Using a Process Algebraic Approach of Verifying Access Control in XML-based Healthcare Systems

Ji-Yeon Lee and Jin-Young Choi*

Department of Computer Science and Engineering, Korea University
Anam-dong, Sungbuk-gu, Korea
Email: {jylee, choi}@formal.korea.ac.kr

Il-Gon Kim

Korea Internet and Security Agency
IT Venture Tower, Jungdaero 135, Songpa, Seoul, Korea
Email: igkim@kisa.or.kr

Ki-Seok Bang

College of General Education, Hallym University
Hallymdaehak-gil, Chuncheon, Gangwon-do
Email: mysaver@hallym.ac.kr

Healthcare information and its secure management has become an active research topic along with an increase in the usage of XML documents and the importance of access control in maintaining patient records. In this paper, we present a methodology to describe a formal specification for an authorized view of an XML-based healthcare system having a schema-level access control by assigning well-established concurrency semantics to the system. To achieve this goal, we translate the semantics of the schema, query, access control for XML-based health records, and XPath expressions, into a CSP-like process algebra language through an illustrative example. Finally, our experimental results show the possibility to reason about security properties of an XML-based access control model with the support of automated model checking tools, because it provides the formal semantics for access control policies and XML documents with the tree structure.

Keywords: *Formal specification, process algebra, CSP, health records, XPath, authorized view*

ACM Classifications: *D.2.4 (Software Engineering – Software/Program Verification – Formal Methods), D.4.6 (Operating System – Security and Protection – Access Controls), J.3 (Computer applications – Life and Medical Sciences – Medical Information Systems)*

1. Introduction

In recent years, electronic health records (EHRs) (ISO/TR 20514, 2005; Bird *et al*, 2003) have attracted considerable attention. One important project that has been considered in their study is the development of a healthcare system that uses XML data and web services; that is to say, the issue of access control in light of the complexity and diversity of the data formats and data

* Corresponding author: Jin-Young Choi

Manuscript received: 14 August 2012
Communicating Editor: Seok-Hun Kim

structures involved in a healthcare system has been prominent (Xie *et al*, 2010; Hori and Ohashi, 2005; Sachdeva *et al*, 2010). One of the most important requirements for an EHR system is access control and authorization, so as to prevent illegitimate actions and verify that a user has permission to access confidential information such as patient records. To satisfy the authorization requirement for an EHR system, access control model should only be granted under precise conditions and circumstances. However, considering the complexity of the medical workflow, the large number of health records, and the users and systems involved, it seems likely that access control management would be prone to errors.

In addition, access control for healthcare systems using XML documents differs from a general access control because of the XML data structure and the semantics of the XPath (XML Path Language) expression. Thus, it is not trivial to derive the authorized view and verify security issues pertaining to access control for XML documents.

Access control policies for XML documents generally use path expressions such as XPath (Clark and Derose, 1999) or XQuery (Boag *et al*, 2002) to specify an authorized view for objects (Damiani *et al*, 2002; Gowadia and Farkas, 2005). An *authorized view* is a restricted view of an XML document that consists of only that information that the user is authorized to access after the enforcement of access control policies. In other words, given a query q over a confidential XML document D for patient records, it is very important that the result of the query evaluation contains only the selective nodes in the context of authorized views. Therefore, reasoning about the access control model for XML-based health records is a non-trivial topic, considering the inherent challenges, namely, the hierarchical nature of XML and query expressions.

Existing approaches (Qunoo and Ryan, 2010; Fisler *et al*, 2005; Fundulaki and Marx, 2004) state the semantics of access control using an ambiguous natural language, and therefore it becomes difficult to validate whether the access control policies are enforced correctly and efficiently. Furthermore, the hierarchical nature of XML documents makes the task of reasoning about security in the authorized view non trivial; XML nodes as objects are restricted by the form of regular path expressions such as XPath, and the authorized view depends on the structural relationship between the parent and the child nodes. Therefore, we saw the need for a method of describing the hierarchical nature of the authorized view in terms of the semantics of access control policies.

The main contributions of this paper can be summarized as follows:

- We present a methodology to specify an access control model for XML-based health records by translating a schema, query and access control policy in CSP (Communicating Sequential Process) formal language.
- We provide a process algebra view of specifying an authorized view to an XML document for EHR.
- We construct an equivalent process algebra model to an authorized view on XML documents, after interpreting XPath and the semantics of an access control policy.
- We provide a preliminary study for specification and verification of access control for XML documents for EHR, with the FDR (Failure-Divergences Refinement) model checker (Formal Systems, 1999).

The remainder of this paper is organized as follows. Section 2 describes some related works. In Section 3, we give a brief overview of CSP process algebra. In Section 3, we illustrate a system architecture and a scenario of EHR running example. In Section 4, we summarize the semantics of access control policies for XML documents, and XPath expressions. In Section 5, we show how

CSP process algebra can capture the hierarchical nature of XML documents using both an automata-based formal model and the semantics of access control policies. In Section 6, we illustrate how to verify the access control policies for XML documents to show the feasibility of our approach. Finally, we present our conclusions in Section 7.

2. Related Works

As access control for XML documents is receiving significant attention, model checking approaches for verifying security properties of the access control policies have been studied recently (Fisler *et al*, 2005; Fundulaki and Marx, 2004). Graham *et al* (Hughes and Bultan, 2004) presented a formal model for the access control policies in the XACML (eXtensible Access Control Markup Language) language. Fisler *et al* (2005) developed a software suite, Margrave, for analyzing role-based access control policies written in XACML (Godik and Moses, 2003) and translating the result into a form of decision-diagram to answer queries. Qunoo and Ryan (2010) showed how to formalize and reason about access control policies in web-based collaborative systems using the X-policy language.

However, all of these approaches (Qunoo and Ryan, 2010; Fisler *et al*, 2005; Fundulaki and Marx, 2004; Hughes and Bultan, 2005; Zhang *et al*, 2005) focus on translating the access control policies written in XACML language into formal specifications and none consider the hierarchical nature of the object specified by regular path expression such as XPath.

To the best of our knowledge, our approach is the first to provide a process algebra view of the authorized view of XML documents or describe their hierarchical nature. Furthermore, this approach can also be easily applied to other process algebra languages such as LOTOS (Van Eijk *et al*, 1989). Thus, we believe that it provides an initial step for specifying and verifying correctness and completeness of access control policy for XML documents with hierarchical tree structures, with the support of automated model checking tools.

3. Illustrative Example: XML-based Electronic Health Record

3.1 System Architecture

As shown in Figure 1 and Figure 2, an XML-based EHR system depends upon the XML data structure and the access control rules established by the security administrator. The system searches the XML-based EHR system for the medical record of an admitted patient by inter-

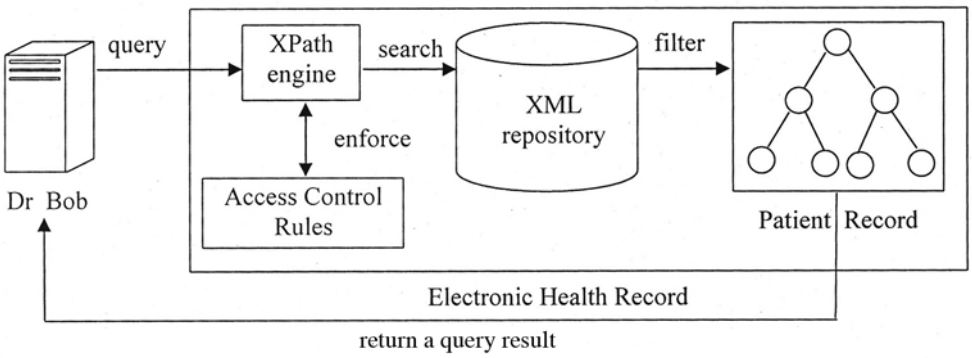

Figure 1: Access control in XML-based EHR

```
<Records>
  <Patient @Name=Joe>
   <Medical>
     <Doctor> Bob </Doctor>
     <Diagnosis> vertebral f. </Diagnosis>
     <Prescription> drug number 10 </Prescription>
   </Medical>
  </Patient>
</Records>
```

Figure 2: Example of an XML data structure of an EHR

preting XML query languages such as XPath. To achieve this, the security enforcement module enforces access control rules by refining the XPath language against the EHR in XML. In this paper, we focus on the access control on EHR which is widely used in Web services.

3.2 Scenario

Dr Bob (in future, simply B) receives a patient named Joe who complains of back pain. Hence, Dr Bob decides to consult Joe's record in the EHR system for his medical examination before diagnosing his illness. In the first step, Dr Bob logs in to the EHR system and sends a query *q*. For the sake of readability, we omit the expression '[@Name=Joe]'(inherent in query *q*) in the remainder of the paper.

*q: /Records/Patient[@Name=Joe]/Medical/**

We assume that before Dr Bob sent the query, he was authenticated by the EHR system when he entered his user ID and password that were transmitted over secure channels that employ protocols such as the SSL. In the second step, the EHR system imposes access control by enforcing relevant access control rules such as *R1*, *R2* and *R3*.

R1: <B, /Records/Patient[@Name=Joe]/Medical/Diagnosis, r , + , PL>

R2: <B, /Records/Patient[@Name=Joe]/Medical/Prescription, r , + , PL>

R3: <B, /Records/Patient[@Name=Joe]/Medical/, r , – , NL>*

These rules are enforced by the EHR system before finding a relevant electronic patient record in XML. *PL* and *NL* represent the scope of access control rule. PL (positive local) is the scope of the grant rule and is propagated to only the specified node. NL (negative local) is the scope of the deny rule and is propagated to only the specified node. More details regarding the notation and semantics of access control rules can be found in Fundulaki and Marx (2004). These systems can allow both grant and deny access rules (denoted by + and –, respectively). In the third step, after interpreting the query and enforcing the access control rules, the EHR system returns Joe's patient record, as shown in Figure 2. The notation and semantics of access control rules will be described in Section 4.

4. Semantics of Access Control Policy and Xpath

4.1 Schema-level Access Control Policy

An existing access control policy (Damiani *et al*, 2002) to XML documents normally consists of a 5-tuple of the form <*subject, object, action, effect, propagation*>, where:

- *subject* is the requester who may access the object (e.g., a user or group).
- *object* is the data that the requester is allowed to access and has the form of a path expression.
- *action* represents the action, such as read, write, and delete, which the subject is permitted to perform on the object.
- *effect* ∈ {+, –} is the sign for the grant (denoted by '+' sign) or the deny (denoted by '–' sign) of accessing the requested object.
- *propagation* ∈ {PL, PR, NL, NR} refers to the scope of the rule (see Section 4.3.1).

An access control policy (in short, ACP) can be specified on the level of a single XML document or schema. We focus on the schema-level access control semantics in this paper and consider only the read action (in short, *read=r*) for the sake of simplicity.

4.2 XPath Expression

Most of the existing approaches for regulating access to XML documents use the XPath path expression to specify a document or node regulated by an access control policy. We also adopt the XPath language and translate it into a process algebra model by mapping its semantics of the access control policy for XML documents, based on its advantages.

Existing approaches use a simplified version of XPath expression to address XML nodes (objects). We will heavily use the axis expressions 'node', '/', '//', '*', and '@' throughout the remainder of this paper.

- '*node*' is used to select all the children of the node.
- '/' separates a sequence of element names and selects from the root node.
- '//' selects nodes in the document from the current node that conforms to the selection wherever they are.
- '*' is used to select all the applicable nodes.
- '@' refers to the attribute nodes in conditions (also called predicates).

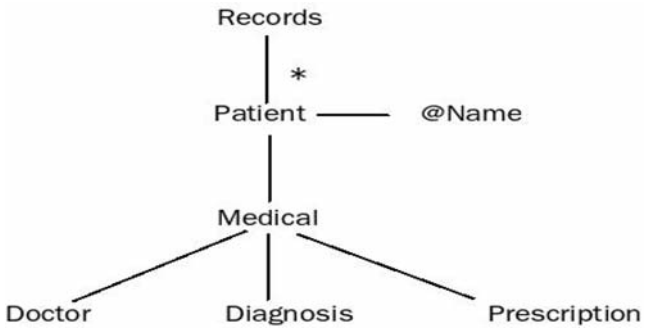

Figure 3: Example of the schema structure for a health record

Let us consider the example tree in Figure 2 and Figure 3 for a better understanding of the above expressions.

- */Records*: this expression addresses the node *Records*.
- */Records/**: this expression specifies all the children nodes of *Records*.
- */Records//Medical*: this expression addresses the descendant nodes of *Medical*.
- */Records/Patient[@Name=Joe]*: this expression specifies the node *Patient* that has an attribute named *Name* with the string value of "*Joe*".

4.3 Semantics of Access Control Rules

Most of the existing approaches are distinguishable from the semantics of their access control policy which contains the following issues: the scope of access control rules and conflict resolution such as *grant-overwrites* and *deny-overwrites* algorithms.

4.3.1 Scope of Rules

By the hierarchical nature of XML documents, the definition of propagation scope of rules becomes an important basis to implement a fine-grained access control model. As mentioned earlier, the scope of rules consists of four elements PL, PR, NL, and NR.

- **PL** (Positive Local): the scope of grant rule is propagated to only the specified node.
- **PR** (Positive Recursive): the scope of grant rule can be applied to the specified node and its descendants.
- **NL** (Negative Local): the scope of deny rule is propagated to only the specified node.
- **NR** (Negative Recursive): the scope of deny rule can be applied to the specified node and its descendants.

For example, the subject named Bob can obtain the read permission on the object *Diagnosis* if the policy is defined as:

<B, /Records/Patient[@Name=Joe]/Medical, r, +, PR>

then the scope of rule to the specified node *Medical* by the path expression can be propagated to the node *Medical* and all its descendants. If the policy is specified as below:

<B, /Records/Patient[Name=@Joe]/Medical, r, –, NL>

which means that the scope of access deny to the object is restricted to only the local node *Medical*, not including its descendants.

4.3.2 Conflict Resolution

As mentioned previously in the definition of ACP, it can allow both grant and deny access rules (denoted by + and –, respectively). Therefore, simultaneously, a node is permitted by a grant rule and rejected by a deny rule. In this case, the two rules are in conflict with their semantics, so an algorithm such as *grant overwrites* or *deny overwrites* is normally used to resolve rule conflicts.

- *grant overwrites:* this algorithm grants a request to a node if a grant access rule to the same node is defined.
- *deny overwrites:* this algorithm denies a request to a node if a deny access rule to the same node is defined.

For instance, assume that the following two rules are defined for the subject *Bob*:

<B, /Records/Patient[@Name=Joe]/Medical, r, + , PL>

<B, /Records/Patient[@Name=Joe]/Medical, r, –, NL>

If we adopt the algorithm *deny overwrites*, the deny rule takes precedence over the grant rule. In other words, Bob has the deny permission to read the local node *Medical*. On the other hand, if we select the algorithm grant overwrites, the grant rule takes precedence over the deny rule. That is, Bob can perform the read action on the local node *Medical*.

5. Communicating Sequential Process (CSP) Models

In this section, we demonstrate the method for modeling a schema, a query, and access control on the EHR system in a tree structure in the CSP language. Further information regarding the notation and semantics used in the CSP language can be found in (Hoare, 1985; Roscoe, 1997).

5.1 Modeling of Schema

We assume that the conditional expression written in XPath is always satisfied when constructing CSP models. For example, the expression */Records/Patient[ID="Joe"]* is simplified to */Records/Patient* for the CSP model. Thus, the schema *S* for the illustrative example shown in Figure 2 can be modeled in CSP as shown below:

$S = records \rightarrow patient \rightarrow medical \rightarrow (doctor \rightarrow STOP \,\square\, diagnosis \rightarrow STOP \,\square\, prescription \rightarrow STOP)$

In CSP notation, the *STOP* process indicates the termination of a process and □ denotes the deterministic choice of a process.

5.2 Modeling of Query

A query expression *q*, can be modeled in CSP language by interpreting an XPath expression similar to the method described in Section 4.2. For example, the query expression */Records/Patient[@Name=Joe]/Medical/** in the illustrative example can also be modeled as shown below:

$Q = records \rightarrow patient \rightarrow medical \rightarrow (doctor \rightarrow STOP \,\square\, diagnosis \rightarrow STOP \,\square\, prescription \rightarrow STOP)$

5.3 Modeling of Access Control Policy

The access control policy (ACP) is composed of access control rules. The ACP restricts the view of a document by enforcing its access control rules. We construct a CSP model for access control using both an interleaving parallel operator ||| to apply the *grant-overwrites* semantics to the ACP as shown in the process ACP_{grant}, and an interface parallel operator |[event]| to apply the *deny-overwrites* semantics to the ACP as shown in the process ACP_{deny}.

events = {records, patient, medical, doctor, diagnosis, prescription}

RULE1 = records → patient → medical → diagnosis → STOP

RULE2 = records → patient → medical → prescription → STOP

RULE3 = records → patient → medical → STOP

$ACP_{grant} = RULE1 \,|||\, RULE2 \,|||\, RULE3$

$ACP_{deny} = (RULE1 \;[|events|]\; RULE2) \;[|events|]\; RULE3$

6. Verification Using Failure-Divergences Refinement (FDR)

An important feature of formal specification of the authorized view is its ability to verify the security of the configuration of an ACP. A policy configuration is *secure* when the rules of the policy not only prevent the interception of data but also provide access to the required information. The basic approach of verifying security properties in FDR model checker is to compare whether authorized view given to a user is trace-equivalent to the policy and schema.

Example 1. *Given a security requirement that defines the restricted information that a user is allowed to access, a schema-level access control model is secure if it meets the security requirement.*

Model Checking 1. *Given a security requirement process SR1 and an access control model M consisting of a schema process S and a policy process ACP, M is secure if tr(S)∩ tr(ACP) ⊆ tr(SR1).*

assert SR1 $\sqsubseteq_T$ *S* |[A]| *ACP*

where *A* is the event set between *S* and *ACP* such that $A = \alpha(S) \cup \alpha(ACP)$. Furthermore, *assert* is the reserved keyword for model checking, and $\sqsubseteq_T$ denotes the trace refinement checking in FDR (Formal Systems, 1999): *SR1* $\sqsubseteq_T$ *S*|[A]| *ACP* ⇔ *tr(S) ∩ tr(AC) ⊆ tr(SR1)*

The parallel process *S* |[*A*]| *ACP* represents a schema-level access control model for the XML document. For example, we assume that the following is the path expression that specifies the security requirement for the patient record:

/Records/Patient[@Name=Joe]/Medical/Diagnosis

Then, we can express the process *SR1* after interpreting the above path expression:

SR1 = records → patient → medical → diagnosis → STOP

After the FDR tool is executed, the following counterexample to the model *S* |[*A*]| *ACP* is obtained when the *grant-overwrites* algorithm is assumed to be applied:

< records, patient, medical, prescription >

This trace event indicates that access to the node *Prescription* is not allowed in the above *schema-level* access control model.

Example 2. *The result of a query in an access control system is secure if an authorized view of the result of the query not only prevents the interception of data but also provides access to required information.*

Model Checking 2. *Given a query process Q in a schema-level access control model M, the query answering Q |[A]| M* is secure if *tr(Q) ∩ tr(M) ⊆ tr(SR2).*

Let us consider an example in which there exists a security requirement (represented by the process *SR2*) that specifies that a user Bob should not have access to the node *Doctor*. We further assume that there exists a query *q* (represented by the process *Q*) against a patient record as given below:

*q: /Records/Patient[@Name=Joe]/Medical/**

Then, the process *Q* for the query is given as follows:

Q = records → patient → medical → (doctor → STOP □ diagnosis → STOP □ prescription → STOP)

The process *SR2* for the security constraint is expressed as follows:

SR2 = records → patient → medical → (diagnosis → STOP □ prescription → STOP)

Finally, the process $Q \,|[A]|\, M$ for a query answering for the patient record is as follows:

Q |[A]| M = Q |[A]| (S |[A]| ACP)

where A is the event set between Q and M such that $A = \alpha(Q) \cup \alpha(M)$.

The condition $tr(SR2) \subseteq tr(Q) \cap tr(M)$ is satisfied in the above example. Therefore, we can say that the schema-level access control model satisfies the security requirement. However, maintaining the consistency of the ACP across several distributed database systems becomes increasingly complicated. It is possible that the alteration of a single rule in an ACP can put confidential information (e.g., the patient record) at risk. For example, let us assume that process *RULE4* for a new access control rule is added in the ACP as given below:

RULE4 = records → patient → medical → doctor → STOP

When the *grant-overwrites* algorithm is applied, the process *ACP'* for the altered ACP is given as follows: *((RULE1 [ ] RULE2) [ ] RULE3) [ ] RULE4)*

Thus, the process of query answering over the altered schema-level access control model M' is given as follows: Q |[A]| (S |[A]| ACP')

In this case, we observe that the set of trace events for the processes Q and M' is not a subset of that of the process *SR2*, that is, $tr(Q) \cap tr(M') \subseteq tr(SR2)$. After running the FDR tool, the following counterexample can be shown for Q |[A]| M':

< *records, patient, medical, doctor* >

This trace event means that a user Bob can access the node *Doctor* when a single rule in the existing policy is altered.

7. Conclusion

In this paper, we presented a methodology for describing the semantics of access control for health records in XML documents using a CSP-like process algebra language.

First, we demonstrated a method for expressing a permissible/deniable path to the targeted XML nodes by interpreting the XPath expression. Second, we illustrated how the semantics of confliction resolution can be captured by parallel and interleaving operators. Finally, we examined the possibility of verifying the security of access control policies using automated model checking tools such as FDR.

Given CSP models for a schema, query and ACP for XML-based health records, our verification methodology can not only determine whether the requested query is permitted by the schema-level ACP, but also show a hierarchical path if access to data is allowed.

References

BIRD, L., GOODCHILD, A. and TUN, Z. (2003): Experiences with a two-level modeling approach to electronic health records, *Journal of Research and Practice in Information Technology*, 35(2): 121–138.

BOAG, S., CHABERLIN, D., FERANDEZ, M.F., FLORESCU, D., ROBIE, J. and SIMEON, J. (2002): XQuery 1.0: An XML query language. W3C working draft 16. http://www.w3.org/TR/xquery.

BRAY, T., PAOLI, J. and SPERBERG-McQUEEN, C.M. (1998): Extensible Markup Language (XML) 1.0. W3C Recommendation, http://www.w3.org/TR/REC–xml.

CLARK, J. and DEROSE, S. (1999): XML Path language (XPath) Version 1.0, W3C Recommendation, http://www.w3c.org/TR/xpath.

DAMIANI. E., DE CAPITANI DI VIMERCATI, S., PARABOSCHI, S. and SAMARATI, P. (2002): A Fine-Grained Access Control System for XML Documents, *ACM Trans. on Information and System Security*: 149–202.

FISLER, K., KRISHNAMURTHI, S., MEYEROVICH, L.A. and TSCHANTZ, M.C. (2005): Verification and Change–Impact Analysis of Access-Control Policies, in *Proc. 27th Int. Conf. on Software Engineering (ICSE'05)*, 196–205.
FORMAL SYSTEMS (1999): FDR2 User Manual.
FUNDULAKI, I. and MARX, M. (2004): Specifying Access Control Policies for XML Documents with XPath, in *Proc. 9th Symposium on Access Control Model and Technologies (SACMAT'04) Conference*, 61–69.
GODIK, S. and MOSES, T. (2003): eXtensible Access Control Markup Language (XACML) version 1.0, Technical Report, OASIS.
GOWADIA, V. and FARKAS, C. (2005): Tree Automata for Schema-level Filtering of XML Associations, *Journal of Research and Practice in Information Technology*, 38(1): 97–109.
HOARE, C.A.R. (1985): *Communicating Sequential Processes*, Prentice Hall, NJ.
HORI, M. and OHASHI, M. (2005): Applying XML Web Services into Health Care Management, *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*.
HUGHES, G. and BULTAN, T. (2004): Automated Verification of Access Control Policies, Technical Report 2004-22.
ISO/TR 20514 (2005): Health informatics – Electronic health record – Definition, Scope, and Context, Jan 22.
PASUPATHINATHAN, V., PIERZYK, J. and WANG, H. (2006): Security Analysis of Australian and E.U. E-passport Implementation, *Journal of Research and Practice in Information Technology*, 38(1): 19–29.
QUNOO, H. and RYAN, M. (2010): Modelling Dynamic Access Control Policies for Web-Based Collaborative Systems. *Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*. LNCS 6156: 295–302.
ROSCOE, A.W. (1997): *The Theory and Practice of Concurrency*, Prentice Hall, NJ.
SACHDEVA, S., MCHOME, S. and BHLLA, S. (2010): Web Services Security Issues in Healthcare Applications, *9th IEEE/ACIS International Conference on Computer and Information Science*, 91–96.
SHAIKH, S. and BUSH, V. (2008): Analysing the Woo-Lam Protocol Using CSP and Rank Functions, *Journal of Research and Practice in Information Technology*, 40(3): 187–205.
VAN EIJK, P.H.J., VISSERS, C.A. and DIAZ, M. (1989): The Formal Description Technique LOTOS: Results of the Esprit Sedos Project, Elsevier Science Inc. New York, NY.
XIE, L., YU, C., LIU, L. and YAO, Z. (2010): XML-based Personal Health Record System, *3rd International Conference on Biomedical Engineering and Informatics (BMEI 2010)*, 2536–2540.
ZHANG, N., RYAN, M. and GUELEV, D.P. (2005): Evaluating Access Control Policies Through Model Checking, in *Proc. 8th Conference on Infomation Security (ISC)*, 446–460.

Biographical Notes

Ji-Yeon Lee *is an associate professor in the Department of Business Administration in Dongnam Health University. She received her MSc and PhD degrees from the Department of Computer Science and Engineering, Korea University. Her research interests are formal methods, e-commerce, testing, network security.*

Ji-Yeon Lee

Jin-Young Choi *is a professor in the Department of Computer Science and Engineering in Korea University. He received his BSc degree in computer engineering from Seoul National University in 1982, his MSc degree in computer science from Drexel University in 1986 and acquired a PhD degree in computer science from the University of Pennsylvania in 1993. His research interests are real-time computing, formal methods (formal specification, formal verification, model checking), process algebras, security and software engineering.*

Jin-Young Choi

Il-Gon Kim *has worked for the Attached Institute of ETRI as a researcher since September 2012. He worked at the Korea Internet & Security Agency from January 2007 to August 2012. He worked for IRISA/INRIA as a postdoctorate researcher during 2005 and 2006. He received his MSc and PhD degrees from the Department of Computer Science and Engineering from Korea University. His research interests are formal methods, process algebra, CSP, Casper, FDR, security protocol, and security model.*

Il-Gon Kim

Ki-Seok Bang *is an associate professor in the College of General Education in Hallym University. He received his MSc and PhD degrees from the Department of Computer Science and Engineering from Korea University. His research interests are formal methods, model checking, secure coding and software engineering.*

Ki-Seok Bang