A Safe Delegation Method for Web Services in Pervasive Computing Environments

Hyun Sik Hwang[1] , Hyuk Jin Ko[1], Kyu Il Kim[1], Hae Kyung Lee[2], Woojun Kang[3], Ung Mo Kim[1]

Department of Computer Science and Engineering, Sungkyunkwan University[1]

Department of Computer Game&Information, Songdam College[2]

College of Management and Information Technology, Korea Christian University[3]

Summary
Web Services are the new building block of today's Internet, and provides interoperability among heterogeneous distributed systems. In this environment, security is one of the most critical issues. An attacker may expose user privacy and service information without authentication. Furthermore, in the pervasive computing environment, the users of web services must temporarily delegate some or all of their rights to an agent to perform actions on their behalf. This results in the exposure of user's privacy information by agents. We present a delegation model to support secure web services in pervasive computing environments. In order to support privacy protection, service confidentiality, and assertion integrity, encryption and a digital signature mechanism is deployed. We build web service management server based on XACML, in order to manage services and policies of web service providers. SAML is extended, in order to declare delegation assertions transferred to web service providers, by delegation among agents.

Key words:
XACML, SAML, Agent

1. Introduction

A web service is designed for easy integration with Business to Business (B2B) systems, and also to support Business to Consumer (B2C) systems, using public standards such as Simple Object Access Protocol (SOAP) [19], Web Service Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI), which are based on XML. Web service providers are able to register their services to UDDI registry [8] using WSDL. Web service consumers are able to retrieve web services from the UDDI registry, according to their retrieval conditions, and subsequently perform the services. In pervasive computing environments, users use agents, in order to initiate web services. Agents transfer user's credentials to web service providers which then perform web services on their behalf. Furthermore, in order to efficiently support web services, agents delegate their various rights to others. Users are required to disclose their privacy information by agents in order to perform web services. This results that user's privacy information is disclosed to the outside. In order to support privacy protection, service confidentiality and assertion integrity, XML-based encryption and a digital signature mechanism [10] [20] is adopted. Security Assertion Markup Language (SAML) is exploited in order to declare delegation assertions transferred to web service providers using a delegation mechanism among agents. An eXtensible Access Control Markup Language (XACML) is adopted in order to manage the services and policies supported by web service providers. OASIS standards SAML and XACML contain Single Sign-On (SSO) [15] [16], Trust Management, Authentication, and Authorization. The OASIS specification provides this information is greater detail [1] [2].

Manuscript received January 5, 2007
Manuscript revised January 25, 2007

2. Backgrounds

2.1 SAML

SAML (Security Assertion Markup Language) developed by the Security Services Technical Committee (SSTC) of OASIS, is a XML-based framework for exchanging user authentication, authorization and attribute information. One of the major goals for SAML is single sign-on (SSO), the ability of a user to authenticate in one domain and use resources in other domains without re-authenticating. SAML enables web SSO through the exchange of an authentication assertion from first site to second site. SAML specifications define the below components [14]:

- *Core* (the syntax and semantics for XML encoding assertion and request/response protocols)
- *Bindings* (protocol binding for the use of request / response messages)
- *Profiles* (for embedding and extracting SAML assertions in a framework or protocol)
- *Authentication context* (the syntax for the definition of authentication context and initial list)

• *Conformance*
• *Metadata*
• *Security Consideration*
In order to achieve web single sign-on, SAML must use request/response protocols based on assertions. SAML defines three different types of assertions:

• *Authentication* (the subject has been authenticated by some means at a given time)
• *AuthorizationDecision* (response to an access request, whether the access has been granted or denied)
• *Attribute* (the subject is associated with the given attributes and values)

SAML can be bound to multiple communication and transport protocol. It can be in conjunction with Simple Object Access Protocol (SOAP) over HTTP [1].
SAML lacks delegation capabilities. However, SAML provides inherent extensibility to create our delegation assertion.

2.2 XACML

XACML (eXtensible Access Control Markup Language) [17] is the result of a recent OASIS standardization effort proposing an XML-based language to express and interchange access control policies. XACML was designed to accommodate most system needs. At its core [2], XACML defines the syntax for policy language and the semantics for processing those polices. There is also a request and response format to query the policy system and semantics for determining applicability of policies to requests. The request and response formats represent a standard interface between a Policy Enforcement Point (PEP) that issues requests and handle responses and a Policy Decision Point (PDP) presents standard behavior when processing policy. Figure 1 illustrates XACML overview. This is based on policy framework definitions used in the IETF [13].
XACML can represent the functionalities of most policy representation mechanisms and has standard extension points for defining new function, data types, policy combining logic, and so on. The major functionalities offered by XACML can be summarized as follows [11].

• *Combination policy support*. XACML provides a method for combining policies independently specified. Different entities can then define their policies on the same resource. When an access request on that resource is submitted, the system has to take into consideration all these policies. XACML defines three elements for the specification of access control policies: Rule, Policy, and PolicySet. The Rule element corresponds to the traditional concept of authorization: it defines who can access to which resource and under which conditions. The Policy element consists of a set of rules and specifies how to combine the results of their evaluation. Finally, the PolicySet element contains a set of Policy or PolicySet.

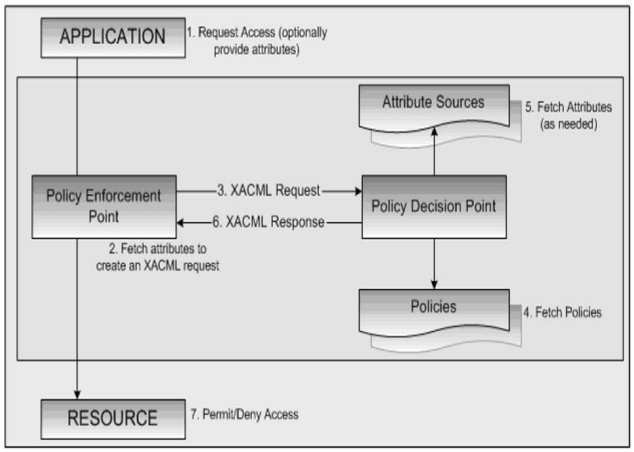

Fig. 1 XACML Overview

• *Combining algorithms support*. Since both a Policy and a PolicySet element can contain multiple policies or rules, each of which can evaluate to different access control decisions, XACML needs to define a method for reconciling such decisions. XACML supports different combining algorithms, each representing a way of combining multiple decisions into a single decision. To this purpose, XACML defines two attributes, namely RuleCombiningAlgId, and PolicyCombiningAlgId. The first attribute indicates a method for combining the individual results obtained from the evaluation of a set of rules. The second attribute indicates a method for combining the individual results of evaluation of a set of policies.

• *Attribute support*. XACML supports the definition of policies based on properties (attributes) associated with subjects and resources other than their identities. This allows the definition of powerful policies based on generic properties associated with subjects (e.g. name, address, and occupation) and resources. To this purpose, XACML provides two elements, namely SubjectAttributeDesignator and ResourceAttributeDesignator that together with SubjectMatch and ResourceMatch elements allow identifying a particular subject and resource attribute, respectively.

• *Operators support*. XACML includes some built-in operators for comparing attribute values and provides a method of adding non-standard functions.

• *Multiple subjects*. XACML allows the definition of more than one subject relevant to a decision request.

• *Policy distribution support*. Policies can be defined by different parties and enforced at different enforcement points. Also, XACML allows one policy to contain or refer to another.

• *Implementation independency*. XACML provides an abstraction-layer that isolates the policy writer from the implementation details. This means that different implementations should operate in a consistent way, regardless of the implementation itself. As we will see later on, XACML defines a canonical form for the request and response, called XACML context.

• *Obligations support*. XACML provides a method for specifying actions, called obligations that must be fulfilled in conjunction with the policy enforcement. The element that provides this feature is the Obligation element.

3. Related Works

Navarro, et al [3] describe SAML assertion for constrained delegation. In order to support delegation, they extend SubjectStatement to SubjectDelegationStatement which is not supported by the SAML1.1/2.0 Assertion specifications. J. Wang and D. Del Vecchio [9] also extend SAML to support delegation. They propose verification rules for delegation assertions relying on a WS-Security X.509 Signature [4]. However, they must verify an ordered delegation chain to prove a trust relationship between a delegater and a web service portal when the indirect delegation is applied. Y. J. Hu [6] presents a method for creating agent systems, which is based on Public Key Infrastructure (PKI). He considers various kind of delegation such as chain-ruled delegation, threshold delegation and conditional delegation. However, this approach does not consider mobile agent systems or heterogeneous multi-agent systems. Navarro, et al [7] present an access control framework for a mobile agent platform, which is based on RBAC. In this framework, the Authorization Manager (AM) is exploited, to manage the delegation of authorizations, and issuing of authorization certificates. However, they only consider agents complying with its local policy. In our work, we extend the AttributeStatement supported by SAML 1.1/2.0 specification to the AttributeStatement containing delegation information. We exploit the authentication/delegation authorities to omit the step of verifying an ordered delegation chain. Our model only requires a delegation assertion to verify a trust relationship. An important contribution of our work is applicable to delegation for web service in the pervasive computing environments using all kinds of agents.

4. Delegation Framework

4.1 Overview of Delegation Framework

Figure 2 describes a delegation model to support secure web services in pervasive computing environments using agents. We assume that users are able to have two kinds of roles. The first is a role as the consumer of web service management server, the other is a role (least role) as the consumer of service supported by a web service provider. The least role granted by a web service management server is shared with web service providers corresponding to user's retrieval conditions.
Our model enables users to manage delegation of their own rights. Agents are controlled by the use of services, based on delegated rights by the users. In order to manage delegation, users are assisted by the delegation authority, which authorizes delegation from one delegating agent to other agents [5].

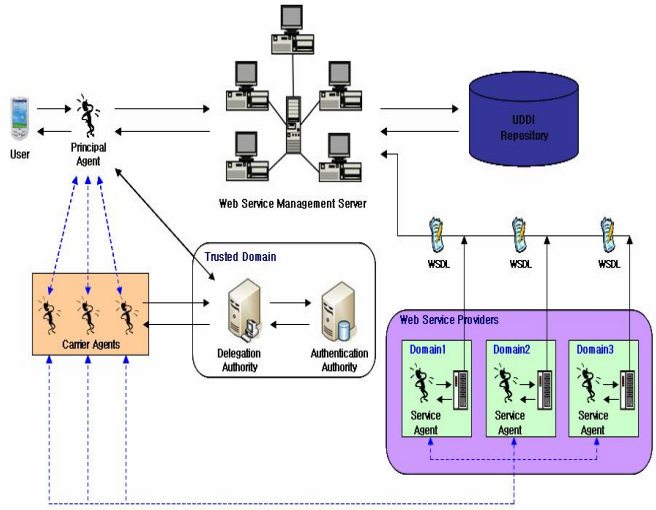

Fig. 2 Overview of Delegation Framework

The authentication authority assists this process by authentication of user and agent's identities. In the Figure 2, a solid arrow indicates general request/response and a dotted arrow indicates delegation request/response. The following presents basic components used in our delegation model. The *web service management server (WSMS)* is a mediator between users and web service providers. It is a system that manages web services and registered policies of web service providers. In addition, it processes principal requests. *Principal (P)* is a user that delegates his/her rights to agents in order to perform web services. *Principal agent (PA)* is software that interacts

with the *P* and other agents on behalf of the *P*. *PA* can be controlled by a *P*, and can undertake actions for the principal. *Carrier agent (CA)* and *service agent (SA)* is similar to *PA*. However, *CA* is only interacts with other agents and it is able to delegate rights based on the *P*'s delegation consent. *SA* is a web service provider's agent verifying whether delegation assertion is valid. *SA* also processes *P*'s service request. *Authentication authority (AA)* is authority to authenticate *P*s or agents. *AA* issues an authentication assertion, indicating whether a *P* or agent is valid. *Delegation authority (DA)* is authority to issue a delegation assertion, guaranteeing *DA* grants the delegation right of transferring the *P*'s rights from a *P* or an agent, to agents. Prior to issuing delegation assertion, the *DA* must acquire the agent's authentication assertion by *AA*, and a *P*'s delegation consent.

4.2 Specification of Delegation Assertion

A SAML specification defines three different types of assertions [1]. However, the schema for delegation is not directly defined by the SAML specifications. Fortunately, by exploiting SAML schema extensibility, we can create our delegation assertion.

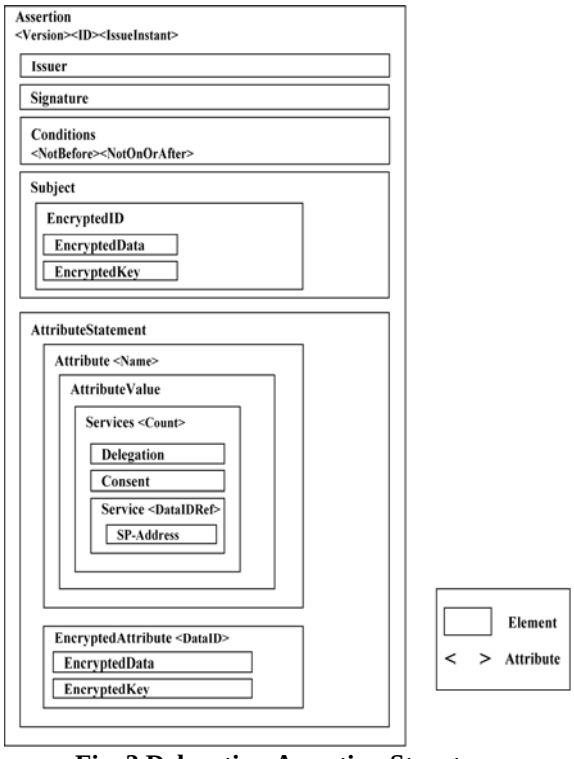

Fig. 3 Delegation Assertion Structure

Our model extends the attribute statement in order to encode information such as delegation, subject, and input parameters of WSDL. Delegation assertion indicates that the *P*/agent is capable of delegating rights. The web service provider is able to omit the step of authentication, because delegation assertion is issued after authentication assertion is verified.

A delegation assertion structure is built by extending attribute statement with additional elements. The elements and attributes of delegation assertion are described in Figure 3. *Issuer* element is the issuer of delegation assertion, the *DA*. *Signature* element is the digital signature of the *DA* to support integrity of delegation assertion. *Conditions* element indicates the valid duration of delegation assertion using *NotBefore* and *NotOnOrAfter* attributes. *Subject* element contains the *P's* information such as the name identifier. *EncryptedID* element is the name identifier of the subject in an encrypted format, using *AA*'s public key. This supports the privacy protection of the *P*, because this is only decrypted by *AA*'s private key. *Count* attribute is the number of services. *Delegation* element indicates whether the agent that receives this delegation assertion, is allowed to delegate the right to another agent. If this is "true", delegation is granted. *Consent* element indicates whether the *DA* obtains the *P's* consent to issue delegation assertion. If this is "true", consent has been obtained. *Service* element refers to the *EncryptedAttribute* element using the *DataIDRef* attribute. *SP-Address* element is the URL of the web service provider. *EncryptedAttribute* element is the *P's* input data, in which is used to perform web services. This corresponds to the input in WSDL [18], and contains additional information such as the least role, the recipient agent (*PA*) of results and so on. This is encrypted by the web service provider's public key in order to provide confidentiality of service. Figure 4 shows the example of a delegation assertion.

In this paper, we do not explain how to process *EncryptedData* and *EncryptedKey* elements. However, this can be processed using the technique described in the XML Encryption Specification [10].

4.3 Verification of Delegation Assertion

Delegation assertion is issued in the delegation authority, and is verified by web service providers. *DA* verifies the validity of credentials of *P*/agents assisted by *AA*, issues delegation assertion using the Procedure for Issuing Delegation Assertion (PIDA) algorithm.

Web service provider verifies the validity of delegation assertion using the Procedure for Verification Delegation Assertion (PVDA) algorithm.

```
<Assertion ID="_a75adf55-010d-dadsl42-tcbadb434d"
          IssueInstant="2005-03-05T02:46:02Z" Version="2.0">
  <Issuer>http://www.delegation-authority.com</issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     digital signature of delegation authority </ds:Signature>
  <Conditions NotBefore="2005-03-05T02:46:02Z"
              NotOnOrAfter="2005-03-05T02:55:00Z">
  <Subject>
    <EncryptedID>
      <xenc:EncryptedData> ... </xenc:EncryptedData>
      <xenc:EncryptedKey> ... </xenc:EncryptedKey>
    </EncryptedID>
  </Subject>
  <AttributeStatement>
    <Attribute Name="DeleInfo1">
      <AttributeValue>
        <Services count="2">
          <Delegation>true</Delegation>
          <Consent>true</Consent>
          <Service DataIDRef="Serv1">
            <SP-Address>http://www.SP1.com</SP-Address>
          </Service>
          <Service DataIDRef="Serv2">
            <SP-Address>http://www.SP2.com</SP-Address>
          </Service>
        </Services>
      </AttributeValue>
    </Attribute>
    <EncryptedAttribute DataID="Serv1">
      <xenc:EncryptedData> ... </xenc:EncryptedData>
      <xenc:EncryptedKey> ...</xenc:EncryptedKey>
    </EncryptedAttribute>
    <EncryptedAttribute DataID="Serv2">
      <xenc:EncryptedData> ... </xenc:EncryptedData>
      <xenc:EncryptedKey> ... </xenc:EncryptedKey>
    </EncryptedAttribute>
  </AttributeStatement>
</Assertion>
```

Fig. 4 Example of Delegation Assertion

Algorithm 1 : PIDA algorithm

Input : C_{PA} / $C_?$ and *DAT*
Output : DAT / Reject

(1) *DA* gets the *DAT* request from *PA* or other agents.
(2) If (*DA* receives C_{PA} from *PA*)
Then
(3) *DA* request *AA* to issue *AAT* with C_{PA}
(4) If (*AAT* is not issued) Then
(5) Return Reject
Else
(6) Set Consent element is true
(7) Issues *DAT* signed by PK_{DA}^-
Else
(8) Confirms *DAT's* Consent element.
(9) If (Consent element is false) Then
(10) Return Reject
Else
(11) *DA* request *AA* to issue *AAT* with $C_?$
(12) If (*AAT* is not issued) Then
(13) Return Reject
Else
(14) Decrypts *DAT's* Subject using Pk_{DA}^-.
(15) Query to Subject whether consent is true or not.
(16) If (Consent is true) Then
(17) Repeat Step 6 ~ Step 7
(18) Else
(19) Set Consent element is false
(20) Issues *DAT* singed by PK_{DA}^-

Algorithm 2 : PVDA algorithm

Input : *DAT*
Output : Success / Fail

(1) Service Provider (*SP*) gets the *DAT* from *SA*.
(2) *SP* confirms the signature of *DA* in *DAT* using PK_{DA}.
(3) If (Signature is not valid) Then
(4) Return Fail
Else
(5) Decrypts encrypted input data using PK_{sp}^-
(6) Confirms input data and process it.
(7) If (Result value is successful) Then
(8) Return Success
Else
(9) Return Fail

Our verification consists of PIDA and PVDA algorithms. The following notation is adopted in order to describe our algorithms. *DAT* denotes delegation assertion issued by the *DA*. *AAT* denotes authentication assertion issued by the *AA*. '?' denotes agents containing *PA*, *CA* and *SA*. '!' denotes *service provider (SP)* or authorities containing *AA* and *DA*. $C_?$ denotes the credentials of '?'. $PK_!$ / $PK_!^-$ denotes the public/private key pairs of '!'. Agents are able to delegate their rights in order to improve the performance of service usage. For example, suppose Alice, *P* wants to book a flight to Seoul, also wants to hold a hotel reservation for a business trip. Alice will first delegate her right to her agent, *PA* that performs the task on her behalf. *PA* delegates its rights to other agents as many as the number of requesting services in parallel. This is more efficient than delegation mechanism such as SPKI/SDSI [12] because SPKI/SDSI must configure an ordered certificate chain to verify the validity of delegation. It results in increasing time complexity on the verifying side. However, in proposed model, the web service provider is not required to configure a delegation chain, because the issue of *DAT* is managed by *P* and *DA*. The web service provider only requires *DAT* for verification. Each result of service is directly transferred to the recipient agent (*PA*) because the *EncryptedAttribute* element of *DAT* contains recipient information. For delegation among agents, delegating agent must contain *DAT* that consents element is 'true'. Otherwise the delegation request of agent is rejected.

5. Security Analysis

We analyze a proposed framework with respect to security analysis as following:

(1) *Confidentiality*: confidentiality is one of the cornerstones of information security for ensuring information that is accessible only to requests authorized to have access. In the proposed framework, *DA* issues delegation assertion in order to support secure and efficient delegation of rights among agents. We ensure

confidentiality by means of encrypted tags defined in delegation assertion (see the Figure 3). Only dedicated recipient is able to decrypt *EncryptedData* because *EncryptedKey* is encrypted by public key of recipient.

(2) *Integrity*: integrity means that assets can be modified only by authorized parties or only in authorized ways. In the proposed framework, delegation assertion is only manipulated by *DA*, and contains digital signature value as the content of *Signature* element. *Signature* element also contains information such as digest value and method in order to support the integrity of delegation assertion [20]. We ensure integrity of delegation assertion using these digest information.

(3) *Replay attack*: A replay attack is a form of network attack in which a valid data transmission is maliciously repeated or delayed by an adversary who intercepts the data. An attacker wants to impersonate the legal request to web service provider using a stolen delegation assertion from legal agent. However, it is impossible because delegation assertion contains digest information in order to prevent fabrication, and is valid during the period defined in *NotBefore* and *NotOnOrAfter* attributes of the *Condition* element.

(4) *User's privacy*: In the pervasive computing environments, to protect user's privacy is very important because an attacker may expose information related user's privacy to the outside without authentication. We ensure user's privacy by means of public key cryptography method. In the proposed framework, *AA* only knows who *P* is because subject element of delegation assertion is encrypted by *AA*'s public key. Only dedicated web service provider is able to decrypt the *EncryptedAttribute* element of delegation assertion because *EncryptedAttribute* element is encrypted by public key of dedicated web service provider.

6. Conclusion

This paper has proposed a delegation framework for ensuring security of web service in pervasive computing environments. Firstly, we have presented a delegation framework which is managed by user with assistance of authentication/delegation authorities. When a delegating agent transfers a right to the other agents, it must have obtained a user's consent across the delegation authority. Secondly, we have described a delegation assertion without any privacy information disclosure. We have exploited a XML-based encryption and digital signature mechanism in order to prevent the disclosure of information from unrelated agents. Thirdly, we have presented an efficient delegation mechanism without delegation chain. Our framework has only required a delegation assertion because delegation assertion is issued by delegation authority having trust relationship with web service providers. Future work will be applied to mobile devices with limited memory capacity.

Acknowledgments

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

References

[1] OASIS "Profile for the OASIS Security Assertion Language (SAML) V2.0" OASIS Standard, 15 March 2005.

[2] OASIS "eXtensible Access Control Markup Language (XACML) V2.0" OASIS Standard, 1 February 2005.

[3] G. Navarro, B.S. Firozabadi, E. Rissanen and J. Borrell, Constrained delegation in XML-based Access Control and Digital Rights Management Standards, In Proceedings of Communication, Network, and Information Security (CNIS '03) 2003

[4] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder and F. Siebenlist, X.509 Proxy Certificates for Dynamic Delegation, 2004

[5] Hidehito Gomi, Makoto Hatakeyama, Shigeru Hosono, Satoru Fujita, A Delegation Frame-work for Federated Identity Management, In Proceedings of DIM workshop, 2005

[6] Y. J Hu, Some thoughts on agent trust and delegation, In Proceedings of the fifth International Conference on Autonomous Agents, 2001

[7] G. Navarro, J. A. Ortega-Ruiz, J. Ametller, S. Robles, Distributed Authorization Framework form Mobile Agents, 2003

[8] Juan Dai, Robert Steele, UDDI Access Control, In Proceedings of the Information Technology and Applications, 2005

[9] Jun Wang, David Del Vecchio, Marty Humphrey, Extending the Security Assertion Markup Language to Support Delegation for Web Services and Grid Services, In Proceedings of the IEEE International Conference on Web Services, 2005

[10] XML Encryption Syntax and Proceeding, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[11] C. A. Ardagna, E. Damiani, S. De Capitani di Vimercati, P.Samarati, XML-based Access Control Language, 2004

[12] D. Clark, J. Elien, C. Ellison, M. Fredette, A. Marcos, R. Rivest, Certificate Chain Discovery in SPKI/SDSI, 2001

[13] R Yavatkar, D Pendarakis, and R Guerin, A Framework for Policy-based Admission Control, IETF Informational Standard, RFC 2753, January 2000

[14] Jongil Jeong, Dongkyoo Shin and Dongil shin, An XML-based Single Sign-On Schema Supporting Mobile and Home Network Service Environments

[15] V. Semar, Single Sign-On Using Cookies for Web application. Proceedings, IEEE 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise (WET ICE 99), 1999

[16] B. Pfitzmann, B. Waidner, Token-based web Single Sign-On with Enabled Clients, IBM Research Report RZ 3458(#93844), Nobember 2002

[17] Andreas Matheus, How to declare access control policies for XML structured information objects using OASIS's eXtensible Access Control Markup Language (XACML), Proceedings of the 38th Hawaii International Conference, 2005

[18] W3C. , Web Service Description Language (WSDL) 1.1, W3C Note, March 2001. http:// www.w3.org/TR/WSDL.

[19] OASIS. , Web Service Security : SOAP Message Security 1.0, OASIS Standard, March 2004

[20] XML Signature, http://www.w3.org/TR/xmldsig-core

Authors

Name: Hyun Sik Hwang

Address: Department of Computer Engineering, Sungkyunkwan Univ., 300 Chunchun, Jangan, Suwon, Gyeonggi, Korea

Education & Work experience:

B.S. Science in Electrical and Computer Engineering, Dongguk University. 1996.3 -2004.2

Other information: He is a researcher at Database Laboratory in Sungkyunkwan University. His research interests include XML Security, Access Control, Context-aware Access Control and Ubiquitous computing.

Name: Hyuk Jin Ko

Address: Department of Computer Engineering, Sungkyunkwan Univ., 300 Chunchun, Jangan, Suwon, Gyeonggi, Korea

Education & Work experience:

M.S. Science in Electrical and Computer Engineering, Sungkyunkwan University 1993
B.S. Science in Electrical and Computer Engineering, Sungkyunkwan University. 1991

Other information: He is a researcher at Database Laboratory in Sungkyunkwan University. His research interests include XML Security, Access Control, Distributed Database, Ubiquitous computing, Ontology Modeling, Database Security.

Name: Kyu Il Kim

Address: Department of Computer Engineering, Sungkyunkwan Univ., 300 Chunchun, Jangan, Suwon, Gyeonggi, Korea

Education & Work experience:

M.S. Science in Electrical and Computer Engineering,
Sungkyunkwan University 2003.3 – 2005.2
B.S. Science in Electrical and Computer Engineering, Wonkwang University. 1996.3 - 2003.2

Other information: He is a researcher at Database Laboratory in Sungkyunkwan University. His research interests include Access Control, Context-aware Access Control and Workflow.

Name: Woojun Kang

Address: Department of Management Information System, Korea Christian Univ., Hwagok 6-Dong, Gangseo-Gu, Seoul, Korea

Education & Work experience:

Ph.D. Computer Science in Electrical and Computer Engineering, Sungkyunkwan University, 2001
M.S. Computer Science, Yonsei University, 1994
B.S. Electrical Engineering, Yonsei University, 1984
Researcher of Korea Software Development Institute, IBM Korea, 1984 - 1999

Other information: He is a Professor in the Department of Management Information System in Korea Christian University, Korea. His current research interests include XML/Web Mining, Access Control, and DRM.

Name: Hae Kyung Rhee

Address: Department of Computer Game Information, YonginSongdam College, 571-1 Mapyung, Cheoin, Yongin, Gyeonggi, Korea

Education & Work experience:

Ph.D. Computer Science, Sungkyunkwan University, 2000
M.S. Computer Science, University of Illinois at Urbana-Champaign, 1985
B.S. Computer Science, Soongsil Univ., 1979

Other information: She is a Professor in the department of Computer Game Information in YonginSongDam College, Korea. Her current research interests include Data Modeling, Database Security.

Name: Ung Mo Kim

Address: Department of Computer Engineering, Sungkyunkwan Univ., 300 Chunchun, Jangan, Suwon, Gyeonggi, Korea

Education & Work experience:

Ph.D. Computer Science, Northwestern University, 1990.
M.S. Computer Science, Old Dominion University, 1986
B.S. Mathematics, Sungkyunkwan University, 1981

Other information: He is a Professor in the School of Information and Commnunication Engineering in Sungkyunkwan University, Korea. His current research interests include XML/Web Mining, Access Control, and Ontology Modeling.