# SAML을 이용한 그리드와 웹 서비스 보안을 위한 자바 기반 Single Sign-On 라이브러리의 설계 및 구현

정 종 일[†] · 유 석 환[††] · 신 동 규[†††] · 신 동 일[††††] · 차 무 홍[†††††]


## 요 약

최근 그리드의 개발 초점은 자원에서 서비스로 이동되고 있다. 그리드 서비스는 잘 정의된 인터페이스들의 집합을 제공하고 특정 규약들을 준수하는 웹 서비스로 정의된다. SAML은 이질적인 개체들 간에 인증, 인가 그리고 프로파일 정보를 교환하는 것을 가능하게 하는 웹 서비스를 위한 XML기반의 단일인증 표준으로서 분산 환경에서 이질적인 보안 서비스들 간에 상호운용성을 제공한다. 본 논문에서 구현한 XML기반의 단일인증 구현을 위한 API는 기존의 단일인증 기술들을 통합할 수 있는 방법을 제시하는 SAML명세를 구현하여 비 XML기반의 인증기술들과의 상호운용성을 제공함으로써 웹 서비스 표준을 채택한 그리드 아키텍처에서 이질적인 서비스들의 접근 용이성을 제시한다.


**키워드 : Single Sign-On, SAML, 전자상거래 보안**

# Design and Implementation of a Java-Based Single Sign-On Library Supporting SAML (Security Assertion Markup Language) for Grid and Web Services Security

Jongil Jeong[†] · Seokhwan Yu[††] · Dongkyoo Shin[†††] · Dongil Shin[††††] · Moohong Cha[†††††]


## ABSTRACT

In recent years, the Grid development focus is transitioning from resources to services, A Grid Service is defined as a Web Service that provides a set of well-defined interfaces and follows specific conventions. SAML as a standard for Web Services which enables exchange of authentication, authorization, and profile information between different entities provides interoperability among different security services in distributed environments. In this paper, we implemented SAML API. By offering interoperability for non XML-based authentication technologies using SAML specification offering a method to integrate the existing Single Sign-On technologies, the API provides convenience for accessing different services in Grid architecture.


**Key Words : Single Sign-On, SAML, E-commerce Security**

## 1. 서 론

그리드(Grid) 컴퓨팅은 진보된 과학과 공학을 위한 분산 컴퓨팅 기반을 의미하며 자원공유를 지원하고 동적이고 지리적으로 분산된 조직들을 연결하는데 반드시 해결해야 할 문제들에 대한 해결책을 지원한다. 최근 그리드 개발의 초점은 자원에서 서비스로 이동하고 있다. 그리드 서비스는 잘 정의된 인터페이스의 집합을 제공하고 특정 규약들을 따르는 웹 서비스로 정의된다[1]. 웹 서비스와 그리드 같은 분산 환경에서 자원의 공유는 파일 교환뿐 아니라 컴퓨터, 소프트웨어, 데이터 그리고 다른 자원들에 대한 직접적인 접근과도 관련된다[2]. 따라서 이질적인 시스템 간에 공유된 다수의 자원에 접근하려는 사용자의 접근편의성과 보안 취약성을 개선하기 위한 연구가 필요하다. 단일인증(Single Sign-On)은 사용자가 그리드[3]나 웹 서비스 같은 분산 시스템에서 제공하는 다수의 서비스에 로그인 하는 것을 허용하는 보안 특징을 갖는다. 이 과정에서 사용자는 단 한 번의 인증만을 거치거나 적어도 항상 같은 방법으로 인증과정을 수행하는 것이 필요하다[4, 5]. 단일인증을 구현하기 위해 커버로스, 패스워드-스토어 및 공개키 기반(Public Key Infrastructure : PKI) 같은 기술에 의존하는 다양한 단일인증 기술들이 제안되었으나 제안된 기술들은 클라이언트 측에 부가적인 기반이 구축되어야하고 새로운 관리 단계를 필요


† 준 회 원 : 세종대학교 대학원 컴퓨터공학과 박사과정
†† 준 회 원 : 세종대학교 대학원 컴퓨터공학과 석사과정
††† 종신회원 : 세종대학교 컴퓨터공학과 부교수
†††† 종신회원 : 세종대학교 컴퓨터공학과 부교수
††††† 준 회 원 : 세종대학교 컴퓨터공학과 석사과정

논문접수 : 2004년 5월 28일, 심사완료 : 2005년 5월 2일

로 한다[6]. 무엇보다 이러한 기술들은 사용자 인증과 관련하여 비 XML형태의 메시지를 생성하고 교환하기 때문에 해당 기술을 구현하지 않는 시스템과의 상호운용은 불가능하다. 따라서 다양한 서비스들을 공유하는 그리드 환경에서 사용자의 단일인증을 구현하기 위해서는 각 구현기술들로부터 생성된 사용자의 인증정보를 기술하기 위한 일관되고 표준적인 방법이 필요하다.

최근 OASIS(Organization for the Advancement of Structured Information Standards)는 XML기반에서 보안관련 정보의 안전한 교환을 위한 프레임워크인 SAML(Security Assertion Markup Language)[7, 8]을 표준으로 승인하였다. SAML은 그리드와 웹 서비스 같은 분산 환경에서 제공되는 이질적인 보안 서비스들에게 상호운용성을 제공하기 위해 서로 다른 개체들 사이에서 인증, 인가 및 프로파일 정보를 교환하는 것을 가능하게 한다. SAML이 생성해내는 보안정보는 XML의 형태로 표현되며 이를 "주장[1]"이라고 한다. 주장은 요청하는 목적에 따라 인증 주장(authentication assertion), 속성 주장(attribute assertion), 또는 인가 결정 주장(authorization decision assertion)들 중 하나가 될 수 있다. 주장은 반복적인 인증과 접근 제어검사를 피하는 방법을 제공함으로써 다양한 목적의 환경 전반에 단일인증 기능을 제공한다. SAML은 또한 서비스 소비자가 SAML 요청을 발행하고 SAML 승인기관이 주장을 포함한 SAML 응답을 반환하는 프로토콜을 정의한다. SAML은 웹 서비스를 위한 인증표준이며 또한 OGSA(Open Grid services Architecture) 승인 서비스로부터 인가 주장과 결정을 요청하고 표현하기 위한 메시지 포맷으로서 제안된다[9].

본 논문에서는 SAML API로 구성된 자바기반의 단일인증 라이브러리를 설계 및 구현하고 SAML API를 검증하기 위해 분산된 애플리케이션의 프로토타입을 구성하였다.

## 2. 관련 연구

단일인증의 기본 아이디어는 보안 아키텍처의 복잡성을 단일인증 서비스로 전가시키는 것이다. 단일인증 구조에서 모든 보안 알고리즘들은 단일인증 서비스 내에서 정의된 도메인을 위한 단일 인증 포인트로서 그리고 유일한 인증 포인트로서 존재한다. 따라서 분산 환경에서 단일인증 서비스는 인증과 인가 같은 다양한 보안 특징들을 생산해내는 기존 보안 기반들을 총괄하는 역할을 한다[4, 5].

단일인증을 구현하는 전통적인 기술로는 커버로스와 Needham-Schroeder와 같은 키 교환을 이용하는 인증 프로토콜들이 있다. 이런 프로토콜들은 사용자 인증단계를 키 교환이나 키 확인 단계부터 시작하기 때문에 사용자 응용프로그램은 암호화와 사용자 인증을 위한 새로운 키를 사용하거나 검증된 키를 사용하는 특징이 있다. 단일인증을 구현하기 위한 또 다른 방법으로 쿠키와 SAML같은 토큰기반 프로토콜이 사용된다[10]. 키 교환 프로토콜과 달리 인증 토큰은 독립적으로 설치된 안전한 채널을 통해 전송된다. 안전한 채널이란 브라우저와 함께 사용되는 SSL(Secure Socket Layer)[11]처럼 인증된 클라이언트 키 없이 설치되는 채널을 의미한다. 따라서 인증 토큰은 키 전달 없이 안전한 채널을 통해서 전송될 수 있다. 토큰기반 프로토콜들의 주요 장점은 대부분의 서비스 제공자들이 이미 SSL서버의 인증서를 갖고 있으며 적절한 암호학적인 구현물이 브라우저들을 통해서 모든 클라이언트 기기에서 사용가능하다는 것이다. 뿐만 아니라 서비스 제공자와 동일한 안전한 채널을 통해 사용자에 대한 정보를 제공하기 위한 방법으로 직접적인 관련성이 없는 다양한 인증 토큰들을 사용할 수 있다[10].

쿠키기반의 단일인증 구현은 다음과 같은 문제점들을 갖는다[12]:

- 쿠키들은 때때로 동일한 세션 키를 가지고 암호화되기 때문에 공격자가 단 하나의 쿠키에 대한 세션 키를 찾는다면 시스템내부의 모든 사용자의 쿠키들이 취약해질 수 있다.
- 브라우저 내의 쿠키들은 플러그인이나 다른 방법을 통해 도난당할 수 있다.
- 스푸핑(spoofing) 공격은 쿠키가 다른 도메인들로부터 개별적인 서버들에게 보내져야 한다는 요구를 기술할 방법이 없기 때문에 쿠키의 목적지 제어를 방해할 수 있다.

SAML은 단일 인증 후에 신뢰된 보안 도메인들 사이에서 사이트 접근을 용이하게 하는 적합한 표준이기 때문에 쿠키 기반의 단일인증 솔루션 이상의 장점을 갖는다. 특히, 토큰 역할을 하는 artifact는 보안 도메인 내에서 생성되고 사용자 인증을 위한 다른 보안 도메인들에 보내진다. 다른 보안 도메인에 보내진 artifact는 원래의 보안 도메인에 반환되고 사용자 인증 후 제거된다. 따라서 artifact를 이용한 단일인증 메커니즘을 통해 세션 키가 노출되는 문제와 브라우저에서 토큰들이 도난당하는 문제를 해결할 수 있고 artifact가 URL(Uniform Resource Locator)에 첨부되어 사용자의 인증정보를 담고 있는 메시지를 목적지로 전송하기 때문에 목적지 제어에 대한 문제점을 해결할 수 있다[7].

### 2.1 SAML(Security Assertion Markup Language)

SAML은 시스템들 사이에서 자동적이고 수동적인 상호작용을 위한 단일인증을 제공하도록 설계되어 사용자가 반복적인 인증 없이 또 다른 도메인으로 로그인하는 것을 허용하고 그들의 모든 권한을 정의하고 두 시스템 사이에서 자동화된 메시지 교환을 관리한다. 특히, 기존의 단일인증 구현을 위해 제시된 다양한 기술들과의 상호운용성을 고려하여 Password, Kerberos, Secure Remote Password (SRP), Hardware Token, SSL/TLS Certificate Based Client Authentication, X.509 Public Key, SPKI Public Key,

1) assertion을 주장이라고 번역하였음.

XKMS Public Key, XML Digital Signature 같은 다양한 사용자 인증 방법들을 지원한다. SAML은 다음과 같은 명세서들의 집합으로 구성된다[8].

- Assertions and request/response protocols
- Bindings (SOAP-over-HTTP를 이용한 SAML 요청과 응답 전송 방법)
- Profiles (프레임워크나 프로토콜에 SAML asssertion을 적용하고 추출하는 방법)
- Security considerations while using SAML
- *Conformance guidelines and a test suite*
- Usecase and requirements

SAML은 사용자, 기기 또는 구분이 가능한 모든 개체들에 대한 인증 및 승인 정보를 교환하는 것을 가능하게하기 위해 XML의 부분적인 집합들을 사용하여 시스템이 주장 기반 주체 (subject)의 인증 요청을 허용하거나 거부하는 요청-응답 프로토콜 (Request-response protocol)을 정의한다 [7, 8]. Assertion은 주체에 관한 특정 사실에 대한 선언으로 다음과 같은 세 가지의 주장으로 정의된다.

- 인증(Authentication) 주장: 주체가 이전에 특정한 방법(패스워드, 하드웨어 토큰 또는 X.509 공개 키 등)으로 인증되었다는 것을 나타낸다.
- 인가(Authorization) 주장: 주체가 자원 접근을 허용하거나 거부해야 함을 나타낸다.
- 속성(Attribution) 주장: 주체가 갖는 속성들을 나타낸다.

(그림 1)은 assertion 스키마를 보여주고 (그림 2)는 SAML 승인기관에 의해 발행된 인가 주장을 포함하는 주장을 보여준다.

로컬 시스템들은 주어진 응용프로그램의 보안 수준과 정책들이 시스템을 보호하기에 충분한가를 결정하고 시스템에 발생한 보안 사고가 부정확한 주장이 기반이 된 인가 결정으로부터 야기되었는지의 여부를 판단한다. 이와 같은 SAML의 특징은 주장을 수락하기 전에 기본적인 검증 단계를 반드시 거칠 것을 요구하는 것에 동의한 웹 기반의 비즈니스들 사이에서 신뢰관계와 운영상의 협정을 맺을 것을 요구한다. SAML은 다양한 통신 및 전송 프로토콜들과 연결될 수 있으나 현재의 SAML 표준 명세서에는 HTTP상의 SOAP (Simple Object Access Protocol)과 연결하고 있다[7, 8].

SAML은 Browser/artifact와 Browser/post의 두 가지 프로파일 방식으로 작동될 수 있으며 작동에 있어서 쿠키가 필요하지 않다. Browser/artifact를 사용한다면 SAML artifact는 (그림 3)처럼 URL 쿼리 문자열의 일부로 전달된다. 그림에서 artifact는 주장에 대한 포인터 역할을 한다.

(그림 3)의 순서에 대한 설명은 다음과 같다.

(1) 서버 A에서 인증된 브라우저의 사용자는 서버 B의 데이터베이스에 대한 접근을 요청하면 서버 A는 URL redirect를 생성한다. 생성된 URL redirect는 SAML artifact와 서버 B의 위치 정보를 포함한다.

(2) 브라우저는 사용자를 서버 B로 이동시킨다. 이때 서버 B는 서버 A가 보관하고 있는 주장에 대한 포인터 역할을 하는 artifact를 받는다.

(3) 서버 B는 artifact를 서버 A에게 전송하고 주장을 요청한다.

(4) 서버 B는 전송받은 주장을 체크하고 데이터베이스에 대한 사용자의 접근요청을 수용할지 거절할지를 결정한다.

Browser/post에서 SAML 주장은 HTML form내부에서 브라우저에 업로드 되고 그림 4처럼 HTTP post 페이로드의 일부에 포함되어 목적지 사이트에 전달된다. (그림 4)의 순서에 대한 설명은 다음과 같다.

(1) 서버 A에서 인증된 브라우저의 사용자는 서버 B의 데이터베이스에 대한 접근을 요청한다.

(2) 서버 A는 주장을 갖는 HTML form을 생성하고 사용자에게 되돌려준다.

(3) 브라우저는 서버 B에게 HTML form을 전송한다.

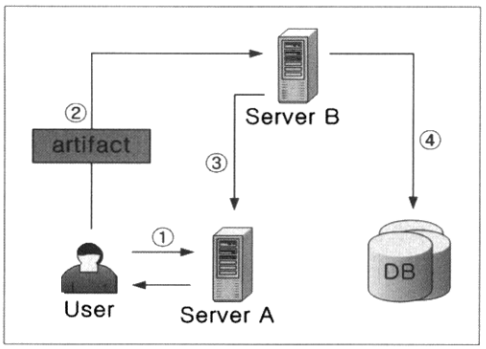
(4) 서버 B는 주장을 체크하고 데이터베이스에 대한 사용자의 접근 요청을 허용하거나 거부할지의 여부를 결정한다.

```
<element name="Assertion" type="saml:AssertionType"/>
 <complexType name="AssertionType">
  <sequence>
   <element ref="saml:Condition" minOccurs="0"/>
   <element ref="saml:Advice" minOccurs="0"/>
   <choice maxOccurs="unbounded"/>
    <element ref="saml:Statement"/>
    <element ref="saml:SubjectStatement"/>
    <element ref="saml:AuthenticationStatement"/>
    <element ref="saml:AuthorizationDecisionStatement"/>
    <element ref="saml:AttributeStatement"/>
   </choice>
   <element ref="ds:Signature" minOccurs="0"/>
  </sequence>
  <attribute name="MajorVersion" type="integer" use="required"/>
  <attribute name="MinorVersion" type="integer" use="ruquired"/>
  <attribute name="AssertionID" type="saml:IDType" use="required"/>
  <attribute name="Issuer" type="string" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
 </complexType>
```

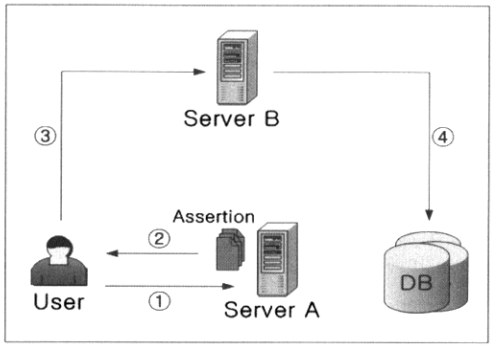
(그림 1) 주장 schema의 구조

```
<saml:Assertion AssertionID="00cda300-0d5de-8521-83c5-c2d9f6847b91"
        IssueInstant="2003-03-23T14:37:56Z"
        Issuer="verisign,Inc." MajorVersion="1" MinorVersion="0">
  <saml:Conditions NotBefore="2003-03-23T14:37:56Z"
                   NotOnOrAf="2003-03-23T18:37:56Z"/>
  <saml:Advice/>
  <saml:AuthenticationStatement
       AuthenticationMethod="password"
       AuthenticationInstant="2003-03-23T14:37:56Z"
    <saml:Subject>
     <saml:NameIdentifier NameQualifier="sejong.ac.kr">
       jijeong
     </saml:NameIdentifier>
    </saml:Subject>
  <saml:AuthenticationStatement>
</saml:Assertion>
```

(그림 2) 주장을 포함하는 assertion




(그림 3) Browser/Artifact



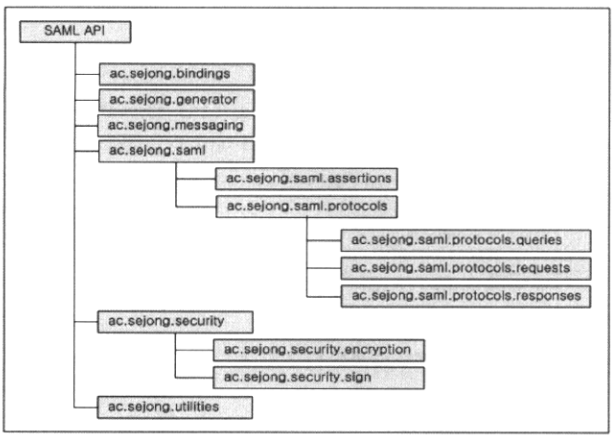

(그림 4) Browser/Post

## 3. 자바 기반 SAML API의 설계 및 구현

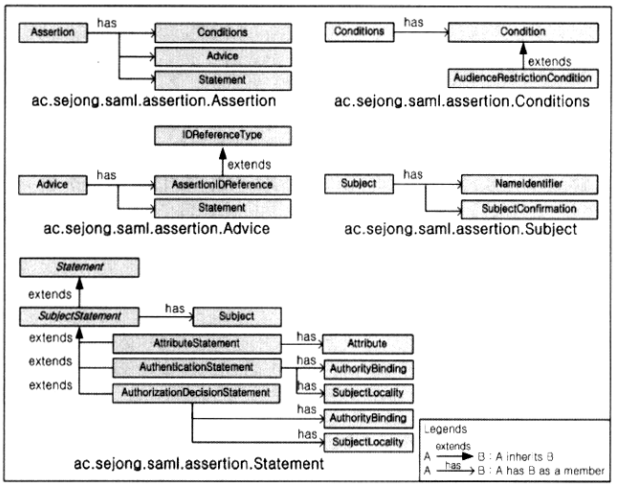
(그림 5)는 JAVA기반으로 구현된 SAML API구조를 보여준다. 패키지의 분류는 "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)" 명세서를 기반으로 한다. 본 논문에서는 assertion, protocol 그리고 messaging 패키지를 기본 패키지로 설계했으며 메시징 기능을 지원하기 위해 generator, utilities 그리고 security 패키지를 설계했다.

(그림 5)는 구현된 SAML API의 전체적인 구조를 보여준다. SAML API는 아래와 같은 패키지들로 그룹화 되었고 각 패키지의 기능은 다음과 같다.

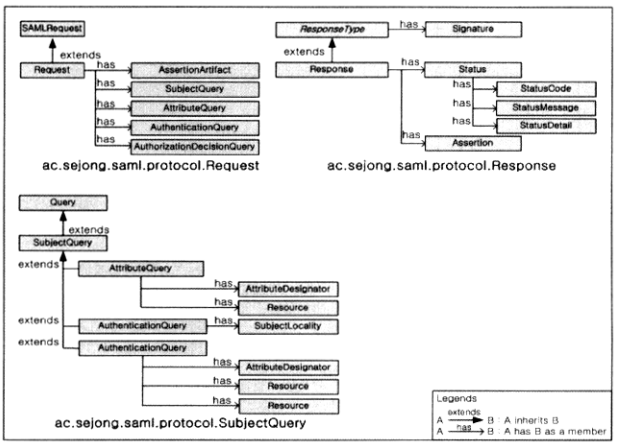
- *Assertion* package : 인증, 인가, 속성 정보를 처리한다.
- *Protocol* package : 주장을 처리하기 위한 SAML 요청과 응답을 처리한다.
- *Messaging* package : 주장을 전송하는 메시징 프레임워크를 포함한다.
- *Security* package : 주장을 전자서명하고 암호화한다.
- *Utilities* package : UUID, UTC 데이터 포멧과 artifact 등을 생성한다.
- *Generator* package : SAML 요청과 응답 메시지를 생성한다.




(그림 5) SAML API의 자바 패키지

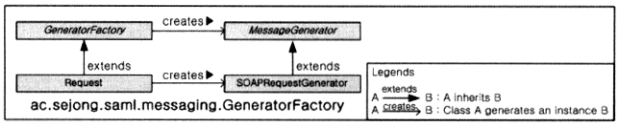



(그림 6) ac.sejong.saml.assertion 패키지의 구조




(그림 7) ac.sejong.saml.protocol 패키지의 구조

(그림 6)과 (그림 7)은 각각 *Assertion* 패키지의 구조와 *Protocol* 패키지의 구조를 보여준다. 프로토콜은 주장을 요청하고 수신하기 위해 협의된 방법을 정의한다[13]. (그림 8)은 *Messaging* 패키지의 구조를 보여준다. *Messaging* 패키지는 SOAP 메시지 내부로 도큐먼트를 전환하고 SAML 메시지가 어떻게 표준 전송 프로토콜과 메시징 프로토콜 위에서 전송되는지를 정의한다[13].




(그림 8) ac.sejong.saml.messaging 패키지의 구조

```
<samlp:Request IssuenInstant="2002-08-08T07:58:32.338Z"
               MajorVersion="1"MinorVersion="0"
               RequestID="a207b-a0-aaa4-11d6-9e6d-75a01a1d3688"
               xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol">
  <samlp:AttributeQuery>
    <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
      <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:am:password
      </saml:ConfirmationMethod>
      <saml:SubjectConfirmationData>
        ...j&5D#Siks233$...
      </saml:SubjectConfirmationData>
    </saml:Subject>
    <saml:AttributeDesignaor attributeName="//sejong.ac.kr/email"
        attributeNamespace="sejong.ac.kr/ams/namespace/common"
        xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"/>
  </samlp:AttributeQuery>
</samlp:Request>
```

(그림 9) SAML Request Message의 생성

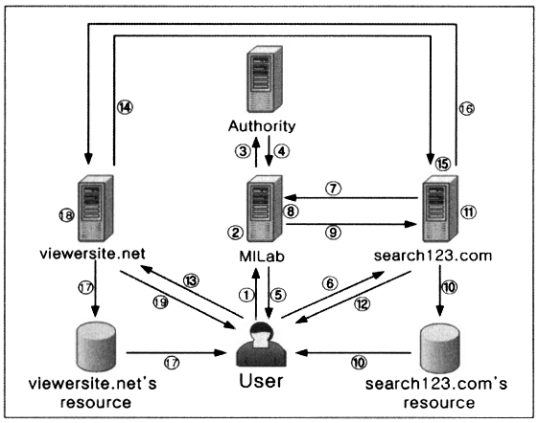
```
<samlp:Request··· >
   <ds:Signature xmlns:http://www.w3.org/2000/09/xmldig#>
     <ds:signedInfo>
       <ds:CanonicalizationMethod Algorithm="···"></ds:CanonicalizationMehtod>
       <ds:SignatureMethod Algorithm="···"></ds:SignatureMethod>
       <ds:Reference URI="">
         <ds:Transforms>
           <ds:Transform Algorithm="···"></ds:Transform>
         </ds:Transforms>
         <ds:DigestMethod Algorithm="···"></ds:DigestMethod>
         <ds:DigestValue>xFXrugYsqtRgnk9wXr7znAenuew=</ds:DigestValue>
       </ds:Reference>
     </ds:SignedInfo>
   <ds:SignatureValue>···zQdWDKC···</ds:SignatureValue>
   <ds:KeyInfo>
     <ds:KeyValue>
     <ds:RSAKeyValue>
       <ds:Modulus>···gdADMHy0=···</ds:Modulus>
       <ds:Exponent>AQAB</ds:Exponent>
     </ds:RSAKeyValue>
     </ds:KeyValue>
     <ds:X509Data>
      <ds:X509IssuerSerial>
        <ds:X509IssuerName>···</ds:X509issuerName>
        <ds:X509IssuerNumber>1045440891</ds:X509SerialNumber>
      </ds:X509IssuerSerial>
      <ds:X509SubjectName>···</ds:X509subjectName>
      <ds:X509Certificate>···Xs+69s=···</ds:X509Certificate>
     </ds:X509Data>
   </ds:KeyInfo>
   <samlp:AttributeQuery>
     ···············
   </samlp:AttributeQuery>
</samlp:Request>
```

(그림 10) Enveloped 형식으로 전자서명 된 SAML Request 메시지

(그림 9)는 SAML 명세를 따르는 메시지를 검증하기 위해 *RequestGenerator* 클래스를 사용하여 생성한 SAML 요청 메시지이다(그림 3과 4의 순서 1에서 생성되는 메시지이다). 생성된 SAML 요청 메시지는 *seucurity.sign* 패키지내의 *Singature* 클래스를 사용해서 서명된다. (그림 10)은 서명된 결과이다. *Signature* 클래스의 서명 과정은 enveloped form의 XML서명 표준을 따른다. (그림 11)은 *generator*

```
<samlp:response InResponseTo="a207b-a0-aaa4-11d6-9e6d-75a01ad3688"
                IssurInstant="2003-03-24T14:36:56Z"
                MajorVersion="1" MinorVersion="0"
                ResponseID="00cda300-0d5e-8521-83c5-c2d9f6847b91"
                xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
                xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol">
  <saml:Status>
    <samlp:StatusCode Value="saml:Success"/>
    <samlp:StatusMessage>this is a message about status</samlp:StatusMessage>
    <samlp:StatusDetail>·····</samp:StatusDetail>
  </saml:Status>
  <saml:Assertion AssertionID="00cda300-0d5e-8521-83c5-c2d9f6847b91"
                  IssueInstant="2003-03-24T14:36:56Z"
                  MajorVersion="1" MinorVersion="0">
    <saml:AutherticationStatement AuthenticationMethod="password"
                                  AuthenticationInstant="2003-03-24T14:36:56Z">
      <saml:Subject>
        <saml:NameIdentifier>jeongjongil </saml:NameIdentifier>
      </saml:Subject>
    </saml:AuthenticationStatement>
  </saml:Assertion>
</samlp:Response>
```

(그림 11) SAML Response 메시지의 생성




(그림 12) SAML기반의 Single Sign-On의 구현

package의 *ResponseGenerator* 클래스를 사용하여 생성된 SAML 응답 메시지를 보여준다 (그림 3과 4의 순서 4에서 생성되는 메시지이다). 생성된 SAML 응답 메시지는 *security.sign*패키지의 *Signature* 클래스로 서명된다.

본 논문에서는 분산 서비스의 프로토타입을 구성하여 개발된 SAML API를 검증하였다. SAML API를 사용하여 구현된 단일인증 모델의 전반적인 구조는 (그림 12)와 같다. 개략적인 흐름은 다음과 같다. 사용자는 MILab의 도서 검색 시스템을 사용하기 위해 login절차를 거친다. MILab의 도서검색시스템에서 사용자가 찾는 도서가 발견되지 않을 경우 MILab과 신뢰관계가 형성된 여러 inter-site들 중 search123.com에 접속해서 도서검색 시스템을 이용한다. 사용자는 더 많은 도서를 찾기 위해 viewersite.net에 접속해서 도서검색 시스템을 이용할 수 있다. MILab에서 제공되는 모든 inter-site는 신뢰관계가 형성되어 있으므로 각 site에서 생성한 artifact를 전달받고 비교해서 일치할 경우 사용자 인증을 위해 각 site마다 개별적인 login절차를 생략하고 원하는 자원을 이용할 수 있게 된다. 세부적인 설명과 패키지의 사용은 다음과 같다.

1) login to MILab
2) artifact 생성 - MILab (ac.sejong.utilities.Artifact)
3) request 처리과정 - MILab
  3-1) 제공된 credential을 이용하여 AttributeQuery를 포함하는 request document 생성한다. (ac.sejong.saml.RequestGenerator)
  3-2) 생성된 request document에 전자서명 한다. (ac.sejong.security.sign.DesigGenerator)
  3-3) 전자서명 된 request document를 SOAPMessage로 변환 후 Authority에 전송 후 응답을 기다린다.
    (ac.sejong.messaging.SOAPMessageGenerator)
4) response 처리과정 - Authority
  4-1) MILab에서 전송된 SOAPMessage에서 Request document객체만 추출한다. (ac.sejong.messaging.ExtractRequest)
  4-2) Request document의 전자서명을 검증한다. (ac.sejong.security.verify.DsigVerifier)
  4-3) Request document에서 request id와 subject를 추출한다. (ac.sejong.saml.RequestFilter)
  4-4) id와 subject에 관련된 정보로 assertion을 생성하고 Response document를 생성한다. (ac.sejong.saml.ResponseGenerator)
  4-5) Response document에 전자서명 한다. (ac.sejong.security.sing.DesigGenerator)
  4-6) 전자서명 된 response document를 SOAPMessage로 변환 후 MILab에 전송한다. (ac.sejong.messaging.SOAPMessageGenerator)
5) Artifact의 부여
  5-1) Authority에서 전송받은 SOAPMessage에서 Response document 객체만 추출한다. (ac.sejong.messaging.ExtractResponse)
  5-2) Response document의 전자서명을 검증한다. (ac.sejong.security.verify.DsigVerifier)
  5-3) Response document에서 assertion을 추출하여 결합한 document를 생성한다. (ac.sejong.saml.ArtifactListMaker)
  5-4) Artifact를 사용자에게 부여한다.
6) Passport역할을 하는 Artifact의 전송
  6-1) 사용자는 search123.com에 접속 시 MILab에서 부여받은 artifact만 전달한다.
7) 사용자 검증을 위한 Artifact의 전송
  7-1) search123.com은 전달받은 artifact를 MILab으로 전송한다.
8) Artifact의 비교
  8-1) MILab은 search123.com으로부터 전달받은 artifact와 사용자에게 부여한 artifact를 비교한다. (ac.sejong.utilities.Artifact)
9) Assertion의 전송
  9-1) artifact일치 시 Assertion에서 artifact를 제거하고 Assertion document를 생성한다.
    (ac.sejong.utilities.Artifact ac.sejong.saml.ResponseGenerator)
  9-2) Assertion document객체에 전자서명 한다. (ac.sejong.security.sign.DesigGenerator)
  9-3) 전자서명 된 assertion document를 SOAPMessage로 변환 후 search123.com에 전송한다. (ac.sejong.messaging.SOAPMessageGenerator)
  9-4) MILab에서 생성한 assertion과 artifact를 폐기한다. (ac.sejong.saml.ArtifactListMaker )
10) Resource의 제공
  10-1) search123.com은 전송 받은 SOAPMessage에서 assertion document만을 추출한다. (ac.sejong.messaging.ExtractResponse)
  10-2) assertion document의 전자서명을 검증한다. (ac.sejong.security.sign.DesigGenerator)
10-3) assertion을 분석해서 사용자에게 Resource를 제공할 것인지 거부할 것인지 결정한다. (ac.sejong.saml.AssertionResolver)

SAML API를 이용한 단일인증 모델은 1)에서 10)번까지의 순서를 마침으로써 사용자의 인증이 완성되고 신뢰관계가 형성된 보안 도메인의 자원에 재 인증절차 없이 접근하는 것이 가능해진다. 11)에서 19)번까지의 순서는 1)에서 10)번까지의 순서와 동일한 과정이고 단지 최초 로그인 시 인증정보를 제공하는 과정만 제외되었다. 따라서 신뢰관계가 형성된 보안 도메인 사이에서 1)에서 10)번째 순서까지의 과정을 수행하는 인증에이전트를 각각의 보안 도메인에 설치하는 것으로 구현할 수 있다.

## 4. 결 론

본 논문에서 구현한 API는 기존의 단일인증 기술들을 통합할 수 있는 방법을 제시하는 SAML명세를 충실히 구현하여 비 XML기반의 인증기술들과의 상호운용성을 제공한다. 본 논문에서 구현한 SAML API는 다음과 같은 주요 특징들을 갖는다.

- SOAP을 이용한 XML기반 메시지 구조의 전송 : XML 기반의 메시지 구조를 완벽하게 보존할 수 있기 때문에 유효한 바인딩을 가능하게 한다.
- 전자서명 지원 : 각 시스템 간에 전송되는 메시지를 전자서명 하여 메시지의 무결성과 부인방지기능을 제공한다.
- XML암호화 지원 : XML암호화는 SAML 명세서에 선택적인 사항으로 명시되어 있지만 시스템 간에 전송되는 메시지를 효율적으로 암호화하고 메시지의 기밀성을 보장하기 위해 구현하였다[14].

특히, 분산 환경에서 메시지의 교환이 빈번하고 메시지가 중간매개체를 경유하여 전달되는 특성상 SSL(Secure Socket Layer)과 같은 점 대 점 (Point-to-Point)간의 채널 보안만으로는 메시지의 무결성과 기밀성을 유지하는데 한계가 있다. 즉, SSL은 통신 상대방이 X.509 인증서를 통해 식별된 사용자라는 사실만을 증명하고 통신 상대방이 메시지를 보낼 수 있는 권한을 부여 받은 X.509 인증서의 소유주임을 보장할 수는 없다.

다중 도메인 (End-to-End) 간에 보안정보를 교환하는 과정에서 발생할 수 있는 "메시지 삽입", "메시지 삭제", "메시지 변조", 및 "메시지 노출" 등의 공격을 방지하기 위해서는 응용계층 수준에서 메시지에 대한 전자서명 및 암호화가 필요하다. 본 논문에서 구현된 SAML API가 제공하는 전자서명 및 XML 암호화는 서명 및 암호화가 필요한 대상을 메

시지 전체 또는 메시지의 일부분으로 선택적으로 적용할 수 있기 때문에 필요에 따라 연산부담을 경감할 수 있는 방안을 제공한다.

따라서 구현된 SAML API는 웹 서비스 환경에서 단일인증 구현기술에 관계없이 사용자의 인증정보 교환을 가능하게 함으로써 웹 서비스 표준을 채택한 그리드 아키텍처에서 다양한 서비스 이용의 편리성과 보안 성능의 개선 방안을 제시한다.

향후에는 개발된 SAML 라이브러리를 EDMS (Electronic Document Management System), 그룹웨어 시스템 같은 실세계 시스템에 적용하고 사용자를 위한 인가에 관한 연구가 진행될 것이다. 또한 단일인증과 인가 표준인 XACML (eXtensible Access Control Markup Language)[15]과 XKMS (XML Key Management System)[16] 와의 연동에 대한 연구를 수행할 것이다.

## 참 고 문 헌


[1] I. Foster, C. Kesselman, J.M. Nick, S. Tueckse, "The Physiology of the Grid" An Open Grid Services Architecture for Distributed Systems Integration, http://www.globus.org/research/papers/ogsa.pdf

[2] I. Foster, C. Kesselman, "The Globus Project : A Status Report." Future Generation Computer Systems, Volume.15, pp.607-621, 1999.

[3] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecure for Computational Grids." Proc. 5th ACM Conference on Computer and Communications Security Conference, pp.83-92, 1998.

[4] A. Volchkov, "Revisiting Single Sign-on", A Pragmatic Approach in a New Context. IT Professional, Volume.3, Issue.1, pp.39-45, Jan/Feb., 2001.

[5] T.A. Parker, "Single Sign-On Systems-The Technologies and The Products", European Convention on Security and Detection, pp.151-155, 16-18 May, 1995.

[6] B. Pfitzmann, "Privacy in Enterprise Identity Federation Policies for Liberty Single Signon.", 3rdWorkshop on Privacy Enhancing Technologies(PET 2003)m Dresden, March, 2003.

[7] Assertion and Protocol for the OASIS Security Assertion Markup Language(SAML) V1.0, http://www.oasis-open.org/committees/security

[8] Binding and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1, http://www.oasis-open.org/committees/security

[9] Global Grid Forum OGSA Security Working Group, "Use of SAML for OGSA Authorization", http://www.globus.org/ogsa/Security

[10] B. Pfitzmann, B. Waidner, "Token-based Web Single Signon with Enabled Clients" IBM Research Report RZ 3458 (#93844), November, 2002.

[11] A. Frier, P. Karlton, P. Kocher, "The SSL 3.0 Protocol.", Net Scape Communications Corporation, Nov., 18, 1996.

[12] V. Semar, "Single Sing-O Using Cookies for Web Applications", Proceedings, IEEE 8th International Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprises (WET ICE '99), pp.158-163, 1999.

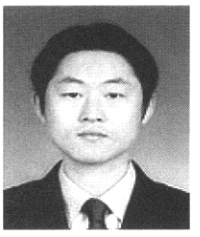
[13] B. Galbraith, R. Trivedi, D. Whitney, D.V. Prasad, M. Janakiraman, A. Hiotis, W. Hankison, "Professional Web services Security", Wrox Press, 2002.

[14] XML Encryption WG, http://www.w3.org/Encryption/2001/

[15] eXtensible Access Control Markup Language TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

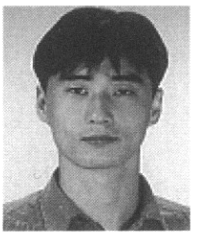
[16] XML Key Management Specification(XKMS), http://www.w3.org/TR/xkms





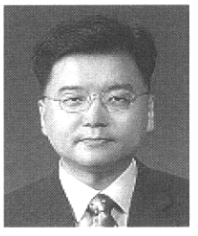
### 정 종 일

e-mail : jijeong@gce.sejong.ac.kr
2002년 세종대학교 컴퓨터공학과(학사)
2004년 세종대학교 대학원 컴퓨터공학과 (공학석사)
2004년~현재 세종대학교 대학원 컴퓨터공학과 박사과정
관심분야 : XML 보안, XML 기반의 미들웨어, 전자상거래



### 유 석 환

e-mail : shwyu@gce.sejong.ac.kr
2002년 세종대학교 컴퓨터공학과(학사)
2003년~현재 세종대학교 대학원 컴퓨터공학과 석사과정
관심분야 : 웹 서비스 보안, 임베디드 컴퓨팅



### 신 동 규

e-mail : shindk@sejong.ac.kr
1986년 서울대학교 계산통계학과(학사)
1992년 M.S. in Computer Science, Illinois Institute of Technology
1997년 Ph.D in Computer Science, Texas A&M University
1986년~1991년 한국국방연구원, 연구원
1997년~1998년 현대전자 멀티미디어연구소, 차장(책임연구원)
1998년~현재 세종대학교 컴퓨터공학과 부교수
관심분야 : XML 보안, 전자상거래, MPEG

## 신 동 일

e-mail : dshin@sejong.ac.kr
1988년 연세대학교 전산과학과(이학사)
1993년 M.S. in Computer Science, Washington State University
1997년 Ph.D in Computer Science, University of North Texas
1997년~1998년 시스템공학연구소 선임연구원
1998년~현재 세종대학교 컴퓨터공학과 조교수
관심분야 : 무선인터넷, HCI, 게임엔진, CSCW

## 차 무 홍

e-mail : bidon@gce.sejong.ac.kr
2003년 세종대학교 컴퓨터공학과(학사)
2003년~현재 세종대학교 컴퓨터공학과 석사과정
관심분야 : XML Security, Web Services, ebXML