산학기술성공학회논문지
Vol. 3, No. 4, pp. 319-324, 2002

# PKI 기반의 보안 다중 에이전트 엔진

장 혜 진*

# A PKI-based Secure Multiagent Engine

**Hai Jin Chang***

**요 약** 전자 상거래와 같이 통신 보안이 요구되는 응용 분야에 에이전트 기술을 효과적으로 적용하려면 에이전트 기술과 보안 기술의 결합이 요구된다. 본 논문은 공개키 기반 구조 기술과 에이전트 기술을 결합하여, 기밀성(privacy), 완전성(integrity), 신원 확인(authentication), 부인 방지(non-repudiation) 등의 보안 기능을 지원하는 다중 보안 에이전트 엔진 모델을 제안한다. 각 에이전트는 구조적으로 에이전트 엔진 계층과 에이전트 응용 계층으로 구성된다. 제안한 보안 에이전트 모델의 설계에는 자가 송수신 메시지, 보안 채널 그리고 에이전트 응용 계층에서의 KQML(Knowledge Query and Manipulation Language) 메시지와 에이전트 엔진 계층에서의 메시지의 구분이라는 개념들이 사용되었다. 제시된 보안 에이전트 엔진 모델이 제공하는 에이전트 언어는 기존 에이전트 언어인 KQML의 내용층 및 메시지층에 대한 아무런 수정이나 제약 조건 없이 통신 계층만을 확장하여 보안 기능을 표현하며, 에이전트간의 보안 통신은 에이전트 언어 상에서 투명하게 표현되는 보안 채널을 통해 이루어진다는 특징을 갖는다.

**Abstract** The Integration of agent technology and security technology is needed to many application areas like electronic commerce. This paper suggests a model of extended multi-agent engine which supports privacy, integrity, authentication and non-repudiation on agent communication. Each agent which is developed with the agent engine is composed of agent engine layer and agent application layer. We describe and use the concepts self-to-self messages, secure communication channel, and distinction of KQML messages in agent application layer and messages in agent engine layer. The suggested agent engine provides an agent communication language which is extended to enable secure communication between agents without any modifications or restrictions to content layer and message layer of KQML. Also, in the model of our multi-agent engine, secure communication is expressed and processed transparently on the agent communication language.

**Key Words :** public key infrastructure, multi-agent, KQML, security, agent communication language

## 1. 서 론

에이전트 기술은 정보 접근, 정보 필터링, 전자 상거래, 워크플로우(workflow) 관리, 생산, 교육, 연예(entertainment) 등의 다양한 분야에 적용될 수 있는 범용 기술이다[1]. 인터넷을 사용하는 다양한 종류의 응용 시스템들이 통신 보안을 요구한다. 전자 상거래와 같은 통신 보안을 요구하는 응용 분야에 에이전트 기술을 효과적으로 적용하려면 통신 보안 기술이 에이전트 기술과 결합되어야 한다. 에이전트 기술과 통신 보안 기술의 결합에 대한 요구 및 연구들이 존재한다[3, 5, 6, 8]. 본 논문은 기존의 에이전트 언어인 KQML(Knowledge Query and Manipulation Language)을 보안적으로 확장한 새로운 에이전트 언어를 지원하는 보안 에이전트 엔진에 대한 모델을 제시한다. 여기서, 에이전트 엔진이란 에이전트를 구성하는 요소들 중 모든 에이전트들이 공통으로 가져야하는 에이전트간 통신 기능, 메시지 처리 기능, 에이전트 언어 기능 등의 집합을 의미한다. 상대적으로 에이전트 프레임웍이란 에이전트 엔진 뿐 아니라 에이전트들의 조정 및 관리를 담당하는 시스템 에이전트들을 포함하는 개념이다.

에이전트 보안에 대한 기존의 연구들에는 이동형 에이전트의 보안에 대한 연구들[5, 6]과 일반 에이전트의 보안에 대한 연구들[3, 8]이 있다. 이동형 에이전트에서는 에이전트의 이동에 따른 적대적 환경에서의 에이전트의 자신의 보호 및 에이전트의 이동을 허락한 호스트의 보호에 관련된 보안 문제들이 중요하다. 반면에 이

*상명대학교 컴퓨터정보통신공학부
Tel: 016-245-5460

동하지 않는 에이전트에서는 에이전트간 통신에 관련된 통신 보안 문제들이 중요하다.

### 1.1 공개키 기반 구조

공개키 기반 구조(public key infrastructure)는 공개키 증명서를 중심으로 암호, 전자 서명, 축약 등의 보안 알고리즘을 통하여 기밀성(privacy), 완전성(integrity), 신원 확인(authentication), 부인 방지(non-repudiation) 등의 보안 기능을 제공한다.

대칭키 암호 알고리즘은 효율적이지만 키 배포의 문제를 갖는다. 비대칭키 암호 알고리즘은 키 배포의 문제를 갖지 않지만 암호화 및 해독 효율이 나쁘다. 따라서 대칭키 암호와 비대칭 키 암호 알고리즘을 함께 사용하는 경우가 많다. 키교환 알고리즘은 통신 쌍방이 안전하지 않은 통신 채널을 통해 서로의 공개키 정보를 교환하여 서로간에 공통의 비밀키(secret key)를 확보하는 알고리즘이다. SSL(Secure Socket Layer)은 응용 프로그램에 투명한 보안 채널(구체적으로는 소켓)을 제공하므로 응용 프로그래머에게 보안에 관련된 자세한 프로그래밍을 요구하지 않으므로 널리 쓰이고 있다. 본 논문의 보안 에이전트 모델은 키교환 알고리즘을 사용하여 공유 비밀키를 교환하며, 비밀키를 이용한 논리적이고 투명한 보안 채널 개념을 사용한다. SSL의 경우 내부적으로 RSA, Diffie-Hellman과 같은 키교환 알고리즘을 사용하고 있다[7].

### 1.2 KQML

KQML[2-4]은 에이전트들간의 지식이나 명령의 교환을 위한 대표적인 에이전트 통신 언어의 하나이다. KQML은 ARPA Knowledge Sharing Effort의 연구 결과의 하나이다. KQML은 여러 가지 장점을 갖고 있으며, FIPA(Foundation For Intelligent Physical Agents) 에이전트 프레임웍[9]과 같은 다른 에이전트 프레임웍들에 영향을 미치고 있다.

KQML 메시지는 통신 계층, 메시지 계층, 내용 계층으로 계층화되어 있으며 그들간에는 독립성이 존재한다. 다음은 간단한 KQML 메시지의 예이다. KQML 메시지의 문법은 짝이 맞는 괄호에 기반을 두고 있다. 여는 괄호로 시작하여 퍼모머티브(performative)가 나오고, 파라미터와 파라미터 값의 쌍들이 나온 후 닫는 괄호로 끝난다.

```
(ask
  :sender Joe      /* 통신 계층 */
  :receiver Mary   /* 통신 계층 */
  :content (PRICE IBM ?price)    /* 내용 계층 */
  :language LPROLOG              /* 메시지 계층 */
  :reply-with ibm-stock )        /* 통신 계층 */
```

**Figure 1.** KQML 메시지의 예.

위 KQML 메시지는 송신 에이전트 Joe가 수신 에이전트 Mary에게 LPROLOG 언어로 작성된 질문(PRICE IBM ?price)를 보내는 메시지이다. 이 예제 메시지의 :content 파라미터의 값은 LPROLOG가 아닌 KIF[10] 등의 다른 내용 언어(content language)로 기술될 수도 있다. 계층 간의 독립은 KQML의 중요한 장점중의 하나이다.

본 논문은 PKI 기반의 통신 보안 기능을 갖는 확장 KQML 및 확장 KQML을 사용하는 에이전트 엔진 모델을 제안한다. 제안된 확장 KQML 언어는 KQML의 내용층이나 메시지층에 대한 아무런 가정이나 제약없이 KQML을 확장하였으며, 자가 수신 메시지, 보안 채널 등의 개념들을 사용하여 통신 보안에 관련된 복잡한 절차들을 추상화하여 확장 KQML 언어에서 투명하게 보안 통신을 제공할 수 있다는 점에서 통신 보안을 지원하는 에이전트 언어 설계의 한 대안을 제시하였다고 할 수 있을 것이다. 본 논문의 구성은 다음과 같다. 2장에서는 에이전트 엔진의 설계 근거 개념 및 설계 목표가 기술된다. 3장에서는 제시한 에이전트 엔진의 구조가 기술된다. 4장에서는 제시한 에이전트 엔진 모델이 지원하는 확장 KQML을 사용하는 보안 통신 모델이 기술된다. 마지막으로 5장은 결론이다.

## 2. 보안 에이전트 엔진의 설계 근거 및 목표

### 2.1 설계 근거 개념

본 논문은 KQML을 사용하는 보안 에이전트 엔진을 설계하기 위하여 자가 수신 메시지, 보안 채널, 에이전트 응용 계층에서의 KQML 메시지와 에이전트 엔진 계층에서의 KQML 메시지의 구분과 같은 개념들을 설계 근거로 사용한다.

자가 수신 메시지란 에이전트 자신에게 보내는 메시지를 의미한다. 예를 들어, 어떤 에이전트 자신의 상태 정보를 스스로 알고자 하는 경우, 자신에게 질의 메시지를 보내 질의에 대한 답을 받는 것이 필요하다. 자신의 상태를 원하는 상태로 설정하고자 하는 경우에도 자가 수신 메시지는 필요하다.

보안 통신 채널이란 두 에이전트간에 설정되는 통신 채널의 한 종류로 채널의 입구와 출구에서 보안에 관련된 처리가 일어나는 통신 채널을 의미한다. 보안 채널은 일단 연결되면 송수신 에이전트 양방간의 통신에 대한 암호, 전자 서명 등의 보안 기능을 추상화하여 투명하게 제공한다. 보안 채널로 입력된 평문 메시지는 보안 채널에 의해 보호된 형태로 변환되어 수신 에이전트

로 전달되며, 보안 채널의 출구에서는 원래의 메시지 M이 복원되어 출력된다. 일반적으로 통신 프로그램 쌍방 간에 보안 통신이 요구될 때, 구현 방법은 두 가지가 있다. 하나는 통신 프로그램의 응용 로직에서 보안 통신을 직접 책임지는 방법이다. 이 경우 통신 메시지의 암호화, 전자 서명 등이 응용 로직을 구현하는 코드에서 직접 이루어진다. 다른 하나는 SSL과 같이 보안 채널 개념을 이용하는 것이다. 보안 채널을 열고 닫는 것은 응용 로직에서 담당하지만 일단 보안 채널이 열린 후에는 평문 형태의 메시지를 보안 채널로 보내면 보안 채널이 보안에 대한 책임을 담당하는 방법이다. 보안 채널을 사용하는 경우, 응용 로직에서의 보안 관련 처리 부담이 적다.

에이전트 응용 계층에서의 메시지와 에이전트 엔진 계층의 메시지의 구분 개념은 에이전트들간 보안 채널의 형성과 같은 복잡하고 물리적인 수준의 핸드쉐이킹을 에이전트 언어 수준에서 투명하고 추상적인 방식으로 지원하기 위한 것이다. 즉, 특히 본 논문이 제시하는 에이전트 모델에서는 응용 수준의 하나의 자가 수신 메시지가 엔진 수준에서는 보안 채널의 구축을 위한 채널 연결 대상 에이전트와의 핸드쉐이킹을 위한 여러번의 메시지 교환을 일으키게 된다. 응용 수준에서는 보안 채널을 형성하라는 자가 수신 메시지와 자가 응답 메시지만으로 보안 채널이 구축된다. 일반적으로, 응용 프로그램 수준(application level)에서 보안 알고리즘들과 핸드쉐이킹 프로토콜들을 구체적으로 처리하는 방법은 응용 프로그램 개발자에게 보안 프로그래밍에 관련된 많은 지식을 요구하며, 구현된 알고리즘과 프로토콜의 재사용성 및 품질 개선이 어렵다는 문제점을 갖는다. 즉, 보안 채널의 구축에 필요한 복잡한 핸드쉐이킹의 세부적 절차를 에이전트 응용 계층에게 가리는 것은 보안 통신 에이전트 프로그래밍의 생산성 측면에서 바람직하다.

### 2.2 설계 목표

본 논문의 보안 에이전트 엔진은 다음과 같은 설계 목표를 갖는다. 이 목표들은 2.1절에서의 근거 개념들을 사용하여 KQML의 근본 개념이나 문법 구조에 영향을 주지 않으면서 보안 채널을 통한 효과적인 보안 통신을 필요로 하는 보안 응용 에이전트의 구현을 쉽게 하기 위한 것이다.

[목표 1] 기밀성, 완전성, 신원 확인, 부인 방지 기능을 갖는 보안 통신을 지원한다.

[목표 2] 보안 통신을 위하여, KQML을 최소한 수정하고 확장한다. 즉, 계층간 독립성을 훼손하지 않으며, KQML의 내용층이나 메시지층에 대한 아무런 가정이나 제약없이 KQML을 확장한다.

[목표 3] 보안 통신에 필요한 암호, 전자 서명 등의 보안 알고리즘들과 핸드쉐이킹 프로토콜 각각을 응용 계층의 에이전트 메시지에 직접 표현하는 방식이 아니라, 보안 채널의 개념을 이용하여 보안 통신에 필요한 구체적인 알고리즘들이나 핸드쉐이킹 절차들을 추상화한다. 즉, 보안에 관련된 기능은 응용 계층의 KQML 메시지에게 투명하며 추상화된 방식으로 제공되도록 한다. 통신 메시지의 기밀성, 완전성 등의 보안 처리는 엔진 내부에서 수행된다.

[목표 4] 에이전트가 공개키 증명서와 개인키 쌍을 갖는 환경을 가정한다. 키배포의 문제와 비효율성의 문제를 극복하기 위하여 대칭키 암호 방식과 비대칭키 암호 방식을 혼합한 세션키 체계를 사용하여 보안 채널을 구현한다.

다음 장의 보안 에이전트 엔진은 본 장에서 기술한 설계 근거 개념들 및 설계 목표를 만족한다.

## 3. 보안 에이전트 엔진의 구조

### 3.1 확장 KQML 언어의 규격

확장 KQML 언어는 보안 기능의 결합을 위하여 두 개의 파라미터만을 추가할 뿐이다. 이런 작은 확장만으로 KQML 언어와 통신 보안을 결합할 수 있는 이유는 확장 KQML 언어가 2.1절에서 기술된 설계 근거 개념들을 사용하기 때문이다. 이들 파라미터들에 대한 보다 상세한 내용은 4장에서 기술된다.

**:private-channel-to 파라미터의 추가**: :private-channel-to 파라미터는 보안 채널을 생성하고, 단절할 때 필요한 정보를 제공하기 위하여 도입되었다. :private-channel-to 파라미터의 값은 채널 연결의 목표 에이전트, 보안 채널 ID, 채널 상태 값의 목록으로 표현된다.

**:through-private-channel 파라미터의 추가**: :through-private-channel 파라미터는 보안 채널이 열린 후 실제로 보안 통신을 수행할 보안 채널을 지정하기 위하여 도입되었다. 파라미터의 값은 두 개로 구성된다. 하나는 보안 채널 ID이고 다른 하나는 전자 서명 여부를 결정하는 값이다. 동일한 송신 에이전트와 수신 에이전트간에 복수개의 보안 채널이 개설될 수 있다.

### 3.2 확장 KQML을 지원하는 보안 에이전트 엔진의 구조

보안 에이전트 엔진을 사용하는 에이전트의 구조는

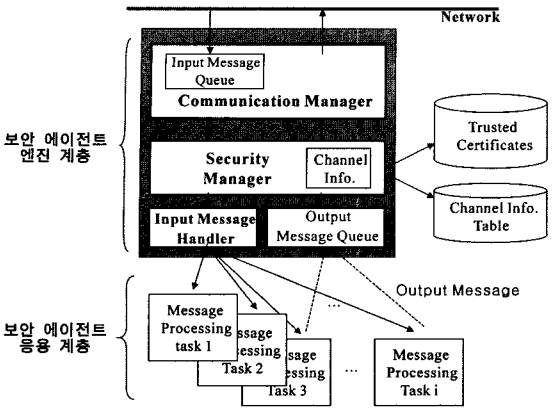


**Figure 2.** 보안 에이전트 엔진의 구조.

그림 2와 같다. 에이전트 엔진은 메시지 큐잉, 메시지 암호화 및 복호화, 개별 송신 및 수신 메시지의 처리를 위한 태스크의 생성 등을 담당한다. 입력 메시지 큐에 입력된 수신 메시지 각각에 대하여 그것을 처리하는 태스크가 생성되며 각 태스크들은 쓰레드로 구현된다. 에이전트의 역할은 수신 메시지들에 대한 태스크들의 프로그래밍에 따라 결정된다.

본 논문이 제안하는 모델에서 에이전트의 응용 계층은 다른 에이전트들 뿐 아니라 자기 자신에게 자가 수신 메시지(self-to-self message)를 보낸 수 있다.

그림 2에서 보안 관리기(security manager)는 보안 채널에 대한 관리 및 메시지의 암호화, 복호화, MAC (Message Authentication Code)의 부착 및 검증, 전자 서명 부착 및 검증 등의 역할을 수행한다. 에이전트간 설정된 보안 채널은 채널 정보 DB에 저장되어 관리되며, 보안 관리기는 설정된 보안 채널에 대한 정보 DB를 메모리상의 내부적 채널 정보 자료 구조에 적재하여 사용한다. 채널 정보 DB에 저장된 채널 정보 레코드의 포맷은 그림 3과 같다.

| 채널 ID | 목표 에이전트ID | 암호화된공유 Secret key | 대칭암호 알고리즘명 | 대칭암호 모드 | 키길이 | 채널 개통 시점 |
|---|---|---|---|---|---|---|

**Figure 3.** 보안 채널 정보 레코드의 구조.

보안 채널 정보 레코드의 채널 ID는 두 에이전트간에 고유한 값을 갖는다. 즉 채널 ID와 목표 에이전트 ID 필드의 합은 보안 채널 정보 레코드의 키에 해당된다. 보안 채널은 두 에이전트간에 키교환을 통해 공유되는 대칭 암호키 값을 사용한다. 그 비밀키 값은 에이전트가 보유한 공개키로 암호화된 형태로 DB에 저장되며, 주메모리로 적재될 때 에이전트의 개인키에 의해 해독된다.

보안 채널의 개설을 요청받는 에이전트가 보안 채널 명을 정한다. 대칭 암호 모드 필드는 ECB(Electronic CodeBook), CBC(Cipher Block Chaining) 등의 암호 모드값을 저장한다. 그림 1의 KQML 메시지에서, Joe 에이전트와 Mary 에이전트간에 ID가 "CH1"인 보안 채널이 설정되어 있고, 그 보안 채널을 통해 Joe가 그림 1과 같은 메시지를 Mary에게 보낸다면, 내용 계층의 값 "PRICE IBM ?price"는 "CH1"과 "Mary"를 키로 사용해서 검색된 보안 채널 레코드의 정보에 따라 해당 암호 알고리즘, 해당 암호 모드, 해당 비밀키로 암호화되고, MAC(Message Authentication Code)을 붙인 결과를 BASE64 등의 인코딩(encoding) 방식으로 변환하여 보내게 된다. 부인 방지를 위하여 전자 서명 필드가 부착될 수도 있다. BASE64는 메시지를 이진 형태가 아닌 형태로 만들기 위해서 사용한다.

보안 처리가 된 메시지의 형태는 에이전트 엔진 내부에서만 이해하면 되므로, 다양한 방식이 적용될 수 있다. 예를 들어, 보안 처리된 :content 파라미터의 값을 "(" + <"PRICE IBM ?price"를 암호화하고 MAC을 붙인 후 BASE64로 인코딩한 문자열> + ")"과 같은 형태로 네트웍으로 보내면 된다. 여기서 +는 문자열 결합 연산이다. 보안 메시지인가 아닌가는 4장에서 기술할 :through-private-channel 파라미터에 의해 구분된다. :language 파라미터의 값은 해독된 메시지 내용값에 대한 값을 갖는다.

### 3.3 보안 채널의 개통 절차

본 논문의 보안 에이전트 엔진에서, 보안 채널의 개통이란 두 에이전트간에 RSA 또는 Diffie-Hellman 알고리즘 등을 사용하여, 두 에이전트간의 안전하지 않은 통신 채널을 통해 안전하게 두 에이전트간 공유 비밀을 생성하는 것을 의미한다. 그 공유 비밀은 두 에이전트간 보안 통신을 위한 비밀키로 사용된다. 보안 처리된 메시지의 :content 파라미터의 값은 비밀키로 암호화되며, MAC 등의 메시지 인증 코드를 갖도록 하여, 기밀성, 완전성, 신원 확인과 같은 보안성이 보장될 수 있다. 수신자와 송신자만이 공유 세션키를 알기 때문이다. 부인 방지가 필요한 경우 보안 처리된 메시지에 전자 서명을 붙임으로서 가능해진다. 다음 장에는 보안 채널의 개통을 위해 구체적인 KQML 메시지를 사용하는 방법이 기술된다.

## 4. 확장 KQML을 사용하는 보안 통신 모델

### 4.1 보안 채널 개통

본 논문의 확장 KQML에서 송신 에이전트가 수신

에이전트에게 보안 채널의 구축을 요청하는 메시지는 다음과 같다. :private-channel-to 파라미터의 첫 번째 요소는 보안 채널을 열고자 하는 대상 에이전트 ID이다. 두 번째 요소의 값은 송신 에이전트와 수신 에이전트간의 기존 보안 채널 ID이거나 null 이다. null의 경우 새로운 보안 채널의 구축을 요청한다는 의미이다. 새로운 보안 채널의 ID는 수신 에이전트 쪽에서 정하기 때문이다. 세 번째 요소의 값은 opened 또는 null 이며, 보안 채널의 개통을 요청 메시지에는 opened가 사용된다.

```
(achieve
   :sender <sender>
   :receiver <sender>
   :private-channel-to (<Target Agent ID>
    <보안 채널ID 또는 null> opened)
   :reply-with <reply tag>)
```

보안 채널 개통 요청은 자가 수신 메시지의 형태를 갖는다. 즉 동일 에이전트의 에이전트 응용 계층에서 에이전트 엔진 계층에게 보안 채널을 열라는 명령을 보내는 것이다. 보안 채널 개통 메시지를 스스로 보내고 받은 에이전트 엔진은 보안 채널 개통 대상 에이전트와 보안 채널을 열기 위한 핸드쉐이킹을 수행한다. 이 핸드쉐이킹은 에이전트의 보안 관리기와 통신 관리기 모듈에서 내부적으로 수행된다. 여기서 2.1절에서 기술된 응용 계층의 메시지와 엔진 계층의 메시지의 구분 개념이 사용되고 있다. 보안 채널의 개통에 합의가 되는 경우, 송신 에이전트의 에이전트 엔진은 다음과 같은 응답 메시지를 생성하여 스스로에게 보낸다.

```
(reply
   :sender <sender>
   :receiver <sender>
   :private-channel-to (<Target Agent ID>
    <보안 채널 ID> opened)
   :in-reply-with <reply flag>)
```

새로운 보안 채널 열기에 성공한 경우, 에이전트 엔진은 새로운 보안 채널 정보에 해당하는 보안 채널 정보 레코드를 생성하게 된다. 보안 채널의 구축에 성공하지 못한 경우 다음과 같은 응답 메시지를 스스로 받게 된다. 새로운 보안 채널의 열기에 실패한 경우에는 채널 ID가 null 값이 된다.

```
(reply
   :sender <sender>
   :receiver <sender>
   :private-channel-to (<Target Agent ID>
    <보안 채널ID 또는 null>  null)
   :in-reply-with <reply flag>)
```

### 4.2 보안 채널의 폐기

두 에이전트간 구축된 보안 채널은 필요에 따라 폐기되어야 한다. 보안 채널에 대응하는 비밀키가 누출되거나 너무 오래 사용하여 보안적으로 염려가 되는 경우 등에 보안 채널을 닫아야 할 필요가 있다. 보안 채널을 폐기하기 위한 메시지도 :private-channel-to 파라미터를 사용한다. 파라미터 값의 첫 번째 요소는 대상 에이전트이고, 두 번째 요소는 보안 채널 ID 이다. 첫 번째와 두 번째의 요소를 합치면 보안 채널 정보 레코드들을 구분하기 위한 키가 형성된다. 세 번째 요소는 null 값을 사용하여 채널을 닫겠다는 요청임을 표현한다.

```
(achieve
   :sender <sender>
   :receiver <sender>
   :private-channel-to (<Target Agent ID>
    <기존 보안 채널 ID> null)
   :reply-with <reply tag>)
```

보안 채널 폐기에 성공하는 경우, 양방 에이전트는 다음과 같은 응답 메시지를 수신하게 된다. 이 메시지가 보안 채널 폐기에 대한 응답 메시지라는 사실은 :in-reply-with 파라미터의 값을 통해 알 수 있다.

```
(reply
   :sender <self>
   :receiver <self>
   :private-channel-to (<Target Agent ID>
    <기존 보안 채널 ID> null)
   :in-reply-with <reply flag>)
```

보안 채널 폐기에 실패하는 경우 폐기 요청 에이전트는 다음과 같은 메시지를 수신하게 된다.

```
(reply
   :sender <sender>
   :receiver <sender>
   :private-channel-to (<Target Agent ID>
    <기존 보안 채널 ID> opened)
   :in-reply-with <reply flag>)
```

### 4.3 보안 채널을 사용한 보안 통신

보안 채널의 구축에 합의한 쌍방은 :through-private-channel 파라미터를 사용하는 메시지를 통해 보안 통신을 할 수 있다. 이 메시지의 <sender ID> 또는 <receiver ID> 값과 <channel ID> 값을 결합하면 하나의 보안 채널 정보 레코드를 검색할 수 있다. :through-private-channel 파라미터 값에 <signed 또는 null>이 필요한 이유는 다음과 같다: 두 에이전트간 보안 채널의 근본은 두 에이전트간 공유하는 비밀키이다. 이 비밀키는

보안성이 없는 망을 사용하는 경우에도 안전한 키합의 방식을 사용하여 생성되므로 두 에이전트간에는 통신의 기밀성이 보장된다. 또한 보안 메시지는 에이전트 엔진 내부에서 MAC을 붙이므로 완전성과 신원 확인이 이루어진다. 하지만 보안 채널을 구성하는 비밀키는 통신 양방이 알고 있으므로 부인 방지에 취약하다. 따라서 :through-private-channel 파라미터의 값의 하나로 부인 방지가 팔요한 경우 전자 서명 여부에 대한 값을 사용한다.

```
(<performative>
   :sender <sender ID>
   :receiver <receiver ID>
   :content <content>
   :through-private-channel (<channel ID>
    <signed 또는 null>) ... )
```

보안 채널을 사용하는 KQML 메시지는 <performative>의 종류나 컨텐츠 언어의 종류, 통신 계층의 파라미터들의 사용에 제약이 없다. 단 스스로에게 보안 채널을 생성하는 경우를 고려하지 않는다면 :sender와 :receiver 파라미터의 값들은 같을 수 없다.

## 5. 평가 및 결론

본 논문은 에이전트 통신 언어로 KQML을 사용하는 다중 에이전트 엔진에 공개키 기반 보안 기술을 결합하여 기밀성, 완전성, 신원 확인, 부인 거부 등의 보안 기능을 제공하는 다중 보안 에이전트 엔진 및 에이전트 언어 모델을 제안하였다. 제안된 에이전트 엔진의 설계 근거로 사용된 자가 수신 메시지, 보안 채널, 응용 계층 메시지와 엔진 계층 메시지의 구분에 대한 개념들을 기술하였다. 본 논문이 제안하는 에이전트 엔진의 에이전트 통신 언어는 내용층 및 메시지층에 대한 아무런 수정이나 제약 조건 없이 KQML 언어의 통신 계층만을 확장하여 에이전트 언어 수준에서의 보안 기능들을 제공하며 에이전트 응용 계층에게 투명하고 효율적인 보안 채널의 개념을 지원한다.

## 감사의 글

이 연구는 상명대학교 교내학술연구비 지원을 받음.

## 참고문헌

[1] Michael N. Huhns and Munindar P. Singh, "Agents and Multiagent Systems: Themes, Approaches, and Challenges", *Readings in Agents*, Morgan Kaufmann, 1998.

[2] Tim Finin, Don McKay, Rich Fritzson, and Robin McEntire, "KQML: An Information and Knowledge Exchange Protocol", Knowledge Building and Knowledge Sharing , Ohmsha and IOS Press, 1994.

[3] Chellah Thirunavukkarasu, Tim Finin, and James Mayfield, "Secret Agents - A Security Architecture for the KQML Agent Communication Language", CIKM '95 Intelligent Information Agent Workshop, Baltimore, December 1995.

[4] Tim Finin, Yannis Labrou, and James Mayfield, "KQML as an agent communication language", Software Agents", MIT Press, Cambridge, 1977.

[5] David M. Chess, "Security Issues in Mobile Code Systems", LNCS 1419, 1998.

[6] Tomas Sander and Christian F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts", LNCS 1419, p. 44, 1998.

[7] Jess Garms and Daniel Somerfield, Professional Java Security, chaper 9. Wrox, 2001.

[8] FIPA, FIPA Security SIG Request For Infomation, http://www.fipa.org, 2000

[9] FIPA, FIPA Specifications Published in 2000, http://www.fipa.org/repository/ fipa2000.html, 2000.

[10] Michael R. genesereth, "Knowledge Interchange Format Specification", http://logic.stanford.edu/kif/specification.html.