A Generalized Policy Support System and Its Hierarchy Semantics

Yibing Kong, Janusz R. Getta, Ping Yu, and Jennifer Seberry

School of Information Technology and Computer Science,
University of Wollongong,
Wollongong, NSW, Australia

Abstract. One common characteristic of many *Policy Support Systems* ($\mathcal{PSS}$s) is their dependency on the concept of *hierarchy*. Hierarchy does not need to be limited to a hierarchy of roles (subject centric) as in traditional Role-Based Access Control (RBAC). Instead, it can be applied to other aspects of $\mathcal{PSS}$ such as object, environment, purpose and so on. In this paper, we propose a new generalized model for $\mathcal{PSS}$. The model unifies Generalized Role-Based Access Control (GRBAC) and Enterprise Privacy Practices (E-P3P) policy support systems and generalizes their hierarchy semantics.

Keywords: Access Control, Hierarchy, Hierarchy Semantics

1 Introduction

Organizations must enforce data protection policies to secure data collected and generated during their daily operational procedures. At a conceptual level, data protection polices are expressed as sequences of statements written in a natural language. In the past, the enforcement of data protection policies was based on manual work or inflexible mechanisms such as Discretionary Access Control (DAC), Mandatory Access Control (MAC) models. The emerge of Role-Based Access Control (RBAC) model improved efficiency of data protection. Unfortunately, RBAC model is still not expressive enough to efficiently and effectively enforce more sophisticated data protection policies in an organization. Recently more powerful systems have appeared; Generalized Role-Based Access Control (GRBAC) [1] and Enterprise Privacy Practices (E-P3P) [2] are representatives of such systems.

GRBAC, proposed by Moyer and Ahamad in [1], is an extension of traditional RBAC. It generalizes the classical concept of *role* through the new concepts such as *subject role*, *object role* and *environment role*. These concepts are used to structure the subjects (users), objects (data) and environments (conditions). With these new types of roles, GRBAC is capable of creating rich access control policies. GRBAC provides an algorithm to enforce the access control policies defined in the model. E-P3P, developed by Ashley *et al.* in [2], has a well-defined privacy architecture and semantics. It enables an organization to express its privacy policies in E-P3P format and to enforce the policies automatically. We shall call these two systems as *Policy Support Systems* ($\mathcal{PSS}$s) because they are designed for expressing and enforcing data protection policies.

A *hierarchy* is a partial order on a set of elements that defines a seniority relationship between elements [3]. Hierarchy is not a new concept, it has been extensively studied in the past. *Role hierarchy* [4–7] in role-based access control and *hierarchy* in Flexible Authorization Framework (FAF) [8] are examples of such studies. Hierarchy semantics is an inseparable part of hierarchy that defines rules of authorization propagation. Various hierarchy semantics was defined in GRBAC and E-P3P. However, the hierarchy semantics defined is either incomplete or incorrect. In this paper, we propose a generalized $\mathcal{PSS}$ model that covers GRBAC and E-P3P. Based on this model, the hierarchy semantics of GRBAC and E-P3P is analyzed. We propose new hierarchy semantics to solve the problems encountered in GRBAC and E-P3P.

The organization of the rest of the paper is as follows. Section 2 introduces the concept of hierarchy. A generalized $\mathcal{PSS}$ is proposed in section 3 and hierarchy semantics of GRBAC and E-P3P is analyzed in section 4. Section 5 presents new hierarchy semantics. Finally, in section 6, we conclude the paper and outline the plans of future research.

2 Definition of Hierarchy

A mathematical structure called *hierarchy* was defined by Jajodia *et al.* in [8] as a triple $(X, Y, \leq)$, where X is the set of *primitive entities*, e.g. a user, an object; Y is the set of *categories*, e.g. a group, an object type; $\leq$ is a partial order on $(X \cup Y)$ such that each $x \in X$ is a *minimal element* of $(X \cup Y)$; an element $x \in X$ is said to be minimal iff there are no elements below it in the hierarchy, that is iff $\forall y \in (X \cup Y) : y \leq x \Rightarrow y = x$. This definition is rich enough to capture all hierarchy structures presented in [1, 2]. We simplify this definition of hierarchy to a two-entry tuple $(Y, \leq)$, i.e. $H = (Y, \leq)$ where:

- Y is the set of categories, such that a primitive entity is treated as a category of itself, called *primitive category*. A primitive category contains one primitive entity and the name of the category is the same as the name of the primitive entity. For example a primitive entity *James Bond* belongs to a primitive category called `James Bond`.
- $\leq$ is a partial order on Y such that each primitive category in Y is a minimal element of Y.

In addition, we define the following two binary relations over Y. The first binary relation $<$ describes *descendant-ancestor* relationship between the elements in Y. If $y_i, y_j \in Y$ and $y_i < y_j$, y_j is said to be the ancestor of y_i; y_i is said to be the descendant of y_j. The relation $y_i < y_j$ is interpreted as y_i is *in the category of* y_j. For an element y_i, a set of all its ancestors is defined as $Aset_{y_i} = \{y_k : y_k \in Y \text{ and } y_i < y_k\}$; a set of all its descendants is $Dset_{y_i} = \{y_k : y_k \in Y \text{ and } y_k < y_i\}$. The second binary relation $<^C$ describes *child-parent* relationship between elements in Y. If $y_i, y_j \in Y$ and $y_i <^C y_j$, y_j is said to be the parent of y_i; y_i is said to be the child of y_j. For an element y_i, a set of all its parents is defined as $Pset_{y_i} = \{y_k : y_k \in Y \text{ and } y_i <^C y_k\}$; a set of all its children is $Cset_{y_i} = \{y_k : y_k \in Y \text{ and } y_k <^C y_i\}$.

Some of the hierarchies used in practice are listed as follows. *Subject hierarchy* $GH = (G, \leq_G)$ where G is a set of groups (roles), $\leq_G$ defines hierarchy relationships between groups in G. *Object hierarchy* $TH = (T, \leq_T)$ where T is a set of types, $\leq_T$ defines hierarchy relationships between types in T. *Environment hierarchy* $EH = (E, \leq_E)$ where E is a set of environments, $\leq_E$ defines hierarchy relationships between environments in E. *Purpose hierarchy* $PH = (P, \leq_P)$ where P is a set of purposes, $\leq_P$ defines hierarchy relationships between purposes in P.

3 A Generalized Model of a Policy Support System

It is common for a $\mathcal{PSS}$ to define more than one hierarchies. For example, GRBAC [1] defines GH, TH and EH hierarchies and E-P3P [2] defines GH, TH and PH hierarchies. Furthermore these systems define some sets of elements, such as the set of actions, the set of obligations, the set of authorization types etc. In this section, we define a generalized model which unifies GRBAC and E-P3P.

A generalized *Policy Support System* $\mathcal{PSS} = (\mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, where:

- $\mathcal{H}$ denotes a set of n hierarchies $H_1, ..., H_n$, $n \geq 1$.
- $\mathcal{S}$ is a set of m sets $S_1, ..., S_m$, where $m \geq 0$. The sets defined in $\mathcal{S}$ provide additional restrictions on $\mathcal{PSS}$; for example the sets of obligations and conditions in E-P3P system are instances of such sets. $\mathcal{S}$ is optional and its existence depends on the designer of a $\mathcal{PSS}$, e.g. in GRBAC $\mathcal{S}$ is absent.
- $\mathcal{A}$ is a set of actions to be performed on data.
- $\mathcal{R}$ is a set of authorization types (rulings). $R = \{+, -, \oplus, \ominus, \odot, \otimes\}$, where $+$ means *positive authorization*; $-$ means *negative authorization*; $\oplus$ means *implicit positive authorization*; $\ominus$ means *implicit negative authorization*; $\odot$ means *authorization pending* and $\otimes$ means *no authorization*. $+$ and $-$ are used for explicit authorization assignment through policy rules; $\oplus$ and $\ominus$ are used for authorization propagation; $\odot$ is used for conflicts resolution; $\otimes$ is used when none of the above five rulings is applicable.
- $\mathcal{P}$ is a set of precedences over policy rules. $\mathcal{P} = \mathbb{Z}$, i.e. $\mathcal{P}$ is the set of integers that determines precedence orders over a set of policy rules; the greater number denotes the higher precedence. $\mathcal{P}$ is also optional; the absence of $\mathcal{P}$ means that all policy rules have the same precedence.

The elements of the above five parts are used as basic units to form policy rules. The collection of all the policy rules in a $\mathcal{PSS}$ is a policy rule set, denoted as Γ. A *policy rule* $\gamma \in \Gamma$ is a tuple $(\gamma_{H_1}, ..., \gamma_{H_n}, \gamma_{S_1}, ..., \gamma_{S_m}, \gamma_{\mathcal{A}}, \gamma_{\mathcal{R}}, \gamma_{\mathcal{P}})$ where

- $\gamma_{H_i} \in H_i$ or $\gamma_{H_i} = null$ (i.e. nothing is specified for γ_{H_i}), where $n \geq i \geq 1$.
- $\gamma_{S_i} \in S_i$ or $\gamma_{S_i} = null$, where $m \geq i \geq 0$.
- $\gamma_{\mathcal{A}} \in \mathcal{A}$ is the action entry that specifies the action to be performed.
- $\gamma_{\mathcal{R}} \in \{+, -\}$ is the ruling entry that specifies either positive or negative authorization.
- $\gamma_{\mathcal{P}} \in \mathcal{P}$ is the precedence of γ.

It is possible to show that the model defined above "includes" GRBAC [1] and E-P3P [2]. A GRBAC system is a triple $(\mathcal{H}, \mathcal{A}, \mathcal{R})$, where $\mathcal{H} = \{GH, TH, EH\}$ (GH is subject hierarchy, TH is object hierarchy and EH is environment hierarchy). A GRBAC policy rule [1] is a tuple $(S, O, E, op,$ *permission bit*$)$, where $S \in GH, O \in TH, E \in EH, op \in \mathcal{A}$ and *permission bit* $\in \{+, -\}$. There is no precedence over GRBAC policy rules, hence the set $\mathcal{P}$ is absent. An E-P3P system is a tuple $(\mathcal{H}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, where $\mathcal{H} = \{GH, TH, PH\}, \mathcal{S} = \{O, C\}$ (GH is subject hierarchy, TH is object hierarchy, PH is purpose hierarchy, O is the set of obligations and C is the set of conditions). An E-P3P policy rule [2] is a tuple $(i, t, p, u, r, a, \bar{o}, \bar{c})$, inside which $i \in \mathcal{P}, t \in TH, p \in PH, u \in GH, r \in \{+, -\}, a \in \mathcal{A}, \bar{o} \in O$ and $\bar{c} \in C$.

According to the definition of a policy rule, an *access request* α can be expressed as $\alpha = (\alpha_{H_1}, ..., \alpha_{H_n}, \alpha_{S_1}, ..., \alpha_{S_m}, \alpha_{\mathcal{A}})$ where $\alpha_{H_i} \in H_i$ or $\alpha_{H_i} = null$, $n \geq i \geq 1$; $\alpha_{S_i} \in S_i$ or $\alpha_{S_i} = null$, $m \geq i \geq 0$; $\alpha_{\mathcal{A}} \in \mathcal{A}$. A set of policy rules Γ_α is used to validate α, where $\Gamma_\alpha \subseteq \Gamma$. All policy rules in Γ_α are called *matching rules* of α. Matching rules must satisfy the following properties:

- $\forall \gamma \in \Gamma_\alpha, \alpha_{H_i} \leq \gamma_{H_i}$, where $n \geq i \geq 1$.
- $\forall \gamma \in \Gamma_\alpha, \gamma_{S_i} = \alpha_{S_i}$, where $m \geq i \geq 0$.
- $\forall \gamma \in \Gamma_\alpha, \gamma_{\mathcal{A}} = \alpha_{\mathcal{A}}$.

The validation of α in Γ_α consists of n sub-validations from α_{H_1} to α_{H_n}. That is, $\forall \alpha_{H_i} \in \alpha$ where $n \geq i \geq 1$, α_{H_i} needs to be validated according to the matching rule set Γ_α whether α_{H_i} is authorized for $\alpha_{\mathcal{A}}$. If and only if all of these hierarchy elements are authorized for $\alpha_{\mathcal{A}}$, α is granted.

$\forall \gamma \in \Gamma_\alpha$, where $\gamma = (\gamma_{H_1}, ..., \gamma_{H_n}, \gamma_{S_1}, ..., \gamma_{S_m}, \gamma_{\mathcal{A}}, \gamma_{\mathcal{R}}, \gamma_{\mathcal{P}})$, we can assign a tuple $(\gamma_{\mathcal{R}}, \gamma_{\mathcal{P}})$ to $\gamma_{H_1}, ..., \gamma_{H_n}$. An *authorization* of an element $y \in H$ is a tuple $(ruling, precedence)$ inside which $ruling \in \mathcal{R}, precedence \in \mathcal{P}$; it is denoted as A_y^i where i is the index of the authorization. Due to the optional property of $\mathcal{P}$, the precedence entry of an authorization is also optional. The ruling entry of A_y^i is denoted as $A_y^i.ruling$ and the precedence entry of A_y^i is denoted as $A_y^i.precedence$. We call an authorization explicitly defined by policy rules *explicit authorization*; obviously for an explicit authorization A, $A.ruling \in \{+, -\}$. Authorization may also be derived by hierarchy semantics, we call a derived authorization *implicit authorization*; for an implicit authorization A, $A.ruling \in \{\oplus, \ominus\}$. If an explicit authorization $A_{y_i}^1$ of $y_i \in H$ propagates to $y_j \in H$, then $A_{y_i}^1$ is converted to an implicit authorization $A_{y_j}^1$ by the following processes: if $A_{y_i}^1.ruling = +$, then $A_{y_j}^1.ruling = \oplus$; if $A_{y_i}^1.ruling = -$, then $A_{y_j}^1.ruling = \ominus$; $A_{y_j}^1.precedence = A_{y_i}^1.precedence$.

Here is an example of assigning explicit authorizations, assume $\Gamma_\alpha = \{\gamma_1, \gamma_2\}$; $\gamma_1 = (..., \gamma_{1H_i}, ..., read, +, 1)$, $\gamma_2 = (..., \gamma_{2H_i}, ..., read, -, 2)$, where $\gamma_{1H_i} = \gamma_{2H_i} = y \in H_i$; then $A_y^1 = (+, 1)$, $A_y^2 = (-, 2)$. Two authorizations A_y^i, A_y^j are *inequable* if either $A_y^i.ruling \neq A_y^j.ruling$ or $A_y^i.precedence \neq A_y^j.precedence$ holds. When an element of a hierarchy has more than one inequable authorizations, conflicts arise. Our system provides methods of conflicts resolution, called *authorization precedence policy*. In our system, conflicts can be solved either *manually* or *automatically*. For a hierarchy $H_i = (Y_i, \leq_i)$, the *System Security Officer* (SSO) defines a *manual resolution set* $MR_i \subseteq Y_i$. If an element $y \in MR_i$ has authorization conflicts, we assign y with the authorization $(\odot,)$. In this case, the decision of the access request α that causes the conflicts will be pending until the conflicts are manually solved by SSO. If an element $y \in Y_i \setminus MR_i$ has authorization conflicts, these conflicts are solved automatically by the following rules.

- $\odot$ authorizations are with the highest precedence; $\otimes$ authorizations are with the lowest precedence. There are no maximum and minimum integers in $\mathbb{Z}$, hence the precedence entries of these two types of authorizations are absent. An authorization with higher precedence overrides authorizations with lower precedences.
- If the precedences are the same, *denies-take-precedence* will apply. The priority order is: $-, \ominus, +, \oplus$. An authorization with higher priority order overrides authorizations with lower priority orders.

Our authorization precedence policy is very flexible. Authorization pending gives SSO opportunities to review authorization conflicts. By doing that, SSO may find bugs of Γ and give user better response. After we perform all authorization derivations and conflicts resolutions on H_i, H_i's *final authorization state* is obtained, where $\forall y \in H_i$, y has a *final authorization* FA_y that is not conflicting. The result of the sub-validation of α_{H_i} is $FA_{\alpha_{H_i}}.ruling$. The decision of α is processed as follows.

- If $\forall FA_{\alpha_{H_i}}.ruling = +$ (or $\oplus$) where $n \geq i \geq 1$, then α is approved.
- If $\exists FA_{\alpha_{H_i}}.ruling = -$ (or $\ominus$ or $\otimes$), then α is denied.
- If α is not denied and $\exists FA_{\alpha_{H_i}}.ruling = \odot$, then α is pending.

4 Hierarchy Semantics of GRBAC and E-P3P

Hierarchy semantics defines rules of authorization propagation. In this section, the hierarchy semantics of GRBAC and E-P3P is depicted.

4.1 Hierarchy Semantics of GRBAC

The hierarchy semantics in GRBAC [1] is defined by *permission inheritance*. There are three types of permission inheritances: *standard*, *strict* and *lenient*. Suppose we have an access request $\alpha = (\alpha_{GH}, \alpha_{TH}, \alpha_{EH}, \alpha_{\mathcal{A}})$, where $\alpha_{GH} = y_4 \in GH$, denoted as $GH.y_4$, $\alpha_{TH} = TH.y_5$ and $\alpha_{EH} = EH.y_2$. Here we utilize the validation of $GH.y_4$ to illustrate the semantics of the three types of permission inheritances.

- *Standard permission inheritance*:
  If $\exists y_i \in Aset_{GH.y_4} \cup \{GH.y_4\}$ such that $FA_{y_i}.ruling = +$ and $\neg\exists y_j \in Aset_{GH.y_4} \cup \{GH.y_4\}$ such that $FA_{y_j}.ruling = -$, then $GH.y_4$ is authorized for $\alpha_{\mathcal{A}}$. Otherwise, it is not authorized for $\alpha_{\mathcal{A}}$.
- *Lenient permission inheritance*:
  If $\exists y_i \in Aset_{GH.y_4} \cup \{GH.y_4\}$ such that $FA_{y_i}.ruling = +$, then $GH.y_4$ is authorized for $\alpha_{\mathcal{A}}$. Otherwise, it is not authorized for $\alpha_{\mathcal{A}}$.
- *Strict permission inheritance*:
  If $\forall y_i \in Aset_{GH.y_4} \cup \{GH.y_4\}$ such that $FA_{y_i}.ruling = +$, then $GH.y_4$ is authorized for $\alpha_{\mathcal{A}}$. Otherwise, it is not authorized for $\alpha_{\mathcal{A}}$.

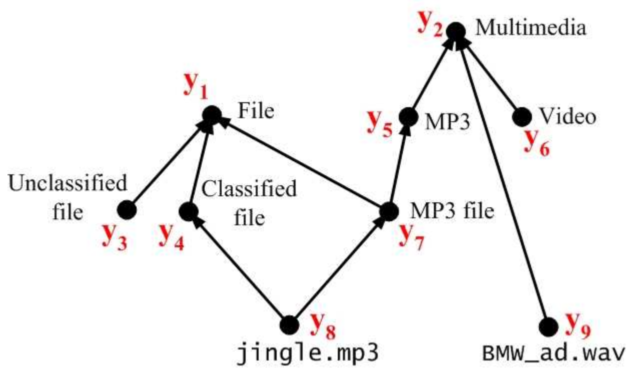

Fig. 1. Example object hierarchy TH

An example below illustrates the hierarchy semantics of GRBAC. Consider the object hierarchy TH shown in figure 1; there are 9 elements in TH, among which `jingle.mp3` (y_8) is a primitive category that is a child of *classified file* (y_4) and *MP3 file* (y_7). Now there is an access request α from user u (we assume u is y_6 in GH) to read `jingle.mp3` ($TH.y_8$) under environment e (we assume e is y_1 in EH), i.e. $\alpha = (GH.y_6, TH.y_8, EH.y_1, read)$. The policy rule set $\Gamma = \{\gamma_1, \gamma_2\}$; $\gamma_1 = (GH.y_6, TH.y_4, EH.y_1, read, +)$, $\gamma_2 = (GH.y_6, TH.y_7, EH.y_1, read, +)$. Obviously, the matching rule set $\Gamma_\alpha = \Gamma$. According to GRBAC hierarchy semantics, we can derive that u can read `jingle.mp3` if standard or lenient permission inheritance is applied; u cannot read it if strict permission inheritance is applied.

The original intention of strict permission inheritance is to restrict accesses to the elements in the category of sensitive/vulnerable categories defined by SSO. GRBAC's strict permission inheritance is trying to fulfill this intention. However this definition is too strict to be practical. In the example above, user u is explicitly authorized to read both *classified file* and *MP3 file* and there is no policy rule disallow these accesses (as shown in the example Γ above). In this case, even if we are very strict, user u should have read access to `jingle.mp3`, which is a child of *classified file* and *MP3 file*. GRBAC will deny this access because GRBAC's strict permission inheritance requires the requested element and all its ancestors to be explicitly authorized for the access; obviously this is too strict. If a system deploys this strict permission inheritance, few access requests can be granted.

4.2 Hierarchy Semantics of E-P3P

In E-P3P, an access request α is processed in the following two steps [2]. The first step creates a set of preliminary authorization rules PA; PA is a rule set defined as the union of Γ and $D\Gamma$, where $D\Gamma$ contains all rules derived from Γ by using the hierarchy semantics defined in this system. The second step processes access request α according to PA.

The hierarchy semantics of E-P3P is defined as follows [2]:

- Down-inheritance: For each rule $(i, t, p, u, r, a, \overline{o}, \overline{c}) \in PA$, for every (t', p', u') such that $t' \leq_T t$, $p' \leq_P p$, and $u' \leq_G u$, a tuple $(i, t', p', u', r, a, \overline{o}, \overline{c})$ is added to PA.
- Up-inheritance of deny (negative authorization): For each rule $(i, t, p, u, -, a, \overline{o}, \overline{c}) \in PA$, for every (t', p', u') such that $t \leq_T t'$, $p \leq_P p'$, and $u \leq_G u'$, a tuple $(i, t', p', u' -, a, \overline{o}, \overline{c})$ is added to PA.

In this system, if contradicting policy rules coexist, *denies-take-precedence* will be applied to remove contradicting policy rules with lower precedences from PA.

We can identify two problems existing in this system. The first problem is that the concept of *permission inheritance* is omitted. As a consequence, the system is not flexible in practice. For example, it provides no mechanism for enforcing *strict permission inheritance*. The second problem is that the definition of *up-inheritance of deny* is reasonless. The first problem is apparent; here we will give an example that reveals the second problem. Following the semantics of *up-inheritance of deny*, some reasonable requests from users are denied. Let us consider the following scenario. There is a primitive category `BMW_ad.wav` ($TH.y_9$ in figure 1) in the category of *multimedia* ($TH.y_2$ in figure 1), besides we assume that $GH.y_6$ is user u and $PH.y_3$ is a purpose. There are two policy rules in Γ, i.e. $\Gamma = \{\gamma_1, \gamma_2\}$; $\gamma_1 = (1, TH.y_7, PH.y_3, GH.y_6, -, read, null, null)$, $\gamma_2 = (1, TH.y_2, PH.y_3, GH.y_6, +, read, null, null)$. Now there is an access request from user u: $\alpha = (TH.y_9, PH.y_3, GH.y_6, read, null, null)$; i.e. user u requests to read `BMW_ad.wav` for the purpose of $PH.y_3$ with no specified condition and obligation. Because $TH.y_7 \leq_T TH.y_2$ (see figure 1), $PH.y_3 \leq_P PH.y_3$ and $GH.y_6 \leq_G GH.y_6$, following up-inheritance of deny, there will be a policy rule γ_3 derived from γ_1: $\gamma_3 = (1, TH.y_2, PH.y_3, GH.y_6, -, read, null, null)$, that is user u is not allowed to read *multimedia* for the purpose of $PH.y_3$. Then $PA = \Gamma \cup \{\gamma_3\}$, i.e. $PA = \{\gamma_1, \gamma_2, \gamma_3\}$ (here we skip other derived rules). The two policy rules γ_2 and γ_3 are contradicting policy rules. Because of denies-take-precedence, the rule γ_2 will be removed from PA, now $PA = \{\gamma_1, \gamma_3\}$. The system will validate α according to $PA = \{\gamma_1, \gamma_3\}$, hence according to γ_3, user u's request α is denied.

5 Solution

This section presents the hierarchy semantics defined in our generalized $\mathcal{PSS}$. The semantics described below eliminates the problems mentioned in section 4 and extends the hierarchy semantics of GRBAC and E-P3P.

5.1 Hierarchy Semantics

Our interpretation of the concept of hierarchy is such that the relationship between a descendant element and its ancestor element is *in the category of*. The common rationale is that when an authorization is applied on an ancestor element (superior category), this authorization may also be applied to its descendant elements (inferior categories) implicitly. As a consequence, authorizations propagate downwards ($\odot$ and $\otimes$ authorizations do not propagate; the term *authorization* in this sub-section denotes authorizations other than $\odot$ and $\otimes$ authorizations). We only consider authorization propagations between parents and children here; any complex authorization propagation is an aggregation of such simple propagations. In a hierarchy, two different types of elements must be clearly distinguished. *Pure element* is an element that has only one parent; *hybrid element* is an element that has more than one parents. Based on these two types of elements, two different situations of down-propagation of authorizations can be identified.

- If a child is a pure element, all authorizations of its parent propagate down.
- If a child is a hybrid element, the hierarchy semantics is complicated. There are many options that represent different strictness of authorization propagation. These options are listed as follows.
  (a) *Strict down-propagation*: SSO defines a combination of elements called *Strict Combination* (SC). For a child y, if $\exists y_j \in Pset_y$ such that $y_j \in SC$, then the authorization propagation from y's parents to y will follow the semantics of strict down-propagation. We define a set $SC_y = \{y_i : y_i \in Pset_y$ and $y_i \in SC\}$. The semantics of strict down-propagation is as follows.
     i. All (implicit) negative authorizations of elements in $Pset_y \backslash SC_y$ propagate down to y; all other authorizations of elements in $Pset_y$ propagate down to y iff $\forall y_i \in SC_y$ such that $FA_{y_i}.ruling = +$ (or $\oplus$).
  (b) *Lenient down-propagation*: SSO defines a combination of elements called *Lenient Combination* (LC). For a child y, if $\exists y_j \in Pset_y$ such that $y_j \in LC$ and $\neg\exists y_k \in Pset_y$ such that $y_k \in SC$ (for security concern, strict down-propagation overrides lenient down-propagation), then the authorization propagation from y's parents to y will follow the semantics of lenient down-propagation. We define a set $LC_y = \{y_i : y_i \in Pset_y$ and $y_i \in LC\}$. The semantics of lenient down-propagation is as follows.
     i. If $\neg\exists y_i \in LC_y$ such that $FA_{y_i}.ruling = +$ (or $\oplus$), all authorizations of elements in $Pset_y$ propagate down to y.
     ii. If $\exists y_i \in LC_y$ such that $FA_{y_i}.ruling = +$ (or $\oplus$), all authorizations of elements in $\{y_k : y_k \in Pset_y$ and $FA_{y_k}.ruling = +$ (or $\oplus$)$\}$ propagate down to y.
  (c) *Standard down-propagation*: For a child y, if $\neg\exists y_j \in Pset_y$ such that $y_j \in LC$ and $\neg\exists y_k \in Pset_v$ such that $y_k \in SC$, then the authorization propagation from y's parents to y will follow the semantics of standard down-propagation. The semantics of standard down-propagation is as follows.
     i. All authorizations of y's parents propagate down to y.

The semantics described above generalizes the hierarchy semantics in GRBAC and E-P3P.The strict permission inheritance defined in GRBAC (section 4.1) is a special case of our definition of strict down-propagation where $\forall y_i \in Pset_y, y_i \in SC$. The lenient permission inheritance defined in GRBAC (section 4.1) is a special case of our definition of lenient down-propagation where $\forall y_i \in Pset_y, y_i \in LC$. The proposed hierarchy semantics also eliminates the questionable semantics in GRBAC and E-P3P. If our strict down-propagation is applied in the examples shown in section 4.1 and section 4.2, the reasonable access requests will be approved. The semantics of up-inheritance of deny (section 4.2) is incorrect; hence in our hierarchy semantics it is not included.

5.2 Scenarios of the Use of Hierarchy Semantics

This section shows some examples of using the hierarchy semantics defined in this paper. In these examples, we assume authorization conflicts are resolved automatically. Due to limited space, the examples of down-propagation to a pure element and standard down-propagation are not included in this paper.

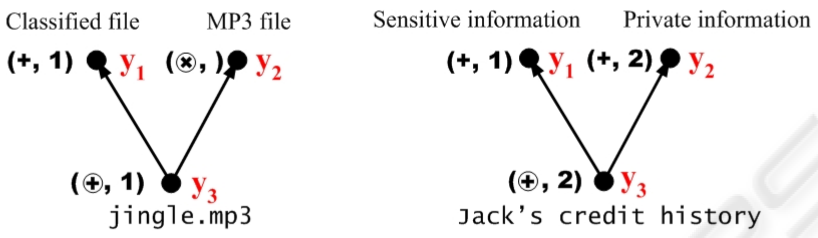

Fig. 2. Example strict down-propagation of object hierarchy

Figure 2 shows two examples of strict down-propagation. In the first example, a primitive category `jingle.mp3` (y_3) enters two categories: *classified file* (y_1) and *MP3 file* (y_2). In this case, SSO wants to be strict to accesses to elements entering category y_1. SSO defines $SC = \{y_1\}$. Because $FA_{y_1}.ruling = +$, FA_{y_1} propagates down to y_3: $A^1_{y_3} = (\oplus, 1)$. $A^1_{y_3}$ is the only authorization that y_3 has, hence $FA_{y_3} = A^1_{y_3}$. In the second example, a primitive category `Jack's credit history` (y_3) enters two categories: *sensitive information* (y_1) and *private information* (y_2). SSO wants to be strict to accesses to elements entering y_1 or y_2. SSO defines $SC = \{y_1, y_2\}$. Because $FA_{y_1}.ruling = +$ and $FA_{y_2}.ruling = +$, FA_{y_1} and FA_{y_2} propagate down to y_3. After conflict resolution, $FA_{y_3} = (\oplus, 2)$.

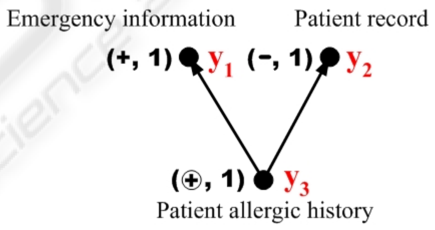

Fig. 3. Example lenient down-propagation of object hierarchy

An example of lenient down-propagation is shown in figure 3. SSO wants to be lenient to those who are allowed to access *emergency information* (y_1), because *emergency information* is often related to vital event. SSO defines $LC = \{y_1\}$ and we assume $SC = \emptyset$. Then even if the other parent y_2 of *patient allergic history* (y_3) is denied access, y_3 is still accessible to those who are allowed to access y_1.

6 Conclusion and Future Work

$\mathcal{PSS}$s are capable of expressing and enforcing rich data protection policies. GRBAC and E-P3P are representatives of such systems. In GRBAC and E-P3P, hierarchy is an important and widely used concept. Being an inseparable part of hierarchy, hierarchy semantics defines rules of authorization propagation. In this paper, we have proposed a generalized $\mathcal{PSS}$ that covers GRBAC and E-P3P. Based on this generalized $\mathcal{PSS}$, we analyze the hierarchy semantics used in GRBAC and E-P3P. We point out errors and limitations of GRBAC and E-P3P hierarchy semantics. Finally, we present new hierarchy semantics to address the problems discovered.

In the future, the following research work interests us:

- finding more useful hierarchy semantics.
- investigating efficient access request processing mechanisms.
- reviewing other related work such as access control for XML document etc.

References

1. Moyer, M.J., Ahamad, M.: Generalized role-based access control. In: Proceedings of 21st International Conference on Distributed Computing Systems. (2001) 391–398
2. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In: Proceeding of the ACM workshop on Privacy in the Electronic Society, ACM Press (2002) 103–109
3. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security (TISSEC) **4** (2001) 224–274
4. Moffett, J.D.: Control principles and role hierarchies. In: Proceedings of the third ACM workshop on Role-based access control, ACM Press (1998) 63–69
5. Sandhu, R.: Role activation hierarchies. In: Proceedings of the third ACM workshop on Role-based access control, ACM Press (1998) 33–40
6. Joshi, J.B.D., Bertino, E., Ghafoor, A.: Hybrid role hierarchy for generalized temporal role based access control model. In: Proceedings of 26th Annual International Computer Software and Applications Conference. (2002) 951–956
7. Moffett, J.D., Lupu, E.C.: The uses of role hierarchies in access control. In: Proceedings of the fourth ACM workshop on Role-based access control, ACM Press (1999) 153–160
8. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible support for multiple access control policies. ACM Transactions on Database Systems (TODS) **26** (2001) 214–260