Recommended Paper

Information Flow Control in Role-based Model for Distributed Objects

Masashi Yasuda,† Keiji Izaki,† Katsuya Tanaka† and Makoto Takizawa†

Various kinds of distributed applications are realized by using object-oriented technologies. Object-oriented systems are composed of multiple objects which cooperate to achieve some objectives by passing messages. Object-oriented technologies are used to realize the interoperability among the applications. In addition to realizing the interoperability, it is essential to make the system secure. The secure system is required to not only protect objects from being illegally manipulated but also prevent illegal information flow among objects. In this paper, we discuss role-based an access control model in the object-oriented systems and how to resolve illegal information flow by using the roles.

1. Introduction

Various kinds of object-oriented systems like object-oriented database systems [2] and JAVA [11] are widely used to design and implement applications. Object-oriented systems are composed of multiple objects which cooperate to achieve some objectives by passing messages. An object is an encapsulation of data and methods for manipulating the data. The Common Object Request Broker Architecture (CORBA) [14] is now getting a standard framework for realizing the interoperability among various kinds of distributed applications. In addition to realizing the interoperability, secure systems are required to not only protect objects from illegally being manipulated but also to prevent illegal information flow [4),7),15)] among objects in the system.

In the basic access control model [12], an access rule is specified in a form $\langle s,\ o,\ t\rangle$ which means that a subject s is allowed to manipulate an object o in an access type t. A pair $\langle o, t\rangle$ is an *access right* granted to the subject s. Only the access request which satisfies the authorized access rules is accepted to be performed. However, the access control model implies the *confinement* problem [13], i.e., illegal information flow may occur among subjects and objects. In order to make every information flow legal in the system, the *mandatory lattice-based* access control model [1),4),15)] is proposed. Here, objects and subjects are classified into security classes. The legal information flow is defined in terms of the *can-flow* relation [4] between classes of objects and subjects. In the mandatory model, the access rules are specified by the authorizer so that only the legal information flow occurs. For example, if a subject s reads an object o, information in o flows to s. Hence, s can read the object o only if a *can-flow* relation from o to s is specified. In the discretionary model [3),5),7)], the access rules are defined in a distributed manner while the mandatory access rules are specified only by the authorizer in a centralized manner. For example, the access rules can be granted to other subjects in the relational model [17]. In the role-based model [8),16),19)], a *role* is defined to be a collection of access rights, which shows a job function in the enterprise. The access rule is specified by granting subjects roles which show jobs assigned to the subjects.

The traditional access control models discuss what object can be manipulated by what subject in what access type. The authors [18),20)] newly propose a *purpose-oriented* model which takes into account a *purpose* concept showing why each subject manipulates objects in the object-based system. The object-based system is a restricted version of the object-oriented system where inheritance hierarchy is not supported. The purpose concept is modeled to be a method which invokes another method in the object-based system. In the object-based system, methods are invoked in a nested manner.

In this paper, we discuss how to incorporate the role concepts into the purpose-oriented model in an object-oriented system

† Department of Computers and Systems Engineering, Tokyo Denki University

The initial version of this paper was presented at the DPS workshop held on Dec. 1999, which was sponsored by SIGDPS. This paper was recommended to be submitted to the Journal of IPSJ by the chairperson of SIGDPS.

where methods are invoked in the nested manner. Then, we discuss information flow to occur among objects if the objects are manipulated by transactions with the roles. We define a $safe$ set of roles where no possible illegal information flow occurs. In addition, we discuss an interpretive algorithm to check if an illegal information flow possibly occurs each time a method is issued to an object.

In Section 2, we present the object-oriented system. In Section 3, we discuss access rules and roles in the object-oriented model. In Section 4, we discuss how to resolve illegal information flow by using the role concepts.

2. System Model

An object-oriented system is composed of classes and objects. A class is an encapsulation of attributes and methods for manipulating the attributes. Objects are created by giving values to the attributes of the class. The objects are instances of the class. A method of an object is invoked by sending a request message to the object. The method specified by the message is performed on the object. On completion of the method, the response is sent back to the sender object of the message.

A new class s can be derived from an existing class c. The class s is a *subclass* of the class c. The subclass s inherits attributes and methods of the class c. There is an *is-a* relation from s to c. A subclass may *override* the attributes and methods from the class. In **Fig. 1**, a pair of classes *Clock* and *Alarm* are superclasses of a class *AlarmClock*. *AlarmClock* inherits attributes *time* and *setAlarm* from *Clock* and *Alarm*, respectively. *AlarmClock* also inherits the method *show* from *Clock* and the other methods *set* and *ring* from *Alarm*.

In the object-oriented system, a *subject* shows a user or an application program. A subject is an active entity in the system, which can issue access requests to objects. The subject manipulates objects by invoking their methods. On the other hand, an *object* is a passive entity. A method is performed on an object only if the method is invoked. The method invoked may invoke further methods of other objects, i.e., invocation is *nested*. Thus, relation of subjects and objects are relative. In object-oriented systems, everything is perceived to be an object. An object which plays a role of subject is referred to as *client* object. An object which plays a role of object is a *server* object. From here, *subjects* mean *client* objects and *object* indicate *server* objects.

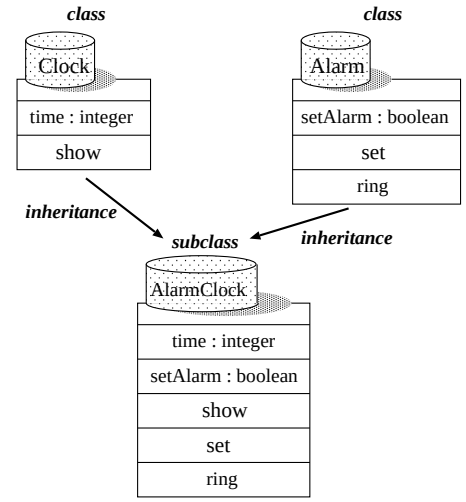

Fig. 1 Class hierarchy.

3. Role-based Purpose-oriented Model

3.1 Roles

Each subject plays some *role* in an organization, e.g., designer and clerk in a company. A role represents a job function in the organization. In the role-based model [8),16),19)], a *role* is modeled in a set of *access rights*. An access right means an approval of a particular mode of access, i.e., methods to an object in the system. That is, a role is specified as a pair $\langle o, t\rangle$ of an object o and a method t meaning which method t can be performed on which object o. Only a subject s granted an access right $\langle o, t\rangle$ is allowed to manipulate the object o by issuing the method t. A role r is a collection of access rights.

Let R be a set of roles in the system. In the role-based model, a subject s is granted a role which shows its job function. On the other hand, the subject s is granted access rights in the access control model. Here a subject s is referred to as *bound* with the role r. The subject s is also referred to *as belong* to the role r. This means that the subject s can perform a method t on an object o if $\langle o, t\rangle \in r$. For example, a role *chief* is $\{\langle book, read\rangle, \langle book, enter\rangle\}$ and *clerk* is $\{\langle book, read\rangle\}$ in **Fig. 2**. A person A who works as a chief in the company is granted the role *chief* in the organization. A *clerk* B is granted *clerk*. Thus, it is easy to grant access rights to subjects in the role-based model.

In a role-based access control model, each

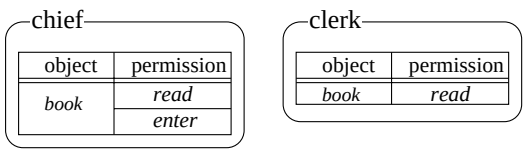

Fig. 2 Examples of roles.

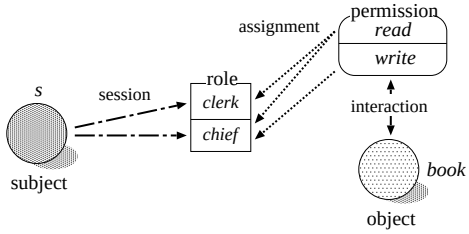

Fig. 3 Role-based access.

subject s can manipulate an object o by a method t only if the subject s is granted a role including an access right $\langle o, t\rangle$. The method is performed on the object on receipt of the request message. If a subject s would like to exercise the authority of a role r which s belongs to, the subject s establishes a *session* to the role r. For example, an object *book* supports a pair of methods *read* and *write* as shown in **Fig. 3**. There are two roles *clerk* and *chief* shown in Fig. 2. A subject s can perform *write* on the object *book* while a session between s and a role *chief* is established. Even if the subject s belongs to *chief*, s cannot perform *write* on the object *book* if a session between s and *chief* is not established. The authority of a role r can be exercised only while a subject s establishes a session to the role r.

3.2 Purpose-oriented Model

The purpose-oriented access control model [18),20)] newly introduces a *purpose* concept to the access control model. A purpose shows why each subject s manipulates an object o by invoking a method t of the object o. In the object-based system, methods are invoked in the nested manner. Suppose that a subject s invokes a method t_1 of an object o_1 and then t_1 invokes a method t_2 of another object o_2. In the purpose-oriented model, the purpose is modeled to be the method t_1 invoking t_2 of o_2 while the access control model specifies whether or not o_1 can manipulate o_2 by t_2. For example, let us consider a person s who would like to withdraw money from a *bank*. In the access control model, a subject s can withdraw money from *bank* if an access rule

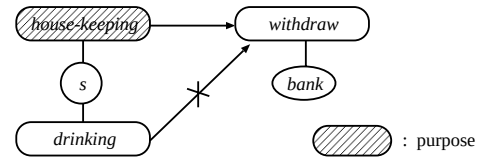

Fig. 4 Purpose-oriented model.

$\langle s, bank, withdraw\rangle$ is authorized independently of for what s spends the money. In our real life, the subject s can get money from *bank* for purpose of *house-keeping* but not for *drinking*. An access rule $\langle s$: *house-keeping*, *bank*: *withdraw*$\rangle$ is specified where a method *house-keeping* of the subject s shows the purpose for which s can withdraw money from *bank* (**Fig. 4**). However, $\langle s$: *drinking*, *bank*: *withdraw*$\rangle$ is not authorized.

A role is specified in a collection of access rights in the role-based model [8),16),19)]. We extend the purpose-oriented access control model by incorporating the role concept. In the object-oriented system, methods are invoked in a nested manner. Here, suppose that a subject s invokes a method t_1 on an object o_1 and then t_1 invokes another method t_2 on an object o_2. Here, suppose the subject s is granted an access right $\langle o_1, t_1\rangle$. In one way, only if the subject s is granted an access right $\langle o_2, t_2\rangle$, the method t_1 can invoke t_2. However, it is cumbersome for each object o to specify which object can manipulate the object o. In relational database systems [17)], the ownership chain method is adopted. Here, if the object o_2 has the same owner as the object o_1 or the subject s is granted an access right $\langle o_1, t_1\rangle$, the method t_1 can invoke the method t_2 even if s is not granted an access right $\langle o_2, t_2\rangle$. Otherwise, the method t_1 is allowed to invoke t_2 only if s is granted an access right $\langle o_2, t_2\rangle$. Suppose the response of the method t_2 carries some data stored in the object o_2. On receipt of the response, the data carried by the response may be stored in the persistent storage of the object o_2 while the method t_1 is being performed by using the response. This means, information in the object o_2 flows to the object o_1 through the invocation. The data may be brought to other objects by other invocations. By using the ownership chain method, illegal information flow may occur. In this paper, we assume that the system is composed of multiple autonomous objects, that is, objects have different owners. Furthermore, it is difficult, maybe impossible for each autonomous object to grant access rights to other objects since the objects are dynami-

cally autonomously changed. In this paper, we take an object pairwise approach where access rules are specified for a pair of autonomous objects.

Suppose an object o_i supports a method t_i which invokes other methods. Each method t_i of an object o_i is granted a role $r_i = \{\langle o_{i1}, t_{i1}\rangle, \ldots, \langle o_{ih_i}, t_{ih_i}\rangle\}$. This means, the method t_i is allowed to invoke a method t_{ij} of an object o_{ij} (for $j = 1, \ldots, h_i$). In turn, the method t_{ij} may be granted a role $r_{ij} = \{\langle o_{ij1}, t_{ij1}\rangle, \ldots, \langle o_{ijh_{ij}}, t_{ijh_{ij}}\rangle\}$. The method t_{ij} can invoke a method t_{ijk} of an object o_{ijk} if the method t_{ij} is granted the role r_{ij}. An access rule has to show in what role the method t_i of the object o_i is bound to the role r_i.

[Purpose-oriented Role-based Access (POR) Rule] An access rule $\langle r\colon\ o_i\colon\ t_i,\ r_i\rangle$ means that a method t_i of an object o_i is invoked in a role r and t_i can invoke methods specified in a role r_i. □

4. Information Flow Control

4.1 Illegal Information Flow

In the role-based access control model, subjects are allowed to manipulate objects based on roles to which the subjects belong. However, illegal information flow among objects may occur. For example, there are a pair of objects o_1 and o_2 each of which supports a pair of methods *read* and *write* (**Fig. 5**). There are two roles r_1 and r_2, where $r_1 = \{\langle o_1, read\rangle, \langle o_2, write\rangle\}$ and $r_2 = \{\langle o_2, read\rangle\}$. Suppose that a subject s_1 invokes *write* on the object o_2 after invoking *read* on the object o_1 by the authority of the role r_1. This means that the subject s_1 may write data obtained from o_1 to o_2. The subject s_2 can read data in o_1 even if an access right $\langle o_1, write\rangle$ is not authorized in a role r_2. This is the *confinement problem* [13] pointed out in the basic access control model, i.e., illegal information flow might occur. In addition, a subject can be granted multiple roles in the role-based model even if they can play only one role at the same time. Suppose that a subject s belongs to two roles r_1 and r_2. The subject s obtains information from o_1 as the role r_1 and then stores some of the information into o_2 as another role r_2. Here, information in o_1 flows to o_2.

As discussed here, *write* brings information into an object while information in the object flows out by performing *read*. Hence, we classify each method t supported by each object o_i with respect to the following points:

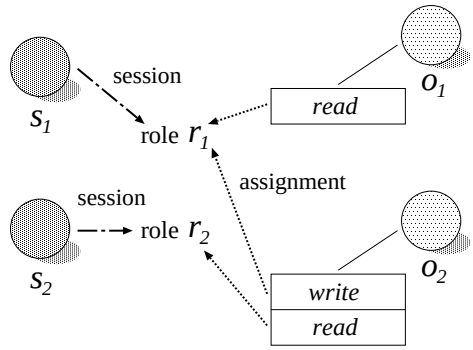

Fig. 5 Illegal information flow.

1. whether or not a value of an object o_i is output by the method t.
2. whether or not a value of a_i in o_i with input parameter is changed by the method t.

The methods are classified into four types: m_R (*out* type), m_W (*into* type), m_{RW} (*in-out* type), and m_N (*neutral*). An m_R method t means that the method t outputs a value but does not change the object o_i. A *display* method of a *video* object is an example of m_R type. An m_W method t means that the method t does not output but changes the object o_i. A *count-up* method of a *counter* object is an example of m_W type. An m_{RW} method outputs a value and changes o_i. A *modify* method is an example of m_{RW} type. The method m_N neither outputs a value nor changes the object o_i.

[Example 1] Let us consider a simple example about information flow between a pair of objects o_i and o_j as shown in **Fig. 6**. A role r_1 is $\{\langle o_i, t_{1i}\rangle, \langle o_j, t_{2j}\rangle\}$. Other roles r_2 and r_3 are $\{\langle o_i, t_{2i}\rangle\}$ and $\{\langle o_j, t_{3j}\rangle\}$, respectively. The method types of t_{1i} of the object o_i is an m_R type. The methods t_{2i} is m_W. The method t_{1j} is m_{RW}. The method t_{3j} is m_W. Each object has an access list composed of tuples of role, method, and method type. A subject s is now in a session with a role r_i. Here, the subject s can invoke a pair of methods t_{1i} and t_{1j} classified into m_R on the object o_i and m_W on o_j by the authority of the role r_{1i}, respectively. Suppose the subject s obtains information from the object o_i through the m_R method t_{1i}, e.g. *display*. It is critical to discuss whether or not the subject s can invoke the m_W method t_{1j} on the object o_j after the invocation of the m_R method t_{1i} on the object o_i in order to pre-

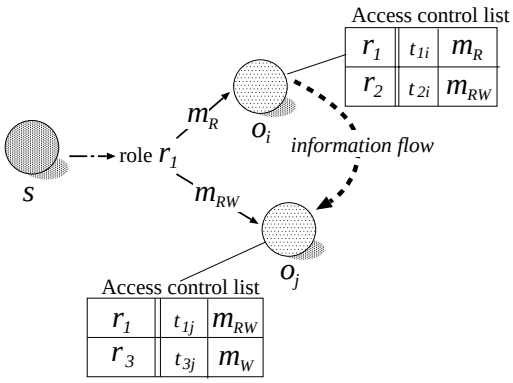

Fig. 6 Information flow among objects.

vent illegal information flow. Here, the information in the object o_i flows into the object o_j. If the information flow from o_i to o_j does not violate the other roles, the method t_{1j} can be performed. A subject playing the role r_3 can carry information to the object o_j but cannot derive information from o_j. Hence, even if the information flows from o_i to o_j, there is no subject who is granted an access right to derive data from o_j. □

We discuss whether a set of roles authorized are safe or not, i.e., no illegal information occurs. Let $R(o)$ be a set of roles which include access rights on an object o.

[**Definition**] Information in an object o_i *possibly flows* to another object o_j in a role r ($o_i \xrightarrow{r} o_j$) iff

1. $r \in R(o_i) \cap R(o_j)$, $\langle o_i, t_i \rangle \in r$, $\langle o_j, t_j \rangle \in r$, t_i is m_R or m_{RW} type, and t_j is m_{RW} or m_W type.
2. $o_i \xrightarrow{r} o_k \xrightarrow{r} o_j$ for some object o_k. □

The relation "$o_i \xrightarrow{r} o_j$" means that information derived from the object o_i may flow into another object o_j if some subject is bounded with the role r. "$o_i \rightarrow o_j$" if there is some role r such that $o_i \xrightarrow{r} o_j$.

[**Definition**] A flow relation "$o_i \xrightarrow{r} o_j$" is *safe* for a pair of objects o_i and o_j iff for every role r' in $R(o_j)$ such that $\langle o_j, t_j \rangle \in r'$ and t_j is m_R or m_{RW} (**Fig. 7**),

1. $r' \notin R(o_i)$.
2. $\langle o_i, t_i \rangle \in r'$ and t_i is m_R or m_{RW}. □

Suppose some subject s is bounded with a role r such that $o_i \xrightarrow{r} o_j$. If s manipulates an object o_j by an m_{RW} method after manipulating another object o_i by an m_R method, information in o_i flows into o_j. If "$o_i \xrightarrow{r} o_j$" is not safe, there is such a role r' that o_j can be manipulated by an m_R method but o_i cannot be manipulated by an m_R method in r' (**Fig. 8**). Hence, if another subject s' is bounded with the role r', s' can get information derived from the object o_i by manipulating the object o_j although s cannot access to o_j.

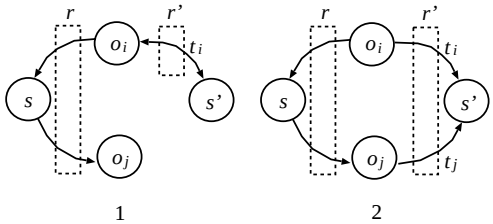

Fig. 7 Safe role.

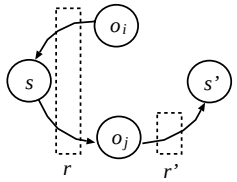

Fig. 8 Unsafe role.

[**Definition**] A role r is $safe$ iff $o_i \xrightarrow{r} o_j$ is safe for every pair of objects o_i and o_j in the role r. □

It is straightforward for the following theorem to hold from the definitions.

[**Theorem**] If every role is safe, there is no illegal information flow. □

4.2 Safe Roles

In the mandatory model, every role is defined so as to be safe when the role is defined by the authorizer. In the discretionary model, roles are defined by subjects who are granted an access right to define the roles. Hence, roles are dynamically created and dropped. We consider an approach for the discretionary where each request is checked if illegal information flow possibly occurs by performing the method t. Let $AL(o)$ be an access list $\{\langle r, t, type \rangle\}$ given for an object o, where r is a role, t is a method, and $type$ is a type of the method t, i.e., $type \in \{m_R, m_W, m_{RW}, m_N\}$. An access list $AL(o)$ is maintained for each object o. For example, a tuple $\langle r, t, type \rangle$ is added to the access list $AL(o)$ of an object o if an access right $\langle o, t \rangle$ is added in a role r.

Variables AL and OL are manipulated for each subject s as follows:

1. Initially, $AL = \phi$ and $OL = \phi$.
2. If the subject s issues a method t to an

object o, the access list $AL(o)$ is obtained from the object o and $AL := AL \cup AL(o)$.

3. The object o is appended in the tail of the list OL.
4. By using the *safeness condition* to be discussed later, it is checked if illegal information flow might occur after the method t is performed. If no illegal information flow occurs, the method t is performed and the object o is appended into the list OL. Otherwise, t is rejected.

The list OL shows a sequence of objects to which the subject s issues methods. OL is named *object list*. Here, if an object o_i is stored before o_j in the list OL, o_i is referred to as *precede* o_j in OL. By using the object list OL, the method t is performed on the object o if the following condition is satisfied.

[Safeness condition] For every object o_i in the object list OL, the relation "$o_i \xrightarrow{r} o$" is safe for every *role* r which includes access rights on the object o_i and o. □

If the safeness condition is satisfied at step 4, the method t is performed on the object o. Otherwise, t is rejected since there might occur illegal information flow from some object in OL to the object o after the method t is performed. Even if some subject derives information from the object o_j, the subject cannot obtain information of o_i unless the subject is granted an access right to manipulate o_i. In this algorithm, the safeness condition is checked for a subject s each time the subject s issues a method. It depends on a sequence of methods, i.e., the object list OL whether or not the safeness condition is satisfied. For example, suppose the subject s performs an m_W method t_{1j} on an object o_j before an m_R method t_{1i} on another object o_i. Suppose there is a role r' including an access right $\langle o_j, t_j \rangle$ where t_j is an m_R method. Here, a flow relation "$o_i \xrightarrow{r} o_j$" is not safe according to the definition because there is a possibility that the illegal information flow occurs as shown in **Fig. 9** (1). However, there is no information flow from o_i to o_j since s manipulates o_j before o_i (Fig. 9 (2)). Thus, even if some role is not safe, illegal information flow does not occur if every method is performed according to the algorithm.

It is straightforward for the following theorem to hold from the definitions.

[Theorem] If every method is performed according to the algorithm, no illegal information flow occurs. □

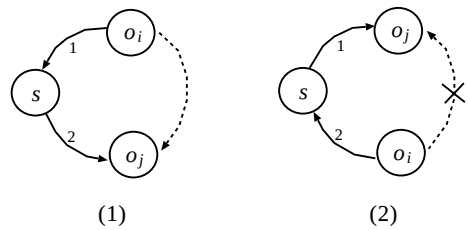

Fig. 9 Information flow.

In the algorithm presented here, an access list AL and object list OL are manipulated to check the safeness condition each time a method is performed on an object. The more number of methods a subject s performs, the larger AL and OL are getting. This implies larger computation overhead. On the other hand, some methods are allowed to be performed according to the algorithm even if a set of roles are not safe as shown in the theorem. Thus, more number of methods can be performed by using the algorithm. There is some tradeoff between number of methods which can be performed and the performance overhead. We are now evaluating the performance of the algorithm and would like to present it in another paper.

5. Concluding Remarks

This paper presented an access control model for object-oriented systems with role concepts. Roles are higher level representation of access control models. We defined a role concept to mean what method can be performed on which object. Roles are incorporated in the purpose-oriented model. Furthermore, we discussed how to control information flow through roles. We defined a set of safe roles where no illegal information flow possibly occurs. We presented the safeness condition to decide whether the roles are safe or not. We also presented the interpretive algorithm to check if each method could be performed, i.e., illegal information flow possibly occurs after the method is performed. By using the algorithm, some methods can be performed depending on in which order a subject performs the methods even if the methods are not allowed to be performed due to the unsafeness of the roles. We are at present evaluating the algorithm presented in this paper.

Acknowledgments This research is partially supported by the Research Institute for Technology, Tokyo Denki University.

References

1) Bell, D.E. and LaPadula, L.J.: Secure Computer Systems: Mathematical Foundations and Model, *Mitre Corp. Report*, No.M74–244, Bedford, MA (1975).
2) Bertino, E. and Martino, L.: Object-Oriented Database Management Systems: Concepts and Issues, *IEEE Computer*, Vol.24, No.4, pp.33–47 (1991).
3) Castano, S., Fugini, M., Matella, G. and Samarati, P.: *Database Security*, Addison-Wesley (1995).
4) Denning, D.E.: A Lattice Model of Secure Information Flow, *Comm. ACM*, Vol.19, No.5, pp.236–243 (1976).
5) Denning, D.E. and Denning, P.J.: *Cryptography and Data Security*, Addison-Wesley (1982).
6) Fausto, R., Elisa, B., Won, K. and Darrell, W.: A Model of Authorization for Next-Generation Database Systems, *ACM Trans. Database Systems*, Vol.16, No.1, pp.88–131 (1991).
7) Ferrai, E., Samarati, P., Bertino, E. and Jajodia, S.: Providing Flexibility in Information Flow Control for Object-Oriented Systems, *Proc. 1997 IEEE Symp. on Security and Privacy*, pp.130–140 (1997).
8) Ferraiolo, D. and Kuhn, R.: Role-Based Access Controls, *Proc. 15th NIST-NCSC Nat'l Computer Security Conf.*, pp.554–563 (1992).
9) Harrison, M.A., Ruzzo, W.L. and Ullman, J.D.: Protection in Operating Systems, *Comm. ACM*, Vol.19, No.8, pp.461–471 (1976).
10) Izaki, K., Tanaka, K. and Takizawa, M.: Authorization Model in Object-Oriented Systems, *Proc. IFIP Database Security* (2000).
11) Gosling, J. and McGilton, H.: *The Java Language Environment*, Sun Microsystems (1996).
12) Lampson, B.W.: Protection, *Proc. 5th Princeton Symp. on Information Sciences and Systems*, pp.437–443 (1971). Also in *ACM Operating Systems Review*, Vol.8, No.1, pp.18–24 (1974).
13) Lampson, B.W.: A Note on the Confinement Problem, *Comm. ACM*, Vol.16, No.10, pp.613–615 (1973).
14) Object Management Group Inc.: The Common Object Request Broker : Architecture and Specification, Rev. 2.1 (1997).
15) Sandhu, R.S.: Lattice-Based Access Control Models, *IEEE Computer*, Vol.26, No.11, pp.9–19 (1993).
16) Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E.: Role-Based Access Control Models, *IEEE Computer*, Vol.29, No.2, pp.38–47 (1996).
17) Sybase, Inc.: Sybase Adaptive Server Enterprise Security Administration (1997).
18) Tachikawa, T., Yasuda, M. and Takizawa, M.: A Purpose-oriented Access Control Model in Object-based Systems, *Trans. IPSJ*, Vol.38, No.11, pp.2362–2369 (1997).
19) Tari, Z. and Chan, S.W.: A Role-Based Access Control for Intranet Security, *IEEE Internet Computing*, Vol.1, No.5, pp.24–34 (1997).
20) Yasuda, M., Higaki, H. and Takizawa, M.: A Purpose-Oriented Access Control Model for Information Flow Management, *Proc. 14th IFIP Int'l Information Security Conf.* (*SEC'98*), pp.230–239 (1998).

(Received November 10, 2000)
(Accepted March 9, 2001)

Editor's Recommendation

The authors propose a novel role-based access control model in the object-oriented systems. Using the proposed access control model, not only the interoperability among the applications but also the system secure can be realized, because this access control mechanism prevents the illegal information flow among objects. The model is of wide application and useful for many members.

(Chairman of SIGDPS Hiroshi Miyabe)

Masashi Yasuda was born in 1974. He received his B.E. degree in computers and systems engineering from Tokyo Denki University, Japan in 1997. Currently, he works NS Solutions Corporation. His research interests include secure distributed systems and computer networks.

Keiji Izaki was born in 1978. He received his B.E. degree in computers and systems engineering from Tokyo Denki Univ., Japan in 2000. He is now a graduate student of the master course in the Dept. of Computers and Systems Engineering, Tokyo Denki Univ. His research interests include distributed database systems and security. He is a student member of IPSJ.

Katsuya Tanaka was born in 1971. He received his B.E. and M.E. degrees in Computers and Systems Engineering from Tokyo Denki University, Japan in 1995 and 1997, respectively. From 1997 to 1999, he worked for NTT Data Corporation. Currently, he is an assistant in the Department of Computers and Systems Engineering, Tokyo Denki University. He received the D.E. degree from Dept. of Computers and Systems Engineering, Tokyo Denki University, Japan, in 2000. His research interests include distributed systems, transaction management, recovery protocols, and computer network protocols. He is a member of IEEE CS and IPSJ.

Makoto Takizawa was born in 1950. He received his B.E. and M.E. degrees in Applied Physics from Tohoku Univ., Japan, in 1973 and 1975, respectively. He received his D.E. in Computer Science from Tohoku Univ. in 1983. From 1975 to 1986, he worked for Japan Information Processing Developing Center (JIPDEC) supported by the MITI. He is currently a Professor of the Dept. of Computers and Systems Engineering, Tokyo Denki Univ. since 1986. From 1989 to 1990, he was a visiting professor of the GMD-IPSI, Germany. He is also a regular visiting professor of Keele Univ., England since 1990. He was a program co-char of IEEE ICDCS-18, 1998 and serves on the program committees of many international conferences. He chaired SIGDPS of IPSJ from 1997 to 1999. He is IPSJ fellow. His research interests include communication protocols, group communication, distributed database systems, transaction management, and security. He is a member of IEEE, ACM, and IPSJ.