제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

이동 에이전트 환경을 위한 연속 위임 구현 기법

이현석*, 권혁만**, 엄영익*
*성균관대학교 정보통신공학부
**㈜에치에프알
e-mail : ciga2000@ece.skku.ac.kr, hmkwon@hfrnet.com, yieom@ece.skku.ac.kr

A Cascaded Delegation Scheme for Mobile Agent Environments

Hyun-suk Lee*, Hyeog Man Kwon**, Young Ik Eom*
*School of Information and Communication Engineering, Sungkyunkwan University
**HFR, Inc.

요 약

이동 에이전트 환경에서는 에이전트의 이동성으로 인하여 에이전트의 이주(migration)가 연속적으로 발생할 수 있다. 이에 따라 에이전트를 실행할 권한을 위임(delegation)하기 위해 플레이스(place)간에 연속위임(cascaded delegation)이 발생할 수 있다. 기존의 연구는 에이전트 이주에 관련한 두 플레이스만을 위임의 대상으로 고려하기 때문에 연속 위임을 위한 기법으로는 부적절하다. 본 연구에서는 이동 에이전트 환경에서 연속 위임을 안전하게 수행하는 연속 위임 구현 기법을 제안한다. 제안 기법은 플레이스간의 신뢰관계에 따라 각 위임토큰(delegation token)을 다음에 생성되는 위임토큰에 내포시킨 후 서명하는 방법과 에이전트의 송신자에 의해 서명된 초기 토큰(initial token)만을 내포시킨 후 서명하는 방법을 나눠서 사용한다. 또한 본 제안 기법이 메시지 재연에 의한 공격과 위임토큰 치환 공격에 안전함을 증명한다.

1. 서론

이동 에이전트란 지능을 가지고 여러 호스트들을 자율적으로 이동하면서 호스트에서 제공되는 자원들을 이용하여 노드에 독립적으로 작업을 수행하거나 다른 에이전트들과 상호작용을 하면서 사용자의 작업을 수행할 수 있는 프로세스로 정의된다. 기존의 클라이언트/서버 구조에 비교해 볼 때 자율성과 이동성을 가진 이동 에이전트는 네트워크의 트래픽 감소, 클라이언트와 서버간의 비동기 연산 지원, 서비스의 분산 및 병렬 처리, 그리고 동적인 서버 인터페이스의 변경 지원과 같은 여러 장점들을 가지기 때문에 최근에는 이동 에이전트 이동성을 보다 안전하게 보장하기 위하여 에이전트 및 호스트의 인증 문제 또한 에이전트 및 호스트 보안 문제에 관한 연구들이 진행되고 있다 [1,2,3,4,5,6].

본 논문에서는 에이전트 이주가 연속적으로 발생하는 경우 요구되는 연속 위임을 안전하게 수행할 수 있는 연속 위임 구현 기법을 제시한다. 위임(delegation)은 한 주체(principal)가 다른 주체에게 자신을 대신하여 에이전트를 실행시킬 수 있도록 권한을 부여하는 과정으로 정의되며, 연속 위임은 셋 이상의 주체들간에 위임이 연속적으로 수행되는 과정으로 정의된다[7,8].

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로서 이동 에이전트 환경에서의 두 플레이스간의 위임과 분산 환경에서의 연속 위임에 대해 설명한다. 3 장에서는 본 논문에서 제안하는 이동 에이전트 환경을 위한 내포된 토큰 기반의 연속 위임 구현 기법에 대해 설명하고, 알고리즘을 통해 제안 기법의 구체적인 동작 과정에 대해 알아본다. 4 장에서는 본 제안 기법의 안정성을 증명한다. 마지막으로, 5 장에서는 결론 및 문제점들을 지적하고 향후 연구 과제에 대해서 기술한다.

2. 관련 연구

이동 에이전트 환경에서의 위임을 위한 대표적인 연구로 Berkovits 의 연구를 들 수 있다[2]. 그러나 이 연구는 이주와 관련된 두 플레이스간의 위임만을 고려하므로 셋 이상의 플레이스간에 요구되는 연속 위임에는 부적절하다. 본 장에서는 이동 에이전트 환경에서의 두 플레이스간의 위임 기법과 분산 환경에서의 연속 위임에 대해 설명한다.

2.1. 이동 에이전트 환경에서의 두 플레이스간의 위임

Berkovits 의 연구에서는 이주와 관련된 주체들간의 신뢰관계에 따라 에이전트의 실행 주체(executing principal)가 변경된다. 플레이스 I_1 에서 주체 P_1 으로 에이전트 A 를 실행시키고 플레이스 I_2 로 에이전트를 이주시켜 새로운 주체 P_2 로 에이전트를 실행시키는 상황을 가정할 때, I_2 가 I_1 에 의해서 신뢰를 받는지 아니면 A 에 의해 신뢰를 받는지에 따라서 4 가지 다른 신뢰관계를 표현하는 P_2의 값이 달라진

다. 즉, I_2가 A를 실행시키기 위한 권한이 변경되는 것이다. 4 가지 신뢰관계는 다음과 같다.

(1) 플레이스 핸드오프(place handoff): 4 가지의 신뢰관계 중 신뢰도가 가장 높은 관계이다. 이 경우 I_1은 에이전트를 I_2로 핸드오프 할 수 있다. 그 후 I_2는 P_1을 대신하여 에이전트를 실행하게 된다.

(2) 플레이스 위임(place delegation): 플레이스 핸드오프보다 낮은 신뢰도를 가지는 관계이며 이 경우 I_1은 I_2로 에이전트를 위임할 수 있다. 그 후 I_2는 P_1의 권한에 자신의 권한을 추가하여 에이전트를 실행시킨다.

(3) 에이전트 핸드오프(agent handoff): 에이전트는 직접 I_2로 자신을 핸드오프 할 수 있다. 그 후 I_2는 P_1에 상관없이 에이전트 자체의 권한으로 에이전트를 실행하게 된다.

(4) 에이전트 위임(agent delegation): 에이전트는 I_2로 자신을 위임할 수 있다. 그 후 I_2는 에이전트 자체의 권한에 자신의 권한을 추가하여 P_1에 상관없이 에이전트를 실행하게 된다.

신뢰관계에 따른 에이전트 실행 주체의 변경은 표 1 과 같다. 이는 Lampson 의 형식 이론(formal theory)[9,10]을 기반으로 하여 안정성이 증명된 인증 기법이다. 여기에서 A for S 는 송신자에 의해 서명된 에이전트의 자체 권한을 의미한다.

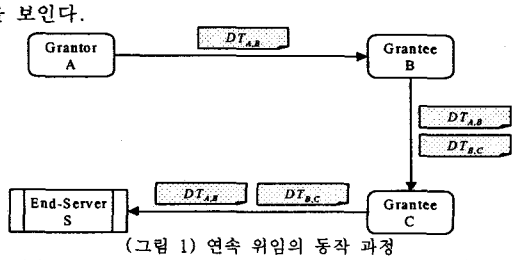
<표 1> 신뢰관계에 따른 실행 주체의 변경

에이전트 이주요청	신뢰관계	플레이스 I_1의 주체	플레이스 I_2에서의 실행 주체
플레이스	플레이스 핸드오프	P_1	$P_2 == P_1$
플레이스	플레이스 위임	P_1	$P_2 == I_2$ for P_1
에이전트	에이전트 핸드오프	P_1	$P_2 ==$ A for S
에이전트	에이전트 위임	P_1	$P_2 == I_2$ for A for S

그러나 Berkovits 의 연구는 신뢰관계의 정의에 있어서 이주와 관련된 두 플레이스만을 고려하기 때문에, 셋 이상의 플레이스들간의 연속 위임에는 부적절하다.

2.2. 분산 환경에서의 연속 위임

연속 위임은 셋 이상의 주체들간에 위임이 연속적으로 수행되는 과정으로 정의된다. 그림 1 은 수여자(grantor) A에 의해 위임이 시작되어 피수여자(grantee) B, C 로 연속 위임이 일어난 후, 종단 서버 S 로 서비스를 요청하는 과정을 보인다.



(그림 1) 연속 위임의 동작 과정

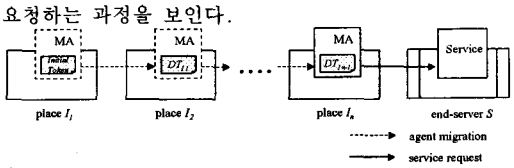
연속 위임을 위하여 위임토큰을 사용하며 위임토큰 $DT_{X,Y}$는 플레이스 X 가 플레이스 Y 로 위임할 권한과 위임기한을 나타내는 타임스탬프 등을 포함하여 연속 위임을 수행한다. 그러나 그림 1 에서는 위임토큰들간에 안전한 관계를 맺지 않았으므로 위임토큰 치환 공격으로부터 안전하지 않을 수 있다.

3. 연속 위임 기법

본 연구에서는 이동 에이전트 환경에서 연속 위임을 안전하게 수행하는 연속 위임 구현 기법을 제안한다. 우선 본 제안 기법의 시스템 구조와 자료구조를 설명하고 알고리즘을 통해 본 제안 기법의 동작을 설명한다.

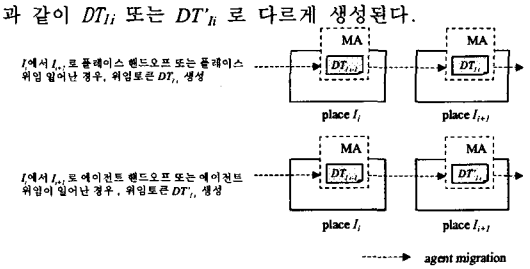
3.1. 시스템 구조

그림 2 는 에이전트 MA 가 플레이스 I_1부터 I_n까지 이주하고 위임토큰 DT_{In-1}을 기반으로 종단 서버 S에게 서비스를 요청하는 과정을 보인다.



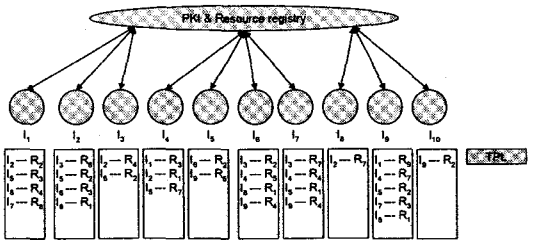
(그림 2) 이동 에이전트 환경에서의 연속 위임

그림 2 에서 I_1은 에이전트 MA 의 송신자에 의해 서명된 초기토큰을 가지고 MA 를 실행한다고 가정한다. I_1에서 실행중인 MA 는 I_2로 위임할 권한을 담은 DT_{I1}을 생성하여 MA 에 포함시킨 후 MA 를 I_2로 전송한다. 이 때 두 플레이스간의 인증은 공개키 기반(PKI: Public Key Infrastructure)의 인증방식을 사용한다. MA 를 받은 I_2는 I_3로 위임할 권한을 담은 DT_{I2}를 생성하야 MA 에 포함시킨 후 MA 를 I_3로 전송한다. 이와 같은 연속 위임 과정을 거쳐 I_n에 도착한 MA 는 DT_{In-1}을 기반으로 종단 서버 S에게 서비스를 요청한다. 임의의 위임토큰 $DT_{Ii}(1 \le i \le n-1)$는 신뢰관계에 따라 그림 3 과 같이 DT_{Ii} 또는 DT'_{Ii} 로 다르게 생성된다.



(그림 3) 신뢰관계에 따른 두가지의 토큰방식

이러한 위임과정을 수행하기 위해서는 이주에 관련된 플레이스들간의 신뢰관계를 알아내야 한다. 이를 위해서는 그림 4 와 같은 구조가 필요하다.

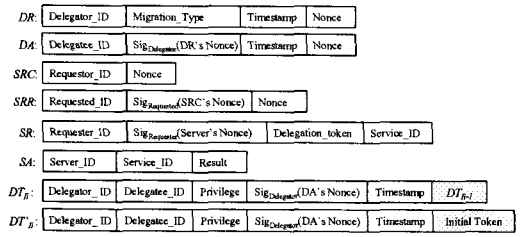


(그림 4) 신뢰관계를 알아내기 위한 추가 구조

본 제안 기법에서는 플레이스간의 인증을 위해서 공개키 기반의 인증방식을 사용한다. 따라서 플레이스들이 주기적으로 PKI 에 접속해야 한다. 또한 플레이스는 자체적으로 신뢰할 수 있는 플레이스들의 목록(TPL: Trustable Place List)을 가지고 있어야 한다. TPL 에는 그 플레이스들이 어떠한 자원을 제공하고 있는지에 대한 정보도 같이 가지고 있다. TPL 은 다른 플레이스들이 어떤 자원을 제공하는가에 대한 정보를 가지고 있는 Resource registry 에 주기적으로 접근하여 새로운 정보로 갱신하여야 한다.

3.2. 자료 구조

본 절에서는 제안 기법에서 사용하게 될 플레이스들간의 통신 메시지 구조, 종단 서버에 서비스를 제공받기 위한 통신 메시지 그리고 위임토큰의 구조에 대해서 설명한다. 그림 5 는 제안 기법에서 사용하는 통신 메시지들과 위임토큰의 구조를 보인다.



(그림 5) 통신 메시지와 위임토큰 구조

DR 메시지와 *DA* 메시지는 플레이스간에 인증을 하기 위해 사용된다. *SRC* 메시지와 *SRR* 메시지는 위임의 마지막 플레이스와 종단 서버간의 인증을 하기위해 사용된다. *SR* 메시지와 *SA* 메시지는 각각 종단 서버에 서비스를 요청하고 서비스의 결과를 받기위해 사용된다. 위임토큰 *DT* 는 위임받는 플레이스를 인증하고 에이전트를 실행할 권한을 넘겨주기 위하여 생성된다. 본 제안 기법에는 두 가지의 위임토큰이 존재한다. 에이전트 이주 시 플레이스간의 신뢰관계가 플레이스 핸드오프 또는 플레이스 위임인 경우 I_i 는 DT_{Ii-1} 을 축적시켜 DT_{Ii} 를 생성한다. 그러나 에이전트 핸드오프 또는 에이전트 위임인 경우에는 이전의 내포된 위임토큰 DT_{Ii-1} 을 버리고 송신자에 의해서 서명된 초기 토큰만 내포시켜 DT'_{Ii} 를 생성한다. 이는 I_i 이전의 플레이스들에게 위임받은 권한이 더 이상 필요지 않으며 또한 그 플레이스들에 의해서 내포된 위임토큰이 없이도 위임토큰 치환공격에 안전하기 때문이다.

3.3 알고리즘

본 제안에서 전체 위임경로는 $I_1I_2I_3\cdots I_{n-1}I_nS$ 라고 가정한다. I_1에서 P_1의 권한을 가지고 실행되고 있는 MA 는 S 에게 서비스를 제공받기 위하여 $I_2I_3\cdots I_{n-1}I_n$ 을 순서대로 이주한다. 이때 각 플레이스 $I_i(2 \le i \le n)$에서는 MA 를 실행하기 위한 변경된 권한을 위임받는다. MA 가 위임경로 상 마지막 플레이스인 I_n 에 도착한 후, 내포된 토큰을 기반으로 S에게 서비스를 요청한다. 본 제안 기법은 다음의 4 가지로 나뉜다: (1) 위임할 플레이스 I_1 은 I_2 에게 연속 위임을 시작한다. (2) 중간에 있는 플레이스 I_i $(2 \le i \le n-1)$는 연속 위임을 중재한다. (3) 연속 위임의 마지막 플레이스 I_n 은 종단 서버 S 에게 서비스를 요청한다. (4) 종단 서버 S 는 I_n이 요청한 서비스를 제공한다.

플레이스 I_1이 MA 를 I_2로 이주시키려 하거나 I_1에서 P_1의 권한으로 실행되고 있는 MA 가 I_2로 이주를 요청한 경우, I_1 은 알고리즘 1 을 사용하여 연속 위임을 초기화한다. 동시에 I_2 는 I_1 을 인증하기 위하고 자신을 I_1 에게 인증받기 위하여 알고리즘 2 의 Segment A 을 사용한다. 아래의 알고리즘에서, *DR* 메시지의 MT 는 Migration Type 을 의미하며 PH(플레이스 핸드오프), PD(플레이스 위임), AH(에이전트 핸드오프) 그리고 AD(에이전트 위임) 중 하나의 값이 들어간다. N_X 는 플레이스 X 에 의하여 생성된 난수를 의미하며, $Sig_X(N_X)$는 플레이스 X 에 의해 생성된 난수에 플레이스 Y 의 공개키 기반 디지털 서명을 하는 것을 의미한다. 또한, $Priv_X(Y)$는 Y 가 에이전트를 실행할 때 가지는 권한을 플레이스 X 가 위임하는 것을 의미하며 Y 는 *DR* 메시지의 Migration Type 에 따라 위임받은 권한을 그대로 사용하거나 자신의 권한을 추가하여 MA 를 실행하게 된다. 그리고 t_X는 플레이스 X에 의해 생성된 위임토큰의 유효한 기간을 의미한다.

```
I1 은 MA 의 송신자에 의해 서명된 초기 토큰을 가지고 MA 를 실행;
when 플레이스 I1이 에이전트를 I2로 이주 {
   if I2가 I1 의 TPL 에 있으면 DR(I1, PH, N11)을 I2로 보냄;
   else DR(I1, PD, N12)을 I2로 보냄;
}
when MA 가 I1에서 실행하다가 I2에게 이주를 요청 {
   if I2가 I1 의 TPL 에 있으면 DR(I1, AH, N11)을 I2로 보냄;
   else DR(I1, AD, N11)을 I2로 보냄;
}
DR 메시지의 Timestamp 동안 I2로부터 DA(I2, Sig12(N11), N12) 메시지가
   올 때까지 기다림;
if (DA 메시지 수신 시) {
   I2를 인증하는 과정을 거쳐서 인증을 실패하면 MA 를 버리고 FALSE 를 반환;
   if (DR 메시지의 MT == PH or PD)
      초기 토큰을 내포시키고 I1의 privilege 를 넣어서 위임토큰 DTI1
      을 생성하고, MA 에 생성된 위임토큰을 포함시켜 MA 를 I2로 보냄;
   else 송신자에 의해 서명된 에이전트의 자체권한과 초기 토큰을 넣어서
      위임토큰 DT'11 을 생성하고, MA 에 생성된 위임토큰을 포함시켜 MA 를
      I2로 보냄;
   return TRUE;
}
return FALSE;
```

(알고리즘 1) 연속위임의 초기 알고리즘

우선 MA 의 이주가 일어나는 상황은 플레이스가 에이전트를 이주시키는 경우와 에이전트가 플레이스에게 이주를 요청하는 두 가지로 나뉜다. 알고리즘 1 에서 I_1이 MA 를 이주시키려 하는 경우, I_2가 I_1의 TPL 에 있으면 플레이스 핸드오프로 에이전트를 이주시키고①, 없으면 플레이스 위임 으로 에이전트를 이주시킨다②. 그러나 I_1 에서 실행중인 MA 가 I_1에게 이주를 요청하는 경우, I_2가 I_1의 TPL 에 있으면 에이전트 핸드오프로 에이전트를 이주시키고③, 없으면 에이전트 위임으로 에이전트를 이주시킨다④. 만약 알고리즘 1 이 TRUE 를 반환하면(이것은 연속 위임이 성공적으로 초기화됐음을 의미한다), DT_{I1} 을 포함한 MA 는 I_2 에 도착한다. 그러면 I_2 는 DR 메시지안의 Migration Type 에 따라 MA 를 재실행한다. 즉, Lampson 의 형식 이론을 기반으로 ①과 ③의 경우에는 DT_{I1} 안의 privilege(각각 P1, A for S)로, MA 를 재실행하고, ②와 ④의 경우에는 DT_{I1} 안의 privilege 에 자신 I_2 의 권한을 추가하여(각각 I2 for P1, I2 for A for S), MA 를 재실행한다. 이는 MA 를 실행시키는 주체를 변경하기 위함이다. 계속해서 I_2에서 I_3으로 에이전트의 이주가 일어나면 I_3 은 알고리즘 2 의 Segment B 를 사용한다. 동시에 I_3 는 I_2 를 인증하고 I_2 에게 자신을 인증시키기 위하여 알고리즘 2 의 Segment A 를 사용한다. 만약 알고리즘 2 에서 TRUE 를 반환하면 DT_{I2}를 포함한 MA 는 I_3에 도착한다.이 과정은 I_3, I_4 ⋯ I_{n-1} 까지 반복된다. 만약 I_{n-1} 까지의 연속 위임이 성공적으로 끝나면 I_n에 DT_{In-1} 을 포함한 MA 가 도착한다.

```
/* Ii이 Ii-1로부터 DR(Ii-1, MT, Nii-1) 메시지를 받은 상태임 */
/* Segment A: Ii-1과 연결을 시도함*/
DA(Ii, Sigii (Nii-1), Nii) 메세지를 Ii-1에게 보냄;
when (위임토큰 DTii-1을 포함한 Ii-1로부터 MA가 도착) {
   DTii-1안의 Sigii-1(Nii)를 이용하여 Ii을 인증;
   if (인증에 실패) MA 를 버리고 FALSE 를 반환;
}
/*End of segment A */
if (DR 메시지안의 MT 가 PH 또는 AH)
   DTii-1안의 privilege 로 MA 을 재실행;
else (MT == PD or AD)
   DTii-1안의 privilege 와 Ii의 권한을 합쳐서 새로운 privilege 를
      생성하고, 새로이 생성된 privilege 로 MA 를 재실행;
/* Segment B: Ii+1과 연결을 시도함*/
when (플레이스가 에이전트를 Ii+1로 이주) {
   if (Ii+1이 Ii의 TPL 에 있음) DR(Ii, PH, Nii)을 Ii+1로 보냄;
   else DR(Ii, PD, Nii)을 Ii+1로 보냄;
}
when (MA 가 Ii에서 실행하다가 Ii에게 이주를 요청) {
   if (Ii+1이 Ii의 TPL 에 있음) DR(Ii, AH, Nii)을 Ii+1로 보냄;
   else DR(Ii, AD, Nii)을 Ii+1로 보냄;
}
DR 메시지의 Timestamp 동안 Ii+1 로부터 DA(Ii+1, Sig ii+1 (Nii), Nii+1) 메시지가 올
   때까지 기다림;
if (DA 메시지 수신 시) {
```

```
    I_{i+1}을 인증하는 과정을 거쳐서 인증을 실패하면 MA를 버리고 FALSE를 반환;
    if (DR 메시지의 MT == PH or PD)
      DT_{Ii-1}를 내포시키고 I_i의 privilege를 넣어서 위임토큰 DT_{Ii}를
        생성하고, MA 에 생성된 위임토큰을 포함시켜 MA를 I_{i+1}로 보냄;
    else 송신자에 의해 서명된 에이전트의 자체권한과 초기 토큰만을
        내포해서 위임토큰 DT1_i 을 생성하고, MA 에 생성된 위임토큰을 포함시켜
        MA를 I_{i+1}로 보냄;
    return TRUE;
}
return FALSE;
```

(알고리즘 2) 연속위임의 증재 알고리즘

만약 I_n에서 실행중인 MA 가 종단 서버 S에 서비스를 요청하면 I_n은 알고리즘 3 을 사용하여 SR메시지를 S에게 보낸다. 그러면 종단 서버 S는 알고리즘 4 를 사용하여 SR메시지 안에 있는 I_n 의 디지털 서명 $Sig_{In}(N_s)$와 위임토큰 DT_{In-1} 을 확인하는 과정을 거치고 이 과정이 성공적으로 끝나면 S는 I_n으로부터 요청된 서비스를 수행하여 그 결과를 SA메시지에 담아서 I_n에 보내게 된다.

```
/*종단 서버 S와 연결을 시도*/
when (MA가 I_n에서 실행하다가 종단 서버 S에게 서비스를 요청) {
    S에게 SRC(I_n, N'_{In}) 메시지를 보냄;
    S로부터 SRR(S, Sig_s(N'_{In}), N_s) 메시지가 올 때까지 기다림;
    SRR 메시지에 있는 Sig_s(N'_{In})을 이용하여 S를 인증;
    if (인증이 실패) MA를 버리고 FALSE를 반환;
    S에게 SR(I_n, Sig_{In}(N_s), DT_{In-1},service) 메시지를 보냄;
    S로부터 SA(S, service, result) 메시지가 도착할 때까지 기다림;
}
return TRUE;
```

(알고리즘 3) 서비스 요청 알고리즘

```
/* 종단 서버 S가 I_n으로부터 SRC(I_n, N_{In}) 메시지를 받은 상태임 */
I_n에게 SRR(S, Sig_s(N_{In}), N_s) 메시지를 보냄;
when (SR 메시지가 도착) {
  SR 메시지 안에 있는 Sig_{In}(N_s)을 이용하여 I_n을 인증;
  if (인증이 실패) MA를 버리고 FALSE를 반환;
  else I_n으로부터 요청된 서비스를 수행;
  서비스 수행 결과를 SA(S, service, result)에 포함시켜 I_n에 SA 메시지를 보냄;
}
return TRUE;
```

(알고리즘 4) 서비스 응답 알고리즘

4. 안정성 증명

본 제안 기법은 공개키 기반 디지털 서명 방식을 취하고 있으므로 인증된 플레이스만이 연속 위임에 참여할 수 있다는 사실을 전제로 한다. 본 장에서는 제안 기법이 메시지 재연 공격(replay attack)과 위임토큰 치환 공격(substituting attack)으로부터 안전함을 증명한다.

정리 1: 본 제안 기법은 메시지 재연 공격에 안전하다.
증명: 전체 위임경로 $I_1I_2I_3\cdots I_{n-1}I_nS$ 로 가정한 경우, 악의있는 플레이스 I'_i가 재연 공격을 시도한다고 가정한다. 만약 I'_i가 DT_{Ii}메시지 또는 SR메시지의 복사본을 만들 수 있다고 할 때, I'_i는 메시지를 재사용하려고 할 수 있다. 그러나 DT_{Ii} 메시지 안의 $Sig_{Ii}(N_{I+1})$ 또는 SR 메시지의 안의 $Sig_{In}(N_s)$에서 각각의 난수 N_{I+1} 와 N_s 는 무작위로 생성되어 한번 사용된 후 버려지므로 위와 같은 메시지 재연 공격은 성공할 수 없다. 그러므로 본 제안 기법은 메시지 재연 공격에 안전하다.

정리 2: 본 제안 기법은 위임토큰 치환 공격에 안전하다.
증명: 악의있는 플레이스 I'_i가 위임토큰 치환 공격을 시도한다고 가정한다. 우선 두가지 다른 위임경로를 가지는 연속위임의 경우, (A) MA 가 플레이스 $I_1, I_2, I_i, I_{i+1}, I_j$ 그리고 I_n을 순서대로 이주하다가 종단 서버 S에게 서비스를 요청하는 경우와 (B) MA 가 플레이스 $I_1, I_2, I'_i, I_{i+1}, I_k$ 그리고 I_n을 순서대로 이주하다가 종단 서버 S에게 서비스를 요청하는 경우를 고려한다. (A)의 경우 위임경로 $Dpath1$: $I_1I_2I_iI_{i+1}I_jI_nS$ 가 생성되고, (B)의 경우 위임경로 $Dpath2$: $I_1I_2I'_iI_{i+1}I_kI_nS$ 가 생성된다. $Dpath1$ 과 $Dpath2$ 를 생성하는 과정에서 악의있는 플레이스 I'_i 는 위임토큰들 각각의 복사본을 만들 수있다. 이 때 $Dpath3$: $I_1I_2I_iI_{i+1}I_kI_nS$ 를 생성하기 위하여 $Dpath2$ 의 위임토큰 DT_{Ii}를 $Dpath1$ 의 DT_{Ii}로 치환하려고 할 수 있다. 그러나 I'_i 는 I_{i+1}의 비밀키를 알 수 없기 때문에 위임토큰을 치환하려는 공격은 실패하게 된다. 이는 본 제안기법이 내포된 토큰방식을 사용하는 특성에 기인한다. 그러므로 본 제안기법은 위임토큰 치환공격에 안전하다.

5. 결 론

이동 에이전트 환경에서의 위임을 위한 기존의 연구는 이주와 관련된 두 플레이스만을 위임의 대상으로 고려하기 때문에, 연속 위임에는 부적절하다. 본 논문에서는 이동 에이전트 환경에서 빈번하게 발생하는 플레이스들간의 연속 위임을 안전하게 수행할 수 있는 방법을 제시하였다. 제안기법으로 연속 위임을 수행하는 경우, 각각의 플레이스가 다른 플레이스로 권한을 위임할 때 플레이스간의 신뢰관계에 따라 다른 위임토큰의 생성과 권한변경 방식을 사용하여 Berkovits 의 연구에서 제시되었던 위임방식을 적용시켰으며 에이전트의 이주가 발생함에 따라 생성되는 위임토큰들을 내포시키는 방식을 사용한다. 따라서 본 제안기법은 플레이스들간의 연속 위임을 안전하게 수행한다. 향후 연구 과제는 본 논문에서 제안한 TPL 을 갱신하는 것에 있어서 신뢰할 수 있는 플레이스의 목록을 추가 또는 삭제하는 기준을 마련하는 것이다.

참고문헌

[1] C. Harrison, D. Chess, and A. Kershenbaum, "Mobile Agents: Are they a good idea?," Research Report 1987, IBM Research Division, 1994.

[2] S. Berkovits, J. Guttman, and V. Swarup, "Authentication for mobile agents," Lecture Notes in Computer Science #1419: Mobile Agents and Security, Springer-Verlag, pp. 114-136, 1998.

[3] W. Farmer, J. Guttman, and V. Swarup, "Security for mobile agents: issues and requirements," Computer Communications: Special Issue on Advances in Research and Application of Network Security, 1996.

[4] W. Jansen, "Countermeasures for mobile agent security," Computer Communications: Special Issue on Advances in Research and Application of Network Security, 2000.

[5] A. Corradi, R. Montanari, and C. Stefanelli, "Mobile agents protection in the internet environment," Proc. 23th Annual International Computer Software and Applications Conference, pp. 80-85, 1999.

[6] U. Wilhelm, S. Stamann, and L. Buttyan, "A pessimistic approach to trust in mobile agent platforms," IEEE Internet Computing, Vol. 4, No. 5, pp. 40-48, 2000.

[7] Y. Ding and H. Petersen, "A new approach for delegation using hierarchical delegation tokens," Proc. 2nd Conference on Computer and Communications Security, pp. 128-143, 1996.

[8] G. Vogt, "Delegation of tasks and rights," Proc. 12th Annual IFIP/IEEE International Workshop on Distributed systems: Operations & Management, pp. 327-337, 2001.

[9] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin, "A calculus for access control in distributed systems," ACM Transactions on Programming Language and Systems, Vol. 15, No. 4, pp. 706-734, 1993.

[10] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in distributed systems: theory and practice," Proc. 13th ACM Symposium on Operating Systems Principles, pp. 165-182, 1991.