SAML인증을 적용한 SSO (Single Sign On)모델의 구현

정종일*, 성백호, 박병철, 신동규, 신동일
세종대학교 컴퓨터공학과
e-mail : {jijeong, guardia, leon, dkshin, dshin}@gce.sejong.ac.kr

Implementation of the SSO model applying the SAML authentication

JongIl Jeong*, Baekho Sung, ByungChul Park, Dongil Shin, Dongkyoo Shin
Dept of Computer Engineering, Sejong University

요 약

인터넷 사용자들의 다양한 요구에 따라 존재하는 많은 자원들에 접근하기 위해 이용되었던 기존의 개별적인 인증절차는 패스워드 관리와 보관 그리고 공개된 네트워크를 통해 빈번히 전송되어지는 보안상의 취약점이 노출되어있다. 단일 인증을 통해 보다 효율적이고 안전하게 필요한 자원에 접근하는 방법으로 SAML인증을 적용한 Single Sign On모델을 구현하였다.

1. 서론

수많은 인터넷 사이트와 인트라넷의 자원을 사용하기 위해서 사용자를 인증하고 권한을 부여하는 일은 매우 중요하다.

사용자가 접근하려는 자원들은 서로 다른 위치에 분산되어 존재하며 사용자는 각 자원을 보유하고 있는 사이트마다 인증절차를 거쳐야만 자원을 이용할 수 있다. 반복적인 인증 절차는 사용자와 시스템 관리자 모두에게 패스워드관리에 대한 부담을 갖게 한다. 사용자는 자신소유의 수많은 패스워드들을 항상 기억해야하고 각각의 사이트에 해당하는 패스워드를 정확히 선택해야 하는 어려움이 있다. 시스템관리자로서는 사용자의 인증절차를 위해서 사용자의 정보와 패스워드들을 데이터베이스에 저장 및 관리하고 네트워크를 통해 빈번하게 전송되는 패스워드들과 관련된 잠재적인 보안문제들을 다뤄야만 한다 [1].

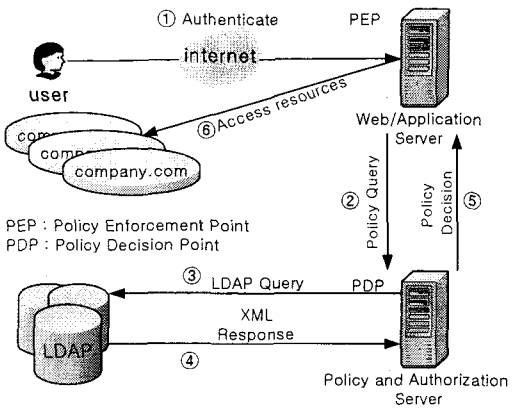
패스워드 관리의 어려움과 잠재적인 보안문제의 해결을 위해서는 단일 인증 후 신뢰관계가 형성된 도메인간의 사이트 접근을 용이하게 하는데 적합한 표준인 SAML(Security Assertion Markup Language)인증의 적용이 대안이 될 수 있다. 따라서 본 논문에서는 Single Sign On모델 구현을 위한 SAML인증 API를 구현하였다.

2. 관련연구

2.1 Single Sign On Model의 연구

SSO(Single Sign-on)[2]이란, 단 한번의 로그인만으로 기업의 각종 업무 시스템이나 인터넷 서비스에 접속할 수 있게 해주는 보안 응용 솔루션으로, 최근 각 기업들의 인트라넷 시스템과 웹 서비스가 대폭 확장됨에 따라 SSO도 급속히 확대되고 있다. SSO를 도입하면 각각의 시스템마다의 인증 절차를 밟지 않고도 사용자에 부여된 1개의 계정만으로 다양한 시스템에 접근할 수 있어 사용자 편의가 대폭 높아지고, 관리자 입장에서도 인증정책의 변경이나 권한 설정이 수월해져 관리비용 및 수고를 크게 덜 수 있다 [2]. 그러나 기존의 SSO의 개념을 적용해서 구현된 솔루션들은 대부분의 시스템이 host시스템, client/server시스템 등 서로 다른 시스템에 존재하므로 완전한 단일 SSO를 구축하기가 힘들다 [3].

단일 SSO를 구축하기 위한 방법으로 SAML(Security Assertion Markup Language)을 적용한 SSO모델을 제시한다. 아래의 [그림 1]은 SSO 모델의 개념을 나타낸다. 모든 인증과 권한에 대한 질의와 응답을 XML기반으로 처리하게 되므로 다른 시스템과의 상호운용성면에서도 커다란 장점을 가질 수 있다.



[그림 1]Single Sign On 모델의 개념도

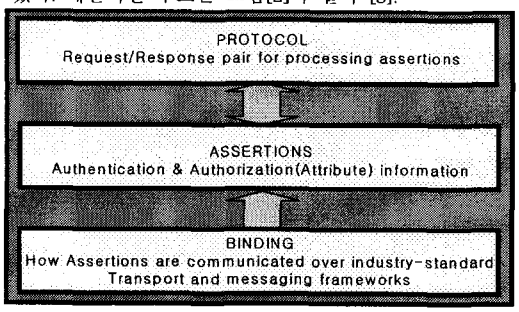
Single Sign On의 개념에 대한 설명은 다음과 같다.

① User는 Web/Application server에 로그인을 한 후 필요한 자원에 접근을 시도한다.
② Web/Application Server는 user에 대한 policy를 Policy and Authentication server에 질의한다.
③ Policy and Authentication server는 LDAP에 해당 user의 policy에 대해 질의를 한다.
④ LDAP에서는 Policy and Authorization server에 질의에 대한 결과를 XML기반으로 회신한다.
⑤ Policy and Authorization server는 user의 Authorization을 결정해서 Web/Application server에 회신한다.
⑥ Web/Application sever는 user에게 부여된 권한에 따라 자원에 접근한다.

2.2 SAML(Security Assertion Markup Language)

SAML[4]은 인터넷상에서의 자원 요청 자에 대한 인증, 승인, 그리고 속성확인 등을 수행하는 역할을 하며 이는 XML 기반의 다른 보안 기술들(XML Signature, XML Encryption, XKMS, XACML 등)과 통합되어 전체 보안 시스템을 구성하는 일부 요소로서 기능을 가진다.

SAML 명세는 *Assertion, Protocol, Binding*으로 구성되어 있다. 개괄적인 구조는 그림[2]와 같다 [5].



[그림2] SAML 구조

▲ Assertion

Assertion은 인증 및 승인 정보를 포함하는 XML 기반 구조를 가진다. Assertion 스키마의 구성은 [그림 3]과 같다 [4].

```
<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

Assertion 엘리먼트는 하위에 여러 개의 엘리먼트들과 속성들을 갖는다. Conditions엘리먼트는 Assertion의 유효성을 검증한다. Advice엘리먼트는 Assertion 발행자가 제공하는 부가적인 정보를 포함한다. Statement와 SubjectStatement엘리먼트는 assertion기반의 다른 애플리케이션이 SAML assertion framework를 재 사용할 수 있도록 하는 확장 지점의 역할을 한다.

AuthenticationStatement엘리먼트는 인증기관에서 발행한 인증 요청 자에 대한 성공적인 인증과정 수행을 보증한다. AttributeStatement엘리먼트는 인증 Assertion과 속성관리기관에서 발행하며 요청 자에 대한 자격을 확인한다. AuthorizationDecisionStatement엘리먼트는 승인기관에서 발행하는 인증된 요청 자가 요청한 자원에 대해 접근 허용 여부를 결정해 그 결과로서 발행하게 된다. Signature엘리먼트는 Assertion의 인증을 위해 XML 전자서명을 적용한다. 각각의 Assertion 발행 기관은 한 곳에 위치할 수 있다. 그림[4]는 AuthenticationStatement를 포함하는 Assertion을 생성한 결과이다.

```
<saml:Assertion AssertionID="00cda300-0d5e-8521-83c5-c2d9f6847b91"
                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

AuthenticationMethod속성의 값 password는 subject를 인증하는 방법을 나타낸다. subject를 인증하는 방법으로는 password방식 외에도 Kerberos, X.509 Public Key, XMKS Public Key, 그리고 XML Digital Signature 방식이 사용 될 수 있고 기타 다른 인증방식들도 사용할 수 있다.

AuthenticationInstant속성의 값은 subject를 인증한 시간을 나타내며 UTC데이터 포맷으로 인코딩 된 값이다. NameIdentifier엘리먼트는 subject의 이름과 security domain으로 subject를 식별하는 역할을 한다. NameQualifier속성은 충돌 없이 서로 다른 사용자 스토어로부터 이름을 통합할 방법을 제공한다. NameQualifier 속성은 생략이 가능하다 *[4]*.

▲ Protocol

SAML 프로토콜은 XML 기반의 메시지 형태로서 요청 및 응답의 쌍으로 구성되어 각 Assertion에 대한 전송을 담당한다. 일반적으로 Assertion은 SAML 프로토콜의 응답을 통해 얻어진다 [6].

▲ Binding

*"binding"*은 SAML *request/response*메시지를 표준 통신 프로토콜로 교환하는 매핑으로 SAML request/response 프로토콜 전송 메커니즘으로서 SOAP을 사용한다 [6].

2.3 SOAP(Simple Object Access Protocol)

SOAP은 클라이언트의 작업요청과 시스템의 응답을 XML문자열로 구성하고 전송 프로토콜로는 HTTP를 사용한다. SOAP은 액세스 요구나 리턴 되는 결과 값으로 XML을 사용함으로써 특정 포맷의 제약이 없고, 유연성 높은 범용의 액세스 기능을 제공하고 있다. XML에서는 데이터와 함께 데이터 명, 데이터 속성을 나타내는 태그도 동시에 포함할 수가 있기 때문에 단순한 수치형이나 문자형뿐만 아니라 배열과 같은 반복형의 데이터나 복잡한 구조를 표현할 수 있다 [7].

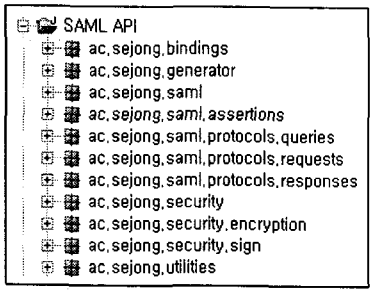
3. SAML 인증 API의 구현

사용자의 최초 인증 시 인증 Assertion에 대한 참고는 SAML artifact(8-byte base64 string)로 생성된다 [8]. artifact는 사용자가 사용자를 인증한 도메인과 신뢰관계가 형성된 다른 도메인의 destination 사이트에 접근 시 사용자에게 재 인증절차를 생략할 수 있게 해주는 패스포트의 역할을 한다.

사용자가 source 사이트에 접근 시 사용자를 인증하는 방식과 사용자 정보를 request 프로토콜에 명시하면 명시된 수단과 정보에 상응하는 사용자의 인증정보(Assertion)를 response 프로토콜에 포함시켜 되돌려준다. 성공적인 response는 인증 statement를 포함하는 assertion의 형태이다 [6].

3.1 패키지의 구성

구현된 SAML인증 API패키지의 구조는 아래의 [그림5]와 같다.



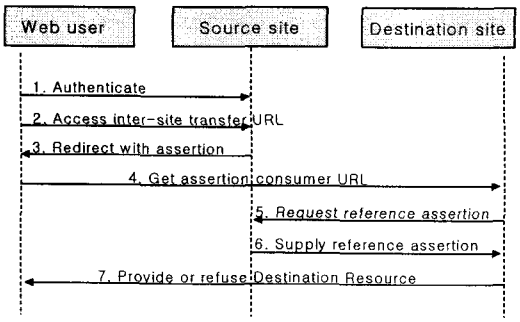
[그림 5] SAML인증 API 패키지 구조

패키지의 분류는 Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)명세를 바탕으로 했으며 기본적으로 크게 assertion, protocol, binding패키지로 나누고 부가적으로 generator, utilities, security패키지를 생성했다. 각 패키지의 역할은 다음과 같다.

assertion패키지는 인증과 권한(속성)정보를 처리한다. protocol패키지는 assertion의 처리를 위한 SAML response/request메시지 쌍을 처리한다. binding패키지는 assertion을 전송하는 메시징 프레임워크를 포함한다. security패키지는 assertion에 전자서명과 암호화처리를 한다. utilities패키지는 UUID, UTC데이터 포맷, artifact등을 생성하는 역할을 한다. generator패키지는 실제적으로 XML기반의 SAML request/response메시지를 생성하는 역할을 한다.

3.2 SAML 인증과 Assertion의 생성 시나리오

[그림 6]은 사용자의 인증 후 request문서의 생성, assertion의 생성, artifact의 생성, response문서의 생성 과정을 통한 Single Sign On(pull model)의 적용을 도식화한 것이다 [6].



[그림 6]인증과 assertion의 생성과정

인증과 assertion의 생성과정은 다음과 같다.

1. 사용자는 source사이트에 인증과정을 수행한다.
2. 사용자는 목적지 사이트의 링크를 클릭 한다.
3. source사이트는 목적지 웹사이트에 사용자의 인증 식별 참조(artifact)를 전달하고 목적지 웹사이트로 이동시킨다.
4. 웹 사용자는 목적지 사이트의 자원을 요청하고 인증 참조를 전달한다.
5. 목적지 웹사이트는 인증 Assertion을 source 웹사이트에 요청하면서 인증 참조(artifact)를 전달한다.
6. source 웹사이트는 Assertion을 전달한다.
7. 목적지 웹사이트는 Assertion을 분석해 웹 사용자에게 자원을 제공하거나 거부한다.

[그림 7]은 과정2번을 수행 시 generator패키지 내의 RequestGenerator 클래스를 이용해서 인증을 요구하는 사용자의 정보(도메인명, 사용자명, 그리고 인증 수단)를 기술한 SAML request메시지를 생성한 결과이다.

```
<samlp:Request IssueInstant="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: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/ams/namespace/common"
          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" />
  </samlp:AttributeQuery>
</samlp:Request>
```

[그림 7]SAML request메시지의 생성

[그림 8]은 SAML request메시지에 대해서 security.sign 패키지 내의 signature 클래스를 이용해서 서명한 결과이다. 서명하는 방식은 Enveloping과 Enveloped 그리고 Detached 방식을 선택적으로 사용할 수 있다. 아래의 서명방식은 Enveloped 서명방식으로 서명한 결과이다.

```
<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>xFXrupYsqtRgnk9wXi7znAenuew=</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 8]Enveloped방식으로 서명된 SAML request메시지

[그림 9]는 과정7번을 수행 시 generator패키지 내의 ResponseGenerator 클래스를 이용해서 SAML request메시지에서 요구하는 인증에 대한 인증정보(Assertion)를 포함하는 SAML response메시지를 생성한 결과이다.

```
<samlp:Response InResponseTo="a207b-a0-aaa4-11d6-9e6d-75a01a1d3688"
                IssueInstant="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>http://namespace1</samlp:StatusDetail>
  </saml:Status>
  <saml:Assertion AssertionID="00cda300-0d5e-8521-83c5-c2d9f6847b91"
                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>jijeong</saml:NameIdentifier>
      </saml:Subject>
    </saml:AuthenticationStatement>
  </saml:Assertion>
</samlp:Response>
```

[그림 9]SAML response메시지의 생성

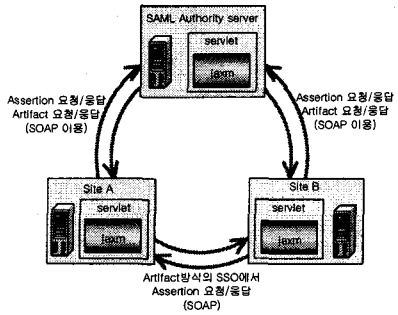
[그림 10]은 SAML response메시지에 대해서 security.sign패키지 내의 signature 클래스를 이용해서 서명한 결과이다. 아래의 서명방식은 Enveloped 서명방식으로 서명한 결과이다.

```
<samlp:Response...>
  <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>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:X509SerialNumber>1045440891</ds:X509SerialNumber>
      </ds:X509IssuerSerial>
      <ds:X509SubjectName>...</ds:X509SubjectName>
      <ds:X509Certificate>...Xs+69s=...</ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
  <saml:Status>
    ...
  </saml:Status>
  <saml:Assertion...>
    <saml:AuthenticationStatement...>
      <saml:Subject>
        ...
      </saml:Subject>
    </saml:AuthenticationStatement>
  </saml:Assertion>
</samlp:Response>
```

[그림 10]Enveloped방식으로 서명된 SAML response메시지

3.3 SAML request/response메시지의 전송

SAML request/response 메시지는 binding패키지내의 Mapper클래스를 사용하여 SOAP-over-HTTP 프로토콜에 payload한 후 사용자의 인증절차를 처리하는 source 사이트와 인증정보를 요구하는 destination 사이트간의 전송이 이루어진다. [그림 11]은 메시지 전송의 개념을 나타낸다.



[그림 11]SOAP을 이용한 binding 개념

사이트간의 Assertion 및 Artifact에 대한 요청 및 응답의 처리는 servlet엔진에서 jaxm API를 통해 이루어진다.

4. 결론 및 향후 연구방향

앞으로 인터넷 전자상거래 및 인트라넷에서의 사용자 인증에 관한 메커니즘으로 SAML인증을 적용한 Single Sign On 모델의 구현이 보편화 될 것으로 전망된다. Single Sign On 모델에는 인증 서비스, B2B Transaction, 최종사용자를 위한 세션 기술을 포함할 것이다. 이것은 single sign-on 기술로 서로 다른 웹사이트를 접근할 때 인증이 유지되는 기술이다. 이러한 기술뿐만 아니라, SAML은 현재 다양한 부분에서 활발한 활동을 하고 있으며 빠른 속도로 더 나은 기술의 개발이 진행될 것으로 예상된다. 따라서 본 논문에서는 SAML인증을 적용한 Single Sign On모델을 구현했다. 향후 연구에는 기존의 IPSec이나 SSL등을 통해 일괄적으로 암호화를 수행하는 방법에서 벗어나 특정 데이터만을 선별적으로 암호화하고 다양한 암호화 알고리즘을 적용할 수 있도록 개선할 것이다.

참고 문헌

[1] Netscape communications Corporation 1997 "Single Sign-On Deployment Guide"

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

[3] http://www.gate4india.com/about/infra_SSO.htm

[4] Security Assertion Markup Language (SAML) http://www.oasis-open.org/committees/security/

[5] Netegrity, Inc "Security Assertions Markup Language(SAML)" The standard XML framework for secure information exchange

[6] Jason Rouault "Introduction to SAML An XML based Security Assertion Markup Language"

[7]Simple Object Access Protocol(SOAP) http://www.w3.org/TR/SOAP/

[8]Dournaee - "XML Security"