이동 에이전트 보호를 위한 일회용 키 생성 시스템

°박종열,이동익
광주과학기술원 정보통신공학과
{jypark, dilee}@kjist.ac.kr

One Time Key Generation System for Mobile Agent Protection

° Jong-Youl Park and Dong-Ik Lee
Dept. of Information and Communications, Kwang-Ju Institute of Science and Technology

요 약

이동 에이전트는 자율적인 개체 이나 이동 에이전트 서버에 의해 실행되기 때문에 많은 보안 문제를 안고 있다 이동 에이전트 시스템에서의 보안 문제는 이동 에이전트 서버를 보호하는 측면과 이동 에이전트 자체를 보호하는 분야로 나뉘어 지며, 지금까지 많은 시스템에서 이동 에이전트 서버의 보안에 대한 연구가 진행되어 왔다. 인증 언어를 비롯하여 다양한 형태의 방법들이 제시 되었고, 실제로도 시스템에서 많이 적용되고 있다 하지만 이동 에이전트 자체의 보호에 대한 연구는 아직 초기의 문제 제기 단계에 있다 이동 에이전트를 보호하기 위해 가장 어려운 점은 이동 에이전트가 각기 다른 서버에 의해서 수행 되고 그 수행이 상호 독립적이어야 한다는 것이다 본 논문에서는 이동 에이전트 시스템에서 일회용 이동 에이전트 키를 생성하여 각 서버가 제공하는 정보를 독립적으로 암호화하여 이동 에이전트가 수집한 데이터들의 신뢰성을 높인다.

1.이동 에이전트 시스템

이동 에이전트모델은 클라이언트/서버 모델에서 발생하는 여러 가지 문제점을 해결하는 새로운 분산처리 시스템 모델로 최근 주목을 받고 있으며 많은 연구가 진행되고 있다

이동 에이전트란 이동성을 가지는 지능형 에이전트를 의미 한다 일반적인 에이전트가 가지는 역할 외에 이동성이라는 특징을 더 가지며 [표 1]은 지금까지 개발된 이동 에이전트 시스템의 예를 보여준다

표 1 이동 에이전트 시스템

이 름	언 어	소 속
AgentTcl	Tcl	Dartmouth College, USA
Ara[1]	Java	University of Kaiserslautern, Germany
Aglets[4]	Java	IBM, Japan
Concordia	Java	Mitsubishi, USA
InforSleuth	Java	MCC, USA
JATLite	Java	Stanford University, USA
MOA	Java	The Open Group, USA
Mole[3]	Java	University of Stuttgart, Germany
Odyssey	Java	General Magic, USA
Voyager	Java	ObjectSpace, Inc , USA
X-MAS[8]	Java	K-JIST, Korea

1.1 보안 문제점

이동 에이전트기 가지는 문제는 자신의 수행 코드를 여러 호스트에 의해서 수행 되어야만 하는 특징 때문에 발생한다. 즉 에이전트는 여러 서버를 옮겨 다니기 때문에 비이러스로 서식하기에 좋은 환경을 제공할 뿐 아니라 에이전 서버는 쉽게 이동 에이전트를 변경하고 도청할 수 있다. 이동 에이전트 기술이 실용화 기술로 자리잡기 위해서 보안 문제에 대한 연구,고찰을 통한 해결이 필수적이다 이동 에이전트 시스템의 보안 문제는 2 가지로 구분할 수 있다

❑ 이동 에이전트로부터 이동 에이전트 서버의 보호
❑ 이동 에이전트 서버로부터 이동 에이전트의 보호

이동 에이전트로부터 이동 에이전트 서버의 보호 - 이동 에이전트는 이동 에이전트 서버상에서 수행된다. 악의를 가진 이동 에이전트의 수행이 이동 에이전트 서버에 악영향을 미칠 수 있다. 예로는 자원 고갈, 시스템 영역 침해, 잘못된 정보 제공 등이 있다. 지금까지 이 부분에서 많은 연구가 이루어져 왔다.

이동 에이전트 서버로부터 이동 에이전트의 보호 - 이동 에이전트는 이동 에이전트 서버의 도움 없이는 이동 하거나 수행이 불가능 하며, 이동 에이전트 서버는 쉽게 이동 에이전트의 코드나 데이터를 복사/수정/삭제할 수 있다 이러한 공격의 예로는 원하지 않는 수행(이동 에이전트), 에이전트 수집 데이터의 변경, 조작 등이 있을 수 있다.

1.2 현재의 보안 모델

표 2 이동 에이전트 시스템의 보안 모델

이동 에이전트 서버 보호	❑ 자바 보안 모델 – Sandbox 보안 모델 [7]
	❑ 컴퓨터 자원의 접근 권한에 대한 정의와 제어 [3]
	❑ 인증 언어 [4]
이동 에이전트 보호	❑ Ara 이동 에이전트 시스템 모델 [1]
	❑ Blackbox 보안 모델 [5]

이동 에이전트 서버 보호는 기존의 통신 시스템에서 사용 되고있는 사용자 인증 방법을 이용해서 많은 시스템에서 구현 되었다. 가장 쉽고 널리 사용되는 것이 자바 보안 모델을 적용한 것이다 자바 1.0 에서 보안 문제로 제한 되었던 기능들을 활성화 하기 위해 개발된 **Sandbox 보안 모델**은 인터넷 상에서 수행 되어질 코드 서명함으로서 인증된 애플릿만을 수행 한다는 것이다[7] 이러한 모델은 서명된 애플릿에[그림 1] 의해서만 서버의 제한된 영역을 접근 할수 있도록 하게 함으로 해서 인증되지 않은(익명의) 사용자가 접근 하려는 것을 차단 하는 효과를 가진다. 반면 **컴퓨터 자원의 소유와 권한에 대한 정의와 제어**는 응용 프로그램 차원에서 사용자의 권한을 검사하고 그에 따라 권한을 부여 하는 방식이다 Ara[1]

시스템, Concordia 시스템이 이 방법을 사용하고 있다. 마지막으로 이동 에이전트 서버의 보호를 위한 가장 진보한 방법으로 인증언어 (Authorization Language)를 사용하는 시스템이다 이 모델은 초기의 접근제어 방식과 달리 인증언어를 사용한다. 인증 언어란 사용자의 권한을 설정하기 위해 다단계의 권한 부여와 소유자 인증 기능으로 더욱 유연하고 동적인 모델이다 최근에 제안된 Aglets[4]과 Mole[3] 시스템이 이 모델을 적용하고 있다

이동 에이전트 보호를 위해 제안된 모델을 보면 안전한 AgentTcl 언어(Safe-Tcl language), UC버클리의 Towards Mobile Cryptography [6], Mole 시스템에서 제안한 **Time Limited Blackbox Security[5], Ara 시스템**이 제안한 Passport model이 있다. 먼저 AgentTcl은 Safe Tcl이라는 언어를 정의하여 언어 차원에서 보안문제를 해결 하려고 한다 또한 인증을 위해서 PGP(Pretty Good Privacy)를 사용 하기도 하지만 스크립트 언어의 한계로 널리 사용되지는 못하고 있다. AgentTcl 과 달리 암호학적인 방법을 이용한 연구가 진행 되고 있는데, 간단히 설명 하면 이동 에이전트가 수행될 코드를 암호화하여 서버에서 수행되는 방법을 말한다. 하지만 이 방법은 암호화된 함수(homomorphic encryption schemes)를 찾는 것이 쉽지 않고 매우 제한적인 기능만을 수행하기 때문에 시스템 구현에 사용되기 힘들다. 또 다른 방법으로는 Time Limited Blackbox[5] 모델인데 제한된 시간 동안 에이전트를 보호 하고자 하는 시도이지만 Blackbox 함수가 단순한 code mess-up 방법으로 복제된 에이전트를 이용한 공격이나 병렬 컴퓨팅 환경과 같은 좋은 시스템에서는 쉽게 풀릴수 있다 마지막으로 Ara 시스템[1]에서 제안된 모델을 보면 다른 시스템과 달리 실제 구현에 중점을 두었다는 점이다 그래서 전자 서명과 Passport를 이용한 인증 작업과 SSL를 이용한 안전한 통신 시스템을 제공하고 있다 다른 시스템이 개방형 시스템을 지향한 반면, Ara 시스템은 공개키 기반의 시스템으로 구현 됨으로 해서 효과적인 모델을 제시하고 있다

1.3 현재 보안 모델의 문제점

지금 까지 많은 이동 에이전트 보호 모델이 제안 되었다 하지만 실제로 구현된 시스템은 Ara 이동 에이전트 시스템 뿐이며, 이 시스템 역시 몇 가지 문제점을 가지고 있다 Ara 시스템은 인증된 사용자의 접근 제어 방법으로 Allowance 라는 것을 사용하며, 코드를 서명한 Passport 라는 영역을 만들어 사용하고 있다 코드의 신뢰성이나 서버에 대한 문제는 해결 하고 있지만, 이동 에이전트가 수집한 데이터에 대해서는 아무런 보장도 할 수가 없다. 실제로 한 서버가 다른 서버에서 생성된 데이터를 마음대로 분석/수집/삭제 할 수 있다 보통 이동 에이전트의 코드 중에는 중요한 정보(에이전트 행동 양식, 의사 결정 조건, .)가 포함되어 있고, 서버는 쉽게 이동 에이전트 코드의 분석이 가능 하다[2].

2. 이동 에이전트 보호

본 논문은 이동 에이전트가 수집한 데이터를 각 서버로부터 보호 하고 변경/삭제 되는 것을 막기 위해 일회용 키 시스템을 제안한다. 이동 에이전트가 수집한 데이터를 변경/삭제할 수 없도록 하기 위해서는 각 서버가 제공한 데이터를 다른 시스템에서 읽을 수 없어야 하고, 또 그 데이터 암호화에 사용된 키를 유추할 수 없어야 한다. 즉 모든 서버가 제공한 데이터를 암호화 하는 과정에서 매번 다른 키를 사용해야 하며, 그 키들 사이에는 어떤 일관된 연관성을 가지도록 해야한다

2.1 일회용 이동 에이전트 키

이동 에이전트가 자신의 고유키를 가지고 이동하면서 수집한 데이터를 암호화 할 수는 없다 암호화 하는 과정이 서버에 의존된 형태이므로 어떤 형태로든 서버에 키가 알려질 수밖에 없다. 하지만 한 에이전트가 서버에서 사용되어질 키만을 알려 준다면 다른 서버로부터 수집한 데이터를 보호 할수 있을 것이다 뿐만 아니라 일회용 에이전트 키를 이용하면 키들 사이에 일정한 관계를 가지게 되므로 한 서버에서 수집한 데이터를 임의로 삭제 할수도 없다.

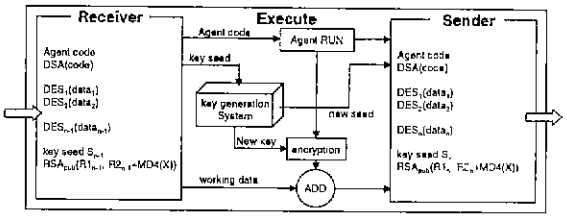


그림 2 일회용 키 시스템

일회용 이동 에이전트 키란 이동 에이전트가 매번 새롭게 만든 키를 사용하고 다음 서버로 이동 하면서 새로운 키를 만들어 사용하는 시스템[그림 2]을 의미 한다. 즉 에이전트 서버가 자신이 제공한 데이터를 보호하기 위해 새로운 키를 생성하고 이를 이용하여 암호화 하는 방법을 의미 한다. 즉 각 서버가 제공한 정보를 각 서버가 암호화 함으로 해서 에이전트가 수집한 데이터가 신뢰성을 갖도록 해준다

2.2 전자 서명과 이동 에이전트

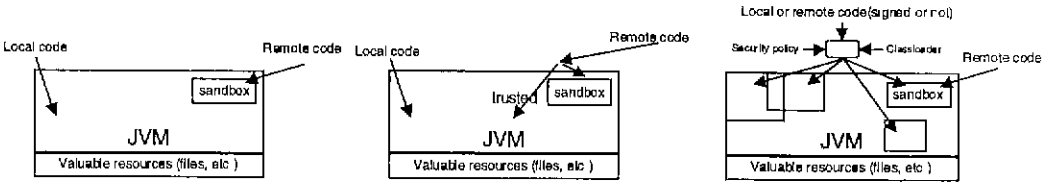
이동 에이전트는 전자 서명기법을 이용하여 코드와 데이터의 신뢰성을 줄수 있다. 많은 이동 에이전트 시스템에서 이동 에이전트의 코드는 서명된 형태로 전송되고 수행 된다. 본 논문에서 제안한 시스템 또한 공개키기반구조(Public Key Infrastructure)를 가정하고 있다 즉 에이전트와 에이전트 서버 사이의 인증 작업이 수행되고 에이전트 코드는 홈 에이전트 서버의 공개키를 이용 서명된 형태로 이동하게 된다. 서명된 코드는 다른 서버에 의해서 불법적으로 수정되지 못한다.

에이전트 서버는 이동 에이전트를 오랜 시간 다른 서버로 전송하지않고 에이전트 코드를 분석하거나 키를 알아 내기 위해 불법적인 시도를 할 수 있다. 또한 복제(Clone) 에이전트를 생성 시켜 다른 시스템을 공격할수 있다(예:항공기 좌석 예매 시스템에서의 예매 에이전트). 그런 이유로 이동 에이전트 시스템은 타임스템프(Timestamp)를 사용한다. 즉 각 서버는 에이전트를 받고 다른 서버로 이동 하는 시간을 기록한다 타임스템프가 가지는 효과는 두가지가 있다. 먼저 타임스템프를 이용하여 서버에서 에이전트가 수행된 시간을 알수 있다 서버에서 불법적인 시도가 있다면 또는 불법적으로 고쳤다 하더라도 많은 시간을 사용 하게 되고, 타임스템프에 의해서 홈 서버에 의해 감지 된다 타임스템프가 가지는 또 다른 효과로는 복제 에이전트를 막을수 있다 복제 에이전트를 만들게 되면 복제 에이전트를 생성한 서버 이전까지 같은 타임스템프를 가지는 에이전트들이 존재 하게 되고, 복제를한 서버에 의해서만 서로 다른 타임스템프를 가지고 있게 되어서 복제 에이전트를 생성한 서버를 알수 있다.

2.3 키 생성을 위한 시스템

키 생성을 위한 시스템은 단방향 해쉬함수와 밀접한 관계가 있다. 단방향 해쉬함수란 함수 $y=f(x)$ 에 대해 $x=f^{-1}(y)$ 인 함수를 찾기 힘들다는 것이다. 이런 단방향성은 여러 번의 해쉬함수를 실행한 경우 현재의 해쉬 값으로 현재와 미래의 가능한 해쉬 값을 알 수는 있지만 과거에 해쉬 함수로 만들어 졌던 값을 알 수는 없다. 이동 에이전트가 사용하게될 일회용 키는 이 단방향 해쉬 함수로 생성 되어지고 초기값을 설정한 홈 서버만이 모든 키를 다시 생성 할수 있는 시스템을 의미한다.

키 생성 시스템을 위해 몇가지 가정을 한다 먼저 본 논문에서 사용되는 기본 통신 체널은 공개키기반구조(public key infrastructure)를 기반으로 한다. 단방향 해쉬 함수를 위해 128 비트 출력을 내는 MD4 를 사용하고, 전자 서명을 위해서는 DSA(Digital Signature algorithm)를 가정 하고 있다 전체적인 일회용 키 생성 과정을 그림 3 에서 보여주고 있다.



a. JDK 1.0 Security Model　b. JDK 1.1 Security Model　c. JDK 1.2 Security Model

그림 1 자바 Sandbox 보안 모델

RANDOM NUMBER : $R1_n$(64bit),$R2_n$,(8bit)$R3_n$(8bit)

RUN : code signature check : DSA(code)
RUN : $S_t = MD4^{R2n}(R1_n \oplus S_{n-1})$
RUN : $data_n$ = Agent code execution result

CREATE:
- **DES key = $S_t/2^{64} \oplus S_t\%2^{64}$**
- **$DES_n(data_n)$**
- **$X = DES_1(data_1) + DES_2(data_2) + ... + DES_n(data_n)$**
- **$RSA_{pub}(R1_n, R2_n, R3_n, MD4(X), TimeStamp)$**

Replace $S_n = MD4^{R3n}(S_t)$
Append $DES_n(data_n)$
Append $RSA_{pub}(R1_n, R2_n, R2_n, MD4(X), TimeStamp)$

그림 3 일회용 키 생성을 위한 과정

새로운 키를 생성하기 위해 가장 먼저 해야 할 일은 각 서버가 자신의 비밀 정보인 3 개의 난수를 생성하는 것이다.

RANDOM NUMBER : $R1_n$(64bit),$R2_n$,(8bit)$R3_n$(8bit)

3 개의 난수는 에이전트 서버가 제공한 정보를 보호 하기 위해서 만들어 낸 수이며, 홈 에이전트 서버에 의해서만 읽을수 있도록 공개키 방식으로 암호화 해서 전송하게 된다.

RUN : code signature check : DSA(code)

이동 에이전트 서버는 이동 에이전트가 전송되면 난수 생성 후에 코드의 변경 유무를 전자 서명을 이용하여 검사하게 된다. 공개키 기반 시스템이므로 DSA 로 쉽게 구현 가능하다.

RUN : $S_t = MD4^{R2n}(R1_n \oplus S_{n-1})$

S_t 란 에이전트 서버가 사용할 키의 생성 정보를 의미한다. 먼저 64 비트 난수인 $R1_n$과 이전 에이전트 서버로부터 받은 128 비트 S_{n-1}값을 XOR 한다 S_{n-1}값은 이전 에이전트 서버로부터 전송 되어진 데이터이므로 그냥 사용될 경우 이전 서버와 공모한 다른 서버에 의해서 공격 될수 있다 그래서 64 비트의 $R1_n$를 $R1_n$:$R1_n$ 인 128 비트로 만들어 S_{n-1}와 XOR 한다. 그리고 그 결과를 $R2_n$번 MD4 로 해쉬함수를 돌린다. 이 값은 $R2_n$는 8 비트의 데이터로 짧은 시간 안에 구할수 있다.

RUN : $data_n$ = Agent code execution result

$data_n$은 이동 에이전트 코드를 수행 하여 발생한 결과 데이터를 의미한다

- **DES key = $S_t/2^{64} \oplus S_t\%2^{64}$**
- **$DES_n(data_n)$**

실제 이동 에이전트가 얻은 결과는 새로 생성된 키로 암호화 되며, 빠른 시간의 수행을 위해 대칭형 암호화 방법을 이용한다 위에서 S_t의 값은 128 비트이며 DES 키는 64 비트 값을 가진다. 새로이 생성된 S_t값의 상위 64 비트와 하위 64 비트를 XOR 하여 DES 키를 만들수 있다 이때 생성된 값이 취약한 키 64 개중 하나이면 S_t값을 다시 생성한다. 이렇게 생성된 키를 이용 수집한 데이터를 암호화 하게 된다.

- **$X = DES_1(data_1) + DES_2(data_2) + ... + DES_n(data_n)$**
- **$RSA_{pub}(R1_n, R2_n, R3_n, MD4(X), TimeStamp)$**

수집한 데이터를 사이의 연관성을 위해 수집된 전체 데이터의 해쉬값과 시간 정보(도착시간+서버 수행시간), 그리고 에이전트 서버가 새로 생성한 3 개의 난수를 홈 에이전트 서버의 공개키로 암호화 하게 된다. 만약 다른 서버에 의해서 위 내용중 일부 또는 전부가 삭제 되어도 일회용 키의 연관성에 의해서 감지 된다.

Replace $S_n = MD4^{R3n}(S_t)$
Append $DES_n(data_n)$
Append $RSA_{pub}(R1_n, R2_n, R2_n, MD4(X))$

사용된 키를 다음 서버가 유추할수 없도록 S_t는 $R3_n$번의 해쉬한 값으로 전송되어진다.이렇게 생성된 S_n, **$DES_n(data_n)$**, **$RSA_{pub}(R1_n, R2_n, R2_n, MD4(X))$** 값은 다음 서버에게 전송 되어 진다

위 시스템에서 추가로 전송되는 데이터의 양은 수십 바이트에 지나지 않는다 또한 시스템에서 사용된 DES, MD4 는 비교적 빠른 시간에 수행 됨으로 적은 컴퓨팅 파워로 구현이 가능하다.

2.4 에이전트 데이터 감사 과정

이동 에이전트가 이동하여 수집한 데이터는 홈 에이전트 서버에 의해서 원래의 모습을 갖게 된다 처음 홈 에이전트 서버는 전체 일회용 키 생성을 위한 초기값을 만들고 각 서버에서 선택된 키 값들이 자신의 공개키로 암호화 되어 있어 모든 과정의 키를 다시 생성하여 수집한 데이터를 볼수 있다

$S_1 = MD4^{R31}(MD4^{R21}(R1_1 \oplus S_0))$
$S_2 = MD4^{R32}(MD4^{R22}(R1_2 \oplus S_1))$
.....
$S_n = MD4^{R3n}(MD4^{R2n}\{R1_n \oplus S_{n-1}\})$

3.결론

이동 에이전트 시스템이 가지는 문제점은 아래와 같이 3 가지 문제로 구분할수 있다

- **이동 에이전트의 코드를 분석,수정**
- **이동 에이전트의 데이터를 분석,수정,삭제**
- **이동 에이전트를 불법적으로 수행**

일회용 키 시스템을 이용하여 이동 에이전트가 수집한 데이터의 분석/수정/삭제를 막을수 있다 일회용 키 시스템이 기지는 가장 큰 문제점은 암호화 기법을 이용함으로 해서 이동 에이전트의 무게가 커지지 않을까 하는 부분이다. 하지만 본 시스템에서 암호화를 위해 필요한 컴퓨팅 파워는 비교적 적으며, 수행 과정에서 생기는 추가 정보도 크지 않다. 이동 에이전트 코드의 분석, 수정에 관한 부분은 이동 에이전트 설계시 중요 결정을 하는 코드를 따로 분리하는 방법을 생각할수 있다[2]. 마지막으로 이동 에이전트의 불법적인 수행을 막기 위해 공개키 기반 시스템을 이용 인증된 코드를 수행하여 불법적인 수정을 방지할 수는 있지만 불법적인 수행을 막지는 못한다.

반면 이동 에이전트 시스템이 일회용 키를 사용함으로 해서 이동 에이전트가 수집한 데이터의 가치는 줌가 하게 되고 완전한 정보의 전달을 위한 이동 에이전트 시스템의 이용에도 많은 도움이 될것이다.

4.향후 연구 과제

본 시스템은 이동 에이전트가 수집한 데이터의 신뢰성을 높이기 위한 방법으로 제시 되었다. 전체 시스템이 이동 에이전트를 보호 하기 위해서는 일회용 키 시스템 외에 이동 에이전트의 중요 코드(이동 에이전트의 판단기준,데이터 처리 방법)를 숨겨야 한다. 향후 이러한 코드를 숨기기 위한 방법이 연구 되어야 하며, 이동 에이전트 시스템인 X-MAS[8]에서 적용 구현 하고자 한다.

5.참고문헌

[1] Holger Peine. Security Concepts and Implementation in the Ara Mobile Agent System 7th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 17-19th, Stanford University, USA

[2] W. Farmer, J Guttman, and V.Swarup Security for mobile agents Authentication and state appraisal. In to appear in the proceedings of the European Symposium on Research in Computer Security(ESORICS), Lecture Notes in Computer Science, September 1996

[3] Baumann, J., Hohl, F., Rothermel, K., and Strasser, M (1998) Mole – Concepts of a mobile Agent System, The Wide Web Journal, special issue on Software Agents.

[4] Karjoth, G., Lange, D B and Oshima, M (1997) A Security Model for Aglets IEEE Internet Computing, Vol. 1, No 4, July - August 1997, pp. 68-77 http://computer org/internet/ic1997/w4068abs htm

[5] Hohl, F.(1998) Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts, Springer Lecture Notes in Computer Science http://www informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/vignabuch ps gz

[6] T. and Tschudin, Chr.. Towards Mobile Cryptography; the 1998 IEEE Symposium on Security and Privacy. http //www.icsi.berkeley edu/~sander/publications/

[7] Sun Microsystems. Java Security ,http //java.sun.com/security/whitepaper ps

[8] J.J.yoo X-MAS: Mobile Agent Platform for Workflow System Considering Time Constraints, submitted to ISADS'99 (International Symposium on Autonomous Decentralized Systems), http://atom.kjist ac.kr/~jjyoo/seminar/isads99-jjyoo.doc