Enhanced Access Control with Semantic Context Hierarchy Tree for Ubiquitous Computing

Hyuk Jin Ko[1], Woojun Kang[2]

Department of Computer Engineering, Sungkyunkwan University, Korea[1],
College of Management and Information Technology, Korea Christian University, Korea[2]

Abstracts:
In most ubiquitous applications involving ubiquitous hospital, security mechanisms supporting the context of users and systems has becoming a key issue. Access control systems have to make decisions with users' current security-relevant context, such as time, location, user activity, and other environmental information available when the access requests arrive. In ubiquitous context-aware access control systems, a query issued by an authorized user might not be resolved if context specified by policy does not match that specified in the query, as though the semantic similarity of both contexts is much close. In this paper, semantic context-aware access control (SCAC), is proposed, to solve the problem mentioned earlier. The proposed system fetches users' context and ontology from middleware, with which context hierarchies are built. Using the context hierarchies and reasoning rules extracted from relevant ontology, the proposed model can bridging the semantic gap between policy specific context and query specific context in ubiquitous application systems such as ubiquitous hospital management systems.

Key words:
Access Control, Context, Context Hierarchy, Ontology.

1. Introduction

In a ubiquitous environment, users typically access resources using mobile devices [1, 2]. Since the context of a user is highly dynamic, granting a user access without considering the user's current context can cause a security problem, that is, even an authorized user can damage the system as the system may have different security requirement within different context. In terms of context-aware access control, the context information referenced in design can be different from that specified in query on execution time. As a result, it seems that the query cannot be resolved properly when the context specified in access control policy does not exactly match with the context in query, even though both context are semantically equivalent.

In this paper, Semantic Context-aware Access Control system (SCAC) is proposed to solve the problem mentioned above. SCAC takes context and ontology from the context middleware and arranges context according to the abstraction level to build context hierarchies. Using these context hierarchies and inference rules extracted from the ontology, SCAC can overcome the semantic gap between context specified in the policy and context collected from highly dynamic context sources in ubiquitous environments.

The subsequent sections of this paper are organized as follows. In Section 2, the preliminaries relevant to the context-aware access control and the ontology are explained. In Section 3, a summary of related work in the area of access control is discussed. In Section 4, the design of the SCAC model and authorization enforcement algorithm is described. In Section 5, the overall implementing architecture is explained. Section 6 concludes this paper.

2. Backgrounds

2.1. Representation of Context

Typically, context can be represented as first-order predicates. The name of the predicates corresponds to the type of context to be described. This convention allows us an uniform representation for different kinds of context [11]. For example, context predicates are like *LocatedIn(Bob, room209), TemperatureOf(get_room#( ), 26^{o}C)*. A predicate consists of many terms. The values of each term in a predicate are actually constrained by the domain of context. Some of terms in a context predicate can be functions that return some value. This logical model for context using first-order logic is so powerful as to express a rich variety of context. Complex context expressions can be represented by combining Boolean operations such as conjunction, disjunction and negation.

Manuscript received October 5, 2008
Manuscript revised October 20, 2008

The predicate model also allows both universal and existential quantification over variables. This allows us a parameterization of context for representing of a much richer set of context. For example, an context expression is like *LocatedIn(Bob, room209)* $\wedge$ *WhileOnDuty(Bob)*, which refers to the context "Bob is in room 209 while on his duty".

2.2 Inference of Context

Generally, an ontology is a description of important concepts in a domain, crucial properties of each concept and restrictions on properties such as property cardinality, property value type, domain and range of a property. Context ontology gives us a chance to derive new context from other existing context. The rules extracted from ontology are used to infer new context from existing context.

Inference rules used in this paper are adapted from *owl:ObjectProperty* [12], which defines the relationship between many concepts in a specific domain. Table 1, quoted from [7], presents the inference rules extracted from the context ontology hierarchy by defining relationship between concepts. Another similar approach, CONON [12], also presents a generation of inference rules with the equivalence of OWL and Description Logic. By referring each relationship among context concepts in Table 1, we realize that the instances of high-level concepts can be inferred from the instances of low-level concepts.

Table 1. Inference rule extracted from ontology (quoted from [7])

Relationship	Inference rule
EQUIVALENCE	if $C_i \equiv C_j$ then $C_i \Rightarrow C_j$
IS-PART-OF	if $C_j \in \{C_i\}$ then $C_i \Rightarrow C_j$
IS-A	if $C_i \subset C_j$ then $C_i \Rightarrow C_j$
UNION	if $C_j = C_1 \cup C_2 \cup ... \cup C_k$
	then $C_i \Rightarrow C_j$, i = 1,...,k
INTERSECTION	if $C_i = C_1 \cap C_2 \cap ... \cap C_k$
	then $C_i \Rightarrow C_j$, j = 1,...,k

3. Related Works

Role-Based Access Control (RBAC) [8, 9, 10] is proposed to restrict the actions that legitimate users can perform based on the set of authorizations applicable to a group of users. A major benefit of RBAC is the ease of administration of the security policy and its scalability. In Generalized RBAC model [3, 4], RBAC is extended by applying the roles to all entities in a system. By defining three types of roles, i.e., subject roles, environment roles, and object roles, it uses context information for making access decisions. A Content-based RBAC [5] supports an flexible specification of authorization based on the qualification and characteristics of users, called credentials. A CS-RBAC [6] extends RBAC to make it sensitive to the context of both user and the target object. A concept-level access control model [7] for a semantic web is proposed for specifying authorizations over concept ontology and enforcing them upon data instances with support for propagations based on the relationship among concepts.

A major drawback of the previous approaches mentioned above does not exploit the semantic information of context, which provide various advantages to the context-aware access control systems. Our approach using inference rules extracted from context ontology can overcome semantic gap between static context information and dynamic context information. It also guarantees that the context-aware access control system can have scalability against context evolution by using mature ontology integration technologies.

4. Semantic Context-aware Access Control (SCAC)

4.1. Motivating Example

To illustrate our motivating example, let us consider an intelligent hospital in a ubiquitous environment. We assume that the sensors in this building can capture, process and store a variety of context information regarding location, time, and user activities, etc. In this case, access privilege rules are already specified in the access control policy. For instance, the privilege of doctors to access patient information could be constrained in some context like the following:

Full-time doctor can do all operations on the patient records at any time.
Part-time doctor can do all operations on the patient records only on duty.
Nurse can only retrieve the patient records on duty.

The innate property of ubiquitous computing is so dynamic that the collected context from sensors could be different from those specified in access control policy. In a situation, for example, that a doctor Bob wants to access the medicine information about parents of his patient, the following case can happen:

Policy authorization: *"A doctor Bob in a pediatrics ward could have privilege to access the information of infants' parents."*
Collected context from sensor: *"Doctor Bob is in room 209"*

At this time, even if room 209 belongs to a pediatrics ward, the access was not permitted because the specified context "is in a pediatrics ward" and collected context "is in room 209" are not explicitly matched with each other even though both context have equal meaning implicitly. It is consequently natural that access should be permitted. Previous context access control approaches did not mention this problem. To handle this problem, we propose a semantic context access system based on inference of concept ontology.

4.2. Semantic Context-aware Access Control (SCAC) Model

Our SCAC model is an extension of the RBAC model [10]. The SCAC model is composed of the following entities and relationships.

Definition 1. SCAC Entities
An authorization *au* is represented as a 4-tuple <*S,R,P,C*>. Subject *S* is a subject of the system. Role *R* is a grouping primitive for users. Permission *P* is an access privilege to data object which is defined as a triple <*sign, mode, object*>, where *sign={+,-}, mode={create, delete, read, write }*. Context *C* is an expression by Boolean operations over context predicates.

Definition 2. SCAC Relationships
Access control policy, *ACP*, is a set of authorizations. *AR(S)* denotes an authorized role set of subject *S*. *AP(R)* denotes an authorized permission set of role *R*. The function *basic_eval(S,P)* is true if and only if there exist any role *R*: $R \in AR(S)$ and $P \in AP(R)$. The function *bas(au)* and *cxt(au)* extract <*S,R,P*>, called basic authorization, and <*C*>, called context constraints, from a SCAC authorization *au* <*S,R,P,C*>. The function *context_eval(C)* is true if and only if the context constraints *C* on the basic authorization <*S,R,P*> are satisfied. An authorization query is expressed as *"?<S,P,C>"*.

Example 1: Authorization "*Doctor Bob has write access to the record of patient Jane only in pediatrics ward"* can be specified as *au = <Bob, Doctor, <+, write, Jane's Record>, LocatedIn(Bob, pediatricsWard)>*, where *bas(au)=<Bob, Doctor, <+, write, Jane's Record>>* and *cxt(au)=<LocatedIn(Bob, pediatricsWard)>*.

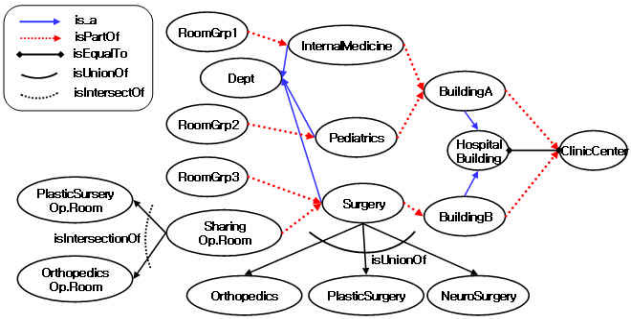

Fig. 1. Hospital Ontology

In example 1, the URIs of specified concepts in the domain-specific hospital ontology are omitted to make it simple. An example of hospital ontology and its OWL serialization are shown in Figure 1 and Figure 2, respectively.

4.3. Context Concept Hierarchy

Definition 3. Inference Rule Set (RS)
Given the relationship among context concepts, instances of one context concept can be inferred from the instances of another. If the instances of context concept Cj can be inferred from the instances of Ci, we denote it as Ci $\rightarrow$ Cj. An inference rule set extracted from each relationship is denoted as RS(relationship). A set of all inference rule set, denoted as RS, is an union set of RS(relationship) for each relationship in a domain-specific ontology.

Example 2: In Figure 1, for example, *RS(IS-A)* = *{ InternalMedicine* $\rightarrow$ *Dept , Pediatris* $\rightarrow$ *Dept, Surgery* $\rightarrow$ *Dept, BuildingA* $\rightarrow$ *HospitalBuilding, BuildingB* $\rightarrow$ *HospitalBuilding}* and *RS(INTERSECTION)* = *{SharingOp.Room* $\rightarrow$ *PlasticSurgeryOp.Room, SharingOp.Room* $\rightarrow$ *OrthopedicsOp.Room}* and *RS = RS (IS-A)* $\cup$ *RS (IS-PART-OF)* $\cup$ *RS (EQIVALENCE)* $\cup$ *RS (UNION)* $\cup$ *RS (INTERSECTION)* .

Definition 4. Inference Chain
An inference chain is built by joining inference rules that participate in inferring a specific common concept. There may be many inference chains in a *RS*. A set of inference chains of a *RS* is denoted as *IC(RS)*.

Example 3: Let inference rule set *RS* = $\{C_1 \rightarrow C_2, C_2 \rightarrow C_3, C_3 \rightarrow C_4, C_a \rightarrow C_3, C_a \rightarrow C_b, C_b \rightarrow C_c, C_c \rightarrow C_d\}$ then inference chain of a *RS*, *IC(RS)*=$\{C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4, C_a \rightarrow C_3, C_a \rightarrow C_b \rightarrow C_c \rightarrow C_d\}$.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns="http://dblab.skku.ac.kr/hospital.owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://dblab.skku.ac.kr/hospital.owl
">

  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Ward"/>
  <owl:Class rdf:ID="Pediatrics">
    <rdfs:subClassOf rdf:resource="#Ward"/>
  </owl:Class>
  <owl:Class rdf:ID="Floor_A_2">
    <rdfs:subClassOf rdf:resource="#Floor"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="located_in">
    <rdfs:range rdf:resource="#Floor_A_1"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf
             rdf:parseType="Collection">
          <owl:Class rdf:about="#Rgroup_A_1"/>
          <owl:Class rdf:about="#Rgroup_A_2"/>
          <owl:Class rdf:about="#Rgroup_A_3"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>
  …
  <Rgroup_A_2 rdf:ID="ra_203"/>
  <Nurse rdf:ID="Margarette"/>
  <Rgroup_A_3 rdf:ID="ra_304"/>
  <Rgroup_A_2 rdf:ID="ra_204"/>

   ...

</rdf:RDF>
```

Fig. 2. OWL Serialization of Hospital

Definition 5. Low/High-level concept set
Given a concept *C*, *LLC(C)* is defined as a set of low-level concepts that can infer *C*. Similarly, *HLC(C)* is defined as a set of high-level concepts that can be inferred from *C*. Additionally, *HGC(RS)* is defined as a highest-level concept set; that means a set of concepts that cannot infer any other concepts.

Example 4: Let $RS=\{C_1 \rightarrow C_2,\ C_2 \rightarrow C_3,\ C_3 \rightarrow C_4,\ C_a \rightarrow C_3,\ C_a \rightarrow C_b,\ C_b \rightarrow C_c,\ C_c \rightarrow C_d\}$ then $IC(RS) = \{C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4,\ C_a \rightarrow C_3,\ C_a \rightarrow C_b \rightarrow C_c \rightarrow C_d\}$, $LLC(C_3) = \{ C_1, C_2, C_a \}$, $HLC(C_3) = \{C_4\}$, $LLC(C_a) = \{ \}$, $HLC(C_a) = \{C_3, C_b, C_c, C_d\}$ and $HGC(RS) = \{ C_3, C_4, C_d\}$.

Definition 6. Context Concept Hierarchy
A set *LLC(C)* is regarded as a partial ordered set poset $(LLC(C), \prec)$, where $\prec$ is a concept-level order. If $C_i \rightarrow C_j$ then $C_i \prec C_j$. $C_i \prec C_j$ means that C_j is higher than C_i in concept-level order. The partial ordered set poset builds a hierarchy, depicted by Hasse diagram, The context concept hierarchy can thus be defined as follows;
The context concept hierarchy set in a *RS*, denoted as *CCH(RS)*, can be defined as a set of *LLC(C)*, for all $C \in HGC(RS)$.

Example 5: From Figure 1, *CCH(RS)* = { *RoomGrp1* → *InternalMedicine* → *Dept*, *RoomGrp1* → *InternalMedicine* → *BuildingA* → *ClinicCenter*, ..., *NeuroSurgery* → *Surgery* → *BuildingB* → *HospitalBuilding* }. Hasse Diagram shown in Figure 3 depicts *CCH(RS)*.

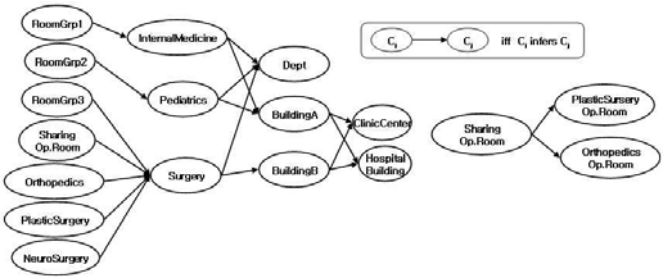

Fig. 3. Hospital Context Concept Hierarchy

4.4. Policy Enforcement

To resolve the query *?<S,P,C>*, the function *SCAC_enforcement(S,P,C)* is executed. In it, the sub-function *basic_eval(S,P)* is firstly evaluated. If the result is true, the sub-function *context_eval(C)* is evaluated with inference rules extracted from context ontology. Our method can overcome problems that context constraints specified in policy and those in a query cannot be matched with each other, even though they both have an implicitly equal meaning. Table 2 presents the authorization matrix of function *SCAC_enforcement* according to the sign of the query and the relevant authorization.

Table 2. Authorization Matrix based on Context Concept Hierarchy

		Sign of relevant au in policy	
	Condition	+ (positive)	- (negative)
Query ?< S,P,C^q>	$C^q \preceq C^p$	Authorized	Non Authorized
	$C^p \prec C^q$	Non Authorized	Authorized

(C^q : context concept in the query, C^p : context concept in the policy authorization)

Example 6: Let Policy *ACP={ <Bob, Doctor, <+, write, InpatientRecord>, LocatedIn(Bob, BuildingA)>}*, Query q *?<Bob, <+, write, InpatientRecord>, LocatedIn(Bob, Orthopedics)>* and Context Concept Hierarchy from Example 5 then *bas(q)=<Bob, Doctor, <+, write, InpatientRecord>>*, *cxt(q) = < LocatedIn(Bob, Orthopedics) >* and *SCAC_enforcement(q)*, in which both *basic_eval(bas(q))* and *context_eval(cxt(q))* are *TRUE*, returns *Authorized*. In *context_eval(cxt(q))*, *can_do(LocatedIn(Bob, Orthopedics), LocatedIn(Bob, BuildingA))* is executed to *TRUE*.

Algorithm: Policy Enforcement

```
Input : (s, p, ceq)
subject s,
permission p 3-tuple <sign,mode,object>,
context expression ceq

Output:
enum {Authorized, Non-authorized}

Desc.:
check whether a subject S has an authorization fo
r permission P under ceq

Sub-function  findConcept(c):
if c is an instance then return concept of c, els
e return c (when c is a concept itself).

Sub-function can_do(Ci, Cj):
check whether Ci can infer Cj, where Ci,Cj∈CCH(R
S), RS = { a set of reasoning rules extracted fro
m the ontology on a specific domain }


enum SCAC_enforcement(s,p,ceq)
{
   // check firstly basic-authorization
   if( basic_eval(s,p) == FALSE )
   then return(Non-authorized);

   // then, check context constraints
   if( context_eval(ceq) == TRUE )
   then return(Authorized);
   else return(Non-authorized);
}
```

Algorithm: Context Expression Evaluation

```
BOOLEAN context_eval(ceq)
{
   // Find a set of context concepts of policy
   // authorizations associated with input query,
   // then store in CCp_set
   CCp_set = { CCp | CCp = findConcept(cp),
      cp = cxt(au), au is a correspondent to
      bas(input),input=<s,p,ceq>, au∈ACP }

   For( each context predicate cq in ceq )
   {
      CCq = findConcept(cq); // find cq's concept
      resFlag = 0; // flag for temp result

      For( each CCp, CCp∈CCp_set ){
         If( sign == positive )
            if( can_do(CCq, CCp) )
            then { resflag = 1; break; }
         elseif( sign == negative )
            if( can_do(CCp, CCq) )
            then { resflag = 1; break; }
         else return(FALSE);
      }
      if( resFlag == 0 ) return(FALSE);

   }
   if( resFlag == 1 ) return(TRUE);
   else return(FALSE);
}
```

5. Implementing Architecture

The overall architecture of the SCAC System is presented in Figure 4. On behalf of context consumer SCAC, COAgent module collects, integrates context and annotated ontology from context middleware, such as Gaia [11], through an agent platform, such as JADE [13], and stores this information into CODB.

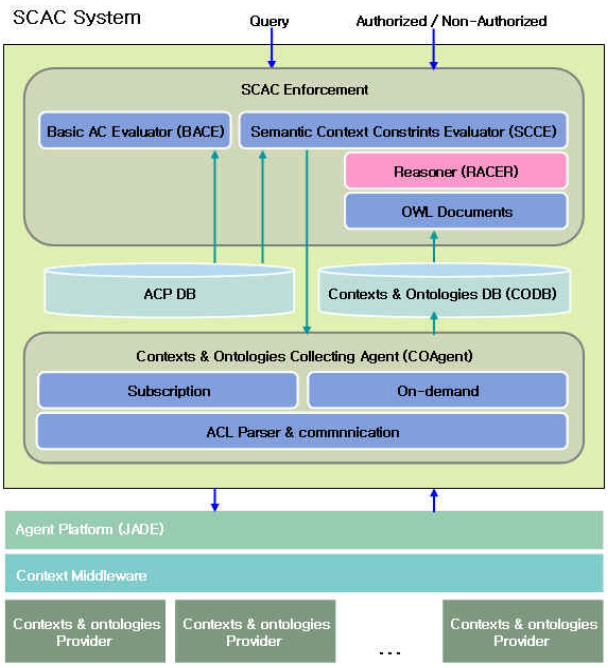

Fig. 4. Architecture of SCAC System

COAgent can obtain context and ontology using two possible methods. The COAgent wanting to obtain context and ontology immediately upon query request, may use the on-demand method. Using the subscription method,

the COAgent can request notification whenever context and ontology are newly updated.

To resolve the query, in the 'SCAC Enforcement' module, the 'Basic AC Evaluator' firstly checks basic authorization against authorization policies in ACP DB. If the result of it is true, the 'SCCE' evaluates context constraints with context information and the reasoning rules in CODB. The reasoning method is implemented with RACER and its Java API [14].

6 Conclusion

In ubiquitous environments, security and context-awareness is an interesting and challenging research subject. The dynamism, ubiquity, and non-intrusiveness of ubiquitous computing, presents additional challenges and raises new issues. In this paper, linking context-aware access control and semantic information of context is presented, for more efficient security administration. The SCAC model for semantic context-aware access control in ubiquitous environments, is proposed.

This model can simplify policy management by separating the entities involved in context-aware access control, into basic access control entities and context constraints. In addition, it supports making more precise, flexible decisions regarding authorization, using semantic information of context and an enforcement algorithm. Further challenges are to extend SCAC into having the capability of processing disjunctive context expression, and the formal proving about conflict-free between inferred positive constraints and inferred negative constraints.

References

[1] M.Weiser. Hot Topics: Ubiquitous Computing. *in IEEE The computer, 1993.*

[2] P. Bellavista, A. Corradi, C. Stefanelli. The Ubiquitous Provisioning of Internet Services to Portable Devices. in *IEEE Ubiquitous Computing, Vol. 1, No. 3, 2002.*

[3] M.J. Moyer, M.J. Covington, M. Ahamad. Generalized role-based access control for securing future applications. in *NISSC2000 23rd National Information Systems Security Conference, 2000.*

[4] M.J. Covington, S. Srinivasan, A. Dey, M. Ahamad, W. Long, G. Abowd. Securing context-aware applications using environment roles. in *SACMAT 2001.*

[5] N.R. Adam, V. Atluri. A Content-based Authorization Model for Digital Libraries. in *IEEE Transactions on knowledge and data engineering, Vol. 14, No. 2, 2002.*

[6] A. Kumar, N. Karnik, G. Chafle. Context Sensitivity in Role-based Access Control. in *Operating Systems Review, Vol. 36, No. 3, IBM Journal, 2002.*

[7] Li Qin, V. Atluri. Concept-level Access Control for the Semantic Web. in *ACM Workshop on XML Security, 2003.*

[8] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role Based Access Control Models. *in IEEE Computer, Vol. 29, No. 2, February 1996.*

[9] R. Sandhu, P. Samarati. Access control: principles and practice. in *IEEE Communication Magazine, vol. 32, 1994.*

[10] R. Sandhu, D. Ferraiolo, and R. Kuhm. The NIST Model for Role-Based Access Control: Towards A Unified Standard. in *Proceedings of the fifth ACM workshop on Role-based access control, 2000.*

[11] A. Ranganathan, R.H. Campbell. An Infrastructure for context-awareness based on first-order logic. in *Personal and Ubiquitous Computing, Vol. 7, Issue 6, 2003.*

[12] X.H. Wang, D.Q. Xhang, T. Gu and H.K. Pung. Ontology Based Context Modeling and Reasoning using OWL. in *PerCom2004 Annual Conference on Ubiquitous computing and Communications Workshop, 2004.*

[13] F. Bellifemine, A. Poggi, G. Rimassa. Developing multi agent systems with a FIPA-compliant agent framework. in *Software - Practice & Experience, John Wiley & Sons, Ltd., 2001.*

[14] V. Haarslev and R. Möller. Racer: A Core Inference Engine for the Semantic Web. in *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), located at the 2nd International Semantic Web Conference ISWC 2003, Sanibel Island, Florida, USA, October 20, 2003.*

Authors

Name: Hyukjin 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 1994, B.S. Science in Electrical and Computer Engineering, Sungkyunkwan University. 1991, R&D Cententer, LG Electronics, Korea, 1994 - 2002.

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: 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.