An XML-Based Security Architecture for Integrating Single Sign-On and Rule-Based Access Control in Mobile and Ubiquitous Web Environments

Jongil Jeong, Dongil Shin, and Dongkyoo Shin*

Department of Computer Science and Engineering, Sejong University
98 Kunja-Dong, Kwangjin-Ku, Seoul 143-747, Korea
jijeong@gce.sejong.ac.kr, {dshin, shindk}@sejong.ac.kr

Abstract. Since mobile and Web applications are integrated, the number of services, a typical mobile user can now access, has greatly increased. With a variety of services, a user will be frequently asked to provide his security information to a system. This iterative request is one critical problem which can cause frequent transmission of user's security information. Another serious problem is how an administrator controls access request of internal users who were authenticated. In order to establish effective security scheme for integrated environments, Single Sign-On and access control also need to be integrated. In this paper, we propose an XML-based architecture integrating authentication and access control policy in integrated environment to be extended to ubiquitous environment. To provide flexibility, extensibility, and interoperability between environments to be integrated, we have implemented an architecture based on SAML and XACML, which are standardized specifications. By specifying security policies in XML schema and exchanging security information according to that schema, the proposed architecture offers the opportunities to build standardized schemes for authentication and authorization. Additionally, the proposed architecture makes it possible to establish a fine-grained access control scheme by specifying the XML element unit as a target to be protected.

Keywords: single sign-on, SAML, access control, RBAC, XACML, mobile device.

1 Introduction

Since mobile and Web applications are integrated, the number of services a typical mobile user can now access has greatly increased. As a result, target services that a user wants to access become various. However, the user will be frequently asked to provide his security information to systems which manage target services. In relation to the integration of different environments, another serious problem is how an

[*] Correspondence author

This study was supported by a grant of the Korea Health 21 R&D Project, Ministry of Health & Welfare, Republic of Korea. (0412-MI01-0416-0002).

R. Meersman, Z. Tari, P. Herrero et al. (Eds.): OTM Workshops 2006, LNCS 4278, pp. 1357 – 1366, 2006.

administrator controls access request of internal users who were authenticated. Recently, the Open Web Application Security Project (OWASP) emphasized the importance of these issues by declaring 'Broken Access Control' as one of the top ten most critical Web application security vulnerabilities [1]. The same situation can be extended to ubiquitous service environments that consist of different kinds of personal equipment, wireless sensors, gateways, Web Servers, and services [2].

A key to the problem of the iterative request is to reduce the number of user authentication. Single Sign-On (SSO) can give the key. As a security feature, SSO allows a user to log into many different services offered by the distributed systems while the user needs to authenticate identification only once, and always in the same way [3]. In relation to the problem of a management of internal users' access request, Access control can be the key to solve it. Access control either permits or denies user access requests by checking whether the user has permission to access target resources. Therefore, it is strongly recommended that SSO and access control also need to be integrated for establishing effective security scheme in integrated environments. In this paper, we propose an XML-based architecture integrating authentication and access control policy to suggest actual guidelines for constructing secure Web Systems in a ubiquitous environment. The proposed architecture is based on two standard specifications which are ratified by Organization for Advancement of Structured Information Standards (OASIS) [4]: Security Assertion Markup Language (SAML) [5] and eXtensible Access Control Markup Language (XACML) [6].

This paper is composed of five sections. (Mention Section 1 also) Section 2 includes an overview of single sign-on, SAML, access control, and XACML. In Section 3, we introduce the principle of least privilege and define rules for testing RBAC policy and then express those rules in XACML. In Section 4, we propose architecture for integrating user authentication and access control for mobile and ubiquitous Web Services environments and describe the structure of Java-based SAML and XACML libraries that we have developed. Finally we conclude our discussion in Section 5.

2 Background

For successful integration between different environments, it is necessary to understand technologies in order to solve the complexity of user authentication and the vulnerable access control mechanism mentioned in the previous section. This section introduces single sign-on, access control, and the tendency of standardization for related technologies, such as SAML and XACML.

2.1 Single Sign-On

The basic idea of single sign-on (SSO) is to shift the complexity of the security architecture to the SSO service and release other parts of the system from certain security obligations.

The SSO service acts as the wrapper around the existing security infrastructure that exports various security features like authentication and authorization [7]. To support single sign-on, the system collects all the identification and user credentials at the time of primary sign-on. This information is then used by SSO Services within the primary domain to support the authentication of the user to each of the secondary domains with which the user may interact.

2.2 Role Based Access Control

Role-Based Access Control (RBAC) makes it simple to allocate and remove privileges for large numbers of users. To simplify such management of privileges for large numbers of users, the RBAC scheme is based on the user's roles [8]. These roles reflect organization structure. For instance, a user with a role of employee can access certain resources, and another user with a role of manager can access certain, perhaps very different, resources. The roles have the permission sets, not the users. A role can be created by inheriting one or more attributes from other roles like an object can inherit attributes from multiple objects in object-oriented programming.

2.3 The Tendency of Standardization for Related Technologies

To implement web-based single sign-on and access control by applying RBAC, OASIS has developed technologies related to each concept. At the same time, the organization has propelled the standardization of these technologies.

a. SAML (Security Assertion Markup Language)

SAML enables the exchange of authentication and authorization information about users, devices, or any identifiable entity called subjects. Using a subset of XML, SAML defines the request-response protocol by which systems accept or reject subjects based on assertions [5].

An assertion is a declaration of certain facts about a subject. SAML defines three types of assertions:

- Authentication: indicating that a subject was authenticated previously by some means (such as a password, hardware token, or X.509 public key).
- Authorization decision: indicating that a subject should be granted or denied resource access.
- Attribution: indicating that the subject is associated with attributes.

The existing SSO schemes, such as Kerberos, limit their applicable scope to a single security domain due to the lack of interoperability. SAML opens the possibility that SSO schemes can be passed between other domains via common XML schema.

b. Single Sign-On Browser/artifact Profile

SAML can be bound to multiple communication and transport protocols. It can be linked with Simple Object Access Protocol (SOAP) over HTTP [5]. SAML operates without cookies in a browser/artifact profile. Using browser/artifact, a SAML artifact is carried as part of a URL query string, as shown in Figure 1, where a SAML artifact is a pointer to an assertion.

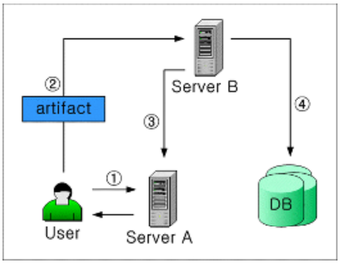


Fig. 1. A Browser/Artifact profile

The steps in Figure 1 are explained as follows.

(1) User of an authenticated browser on Server A requests access to a database on Server B. Server A generates a URL redirect, which contains a SAML artifact, to Server B.

(2) Browser redirects user to Server B, which receives an artifact pointing to the assertion on Server A.

(3) Server B sends artifact to Server A and gets a full assertion.

(4) Server B checks the assertion and either validates or rejects the user's request for access to the database.

c. XACML (eXtensible Access Control Markup Language)

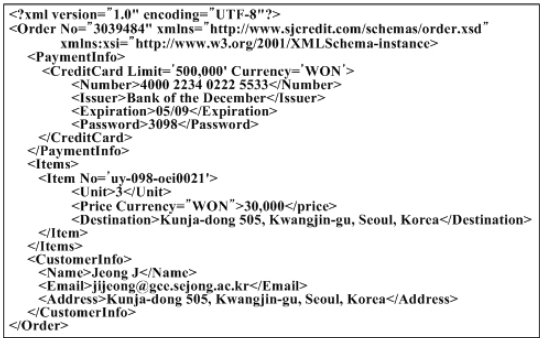
XACML is used in conjunction with SAML and supplements lacking access control policy in SAML. XACML can specify various targets, such as resource, an entire document, a partial document, or multiple documents. It can even specify an XML element as the target to be protected. This aspect makes it possible to implement fine-grained access control. The sequence of data-flow for SAML and XACML is as follows: If Web Services receives a SAML assertion once, sends it to SAML PDP (Policy Decision Point); then the PDP requests XACML PRP (Policy Retrieval Point) to check XACML policies. This route shows the decision making to determine if access request should be granted to certain resources, based on rule sets or policies defined by the provider. Once the policy is evaluated and then returns the true or false, an SAML authorization decision assertion is made by SAML PDP and then returns it to SAML PEP (Policy Enforcement Point). Finally, SAML PEP grants or denies the access request according to the authorization decision assertion [5], [6].

3 The Principle of Least Privilege and the Expression of the Principle Using XACML

One of the most important ways to secure resources is to minimize privileges [8]. A privilege is simply a permission to do something that not everyone is allowed to do. The principle of least privilege means that a user cannot have any privilege except the privileges that he or she can perform [9]. Through RBAC, enforced minimum privileges for general system users can be easily achieved. To minimize privileges, it is necessary to do the following [9]:

- Identify user functions.
- Determine the minimum set of privileges required to perform the specific function.
- Restrict the user to a domain having relevant privileges.

XACML provides an XML-based RBAC profile for expression of authorization policies to build them into Web applications. We propose a practical model for transcribing the RBAC profile into XACML using some examples. The *Order.xml* file consists of three elements. *PaymentInfo* is an element including settlement information. *Items* is an element including information for goods that a user wants to buy. *CustomerInfo* is an element including information for the transaction owner. By describing RBAC in XACML, we can focus on an access control unit in each element within a file. Thanks to it, we can achieve the fine-grained access control we want.



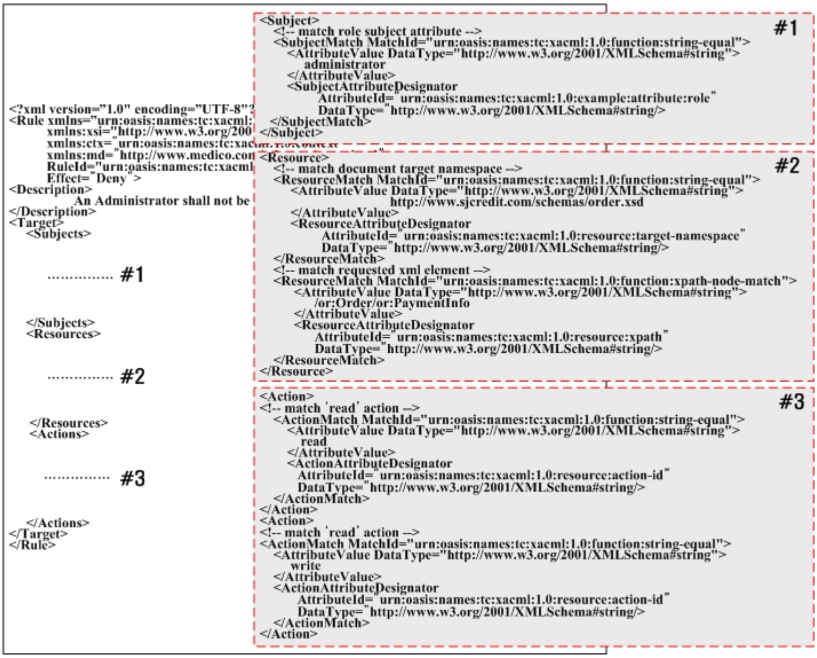
```
<?xml version="1.0" encoding="UTF-8"?>
<Order No="3039484" xmlns="http://www.sjcredit.com/schemas/order.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance>
   <PaymentInfo>
     <CreditCard Limit='500,000' Currency='WON'>
          <Number>4000 2234 0222 5533</Number>
          <Issuer>Bank of the December</Issuer>
          <Expiration>05/09</Expiration>
          <Password>3098</Password>
     </CreditCard>
   </PaymentInfo>
   <Items>
     <Item No='uy-098-oei0021'>
          <Unit>3</Unit>
          <Price Currency="WON">30,000</price>
          <Destination>Kunja-dong 505, Kwangjin-gu, Seoul, Korea</Destination>
     </Item>
   </Items>
   <CustomerInfo>
     <Name>Jeong J</Name>
     <Email>jijeong@gce.sejong.ac.kr</Email>
     <Address>Kunja-dong 505, Kwangjin-gu, Seoul, Korea</Address>
   </CustomerInfo>
</Order>
```

Fig. 2. Order.xml

The following are rules for access control against the *Order.xml* file.

- Rule 1: A person, identified by his credit card number and password may read any record for which he is the designated customer.
- Rule 2: An administrator shall not be permitted to read or write to the *PaymentInfo* element of the Order element.

The above rules will reflect the goal that a least privilege policy is pursuing. In a document including a transaction particular to a customer, the *PaymentInfo* element must be opened only to the transaction owner because it holds crucial information, such as a credit card number and password. Rule 2 is a rule that makes an administrator unable to 'read' and 'write' the *PaymentInfo* element except with his own permission.



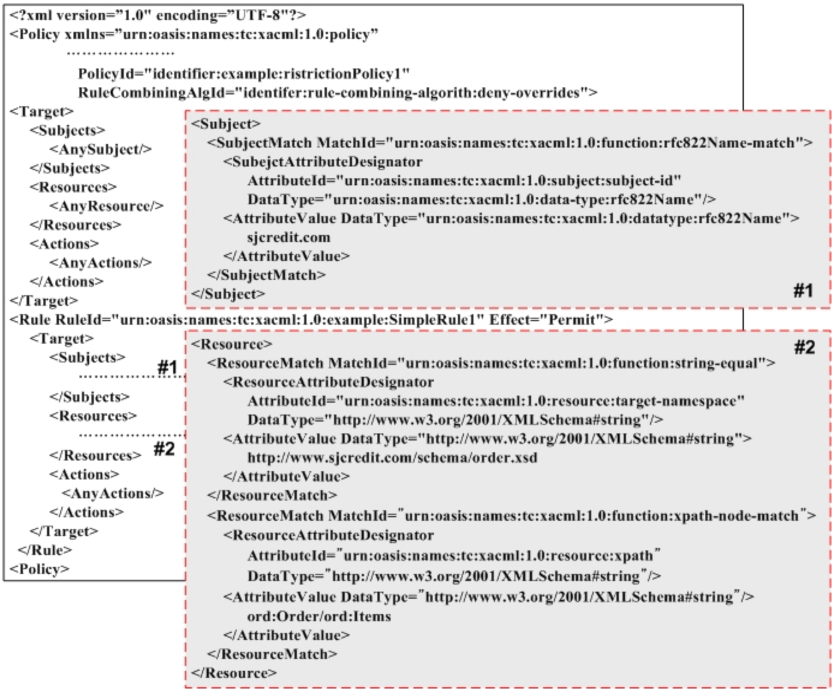
```
<?xml version="1.0" encoding="UTF-8"?
<Rule xmlns="urn:oasis:names:tc:xacml:
        xmlns:xsi="http://www.w3.org/200
        xmlns:ctx="urn:oasis:names:tc:xac
        xmlns:md="http://www.medico.con
        RuleId="urn:oasis:names:tc:xacml
        Effect="Deny">
<Description>
            An Administrator shall not be
</Description>
<Target>
   <Subjects>

        ............... #1

   </Subjects>
   <Resources>

        ............... #2

   </Resources>
   <Actions>

        ............... #3

   </Actions>
</Target>
</Rule>
```

```
<Subject>                                                                        #1
  <!-- match role subject attribute -->
  <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
       administrator
    </AttributeValue>
    <SubjectAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:example:attribute:role"
          DataType="http://www.w3.org/2001/XMLSchema#string/>
  </SubjectMatch>
</Subject>
```

```
<Resource>                                                                       #2
  <!-- match document target namespace -->
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    http://www.sjcredit.com/schemas/order.xsd
     </AttributeValue>
     <ResourceAttributeDesignator
           AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
           DataType="http://www.w3.org/2001/XMLSchema#string/>
  </ResourceMatch>
  <!-- match requested xml element -->
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         /or:Order/or:PaymentInfo
      </AttributeValue>
      <ResourceAttributeDesignator
         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
         DataType="http://www.w3.org/2001/XMLSchema#string/>
  </ResourceMatch>
</Resource>
```

```
<Action>                                                                         #3
<!-- match 'read' action -->
  <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      read
    </AttributeValue>
    <ActionAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string/>
  </ActionMatch>
</Action>
<Action>
<!-- match 'read' action -->
<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
     write
  </AttributeValue>
  <ActionAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string/>
  </ActionMatch>
</Action>
```

Fig. 3. Description of Rule 2 using XACML



```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
          ....................
            PolicyId="identifier:example:ristrictionPolicy1"
            RuleCombiningAlgId="identifer:rule-combining-algorith:deny-overrides">
<Target>
   <Subjects>
      <AnySubject/>
   </Subjects>
   <Resources>
      <AnyResource/>
   </Resources>
   <Actions>
      <AnyActions/>
   </Actions>
</Target>
<Rule RuleId="urn:oasis:names:tc:xacml:1.0:example:SimpleRule1" Effect="Permit">
   <Target>
      <Subjects>
          ................ #1
      </Subjects>
      <Resources>
          ................
      </Resources>  #2
      <Actions>
        <AnyActions/>
      </Actions>
   </Target>
 </Rule>
<Policy>
```

```
<Subject>
  <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
    <SubejctAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
        DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
    <AttributeValue DataType="urn:oasis:names:tc:xacml:1.0:datatype:rfc822Name">
        sjcredit.com
    </AttributeValue>
  </SubjectMatch>
</Subject>                                                                       #1
```

```
<Resource>                                                                       #2
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <ResourceAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        http://www.sjcredit.com/schema/order.xsd
    </AttributeValue>
  </ResourceMatch>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
    <ResourceAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"/>
        ord:Order/ord:Items
    </AttributeValue>
  </ResourceMatch>
</Resource>
```

Fig. 4. A rule restricting a user specific domain having relevant privileges

Figure 4 is a rule to meet the third requirement of least privilege. Rule 2 allows all users who have e-mail address ending with "sjcredit.com" to make out the *Items* element of *Order.xml*. Rule 1 is specifying that the user's e-mail address should correspond with the rfc822Name offered by the Rule. Rule 2 is specifying a resource that the relevant user can access. The subject has to be able to perform an 'action' for only the *Items* element of those elements corresponding with the target namespace within *Order.xml*.

4 Architecture for Integrating User Authentication and Access Control for Mobile and Ubiquitous Web Services Environments

In the transfer of the user's authentication information from a mobile environment to the authentication server of the wired service environment, the following considerations need to be taken into account [10]:

- The equipment, which can translate into an appropriate protocol to transfer the user's authentication information between mobile and wired service networks.
- Confidentiality and integrity, which should be guaranteed during the transfer of the user's authentication information.
- The framework aiming for a single sign-on implementation that transfers the user's authentication information should handle the user authentication mechanism defined in each domain.

One of the examples of a widely used framework, which connects mobile and wired service networks, is OMA's WAP (Wireless Application Protocol) gateway, also known as the WAP proxy [11]. The WAP gateway connects the mobile domain and the wired Internet and acts as a protocol gateway to encode and decode content.

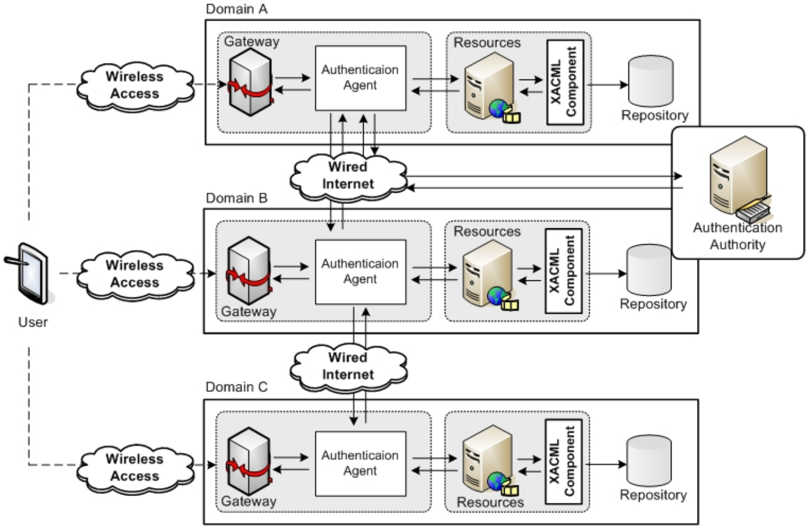


Fig. 5. Architecture for integrating user authentication and role-based access control

We propose an integrated architecture in which a mobile user offers his credential information to the wired service network to obtain user authentication and authorization and then access to another domain using this authentication and authorization, based on the SAML and XACML standard. Figure 5 shows the concept of this architecture. Each box in the diagram denotes an entity involved in the process. Figure 6 explains the messages between entities, applying a user's single sign-on and access control in three domains in which there is a mutual trust relationship.

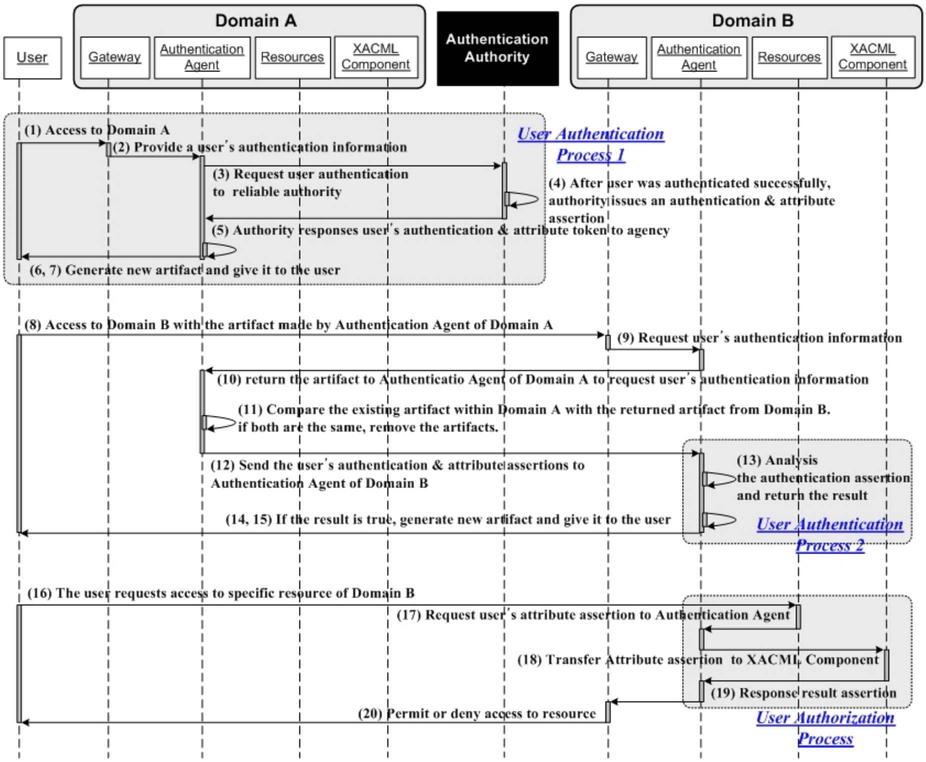
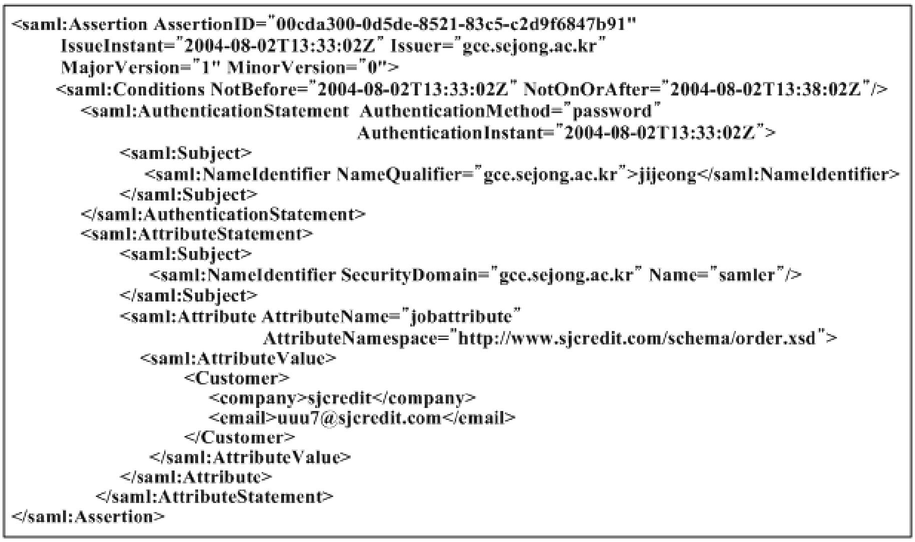

Domain A
User
Gateway
Authentication Agent
Resources
XACML Component
Authentication Authority
Domain B
Gateway
Authentication Agent
Resources
XACML Component
(1) Access to Domain A
(2) Provide a user's authentication information
User Authentication Process 1
(3) Request user authentication to reliable authority
(4) After user was authenticated successfully, authority issues an authentication & attribute assertion
(5) Authority responses user's authentication & attribute token to agency
(6, 7) Generate new artifact and give it to the user
(8) Access to Domain B with the artifact made by Authentication Agent of Domain A
(9) Request user's authentication information
(10) return the artifact to Authenticatio Agent of Domain A to request user's authentication information
(11) Compare the existing artifact within Domain A with the returned artifact from Domain B. if both are the same, remove the artifacts.
(12) Send the user's authentication & attribute assertions to Authentication Agent of Domain B
(13) Analysis the authentication assertion and return the result
(14, 15) If the result is true, generate new artifact and give it to the user
User Authentication Process 2
(16) The user requests access to specific resource of Domain B
(17) Request user's attribute assertion to Authentication Agent
(18) Transfer Attribute assertion to XACML Component
(19) Response result assertion
(20) Permit or deny access to resource
User Authorization Process

Fig. 6. Sequence diagram of the proposed Architecture

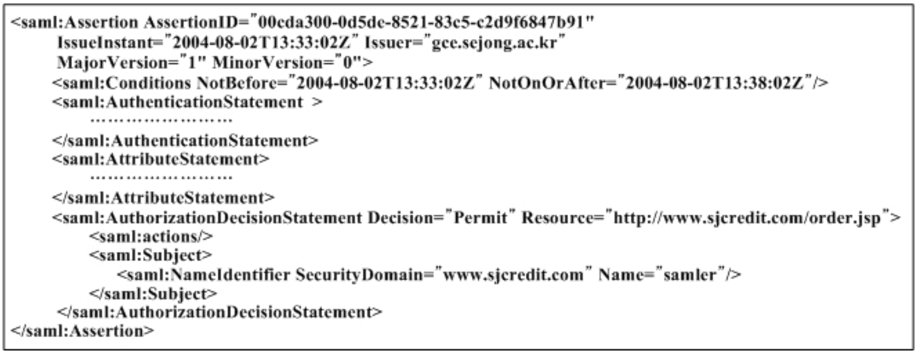
- *User Authentication Process 1* and *User* Authentication *Process 2* is authenticating a user. When the first authentication of a user is completed successfully, an artifact is generated by the *Authentication Agent* and assigned to the user. In the first process, a user must provide ID and password to the system; however, in the second process, the user may not need to provide it because in *Process 2*, when the user requests authentication to the *Authentication Agent* within *Domain B*, the *Authentication agent* of *Domain A* sends the artifact to the *Authentication Agent* of *Domain B* and turns it back around from *Domain B*. The *Authentication Agent* of *Domain A* verifies the artifact and sends user authentication information issued by the *Authentication Authority* to the *Authentication Agent* of *Domain B*.

- The *User Authorization Process* checks whether a user has permission to access specific resources. The *XACML Component* requests the user's attribute information for the *Authentication Agent* and further analysis. Then, the *XACML Component* determines whether the user's access request is permitted or denied.



```
<saml:Assertion AssertionID="00cda300-0d5de-8521-83c5-c2d9f6847b91"
      IssueInstant="2004-08-02T13:33:02Z" Issuer="gce.sejong.ac.kr"
      MajorVersion="1" MinorVersion="0">
     <saml:Conditions NotBefore="2004-08-02T13:33:02Z" NotOnOrAfter="2004-08-02T13:38:02Z"/>
        <saml:AuthenticationStatement  AuthenticationMethod="password"
                                       AuthenticationInstant="2004-08-02T13:33:02Z">
            <saml:Subject>
               <saml:NameIdentifier NameQualifier="gce.sejong.ac.kr">jijeong</saml:NameIdentifier>
            </saml:Subject>
        </saml:AuthenticationStatement>
        <saml:AttributeStatement>
            <saml:Subject>
               <saml:NameIdentifier SecurityDomain="gce.sejong.ac.kr" Name="samler"/>
            </saml:Subject>
            <saml:Attribute AttributeName="jobattribute"
                            AttributeNamespace="http://www.sjcredit.com/schema/order.xsd">
              <saml:AttributeValue>
                   <Customer>
                      <company>sjcredit</company>
                      <email>uuu7@sjcredit.com</email>
                   </Customer>
              </saml:AttributeValue>
            </saml:Attribute>
          </saml:AttributeStatement>
</saml:Assertion>
```

Fig. 7. Assertion with Authentication and Attribute Statement



```
<saml:Assertion AssertionID="00cda300-0d5de-8521-83c5-c2d9f6847b91"
      IssueInstant="2004-08-02T13:33:02Z" Issuer="gce.sejong.ac.kr"
      MajorVersion="1" MinorVersion="0">
     <saml:Conditions NotBefore="2004-08-02T13:33:02Z" NotOnOrAfter="2004-08-02T13:38:02Z"/>
     <saml:AuthenticationStatement >
         ........................
     </saml:AuthenticationStatement>
     <saml:AttributeStatement>
         ........................
     </saml:AttributeStatement>
     <saml:AuthorizationDecisionStatement Decision="Permit" Resource="http://www.sjcredit.com/order.jsp">
         <saml:actions/>
         <saml:Subject>
            <saml:NameIdentifier SecurityDomain="www.sjcredit.com" Name="samler"/>
         </saml:Subject>
      </saml:AuthorizationDecisionStatement>
</saml:Assertion>
```

Fig. 8. Assertion with Authentication, Attribute, and Authorization Decision Statement

Figure 7 is an assertion statement issued by the *SAML Authority* (refers to Step (4) of Figure 6). Figure 8 is an assertion statement issued by the *XACML Component* (refers to Step (19) of Figure 6). These messages were verified by a simulation where two domains were constructed with a mutual trust relationship and the SAML and XACML libraries, which were built from previous work [10].

5 Conclusion

There have been a number of research studies on secure integration of authentication and authorization between distributed systems; however, such integrated models limited their applicable scope to a single security domain because the research projects could not have the interoperability for exchanging security information with other security domains. To provide flexibility, extensibility, and interoperability between environments to be integrated, we have implemented an architecture based on SAML and XACML, which are standardized specifications. By specifying security policies in XML schema and exchanging security information according to that schema, the proposed architecture offers the opportunities to build standardized schemes for authentication and authorization. Additionally, the proposed architecture makes it possible to establish a fine-grained access control scheme by specifying the XML element unit as a target to be protected.

In our future research, we will analyze security threats that can occur in the proposed architecture, and prepare countermeasures against them.

References

1. OWASP (Open Web Application Security Project): http://www.owasp.org/document/topten.html
2. He Q, Khosla P, Su Z.: A Practical Study on Security of Agent-Based Ubiquitous Computing, in Proc. AAMAS'02 Deception, Fraud, and Trust in Agent Societies workshop, 2002.
3. Parker, T.A.: Single sign-on systems-the technologies and the products, European Convention on Security and Detection, 16-18 May (1995) 151-155
4. http://www.open-oasis.org
5. Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1: http://www.oasis-open.org/committees/security/
6. eXtensible Access Control Markup Language (XACML) Version 1.0: http://www.oasis-open.org/committees/xacml/repository/
7. Pfitzmann, B., Waidner, B.: Token-based web Single Signon with Enabled Clients, IBM Research Report RZ 3458 (#93844), November (2002)
8. Barkley J, Cincotta A, Ferraiolo D, Gavrila S, and Kuhn R.: Role based access for the world wide web, In National Information Systems Security Conference, October 1997
9. http://csrc.nist.gov/rbac/NIST-ITL-RBAC-bulletin.html
10. Ferraiolo D, Barkley J, and Kuhn R.: A Role Based Access Control Model and Reference Implementation within a Corporate Intranet, ACM Transactions on Information Systems Security, Volume 1, Number 2, 1999
11. Jeong J, Shin D, Shin D, Oh H.: A Study on XML-based Single Sign-On System Supporting Mobile and Ubiquitous Service Environments, Lecture Notes in Computer Science 3207, (2004).
12. WAPWhite_Paper1.pdf: http://www.wapforum.org/what/WAPWhite_Paper1.pdf