授权中异常冲突的一种解决方法*)

张明生[1,2]　杨　静[1]

(贵州大学计算机科学与工程学院　贵阳 550025)[1]　(贵州民族学院　贵阳 550025)[2]

摘　要　冲突检测和解决是访问控制授权中的重要问题。对这些问题的探索我们通过研究方法比较进行。首先研究基于逻辑程序的一种授权系统规范,然后通过一个保健部门的典型问题来分析基于传统优先和组织结构的两种冲突解决方案,最后提出基于逻辑程序 LPOD(带有序析取的逻辑程序)的针对高水平授权规范中异常冲突的一种解决方法。

关键词　访问控制,授权,异常冲突,LPOD 逻辑,解决

Approach for Resolution of Exceptional Conflicts in Authorization

ZHANG Ming-sheng[1,2]　YANG Jing[1]

(College of Computer Science & Engineering, Guizhou University, Guiyang 550025, China)[1]

(Guizhou Nationalities University, Guiyang 550025, China)[2]

Abstract　The conflict detection and resolution are important issues in authorization of an access control model. Our research on these problems is performed by using comparative approach. We first investigate the authorization system specification based on logic program, then analyze authorization conflicts resolution based on traditional priority and organization through the explanation of an example on the health care sector, finally we propose a conflict resolution schema for exceptional conflicts in high level authorization specification by using the logic program LPOD (Logic Program with Ordered Disjunction).

Keywords　Access control, Authorization, Exceptional conflicts, LPOD, Resolution

1　前言

现在愈来愈多的研究者都注意到这么一个问题:如何确定一个授权策略,它既有强大的表达能力以适应于用户和数据系统的需求,又具有充足的灵活性以捕捉实际中复杂环境的需求变化。通常我们可以引入"正授权"和"负授权"来满足这种需求,但是又带来了冲突问题。因此,我们需要一些策略和机制来解决由此产生的冲突问题。最近关于冲突研究的工作中,F. Cuppens 等人[2]提出一种基于组织水平的冲突管理,他们的方法是定义冲突解决策略用于分配优先水平给组织予允可和禁止对系统的访问; S. Benferhat 等人[3]提出两种方案来处理安全冲突,一是或然性推理,二是词典编纂推理; F. Cuppens 等人[4]提出了一种一般方法的基本思想,它分析由几个可能的冲突安全策略所决定的一些情形; J. Chomicki 等人[5]则研究一种逻辑框架,它用于检测策略中的行为冲突及其解决办法,在那里冲突作为一种行为约束违反被捕获,并且规则的语义、冲突检测及解决由逻辑程序自动地定义。文献[6-8]试图通过基于规则的控制方法来满足表达需求和解决冲突问题,访问控制被定义成一组规则:条件→授权,"条件"被定义成一组有关主体、行为和客体的约束。

本文提出并实施一种基于逻辑程序 LPOD(带有序析取的逻辑程序)的针对高水平授权规范中异常冲突的一种解决方案。我们首先分析一种确定授权策略和冲突解决的逻辑框架,简单地回顾本文要使用的有关 LPOD 逻辑相关概念和结果,然后研究授权冲突解决方法和相关问题,主要分析异常冲突的解决方案。

本文的余下部分作如下安排:第 2 节描述授权规范和例子解释;第 3 节分析我们用于解决冲突的 LPOD 逻辑的功能;第 4 节总结优先规则和基于组织的冲突决策;第 5 节探索如何使用 LPOD 的偏好关系来处理授权冲突;最后是简单小结及展望。

2　授权规范

在访问控制中有许多方法用来表示授权。研究文献[2],我们提出了一种基于逻辑程序的授权规范并作相应的分析。

2.1　授权系统

一个授权系统 SYS 可表示为如下形式:

SYS = AC P∪ CRS ∪ DOM ∪ Constraint

其中它的四个组件分别是规则的集合,具体解释如下:

• ACP(Access Control Policy)　访问控制策略,是一组形如下列形式的规则:

Permission(s,a,o) ← Condition,或

Prohibition(s,a,o) ← Condition

这里 Condition = $P_1 \wedge \cdots \wedge P_n$, P_i 表示一般合式公式, $i=1, \cdots, n$。Permission(s,a,o)表示允许主体 s 对客体 o 作行为 a 的访问;而 Prohibition(s,a,o)则表示禁止。

• CRS(Conflict Resolution Strategy)　冲突解决策略,是一组形如下列形式的规则的集合:

A-Permission(s,a,o) ← Condition 或

A-Prohibition(s,a,o) ← Condition

*)基金项目:贵州大学人才引进科研基金(2006)。张明生　博士研究生,主要研究方向为信息安全、形式化分析;杨　静　博士,副教授,硕士生导师,主要研究方向为人工智能、软件形式化。

其中 A-Permission 和 A-Prohibition 分别表示实际正和负的授权，对应于原来的 prima facie(意思为“第一次出现”)授权允可 Permission(s,a,o)和禁止 Prohibition(s,a,o)。

• DOM(DOMain) 域，确定和表示一个实际的数据系统，其定义为：

$DOM = IDom \cup RDom$

其中 IDom 称为域实例(domain instances)，表示系统实事的一个有限集合；RDom 称为域规则(domain rules)，表示系统有关规则(表示数据系统各成分间的关系)的一个有限集合。

• Constraint 约束，表示系统中有关谓词应满足的条件，其形式为：

← constraint condition

2.2 有关冲突管理的一些概念

2.2.1 冲突(Conflict)

我们定义和说明本文重要的概念“冲突”，冲突(Conflict)的定义如下：

$\exists s, \exists a, \exists o$, Permission($s,a,o$) ∧ Prohibition($s,a,o$)

在一个授权模型中，当可能存在两个正负相反的授权规则导致允许和禁止一个主体对同样一个客体实施同样的行为时，冲突就可能发生。在实际中实施一个访问策略时，授权冲突一定要解决。解决冲突的基本思想是定义一个冲突解决策略，即 CRS，它可以把 prima facie 授权“转化”为实际授权，这样在实际授权中就可以消除冲突。

2.2.2 原本冲突(prima facie conflict)

SYS ⊢ $\exists s, \exists a, \exists o$, Permission($s,a,o$) ∧ Prohibition($s,a,o$)

表示冲突解决之前系统原先就存在的冲突。

2.2.3 实际冲突(actual conflict)

SYS ⊢ $\exists s, \exists a, \exists o$, A-Permission($s,a,o$) ∧ A-Prohibition($s,a,o$)

表示冲突解决之后系统仍然存在的冲突。

2.2.4 有效策略(effective strategy)

当一个冲突解决策略保证在规范的系统中不再有实际授权的冲突，则称它是有效的。

2.3 一个例子

例 1 下列规则确定了一个实际授权系统 SYS_1：

R1: Prohibition(x, read, y) ← nurse(x) ∧ medical-record(y)

R2: Permission(x, read, y) ← nurse(x) ∧ medical-record(y) ∧ patient-record(y,z) ∧ urgent-case(z)

R3: Permission(x, read, y) ← physician(x) ∧ medical-record(y) ∧ patient-record(y,z) ∧ attending-physician(z,x)

R4: Prohibition(x, read, y) ← physician(x) ∧ medical-record(y) ∧ suspended(x)

R5: Prohibition(x, read, y) ← nurse(x) ∧ medical-record(y) ∧ suspended(x)

ACP_1 = {R1, R2, R3, R4, R5}

首先考虑一个简单的冲突管理策略 CRS_1：

CRS_1: A-Prohibition(s,a,o) ← Prohibition(s,a,o)

A-Permission(s, a, o) ← Permission(s, a, o) ∧ ¬Prohibition(s,a,o)

$IDOM_1$: physician(John) ∧ suspended(John) ∧ nurse(Peter) ∧ medical-record(3#) ∧ patient-record(3#, Mary) ∧ attending-physician(Mary, John) ∧ urgent-case(Mary)

$Constraint_1 = \emptyset$

我们可以得出下列的结果：

• Prohibition(Peter, read, 3#) 由 R1 得

• Permission(Peter, read, 3#) 由 R2 得

• Prohibition(John, read, 3#) 由 R3 得

• Permission(John, read, 3#) 由 R4 得

• A-Prohibition (John, read, 3#) 由 S1 得

• A-Prohibition (John, read, 3#) 由 S1 得

因此，我们就用这个例子显示出如何发现冲突和解决冲突。

3 LPOD 逻辑

LPOD 指带有序析取的逻辑程序，是由 G. Brewka 等人[1]提出的，它的语义是基于回答集的偏好关系，不同于其它逻辑程序的语义。LPOD 有成熟的理论基础，能通过回答集解决器(solver)实施，是非常有用的工具。在本文中我们将用它来帮助解决授权冲突，因此在这一节中简单总结下面将要用到的有关 LPOD 的概念和相关结果。

3.1 LPOD 概念

一个 LPOD 是下列形式规则的集合：

$C_1 \times \cdots \times C_n \leftarrow A_1, \cdots, A_m$, not $B_1, \cdots$, not B_k

其中 C_i, A_j, and B_l 是基本文字。规则的头的直观意义是：如果 C_1 为真则 C_1，如果 C_1 为假则 $C_2, \cdots$，如果所有的 $C_1, \cdots, C_{n-1}$ 为假则 C_n。

LPOD 在语义上不同于其它逻辑程序，比如不是最小回答集。它能转化为正规的逻辑程序，但其变形方法又不同于其它析取联结词的逻辑程序。

3.2 LPOD 偏好回答集

在得到一个 LPOD 的转换形式的回答集后，我们需要挑拣其中的一个(些)以使其满足 LPOD 程序的定义和语义。通常挑拣的方法是基于下列具有实际背景的三个标准(原则)：

(1) 满足度(degree of satisfaction)

对 LPODP 的一个规则：

$C_1 \times \cdots \times C_n \leftarrow A_1, \cdots, A_m$, not $B_1, \cdots$, not B_k

程序 P 的一个回答集 S 满足该规则达到度 $deg_S(r)$ 定义如下：

$$deg_S(r) = \begin{cases} 1, & \text{if } Aj \notin S, \text{or } Bi \in S \\ \min\{r \mid Cr \in S\}, & \text{otherwise} \end{cases}$$

(2) 几种偏好标准(preference criterions)

• 基数偏好(cardinality based preference) $\succ_c$

$S_1 \succ_c S_2$ 当且仅当 $|S_1^i(P)| > |S_2^i(P)|$ 对某一 i，且 $|S_1^j(P)| = |S_2^j(P)|$ 对所有 $j < i$

这里 $S^i(P) = \{ r \in P \mid deg_S(r) = i \}$

• 包含偏好(inclusion based preference) $\succ_i$

$S_1 \succ_i S_2$ 当且仅当 $S_1^k(P) \supset S_2^k(P)$ 对某一 k，且 $S_1^j(P) = S_2^j(P)$ 对所有 $j < k$

• Pareto 偏好(Pareto preference) $\succ_p$

$S_1 \succ_p S_2$ 当且仅当 $deg_{S1}(r) < deg_{S2}(r)$ 对某一 r，且 $deg_{S1}(r') > deg_{S2}(r')$ 不存在 r'

• 偏好间的关系(relation between three preferences)

$\succ_p \Rightarrow \succ_i \Rightarrow \succ_c$

4 基于优先和组织的冲突管理

为便于比较我们提出的冲突解决方法，我们先在本节中解释基于优先的冲突管理和基于组织的冲突管理[2]。首先我们给出研究目标，即异常冲突的概念，它涉及到授权规则间的三种关系。

4.1 授权规则间的三种关系

· 无关

规则 Ri 和 Rj 是无关的，当且仅当

(RDom ∪ Constraint) ⊢ ¬(condition(Ri) ∧ condition(Rj))

例子：在下列约束 C1 下，2.3 中 R1 和 R2 是无关的，R3 和 R4 也是无关的，但 R1 与 R3 间无冲突，而 R4 与 R5 则不考虑冲突问题。

C1：← nurse(x) ∧ physician(x)

· 异常

规则 Ri 异常于 Rj，当且仅当

(RDom∪Constraint) ⊢ condition(Ri) → condition(Rj)

例子：R2 异常于 R1。

· 潜在

两个规则 Ri 和 Rj 间有一个潜在冲突，如果：

(1) Ri 导出一个授权"允许"；(2) Rj 导出一个授权"禁止"；(3) Ri 与 Rj 不是无关的。

注意：对潜在冲突不同环境下意义是不同的。上面的定义应该是 prima facie 潜在冲突。在有 CRS 的情况下，潜在冲突指不能由 CRS 消除的实际冲突。

例子　R2 和 R5 之间存在潜在冲突。

4.2 优先冲突管理

· 冲突管理方案

引进一个偏序的优先集合，把它们分配给授权规则以解决授权冲突。因此授权可以是这种形式：Permission(s,a,o,p) 和 Prohibition(s,a,o,p)，其中 p 表示分配的优先权。

对应于第 2 节中的 CRS，有一个称为全局冲突解决策略 GCRS，由下列两个规则组成：

–Permission(s,a,o,p) ∧ ¬Higher-Prohibition(s,a,o,p) → A-Permission(s,a,o)

–Prohibition(s,a,o,p) ∧ ¬Higher-Permission(s,a,o,p) → A-Prohibition(s,a,o)

其中，Higher-Prohibition(s,a,o,p) 是谓词，意为存在一个禁止关于主体 s、行为和客体 o，并且比 p 的优先权高。Higher-Permission(s,a,o,p) 的意思与 Higher -Prohibition (s,a,o,p)类似。

因此，我们通过 GCRS 能解决冲突，但不是对所有情形都是有效的。

· 两个缺点

a)只能解决完全序优先，不适于一般的偏序；

b)它要求有约束：

CP：← Permission(s,a,o,p) ∧ Prohibition(s,a,o,p)

4.3 组织冲突管理

我们考虑把授权规则中的条件结构化成下列的形式：

–condition $=_{def}$ cond-subject(s) ∧ cond-action(a) ∧ cond-object(o) ∧ constraint(s,a,o)

授权规则中的条件结构化使我们能易于检测和处理冲突(比如多项式时间甚至线性时间)，还可以产生推理机制，比如我们可以把角色的层次与授权的继承关系联系起来。

组织冲突管理的基本思想主要有下列两点：

(1)授权源于"组织"

我们考虑通过"组织"来获取授权。例如，一个具有优先权 p 的 prima facie 允可 Permission(s,a,o,p)可以从具有优先权 p 的组织允可 Permission(org,r,a,v,c,p)导出：

Permission(s,a,o,p) ← Permission(org,r,a,v,c,p) ∧ Empower(org,s,r) ∧ Use(org, o,v) ∧ Consider(org,a,a) ∧ Hold(org,s,a,o,c)

其中

角色 r、活动 a 和视图 v 分别抽象于主体 s、行为 a 和客体；

Empower(org,s,r)：组织 org 授予主体 s 角色 r；

Use(org,o,v)：组织 org 以视图 v 使用客体 o；

Consider(org,a,a)：组织 org 考虑行为 a 实现活动 v；

Hold(org,s,a,o,c)：主体 s、行为 a 和客体 o 之间拥有约束 c；

Prohibition(s,a,o,p)的定义类似于 Permission(s,a,o,p)。

(2)分离约束

当 Permission(s,a,o,p)和 Prohibition(s,a,o,p) 分别源于 Permission(org,$r1,a1,v1,c1,p$) 和 Prohibition(org,$r2,a2,v2,c2,p$)时，其中($r1\neq r2 \vee a1\neq a2 \vee v1\neq v2 \vee c1\neq c2$)为真，Permission($s,a,o,p$)和 Prohibition($s,a,o,p$)将导致一个实际冲突，而 Permission(org,$r1,a1,v1,c1,p$)和 Prohibition (org,$r2,a2,v2,c2,p$)是不冲突的，这种情况称为潜在冲突。

分离约束是解决这种潜在冲突的手段。对角色 $r1$ 和 $r2$ 的分离约束意味着主体 s 不能被同时给予以 $r1$ 和 $r2$ 角色，因此 Permission(s,a,o,p) 和 Prohibition(s,a,o,p)不可能产生冲突。同样我们可以定义对 a，v 和 c 的分离约束。

5 基于 LPOD 的冲突解决

基于上面的分析，现在我们提出一种使用 LPOD 逻辑来进行冲突决策的方法。在本节中，我们主要是基于第 2 节中的例 1 来解释和分析对异常冲突解决的思想及方法。

例 2

R1：Prohibition(x, read, y) ← nurse(x) ∧ medical-record(y) ∧ default-case( )

R2：Permission(x, read, y) ← nurse(x) ∧ medical-record(y) ∧ patient-record(y,z) ∧ urgent-case(z)

R3：A-Permission(x, read, y) × A-Prohibition(x, read, y) ← default-case() ∧ urgent-case()

R2 是 R1 的异常冲突，规则 R3 是由我们引进的用于解决这一冲突。

根据 LPOD，R3 意为 A-Permission(x, read, y)优先于 A-Prohibition(x, read, y)，即"病人 z 在紧急情况下" urgent-case()"正授权"优先于"一般情况"default-case()。

考虑下面系统可能的两个事实情况：

DOM1：nurse(Peter) ∧ medical-record(3 #) ∧ patient-record(doc-31,Mary) ∧ urgent-case(Mary) ∧ (default-case( ) ←urgent-case( ))

DOM2：nurse(Peter) ∧ medical-record(doc-31) ∧ default-case(z) ∧ (¬urgent-case(z))

(下转第 115 页)

$f_5=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$，$-600\leqslant x_i\leqslant 600$，$i=(0,1\cdots 30)$，该函数有多个全局极小点。

计算结果和比较内容如表 2 所示，每个函数用 IHQGA，QGA，GA 计算 100 次，算法性能从平均运行次数 t，最好解的平均值 a，未收敛次数 w 等方面分别与 QGA，GA 进行比较。

表 2　测试函数与比较结果

函数	IHQGA			QGA			GA		
名	a	t	w	a	t	w	a	t	w
f_1	0	421.6	0	1.2×10^{-5}	826.3	1	1.3×10^{-4}	962.7	2
f_2	0	625.3	0	1.1×10^{-5}	943.2	2	1.4×10^{-5}	1235.2	8
f_3	1.02×10^{-5}	612.4	2	3.14×10^{-2}	1005.4	5	2.68×10^{-2}	1401.2	13
f_4	−1.009328	604.5	4	1.916143	1262.7	7	12.305423	2461.5	12
f_5	2.68×10^{-4}	2643.7	5	12.62	3215.2	9	103.4	5264.2	15

结束语　本文提出了一种新的混合量子遗传算法，并描述了基因表达结构和相应的算法，提出了旋转量子门的自适应操作；另外，量子交叉和拟 Newton 局部搜索的引入，不仅增强了种群的多样性，也加快了种群的收敛速度，使得算法性能得以较大提高；从理论上证明了算法以概率 1 收敛到全局最优解，并与传统量子遗传算法、遗传算法作了比较。试验也表明该算法在收敛速度和收敛精度上都优于传统量子遗传算法和遗传算法；继续以下几个方面的工作：

(1)将该算法的思想应用多目标优化和组合优化问题中。

(2)改进算法的终止条件，使其具有自适应性，摆脱人为设定；进一步改进旋转量子门，使得算法更好地逃离局部最优。

参 考 文 献

[1] Shor P W. Algorithms for quantum Computation: Discrete Algorithms and factoring[C]//Proceedings of the Annual Sympium Foundations Computer Science Sante Fe. NM, 1994: 124-134

[2] Narayanan A . Moore M Quantum inspired genetic algorithm [A]// Proc. of IEEE International Conference on Evolutionary Computation[C]. Piscataway: IEEE Press, 1996: 61-66

[3] Han K H, Kim J H. Genetic quantum algorithm and its application to combinational optimization problems [A] // Proc. of IEEE Conference on Evolutionary Computation [C]. Piscataway: IEEE Press, 2000: 1354-1360

[4] Han K H, Park K H, et al. Parallel quantum-inspired genetic algorithm for combinatorial optimization problems[A]// Proc. of IEEE Conference on Evolutionary Computation [C]. Piscataway: IEEE Press, 2001: 1429-1442

[5] Zhang Weixiu, Liang Yi. Mathematical Foundation of Genetic Algorithms[M]. Xi'an Jiantong University Press, 2000: 152-155

[6] 王宇平，李英华. 求解 TSP 的量子遗传法. 计算机学报，2007，30(5)：748-755

[7] 张葛祥，金炜东. 基于量子遗传算法的特征选择算法. 2005，22(5)：1000-8152

[8] 王小平，曹立明. 遗传算法——理论、应用于软件实现. 西安交通大学出版社，2002

(上接第 103 页)

对于 DOM1，我们可以得到上面程序逻辑{R1，R2，R3}的回答集：Anset1＝{Prohibition(Peter, read, 3＃)，Permission(Peter, read, 3＃)，A-Permission(Peter, read, 3＃)} 和 Anset2＝{Prohibition(Peter, read, 3＃)，Permission(Peter, read, 3＃)，¬A-Permission(Peter, read, 3＃)，A-Prohibition(Peter, read, 3＃)}。根据 LPOD 的偏好回答的选择标准，我们挑选 Anset1 作为该程序的回答集。因此得到：A-Permission(Peter, read, 3＃)。

对于 DOM2，我们只得到上面程序逻辑{R1，R2，R3}的一个回答集：Anset ＝ {Prohibition(Peter, read, 3＃) }，所以我们得到 A-Permission(Peter, read, 3＃)。

上面的冲突决策中，在回答集中选取授权时，我们规定实际授权 A-总是优于 prima facie 授权，如果没有实际授权，则 prima facie 授权就是实际授权。

下面是对一般情形的异常冲突解决的严格推导分析：

R1：Prohibition(s,a,o) ← condition1 ∧ condition2

R2：Permission(s,a,o) ← condition1 ∧ condition3

R3：A-Permission(s,a,o) × A-Prohibition(s,a,o) ← condition2 ∧ condition3

R2 是 R1 的异常冲突，我们假设 condition3 → condition2。如果域规则满足 condition1 和 condition3，我们得到偏好回答集：{ Prohibition(s,a,o)，Permission(s,a,o)，A-Permission(s,a,o) }，这样就获得 A-Permission(s,a,o) 。如果域规则满足 ondition1 和 condition2，但不满足 condition3，我们得到 { Prohibition(s,a,o)}，因此得相应的实际授权。

由上可知所提出的方法对于异常冲突解决是完备的，并且与第 4 节中的方法的效果是一样的，但其更具灵活性。

结束语　在本文中，我们提出了一种使用 LPOD 逻辑来解决异常冲突的一种方法，这种方法是基于文字和上下文信赖的。与其它方法相比较(如本文所提及的方法)，我们所提出的方法更具灵活性和更适用于实际问题。本文的主要贡献是提出了一种解决访问授权冲突的思想和方法，在下一步研究工作中，我们将用它来处理潜在冲突，并且考虑授权系统中主体、行为和客体的层次和继承等。我们也认为在文献[9,10]中的框架下作相应的探索研究工作是十分有意义的。

参 考 文 献

[1] Brewka G, Niemelä I, Syrjänen T. Logic programs with ordered disjunction. Computational Intelligence, 2004, 20 (2): 335-357

[2] Cuppens F, Cuppens-Boulahia N, Ben Ghorbel M. High Level Conflict Management Strategies in Advanced Access Control Models // Electronic Notes in Theoretical Computer Science (ENTCS), July 2007, 186: 3-26

[3] Benferhat S, El Baida R, Cuppens F. A Stratification-Based Approach for Handling Conflicts in Access Control // 8th ACM Symposium on Access Control Models and Technologies (SACMAT'03). Lake Come, Italy, June 2003

[4] Cuppens F, Cholvy L, Saurel C, et al. Merging regulations: analysis of a practical example. International Journal of Intelligent Systems, 2001, 16(11)

[5] Chomicki J, Lobo J, Naqvi S. A Logical Programming Approach to Conflict Resolution in Policy Management // Proceedings of International Conference on Principles of Knowledge Representation and Reasoning. 2000: 121-132

[6] Jajodia S, Samarati P, Sapino M, et al. Flexible support for multiple access control policies. ACM TODS 26, 2001(2): 214-260

[7] Bertino E, Catania B, Ferrari E, et al. A Logical Framework for Reasoning about Access Control Models. ACM Transactions on Information and System Security, 2003, 6(1)

[8] Bertino E, Catania B, Ferrari E, et al. On comparing the Expressing Power of Access Control Model. In Foundations of Computer Security (FCS'04), Turku, Finland, July 2004

[9] Barker S, Stuckey P. Flexible access control policy specification with constraint logic programming. ACM Trans. on Information and System Security, 2003, 6(4): 501-546

[10] Barker S. Action-status access control // Symposium on Access Control Models and Technologies. SACMAT'07, June 20-22, 2007, Sophia Antipolis, France. Proceedings of the 12th ACM symposium on Access control models and technologies, 2007