분산 Application을 위한 SAML(Security Assertion Markup Language) 지원의 자바 기반 Single Sign-On Library의 설계 및 구현

정종일*, 차무홍*, 신동규*, 신동일*, 문기영**, 김주한**

*세종대학교, 컴퓨터공학과

**한국전자통신연구원

A Java-Based Single Sign-On Library Supporting SAML(Security Assertion Markup Language) for Distributed Applications

Jong-il Jeong*, Moo-hong Cha*, Dong-kyoo Shin*, Dong-il Shin*, Ki-young Moon**, Ju-han Kim**

*Department of Computer Science and Engineering Sejong Univ.

**Electronics and Telecommunications Research Institute

요 약

Single Sign-On(SSO)은 사용자가 단 한번의 인증으로 분산된 시스템에서 제공하는 다른 서비스에 로그인하는 것을 가능하게 하는 보안 특징이다. SAML은 서로 다른 entity들 사이에서 인증, 권한 그리고 profile 정보의 교환을 가능하게 하는 XML기반의 SSO표준이다. 이러한 표준은 분산 환경에서 서로 다른 보안 서비스들 사이에 상호운용성을 제공한다. 본 논문에서는 SAML APIs로 구성된 자바 기반의 SSO library를 설계 및 구현하고 SAML APIs를 검증하기 위해 분산 application의 prototype을 구성하였다. 개발된 library에서 무결성, 부인방지, 그리고 기밀성 같은 보안고려사항들을 XML기반의 서명과 암호화를 적용하여 보장한다.

I. 서론

몇 년 전 만해도 사용자들은 소수의 application에만 접근했다. 현재는 Web application의 통합으로 인해 일반적인 사용자가 접근하는 수많은 서비스가 증가했고 사용자들은 다양한 사용자 name과 password를 매일 관리하도록 강요되고 있다. Web application의 통합이 갖는 주요한 보안 문제는 보안 infrastructure가 분산된다는 것이고 이런 architecture들은 대개 key 보안 특징들이 시스템의 전반에 걸쳐 구성되어야하는 것을 요구한다. 인터넷과 인트라넷 사이트 상에 서로 다른 위치에 분산된 자원들을 사용하기 위해 각 application에 대한 사용자의 인증과 권한은 필수적이고 이러한 필수사항은 시스템의 성능과 보안에 부담을 가중시킨다. 사용자는 반드시 각 application에 대한 username과 password를 기억해야하고 시스템의 관리자는 수많은 password들을 DB에 저장 관리해야하며 site나 application에서 password들의 빈번한 전송으로 인해 잠재적인 보안 문제들에 직면하게 된다[1].

최근에 XML기반의 보안관련 정보교환을 위한 새로운 표준인 SAML(Security Assertion Markup Language)이 OASIS(Originalization for Advancement of Structured Information Standards)에 의해서 권고되었다. SAML은 W3C에서 제안한 Web Services같은 분산 환경에서 서로 다른 보안 서비스들 간에 상호운용성을 제공하기 위해 서로 다른 entity들 간에 인증, 권한 그리고 profile정보의 교환을 가능하게 한다. SAML에서 설명되는 보안정보는 assertion이라는 XML 형태로 표현된다. assertion은 인증 assertion, 속성

assertion 그리고 권한 결정 assertion이 될 수 있다. SAML authority는 assertion을 발행하며 서비스 제공자나 비즈니스 조직이 SAML authority가 될 수 있다. Assertion은 반복적인 인증을 피하는 방법을 제공하고 접근 제어를 체크하여 다양한 목적의 환경을 포괄하는 single sign-on기능을 제공하게 된다. SAML은 또한 서비스 소비자가 SAML request를 발행하고 SAML authority가 assertion을 포함한 SAML response를 반환하는 방법인 protocol을 정의한다.

본 논문에서는 SAML APIs로 구성된 자바 기반의 SSO library를 설계 및 구현하고 SAML APIs를 검증하기 위한 분산 application의 prototype을 구성했다.

II. 본문

1. 배경지식

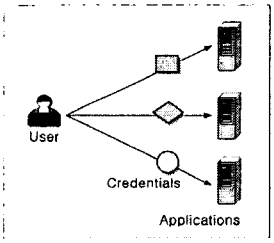

그림 1: 기존 login 절차

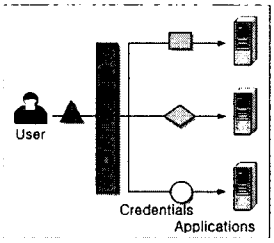

그림 2: SSO login절차

그림 1은 사용자가 접근하기를 원하는 application의 숫자만큼 credential(User ID and password)이 필요하고 각 application을 사용하기 위해서는 개별적으로 login 절차를 거쳐야만 한다. 사용자들은 평균적으로 다섯 개에서 여덟 개의 패스워드를 사용하며 패스워드를 노트에 적어서 보관하거나 극히 단순한 패스워드들만을 사용한다. 따라서 허술한 패스워드 관리는 악의를 가진 제3자에게 쉽게 노출될 수 있으므로 보안이 취약해지는 결과를 초래한다.

SSO(Single Sign-on)[2]이란, 단 한번의 login만으로 기업의 각종 업무 시스템이나 인터넷 서비스에 접속할 수 있게 해주는 보안 응용 솔루션으로 최근 각 기업들의 인트라넷 시스템과 웹 서비스가 대폭 확장됨에 따라 SSO 시스템도 급속히 확대되고 있다. SSO를 도입하면 각각의 시스템마다의 인증 절차를 밟지 않고도 사용자에게 부여된 1개의 계정만으로 다양한 시스템에 접근할 수 있기 때문에 사용자의 편의가 대폭 높아지고, 관리자 입장에서도 인증정책의 변경이나 권한 설정이 수월해져 관리비용 및 수고를 크게 덜 수 있다 [2].

SSO을 구현함으로써 인증절차를 단순화하고 이질적인 platform과 application 환경에서 패스워드의 수를 감소시켜 관리와 잠재적인 보안문제를 해결할 수 있다. 따라서 application의 사용이 더욱 용이해지게 되고 기술적인 지원에 필요한 비용을 절감할 수 있다.

그림 2는 기존의 application과 사용자사이에 SSO layer를 두어 하나의 credential로 인증 후 각 application에 재 인증하지 않고 접근할 수 있음을 보여준다.

2. SAML

최근 OASIS는 도메인 간에 인증과 권한 정보 교환을 위한 표준인 SAML(Security Assertion Markup Language)을 완성했다. SAML은 시스템 간에 자동적이고 수동적인 상호작용 모두를 위한 single sign-on을 제공하도록 설계되었다. SAML은 사용자들이 또 다른 도메인에 로그인 하도록 허용하고 사용자들의 모든 permission을 정의하거나 두 party간에 자동화된 메시지 교환을 관리할 것이다. SAML은 다음과 같은 component들을 정의한 명세서들의 집합이다.

- Assertions and request/response protocols
- Bindings(the SOAP-over-HTTP method of transporting SAML requests and responses)
- Profiles(for embedding and extracting SAML assertions in a framework or protocol)
- Security consideration while using SAML
- Conformance guidelines and test suite
- Use cases and requirements

SAML은 사용자, device 또는 subject라고 불리는 동일성을 증명할 수 있는 entity에 관련된 인증과 권한 정보의 교환을 가능하게 한다. XML의 subset을 사용함으로써 SAML은 시스템이 assertion 기반의 subject를 accept하거나 reject하도록 하는 request-response protocol을 정의한다 [3].

Assertion은 subject에 대한 특정 사실의 선언이다. SAML은 세 가지 type의 assertion을 정의한다.

- Authentication: subject가 다른 방법(password, hardware token or X.509 public key)에 의해 이전에 인증되었다는 것을 지시한다.
- Authorization: subject가 자원에 접근하는 것이 허용 또는 거부되었다는 것을 지시한다.
- Attribute: subject가 속성에 관련되었다는 것을 지시한다.

그림 3은 assertion 스키마를 보여주고 그림 4

는 SAML authority가 발행한 인증 assertion을 포함하는 assertion statement를 보여준다.

```
<element name="Assertion" type="saml:AssertionType"/>
 <complexType name="AssertionType">
  <sequence>
   <element ref="saml:Conditions" 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="required"/>
  <attribute name="AssertionID" type="saml:IDType" use="required"/>
  <attribute name="Issuer" type="string" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
 </complexType>
```

그림 3: Assertion 스키마의 구성

```
<saml:Assertion AssertionID="00cda300-0d5e-8521-63c5-c2db96947b91"
                IssueInstant="2003-03-24T14:36:56Z"
                Issuer="verisign, Inc." MajorVersion="1" MinorVersion="0">
  <saml:Conditions NotBefore="2003-03-24T14:36:56Z"
                   NotOnOrAfter="2003-03-24T14:36:56Z" />
  <saml:Advice />
  <saml:AuthenticationStatement
              AuthenticationMethod="password"
              AuthenticationInstant="2003-03-24T14:36:56Z">
   <saml:Subject>
    <saml:NameIdentifier NameQualifier="sejong.ac.kr" >
              jijeong
    </saml:NameIdentifier>
   </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
```

그림 4: AuthenticationStatement를 포함하는 Assertion의 결과

SAML은 assertion내에 confidence를 몇 개나 배치해야 하는가에 대해서는 기술하지 않는다. Local 시스템은 제공된 application의 security level 과 policy들이 *organization*을 보호하기에 충분한지의 여부를 결정하고 damage가 부정확한 assertion기반의 권한 결정으로부터 기인하는지의 여부를 결정한다. 이러한 SAML의 특징은 assertion을 수용하기 전에 검증의 기본단계를 고수함에 따라 웹 기반의 비즈니스들 사이에서 신뢰관계와 운영적인 동의를 더욱 확고히 할 것이다.

SAML은 SOAP(Simple Object Access Protocol)-over-HTTP등 다양한 communication과 전송 protocol로 binding 될 수 있다[3].

SAML은 browser/artifact와 browser/post 두 가지의 profile로 운영된다. 그림 5는 browser/artifact는 SAML artifact가 URL query string의 일부로서 전송되는 것을 보여준다. SAML artifact는 assertion에 대한 pointer역할을 한다. 그림 5에서의 step은 아래와 같이 설명된다.

(1) Server A에서 인증된 browser의 사용자는 Server B의 database에 접근 요청을 한다. Server A는 SAML artifact를 포함하는 Server B의 URL redirect를 생성한다.

(2) Browser는 사용자를 Server B로 redirect하고 Server B는 Server A의 assertion을 가리키는 artifact를 받는다.

(3) Server B는 Server A로 artifact를 전송하고 full assertion을 얻는다.

(4) Server B는 assertion을 검토하고 database에 대한 사용자의 접근요청을 허용할 것인지 혹은 거절할 것인지를 결정한다.

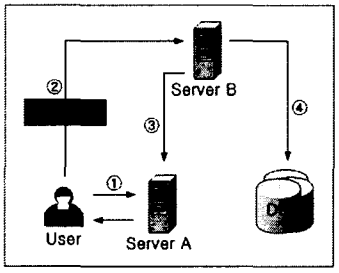

그림 5: browser/artifact방식

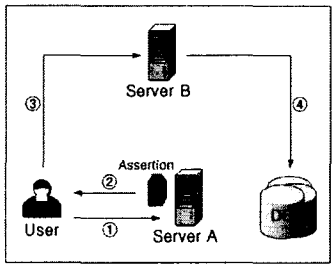

그림 6: browser/post방식

browser/post방식에서 SAML assertion은 HTML form안에서 browser로 upload되고 HTTP post payload의 일부로서 목적지 사이트에 전달된다. 그림 6에서의 step은 아래와 같이 설명된다.

(1) Server A에서 인증된 browser의 사용자는 Server B의 database에 접근 요청을 한다.

(2) Server A는 SAML assertion을 포함한 HTML form을 생성하고 사용자에게 반환한다.

(3) Broswer는 Server A에서 생성된 form을 Server B로 전달한다.

(4) Server B는 assertion을 검토하고 database

에 대한 사용자의 접근요청을 허용할 것인지 혹은 거절할 것인지를 결정한다.

3. SAML APIs의 설계 및 구현

본 논문에서는 그림 7처럼 Java package형태의 SAML APIs를 설계하고 구현했다.

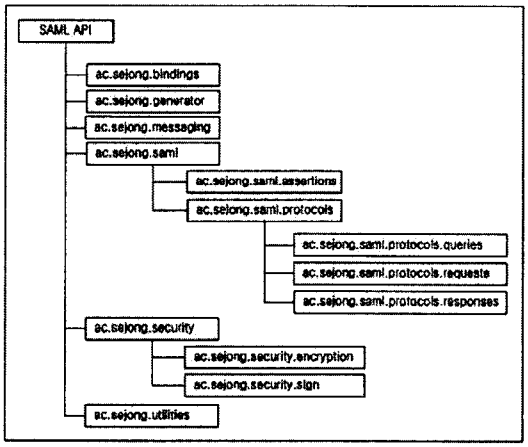

그림 7: SAML APIs의 java package

package의 분류는 "Assertions and Protocol for the OASIS Security Assertion Markup Language(SAML)"[4] 명세서를 기반으로 한다. 본 논문에서는 *assertion, protocol* 그리고 *messaging* 세 개의 기본 패키지를 설계하고 messaging기능을 지원하기 위해 *generator, utilities* 그리고 *security* package를 설계했다. 구현된 SAML APIs들은 아래의 package들로 그룹화 되었다. 각 package의 기능은 다음과 같다.

- *Assertion* package: 인증, 권한, 속성 정보를 다룬다.
- *Protocol* package: assertion을 처리하기 위해 SAML request/response 메시지 쌍을 다룬다.
- *Messaging* package: assertion을 전송하는 messaging framework을 포함한다.
- *Security* package: assertion을 전자서명하고 암호화한다.
- *Utilities* package: UUID, UTC Data format과 artifact등을 생성한다.

본 논문에서는 SAML 명세서를 따르는 message를 검증했다.

그림 8은 *generator* package의 *RequestGenerator* class를 이용하여 SAML request message를 생성한 것이다(그림 5, 6의 step 1을 수행한다).

그림 9는 *security.sign* package의 *signature* class를 사용하여 SAML request message를 전자서명한 것이다. *signature* class의 서명 process는 XML서명 표준으로 enveloped form을 따른다.

```
<samlp:Request IssueInstant="2002-08-08T07:58:32.336Z"
               MajorVersion="1" MinorVersion="0"
               RequestID="a207b-a0-eaa4-11d6-9a6d-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:NameIdentifier>jijeong</saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:am:password
        </saml:ConfirmationMethod>
        <saml:SubjectConfirmationData>****</saml:SubjectConfirmationData>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:AttributeDesignator
          AttributeName="//sejong.ac.kr/email"
          AttributeNamespace="sejong.ac.kr/arms/namespace/common"
          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" />
  </samlp:AttributeQuery>
</samlp:Request>
```

그림 8: SAML request메시지의 생성

```
<samlp:Request...>
  <ds:Signature xmlns:http://www.w3.org/2000/09/xmldsig#>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="..."></ds:CanonicalizationMethod>
      <ds:SignatureMethod Algorithm="..."></ds:SignatureMethod>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="..."></ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="..."></ds:DigestMethod>
        <ds:DigestValue>xFXrugYsqtPgnk9wXi7znAenuew=</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:X509SerialNumber>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 9: Enveloped방식으로 서명된 SAML request메시지

그림 10은 *generator* package의 *ResponseGenerator* class를 이용하여 SAML response message를 생성한 것이다(그림 5, 6의 step 4를 수행한다).

```
<samlp:Response InResponseTo="a207b-a0-eee4-11d6-9e6d-75a01a1d6688"
                IssueInstant="2003-03-24T14:36:56Z"
                MajorVersion="1" MinorVersion="0"
                ResponseID="00cda300-0c5e-6521-63c5-c2b9f6847b91"
                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>http://namespace1</samlp:StatusDetail>
  </saml:Status>
  <saml:Assertion AssertionID="00cda300-0c5e-6521-63c5-c2b9f6847b91"
                IssueInstant="2003-03-24T14:36:56Z"
                Issuer="sejong.ac.kr"
                MajorVersion="1"
                MinorVersion="0">
    <saml:AuthenticationStatement AuthenticationMethod="password"
                                  AuthenticationInstant="2003-03-24T14:36:56Z">
      <saml:Subject>
        <saml:NameIdentifier>xijeong</saml:NameIdentifier>
      </saml:Subject>
    </saml:AuthenticationStatement>
  </saml:Assertion>
</samlp:Response>
```

그림 10: SAML response메시지의 생성

4. SAML APIs검증 prototype

그림 11은 SAML API를 이용하여 구현한 SSO모델의 전체적인 구조와 흐름을 보여준다.

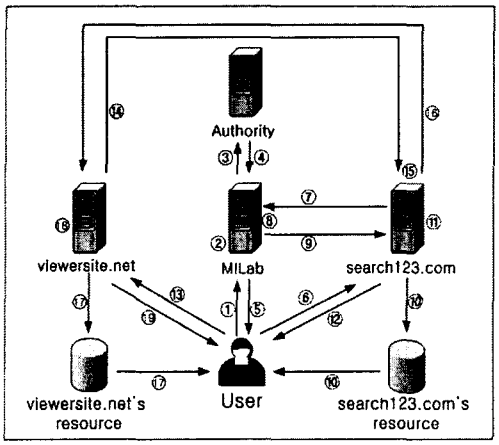

그림 11: SAML 기반의 SSO model의 구현

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.

위와 같은 절차를 적용하기 위해서 SAML library의 적용은 다음과 같다.

①사용자는 credential을 제공하여 login한다.

②MILab은 MILab의 local machine의 정보를 바탕으로 고유한 artifact를 생성한다.

③MILab은 사용자에게 artifact를 부여한다.(MILab이 제공하는 inter-site의 link에 get방식으로 artifact를 첨부한다.)

④MILab은 사용자의 인증 또는 속성을 Authority에게 request한다.

⑤Authority는 사용자의 인증 또는 속성정보를 assertion 형태로 response한다.

⑥사용자는 MILab과 신뢰관계가 형성된 search123.com에 접속해서 도서검색시스템을 사용하기 위해 부여받은 artifact를 가지고 search123.com에 접근한다.

⑦search123.com은 사용자의 인증여부를 확인하기 위해 사용자의 artifact를 MILab에 전송한다.

⑧MILab은 search123.com에서 전송받은 artifact와 MILab에서 사용자에게 부여했던 artifact를 비교한다.

⑨artifact가 일치하면 MILab은 사용자의 인증 또는 속성 assertion을 포함하는 메시지의 무결성을 보장하기 위해 전자서명 후 search123.com에 전송하고 MILab이 생성한 artifact는 폐기한다.

⑩search123.com은 MILab에서 수신한 사용자 assertion의 전자서명을 검증 후 분석하여 사용자에게 serach123.com의 도서검색시스템의 접근허용여부를 결정한다.

⑪search123.com은 search123.com의 local machine의 정보를 바탕으로 고유한 artifact를 생성한다.

⑫search123.com은 사용자에게 artifact를 부여한다.(search123.com이 제공하는 inter-site의 link에 get방식으로 artifact를 첨부한다.)

①~⑫까지의 과정은 viewrsite.net, MILab, 그리고 search123.com에 공통적으로 적용되는 인증 및 접근 절차이다. 따라서 나머지 ⑬~⑲까지의 과정은 credential을 제공하여 login하는 과정을 제외하고는 모두 동일하게 적용된다.

각 과정별 SAML APIs의 적용은 다음과 같다.

②,③,⑧,⑪,⑫번은 artifact를 생성하고 부여 및 비교하는 과정으로 *utilities* package의 *Artifact* class를 이용한다.

④,⑤번은 request와 response메시지를 생성 및

전자 서명하여 각 security domain간 message를 전송하고 수신하기 위해 메시지를 *security* package의 *SingatureUtill* class를 이용하여 전자 서명하고 전자 서명된 message를 전송하기 위해 *messaging* package의 *SOAPRequestGenerator*와 *SOAPResponseGenerator* class를 이용한다.

⑨,⑩번은 메시지의 무결성 및 전자서명의 검증을 수행하는 과정으로 security package의 *SignatureUtil* class를 이용한다.

III. 결론

본 논문에서는 SAML 표준을 지원하는 SSO library를 설계하고 구현하였다. 구현된 SAML APIs는 다음과 같은 특징들을 갖는다.

- SAML message가 SOAP을 통해 전송되기 때문에 XML 기반의 message 구조를 완전하게 유지할 수 있다.
- Integrity와 non-repudiation은 전송된 message에 전자서명을 함으로써 보장된다.
- Confidentiality는 전송된 message에 암호화를 함으로써 보장된다. XML암호화가 적용되었기 때문에 각각의 element는 효율적으로 암호화될 수 있다.

최근 분산 시스템은 W3C에서 제안한 Web services를 채택했다. SAML은 Web services를 위한 XML기반의 SSO표준이다. SAML은 XML기반의 분산 application들이 대중화될 때 Web services로서 널리 사용되게 될 것이다. 본 논문은 EDMS(Electronic Document Manage Systems)와 groupware 시스템 같은 실제 시스템에 본 SAML library를 적용하고 사용자들에 대한 권한을 계속적으로 연구할 것이다. 또한 권한 관련 표준인 XACML(eXtensible Access Control Markup Language)과 XKMS(eXtensible Key Management Specification)에 대한 연구를 지속할 것이다.

참고문헌

[1] Volchkov, A.: Revisiting single sign-on: a pragmatic approach in a new context. IT Professional, Volume: 3 Issue: 1, Jan/Feb(2001) 39-45

[2] Single Sign On(SSO) http://www.oasis-open.org/committees/security/docs/cs-sstc-bindings-01.pdf

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

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