대리서명을 이용한 모바일 에이전트의 안전성 강화 방법

김희선*, 백준상**, 이병천*, 김광조*
한국정보통신대학원대학교*
SECUi.COM**

Enhancing Security of Mobile Agent using Proxy Signature

Heesun Kim*, Joonsang Baek**, Byoungcheon Lee*, Kwangjo Kim*
Information and Communications Univ.*
SECUi.COM**

요 약

모바일 에이전트는 고객을 대신하여 호스트에서 고객의 서명을 대리 수행해야 하는 기능을 가지고 있다. 전자상거래 환경에 적용 가능한 모바일 에이전트 패러다임은 에이전트가 서명에 필요한 값들을 싣고 다니다가 적합한 주문(bidding) 정보를 가진 호스트를 만나면, 그 호스트가 제공하는 환경에서 서명기법을 수행할 수 있으며, 이 때 악의의 호스트로부터 여러가지 공격을 받을 수도 있다. 이러한 공격에 대응하기 위한 기존의 연구로는 고객이 암호화된 서명 함수를 생성하고 이를 에이전트가 호스트에서 수행하도록 하는 기법이 있다. 이러한 경우 에이전트는 모바일 코드로서 함수코드 혹은 실행 코드를 운반해야 할 것이며, 에이전트에 대한 공격에 모바일 코드가 변형될 경우에도 호스트는 이것을 탐지하지 못하고 잘못된 함수를 그대로 수행할 수도 있다. 이러한 문제점을 고려하여 본 고에서는 대리서명 기법을 이용하여 보다 안전하고 효율적으로 모바일 에이전트 패러다임을 구현해 보았다. 또한 암호화된 서명 함수 기법에서 함수의 수행권한이 호스트에 전적으로 부여됨으로 인해 발생할 수 있는 문제점을 고려하여 실패-중단 서명 기법을 응용한 1회 대리 서명 기법도 제안하였다.

제 1 절 서론

1.1 모바일 에이전트 정의 및 기능

모바일 에이전트는 원격지의 여러 가지 다른 실행 환경에서 이용할 수 있는 자동의 소프트웨어 개체이다. 모바일 에이전트의 이동성과 자동성은 영구적인 연결을 필요로 하지 않기 때문에, 모바일 에이전트는 낮은 대역폭의 통신 연결이나 비동기 통신을 제공하기에 적합하며, 이기종 환경에서 수행되는 작업에 유리하다 [3].

이러한 모바일 에이전트의 특성은 전자 상거래에 이상적으로 적용될 수 있어, 특정 물건, 특정 서비스를 검색하거나 서비스 요구자를 대신하는 협상 행위, 상품 판매 행위 등의 기능을 수행할 수가 있다.

모바일 에이전트가 전자상거래에 적용되는 한 사례로서 항공요금 정보 검색을 위한 에이전트의 시나리오를 생각해 볼 수 있다. 에이전트가 여러개의 항공사 서버를 차례로 방문하여 사용자가 요구하는 적합한 항공편을 발견하기 위해 질의를 하고, 일단 최상의 조건이 결정되면 항공편을 예약하고자 한다. 이에 대해 발생할 수 있는 여러 공격들로는 에이전트가 방문한 서버에 대한 기억

을 지우거나 악의가 있는 호스트의 항공편을 선택하도록 에이전트의 상태나 코드를 변조시킬 수가 있다. 또는 에이전트가 내부적으로 사용하는 주문가격을 올리거나 프로토콜의 수행없이 에이전트의 전자화폐를 훔치는 공격도 가능하다. 심지어는 모바일 에이전트가 항공편 예약이나 구입시 디지털 서명을 하고자 할때, 에이전트가 운반하는 사용자의 비밀키가 도용될 수도 있다.

특히, 이와 같이 비밀값을 전달하고 이를 이용하여 계산해야 하는 과정에 있어서 신뢰할 수 없는 환경에서 수행하는 모바일 코드에 대한 기본적인 문제점들은 다음과 같이 요약해 볼 수 있다 [4]:

1) 코드와 실행의 무결성 : 모바일 에이전트가 악의를 품은 호스트의 공격으로부터 자신을 보호할 수 있는가?

2) 코드의 기밀성 : 모바일 에이전트가 수행하고자 하는 프로그램을 숨길 수 있는가?

3) 공개된 기밀 연산 : 모바일 에이전트가 원격지에서 사용자의 비밀키를 노출시키지 않고 문서에 대한 디지털 서명을 수행할 수 있는가?

이러한 문제들은 수학적으로 증명 가능한 계산의 무결성과 비밀성에 대한 연산을 이용하는 소프트웨어적인 접근의 해결책이 요구된다. 따라서 모바일 코드가 신뢰할 수 없는 컴퓨팅 환경에서 수행되더라도 암호 프리미티브를 수행할 수 있거나 본래의 사용자측과의 상호작용 없이 자동으로 수행 가능하게 하도록, 모바일 코드에 대해 비밀성을 제공하기 위한 소프트웨어적인 접근 방법에 초점을 두어 논의하고자 한다.

1.2 보안상 문제점 연구 사례

모바일 에이전트 환경에서 고려해 볼 수 있는 보안상의 문제점은 에이전트가 방문한 호스트에 가하는 공격과 호스트가 에이전트에 가하는 공격으로 나누어 볼 수 있으며, 후자가 보다 어려운 문제로 보여진다.

악의를 가진 호스트로부터 에이전트를 보호하기 위한 연구로는 에이전트 위조 발견과 위조 방지의 두 가지 측면으로 나누어 볼 수 있다. 에이전트에 대한 위조 발견 연구는 주로 잘못된 프로세스를 수행하는 호스트의 신분을 추적하여 잘못된 동작들을 증명하는 것을 포함한다. 이에 대한 연구로 Vigna [11]에 의해 제시된 에이전트의 수행을 기록하는 추적 메카니즘, Yi 등 [12]에 의해 제시된 에이전트 서비스 센타 제안, Kotzanikolaou 등 [8]에 의해 제시된 멀티-에이전트 시스템 등이 있다.

에이전트에 대한 위조 방지 연구는 적극적 방지와 수동적 방지로 이루어져 왔다. 수동적 방지 기법은 시스템의 구성과 구조에 대한 연구가 진행되었고, Farmer 등 [6]에 의해서 제안된 신뢰할 수 있는 실행 환경에서만 수행 가능한 모바일 에이전트 기법과 Merwe와 Sholms [10]에 의해 제시된 분산 객체간에 구현된 거래 에이전트 시스템들이 그 예이다.

적극적 방지에 관한 연구는 모바일 에이전트의 장점에 대해서는 고려하지 않고 에이전트를 호스트의 공격으로부터 보호할 수 있는 기법을 제안하는 것에만 초점이 맞추어져 왔으며, 이에 대한 안전한 하드웨어 장치에 대한 접근은 높은 비용으로 인해 그 개발이 매우 저조한 편이어서 소프트웨어적 기법의 접근이 보다 활발히 연구가 되고 있다. 이에 대한 연구로는 Hohl [7]에 의한 time-limited token-data를 운반하는 에이전트에 관련된 기법과, Loureiro와 Molva [9]에 의한 모바일 코드를 위한 프라이버시 연구가 있다. 그 밖의 연구로, Sander와 Tschudin [4]이 제안한 CEF(Computation with Encrypted Function)의 개념 기반의 undetachable 서명 기법이 있다. 이는 신뢰할 수 없는 호스트에서 수행되는 에이전트를 보호하기 위한 암호화된 서명함수로서, $s \circ f$의 (암호화 함수 f, 서명 함수 s) 함수 분리의 어려움에 안전성을 기반으로 하고 있으며, 대수학적

homomorphic 함수인 birational 함수를 이용하여 구현되었다. 반면, Kotzanikolaou 등 [3]은 RSA 암호 기법을 기반으로 undetachable 서명함수를 구현하였다. 이들의 기법은 에이전트가 사용자의 비밀키를 노출없이 서명을 생성하는데 사용할 수 있는가의 문제를 암호화된 서명함수를 제공하는 것으로 해결하였다.

본 연구에서는 [3]과 [4]에서 제시하고 있는 비밀키의 안전한 계산에 관한 문제를 대리서명기법을 적용함으로써 에이전트가 서명 함수를 전달하지 않고도 공개 가능한 데이타의 전송으로 사용자의 비밀키의 안전성을 보장하고 사용자를 대신한 서명을 생성하는 것을 보일 것이다. 한편, undetachable 서명 기법과 대리서명을 이용한 기법에서는 하나의 서버가 하나 이상의 유효한 주문 정보에 대해 각각 유효한 서명을 계산해 내는 것이 가능하다. 이로 인해 발생할 수 있는 분쟁을 방지하기 위해서는 주문 정보 간의 우선순위 정책(priority policy)이 요구되는데 [3], 본 연구에서는 1회성의 특성을 지닌 서명 기법을 적용하여 이러한 분쟁을 최소화시킬 수 있음을 보이고자 한다.

본 고의 구성은 다음과 같다. 2절에서는 비밀값의 안전한 계산 문제의 연구에 대한 기존의 기법들을 살펴보고, 보다 효율적인 적용 가능한 기법을 제시한다. 3절에서는 2절에서 소개한 기법이 갖는 한계성을 지적하고 이를 개선한 새로운 기법을 제시한다. 제시된 기법에 대한 안전성은 4절에서 논의하며, 결론 및 향후 과제는 5절에서 기술한다.

제 2 절 비밀키의 안전한 계산

2.1 Undetachable 서명 기법

위조 공격이나 정찰 공격 (spying attack)에 대한 모바일 에이전트를 보호하는 문제 해결을 위해 [3]과 [4]의 연구가 진행되었다. 이들은 모바일 에이전트가 어떻게 사용자의 비밀키를 노출하지 않고 안전하게 유지하고 운반하여 사용자의 서명을 수행할 수 있는가의 문제를 해결하기 위해, 모바일 코드에 서명 함수를 실어 호스트로 전달할 것을 제안하였다. 이를 위해 Sander와 Tschudin [4]은 CEF 개념을 기반으로 하여 암호화된 서명함수인 undetachble 서명 기법을 제안하였고, 이 기법은 다음과 같은 문제를 해결하고자 하였다 [4]:

> 엘리스가 함수 f를 계산할 알고리즘을 가지고 있다. 봅은 입력 값 x을 가지고 있고 엘리스를 위해 기꺼이 $f(x)$를 계산하려 한다. 그러나 엘리스는 봅이 f에 관한 어떠한 정보도 알기를 원하지 않는다. 게다가 봅은 $f(x)$을 계산하는 동안 엘리스와 통신을 할 필요가 없어야 한다.

이들은 이러한 개념을 birational 함수를 이용하여 구현하였다. 그러나 이들이 제안한 기법은 안전성을 보장할 수 없으며 [4], Coppersmith, Stern, Vaudenay 공격에도 [13] 취약함이 밝혀져 있다. 반면, Kotzanikolaou 등 [3]은 RSA 암호 기법의 안전성을 기반으로 하여 undetachable 서명 기법을 제안하였다.

본 고에서는 [3]에서 제시한 RSA 암호 기반의 undetachable 서명 기법을 살펴보도록 한다.

표기법

- $hash$: 해쉬 함수
- C: 사용자의 신분 정보
- req_C: 고객의 요구조건 (예를 들어 상품 구매의 경우, 물품 설명, 최대 가격 정보, 배달 기한, 요구조건 발급 일시 등의 정보를 포함)

- $h = hash(C, req\_C)$: RSA modulus n으로 계산되는 바이너리 스트링

- S: 호스트 신분 정보

- bid_S: 호스트의 주문 정보

키 생성

- 큰 소수 p, q에 대하여 $n = pq$인, 모듈러 n 선택 및 공개.

- 사용자의 공개키 e를 계산, $1 < e < \phi(n) = (p-1)(q-1)$, s.t. $gcd(e, \phi(n)) = 1$.

- 사용자의 비밀키 d를 계산, s.t. $1 < d < \phi(n), de = 1 \bmod \phi(n)$.

에이전트의 준비 단계

입력 x에 대한 암호화 함수 f와 암호화된 서명 함수 f_{signed}은 다음과 같이 정의된다:

$$f(x) = h^x \bmod n$$
$$f_{signed}(x) = k^x \bmod n$$

여기서, $k = h^d \bmod n$는 h에 대한 사용자의 RSA 서명이고, $f_{signed}(x)$은 사용자의 RSA 서명 함수 $s(x) = h^d \bmod n$를 암호화한 함수이다. 즉,

$$f_{signed}(x) = s \circ f(x) = s(f(x)) = s(h^x) = (h^x)^d = (h^d)^x = k^x$$

이다. 위에서 제시된 함수 $(f(x), f_{signed}(x))$는 모바일 코드로서 (C, req_C)와 함께 사용자가 지정해준 호스트의 IP 주소에 따라 호스트를 돌아다니며, 적합한 주문 정보를 제공하는 호스트를 찾는다.

에이전트 수행 단계

모바일 에이전트는 호스트에 의해서 입력 $x = hash(S, C, bid_S)$에 대해 수행된다:

$$m = f(x) = h^x \bmod n$$
$$z = f_{signed}(x) = k^x \bmod n = (h^d)^x \bmod n = (h^x)^d \bmod n = m^d \bmod n = s(m)$$

이와 같이, 모바일 에이전트를 수행하여 RSA 서명 (m, z)을 얻는다. 그리고, 모바일 코드 (S, C, bid_S), x, (m, z)은 다시 호스트에 의해 모바일 에이전트에 실려서 사용자에게로 전달된다.

살펴본 바와 같이 사용자를 대신한 호스트의 모바일 코드의 서명 수행은 사용자의 서명을 호스트가 대신 서명하는 대리서명의 개념과 동일하게 볼 수 있다. 따라서 본 연구는 대리서명 기법을 적용하여 모바일 에이전트 패러다임이 어떻게 구현되는지 살펴보기로 한다.

2.2 대리 서명 기법 적용

2.2.1 Petersen과 Horster의 대리 서명 기법

Petersen과 Horster가 제안한 self-certified key는 대리 서명의 응용에 적용가능하다 [1],[14]. 이들이 제시한 self-certified key를 이용한 대리서명 기법은 다음과 같은 대리 서명의 요구사항 [5]을 만족시킨다.

- 강한 위조 : 대리 서명자가 본래의 서명자에 대한 유효한 대리 서명을 생성할 수 있다. 그러나, 본래의 서명자를 포함한 어떠한 제 삼자라도 대리 서명을 위조할 수는 없다.
- 강한 확인 : 누구라도 대리 서명으로부터 대리 서명자를 확인할 수 있다.
- 강한 부인방지 : 대리 서명자는 서명 이후에 누구에게도 유효한 서명을 만든 사실을 부인할 수 없다.

self-certified key를 이용한 대리서명 기법에서는 본래의 서명자가 대리서명 키에 대한 정보를 얻을 수 없으며 대리 서명자 대신 서명을 수행할 수도 없다.

본 고에서는 키 발급 프로토콜을 대리서명에 적용한 모델하에서, 본래의 서명자를 A로 대리 서명자를 B로 가정하여 기술한다.

표기법

- ID_B : 대리 서명자 B의 신원
- x_A : 본래의 서명자 A의 장기 비밀키
- α : Z_p^* 상의 위수 q인 부분군의 생성자

안전한 키 발급 프로토콜

1) 본래의 서명자 A

   a) 비밀키 k_A 선택, $k_A \in_R Z_q^*$.
   b) r_A 계산, $r_A := \alpha^{k_A} \bmod p$.
   c) r_A을 B에게 전달.

2) 대리 서명자 B

   a) 비밀키 b 선택, $b \in_R Z_q^*$.
   b) r_B 계산, $r_B := r_A \cdot \alpha^b \bmod p$.
   c) ID_B와 r_B를 A에게 전달.

3) 본래의 서명자 A

   a) x_A를 이용하여 s_A 계산, $s_A := x_A \cdot hash(ID_B, r_B) + k_A \bmod q$.
   b) s_A을 B에게 전달.

4) 대리 서명자 B

   a) x_B 계산, $x_B := s_A + b \bmod q$.
   b) x_B에 대응하는 공개키 y_B를 계산, $y_B := \alpha^{x_B} \bmod p$.

   이와 같이, 대리 서명자 B는 대리 서명자의 신분 확인에 대한 서명으로 (r_B, x_B)를 얻게 되고, 생성한 위탁키 x_B로 기존의 서명 알고리즘을 이용하여 서명을 수행한다. 비밀키 x_B에 대응되는 공개키 y_B에 대한 인증은 $y_B := y_A^{hash(ID_B, r_B)} \cdot r_B \bmod p$로 수행하며, 이 공개키를 이용하여 서명을 검증할 수 있다.

2.2.2 대리서명 기법 적용 예

앞에서 살펴본 Petersen과 Horster가 제안한 self-certified key 생성 프로토콜을 이용한 대리서명 기법을 모바일 에이전트 패러다임에 그대로 적용시키기 위해선 몇 가지 문제점이 있다. 이들의 기법은 interactiveness의 특성을 갖는 프로토콜인데 반해, 모바일 에이전트 패러다임은 어떤 호스트가 사용자의 서명을 수행할 것인지를 예측할 수 없고, 모바일 코드가 수행되는 동안 사용자와의 상호 메시지 교환이 이루어지지 않을 것을 가정해야 하는 non-interactiveness 특성을 지닌다. 즉, 사용자는 모든 메시지를 한꺼번에 전달해야만 한다. 따라서 이들의 기법에 non-interactiveness의 특성을 결합시킨 Kim [14]의 키 생성 기법을 이용하여 대리서명을 수행하도록 한다. 따라서 Petersen과 Horster의 기법과 Kim의 기법을 응용하여, 대리서명자의 신원과 대리서명자가 생성한 공개키에 본래의 서명자가 서명을 수행하여 키를 만들어내는 과정을, 본 제안 기법에서는 사용자가 사용자의 요구조건에 서명을 수행한 결과를 키로 생성하도록 수정하였다. 본 연구에서 대리서명 기법을 모바일 에이전트 패러다임에 적용한 예는 다음과 같다.

표기법

- (x_C, y_C) : 사용자 C의 장기 비밀키와 공개키 쌍, $y_C = \alpha^{x_C} \bmod p$
- (a, b) : 호스트 S의 장기 비밀키와 공개키 쌍, $b = \alpha^{a} \bmod p$
- $Sign()$: 일반적인 서명 기법

대리서명 기법을 이용한 프로토콜 예

1) 사용자 C은 로컬 환경에서 다음을 수행한다.

   a) 비밀키 k_C 선택, $k_C \in_R Z_q^*$.
   b) r_C 계산, $r_C := \alpha^{k_C} \bmod p$.
   c) s_C 계산, $s_C := x_C \cdot hash(req\_C, r_C) + k_C \bmod q$.

   사용자는 (C, r_C, s_C, req_C)을 실은 모바일 에이전트를 네트워크 상으로 전달한다. 모바일 에이전트는 사용자가 지정한 호스트의 IP 주소에 따라 호스트들을 차례로 방문한다. 사용자의 요구조건에 유효한 주문정보를 가진 호스트를 만나면 모바일 에이전트는 호스트가 제공해주는 수행 환경에서 다음을 수행하게 된다.

2) 호스트 S는 먼저 에이전트가 가져온 사용자의 키가 정당하게 생성된 값인지를 검증하고, 주문정보 bid_S를 이용하여 서명할 메시지 $msg = hash(S, C, req_C, bid_S)$를 계산한다. 또한, 모바일 에이전트로부터 받은 사용자의 키를 이용하여 대리서명을 수행할 수 있는 위탁키를 생성한다. 그리고, 메시지 msg에 대한 서명을 수행하고, 그 결과값들을 에이전트를 통해 사용자에게 전달하게 된다.

   a) s_C에 대하여 식 $\alpha^{s_C} := y_C^{hash(req_C, r_C)} \cdot r_C \bmod p$을 계산하여 에이전트가 전달한 키 값을 검증.
   b) 대리 서명을 수행하기 위한 비밀키 x_S 계산, $x_S := s_C + a \cdot hash(req\_C, r_C) \bmod q$.
   c) 대리 서명 검증에 사용될 공개키 y_S 생성, $y_S = \alpha^{x_s} \bmod p$.
   d) 이렇게 생성된 비밀키 x_S로 서명을 수행하여, 메시지 msg에 대한 서명 σ를 생성, $\sigma = Sign(msg)$.
   e) 호스트 S는 $(S, C, bid_S,), msg, y_S, \sigma$를 에이전트를 통해 사용자 C에게 전달.

3) 사용자 C는 에이전트로부터 전달받은 메시지들을 검증한다.

   a) $m = hash(S, C, req_C, bid_S)$를 계산 후, $m = msg$ 인지를 체크.
   b) 공개키를 검증하여, 사용자가 전달한 키를 이용하여 호스트의 위탁키가 올바르게 생성되었는지 계산, $y_S := (y_C \cdot b)^{hash(req\_C, r_C)} \cdot r_C \bmod p$.
   c) 검증된 공개키를 이용하여 서명이 메시지 msg에 대하여 올바르게 수행되었는지 검증.

2.3 undetachble 서명과 대리서명 적용 기법의 비교 및 특징

위와 같이 모바일 에이전트 패러다임에 대리서명 기법을 적용하여 사용자의 비밀키를 노출시키지 않고도 에이전트로 하여금 호스트의 환경에서 사용자의 대리서명을 수행하게 할 수 있다. 대리서명 기법의 적용은 undetachable 서명 함수의 쌍 $(f(x), f_{signed}(x))$에 대한 메시지 $2|n|$ 대신 (r_C, s_C)에 대한 $|p| + |q|$ 길이 만큼의 메시지를 전달하게 된다. 또한, undetachable 서명 함수는 에이전트가 전달한 함수를 직접 메시지 x를 입력하여 실행시켜서,

$$m = f(x) = h^x$$

$$z = f(x) = k^x = h^{xd} = m^d$$

서명 (m, z)을 얻고, $m = z^e$와 같이 사용자의 공개키 e로 서명값을 복호화하여 서명이 사용자의 비밀키로 서명이 되었는지를 검증할 수 있다. 따라서, 서명을 수행한 이후에 사용자의 공개키로 서명이 제대로 검증되지 않는다면 호스트는 그때서야 모바일 코드가 공격을 받았거나, 악의의 사용자로부터 전달된 에이이전트임을 알고 모바일 코드 수행을 중단하게 된다. 그러나 본 연구에서 제안한 대리서명을 적용한 기법은 호스트가 서명을 수행하기 전에 에이전트가 전달한 데이타 (r_C, s_C)로부터

$$\alpha^{s_C} = y_C^{hash(req\_C, r_C)} \cdot r_C (\mathrm{mod} p)$$

을 수행하므로 전달된 키가 사용자 C로부터 정당하게 생성되어 전달된 값인지를 검증할 수가 있으며, 이를 검증할 수 없을 경우 서명 수행 전에 모바일 코드의 수행을 중단할 수가 있으므로 보다 효율적이다.

또한 undetachable 서명 함수와는 달리 대리서명을 이용한 기법에서는 사용자의 서명에 호스트의 서명 능력이 부가되었다는 특징을 가지므로, 비밀키 a를 알고 있는 호스트가 서명을 하였다는 것을 증명할 수 있다. 따라서, 누구나 함수를 얻기만 하면 자신의 자격에 관계없이 서명을 생성할 수 있는 undetachable 서명 기법에 비해 본 제안 기법이 보다 안전하다.

본 연구의 제안 기법이 모바일 에이전트 패러다임에서 효과적으로 적용되기 위해서 undetachable 서명 함수를 이용한 기법과 마찬가지로 다음과 같은 특징을 지닌다 [3].

- 호스트의 요구조건인 req_C에 상충하는 호스트의 주문 정보 bid_S에 대한 서명은 무효화된다.

- 모바일 에이전트에 의해 수행되는 전체 프로토콜은 비대칭적 특성을 갖는다. 즉, 프로토콜의 수행이 특정 호스트에만 국한된 것이 아니므로, 호스트에 대한 위장 공격에 대비하기 위해선, 호스트의 주문정보 bid_S에 대한 호스트의 서명을 요구할 수 있다.

- 본 제안 기법은 전달되는 모바일 코드를 비밀성을 고려하지 않고 기술하였다. 그러나 모바일 코드에 대한 비밀성 보장과 데이타의 무결성 체크를 위해서는 에이전트가 호스트에 전달하는 데이타들을 사용자의 비밀키로 서명을 함으로써 호스트가 모바일 코드의 무결성을 확인할 수 있어야 한다. 또한, 호스트로부터 사용자에게 전달되는 에이전트의 데이타는 호스트의 서명과 함께 사용자의 공개키로 암호화하면 전달하여 비밀성과 무결성을 동시에 보장할 수 있다.

- 하나의 구매 수행을 위해 에이전트가 여러 개의 호스트로부터 하나 이상의 유효한 주문 정보에 대해 사용자의 서명을 수행할 수 있으며, 심지어는 하나의 호스트가 하나 이상의 유효한 주문 정보에 대해 사용자의 서명을 수행하여 악의를 도모할 수도 있다. 이와 같은 문제의 발생을 피하기 위해 우선순위 정책을 설정할 필요가 있다. 가장 간단한 방법은 에이전트가 특정 호스트로부터 맨 처음 적당한 주문정보에 대해 서명 프로토콜을 수행한 후에, 즉시 구매 수행 메카니즘을 종료하고, 에이전트가 사용자에게로 돌아오게 할 수 있다. 그럼에도 불구하고 분쟁이 발생할 경우에는 서명 수행에 타임 스탬프 (time-stamps) 정보를 이용하므로써 가장 먼저 수행된 서명만을 인정해 줄 수도 있다. 그러나, 이와 같은 분산환경에서는 타임 정보의 안전성과 정확성을 보장하기가 어렵다. 특히, 하나의 호스트가 여러 개의 유효한 주문 정보에 대해 사용자의 서명을 수행하여 발생하게 되는 분쟁의 경우, 타임 스탬프가 임의로 조작될 가능성이 높으므로 유효성 보장은 더욱 어렵다. 따라서, 에이전트 중재자 혹은 신뢰기관과 같은 제 삼자의 개입이 필요할 것이다.

이러한 문제점은 모바일 에이전트 패러다임 환경의 특성상, 사용자의 요구 조건에 대응하는 유효한 주문 정보가 하나 이상 존재할 가능성이 있으며, 사용자의 서명이 호스트의 환경에서 모바일 에이전트에 의해 대리 수행되어야 하기 때문에 발생하게 되는 것이다. 따라서 본 고에서는 이러한 분쟁의 가능성을 피하기 위해 적어도 하나의 호스트에서는 하나의 유효한 주문 정보에 대한 사용자의 서명을 수행하도록 유도하는 서명 기법을 다음 절에서 제안한다.

제 3 절 1회 대리 서명 기법

본 절에서는 앞서 살펴본 바와 같이 우선순위 정책의 설정을 바탕으로, 발생 가능한 분쟁을 최소

화 시킬 수 있는 서명 기법을 제안한다. 즉, 하나의 호스트에 있어서 발생할 수 있는 여러 개의 유효한 주문정보의 서명에 대한 분쟁을 피하기 위해서, 하나의 호스트는 하나의 유효한 주문정보에 대해서만 사용자의 대리서명을 수행할 수 있음을 보장해 주는 것이다. 이것은 호스트에 전적으로 부여되는 서명 수행 권한을 1회로 제한시킴으로써 가능하다. 제안 기법을 설명하기 위해 응용한 서명기법을 먼저 서술한 후에 본 기법을 설명하겠다.

이 절의 전반을 통해 사용할 표기는 다음과 같다.

표기법

- p, q : 큰 소수로서 공개된 값, s.t. $q|(p-1)$
- α : 위수 q를 갖는 Z_p^*의 곱셈 부분군의 생성자

3.1 실패-중단 서명 기법

본 연구에서는 대리 서명자의 서명 권한을 제한하기 위해 E. Heyst와 Pedersen의 실패-중단 (fail-stop) 서명 기법의 1회성을 이용하였다. 즉, 실패-중단 서명 기법은 오직 한 개의 메시지에 대해서만 주어진 키를 가지고 서명을 할 수 있는 1회용 서명 기법으로써, 아주 강력한 제 삼자가 서명을 위조한다고 하더라도 높은 안전성을 제공한다.

다음은 실패-중단 서명기법을 기술하였다.

키 생성 단계

1) 신뢰기관 T
   a) p, q, α 생성
   b) 비밀 값 a 선택, $a \in_R Z_q^*$.
   c) β 계산, $\beta := \alpha^a \bmod p$.
   d) p, q, α, β를 공개.

2) 서명자 A
   a) 비밀 키 x_1, x_2, y_1, y_2를 선택, $x_1, x_2, y_1, y_2 \in_R Z_q^*$.
   b) β_1, β_2 계산, $\beta_1 = \alpha^{x_1}\beta^{x_2}, \beta_2 = \alpha^{y_1}\beta^{y_2} \bmod p$.
   c) β_1, β_2를 공개.

서명 및 검증 프로토콜

1) 서명 기법
   a) 서명할 메시지 m에 대한 서명값 $s_{1,m}, s_{2,m}$을 생성.
      $s_{1,m} = x_1 + my_1 \bmod q, s_{2,m} = x_2 + my_2 \bmod q$

2) 검증 기법

a) v_1, v_2을 계산.

$v_1 = \beta_1 \beta_2^m \bmod p, v_2 = \alpha^{s_{1,m}} \beta^{s_{2,m}} \bmod p$

b) 만일 $v_1 = v_2$이면, 서명을 인정.

이번에는 같은 키를 가지고 서로 다른 메시지에 대해 여러 번 서명을 한다는 것을 가정하여 어떻게 1회용 서명을 보장할 수 있는지 살펴보기로 한다.

서명자가 동일한 키 x_1, x_2, y_1, y_2를 이용하여 메시지 m과 또 다른 메시지 m'에 대해서 서명을 수행한다고 가정한다면,

$$s_{1,m} = x_1 + my_1 \bmod q, s_{2,m} = x_2 + my_2 \bmod q$$

$$s_{1,m'} = x_1 + m'y_1 \bmod q, s_{2,m'} = x_2 + m'y_2 \bmod q$$

이 생성되어 x_1, x_2, y_1, y_2의 비밀키 값을 계산할 수 있게 된다. 따라서, 서명자는 비밀키가 노출되지 않도록 동일한 키로 여러 개의 서로 다른 메시지에 대한 서명을 수행하지 않을 것이다.

3.2 1회용 대리 서명 기법

이번 절에서는 앞서 살펴본 대리 서명을 위한 키 생성과 실패-중단 서명 기법을 이용하여 1회용 대리 서명 기법을 제안한다. 이 기법은 사용자의 서명을 대신 수행하는 대리자 호스트가 사용자의 비밀키에 대한 비밀성을 보장하는 위탁키를 생성한 다음 위탁키로 실패-중단 서명을 수행하게 한다. 즉, 오직 한 개의 메시지에 대해서만 주어진 위탁키를 가지고 대리서명을 수행하도록 하는 서명 기법이다. 따라서, 사용자의 서명을 사용자 비밀키의 노출없이 호스트에서 수행하도록 하며, 동시에 악의를 가진 호스트가 취할 수 있는 여러 개의 유효한 주문정보에 대한 서명 생성의 권한을 1회로 제한시키는 기능도 제공한다.

이 서명 기법은 4개의 비밀키를 필요로 하므로, 앞 절에서 제시한 대리서명 기법 적용의 키 생성 단계를 약간 수정한다. 그리고, 실패-중단 서명의 수행을 위해 공개키 값들을 생성하여 공개하기 위한 에이전트 중재자 혹은 신뢰기관을 가정하며, 사용자의 요구조건인 req_C는 시간 정보를 포함한 값이므로 유일한 값으로 설정됨을 전제로 한다.

표기법

- x_C, y_C, y'_C : 사용자의 장기 비밀키, 공개키 $x_C \in_R Z_q^*, y_C = \alpha^{x_C} \bmod p$
- a_i, b_i : 호스트의 장기 비밀키, 공개키 $a_i \in_R Z_q^*, i \in \{1,2,3,4\}, b_j = \alpha^{a_j} \bmod p, j \in \{1,3\}, b_k = \beta^{a_k} \bmod p, k \in \{2,4\}$
- msg : $h(S, C, req_C, bid_S)$로 계산, 서명 받아야 할 메시지

초기화 단계

- 신뢰기관 T

    a) p, q, α 생성

    b) 비밀 값 a 선택, $a \in_R Z_q^*$.

    c) β 계산, $\beta := \alpha^a \bmod p$.

    d) p, q, α, β를 공개.

위탁키 생성 및 서명 단계

1) 사용자 C는 로컬 환경에서 다음을 수행한다.

   a) 비밀키 $k_{C_1}, k_{C_2}, k_{C_3}, k_{C_4}$ 선택, $k_{C_i} \in_R Z_q^*, i \in \{1,2,3,4\}$.
   b) $r_{C_1}, r_{C_2}, r_{C_3}, r_{C_4}$ 계산, $r_{C_i} := \alpha^{k_{C_i}} \bmod p, i \in \{1,3\}$, $r_{C_j} := \beta^{k_{C_j}} \bmod p, j \in \{2,4\}$.
   c) $s_{C_1}, s_{C_2}, s_{C_3}, s_{C_4}$ 계산, $s_{C_i} := x_C \cdot hash(req\_C, r_{C_i}) + k_{C_i} \bmod p, i \in \{1,2,3,4\}$.

   사용자는 $C, (r_{C_1}, r_{C_2}, r_{C_3}, r_{C_4}), (s_{C_1}, s_{C_2}, s_{C_3}, s_{C_4}), req_C$을 실은 모바일 에이전트를 네트워크 상으로 전달한다. 모바일 에이전트는 사용자가 지정한 호스트의 IP 주소에 따라 호스트들을 차례로 방문한다. 사용자의 요구조건에 유효한 주문 정보를 가진 호스트를 만나면 모바일 에이전트는 호스트가 제공해 주는 수행 환경에서 다음을 수행하게 된다.

2) 호스트 S는 주문정보 bid_S를 이용하여 서명할 메시지 $msg = hash(S, C, req_C, bid_S)$를 계산한다. 또한, 모바일 에이전트로부터 받은 사용자의 키를 이용하여 대리서명을 수행할 수 있는 위임키를 생성한다. 그리고, 메시지 msg에 대한 서명을 수행하고, 그 결과값들을 에이전트를 통해 사용자에게 전달하게 된다.

   a) 에이전트가 전달한 키 값들을 $\alpha^{s_{C_i}} := {y_C}^{hash(req_C, r_{C_i})} \cdot r_{C_i}, i \in \{1,3\}, \beta^{s_{C_j}} := {y'_C}^{hash(req_C, r_{C_j})} \cdot r_{C_j}, j \in \{2,4\}$와 같이 검증. 검증되지 않으면 모바일 코드의 수행을 중단.
   b) 대리 서명을 수행하기 위한 비밀키 $x_{S_1}, x_{S_2}, x_{S_3}, x_{S_4}$ 계산.
      $x_{S_i} := s_{C_i} + a_i \cdot hash(req\_C, r_{C_i}), i \in \{1,2,3,4\}$.
   c) 대리 서명을 위한 공개키 β_1, β_2 계산, $\beta_1 = \alpha^{x_{S_1}} \beta^{x_{S_2}}, \beta_2 = \alpha^{x_{S_3}} \beta^{x_{S_4}} \bmod p$
   d) 생성된 비밀키 $x_{S_1}, x_{S_2}, x_{S_3}, x_{S_4}$를 이용하여, 메시지 msg에 대한 서명 $\sigma_{1,m}, \sigma_{2,m}$을 생성.
      $\sigma_{1,m} = x_{S_1} + msg \cdot x_{S_2} \bmod q, \sigma_{2,m} = x_{S_3} + msg \cdot x_{S_4} \bmod q$
   f) 호스트 S는 $(S, C, bid_S,), msg, (\beta_1, \beta_2), (\sigma_{1,m}, \sigma_{2,m})$를 에이전트를 통해 사용자 C에게 전달.

3) 사용자 C는 에이전트로부터 전달받은 메시지들을 검증하므로써 호스트 S가 생성한 메시지에 대한 인증을 수행한다.

   a) $m = hash(S, C, req_C, bid_S)$를 계산 후, $m = msg$ 인지를 체크.
   b) 공개키를 검증하므로써 사용자는 호스트가 위탁키를 정당하게 생성했는지 검증.
      $\beta_1 := (y_C \cdot b_1)^{hash(req\_C, r_{C_1})} \cdot (y'_C \cdot b_2)^{hash(req_C, r_{C_2})} \cdot r_{C_1} \cdot r_{C_2} \bmod p$.
      $\beta_2 := (y_C \cdot b_3)^{hash(req\_C, r_{C_3})} \cdot (y'_C \cdot b_4)^{hash(req_C, r_{C_4})} \cdot r_{C_3} \cdot r_{C_4} \bmod p$.
   c) 서명 검증을 위해 v_1, v_2을 계산.
      $v_1 = \beta_1 \beta_2^{msg} \bmod p, v_2 = \alpha^{s_{1,m}} \beta^{s_{2,m}} \bmod p$
      만일 $v_1 = v_2$이면, 서명을 인정.

프로토콜의 수행 과정에서 보였듯이, 호스트는 모바일 에이전트가 전달한 사용자 생성의 키값 $s_{C_i}, r_{C_i}, i \in \{1,2,3,4\}$ 들이 올바르게 생성된 값인지를 자신의 위탁키를 생성하기 전에 $\alpha^{s_{C_i}} :=$

${y_C}^{hash(req_C, r_{C_i})} \cdot r_{C_i}, i \in \{1,3\}, \beta^{s_{C_j}} := {y'_C}^{hash(req_C, r_{C_j})} \cdot r_{C_j}, j \in \{2,4\}$을 계산하여 검증할 수가 있다. 한편, 사용자는 호스트가 생성한 메시지를 검증하는 단계에서 먼저 $hash(S, C, req_C, bid_S)$를 계산하여 서명된 메시지의 정당성을 검토한다. 그런다음, 위탁키를 올바르게 생성하였는지를 검증하기 위해 $\beta_1 := (y_C \cdot b_1)^{hash(req\_C, r_{C_1})} \cdot (y'_C \cdot b_2)^{hash(req_C, r_{C_2})} \cdot r_{C_1} \cdot r_{C_2} \bmod p$, $\beta_2 := (y_C \cdot b_3)^{hash(req_C, r_{C_3})} \cdot (y'_C \cdot b_4)^{hash(req_C, r_{C_4})} \cdot r_{C_3} \cdot r_{C_4} \bmod p$을 계산해야 한다. 이때, 검증식은 사용자 C의 공개키 y_C, y'_C와 사용자의 요구조건 req_C, 호스트 S의 공개키인 b_1, b_2, b_3, b_4를 이용하므로, 이는 위탁키가 사용자 C가 호스트 S에게 사용자의 대리서명을 위임하였음을 의미한다. 또한 사용자와 호스트 모두 요구조건 req_C에 연계되어 있음을 알 수 있다.

만일 호스트 S가 악의를 품고 또 다른 주문 정보 bid_S'에 대한 사용자의 서명을 만들려 했다면, 호스트는 주문 정보 bid_S'에 대한 입력값 $msg' = hash(S, C, req_C, bid_S')$을 생성할 것이다. 호스트는 사용자가 전달한 키 (s_{C_i}, r_{C_i})을 이용한 위탁키 만을 사용할 수 있으므로, 역시 앞의 프로토콜에서 수행한 위탁키 $x_{S_i}, i \in \{1,2,3,4\}$를 이용하여 서명을 수행해야 한다. 따라서 메시지 msg'에 대한 서명 $\sigma_{1,m'}, \sigma_{2,m'}$을 생성하게 된다.

$$\sigma_{1,m'} = x_{S_1} + msg' \cdot x_{S_2} \bmod q, \sigma_{2,m'} = x_{S_3} + msg' \cdot x_{S_4} \bmod q$$

그러나, 메시지 msg에 대한 서명 $\sigma_{1,m}, \sigma_{2,m}$과 메시지 msg'에 대한 서명 $\sigma_{1,m'}, \sigma_{2,m'}$이 모두 알려지게 되면 누구나 서명에 사용된 비밀키 $x_{S_i}, i \in \{1,2,3,4\}$을 계산할 수 있게 된다. 또한 비밀키 계산을 위한 $s_{C_i}, i \in \{1,2,3,4\}$은 공개된 값이므로, 서명 비밀키를 아는 사람은 누구나 호스트의 장기 비밀키인 $a_i, i \in \{1,2,3,4\}$ 또한 계산할 수 있다. 그러므로, 호스트는 자신의 위탁키로 두 개 이상의 다른 메시지를 서명하지 않을 것이다. 즉, 대리 서명자에 의한 서명의 1회성이 보장되어, 어떠한 악의를 가진 호스트라도 유효한 하나의 주문 정보에 대한 서명만을 생성해 낼 것이다.

제 4 절 제안 기법의 안전성 분석

앞서 살펴본 바와 같이 모바일 에이전트 패러다임에서의 비밀성을 보장한 비밀키의 안전한 계산으로부터 다음과 같은 요구사항을 생각할 수 있다.

- 요구사항
  1) 사용자의 서명이 모바일 에이전트에 의해 정당한 위임을 받은 호스트에 의해서만 수행되어야 한다.
  2) 여러 개의 유효한 주문 정보에 대해 하나의 주문 정보만 인정할 수 있는 우선순위 정책이 요구된다.

1)의 요구사항으로부터 본 연구에서는 모바일 에이전트 패러다임에 대리서명 기법을 적용시킨 예를 제시하였다. 이를 위해 사용자가 모바일 에이전트를 이용하여 대리서명의 위탁키 생성을 위한 사용자의 키를 발급하여 전달하도록 하였다. 에이전트가 전달하는 사용자의 키는 사용자의 비밀키로 만들어진 임시 공개키로서 사용자의 비밀키를 알고있는 사용자만이 생성해 낸 값임을 호스트가 검증할 수 있으며, 사용자의 비밀키를 노출시키지도 않는다. 또한 에이전트에 의해 전달되는 사용자의 키는 요구조건 req_C에 의해 사용자의 비밀키와 연계되어 있어, 호스트는 반드시 이 키를 이용하여 위탁키를 생성하므로써, 위탁키가 사용자와 그 사용자가 요구하는 요구조건으로 생성된 키임을 증명할 수 있게 하였다. 또한 호스트가 생성한 위탁키는 호스트의 장기 공개키를 이용하여 검증 가능하므로 호스트의 비밀키를 알고 있는 호스트 자신만이 생성한 값임을 증명할 수 있다.

1)의 요구사항만을 만족시키는 undetachable 서명 기법이나 본 연구에서 제시한 대리서명 기법의 예는 여러 개의 서버에서 여러 개의 유효한 주문 정보에 대한 서명을 생성할 수도 있고, 하나의 서버에서 여러개의 유효한 주문 정보에 대한 서명을 생성해 내는 것이 가능하므로, 이를 제어하기 위해 2)의 요구사항을 추가적으로 만족시켜야 한다. 그러나 이러한 우선순위 정책에도 불구하고 발생할 수 있는 분쟁을 해결하기 위해서는 에이전트 중재자 혹은 신뢰기관의 참여가 불가피하게 된다. 따라서, 본 연구는 이러한 분쟁을 최소화시킬 수 있는 1회용 대리 서명 기법을 제안하게 되었다. 실패-중단 서명 기법의 1회성의 특성을 이용하여, 하나의 호스트에서는 반드시 하나의 유효한 주문 정보에 대한 서명값만을 생성해 내도록 유도하였다. 이 기법을 이용하면 에이전트는 첫번째 서명을 수행한 호스트로부터 그 수행을 종료하고 사용자에게로 돌아오게 함으로써, 여러 개의 유효한 서명값이 생성되어 발생할 수 있는 분쟁을 사전에 방지할 수 있게 된다.

제안한 기법이 위와 같은 요구사항을 만족시키는 것은 다음과 같은 정리를 통해서 증명가능하다.

정리 1. 에이전트가 호스트에 전달하는 사용자 C가 생성한 $r_{C_i}, s_{C_i}, i \in \{1, 2, 3, 4\}$는 사용자 C만이 대리서명을 위임하기 위해 생성할 수 있다.

증명 사용자 C가 생성한 키에 대한 안전성은 이산대수 문제의 어려움을 기반으로 하고 있다. 따라서, 누구든지 사용자 C의 비밀키를 모르고는 사용자가 생성한 키를 임의로 만들어 낼 수 없다.

정리 2. 호스트 S가 대리서명을 위해 생성한 위탁키 $x_{C_i}, i \in \{1, 2, 3, 4\}$는 호스트 S만이 대리서명을 위해 생성할 수 있으며, 사용자 C로부터 서명을 위임한 위탁키임을 증명할 수 있다.

증명 호스트 S가 생성한 키에 대한 안전성은 이산대수 문제의 어려움을 기반으로 하고 있다. 호스트의 비밀키 $x_{S_1}, x_{S_2}, x_{S_3}, x_{S_4}$는 호스트의 장기 비밀키 a_1, a_2, a_3, a_4에 의해 블라인드 되는 값이므로, 비밀키 a_1, a_2, a_3, a_4를 아는 사람만이 비밀키 값을 계산할 수 있다. 또한, 정리 1에 의해 호스트는 위임키를 위한 사용자의 키를 임의로 생성할 수 없고, 위탁키의 검증을 위해 사용자가 전달한 키로부터 정당하게 위탁키를 생성해야 한다.

정리 3. 1회용 대리 서명 기법을 이용한 호스트 S의 서명은 서명의 1회성을 보장한다.

증명 에이전트가 전달한 사용자의 키와 호스트의 위탁키는 모두 사용자의 요구조건 req_C에 의해 연계된 값이며, 정리 1과 정리 2로부터 호스트의 위탁키는 요구조건에 대해 유일하게 결정된다. 따라서, 앞절에서 제안한 서명에 대해 증명한 바와 같이 주어진 키에 대해 서로 다른 두개 이상의 서명을 만들어낼 수가 없다.

정리 4. 서명 단계에 있어서의 안전성은 실패-중단 서명 기법의 안전성에 기반을 두고 있다.

제 5 절 결론

본 고에서는 전자상거래에 적용된 모바일 에이전트 패러다임에 대한 보안 문제점들을 살펴보았다. 특히 악의를 가진 호스트에서 수행되는 에이전트를 보호하기 위한 연구 중에서, CEF 개념을 기반으로 비밀성을 보장한 비밀키의 계산에 관한 기존의 연구 사례들을 살펴보았고, 대리서명 기법을 적용시켜 보다 효율적인 예를 제시하였다. 또한, 이러한 기법들에 대해 고려해야 하는 안정

성과 분쟁 가능성을 살펴보고, 여러 개의 유효한 서명 발생으로 인해 생길 수 있는 분쟁을 최소화 시킬 수 있는 방법으로 서명의 1회성의 특성을 이용하여 1회용 대리서명 기법을 제안하였다. 추후 과제로서 본 연구에서 제시한 기법에 대한 다양한 공격 모델과 안전성 강화를 위한 추가적인 연구와 증명이 계속적으로 검토가 되어야 하겠다.

본 연구는 모바일 에이전트 패러다임의 안전성을 강화하기 위해 비밀성을 보장한 비밀값의 계산이라는 하나의 요구사항을 만족시키기 위한 방법만을 제안하였으나 다양한 안전성 요구 사항에 부합되는 연구가 지속적으로 요구된다.

참고 문헌

[1] H. Petersen and P. Horster, “Self-certified Keys : Concepts and Applications”, *Proc. 3. Conf. on Communications and Multimedia Security*, 15 pages, 1997.

[2] E.van Heyst, T.Pedersen, “How to make efficient fail stop signatures”, *In Advances in Eurocrypt’92*, LNCS 658, Springer Verlag, pp. 366–377, 1993.

[3] P. Kotzanikolaou, M. Burmester and V. Chrissikopoulos, “Secure Transactions with Mobile Agents in Hostile Environments”, *ACISP 2000*, LNCS 1841, Springer-Verlag, pp.289-297, 2000.

[4] T. Sander and C. F. Tschudin, “Protecting Mobile Agents Against Malicious Hosts”, *Mobile Agent Security*, LNCS 1419, Springer-Verlag, pp.44-60, 1997.

[5] M. Mambo and E. Okamoto, “Proxy Signatures: Delegation of the Power to Sign Messages”, *IEICE Trans. Fundamentals*, Vol. E79-A, No. 9, 11 pages, 1996.

[6] W. Farmer, J. Gutmann and V. Swarup, “Security for Mobile Agents: Authentication and State Appraisal”, *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, LNCS 1146, Springer-Verlag, pp.118-130, 1996.

[7] F. Hohl, “Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts ”, *Mobile Agent Security*, LNCS 1419, Springer-Verlag, pp.92-113, 1998.

[8] P. Kotzanikolaous, G. Katsirelos and V. Chrissikopoulos, “Mobile Agents for Secure Electronic Transactions”, *Recent Advances in Signal Processing and Communications*, World Scientific and Engineering Society Press, pp.363-368, 1999.

[9] S. Loureio and R. Molva, “Privacy for Mobile Code”, *Proc. of Distributed Object Security Workshop OOPSLA’99*, 6 pages, 1999.

[10] J. Merwe and S.H. Solms, “Electronic Commerce with Secure Intelligent Trade Agents”, *Proc. of ICICS’97*, LNCS 1334, Springer-Verlag, pp.452-462, 1997.

[11] G. Vigna, “Cryptographic Traces for Mobile Agents ”, *Mobile Agent Security*, LNCS 1419, Springer-Verlag, pp.137-153, 1998.

[12] Yi Xun, Wang Xiao Feng and Lam Kwok Yan, “A Secure Intelligent Trade Agent System”, *Proc. of the International IFIP/GI Working Conference*, TREC’98, LNCS 1402, Springer-Verlag, pp.218-228, 1998.

[13] D. Coppersmith, J. Stern and S. Vaudenay, “Attacks on the Birational Permutation Signature Schemes”, Proc. of Crypto’93, LNCS 773, Springer-Verlag, pp.435-443, 1993.

[14] S. Kim, “Improved privacy and authenticity in digital signature/key management”, Ph.D. Thesis, SungKyunKwan University, 1998.