) is in DT_{P_{n-1}}
    if auth_result is false
      then discard MA;
       // terminate cascaded delegation
  }
resume the execution of MA;
// requesting a service to S
if request(S,service) is executed in MA
  then
  { send DR(P_n, N_{P_n}) message to S;
    wait until DA(S, Sig_S(N_{P_n}), N_S)
                  message is sent from S;
    auth_result ← verify(Sig_S(N_{P_n}));
    if auth_result is false
      then discard MA;
       // terminate cascaded delegation;
    send SR(P_n, Sig_{P_n}(N_S), DT_{P_{n-1}}, service)
                                  message to S;
    receive SA(S, service, result) message
                                  from S;
  }
```

를 통해 구체적으로 알아본다. 그림 6은 플레이스 *A*, *B*, *C*를 이주하면서 위임을 받은 에이전트 *MA*가 위임토큰을 기반으로 종단 서버 *S*에게 서비스를 요청하는 과정을 보인다.

그림 6에서 *A*는 *MA*를 실행한다. *A*상에서 실행 중인 *MA*가 *B*로의 이주를 요구하면, *A*는 *B*로 자신의 식별자와 난수 N_A로 구성된 *DR* 메시지를 보낸다. *DR* 메시지를 수신한 *B*가 응답 메시지로서 *B*의 식별자, N_A에 대한 *B*의 서명, 난수 N_B로 구성된 *DA* 메시지를 전송하면, *A*는 *DA* 메시지의 $Sig_B(N_A)$를 검사하여 *B*에 대한 인증을 수행하고, 인증이 성공적으로 이루어지면 위임토큰 DT_A를 생성한 후, 자신의 비밀키로 서명된 DT_A를 포함한 *MA*를 *B*로 송신한다. DT_A는 *A*의 식별자, *B*의 식별자, *A*가 B로 위임하는 권한, N_B에 대한 *A*의 서명, DT_A의 타임스탬프, *MA*의 생성자의 위임토큰을 내용으로 한다. *A*로부터 *MA*를 받은 *B*는 DT_A의 $Sig_A(N_B)$에 대한 검증과정을 거친 후, 검증이 성

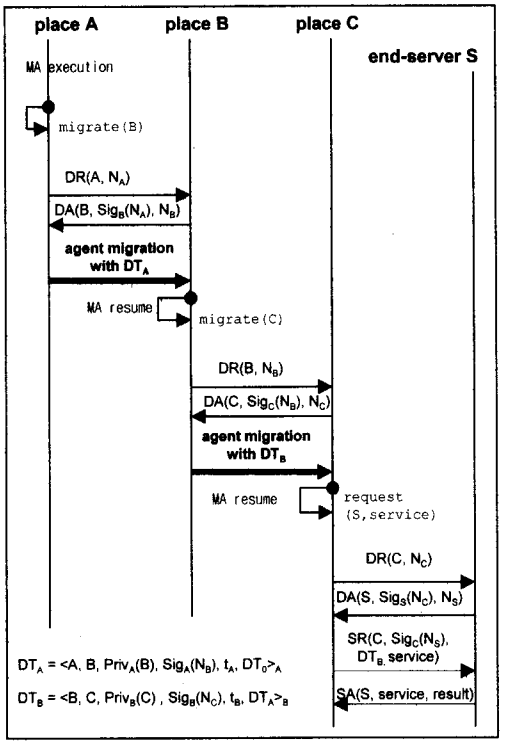


그림 6. 동작 시나리오

공하면 *MA*의 실행을 재개하게 된다. 만약 *B*상에서 실행 중인 *MA*가 *C*로의 이주를 요구하면, *B*는 *A*가 *B*로 *MA*를 전송한 과정과 동일한 과정을 거친 후, *C*로 *MA*를 이주시킨다. *MA*를 수신한 *C*는 위임토큰 DT_B의 $Sig_B(N_C)$에 대한 검증과정을 거친 후, 검증이 성공하면 *MA*의 실행을 재개한다. 이 때, 실행 중인 *MA*가 *S*로 서비스를 요청하면, *C*는 자신의 식별자, 난수 N_C로 구성된 *DR* 메시지를 *S*로 전송하고, *S*의 식별자, N_C에 대한 *S*의 서명, 난수 N_S로 구성된 *DA* 메시지를 수신한다. *DA* 메시지의 $Sig_S(N_C)$에 대한 검증이 성공적으로 이루어지면, *C*는 자신의 식별자, N_S에 대한 *C*의 서명, 위임토큰 DT_B, 요청할 서비스의 식별자로 구성된 *SR* 메시지를 생성하여 *S*로 전송한다. *SR* 메시지를 수신한 *S*는 *SR* 메시지의 $Sig_C(N_S)$와 연속된 위임의 내용이 포함된 DT_B의 내용을 검증한 후, 검증이 성공한 경우 *MA*가 요청했던 서비스를 수행한 후, 수행 결과를 *SA* 메시지에 담아 *C*에게 전송하게 된다.

Ⅳ. 제안 기법에 대한 안전성 증명

이 장에서는 본 연구에서 제시한 이동 에이전트 환경에서의 연속된 위임을 위한 제안 기법이 재연에 의한 공격 및 위임토큰의 치환에 의한 공격으로부터 안전함을 증명한다.

보조정리 1. 인증된 플레이스만이 연속된 위임에 참여할 수 있다.

증 명 에이전트를 현재 실행 중인 플레이스를 P_i라 하자. 만약, 실행 중인 에이전트가 플레이스 P_{i+1}로의 이주를 요구하는 경우, P_i는 P_{i+1}로 $DR(P_i, N_{Pi})$ 메시지를 송신하고 P_{i+1}로부터 $DA(P_{i+1}, Sig_{P_{i+1}}(N_{Pi}), N_{P_{i+1}})$ 메시지를 수신한 후, DA 메시지의 $Sig_{P_{i+1}}(N_{Pi})$를 검증하는 과정을 통해 P_{i+1}에 대한 인증을 수행한다. 반면, P_{i+1}은 P_i로부터 DT_{Pi}가 포함된 에이전트를 수신한 후, DT_{Pi}의 $Sig_{Pi}(N_{P_{i+1}})$를 검증하는 과정을 통해 P_i에 대한 인증을 수행한다. 이 때, 악의적인 플레이스 P_i'가 P_i를 가정하여 $DR(P_i, N_{Pi}')$ 메시지를 P_{i+1}로 전송하는 과정을 통해 연속된 위임의 주체로서 참여를 시도한다고 가정하자. 이 경우, P_i'로부터 $DR(P_i, N_{Pi}')$ 메시지를 수신한 P_{i+1}은 $DA(P_{i+1}, Sig_{P_{i+1}}(N_{Pi}'), N_{P_{i+1}})$ 메시지를 생성하여 P_i'로 전송하고, $Sig_{Pi}(N_{P_{i+1}})$이 포함된 위임토큰 DT_{Pi}가 내장된 에이전트의 수신을 기다린다. 그러나 P_i'은 P_i의 비밀키를 알 수 없기 때문에, $Sig_{Pi}(N_{P_{i+1}})$의 생성이 불가능하게 되고, P_i'의 P_i를 가정한 인증 시도는 실패하게 된다.

제안 기법은 연속된 위임의 수행을 위해 플레이스 간에 위임토큰을 주고받기에 앞서서 DR 메시지와 DA 메시지를 이용하여 플레이스간에 공개키 기반의 인증 과정을 수행하게 된다. 따라서 제안한 기법은 인증된 플레이스만을 연속된 위임의 대상으로 참여시키게 된다.

정리 1. 제안한 위임 기법은 재연(replay)에 의한 공격으로부터 안전하다.

증 명 에이전트 MA가 플레이스 P_1, P_2, P_3, …; P_n을 순서대로 이주하면서 위임을 받은 후, 종단 서버 S로 서비스를 요청한다고 가정하자. 이 경우, 위임경로 $P_1P_2P_3\cdots P_nS$를 따라 연속된 위임이 일어난 후, MA는 P_n에 도착한다. P_n에 도착한 MA는 S로의 서비스 요청을 위해, S로 $DR(P_n, N_{Pn})$ 메시지를 송신하고, S로부터 $DA(S, Sig_S(N_{Pn}), N_S)$ 메시지를 수신한다. 다음으로, P_n은 $SR(P_n, Sig_{Pn}(N_S), DT_{Pn}, service)$ 메시지를 S로 송신하는 과정을 통해, S로 서비스를 요청하게 된다. 이 때, 악의적인 플레이스 P_n'가 P_n이 S로 송신하는 SR 메시지를 가로챈 후, 이것을 재사용하여 서비스 요청을 계속적으로 시도할 수 있다. 그러나 SR 메시지에 포함된 $Sig_{Pn}(N_S)$의 N_S는 S에 의해 일회적으로 발행되는 난수이기 때문에, 재사용은 의미가 없다. 따라서 SR 메시지를 재사용하여 S로부터 서비스를 제공받으려는 P_n'의 시도는 실패하게 된다.

제안 기법의 모든 알고리즘은 연속된 위임에 참여하는 모든 플레이스들이 주고받는 메시지에 일회성을 가진 난수를 이용한 challenge-response 기반의 인증 기법을 사용한다. 즉, 일회성을 가진 난수를 이용하기 때문에, 악의적인 플레이스에 의한 재사용은 아무런 의미가 없다. 따라서 제안한 위임 기법은 재연에 의한 공격으로부터 안전하다.

정리 2. 제안한 위임 기법은 유효하지 않은 위임경로의 생성을 목적으로 하는 위임토큰의 치환에 의한 공격으로부터 안전하다.

증 명 P_1에서 실행 중인 에이전트 MA가 플레이스 P_2, P_3, P_i, P_n을 순서대로 이주하면서 위임을 받은 후, 종단 서버 S로 서비스를 요청한 경우, 위임경로 ① : $P_1P_2P_3P_iP_nS$이 생성된다. 위임경로 ①의 생성에 사용되는 위임토큰들은 다음과 같다.

$$P_1 \rightarrow P_2 : DT_{P_1} = \langle P_1, P_2, Priv_{P_1}(P_2), Sig_{P_1}(N_{P_2}), t_{P_1}, DT_{P_0} \rangle_{P_1}$$
$$P_2 \rightarrow P_3 : DT_{P_2} = \langle P_2, P_3, Priv_{P_2}(P_3), Sig_{P_2}(N_{P_3}), t_{P_2}, DT_{P_1} \rangle_{P_2}$$
$$P_3 \rightarrow P_i : DT_{P_3} = \langle P_3, P_i, Priv_{P_3}(P_i), Sig_{P_3}(N_{P_i}), t_{P_3}, DT_{P_2} \rangle_{P_3}$$
$$P_i \rightarrow P_n : DT_{P_i} = \langle P_i, P_n, Priv_{P_i}(P_n), Sig_{P_i}(N_{P_n}), t_{P_i}, DT_{P_3} \rangle_{P_i}$$

한편, 이전에 플레이스 P_1'에서 MA가 P_2, P_3, $P_j(j \neq i)$, P_n을 순서대로 이주하면서 위임을 받은 후, S로 서비스를 요청했을 경우, 위임경로 ② : $P_1'P_2P_3P_jP_nS$가 생성된다. 위임경로 ②의 생성에 사용되는 위임토큰들은 다음과 같다.

$P_1' \rightarrow P_2$: $DT_{P_1'} = \langle P_1', P_2, Prit_{P_1'}(P_2), Sig_{P_1'}(N_{P_2}), t_{P_1'}, DT_{P_0} \rangle_{P_1'}$

$P_2 \rightarrow P_3$: $DT_{P_2} = \langle P_2, P_3, Prit_{P_2}(P_3), Sig_{P_2}(N_{P_3}), t_{P_2}, DT_{P_1'} \rangle_{P_2}$

$P_3 \rightarrow P_j$: $DT_{P_3} = \langle P_3, P_j, Prit_{P_3}(P_j), Sig_{P_3}(N_{P_j}), t_{P_3}, DT_{P_2} \rangle_{P_3}$

$P_j \rightarrow P_n$: $DT_{P_j} = \langle P_j, P_n, Prit_{P_j}(P_n), Sig_{P_j}(N_{P_n}), t_{P_j}, DT_{P_3} \rangle_{P_j}$

이 때, 공격자가 ①과 ②의 생성에 사용된 위임토큰들을 불법적으로 획득한 후, 유효하지 않은 위임경로 ③ : $P_1P_2P_3P_jP_nS$의 생성을 시도한다고 가정하자. 공격자는 위임경로 ③의 생성을 위해 위임경로 ②의 생성에 사용된 위임토큰들 중에서 $DT_{P_1'}$를 위임경로 ①의 생성에 사용된 DT_{P_1}으로 치환을 시도한다. 그러나 제안 기법은 연속된 위임의 수행을 위해 내포된 토큰 방식을 사용하기 때문에, 치환에 의한 공격을 탐지한다. 즉, 제안 기법에서는 플레이스가 위임토큰을 생성할 때, 이전의 위임토큰을 내포시킨 후, 위임토큰을 생성한 플레이스의 비밀키로 서명을 하게 된다. 그러나 공격자는 P_2의 비밀키를 알 수 없기 때문에, 위임경로 ②의 생성에 사용된 DT_{P_2}안의 $DT_{P_1'}$을 위임경로 ①의 생성에 사용된 DT_{P_1}으로 치환하려는 시도는 실패하게 된다.

제안 기법은 연속된 위임의 수행에 있어서 현재의 플레이스 P_i가 다음 플레이스 P_{i+1}로 전달할 위임토큰을 생성할 때 이전 플레이스 P_{i-1}로부터 받은 위임토큰과 다음 플레이스로 위임할 권한을 묶어서 자신의 비밀키로 서명을 하는 방식으로 동작되기 때문에, 악의적인 플레이스에 의한 위임토큰 치환 공격은 불가능하게 된다. 따라서 제안한 위임 기법은 유효하지 않은 위임경로의 생성을 목적으로 하는 위임토큰의 치환에 의한 공격으로부터 안전하게 된다.

V. 결 론

본 논문에서는 이동 에이전트 환경에서 플레이스 간의 연속된 위임을 안전하게 수행할 수 있는 방법을 제시하였다. 이동 에이전트 환경에서는 에이전트가 이동성을 가지기 때문에, 에이전트는 여러 플레이스들을 이주하면서 실행된다. 기존의 이동 에이전트 환경에서의 위임을 위한 연구는 이주와 관련된 두 플레이스만을 대상으로 하기 때문에, 연속된 위임에는 부적절하다. 제안 기법은 연속된 위임 시 각각의 플레이스가 다른 플레이스로 권한을 위임할 때 사용되는 위임토큰들을 내포된 토큰을 기반으로 생성하기 때문에, 플레이스간의 연속된 위임을 안전하게 수행한다. 또한, 제안 기법이 재연에 의한 공격 및 위임토큰의 치환에 의한 공격으로부터 안전함을 보였다. 향후 연구 과제는 에이전트가 보다 가벼운 상태로 이주할 수 있도록 위임토큰의 구성 요소를 개선하는 것과, 위임토큰간의 관계 형성 기법을 개선하는 것이다.

참 고 문 헌

[1] C. G. Harrison, D. M. Chess, A. Kershenbaum, "Mobile Agents: Are they a good idea?," Research Report, *IBM Research Division T. J. Watson Research Center*, 1995.

[2] S. Berkovits, J. D. Guttman, V. Swarup, "Authentication for Mobile Agents," *LNCS 1419*, pp. 114-136, 1998.

[3] W. M. Farmer, J. D. Guttman, V. Swarup, "Security for Mobile Agents: Issues and Requirements," *Proc. the 19th National Information Systems Security Conference*, pp. 591-597, 1996.

[4] W. A. Jansen, "Countermeasures for Mobile Agent Security," *Computer Communications*, 2000.

[5] A. Corradi, R. Montanari, C. Stefanelli, "Mobile Agents Protection in the Internet Environment," *Proc. the 23th Annual International Computer Software & Applications Conference*, pp. 80-85, 1999.

[6] U. G. Wilhelm, S. M. Staamann, L. Buttyan, "A Pessimistic Approach to Trust in Mobile Agent Platforms," *IEEE Internet Computing*, 4(5), pp. 40-48, 2000.

[7] Y. Ding, H. Petersen, "A New Approach for Delegation using Hierarchical Delegation Token s," *Proc. the 2nd Conf. on Computer & Communications Security*, pp. 128-143, 1996.

[8] G. Vogt, "Delegation of Tasks and Rights," *Proc. the 12th Annual IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, pp. 327-337, 2001.

[9] M. Abadi, M. Burrows, B. Lampson, G. Plotkin, "A Calculus for Access Control in Distributed Systems," *ACM Transactions on Programming Language and Systems*, 15(4), pp. 706-734, 1993.

[10] B. Lampson, M. Abadi, M. Burrows, E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *Proc. the 13th ACM Symp. on Operating Systems Principles*, pp. 165-182, 1991.

[11] V. Vardharajan, P. Allen, S. Black, "An Analysis of the Proxy Problem in Distributed Systems," *Proc. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 255-175, 1991.

[12] B. Crispo, "Delegation Protocols for Electronic Commerce," *Proc. the 6th IEEE Sym. on Computers & Communications*, pp. 674-679, 2001.

[13] B. C. Neuman, "Proxy-Based Authorization and Accounting for Distributed Systems," Proc. the 13th International Conference on Distributed Computing Systems, pp. 283-291, 1993.

[14] N. Li, B. Grosof, J. Feigenbaum. "A Practically Implementable and Tractable Delegation Logi c," *Proc. 2000 IEEE Sym. on Security & Privacy*, pp. 27-42, 2000.

권 혁 만(Hyeog-Man Kwon) 정회원

2001년 2월 : 성균관대학교 전기전자및컴퓨터공학부학사
2003년 2월 : 성균관대학교 정보통신공학부 석사

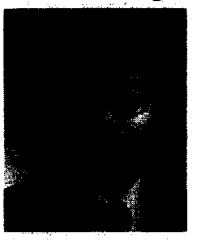
<주관심분야> 이동 에이전트, 분산 시스템, 3G QoS

김 문 정(Moon-Jeong Kim) 정회원

1998년 2월 : 성균관대학교 전기전자및컴퓨터공학부졸업
2000년 2월 : 성균관대학교 전기전자및컴퓨터공학부석사
2002년 2월 : 성균관대학교 정보통신공학부박사수료

<주관심분야> 분산시스템, 이동 에이전트, P2P 네트워크, 유비쿼터크 컴퓨팅

엄 영 익(Young Ik Eom) 정회원



1983년 2월 : 서울대학교 계산통계학과 학사
1985년 2월 : 서울대학교 전산과학과 석사
1991년 8월 : 서울대학교 전산과학과 박사
2000년 9월 ~ 2001년 8월 Dept. of Info. and Comm. Science at UCI 방문
현재 성균관대학교 정보통신공학부 교수

<주관심분야> 분산시스템, 이동 컴퓨팅 시스템, 이동 에이전트, 시스템 소프트웨어 등