공모된 체인 절단 공격으로부터 QCRT를 이용한 이동 에이전트의 보호 기법

정창렬* · 김광오** · 송진국*** · 이성근****

A Security Scheme of Mobile Agent using QCRT from Colluded Truncation Attacks

Chang-ryul Jung* · Kwang-oh Kim** · Jin-kook Song*** · Sung-Keun Lee****

본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구 결과로 수행되었음
(NIPA-2010-(C1090-1021-0009)

요 약

본 논문에서는 프로로밍 에이전트의 데이터의 안전한 수행을 위해 악의적인 요인들에 의해 발생될 수 있는 절단공격으로부터 안전한 실행을 보장한다. 또한 기존의 체인관계보다 확장성을 개선한 쿼리 트리체인 관계(QCRT:query chain relationship tree)구조를 형성함으로써 악의적인 요소를 효과적으로 검출 할 수 있다. 또한 체인관계 구조를 에이전트의 여정에 따라 형성되는 구조로 체크함으로써 에이전트의 체인관계를 확장하였다. 단면적인 체인관계의 확장으로 기존의 취약한 요소들을 보완하여 이동 에이전트의 절단공격으로부터 실행 보장을 하였다.

ABSTRACT

This paper proposes a mechanism that guarantees secured performance against DDoS attack to protect data from free-roaming agent. Also, as it makes up QCRT(query chain relationship tree) structure which is an enhanced chain relation of existing chain relations that improves extension, It also finds malicious elements. The proposed mechanism extends chain relationship of agent as well by checking chain relation structure as a structure according to the route of agent. For it complements existing weakness with single-side chain relation extension, it guarantees secure performance against DDoS and truncation attacks from mobile agent.

키워드

모바일 에이전트, 절단공격, 쿼리 트리체인 관계, 에이전트 보호 메커니즘

Key word

Mobile agent, Truncation attacks, Query chain relationship tree, Agent protection mechanism

* 정회원 : 순천대학교 컴퓨터공학과
** 정회원 : 호원대학교 보건복지대학
*** 종신회원 : 경남과학기술대학교 컴퓨터공학과
**** 정회원 : 순천대학교 멀티미디어공학과(교신저자, sklee@sunchon.ac.kr)

접수일자 : 2010. 12. 10
심사완료일자 : 2011. 01. 26

Ⅰ. 서 론

에이전트 기술은 분산형 컴퓨터 응용기술에서 사용되는 비중이 높아지고 있다. 이동 에이전트의 접근과 실행은 분산 응용시스템에서 설계와 구현됨으로 악의적인 에이전트로부터 호스트의 보호와 악의적인 컴퓨터 환경으로부터의 공격에 에이전트를 보호하는 것은 매우 어려운 문제이다. 에이전트 보안 위협에 대한 최근의 연구는 데이터 보안에 초점을 두고 있다. 이는 에이전트의 임무수행이 안전하게 종료되는 것이 무엇보다 중요하기 때문이다. 본 논문은 데이터의 보안에 초점을 두었으며, 프리로밍 에이전트에서의 안전성을 고려한 진화된 보안 메커니즘이다. 이는 데이터의 안전을 보장하기 위해 악의적인 공격자에 의해 체인 절단공격을 방어하는 메커니즘이다. 이러한 기술들에 대한 연구는 다양하게 이루어졌다[1-5].

그러나 기존 연구들에서 제시한 에이전트 보안 기법들은 복잡한 구조의 체인 공격들에서는 프로토콜이 안전하지 않다. 이들은 대부분 인접 호스트간의 단면적인 체인 인증을 하고 있어 공모에 의한 체인절단공격이 발생할 경우 안전성을 위협받게 된다. 이를 위해서 더욱 발전된 체인 구조의 프로토콜 연구[6]가 진행되었다. 그러나 프리로밍 에이전트의 데이터 안전을 위해서 다수의 공모가 발생할 경우 공격에 취약한 요소가 있어 에이전트의 훌륭한 임무수행을 위한 프로토콜로서는 취약하다. 따라서 본 논문에서는 이러한 기존의 취약한 요소들을 보완하고, 공모된 체인공격으로부터 안전한 에이전트 실행을 보장하는 에이전트 보호 메커니즘을 제시한다. 본 논문은 구성은 다음과 같다. 2장은 에이전트 보안에 대한 몇몇 관련 연구에 대해 기술하고 문제점을 파악한다. 제 3장에는 절단공격으로부터 에이전트 실행보안을 위해 다차원적인 체인관계 구조를 제시하고, 에이전트 보호 메커니즘을 제시한다. 4장에서 제안한 메커니즘의 보안성 분석을 수행하고, 마지막 5장에서 결론 및 향후 연구를 기술한다.

Ⅱ. 관련 연구

에이전트 임무 수행을 위해 결합된 컴퓨터의 환경이나 네트워크 상에 밀집되어 있는 실행 유닛등은 에이전트들의 안전한 실행을 위해 컴퓨터의 상태와 연관된 제어 정보를 포함하는 안전한 실행 지원이 필요하다. 이를 위해 Yee[1]은 에이전트의 결과들을 보호하기 위해 부분결과인증코드를 제안하였다. 에이전트를 응용할 때 키를 생성하거나, 암호키나 공개키 서명 등을 통해 방문하고자 하는 호스트와 인증절차를 거친다. 이때 에이전트의 무결성을 제공할 수 있다. 그러나 에이전트는 이동하기 전에 많은 키들이 요구됨에 따라 미리 이들 키를 결정해야 하는 문제가 발행함으로 키 관리가 어렵다. 그럼으로 에이전트가 프리로밍을 위한 네트워크 환경에서 에이전트 여정 중에 각 호스트와 완벽한 키 사용이 불가능하다.

Karjoth[2]는 Yee의 문제점을 개선하여 키 인증 그룹을 이용하여 디지털 서명과 해시체인을 통해 에이전트를 보호한다. 서명체인은 현재의 호스트와 다음 방문 호스트와의 체인관계를 통해 확인이 가능하다. 그러나 다른 보안 프로토콜을 제공하는 호스트 간의 암호화 메커니즘의 조합이 이루어져야 하는 문제로 프리로밍이 이루어지는 오픈 네트워크 환경에서는 보안에 대한 위협으로부터 안전하지 않다. Karnnik[3]는 이는 에이전트 데이터의 기밀성을 보호하기 위한 에이전트가 신뢰된 호스트에서 데이터를 이용할 수 있도록 하는 방법과 체인관계를 랜덤 값과 서명을 강화할 수 있는 암호 체크합을 이용하여 공격을 감지한다. 그러나 체인절단 공격이 공모에 의해 이루어질 경우에는 체크 합으로 공격자나 오류 검출이 불가능하기 때문에 선의의 호스트가 악의적인 호스트로 남용될 수 있다. 즉 여러 개의 공모 호스트에 의한 공격에 대해서는 방어할 수 없다. 또한 Corradi[4]는 다중 홉(multi-hop)에서 현재의 결과와 식별자를 체인관계형성으로 체크 합을 통해 입증하였다. 즉, 각 호스트는 이전 호스트로부터 암호화 결과 계산을 통해 증명하는 방법이다. 다중 홉(multi-hop)에서 현재의 결과와 식별자를 체인관계형성으로 체크 합을 통해 입증한다.

그러나 다른 프로토콜처럼 공모된 절단 공격으로부터 에이전트의 실행 결과를 안전하게 보호하는데 한계가 있다. Chang[5]은 두 개의 공모에 의한 절단공격들을 서명들 통해 검출하는 방법이다. 현재의 호스트에서 생성된 결과와 호스트에서 서명하는 절차이다. 각각의 호스트의 노출된 암호 키를 사용함으로 보안을 보장할 수

없을 뿐만 아니라 잠재적인 악의적 호스트의 공격에 대한 안전한 실행을 실현할 수 없다. 줄기공격과 공모된 절단공격의 방어를 위해서는 프로토콜의 개선이나 양방향 인증이 필요하다. Xu[6]의 체인관계를 이용한 이동 에이전트의 데이터 보호는 호스트 간 체인관계를 형성함으로써 악의적인 호스트에 의해 발생할 수 있는 에이전트 데이터의 변형을 막기 위한 방법이다. 즉 이전 호스트와 다음 호스트와의 체인관계가 아닌 중첩되는 구조를 갖는 체인구조로 기존의 방법과는 차별화된 구조이다. 그러나 두 개의 호스트들의 공모에 대해서는 방어될 수 있으나, 그 이상의 호스트간의 공모에 대비하기 위해서는 에이전트의 안전을 보장하기 위해서는 프로토콜의 확장이 필요하다. 그러나 프로토콜의 확장은 매우 복잡하여 구조적으로 시스템의 오버헤드를 가중시키게 된다.

Ⅲ. 절단공격으로부터 이동 에이전트 보안

3.1 체인관계구조와 에이전트 여정 경로프로토콜

호스트는 에이전트를 수행할 수 있는 능력을 갖추고 에이전트의 임수 수행을 위한 실행 환경을 제공한다. 이동 에이전트는 인터넷을 통해 목적지 호스트에 도달하여 자신을 복제하여 원래의 에이전트들과 똑같은 행동을 할 수 있을 뿐만 아니라 주어진 임무를 수행한다. 호스트와 호스트간의 체인관계를 형성하여 악의적인 요소 검출과 에이전트의 정상적인 수행을 방해하는 공격들이 있다. 절단공격은 체인을 공모하여 절단하는 경우 발생하는 위협이다. 그림 1은 [6]에서 제시한 체인관계 구조이다.

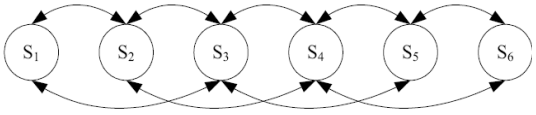

그림 1. [6]에서 제시한 체인관계 프로토콜
Fig.1 Chain relationship protocol proposed in [6]

그러나 이 구조는 다수의 공모가 이루어지는 경우 프로토콜을 확장하여야 하는 문제가 발생한다. 본 논문에서는 체인 연결을 쿼리 트리 구조를 통해 효과적인 체인관계 프로토콜로 절단공격을 방어 하도록 한다.

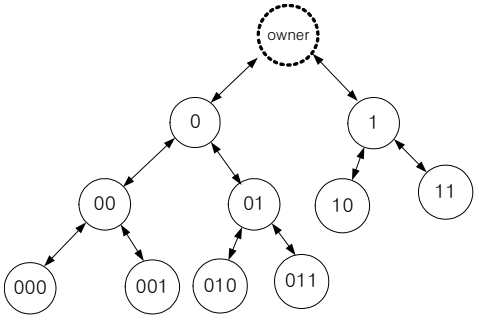

그림 2. 제안한 쿼리트리 체인관계구조
Fig 2. Proposed Structure of Query chain relationship tree

제안한 쿼리 트리 체인관계 구조(Query Chain Relationship Tree)는 에이전트의 여정에 따라 이동한 서버들의 경로를 알 수 있다. 에이전트가 호스트내 임무 완료되면 동적으로 여정을 결정하여 다른 호스트로 이동한다. 호스트에서 에이전트가 이동할 다음 호스트의 식별 ID와 k비트의 프리픽스 Pk를 함께 전송될 수 있도록 요구한다. 응답 호스트는 자신의 식별ID와 Pk를 보낸다.

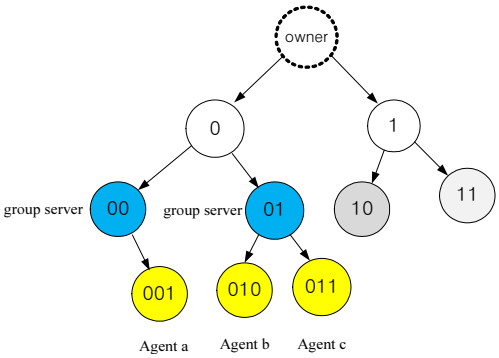

그림 3. 이동 에이전트의 여정 경로
Fig 3. itinerary(routes) of mobile agent

이는 쿼리 트리구조를 생성하기 위함이다. 결국 프리픽스 P_k에 "0"과 "1"을 붙인 새로운 k+1비트 프리픽스가

생성되어 에이전트 데이터에 캡슐화 한다.

```
Query:destination Host
While( Q is empty)
  prifix = pop a prefix form Q
    send Query command to destination Host
    reply =receive reply from host
    update= prefix of agent data
     prefix = k+1
end while
```

그림 4. 프리픽스 요구/응답의 쿼리 알고리즘
Fig. 4. Prefix request/respond query algorithm

초기 프리픽스 {'0', '1'}이 설정된 후 다음 이동이 발생할 경우 응답 호스트가 "0"을 응답하게 되면 이동 후 에이전트 임무 수행을 하고, 에이전트 데이터의 프리픽스(Pk)에는 "0" 기록되어진다. 그 후 에이전트가 이동하고자 하는 호스트가 프리픽스 값 "1"을 응답하게 되면 이는 "0,1"이 된다. 다음 에이전트의 여정에 따라 요구를 하게 되면 "0"을 응답하였다면 "0,1,0"으로 "agent b"의 여정이 될 것이다.

3.2 에이전트의 절단공격 방어 메커니즘

본 논문에서 사용된 각각의 변수의 의미를 표1에 나타내었다.

표 1. 사용된 표기법
Table 1. Terminology

$ENC_{S_i}^{pub}, ENC_{S_i}$	S_i의 공개키와 비밀키
R_i	S_i의 랜덤 수
$ENC_{S_o}^{pub}$	S_o의 공개키
$Sig_{s_i}(m)$	S_i의 자신의 비밀키로 서명
$H_i(m)$	일방향 해시
S_o	에이전트 생성자(owner)
S_i	i번째 호스트
$CSLO_i$	S_i번째의 호스트에서 캡슐화
Pk_i	i번째 호스트의 prefix value
SS_i	S_i의 임시키

3.2.1 에이전트($Agent_0$) 생성

S_o가 $Agent_0$를 생성하고, S_i의 제공된 데이터는 O_i로 캡슐화 된다.

$$CSLO_0 = ENC_{S_o}^{pub}(Sig_{S_i}(data_{S_i}), R_o, S_o, S_1, k_o)$$

$$S_o \rightarrow S_1 : CSLO_o$$

3.2.2 S_1에서 $Agent_0$의 실행

에이전트가 S_1로 이동하면, S_1는 R_1과 SS_i를 생성한다. 에이전트 S_1에서 에이전트를 실행한 후 실행 결과 데이터를 생성한다, 생성된 데이터는 $CSLO_1$으로 캡슐화 한다.

$$S_1 : Create\ R_1$$
$$S_1 : Create\ SS_1$$
$$S_1 : Create\ CSLO_1 = ENC_{S_o}^{pub}$$
$$(ENC_{S_1}^{pub}(Sig_{S_i}(data_{S_1}), R_1, S_1, S_2, k_1))$$

S_1은 S_0에게 S_1의 Pk_1과 SS_1을 보낸다.

$$S_1 \rightarrow S_0 : S_2,\ Pk_1,\ SS_i$$

수신된 S_o은 $CSLO_o$의 데이터를 일방향 해시함수로 암호화하여 다시 캡슐화 한다.

$$S_0 : S_0 = H_0(CSLO_o, R_0, S_1, S_2)$$
$$CSLO_o = Sig(CSLO_o, H_0, SS_1)$$

S_o에서 $CSLO_o$을 다시 S_1으로 보내 체인관계를 위해 검증한다. 그리고 다시 S_1에서 S_2로 보내어서 체인관계를 형성한다.

$$S_0 \rightarrow S_1 : CSLO_o(R_0, S_1, S_2)$$

$$S_1 \rightarrow S_2 : CSLO_1$$

3.2.3 S_n에서의 $Agent_o$의 실행

$Agent_o$가 S_n번째의 호스트에 도달하면 캡슐화된 데이터 $CSLO_0$, $CSLO_1$, $CSLO_2$, ... $CSLO_i, CSLO_{i+1}, \ldots.$, $CSLO_n$로 방문한 호스트의 수는 n+1을 초과여부를 파악한다. 만약 호스트 방문수가 초과되었다면 에이전트 실행을 멈추고 호스트는 다음과 같은 실행을 한다.

S_n : $Create\ R_n$
S_n : $Create\ SS_n$
S_n : $Create\ CSLO_n = ENC_{S_n}^{pub}$
$(Sig_{S_{n-1}}(data_{S_{n-1}}), R_n, S_n, S_{n-1}, k_n)$

S_n은 이전 호스트에서 수행되었던 수행과정을 확인하는 과정을 거치게 된다. 즉 S_n는 S_{n-1}에게 S_n의 Pk_n와 SS_n을 보내서 체인관계 형성을 위한 확인한다.

$S_n \rightarrow S_{n-1}$: S_{n-1}, Pk_n, SS_n

수신된 S_{n-1}은 $CSLO_{n-1}$의 데이터를 일방향 해시함수로 암호화화 하여 다시 캡슐화 한다.

S_{n-1} : $H_{n-1}(CSLO_{n-1}, R_{n-1}, S_n, S_{n-1})$
$CSLO_{n-1} = Sig_{n-1}(CSLO_{n-2}, H_{n-2}, SS_{n-1})$

S_{n-1}에서 $CSLO_{n-1}$을 다시 S_n으로 보내서 체인관계를 생성한다. S_n에서 임무수행 완료 후 S_n은 다음 여정 호스트인 S_o에게 프리픽스 값 Pk_0를 요청한다. S_0의 프리픽스 Pk_0를 S_n이 쿼리 체인구조에 의해 체인을 형성하여 여정 경로의 관리가 이루어진다. 그 후 S_n은 S_0에게 S_n에서의 $CSLO_n$을 보낸다. S_0는 이전 호스트에서 수행되었던 수행과정을 확인하는 과정을 거치게 된다. 즉 S_0는 S_n에게 S_n의 Pk_n와 SS_n을 보내서 체인관계 형성을 위한 확인을 한다. 체인관계를 통한 에이전트 여정 확인을 한다. 에이전트 소유자는 안전한 에이전트 여정에 따른 경로와 수행결과를 확인한다. 과정확인은 다음과 같다.

$S_0 \rightarrow S_n$: S_0, Pk_0, SS_0 : S_n의 자료요청
$S_n \rightarrow S_o$: $CSLO_n(R_n, S_n, S_o)$: S_n의 캡슐정보 전달

S_0는 캡슐화된 에이전트 수행 결과를 확인한다. 에이전트의 여정관리와 수행결과 확인이 종료되면 에이전트의 수명은 종료된다. 이러한 수행이 이루어짐으로써 에이전트 수행을 위해 호스트로부터 전달받은 프리픽스 Pk_i 값 010사이에 공격자에 의한 새로운 Pe_i의 값으로 가능한 삽입은 유일하게 "0"이다.

만약 절단이 성공하여 삽입이 되면 Pe_i의 값에 의해 다른 에이전트와의 충돌이 발생하여 공격되었음을 여정 과정에서 발견된다.

Ⅳ. 보안성 분석

본 논문에서 제안된 여정 프로토콜은 에이전트가 동적으로 이동하는 과정에서 체인관계 탐색을 프리픽스 값에 의해 탐색이 됨으로 길이가 짧아짐으로써 에이전트가 운반되어지는 길이가 짧아 호스트의 수행능력을 향상시킨다. 아울러 에이전트의 이동의 여정을 쉽게 할 수 있다. 기존 [6]에서 제시되는 방법은 최대 21개의 노드를 여정하게 된다. 하지만 우리의 프로토콜은 호스트에서 응답하는 P_k값에 의해 경로가 결정됨으로 쿼리트리 구조에 의한 여정이 이루어져 쉬운 여정관리가 이루어진다. 이는 체인관계의 복잡도가 줄어들어 호스트의 수행능력을 향상한다. 그러나 여러 에이전트가 동일 여정노드의 체인관계가 발생하면 기존 프로토콜과 같은 결과를 가져올 수 있다. 이러한 경우는 최대 연결노드의 데드라인 임계값을 지정함으로써 방지할 수 있다.

데이터의 기밀성 : 각 호스트에 의한 에이전트 데이터는 S_o의 공개키에 의해 캡슐화된 정보로 저장됨으로 에이전트 데이터의 안전히 보존된다. 각 노드에서 S_o의 공개키 암호와 S_i의 서명으로 이루지는 캡슐($Create\ CSLO_i$)은 데이터의 안전성을 보장한다.

부인봉쇄 : 에이전트 데이터의 실행의 안전과 사후 검출되는 오류에 대한 부인을 못하게 각 호스트는 공개키 기반의 서명을 위해 자신의 비밀키에 의해 서명 $Sig_{S_i}(data_{S_i})$이 이루어짐으로 부인할 수 없다.

전방위 프라이버시: 제시된 체인관계는 쿼리트리의 여정과에서 $CSLO_i$로 체인을 포함한 캡슐화가 이루어루어짐에 따라 이전 호스트의 프라이버시가 안전하다.

데이터의 무결성: 데이터의 훼손을 방지하기 위해서는 동시에 체인관계와 서명 후 암호 캡슐화가 이루어짐에 따라 안전하다.

삽입과 절단 공격 방지 : 삽입과 절단공격은 악의적인 호스트간의 공격으로 발생된다. 에이전트의 여정과정에서 호스트 자신의 서명과 일방향 해시에 의해 처리된다. 이는 몇 개의 악의적인 호스트에 의해 행해지는 공격으로 이전 호스트의 프라이버시가 훼손됨에 따라 발생

하지만 전방위 프라이버시가 보장되고, 프리픽스 값을 가지고 있는 트리구조의 여정으로 S_{i-1}과 S_i사이에 새로운 S_e을 삽입하거나 절단 공격은 이루어지기 어렵다. 예를 들어 그림 3의 $Agent\,c$를 공격하기 위해서는 Pk_i 값 010사이에 새로운 Pe_i의 값으로 가능한 삽입은 유일하게 "0"이다.

만약 절단이 성공하여 삽입이 되면 Pe_i의 값에 의해 $Agent\,b$ 와 충돌이 발생한다. 그러므로 체인절단과 삽입은 새로운 체인관계와 프리픽스 Pk_i값에 의해 여정노드 형성이 어려워진다. 또한 각 단계의 노드는 최대 2^n으로 생성되어 각 단계에서 그룹서버 생성이 이루어짐으로 그룹서버S_{i-1}가 설정됨으로 S_{i-1}의 공모만으로 공격은 어렵다.

V. 결 론

이동 에이전트가 네트워크 환경에서 안전한 실행은 악의적인 환경으로부터 안전성을 확보되어야만 가능하다. 이를 위해 에이전트 여정의 호스트들을 체인관계를 형성함으로써 악의적인 요소를 검출할 수 있을 뿐만 아니라 악의적인 남용으로부터 에이전트를 보호 할 수 있다. 본 논문에서는 이동 에이전트의 절단공격으로 보호될 수 있는 프로토콜을 제시하였다. 제안된 프로토콜은 기존의 체인관계보다 확장성이 뛰어난 개선된 쿼리 트리구조이다. 이 구조는 쿼리트리 체인관계를 형성함으로써 악의적인 요소들로부터 에이전트가 안전하게 실행되도록 보장한다. 또한 이동 에이전트의 체인관계 구조는 에이전트의 여정에 따른 프리픽스 Pk_i값을 포함한 체크 함(check sum)으로써 에이전트의 체인 구조의 확장이 가능하다. 전통적인 단면적 체인관계 구조를 개선하여 쿼리 트리구조로 확장하였다. 이로써 다중 체인관계로 인해 발생되는 프로토콜 확장에 따른 시스템 오버헤드의 문제점 해결 뿐 아니라 호스트의 공모된 절단 공격으로부터 에이전트의 안전한 실행보장을 하였다. 향후 연구는 체인관계 확장과 효과적인 탐지 기법에 대한 연구가 지속되어야 한다.

본 논문은 호원대학교 연구비 일부를 지원 받아 수행된 연구 결과임

참고문헌

[1] B.S. Yee. "A sanctuary for mobile agents," Technical Report CS97-537, UC San Diego, Dept. of Computer Science and Engineering, April 1997.

[2] G. Karjoth, N. Asokan, and C. Gülcü. "Protecting the computation results of free-roaming agents," In Proc. Second International Workshop on Mobile Agents (MA'98), K. Rothermel and F. Hohl, editors, LNCS 1477, pp.195 - 207, Springer-Verlag, 1998.

[3] N. M. Karnik and A. R. Tripathi. "Security in the Ajanta Mobile Agent System," Technical Report TR-5-99, University of Minnesota, Minneapolis, MN 55455, U. S. A., May 1999.

[4] A. Corradi, R. Montanari, and C. Stefanelli. "Mobile agents Protection in the Internet Environment"". In The 23rd Annual International Computer Software and Applications Conference (COMPSAC ''99), pages pp. 80 – -85, 1999.

[5] 정창렬,윤홍상,고진광,"TTS기반에서 디지털 서명의 실행 인증을 통한 에이전트의 무결성 보장 기법", 사단법인 한국통신학회 논문지 제31권 제6C호, pp.651-657, 2006.

[6] Darren Xu. LeinHarn, Mayur Narasimhan An. Junzhou Luo, "Improved free-roaming Mobile Agent Security Protocol against Colluded Truncation Attacks," Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC' 2006), vol. 2, pp.309-314, chicago, USA, sept. 2006.

정창렬(Chang-Ryul Jung)

1999년 순천대학교 컴퓨터교육과 교육학석사
2005년 순천대학교 컴퓨터과학과 이학박사

2003년 Visiting Researcher, University of Alberta, Canada
※ 관심분야 : Security, Agent security, RFID Privacy

김광오(Kwang-oh Kim)

1999년 경기대학교대학원 식공간 연출학 석사
2011년 경기대학교대학원 식공간 연출학 박사

2008년-현재 호원대학교 보건복지대학 교수
※ 관심분야 : 식공학 연출, 시각적 디자인 연출 등

송진국(Jin-kook Song)

1994년 홍익대학교 대학원 전자계산학과 이학석사
1998년 홍익대학교 대학원 전자계산학과 이학박사

1998년-2010년 진주산업대학교 교수
2011년-현재 경남과학기술대학교 컴퓨터공학부 교수
※ 관심분야 : 프로그래밍언어, 컴파일러, 보안 등

이성근(Sung-Keun Lee)

1987년 고려대학교 대학원 전자공학과 공학석사
1995년 고려대학교 대학원 전자공학과 공학박사

1997년-현재 순천대학교 멀티미디어공학과 교수
※관심분야 : 유비쿼터스 센서 네트워크, 멀티미디어 통신, 인터넷 QoS 등