一个基于逻辑的存取控制模型*)

阮宏一[1] 彭智勇[2] 雷建军[1]
(湖北教育学院计算机科学系 武汉 430205)[1]
(武汉大学软件工程国家重点实验室 武汉大学计算机学院 武汉 430072)[2]

摘 要 在任何一个安全系统中,存取控制都是一个极为重要的问题。本文提出一个基于逻辑程序设计的方法来管理非集中式的授权及其代理。在这个系统中,允许用户代理管理权限、授权或禁止其他用户使用某些存取权限。给出一组独立于论域的规则来实现代理正确性、解决冲突和沿着主体、客体及存取权限层次结构的授权传递,其基本思想是将这些一般规则与用户定义的一组与论域相关的特殊规则结合起来,以推导出系统中成立的所有授权。此外,还给出一些语义性质。

关键词 逻辑规则,控制模型,授权,代理

A Logic Based Access Control Model

RUAN Hong-Yi[1] PENG Zhi-Yong[2] LEI Jian-Jun[1]
(Department of Computer Science, Hubei Institute of Education, Wuhan 430205)[1]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072)[2]

Abstract Access control is needed in any system where the resources are shared by different users. In this paper, we develop a logic based approach for decentralized authorization management in which users are delegated, granted or forbidden some access rights. A set of general logic rules are provided to handle the delegation correctness, conflict resolution and authorization propagation along the hierarchies of subjects, objects and access rights. By combining these general rules with application-specific rules defined by users, we can derive the authorizations holding in the system. In addition, some semantic properties are further investigated.

Keywords Logic rules, Control model, Authorization, Delegation

基于逻辑的存取控制方法一直受研究者重视,其目的是形式化授权的描述和求值。这种方法的长处是将授权政策和实现机制分开,给予政策以精确的语义,且提供一个一致的体系结构以支持多种政策。

Abadi等提出了一种基于时态逻辑的方法用于分布式系统的存取控制[1],他们的工作集中于怎样相信一个主机(主体)提出了一个请求,无论是代表他自己或别人。在他们的模型中,代理主要是关于存取权限本身。Jason Crampton等的工作也是基于时态逻辑[3],主要是关于现实世界存取控制机制的表达和推理的研究。Woo和Lan提出了一个分布式系统授权的表达语言[7],考虑了存在于授权中的结构特性,并且给出了基于扩展逻辑程序的形式语义求值。Jajodia等提出了一种逻辑语言并描述了怎样用这个语言来描述授权、解决冲突、存取控制和完整性约束检查等[5]。Bertino等提出了一个逻辑体系结构,在此结构中,考虑了授权中的层次结构的主体、客体和存取权限,支持传统的否定和基于失败的否定,并给出了一个冲突解决的方法。

以上这些工作的一个不足是,不支持管理权限的代理。事实上,离散存取控制中的一个重要问题是关于授权的管理政策,也就是授予和取消授权的功能管理。两个主要的管理政策是集中式和非集中式。采用集中式管理,仅有一个中央授权机制可以授予或取消存取权限,这是一个较普遍的模型,也较易实现。但是它的缺陷是非常不灵活,因为通常当客体数量较大时,一个人很难知道对每一个客体适合的存取控制。另一方面,采用非集中式管理,多个主体可以有权限去授予或取消存取权限,而且管理权限还可以在主体间传递代理。此方法非常灵活,且能适合各个主体的特别要求。大多数商用数据库管理系统采用非集中式授权。然而,既然多个主体可以授予或取消存取权限,授权跟踪就会变得较困难,而且循环授权和连锁授权取消的问题也会发生。此外,当肯定和否定授权同时存在时,冲突问题变得更加重要,因为多个授权管理者大大增加了冲突的可能性,且循环授权可能导致不理想的结果。

本文给出了一个体系结构,它支持授权代理、否定授权和授权继承。还提出了一个基于代理关系的冲突解决方法,此方法给予代理路径上的祖先以较高的优先权,以获取有控制的代理。此体系是基于扩展逻辑程序设计[4]的,它支持传统否定和基于失败的否定。扩展逻辑程序设计的表达能力和非单调推理给用户提供了一个有力的方法来表达复杂的安全政策。在我们的模型中,给出了一组独立于应用的逻辑规则,以

*)本文得到国家863数据库重大专项课题《基于"关系+对象+代理"模型的对象管理技术研究》(2002AA4Z3450)资助。阮宏一 副教授,主要研究方向:数据库技术与应用、数据模型、计算机网络与应用;彭智勇 博士,教授,博士生导师,研究方向:数据模型、高级数据库管理系统多媒体数据库、安全数据库、地理信息系统、网上信息集成、工作流管理、面向对象程序设计语言等;雷建军 副教授,主要研究方向:计算机网络与应用、网络安全、数据库技术与应用。

获取代理正确性，冲突解决和沿着主体、客体和存取权限的层次结构的授权传递。其基本思想是将这些一般规则与用户定义的与应用相关规则相结合，以推导出系统中成立的授权。此外，我们还探讨了一些语义性质。

1 语法

我们的语言是一个多类一阶语言，有 4 个不相交类：S、O、A 和 T，分别代表主体（用户，程序）、客体（文件，数据库）、存取权限（读，写）和授权类型。变量用小写字母开始的字符串表示，常量用大写字母开始的字符串表示。在 S、O、A 上分别定义了 3 个偏序 $<_S$，$<_O$ 和 $<_A$，用以表达主体、客体和存取权限上的继承层次结构。用 $\# \in S$ 来表达安全管理者，且它与 S 中的任何主体相对于 $<_S$ 关系不可比较。有 3 个授权类型，即 $T=\{-,+,*\}$。其中，$-$ 表示否定，$+$ 表示肯定，而 $*$ 表示可代理的。一个否定的授权表示该主体禁止使用该权限；一个肯定的授权表示该主体被授予该权限；而一个可代理的授权表示该主体不仅被授予该权限，而且还被授予管理此权限的功能，即它可以进一步对其他主体对此权限授权（$+$，$-$ 或 $*$）。

谓词集 P 由一组用户定义的一般谓词和一个内置谓词 grant 组成，grant 表示可代理的授权，它是一个 5-项谓词：$S\times O\times T\times A\times S$。直观地说，grant$(s,o,t,a,g)$ 表示 g 授予 s 在 o 上类型为 t 的权限 a。在这里，g 是授权者，s 是主体，o 是客体，t 为类型，a 为存取权限。

一个项是一个常量或变量，注意我们在语言中禁止使用函数符号。一个原子是一个形式为 $p(t_1,\cdots,t_n)$ 的构造，这里 p 是 P 中一个含 n 项的谓词，$t_1,\cdots,t_n$ 为项。一个文字（literal）是一个原子 p 或原子的否定 $\neg p$，这里否定符号 $\neg$ 表示传统否定。两个文字互补，如果它们具有形式 p 和 $\neg p$，这里 p 是原子。一个规则 r 具有以下形式：

$$b_0 \leftarrow b_1,\cdots,b_k,\ not\ b_{k+1},\cdots,not\ b_m,\ m\geq 0$$

这里 $b_0,b_1,\cdots,b_m$ 是文字，not 是基于失败的否定符号。一个可代理的授权程序由一个有限的规则集组成，一个项、原子、文字、规则或程序，如果它不含变量则是基本的（ground）。

2 授权求值的政策

为了开发一个可代理的授权程序的形式语义，考虑 3 个影响求值过程的方面：代理正确性，沿着主体、客体及存取权限层次结构的授权传递和冲突解决。

2.1 关于代理正确性

定义 1（代理正确性） 一个授权集合是代理正确的，如果它满足以下两个条件：

(a) 主体 s 可授予另一个主体在 o 上的某个权限 a，当且仅当 s 是安全管理者 $\#$ 或 s 已被授予 o 上的可代理类型 $*$ 的权限 a；

(b) 若主体 s 直接或间接地从另一主体 s' 接受到 o 上的可代理的权限 a，则 s 不能再对 s' 在 o 上对 a 进行授权。

直观地说，条件(a)来自于系统假设和可代理类型授权的性质，条件(b)表示一给定的客体和存取权限。一个主体不能对其授权者或其授权者的授权者等等授权，其目的是避免循环授权。这个限制使授权代理更加合理，且可避免当授权可被代理时存在于大部分冲突解决方法中的问题。为解释这个问题，我们可看一个简单例子：假设一个公司经理创建了一个有关公司开发项目的文件 F，然后授予各项目经理代理权，这样项目经理就可对参加项目的成员进行对 F 的存取权限授权。但是一个不希望的结果是，在获得代理权后这些项目经理可能会拒绝公司经理存取 F，这只需向公司经理发一个否定授权即可。所以，我们要求从可代理程序导出的授权集是代理正确的。

2.2 关于授权传递

基于规则的授权描述允许从授权集合中导出隐含授权，从而可以极大地减少需要明确描述的授权集合。在我们的模型中，通过支持授权继承来支持隐含授权。根据面向对象方法的特性，对一个对象（object）描述的规则自动被其子对象所继承。特别地，我们的模型支持沿着主体、客体和存取权限层次结构的三维继承。比如说，授权给一个组写一个目录 D，则通过继承，可导出以下隐含的授权：所有此组的成员可写目录 D 这个组可以写所有 D 的子目录及文件；这个组可以读 D，从而也就可以读 D 下的所有子目录及文件；所有这个组的成员可以读 D 及以下所有子目录及文件等等。另外，用户也可以通过规则定义来导出隐含授权。

2.3 关于冲突解决

既然允许肯定和否定授权，冲突的授权就可能发生，并且，既然支持隐含授权，隐含冲突也可能发生，使得冲突问题更加复杂。我们解决冲突的方法概括如下：

策略 1 根据代理关系解决冲突。概括代理关系，如果主体 s 直接或间接将客体 o 上关于权限 a 的授权代理给 s'，则当关于 o 和 a 的冲突发生时，s 发出的授权（s 是授权者）将压倒 s' 发出的授权。对一个可代理的授权程序，不同的主体和客体上的代理关系将会从系统中自动导出。

策略 2 根据两个冲突授权的类型来解决。如果两个冲突授权的授权者相同或不具备代理关系，则考虑两个冲突授权的类型。我们用基于否定优先的悲观方法来解决，给－以最高的优先权，然后是 $+$，最后是 $*$。这种方法可以保证最大限度的安全性。

策略 3 连锁覆盖。当一个可代理授权被覆盖时，其被授权者所发出的所有授权则都被覆盖。换句话说，我们支持连锁覆盖。

2.4 关于稳定模型语义

扩展逻辑程序设计有两个主要的语义：良好基础（well-founded）的语义和稳定模型（stable-model）语义[4]。我们选择稳定模型语义，因为它提供更灵活的方法来处理矛盾和不完全信息，从而更适合我们处理冲突信息。

3 授权语义求值

这一节给出一组与应用无关的规则来获取以下性质：代理正确性，解决冲突和沿着主体、客体和存取权限的层次结构的授权传递。基本想法是，用这些规则和用户定义的一组与应用相关的规则结合起来，导出系统中所有成立的授权。

3.1 关于代理正确性的规则

我们要求由可代理程序推导出的授权集是代理正确的。为了保证这一性质，引入几个新的辅助谓词，把它们当作系统保留字。delegate 具有型式：$S\times S\times O\times A$，其变量从左到右分别代表授权者、主体、客体和存取权限。直观地说，delegate(g,s,o,a) 表示主体 g 直接或间接授予 s 客体 o 上的权限 a，且其类型为 $*$。除了 grant1 已经通过了代理正确性检查，grant1 具有与 grant 相同的型和语义。以下的规则用来获取代理正确性性质。

(D1) grant1(s, o, t, a, #) ← grant(s, o, t, a, #)

(D2) grant1(s, o, t, a, g) ← grant(s, o, t, a, g), grant1(g, o, *, a, g'), $g \neq s$, not delegate(s, g, o, a)

(D3) delegate(g, s, o, a) ← grant1(s, o, *, a, g)

(D4) delegate(s, s_1, o, a) ← delegate(s, s_2, o, a), delegate(s_2, s_1, o, a)

规则(D1)和(D2)用来定义 grant1,它代表满足代理正确性的那些授权。规则(D1)表示任何从管理者发出的授权都被接受。规则(D2)表示对任何 grant(s, o, t, a, g),如果存在某 g' 使 grant1(g,o,*,a,g')为真(表示 g 有权授权),且 $s \neq g$(g 不能给自己授权),且 delegate(s,g,o,a)不知为真(s 没有将 o 上的 a 的代理传给 g,以避免循环授权),则 grant1(s,o,t,a,g)为真。规则(D3)和(D4)推出代理关系。

3.2 关于授权传递的规则

下面考虑沿着主体、客体和存取权限的层次结构的授权传递。这个层次结构是通过对应的偏序来体现的,以下规则用来实现传递:

(H1) grant(s, o, t, a, g) ← grant(s', o, t, a, g), $s' <_S s$, $s \neq g$

(H2) grant(s, o, t, a, g) ← grant(s, o', t, a, g), $o' <_O o$, $s \neq g$

(H3) grant(s, o, t, a, g) ← grant(s, o, t, a', g), $a' <_A a$, $t \neq -$, $s \neq g$

(H4) grant(s, o, t, a, g) ← grant(s, o, t, a', g), $a <_A a'$, $t = -$, $s \neq g$

请注意,不像其他传递是沿着层次结构向下传递。当授权类型为-(否定),传递是沿着存取权限的层次结构向上。比如,当某人被禁止读时,他就隐含地被禁止写。

3.3 关于冲突解决的规则

将用一组与应用无关的规则来实现冲突解决。首先,再介绍 4 个系统保留谓词:overriden1, overriden2, grant2 和 hold,它们具有同样的型 $S \times O \times T \times A \times G$。overriden1($s$, o, t, a, g) 表示根据第 3.3 节策略 1,授权 grant1(s, o, t, a, g) 被其他授权所覆盖;grant2(s, o, t, a, g) 表示授权 grant1(s, o, t, a, g)按照策略 1 和 3 没有被覆盖;overriden2(s, o, t, a, g) 表示 grant2(s, o, t, a, g)根据策略 2 被其他授权所覆盖;hold(s, o, t, a, g) 表示在系统中成立的授权(没有被任何其他授权所覆盖)。假设 $-<+<*$,则规则如下:

(C1) overriden1 (s, o, t, a, g) ← grant1(s, o, t, a, g), grant1 (s, o, t', a, g'), delegate (g', g, o, a), $t \neq t'$

(C2) grant2 (s, o, t, a, #) ← grant1(s, o, t, a, #), *not* overriden1(s, o, t, a, #)

(C3) grant2 (s, o, t, a, g) ← grant1(s, o, t, a, g), *not* overriden1(s, o, t, a, g),grant1(g, o, *, a, g')

(C4) overriden2 (s, o, t, a, g) ←grant2(s, o, t, a, g), grant2(s, o, t', a, g'), $t' < t$

(C5) hold (s, o, t, a, #) ← grant2(s, o, t, a, #), *not* overriden2(s, o, t, a, g)

(C6) hold (s, o, t, a, g) ← grant2(s, o, t, a, g), *not* overriden2(s, o, t, a, g), hold(g, o, *, a, g')

规则(C1)对应于冲突解决策略 1,(C1)表示对某给定的授权 grant1(s, o, t, a, g),若存在另一授权 grant1(s, o, t', a, g') 且 delegate(g', g, o, a)为真,则 overriden1(s, o, t, a, g)为真,这里 $t \neq t'$(表示冲突)。规则(C2)及(C3)用来推导根据策略 1 没有被覆盖或连锁覆盖的授权。规则(C4)对应于冲突解决策略 2,它表示对某给定的 grant2(s, o, t, a, g),如果存在另一冲突的 grant2(s, o, t', a, g') 且 $t' < t$,则 overriden2(s, o, t, a, g)为真。规则(C5)及(C6)用来推导不被任何其他授权所覆盖或连锁覆盖的授权。我们用了多个谓词来表示授权,grant 是给用户来描写授权;grant1 和 grant2 是系统导出的中间谓词;hold 表示最后由系统推导出的授权。设 R 表示所有的一般规则,即 $R=\{D1,\cdots,D4, H1,\cdots,H4, C1,\cdots,C6\}$。

3.4 稳定模型语义

设 Π 是一个可代理授权程序,Π 的基 B_Π 是所有可能文字的集合,这些基文字是由出现在 Π 中规则的谓词及出现在 S,O,A,T 中的常量所形成。如果两个基文字具有形式 hold(s, o, t, a, g) 和 hold(s, o, t', a, g') 且 $t \neq t'$,则称它们关于主体 s、客体 o 和存取权限 a 是冲突的。注意,这里类型 * 和 + 被认为是冲突的,因为 * 有管理权限而 + 没有。一个规则 r 的基实例是通过将 r 中的每个变量 x 换为 $\delta(x)$ 而得到,这里 $\delta(x)$ 是一个从变量到同类的常量的映射。设 $G(\Pi)$ 表示 Π 中所有规则的基实例。一个基 B_Π 的子集是一致的,如果其中没有冲突或互补的文字。一个解释 I 是基 B_Π 的任何一个一致子集。

定义 2 给定一个可代理的授权程序 Π,Π 的一个解释是 $\Pi \cup R$ 的任意解释。

定义 3 设 I 是可代理程序 Π 的一个解释,其 Π 关于 I 的一个缩减表示为 Π^I,是一个从 $G(\Pi \cup R)$ 通过删除得到的规则集。即删除其规则体中有一个 not L 且 L 属于 I 的所有规则,以及所有剩余规则的体中型为 not L 的公式。

设 Z 是一个基规则集,我们用 pos(Z)表示 Z 的正版本,它是将 Z 中所有否定谓词 $\neg P(t_1,\cdots,t_n)$ 看作一个正谓词,其谓词符号为 $\neg P$。

定义 4 设 M 是 Π 的一个解释,我们说 M 是 Π 的一个解集(answer set),如果 M 是正版本 pos(Π^M)的一个极小模型。如果 M 是 Π 的一个解集,则其所有谓词名为 hold 的文字的子集叫作 Π 的授权解集,表示为 A。

4 语义性质

这一节将探讨可代理程序的一些有用的性质。先证明所得解集是代理正确的,然后证明在一个客体上的所有授权都是首先从安全管理者传递出来。因而,根据我们的冲突解决策略,安全管理者所发出的授权不会被用户所发出的授权所覆盖。最后,探讨保证只有惟一解集的语法性质。

定义 5 一个可代理授权程序 Π 是良定义的(well-defined)如果它存在一个授权解集。

我们下面在主体上定义一个代理关系。

定义 6(在主体上的代理关系 $<_{o,a}$) 设 G 是一个含谓词 hold 的基文字集。对任何主体 s, $s' \in S$,客体 $o \in O$,存取权限 $a \in A$,我们说在 G 中 s 是关于 a 和 o 代理关联于 s',表示为 $s <_{o,a} s'$,如果存在一个授权 hold(s', o, *, a, s) $\in G$,或存在另一主体 s'',使得 $s <_{o,a} s''$,且 $s'' <_{o,a} s'$。

$s <_{o,a} s'$ 意味着存在一个主体序列 s, s_1, $\cdots$, s_n, s' 使得

(下转第 172 页)

(1) 要求暂停学习

学习者学习累了,要求暂时休息或要求智能 Agent 助理随机地表演一些动作,智能 Agent 助理会根据学习者要求采取相应行为,这样学习过程的中断就不会受到惩罚。

(2) 要求智能 Agent 助理给予帮助

当学习者在学习过程中或答题过程中出现疑问,需要智能 Agent 助理给予帮助的时候,学习者可要求智能 Agent 助理给予提示或其他帮助。

结论 本文提出了用模糊数学的评价集方法对学习者进行综合评价,同时设计一个智能 Agent 助理用于人机交互,并将模糊情感识别的结果应用于本系统,在教学过程中根据学生的学习状态生成它的情绪和反应,对学习者的教学辅助策略做出实时调整,达到智能化和人性化助理教学的目的。今后,我们将在此基础上将进行更深入的研究,在系统中改进语音识别,视觉跟踪等技术,使其智能化和个性化有更加突出的表现。

参 考 文 献

1 Wang Zhiliang. Artificial Psychology-a Most Accessible Science Research to Human Brain. Journal of University of Science and Technology Beijing, 2000, 22(5)

2 Xie Yinggang, Wang Zhiliang, Zhang Qin. Humanized clothing recommendation system based on Intelligent Agent. In: the ICS-CA'2006 Conference. Chongqing, 2006, 5

3 Ahn H, Picard R W. Affective-cognitive Learning and Decision Making: A Motivational Reward Framework For Affective Agents. In: First International Conference on Affective Computing and Intelligent Interaction, 2005. 866～873

4 Wu Jianhui, Luo Yuejia. The Approaches to Study of Congnitive Science on Emotion. In: The 1th Chinese Conference on Affective Computing and Intelligent Interaction, 2003(12): 6～11

5 Huang Hui. Fuzzy Set Theory and Its Application. Journal of Zhongnan University of Finance and Economics, CNKI: ISSN: 1003-5230. 0. 1987-01-012

6 Zhang Xiao, Shen Qian. E-Business Based on Fuzzy Theory and PROLOG. Journal of Tianjin University of Commerce. 2000-06-006

7 Feng Jian, Yao Min. Personalized Internet Service Application Based on Fuzzy Theory. Computer Applications and Software, 2004-07-033

8 Dai Jiahao. A Fuzzy Similarity Measure-based Decision Model to Assist Shopping in the Electronic Store. Taiwan, Fengjia University, 2002

(上接第 154 页)

hold (s_1, o, $*$, a, s), hold (s_2, o, $*$, a, s_1), …, hold (s', o, $*$, a, s_n) 都在 G 中。

命题 1 设Π是一个良定义的可代理授权程序,A 是其授权解集,则 A 是代理正确的。

证明:我们根据代理正确性定义来证明。从规则 (C5) 和 (C6) 我们很容易推出条件(a)成立。为证明条件(b),只需证明$<_{o,a}$在 A 中是一个严格偏序。为此,我们定义另一个关系$<'_{o,a}$如下:设 M 是Π 对应的解集,则 $s<'_{o,a}s'$;如果存在授权 grant1(s', o, $*$, a, s)$\in M$ 或存在某主体 s'' 使得 $s<'_{o,a}s''$且 $s''<'_{o,a}s'$,根据一般规则(D2),不难看出$<'_{o,a}$是一个严格偏序。既然$<_{o,a}$是$<'_{o,a}$的子集(hold 从 grant1 导出),说明$<_{o,a}$也是一个严格偏序。

命题 2 设Π是一个良定义的可代理授权程序,A 是它的一个解集,对任何主体集 S 上的关于 A 的代理关系$<_{o,a}$,设 $S'=\{s\mid \exists\, s'(s, s'\in S\wedge(s<_{o,a}s'\vee s'<_{o,a}s))\}$,则管理者 # 是 S'中关于$<_{o,a}$的最小主体。

注意,设有一个偏序集$(C,\leqslant)$,$c\in C$,如果对 $\forall c'\in C$ 都有 $c<c'$,则 c 是偏序集$(C,\leqslant)$中的最小元素。S'中含有 S 中所有关于$<_{o,a}$可比较的主体。

证明:首先注意任何有限非空偏序集$(Z,\leqslant)$有一个较小元素,设 $s\in S'$是关于$<_{o,a}$的一个较小元素,既然 s 是可比较的,则存在 $s'\in S'$,使得 $s<_{o,a}s'$。不失一般性,设 $s<_{o,a}s'$不是由传递推出,既然我们总可以找到这样一个 s',所以 hold (s', o, $*$, a, s)为真。根据规则(C5)和(C6),$s=\#$ 或对某 $s''\in S'$,hold(s, o, $*$, a, s'')为真。第 2 个条件意味着 $s''<_{o,a}s$,说明 s 不是较小,矛盾,所以可得出 $s=\#$。既然 S'中的所有主体都是关于$<_{o,a}$可比较的且 # 是惟一一个较小主体,所以 # 是 S'中关于$<_{o,a}$的最小元素。

这个命题表示,如果一个主体 s 关于$<_{o,a}$可比较于另一个主体 s',则意味着 s 是 s'的代理或 s'是 s 的代理,则 $\#<_{o,a}s$,因此所有的授权都是从 # 开始传递出来的。

结束语 有几个研究课题值得进一步探讨。首先我们考虑要扩展现有模型,使其能处理动态主体、客体和存取权限的分层关系。其次我们将开发考虑动态优先关系的冲突解决方法,用来支持建立在主体、客体及存取权限的不同性质上的不同的策略。最后我们希望实现一个基于逻辑程序设计的我们所提方法的原型。

参 考 文 献

1 Abadi M, Burrows M, Lampson B, et al. A calculus for access control in distributed systems. ACM Trans on programming languages and systems, 1993, 15(4): 706～734

2 Bertino E, buccafurri F, Ferrari E, et al. A logical framework for reasoning on data access control policies. In: Proceedings of the 12th IEEE Computer Society Foundations Workshop, IEEE Computer Society Press, Los Alamitos, 1999. 175～189

3 Crampton J, Loizou G, O'Shea G. A logic of access control. The Computer Journal, 2001, 44: 54～66

4 Gelfond M, Lifschitz V. Classical negation in logic programs and disjunctive databases. New Generation Computing, 1991, 9: 365～385

5 Jajodia S, Samarati P, Subrahmanian V S. A logical language for express- ing authorizations. In: Proceedings of the 1997 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1997. 31～42

6 Ruan C, Varadharajan V. Resolving conflicts in authorization delegations. In: Proceedings of the 7th Australasian Conference on Information Security and Privacy, 2002

7 Woo T, Lam S. Authorization in distributed systems: a formal approach. In: Proceedings of IEEE on Research in Security and Privacy, 1992. 33～50