유비쿼터스 환경에서 안전한 컨텍스트 전송 프로토콜 구현

임현숙*, 이영록*, 이형효**, 노봉남*

*전남 대학교, 정보보호협동과정

Implementation of Secure Context Transport Protocol in Ubiquitous Environment

Hyun-sook Im*, YoungRok Lee*, HyungHyo Lee**, BongNam Noh*

*Interdisciplinary Program of Information Security, Chonnam National Univ.

**Div. of Information and EC, Wonkwang Univ.

요 약

유비쿼터스 환경을 현실 세계에 구현함에 있어 가장 중요한 핵심 기술 중 하나는 컨텍스트 정보 관리이다. 하지만 기존의 많은 연구들에서 컨텍스트를 요구함에 있어 무엇을 어떻게 요구할 것인지에 대한 방법이 명확하지 않았다. 따라서 본 논문에서는 응용이 필요로 하는 컨텍스트 정보만을 컨텍스트 매니저에게 주고받는 컨텍스트 전송 프로토콜을 소개한다. 그리고 컨텍스트 요청 메시지와 응답 메시지가 어떻게 XML로 구성되고 동작하는지 보인다. 또한, 개인 정보에 관한 컨텍스트를 안전하게 처리하기 위해 정당한 사용자만이 컨텍스트 정보를 요청하고 받을 수 있도록 SPKI/SDSI(Simple Public Key Infrastructure/Simple Distributed Security Infrastructure)를 이용한 인증절차를 소개한다.

I. 서론

유비쿼터스 컴퓨팅 환경에서 말하는 컨텍스트란 사용자와 유비쿼터스 컴퓨팅 환경 사이의 상호 작용과 이와 관련된 주변상황, 객체, 사용자를 둘러싼 조건에 관한 정보 등을 의미한다. 이러한 컨텍스트 정보 관리는 유비쿼터스 환경을 현실 세계에 구현함에 있어 가장 중요한 핵심 기술 중 하나이다[1].

응용(Application)은 사용자에게 적절한 서비스를 제공하기 위해 다양한 환경적 정보를 필요로 한다. 그러나 많은 환경적 정보를 컨텍스트 관리자에게 요구함에 있어서 무엇을 어떻게 요구할 것인지에 대한 방법이 명확하게 제시되고 있지 않다. 또한 컨텍스트 관리자가 응용에게 요구한 컨텍스트 정보를 전달하는 과정에서 여러 연구들이 너무 많은 정보를 포함하여 보내고 있다[2][3][4]. 따라서 유비쿼터스 환경에서는 컨텍스트 관리에 대한 적절한 모델이 필요하다.

본 논문에서는 컨텍스트 정보를 처리하고 관리하는 컨텍스트 관리자(Context Manager, CM)를 구현하고 CM과 응용 사이의 컨텍스트 전송에 관한 프로토콜을 제안한다. 이 프로토콜은 응용이 필요로 하는 컨텍스트 요구사항을 표현한 명세서를 다양하고 체계적으로 사용 가능하도록 하기 위한 컨텍스트 전송 프로토콜이다.

유비쿼터스 환경에서는 사용자의 위치 변화에 관계없이 네트워크상의 자원에 대한 용이한 접근이 가능해야 한다. 또한 특정 도메인 안에 새로운 응용이 나타나기도 하고 기존의 응용이 사라지기도 하는데 이러한 응용이 도메인에서 쉽게 적응할 수 있어야 한다. 따라서 제안된 도메인은 지니 기반으로 설계하고 구현한다.

센서와 상황 인지 모델의 적용에 따라 개인의 위치 정보가 쉽게 유출될 수 있으며, 사용자 자동 지원 시스템이 증가할수록 개인 신상 정보의 노출 수준도 심각하게 될 것이다. 그래서 유비쿼터스 컴퓨팅 환경 구축에서 센서와 상황 인지 응용 모델로부터 생성되는 컨텍스트의 정보보호는 유비쿼터스 보안의 핵심 사안 중 하나로 떠오를 것이며, 컨텍스트에 대한 정보 관리와 보호는 필요한 기술이다[1]. 본 논문에서는 이점을 고려하여 SPKI/SDSI(Simple Public Key Infrastructure / Simple Distributed Security Infrastructure) 인증서를 이용하여 인증과 인가 기능을 제공한다.

* 본 연구는 정보통신부 대학 IT 연구센터 육성, 지원사업의 연구결과로 수행되었습니다.

본 논문의 2장에서는 컨텍스트의 정의와 관련된 연구인 지니(JINI), SPKI/SDSI 인증서에 관해 기술한다. 3장에서는 구현된 컨텍스트 처리 프로토콜의 동작 순서와 구성요소에 대해 기술하고 마지막으로 결론을 기술한다.

II. 관련연구

1. Context

Dey의 정의에 따르면 컨텍스트는 사용자와 다른 사용자, 시스템 또는 디바이스들 간의 상호 작용에 영향을 미치는 실체들의 상황을 규정하는 정보를 의미한다. 즉 컨텍스트란 사람, 객체, 장소 등과 같은 실체의 상황을 특징화 하는데 사용되는 정보를 말한다[5][6]. 이런 컨텍스트는 시스템으로 하여금 다양한 센서와 응용을 통해 수집되고 인식되어 사용되거나 다른 컨텍스트들 간의 연관관계를 통해 새로운 추론을 내리는데 사용된다. 이러한 정의는 응용 개발자가 주어진 서비스 시나리오에 따른 컨텍스트 전개작업을 쉽게 할 수 있도록 한다[1].

2. JINI

JINI는 네트워크상의 모든 종류의 디바이스와 소프트웨어 자원의 통합체를 구성하여 서비스와 자원을 공유하고 사용자의 위치 변화에 관계없이 네트워크상의 자원에 대한 용이한 접근 및 네트워크의 개설, 갱신, 변경 작업의 단순화를 목표로 하는 기술로서 1998년 썬 마이크로시스템즈사에서 발표한 분산 환경의 홈 네트워크 자원 공유 플랫폼이다. JINI 기술은 현재 정의되고 있는 대부분의 미들웨어 기술의 근간을 이루고 있으며, 특히 OSGi의 기술적 기반을 제공한다[7]. 그러나 JVM(Java Virtual Machine)과 RMI(Remote Method Invoke)를 기본적으로 포함해야 하는 부담은 있다.

지니 시스템은 클라이언트(client), 서비스 제공자(service provider), 룩업 서비스(lookup service)로 구성되며 이들 삼자간의 상호 통신으로 이루어진다. 서비스 제공자는 로컬 네트워크상에서 멀티캐스트 요청을 통해 룩업 서비스를 발견하여 서비스 객체(service object) 및 서비스 속성(service attributes)을 룩업 서비스에 등록한다. 클라이언트가 JAVA 유형이나 서비스의 속성 등에 따라 서비스를 요청하면 서비스 객체의 사본이 클라이언트로 이동한다. 이후, 클라이언트와 서비스 제공자 간의 통신이 이루어진다[1].

3. SPKI/SDSI 인증서

SPKI/SDSI는 기존의 PKI 기반의 융통성 없고 복잡한 문제를 단순화하기 위해 연구되었다. SDSI의 특징 중 두드러진 것은 "지역이름공간(local name space)"이라는 개념을 도입해서 공개키의 소유자는 각각 그 공개키에 기반을 둔 지역이름공간을 생성할 수 있다는 것이다. 다시 말하면 누구든지 자신이 보유한 주체(principal)들의 공개키에 자신의 공개키에 기반을 둔 지역이름을 연결하는 이름 인증서를 발행할 수 있다. 또한 이들 지역이름을 이용하여 새로운 이름 인증서를 발행할 수 있다.

SPKI의 특징은 이름 인증서와 권한 인증서를 각각 구별하여 명세할 수 있는 융통성을 제공한다[8]. 예를 들면 한 사용자의 권한은 수시로 바뀔 수가 있는 관계로 기존 PKI 인증서에 권한을 부여하는 통합 방식인 X.509 인증서는 한 사용자의 권한이 바뀔 때마다 매번 그 사용자에게 새로운 인증서를 발행해야 하는 불편함이 있다. 또한 SPKI는 주체의 구별을 이름이 아닌 공개키로 함으로써 주체의 신상정보에 따라 수시로 변경될 수 있는 주체의 전역이름을 그 주체의 구별 대상으로 하는 X.509 인증서의 문제점을 해결한다[9]. 필요한 권한 인증서들을 인증서 캐시에서 찾아주는 "Certificate chain discovery"라는 알고리즘[10]은 서버가 SPKI/SDSI로 보호된 정보를 클라이언트가 요청 할 때 클라이언트는 그 정보를 취득할 권한이 있다는 것을 서버에게 증명해보이기 위해 사용한다. 즉, 클라이언트가 자신이 보유하고 있는 인증서 캐시에서 서버에 증명할 권한 인증서와 관련된 이름 인증서들을 찾아내는 알고리즘이다.

III. 프로토콜 설계

컨텍스트 관리자에게 요구함에 있어서 무엇을 어떻게 요구할 것인지에 대한 방법이 명확하게 제시되지 못하는 기존의 문제와 컨텍스트 관리자가 응용에게 요구한 컨텍스트 정보를 전달하는 과정에서 많은 정보를 포함하여 보내고 있는 문제를 해결하기 위해 프로토콜을 제안한다.

1. context 전송 프로토콜 구현

본 논문에서는 응용과 CM 간에 메시지를 주고받을 수 있는 프로토콜을 정의한다. 응용과 CM은 JINI 조회 서비스를 이용해 통신을 시작한다. XML로 컨텍스트 전송 프로토콜을 위한 CPML(Context Processing Markup Language)를 정의하였다.

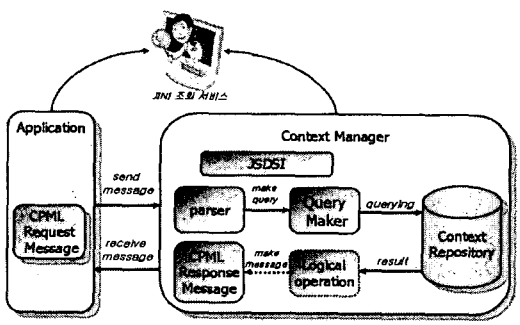

그림 1: 컨텍스트 처리 프로토콜 동작.

응용은 필요한 컨텍스트 정보를 CPML로 작성한다. 작성된 요청메시지를 CM에게 전송한다. CM은 그것을 받아 파싱하고 질의문을 자동 생성하여 컨텍스트 저장소에서 컨텍스트를 얻는다. 이 컨텍스트 저장소에는 여러 센서들로부터 수신된 컨텍스트들이 저장되어 있다. CM은 컨텍스트 저장소에서 얻은 컨

텍스트로 필요한 연산을 하고 다시 CPML로 표현된 응답 메시지를 만들어 응용에게 전송한다. 이를 받은 응용은 결과 값에 따라 응용이 해야 할 일을 결정한다. 그림 1은 컨텍스트 전송 프로토콜의 전체적인 동작 구조이다.

1) Request message

유비쿼터스 환경에서는 다양한 응용이 존재하고 각각의 응용은 컨텍스트 요구 사항도 다양하다. 이런 다양한 요구를 처리하기 위하여 요청 메시지는 단순히 컨텍스트 정보만을 요청하는 연산뿐만 아니라 관계연산, 논리연산을 수행할 수 있도록 설계한다. 따라서 응용이 원하는 컨텍스트의 조건들을 조합할 수 있다. 예를 들면 '온도가 몇 도인가?'를 질의하는 것도 가능하고, '온도가 10도 이상이고 습도가 30%이상인가?'를 조합하여 질의하는 것도 가능하다. CM은 첫 번째 질문에는 '19도'라는 결과 값을 반환해 주고 두 번째 질의에는 TRUE나 FALSE로 결과 값을 반환해 준다.

요청 메시지는 다음과 같은 구조를 가지고 있다.

- <Requester>: 컨텍스트 정보를 요청하는 응용에 대한 정보
- <ContextInfo>: 컨텍스트에 관련된 정보로 컨텍스트 위치와 종류를 의미
- <conditionValue>: TRUE/FALSE를 판단하기 위한 조건 (예를 들어 "온도>19"의 19)
- <ContextCondition>: 속성 값으로 관계연산자를 입력받아 관계연산 수행하여 결과 값으로 TRUE/FALSE 반환하여 논리 연산에 사용할 수 있게 함
- <ContextConditionSet>: 두 개 이상의 <ContextCondition>이 있는 경우 이들을 묶어 AND나 OR의 논리연산을 수행할 때 사용
- <RequestContexts>: 논리연산, 관계연산 없이 저장된 단일 컨텍스트 정보만을 요청

2) Response message

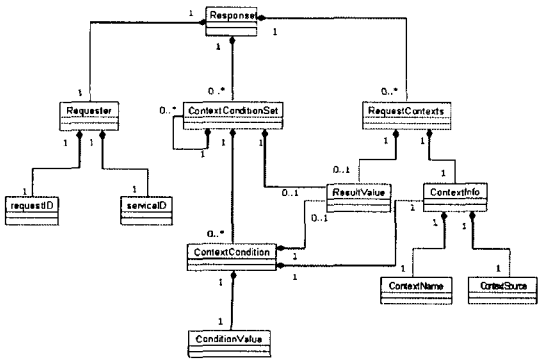

그림 2: Response Message 구조.

그림2는 Response Message 구조이다. 이 구조는 Request Message와 유사하며 차이점은 루트 엘리먼트가 "Request"에서 "Response"로 변경된 것과 <ContextConditionSet>, <ContextCondision>, <ReauestContext>에 하위엘리먼트로 <ResultValue>가 생긴 것이다.

- <ResultValue>: 요청 메시지에서 요청한 context의 조건에 따라 TRUE/FALSE 또는 데이터베이스에 저장된 컨텍스트 값으로 속성으로 단위와 데이터타입을 가질 수 있다.

3) Request Maker

요청 메시지는 스키마를 모두 알고 있어야 작성이 가능하기 때문에 사용자가 어려움을 느낄 수 있다. 따라서 요청 메시지를 자동으로 만들어주는 툴인 그림 3과 같은 Request Maker를 구현한다.

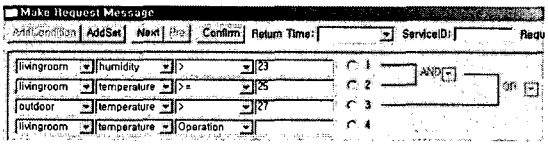

그림 3: Request Maker.

그림 4는 그림 3에서 작성된 요청서이다. 이 요청서가 의미하는 것은 '거실의 습도가 23도를 초과하고 온도가 25도 이상이거나 외부의 온도가 27도를 초과하였는가, 거실의 온도는 몇 도인가?'이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Requester>
    <serviceID>make_Request</serviceID>
    <requestID>any_applicaion</requestID>
  </Requester>
  <RequestContexts>
    <ContextInfo>
      <ContextName>temperature</ContextName>
      <ContextSource>livingroom</ContextSource>
    </ContextInfo>
  </RequestContexts>
  <ContextConditionSet logicalOp="OR">
    <ContextCondition relationalOp="&gt;">
      <ContextInfo>
        <ContextName>temperature</ContextName>
        <ContextSource>outdoor</ContextSource>
      </ContextInfo>
      <ConditionValue>27</ConditionValue>
    </ContextCondition>
    <ContextConditionSet logicalOp="AND">
      <ContextCondition relationalOp="&gt;">
        <ContextInfo>
          <ContextName>humidity</ContextName>
          <ContextSource>livingroom</ContextSource>
        </ContextInfo>
        <ConditionValue>23</ConditionValue>
      </ContextCondition>
      <ContextCondition relationalOp="&gt;=">
        <ContextInfo>
          <ContextName>temperature</ContextName>
          <ContextSource>livingroom</ContextSource>
        </ContextInfo>
        <ConditionValue>25</ConditionValue>
      </ContextCondition>
    </ContextConditionSet>
  </ContextConditionSet>
</Response>
```

그림 4: Request Message의 예.

4) Context Repository

응용이 요청할 때 컨텍스트를 제공하기 위해 컨텍스트 관리자는 현재 상태의 컨텍스트 정보를 수집하고 저장하는 작업을 수행한다. 저장소에는 컨텍스트에 대한 스키마가 정의되어 있어 데이터베이스 형태를 갖추고 있다.

2. 안전한 메시지 전송 절차

본 논문에서 구현된 컨텍스트 처리 프로토콜에서는 보안을 고려한다. 유비쿼터스 컴퓨팅 환경에서 송수신 되는 메시지에는 사용자의 컨텍스트 정보 등의 개인 프라이버시와 기타 비밀스러운 내용이 담겨져 있을 수 있다. 이러한 보안상의 문제 때문에 메시지는 인증되어지고 권한이 있는 정당한 사용자에게만 전달되어야 한다. 하지만 에드혹과 같은 유비쿼터스 환경에서는 온라인 서버가 없을 수도 있으므로, 기존에 네트워크 환경에서 적용하던 인증과 접근제어 방법을 그대로 적용할 수는 없다. 따라서 SPKI/SDSI를 이용하여 인증 및 인가를 한다.

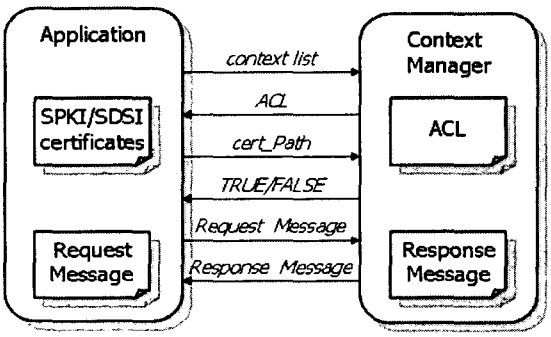

그림 5: SPKI/SDSI를 이용한 인증/인가.

응용은 요청 메시지를 전송하기 전에 사용할 컨텍스트의 목록(context list)을 CM에게 보낸다. CM은 접근 제어 목록(ACL)을 응용에게 보낸다. 응용은 접근 제어 목록과 자신의 인증서들을 통해 컨텍스트에 대한 권한이 있음을 증명하고, 증명된 결과(cert_Path)를 CM에게 보낸다. CM은 응용이 보내온 결과(True/False)를 검증하고 오류가 없으면 응용을 등록한다. 등록된 응용은 이후에 요청 메시지를 보내고 결과 메시지를 받을 수 있다. 그림 5는 설명한 인증 및 인가절차를 나타낸 것이다.

IV. 결론

본 논문에서는 컨텍스트 관리자를 구현하고 컨텍스트 관리자와 응용 간에 전송 프로토콜을 정의하였다. 컨텍스트 관리자에게 무엇을 어떻게 요구할 것인지에 대한 방법이 명확하게 제시되지 못하는 기존의 문제는 응용이 요청하는 컨텍스트에 대한 명확한 요청서를 새로 정의된 CPML로 만들어 전송하도록 하여 해결하였다. 또한 이 프로토콜은 다양한 응용들의 컨텍스트 요청과정에서 발생할 수 있는 개인정보 노출의 문제를 해결하기 위해 SPKI/SDSI를 사용하여 인증과 인가를 하도록 제안하였다.

[참고문헌]

[1] 채명훈, "유비쿼터스 환경의 지니 기반 컨텍스트 보안 관리 시스템 설계 및 구현", 전남대학교 정보보호 협동과정 석사학위 논문, 2005년

[2] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganat, Roy H. Campbell, Klara Nahrstedt, "Gaia: A middleware Infrastructure to Enable Active Spaces", In IEEE Pervasive Computing, Oct-Dec 2002

[3] Harry Chen, Tim finin, Anupam Joshi, "An Ontology for Context-Aware Pervasive Computing Environments", The Knowkedge Engineering Review, Vol. 18, No.03 , pp.197-207, September 2003.

[4] A.Rakotonirainy, J.Indulska, S.W Loke, A. Zaslavsky, "Middleware for Rezctive Components: An Integrated Use of Context, Roles, and Event Based Coordination", Preceeding of the IFIP/ACM International Conference, pp.77-98, 2001.

[5] Anind K. Dey, Gregory Abowd, "Towards a Better Understanding of Context and Context-Awareness", Workshop on the What, who, where, when and how of context-awareness at CHI 2000, April 2000.

[6] Anind K. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing Journal, Vol. 5, no. 1, pp.4-7, February 2001.

[7] 박준희, 손영성, "홈네트워크 미들웨어 및 표준화 동향", ETRI, 전자통신동향분석 19권 5호, October 2004.

[8] Carl M. Ellison, Bill. Frantz, Butler. Lampson, Ron Rivest, Brian M. Thomas, Tatu Ylonen, "SPKI Certificate Theory" RFC2693, September 1999.

[9] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, Tatu Ylonen, "SPKI Examples," Internet-Draft, March 1998.

[10] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest, "Certificate Chain Discovery in SPKI/SDSI," Journal of Computer Security, volume 9, Issue 4, pp. 285-322, December 2000.