コンピュータセキュリティ　14−21
(2001. 7. 25)

必須処理付きセキュリティポリシーのためのアクセス制御モデル

工藤 道治、羽田 知史

日本アイ・ビー・エム(株) 東京基礎研究所
〒242-8502 大和市下鶴間 1623-14
046-215-4642 kudo@jp.ibm.com
046-215-4241 satoshih@jp.ibm.com

あらまし　従来のアクセス制御モデルでは、アクセス制御規則に基づいた評価の結果として、アクセスの許可/ 拒否の二値を返すものとして定義されてきた。しかし、昨今のインターネットアプリケーションではより柔軟なアクセス制御規則を必要とするものが増えている。本稿では二値に追加して必須処理という補足情報も返すモデルを提案する。「アクセスは許可、ただし定められた必須処理(ロギング、暗号化など)を実行しなければならない」のようなセキュリティポリシーを記述することができる。提案方式の定式化を行い、グループなどの木構造上で定義された必須処理付きセキュリティポリシーに基づき、与えられたアクセス要求に対して必須処理を計算するアルゴリズムを示す。提案モデルを用いて、企業におけるセキュリティポリシーが的確に表現できることを具体例を用いて説明する。

キーワード　アクセス制御、セキュリティポリシー、必須処理、暗号化

An Access Control Model for Provision-based Authorization Policies

Michiharu Kudo, Satoshi Hada

IBM Japan, Co., Ltd.　Tokyo Research Laboratory
〒242-8502　1623-14, Shimotsuruma, Yamato-shi
046-215-4642　kudo@jp.ibm.com
046-215-4241　satoshih@jp.ibm.com

Abstract　In most access control systems, authorization is specified using binary decisions, ``yes'' or ``no'', to the access requests resulting in access being permitted or denied respectively. We argue that emerging Internet applications require that this binary decision be extended to ``allow access provided some actions are taken,'' and propose the notion of a ``provisional action'' that tells the user that his request will be authorized provided he (and/or the system) takes certain actions. We formalize an access control model that handles provision-based authorization policies and give an algorithm that resolves a necessary set of provisional actions according to the priorities among hierarchies. We also illustrate how provisional access control policy rules are effectively specified in practical usage scenarios.

Key words　Access Control, Security Policy, Provisional Action, Encryption

1. Introduction

In most access control systems, authorization is specified using binary decisions, "yes" or "no", to the access requests resulting in access being permitted or denied respectively. Recent work such as [8], [4], and [5] aims at providing general frameworks that are capable of supporting flexible and multiple access control policies, but all these models, however, assume that the system either authorizes the access request or denies it.

We propose the notion of a *provisional action* that tells the user that his request will be authorized provided he (and/or the system) takes certain security actions such as signing his statement prior to authorization of his request. Recently there are rapidly increasing numbers of business transactions that require digital signatures and encryption operations in order to provide security properties such as non-repudiatability, integrity, and confidentiality. This paper proposes an access control model for provision-based authorization policies, that provides a simple mechanism for combining access control policy rules and necessary operations in a consistent manner.

The notion of provisional action was first introduced in [6], [7], and [3]. While the first two articles described a similar model, they only provide high-level model definitions or a specific language implementation. This paper gives algorithms that handle propagation policies that work on multiple hierarchies, and presents many comprehensive applications of the provision-based authorization policies. The third article presents a syntax and the semantics of provisional authorizations based on a logic programming model. While the motivations and goals are the same as for the current research, the approach is different. Our model is defined based on the notion of algebraic partial order and lexicographic order. Moreover, they do not present an algorithm for their model, while we give algorithms to resolve necessary provisional actions.

2. Motivating Examples

We present motivating examples that provision-based authorization policies are used effectively.

2.1 Organizational Security Policies

Since organizations usually have to deal with a variety of sensitive information such as marketing strategies and product delivery plans, a set of well-considered security policies is indispensable to protect such information. The following are typical examples of those security policies:

- Business executives can read and write all confidential documents. When executives update confidential documents, the contents must be signed and encrypted. Moreover, the access must be logged.
- Company employees can send e-mail messages to their customers. When employees send e-mail messages to their customers, the content of the mail must be signed and encrypted before sending them out. If the e-mail is sent to other employees, a digital signature is not necessary. Employees can receive e-mail messages sent from the Internet. Upon reception, the e-mail must be scanned using some antivirus utility.
- Employees can update their private information stored in the information server. When employees update their information, it should be encrypted using their public key so that nobody can read the private information of others.

Every policy consists of two portions: usual access control policy rules (e.g. executives can read) and a series of one or more provisions (e.g. the contents must be signed). We define the model syntax and the semantics in Section 4.

2.2 Business-to-Business Transactional Security Policies

These days, business transactions are implemented more and more using open standards such as XML [9], XML digital signature [11], and UDDI [10]. Companies can find a business partner using business registration repositories such as UDDI, place an order in XML format, and change the order via the Internet. Although communication security is guaranteed by using transport layer security standard such as TLS [1], there have been few open standards proposed for specifying transactional security policies. The following are examples of those security policies:

- A customer of a steel company can submit a purchase order using an Internet ordering system. When the customer submits an order, the order document should be digitally signed using the customer's private key. When the system sends a receipt back to the customer, the receipt must be signed with the organization's private key. When the system sends or receives business data, the transactions must be recorded with digital timestamps.
- Registered customers can download digital data (e.g. sample computer graphics) from a digital-content server. When they download such data, the customers' account codes must be invisibly inserted in the digital data for copyright protection purposes.
- Registered customers can make use of Web services (e.g. making an estimate of the cost of something). Whenever they use the Web services, the access must involve charges of some access fee.

3. Authorization System Architecture

The architecture of an authorization system we propose is illustrated in Figure 1. As the basis of our model, we apply the component architecture for access control models known as the Access Control Framework, an international standard [2].

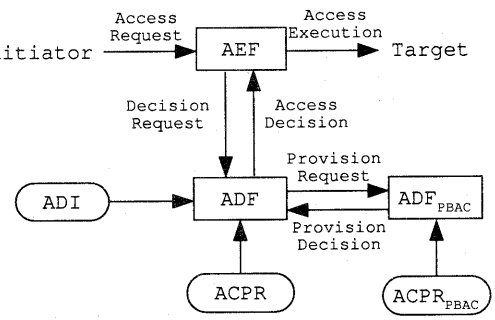

Fig. 1 Authorization System Architecture

The Initiator is either a user or a program that submits an access request to the system. An access request specifies an operation to be performed on Target. The authentication module is beyond the scope of this paper. Access control enforcement functions (AEF) mediate access requests and submit a decision request to access control decision functions (ADF). ADF uses access control policy rules (ACPR) and access control decision information (ADI), such as initiator and context information, to determine whether the access requests should be granted or denied. After making the decision, ADF calls another access control decision function ($\mathsf{ADF_{PBAC}}$) that handles provision-based authorization policies ($\mathsf{ACPR_{PBAC}}$) to retrieve a set of necessary provisional actions. The access decision that consists of a binary decision and optionally a set of provisional actions is returned to AEF and it processes the target resources. A system security officer manages a set of access control policy rules stored in ACPR and $\mathsf{ACPR_{PBAC}}$. Note that AEF, ADF, and ADI are abbreviations defined in the international standard. We define $\mathsf{ACPR_{PBAC}}$, a provision request, and a provision decision in Section 4.2, and $\mathsf{ADF_{PBAC}}$ in Section 4.6.

4. Provision-based Access Control Model

This section describes an access control model for provision-based authorization policies, which enables the authorization system to resolve necessary provisional actions in response to a given decision request. We present primitive components and functions, then define an access control model, $\mathcal{PAM}$.

4.1 Primitive Components

Definition 4.1.1 ($\mathcal{PAM}$ Data Set: PDS): A data set of the $\mathcal{PAM}$ model is a pair (DS, AUTH).

1. DS is a set {OS, GS, RS, AS, FS, PS, IS, US, CM, GM, RM}.
2. AUTH is a 6-ary relation AUTH(OS, GS, RS, AS, FS, PS), which corresponds to a set of access control policy rules $\mathsf{ACPR_{PBAC}}$ defined in Section 4.2.

- OS is a set of target objects organized as a tree structure. OT: $\langle \mathsf{OS}, \leq^{TR} \rangle$ indicates an object hierarchy.
- GS is a set of group names organized as a tree structure. GT: $\langle \mathsf{GS}, \leq^{TR} \rangle$ indicates a group hierarchy.
- RS is a set of role names organized as a tree structure. RT: $\langle \mathsf{RS}, \leq^{TR} \rangle$ indicates a role hierarchy.
- AS is a set of available mode of actions performed on objects. (e.g. read)
- FS is a set of permission flags. FS: $\{+, -\}$
- PS is a set of provisional actions defined in each application domain. Refer to Section 5 with regard to instances of provisional actions.
- IS is a set of target instance objects (e.g. "strategy.pdf".)
- US is a set of user identities in the system (e.g. "Alice".)
- CM is a target instance classification mapping: IS$\rightarrow \mathcal{P}$(OS). The input is a target object instance (IS) and the output is a power set of classifications of the target object to which the target object belongs ($\mathcal{P}$(OS)). For example, security labels such as *Secret* and *Confidential* are instances of OS, a decision request consists of a specific file name, e.g. *strategy.pdf*, and a CM function maps *strategy.pdf* to the *Confidential* security label. When there is no need for such mappings, the CM function just returns the same input value.
- GM is a group membership function: US$\rightarrow \mathcal{P}$(GS). The input is an initiator's identity (US) and the output is a power set of group identities to which the initiator belongs. One user can be a member of multiple groups.
- RM is a role mapping function: US$\rightarrow \mathcal{P}$(RS). The input is an initiator's identity (US) and the output is a power set of role identities that the initiator is activating. One user can be a member of multiple roles.

4.2 Authorization Architecture Interface Definition

We define these data structures for interfaces among the authorization components used in the authorization system architecture in Figure 1.

Definition 4.2.1: (Authorization System Architec-

ture Interface)
Authorization interfaces used in the authorization architecture are a set {DR, AD, PR, PD, ADI, ACPR, $\mathsf{ACPR_{PBAC}}$}.

- Decision request (DR) is a 3-tuple (IS, US, AS).
- Access decision (AD) is a pair (FS, $\mathcal{P}$(PS)).
- Provision request (PR) is a 4-tuple (IS, US, AS, FS).
- Provision decision (PD) is a power set of provisional actions $\mathcal{P}$(PS).
- Access control decision information (ADI) is a 3-tuple (CM, GM, RM).
- Access control policy rule (ACPR) is a 5-ary relation.
  AUTH: AUTH(OS, GS, RS, AS, FS).
- Access control policy rule ($\mathsf{ACPR_{PBAC}}$) is a 6-ary relation.
  AUTH: AUTH(OS, GS, RS, AS, FS, $\mathcal{P}$(PS)).

4.3 Basic Authorization Policies

We present basic authorization policies such as propagation policies and hierarchy priority policies. Each policy is defined in Section 4.6.3 and Section 4.6.2, respectively.

4.3.1 Propagation Policy

When a hierarchical structure is used in an authorization system, two propagation policies are most often applied in existing systems: the *most specific property takes precedence policy*, and the *path traversing policy*.

Most Specific Property Takes Precedence Policy (π^{MS})
This policy is typically found in group-membership propagation where broad and abstract-level security policies are specified for groups that are more abstract. Narrower and concrete-level security policies are specified for groups that are more specific. In this context, a property specified for more specific group overrides the property specified for groups that is more abstract. This policy is built using a provision retrieval function, defined in Section 4.5, where the notion of the most specific property corresponds to a set of maximal elements that indicates the most specific properties of a given hierarchy.

Path Traversing Policy (π^{PT})
The path traversing policy means that the system must check a requested privilege on every element on the authorization hierarchy. For example, suppose that the requested object is file_y, file_y is located at the lower node of dir_a, and dir_a is the root object. The path traversing policy says that whether or not an initiating subject has a privilege on dir_a must be checked in addition to checking the privilege for the requested file_y. This is done by calling the provision retrieval function multiple times for every element along the object path.

4.3.2 Priority Policies for Hierarchy

As for priority policies for hierarchies, we categorized them into three types: *an object hierarchy takes precedence policy, a group hierarchy takes precedence policy,* and *a role hierarchy takes precedence policy.*

Object Hierarchy Takes Precedence Policy (π^{OHP})
This priority policy means that the target object hierarchy takes precedence over the group and role hierarchies. In other words, property propagation is primarily determined according to the object hierarchy. The group and role hierarchies are considered only when more than two properties are specified for the same object.

Group Hierarchy Takes Precedence Policy (π^{GHP})
This propagation policy is the opposite of the π^{OHP} policy. The property propagation is primarily determined according to the group hierarchy. The role and object hierarchies are considered only when more than two properties are specified for the same group.

Role Hierarchy Takes Precedence Policy (π^{RHP})
This propagation policy is similar to the π^{GHP} policy. The property propagation is primarily determined according to the role hierarchy. The group and object hierarchies are considered only when more than two properties are specified for the same role.

4.4 Mathematical Primitives

Definition 4.4.1 (Partial Order): A binary relation $\leq$ that satisfies reflexive, transitive, and asymmetric properties is a partial order. A pair of a set C and a partial order $\leq$ onto C is a partially ordered set and is indicated by $\langle C, \leq \rangle$. $c_1 \leq_C c_2$ means that c_1, that is an element of C, is equal to or smaller than the element c_2. A binary relation $<$ that satisfies non-reflexive and transitive properties is a strict partial order. Other notations are the same as for the partial order.

Definition 4.4.2 (Total Order and Chain): A partial order $\leq^{TO}$ is a total order if a partial order $\leq$ on a set C satisfies $\forall x, y \in C, x \leq y \vee y \leq x$. A subset U of C is a chain if U satisfies a total order.

Definition 4.4.3 (Tree: TR): Let T be a partially ordered set that has a minimum element n_0. Let $L_b^{=}(T)$ be $\{x \in T | x \leq b\}$ for arbitrary element b of T. If any $L_b^{=}(T)$ is a chain for arbitrary elements $b \in T$, then T

is a tree. A tree hierarchy is indicated as $\langle T_1, \leq^{TR} \rangle$. The minimum element n_0 is the Root. Each maximal element is a Leaf. Let U be a subset of a partially ordered set C. An element b of U is a maximal element of U if
$b \leq x \wedge x \in U \Rightarrow x = b$ holds.

Definition 4.4.4: (Lexicographic Order: LO):
Let $\langle X_1, \leq \rangle$, $\langle X_2, \leq \rangle$, ... , $\langle X_n, \leq \rangle$ be partially ordered sets. A lexicographic order $\leq^{LO}$ is defined on the Cartesian product of n partially ordered sets $X_1 \times \cdots \times X_n$ as follows:
$(x_1, \ldots, x_n) \leq^{LO} (x'_1, \ldots, x'_n) \Longleftrightarrow$

$$\begin{cases} x_1 <_{X_1} x'_1 \text{ or} \\ x_1 = x'_1 \text{ and } x_2 \leq_{X_2} x'_2 \text{ or} \\ \vdots \\ x_1 = x'_1 \text{ and } \ldots \text{and } x_{n-1} = x'_{n-1} \text{ and } x_n \leq_{X_n} x'_n \end{cases}$$

A lexicographic ordered set is a pair of a lexicographic order $\leq^{LO}$ and n partially ordered sets $\{X_1, \ldots, X_n\}$, that is indicated by $\langle X_1 \times \cdots \times X_n, \leq^{LO} \rangle$.

4.5 Primitive Functions

We now define three primitive functions, a property relation function (λ), a property extraction function (σ), and a provision retrieval function (ρ). The ρ function is a primary function that receives a query and return a set of provisional actions. The ρ function uses λ and σ as inner functions.

Definition 4.5.1 (Property Relation Function: λ): A property relation is a mapping $\lambda_{\mathsf{AUTH}} : T_1 \times T_2 \times T_3 \rightarrow S_1 \times S_2 \times S_3$ such that: $S_1 = \mathsf{AS}, S_2 = \mathsf{FS}, S_3 = \mathcal{P}(\mathsf{PS})$, $\forall t \in T_1 \times T_2 \times T_3, \forall s \in S_1 \times S_2 \times S_3, s \in \lambda_{\mathsf{AUTH}}(t) \Leftrightarrow AUTH(t, s) \in \mathsf{AUTH}$

Definition 4.5.2 (Property Extraction Function: σ): A property extraction is a mapping $\sigma_i : S_1 \times S_2 \times S_3 \rightarrow S_i$.

Definition 4.5.3 (Provision Retrieval Function: ρ): A provision retrieval is a mapping ρ: $\mathsf{AUTH} \times \mathsf{TS} \times \mathsf{Q}_3 \rightarrow \mathcal{P}(\mathsf{PS})$. The function ρ is defined as $\sigma_3(\lambda_{\mathsf{AUTH}}(v))$ such that: $U = \{u | u \in \mathsf{TS} : \langle T_1 \times T_2 \times T_3, \leq^{LO} \rangle$, $\lambda_{\mathsf{AUTH}}(u)$ satisfies $q_3 \in \mathsf{Q}_3 : S_1 \times S_2\}$, and $V = \{v | v$ is a set of maximal element of $U\}$. The definition of that $\lambda_{\mathsf{AUTH}}(u)$ satisfies q_3 is: $\forall s, s \in \lambda_{\mathsf{AUTH}}(u), \forall j, j \neq 3, \sigma_j(s) = \sigma_j(q_3)$.

A lexicographic order of the Cartesian product $T_1 \times T_2 \times T_3$ (TS) represents a priority policy on hierarchies such that the first hierarchy T_1 has the highest priority and the last hierarchy T_3 has the lowest priority. For example, if a property s_1 is specified for an element $t_1 \in T_1$ and t_1 is hierarchically the highest element in T_1 (the most specific element,) s_1 wins against all other properties specified for lower elements in T_1 (elements that are closer to the Root element) and also against any properties specified for trees other than T_1.

We now present an algorithm that calculates a set of maximal elements of a given set on which a lexicographic order is defined. This algorithm is used in the ρ function when determining a set V.

Algorithm 4.5.1: (Maximal Element Retrieval Algorithm):
This consists of an algorithm body and a maximal function.

Input: $\mathcal{L}, OT_1, OT_2$, and OT_3 such that $\mathcal{L}$ is a set of m-ary relation L($T_1 \times T_2 \times T_3$) and OT_i is $\langle T_i, \leq^{TR} \rangle$, $1 \leq i \leq 3$. L is a relation consisting of only a set of arguments for trees in AUTH.
Output: $\hat{T}_3$: a set of maximal elements.
Variables: $\bar{T}_i, \tilde{T}_i, U_i$: temporary set, $\vec{e}$: (i-1)-ary vector, $\vec{y}$: 2-ary vector, $\hat{L}_i$: the mapping table that maps an i-ary vector to a (3-i)-ary vector (table entries are set at Step 3.3 and Step 4.5.)
Method:
Step 1 $\hat{T}_i = \phi, 1 \leq i \leq 3$
Step 2 For all i, do Step 3 or Step 4.
Step 3 If $i = 1$ then do the following.
Step 3.1 For all $l \in \mathcal{L}$, take the first argument (t_1) and add it to a set U_1.
Step 3.2 call **maximal**($OT_1, U_1, \bar{T}_1$)
Step 3.3 For all $t \in \bar{T}_1$, find a set of L whose first argument matches t. Create a mapping entry as follows: $\hat{L}_1 : t \rightarrow$ a set of arguments that consists of the second element to the last element of the found L.
Step 4 If $i \neq 1$ then for all $\vec{e}_1$ such that $\vec{e}_1$ is a left argument of the table $\hat{L}_{i-1}$, do the following.
Step 4.1 Take the first arguments of the right values of the table entry $\hat{L}_{i-1}(\vec{e}_1)$, and add them to a set U_i.
Step 4.2 call **maximal**($OT_i, U_i, \bar{T}_i$)
Step 4.3 $\tilde{T}_i = \phi$. For all $t_1 \in \bar{T}_i$, add a pair $(\vec{e}_1, t_1)$ to $\tilde{T}_i$.
Step 4.4 Add $\tilde{T}_i$ to $\hat{T}_i$.
Step 4.5 If $i \neq 3$, then for all $t_2 \in \tilde{T}_i$, do the following.
Step 4.5.1 Determine t_3 such that $t_2 = (\vec{e}_1, t_3)$.
Step 4.5.2 Find all $\vec{y}$ such that $(t_3, \vec{y})$ is a member of mapped value of $\hat{L}_{i-1}(\vec{e}_1)$.
Step 4.5.3 If $\vec{y}$ exists, create a mapping entry as follows: $\hat{L}_i : t_2 \rightarrow \vec{y}$ for all $\vec{y}$.

Function: maximal(OT, U, M)
Input: OT: a set of elements of tree, U: a subset of OT
Output: M: a set of maximal elements
Variables: Chain, IntSec: temporary set
Step 1 $M := U$
Step 2 For all $u \in U$ do the following:

Step 2.1 Find $\bar{u}$ that is a parent node of u and add $\bar{u}$ to the Chain set. Let u be $\bar{u}$ and repeat the previous sentence until u reaches the Root node.

Step 3 For all $c \in Chain$, remove c from M if c matches some element of M.

4.6 Access Decision Function

An access decision function δ receives a provision request and returns a set of provisional actions.

Definition 4.6.1 (Access Decision Function: δ): An access decision function is a mapping δ: $\mathsf{PDS} \times \mathsf{PR} \rightarrow \mathsf{PD}$. δ is defined as $\bigcup_{i,j,k} \mathsf{PV}(\rho(\mathsf{AUTH}, \mathsf{TS_{i,j,k}}, q_3))$ such that: PR: (i_q, u_q, a_q, f_q) and $q_3 : (a_q, f_q, \_)$. $\mathsf{TS_{i,j,k}}$ and PV are explained in Section 4.6.3 and in Section 4.6.6, respectively. The δ function corresponds to the access control decision function $\mathsf{ADF_{PBAC}}$.

A δ is defined as including the following six steps: a membership handling step, a propagation handling step, a hierarchy priority handling step, a rule selection step, a provision retrieval function call step, and a provision verification function call step.

4.6.1 Membership Handling Step

A function δ calls a $\mathsf{CM}(\ i_q\ )$ that returns one or more classes, types, or path names $c_j (1 \leq j \leq max_{i_q})$ to which the object instance i_q belongs. Then δ calls $\mathsf{GM}(\ u_q\ )$ and $\mathsf{RM}(\ u_q\ )$ that return one or more group names $g_k (1 \leq k \leq max^{group}_{u_q})$ to which the user u_q belongs, and return one or more role names $r_l (1 \leq l \leq max^{role}_{u_q})$ the user u_q is activating, respectively.

4.6.2 Propagation Handling Step

We explained two propagation policies in Section 4.3.1, π^{MS} and π^{PT}. Since each policy is divided into an object round policy, group round policy, and a role round policy, the resultant six propagation policies are presented as follows:

- Object-round MS Policy (π^{OMS}): $QO_1 : \bigcup_{j=1}^{max_{i_q}} L^{=}_{c_j}(\mathsf{OS})$
- Object-round PT Policy (π^{OPT}): $QO_i : e_i \in \bigcup_{j=1}^{max_{i_q}} L^{=}_{c_j}(\mathsf{OS})$
- Group-round MS Policy (π^{GMS}): $QG_1 : \bigcup_{k=1}^{max^{group}_{u_q}} L^{=}_{g_k}(\mathsf{GS})$
- Group-round PT Policy (π^{GPT}): $QG_i : e_i \in \bigcup_{k=1}^{max^{group}_{u_q}} L^{=}_{g_k}(\mathsf{GS})$
- Role-round MS Policy (π^{RMS}): $QR_1 : \bigcup_{l=1}^{max^{role}_{u_q}} L^{=}_{r_l}(\mathsf{RS})$
- Role-round PT Policy (π^{RPT}): $QR_i : e_i \in \bigcup_{l=1}^{max^{role}_{u_q}} L^{=}_{r_l}(\mathsf{RS})$

As described in Section 4.4, $L^{=}_{c_j}(\mathsf{OS})$ means a chain that starts from c_j and goes to the root element of OS. Suppose that there is a tree $\{o_1, o_2, \ldots, o_5 | o_1 \prec \{o_2, o_3\}, o_2 \prec \{o_4, o_5\}\}$ and two chains that starts from o_3 and o_4. Suppose c_1 is o_4 and c_2 is o_3. In the case of π^{OMS}, QO_1 is determined as $\{o_4, o_2, o_1, o_3\}$. In the case of π^{OPT}, QO_1, QO_2, QO_3 and QO_4 are $\{o_4\}, \{o_2\}, \{o_1\}$ and $\{o_3\}$, respectively.

4.6.3 Hierarchy Priority Handling Step

We explained three hierarchy-priority policies in Section 4.3.2, π^{OHP}, π^{GHP}, and π^{RHP}. We present each definition below:

- Object takes precedence policy (π^{OHP}): $\mathsf{TS_{i,j,k}}$:$\langle QO_i \times QG_j \times QR_k, \leq^{LO} \rangle$
- Group takes precedence policy (π^{GHP}): $\mathsf{TS_{i,j,k}}$:$\langle QG_j \times QR_k \times QO_i, \leq^{LO} \rangle$
- Role takes precedence policy (π^{RHP}): $\mathsf{TS_{i,j,k}}$:$\langle QR_k \times QG_j \times QO_i, \leq^{LO} \rangle$

4.6.4 Rule Selection Step

A rule selection determines $\mathsf{AUTH_{<i,j,k>}}$, a subset of AUTH, that is defined as follows: $\mathsf{AUTH_{<i,j,k>}} \in \mathsf{AUTH}(o, g, r, \_, \_, \_) \Leftrightarrow \forall o \in QO_i, \forall g \in QG_j, \forall r \in QR_k$.

4.6.5 Provision Retrieval Function Call Step

δ calls a ρ function with $\mathsf{AUTH_{<i,j,k>}}$, $\mathsf{TS_{i,j,k}}$ and q_3. The value of q_3 means that the requested property is the third argument of the non-tree property of $\mathsf{ACPR_{PBAC}}$(the provisional action.)

4.6.6 Provision Verification Function Call Step

δ finally verifies a set of provisional actions returned by the previous step and returns a set of provisional actions.

Definition 4.6.2 (Provision Verification Function: PV) A provision verification function is a mapping PV: $\mathcal{P}(\mathsf{PS}) \rightarrow \mathcal{P}(\mathsf{PS})$

By definition, the ρ function returns a set of provisional actions. The PV verifies a set of provisional actions, modifies and removes some elements so that the set satisfies specified constraints such as an ordered set, and returns a set of provisional actions. Compared with conventional access control models, this function corresponds to a conflict resolution function that determines grant or denial permission according to the conflict resolution policies such as a "closed policy" and a "open policy".

4.7 Provisional Authorization Model: $\mathcal{PAM}$

Definition 4.7.1: (Provisional Authorization Model: $\mathcal{PAM}$)
A provisional authorization model $\mathcal{PAM}$ is a 4-tuple (PDS, δ, PR, $\mathcal{P}$(PS)).

Definition 4.7.2 (Resolution of Provisional Actions): A provisional action resolution using $\mathcal{PAM}$ is to call a δ function against a query $q \in$ PR. $\mathcal{PAM}$ returns a set of provisional actions $ps \in \mathcal{P}$(PS). The semantics of the proposed $\mathcal{PAM}$ model are defined as follows: if the permission f in the provision request q is positive, an access request is allowed provided all provisional actions $ps \in \mathcal{P}$(PS) are executed successfully; if the permission f is negative, an access request is not allowed however all provisional actions ps must still be executed.

4.8 Data-complexity

Algorithm 4.5.1 has data-complexity $\mathcal{O}(k \times l^3)$, where k is the number of nodes in each tree, and l is a number of L relations. Since the size of L is determined by the number of the AUTH relation selected in the rule selection step, it is smaller than the number of the access control policy rules $\mathsf{ACPR}_{\mathsf{PBAC}}$. The complexity order is calculated from the following estimation: Step 3 is $\mathcal{O}(k \times l^2)$, Step 4 is $\mathcal{O}(k \times l^3)$, and the maximal procedure is $\mathcal{O}(k \times l^2)$.

We now evaluate the data complexity of the ρ function that is a primary function of the $\mathcal{PAM}$ model. It takes l operations to determine all elements $u \in U$ such that $\forall s, s \in \lambda_{\mathsf{AUTH}}(u), \forall j, j \neq 3, \sigma_j(s) = \sigma_j(q_3)$. Next we must determine all elements $v \in V$ that is defined as a set of maximal elements using a lexicographic order. It takes $\mathcal{O}(k \times l^3)$ operations according to the previous result. Thus the data complexity of the $\mathcal{PAM}$ model is at most cubic in the size of the relation AUTH under an assumption that k is not large. Since the number of values returned by the ρ function is determined by the number of elements of a set V, a finite number of properties is returned.

5. Application Examples

This section describes typical applications of the proposed access control model including the security policies described in Section 2.

5.1 Application Security Policies

Suppose that there is an object hierarchy and a role hierarchy as illustrated in Figure 2. Each node in the object hierarchy represents a security category of target objects. For example, a node labeled as *Confidential* indicates security policies with regard to data containing confidential information. Thus, when someone tries to update a document labeled as confidential, a set of access control policy rules that are associated with the node *Confidential* determines a necessary set of provisional actions.

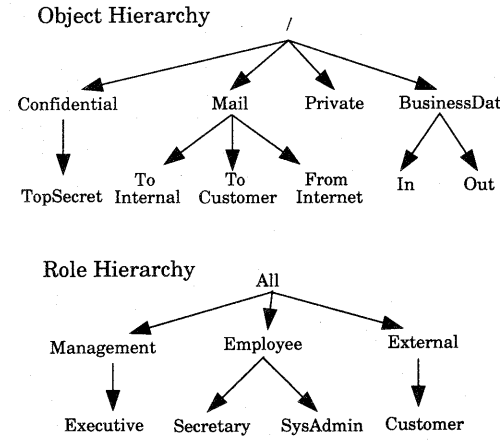

Fig. 2 Object and Role Hierarchy for Security Policy Examples

The organizational security policies mentioned in Section 2 are specified in Table 1 using notations of the $\mathcal{PAM}$ model.

R1 (R2) says that company executives can update (browse) confidential information, provided it is encrypted (decrypted) using a secret key for executives. This policy guarantees confidentiality of the important information even if the information repository were physically stolen and tampered with by a malicious party. R3 says that the system administrator can backup confidential and top secret information, provided it is time-stamped. R4 says that executives can perform any action on top secret information, provided the access is logged. When the object path traversing policy is selected, encrypt and log provisional actions are resolved when the executives update top secret information. R5 says that company employees can send their e-mail to internal destinations in plain-text format. However, the e-mail content must be signed and encrypted when employees send e-mail to their customers. This policy protects e-mail to Internet customer from forgery and wiretapping. R7 says that employees can receive a mail from Internet users, provided it is scanned using an antivirus utility. This policy guarantees that the employees' computer is not affected by any e-mail messages containing known computer viruses. R8 says that each company employee can write his or her own personal information, provided the contents is encrypted using his or her public key. This policy guarantees that nobody can read the private information of others.

Examples of business to business security policies are

Table 1 Example of Organizational Security Policies

ID	Target Object	Role	Action	P	Provision
R1	/Confidential	Exec	write	+	encrypt(exec)
R2	/Confidential	Exec	read	+	decrypt(exec)
R3	/Confidential	SysAd	backup	+	timestamp
R4	/Conf/TopSec	Exec	*	+	log
R5	/Mail/ToInt	Emp	send	+	
R6	/Mail/ToCstm	Emp	send	+	sign, encrypt
R7	/Mail/FromInt	Emp	recv	+	antivirus
R8	/Private	Emp	write	+	encrypt(init)
R9	/Private	Emp	read	+	decrypt(init)

specified in Table 2. R11 says that customers can submit new business data (e.g. a purchase order) provided the digital signature attached to the message body is verified correctly and the appropriate access fee is charged. R12 says that management people can modify business data, provided the content is signed using their private key and encrypted. R13 says that customers can read posted business data, provided the session is encrypted and a notification of the access is sent to the owner of the business data. R10 says that each transaction must be time-stamped and logged. If we use the path traversing policy on the object hierarchy (π^{OPT}), time-stamping and logging operations are performed whenever business transactions occur.

Table 2 Example of Business to Business Security Policies

ID	Object	Role	Act	P	Provision
R10	/BizData	*	*	+	timestamp, log
R11	/BizData/In	Cstm	write	+	verify,charge
R12	/BizData/Out	Mgmt	write	+	sign, encrypt
R13	/BizData/Out	Cstm	read	+	ssl,notify(owner)

5.2 Site Security Policies

These days it is very common to use a firewall between the Internet and an intranet, which protects target security domains from unauthorized network accesses. In the firewall system, there is a set of access control policy rules, such as access or denial of access for certain IP addresses. Using the proposed access control model, examples of site security policies are described in Table 3. R14 (R15) says that access to the ftp (http) server of 123.10.12.2 (123.10.12.3) is permitted. R16 says that other access requests are rejected but the access request must be logged.

Table 3 Example of Firewall Security Policies

ID	Object	Group	Action	P	Provision
R14	ftp/123.10.12.2	*	connect	+	
R15	http/123.10.12.3	*	connect	+	
R16	*	*	connect	−	log

6. Conclusion

There have been few proposals published with regard to the combination of access control policy rules with the necessary cryptographic operations. We propose the idea of "provisional actions" that tells the user that the requests will be authorized provided he (and/or the system) takes certain security-related actions. The major advantage of our approach is that it can be used to specify any operations that can coexist in any access control policy rules. We built a provision-based access control model which captures the notion of the property propagation through multiple authorization hierarchies. We presented algorithms and analyzed their data complexities. Our proposed model supports various authorization policies such as the most specific property takes precedence policy and the path traversing policy. We have illustrated how provisional actions are effectively specified in several practical application scenarios.

References

[1] IETF. *TLS Protocol Version 1.0, RFC 2246*, 1 1999.

[2] ISO/IEC. *Information technology - Open Systems Interconnection - Security frameworks for open systems: Access Control Framework, International Standard, ISO/IEC 10181-3*, 9 1996.

[3] Sushil Jajodia, Michiharu Kudo, and V. S. Subrahmanian. Provisional authorizations. In *1st ACM Workshop on Security and Privacy for Electronic Commerce*. ACM, 11 2000.

[4] Sushil Jajodia, Pierangela Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *IEEE Symposisum on Security and Privacy*, pages 31–42. IEEE, 8 1997.

[5] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *SIGMOD International Conference on Management of Data*, pages 474–485. ACM, 8 1997.

[6] Michiharu Kudo and Satoshi Hada. Xml document security based on provisional authorization. In *7th ACM Conference on Computer and Communications Security*, pages 87–96. ACM, 11 2000.

[7] Michiharu Kudo and Satoshi Hada. Access control model with provisional actions. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(1):295–302, 1 2001.

[8] T. Y. C. Woo and S. S. Lam. A framework for distributed authorization. In *1st ACM Conference on Computer and Communications Security*, pages 143–158. ACM, 9 1993.

[9] World Wide Web Consortium (W3C). *Extensible Markup Language (XML) 1.0*, 2 1998.

[10] World Wide Web Consortium (W3C). *Universal Description, Discovery, and Integration*, 10 2000.

[11] World Wide Web Consortium (W3C). *XML-Signature Syntax and Processing*, 10 2000.