基于逻辑合一的访问控制规则描述

韩道军[1,2] 黄泽龙[1] 翟浩良[1] 李 磊[1]
(中山大学软件研究所 广州 510275)[1] (河南大学数据与知识工程研究所 开封 475004)[2]

摘 要 现有的访问控制规则描述方式不易表达一类主体、客体间具有包含关系的访问控制规则。针对此问题,提出一种基于逻辑中合一思想的算法。算法首先将访问控制请求转换为逻辑提问,同时根据逻辑回答给出相应的访问控制请求应答;然后使用事实描述访问控制规则中的各个要素,并通过在系统运行过程中对非 ground 事实的变量的动态例化实现灵活的访问控制。最后,通过一个实例及分析说明了算法的有效性。

关键词 访问控制,逻辑合一,规则描述
中图法分类号 TP301 **文献标识码** A

Access Control Rule Description Based on Logic Unify

HAN Dao-jun[1,2] HUANG Ze-long[1] ZHAI Hao-liang[1] LI Lei[1]
(Software Research Institute of Sun Yat-Sen University, Guangzhou 510275, China)[1]
(Institute of Data and Knowledge Engineering, Henan University, Kaifeng 475004, China)[2]

Abstract Traditional methods are hard to describe the subsume relationship of subjects and objects among some access control rules. In this paper, we built an algorithm based on logic unify to resolve which problem. Firstly, we converted access control request to logic question, obtained the access control result by means of logic answer. Next, we utilized the facts to describe the each component of access control, and we realized the flexible access control by using the dynamic instantiation of variables in non-ground facts during the system running period. Finally, our experiment results show that our algorithm is effective.

Keywords Access control, Logic unify, Rule description

访问控制是一种常用的资源保护手段,也是复杂信息系统(Complex Information System,CIS)的一个重要组成部分。现有CIS访问控制规则描述中,采用模式(schema,某类对象的描述)而非传统的实例(instance,访问控制中的具体对象,如用户、权限等)为描述对象,如ABAC[1]、UCON[2]中的属性描述方式(以属性方式描述主体、客体和操作等)、RBAC[3]中的角色(相同权限的一类用户)等。这种方式具有两个优点:(1)便于管理。使用模式的描述易于理解、更有意义。(2)便于扩展。现有权限定义与主体和客体的实例相关,这不符合某些系统的实际情况。例如在审批系统中,业务流程运行产生的许多待管理对象(申请单)无法预先定义。

以模式为基本对象的规则描述方式能够适用于常见的CIS,但是对于某些CIS中存在一类特殊的主体、客体具有包含关系的访问规则描述起来则较为困难。其中的主体、客体具有包含关系的访问规则是指系统中存在 $P(S_1,Op,R_1)$ 和 $P(S_2,Op,R_2)$,这里 $S_2 \subset S_1$,$R_2 \subset R_1$,且 R_2 仅仅能够被 S_2 以 Op 方式访问;$P(S,Op,R)$ 表示主体 S 能够对客体 R 执行操作 Op。这种情形常见于一类临时授权,即特殊限定某个访问控制规则中的主体或客体的范围。具体地,系统中存在访问控制规则:S_1 能够对客体 R_1 执行操作 Op;由于外界环境变化,需要限制客体 R_1 中的某类客体 R_2 仅能由 S_1 中的某类主体 S_2 执行操作 Op,而其他用户 $u(u \in \{S_1 - S_2\})$ 不具有对 R_2 的操作(Op)权限。

按照现有基于XACML[4]的策略描述方式,可以产生两条对应的访问规则:

P_1:Permit←sCategory=S_1,aID=Op,rCategory=R_1;

P_2:Permit←sCategory=S_2,aID=Op,rCategory=R_2。

假设存在一个用户 u,$u \in S_1$ 且 $u \notin S_2$,请求以 Op 方式访问 r_2,$r_2 \in R_2$。根据 $R_1 \subset R_2$,对于 r_2,存在 $r_2 \in R_1$。对用户 u 的访问控制请求可以描述为{sCategory=S_1,aID=Op,rCategory=R_1,rCategory=R_2}。根据资源保护的要求,显然,该请求不应该获得许可。但是根据规则 P_1,却可以得到该请求的结果为Permit,这与实际情况不符,产生矛盾。

对于这类问题,现有的解决办法为:(1)设置系统的访问控制策略合并方式[1]。该方法的不足之处是需要涉及到系统中的所有访问控制规则,较为复杂。(2)根据主体、客体集合

到稿日期:2010-11-28 返修日期:2011-02-24 本文受国家自然科学基金(61003140)资助。
韩道军(1979—),男,博士生,讲师,主要研究领域为机器学习、形式概念分析、软件工程,E-mail:hdj@henu.edu.cn;**黄泽龙**(1985—),男,硕士,主要研究领域为软件工程、数据库与知识库;**翟浩良**(1983—),男,博士生,主要研究领域为访问控制、软件工程;**李 磊**(1951—),男,教授,博士生导师,主要研究领域为数据库与知识库。

间的包含关系，产生一系列的新规则[4]。该方法的不足之处是引入的新规则数量较多。不妨假定 S_1 中有 n 个分类，R_1 中有 m 个分类。若准确地描述该主体与资源间的关系，需要规则的个数为 $n*m$ 个，这种方式大大增加了规则数量。下面通过一个实例说明问题产生的场景及现有解决方法的不足。

例 1　在企业信息管理系统中，存在访问规则 P_3：销售部的人能够读取销售计划。即 P_3：Permit←sCategory＝Sales，aID＝Read，rCategory＝Sale plans，可以用图 1 表示。

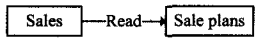

图 1　访问规则示例(未分类)

考虑到部门及资源的进一步分类，图 1 可以等价表示为图 2。

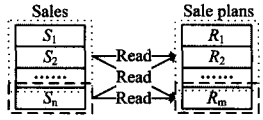

图 2　访问规则示例(按分类方式)

假设外界环境发生变化，系统临时增加访问控制请求 P_4：销售部类别为 S_n 的用户才能够读取类别为 R_m 的销售计划，此时的访问控制关系可以用图 3 表示。并且可以从图 3 中看出，由于 P_4 的引入，使得 P_3 不再成立，需要根据销售部与销售计划的详细分类情况分别描述访问控制规则。此时，描述该主体与资源间的关系需要规则的数量为 $n*m$。

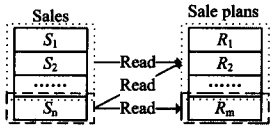

图 3　访问规则示例(引入 P_4 后)

产生上述问题的原因是：新增的访问控制规则(新规则)改变了系统中已有的某些规则(旧规则)，并且新旧规则间主、客体的包含关系使得对应资源的管理规则变得复杂，需要更多的规则描述。

针对上述问题，本文提出一种基于逻辑中合一思想的算法，用以扩展访问规则，并使用非 ground 事实(事实中含有变量)描述主体和资源，将访问控制请求转换为逻辑提问，将逻辑回答转换为相应的访问控制请求应答。由于本文中的算法使用非 ground 事实(非 ground 事实可以表示模式级别的知识)，因此可以在运行中对其中的变量赋值及例化，以减少访问规则的数量，提高访问控制的动态性。

1　基础知识介绍

在介绍本文算法之前，首先引入逻辑程序设计中的一些基本概念[6,7]。

定义 1(原子公式，Atom)　设 P 是一个 n 元谓词符号，$t_1,t_2,\cdots,t_n$ 是项，则称 $P(t_1,t_2,\cdots,t_n)$ 为原子公式，或简称原子。

定义 2(公式，Formula)　公式由且仅由有限次使用如下(1)、(2)、(3)而得：

(1)原子公式是公式；

(2)如果 A,B 是公式，则 $(\neg A),(A\wedge B),(A\vee B),(A\rightarrow B),(A\leftrightarrow B)$ 是公式；

(3)如果 A 是公式，x 是个体变量，则 $(\forall x)A$ 和 $(\exists x)A$ 是公式。

定义 3(文字，Literal)　原子公式(Atom)称为正文字，原子公式的否定称为负文字。文字由正、负文字组成。

定义 4(子句，Clause)　子句是若干个文字组成的有限析取式，如 $C_1=P_1\vee P_2\vee\cdots\vee P_m\vee\neg Q_1\vee\neg Q_2\vee\cdots\vee\neg Q_n$。

定义 5(Horn 子句)　Horn 子句是一类特殊的子句，它被定义为至多包含一个正文字的子句，表示为 $P\leftarrow Q_1,Q_2,\cdots,Q_n$。

定义 6(替换，Substitute)　替换 θ 是一个有限集合 $\{x_1/t_1,\cdots,x_m/t_m\}$，其中 x_i 是不同的个体变量，t_i 是项，且 $x_i\neq t_i$。如果所有的 t_i 是个体常量，则 θ 是 ground 替换，并记为 θ_g；如果 t_i 是变量，则 θ 是 exchange 替换，记为 θ_e。

定义 7(合一，Unify)　设 L,M 是文字，L,M 是可合一的，如果有替换 θ，使得 $L\theta=M\theta$，其中 θ 叫做合一替换。

例如，有两个公式：F_1＝"$p(a,X,b,X,Y,W)$"和 F_2＝"$p(U,U,T,V,c,T)$"，此时存在一个替换 $\theta=\{U/a,X/a,V/a,T/b,W/b,Y/c\}$，使得 $F_1\theta=F_2\theta=p(a,a,b,a,c,b)$。因此称 F_1 和 F_2 可合一，其合一替换是 θ，$\theta=\{U/a,X/a,V/a,T/b,W/b,Y/c\}$。

定义 8(规则，Rule)　定义具有如下形式的 Horn 子句为规则：$P_0(\vec{U_0})$：$-P_1(\vec{U_1}),P_2(\vec{U_2}),\cdots,P_n(\vec{U_n})$，其中 P_i 为谓词，$\vec{U_i}$是个体常量和变量的元组，$1\leqslant i\leqslant n,n\geqslant 1$。$P_0(\vec{U_0})$称为规则头，$P_1(\vec{U_1}),P_2(\vec{U_2}),\cdots,P_n(\vec{U_n})$称为规则体。

定义 9(事实，Fact)　设 P 是 n 元谓词，$t_1,t_2,\cdots,t_n$是项，即 $t_1,t_2,\cdots,t_n$ 可以是常量，也可以是变量，则称 $P(t_1,t_2,\cdots,t_n)$为事实。

事实 $P(t_1,t_2,\cdots,t_n)$中，如果 $t_1,t_2,\cdots,t_n$ 均为个体常量，那么事实 P 是 ground 事实，否则事实 P 是非 ground 事实。例如，$P(a,b,c,d)$是 ground 事实，其中 a,b,c,d 是个体常量，P 为四元谓词；而 $P(a,b,X,d)$是非 ground 事实，其中 X 是个体变量。

定义 10(提问，Question)　设 Q 为 n 元谓词，$t_1,t_2,\cdots,t_n$是项，则称？$-Q(t_1,t_2,\cdots,t_n)$为提问。换句话说，提问是只有负文字的子句，本文限定提问中只有一个负文字。

2　算法介绍

针对现有的访问控制规则描述方式不易表达一类主体、客体间具有包含关系的访问控制规则，本文提出一种基于逻辑中合一思想的算法。该算法首先将访问控制请求转换为逻辑提问，同时根据逻辑回答给出相应的访问控制请求应答；然后使用事实描述访问控制规则中的各个要素，并通过在系统运行过程中对非 ground 事实的变量的动态例化来实现灵活的访问控制。下面进行详细介绍。

2.1　算法基础

在给出详细的算法描述前，我们先定义相关的谓词、事实、规则及提问，并将访问控制的请求转换为逻辑提问。

(1)谓词

定义相关谓词及项如下：

can_access(*Who*,*Op*,*Obj*)：描述表示用户 *Who* 可以对客体 *Obj* 执行操作 *Op*。

who_attr(*Who*,T_1,T_2,…,T_n)：描述用户信息，表示用户 *Who* 与 n 个变量 T_1 到 T_n 的关系。

attr_op_obj(*Who*,T_1,T_2,…,T_n,*Op*,*Obj*)：用来描述权限约束，表示用户 *Who* 与 n 个变量 T_1 到 T_n 与操作 *Op* 和 *Obj* 的关系。

(2)事实

在使用逻辑程序实现访问控制矩阵和基于角色的访问控制时，逻辑程序中的事实都是 ground 的。显然，ground 事实只能描述实例，无法描述模式级内容。本文允许逻辑程序中的事实是非 ground 的，并且在逻辑程序执行过程中，允许对非 ground 事实中的变量进行赋值。显然，在本文的逻辑程序中存在两种事实，分别是用户描述事实和权限约束事实。

用户描述事实：*who_attr* (u,T_1,T_2,…,T_n)，其中 u 为常量，指具体某个用户；T_i 可以为常量，也可以为变量，可用来表示用户的属性，$1\leqslant i\leqslant n, n\geqslant 1$。

权限约束事实：*attr_op_obj* (*Who*,T_1,T_2,…,T_n,*Op*,*Obj*)，其中 *Who* 表示某个用户，*Who* 和 T_i 都可以为常量，也可以为变量，且 T_i 可用来表示用户的属性；*Op* 和 *Obj* 为常量，分别表示具体某个操作和具体某个客体；$1\leqslant i\leqslant n, n\geqslant 1$。例如，设 $n=3$，令变量 T_1 表示用户属性角色，变量 T_2 表示用户属性所在部门，T_3 表示用户属性政治面貌(中共党员，团员，群众)。假定存在一个非 ground 事实 *attr_op_obj*(*W*,科员,综合部门,*T*,提交,申请表)，则其含义为：允许用户属性角色为“科员”、所在部门为“综合部门”的用户提交申请表，而不必关心用户的标识(身份)和政治面貌是什么。

显然，逻辑程序中的变量设置是模式级别的。如果对这些变量进行变化，则意味着动态逻辑程序。对非 ground 事实中的变量进行变化，可以实现动态的访问控制。例如对上面的事实 *attr_op_obj*(*W*,科员,综合部门,*T*,提交,申请表)中的变量 *W* 进行赋值，令 *W*＝小王，则得到新的事实 *attr_op_obj*(小王,科员,综合部门,*T*,提交,申请表)，新的事实的含义是只允许综合部门的科员小王提交申请表。为此，我们做以下定义。

定义 11(模式事实)　我们称非 ground 的事实为模式事实，比如事实 *attr_op_obj*(*a*,*b*,*X*,*Y*,*Op*,*Obj*)，其中 *a*,*b*,*Op*,*Obj* 是个体常量，*X*,*Y* 是个体变量。

定义 12(例事实)　对模式事实中的个体变量进行赋值可以得到新的事实，我们称这新的事实为原模式事实的例，简称为例事实。

例如，有一模式事实 *attr_op_obj*(*a*,*b*,*X*,*Y*,*Op*,*Obj*)，其中 *a*,*b*,*Op*,*Obj* 是个体常量，*X*,*Y* 是个体变量。*X* 赋值为 *c*，可以得到模式事实 *attr_op_obj*(*a*,*b*,*X*,*Y*,*Op*,*Obj*)的例事实 *attr_op_obj*(*a*,*b*,*c*,*Y*,*Op*,*Obj*)。

由模式事实和例事实的定义可知，例事实由模式事实派生出来，则模式事实与其派生出来的例事实之间具有包含关系。这种包含关系可以表述系统的访问控制规则间的包含关系，减少访问控制规则数量。

(3)规则

根据访问控制规则的含义，可以将访问规则转换为逻辑规则，形式如 $Rule_1$：*can_access*(*Who*,*Op*,*Obj*) :-*who_attr*(*Who*,T_1,T_2,…,T_n),*attr_op_obj*(*Who*,T_1,T_2,…,T_n,*Op*,*Obj*)，其中 *Who*,*Op*,*Obj*,T_i 为变量，$1\leqslant i\leqslant n, n\geqslant 1$。*Who* 表示用户，*Op* 表示操作，*Obj* 表示客体，T_i 可以用来表示用户的属性。例如在实际应用中可以把 T_1 解释为主体的角色属性、T_2 解释为主体的所在部门属性等等。T_i 也可以表示与用户相关的其它条件。

本文中的逻辑规则仅仅描述了访问控制中必不可少的几个要素(主体、客体、操作)，常见的一个要素——条件(假定使用谓词 $Condition(\vec{U_i})$)也可以附加在 $Rule_1$ 的规则体中进行描述，此时规则描述为 $Rule_2$：*can_access*(*Who*,*Op*,*Obj*) :-*who_attr*(*Who*,T_1,T_2,…,T_n),*attr_op_obj*(*Who*,T_1,T_2,…,T_n,*Op*,*Obj*),$Condition(\vec{U_i})$。由于条件等要素可以作为规则体中的一个部分，不影响整个规则的表达，因此本文仅讨论 $Rule_1$。

(4)提问

简而言之，访问控制请求就是用户向系统发出对某个客体执行某个操作的请求，可以使用逻辑提问“? -*can_access*(*Who*,*Op*,*Obj*)”，其中 *Who*,*Op*,*Obj* 为个体常量，分别指具体的用户、操作和客体。其表示的意思是用户 *Who* 提问他是否可以对客体 *Obj* 执行操作 *Op*。

为了突出一般性，按照访问控制的常见应用场景，我们可以将逻辑提问“? -*can_access* (*Who*,*Op*,*obj*)”扩展为“? -*can_access* (*Who*,*Op*,*Y*)”，其中 *Y* 为变量。其含义为：用户 *Who* 提问其执行操作 *Op* 时能够访问的客体有哪些。此提问返回的结果为变量 *Y* 的赋值集合，集合可为空。这种扩展符合常规，即一般情况下，需要在用户选择某个操作后，系统才能够根据访问控制策略主动提供能够访问的对象集合。

2.2　算法描述

表 1　RUAC 算法具体过程

输入：提问?-can_access (u,x_1,Y),u,x_1 为常量，Y 为变量；
　　规则 can_access(Who,Op,Obj) :-who_attr(Who,T_1,T_2,…,T_n),
　attr_op_obj(Who,T_1,T_2,…,T_n,Op,Obj)；
　　用户描述事实集 UFactSet；
　　权限约束事实集 PFactSet。
输出：变量的赋值集合 resultSet。
(1)令 resultSet＝Ø；
(2)令 L_1 为提问?-can_access (u,x_1,Y)中的文字 can_access (u,x_1,Y)，L_2 为规则的头部 can_access(Who,Op,Obj)；
　θ_0＝Unify(P,Q)＝{Who/u,Op/x_1,Obj/Y}；
(3)使用 θ_0 替换规则体中的变量，得到文字 L_3：who_attr (u,T_1,T_2,…,T_n)，和文字 L_4：attr_op_obj (u,T_1,T_2,…,T_n,x_1,Y)；
(4) for (UFactSet 中的每个事实 ufact){
　　θ_1＝Unify(L_3,ufact)＝{ T_1/t_1,T_2/t_2,…..,T_n/t_n}；
　　if (θ_1! ＝Ø){
　　使用 θ_1 替换 L_4 中的变量，得 L_5：attr_op_obj (u,t_1,t_2,…..,t_n,x_1,Y)；
　　for (PFactSet 中的每个事实 pfact){
　　θ_2＝Unify(L_5,pfact)；
　　if (θ_2!＝Ø){
　　把 θ_2 中 Y 对应的替换项添加至 resultSet；
　　}}}}
　return resultSet；

在将访问控制请求转换为逻辑提问描述后，本文提供一

种基于逻辑中的反向推理和合一的逻辑提问求解算法(RUAC),其具体过程如表 1 所列。其中提问"? -*can_access*(*Who*,*Op*,*Y*)"回答的结果是提问中变量的赋值列表 *resultSet*;对于提问"? -*can_access*(*Who*,*Op*,*Obj*)",可以首先转换为"? -*can_access*(*Who*,*Op*,*Y*)",在返回的结果 *resultSet* 中查看是否包含 *Obj*。如果 *Obj* 包含在 *resultSet* 中,则访问控制请求结果为"Permit",否则回答"Deny"。

其中,算法中用到的子算法 Unify 描述如表 2 所列,与经典的合一算法[5]的区别是:本文的 Unify 算法处理的两个文字中相同位置的项均为变量。其中,如果返回的合一替换 θ 为 $\varnothing$,则表示输入的文字 L_1 和 L_2 不可合一,否则表示 L_1 和 L_2 可合一。

表 2 Unify 算法描述

```
输入:文字 L1、L2,且 L1=P(T1,…,Tk),L2=Q(V1,…,Vm),其中,k≥1,
    m≥1。
输出:合一替换 θ。
(1)if (P != Q or k != m) 返回 Ø;
(2)C={x|x={ Ti,Vi}},其中 Ti 为 L1 中的项,Vi 为 L2 中的项;
(3)for (i=1 to k) {
      令 E(Ti) 和 E(Vi) 为 C 中包含 Ti和 Vi 的集合;
      if (Ti,Vi 为常量且 Ti!= Vi) 返回 Ø;
   if (Ti!= Vi) 合并 C 中的 E(Ti)和 E(Vi);}
(4)θ=Ø;
(5)for (C 中的每个集合 S) {
      if (S 的元素个数>1) {
        if (S 包含常量 X) { /* S 最多包含一个常量 */
          for (S 中的每个变量 Q )
            增加 Q/X 到 θ;
        else { X 为 S 中的某一个变量;
              for (Q != X in S)
              增加 Q/X 到 θ;
        }}}
(6)if (θ==Ø) 返回 Ø;
   else { for (i=1 to k) {
        if(Ti 为变量){
            for (j=1 to n) /* n 为集合 θ 的大小 */
              if (Qj==Ti 且 Qj 为常量) 返回 Ø;} } }
(7) 返回 θ;
```

3 实例及分析

为了说明本文算法对处理主体、客体具有包含关系的访问规则的描述及实施的有效性,我们选用行政审批中订购单审批流程作为实验场景。

在订购单审批流程中,有 3 张订购单(ID 分别为 1001、1002、1003)处于待处理状态。其中 ID 为 1001 的订购单,在实际应用中要求只允许"采购部的 2 级普通员工"查看,而其它的"采购部的普通员工"不能查看。按照本文算法,在制定订购单审批流程中"待处理"状态下业务流程访问控制规则时,使用事实 $fact_1$:*attr_op_obj*(*Who*,采购部,T_2,普通员工,T_4,…,T_8,*Form*)表示允许采购部的普通员工处理的所有表单,其中 T_2 表示员工级别,T_4,…,T_8 暂时作为保留项。

在系统运行过程中,部长审批完此表单后,把 $fact_1$ 中的 T_2 赋为 2 级,*Form* 赋为 ID 为 1001 的订购单,也即事实派生,得到新的事实 $fact_2$:*attr_op_obj*(*Who*,采购部,2 级,普通员工,T_4,…,T_8,待处理,1001)。同理,对于 ID 为 1002 和 1003 的订购单,可用事实:*attr_op_obj*(*Who*,采购部,T_2,普通员工,T_4,…,T_8,待处理,1002)和 *attr_op_obj*(*Who*,采购部,T_2,普通员工,T_4,…,T_8,待处理,1003)描述允许采购部的普通员工访问它们,如图 4 所示。

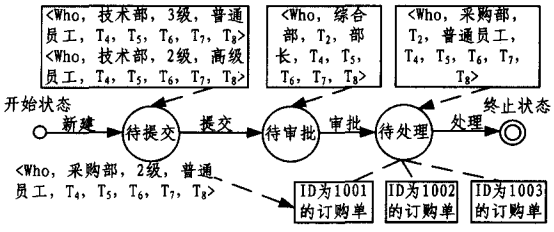

图 4 一个临时授权实例

现有用户 u_1,其所在部门是采购部,岗位级别为 2 级,职位是普通员工,可用事实 *who_attr*(u_1,采购部,2 级,普通员工,T_4,…,T_8)描述 u_1。另一用户 u_2,其所在部门是采购部,岗位级别为 3 级,职位是普通员工,可用事实 *who_attr*(u_2,采购部,3 级,普通员工,T_4,…,T_8)描述 u_2。现在,用户 u_1 和用户 u_2 都询问其可以处理的订购单,分别发起提问?-*can_access*(u_1,处理,*X*)和?-*can_access*(u_2,处理,*X*)。利用 RUAC 算法回答用户 u_1 的提问,如图 5 所示。从图 5 可知,用户 u_1 可以访问这 ID 为 1001、1002、1003 的订购单,符合访问控制规则。

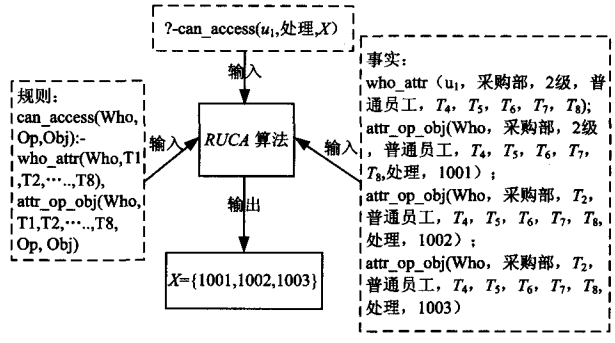

图 5 使用 RUAC 算法回答 u_1 的访问控制请求过程及结果

用 RUAC 算法回答用户 u_2 的提问,输入及输出结果如图 6 所示。由图 6 可知,用户 u_2 只可以访问 ID 为 1002 和 ID 为 1003 的订购单,而不可以访问 ID 为 1001 的订购单,符合实际要求,实现了针对具体任务的临时授权。

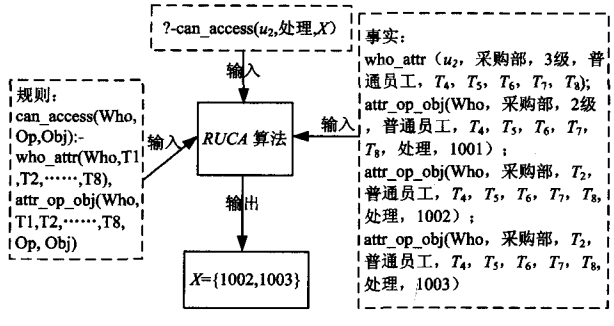

图 6 使用 RUAC 算法回答 u_2 的访问控制请求过程及结果

从上述实例可以看出,本文提出的算法能够使用事实派生方式描述访问规则间的主体、客体具有包含关系的一类访问规则,通过判断事实间是否可合一及归结原理得到访问请求的结果。与现有方法相比,本文提供的方式描述简洁、适应性强,可以表达较为灵活的访问控制规则。

结束语 系统中两个访问控制规则间对应主体、客体具有包含关系的描述是一种较为复杂的问题,很容易造成访问控制规则间存在冲突,致使相关资源保护失败。本文提供一

种基于逻辑中合一思想的算法，使用非 ground 事实（事实中含有变量）对主体和资源进行描述，将访问控制请求转换为逻辑提问，将逻辑回答转换为相应的访问控制请求应答。算法中的非 ground 事实中的变量例化可以表示主体（或客体）间的包含关系，减少了访问控制规则的数量，提高了访问控制的灵活性。进一步的研究将结合基于属性的访问控制，提出模式包含（subsume）、逐步例化的权限控制机制，增强访问控制策略的语义表达能力及动态性。

参 考 文 献

[1] 李晓峰，冯登国，陈朝武，等. 基于属性的访问控制模型[J]. 通信学报，2008，4(3)：90-98

[2] Park J, Sandhu R. The UCON$_{ABC}$ Usage Control Model [J]. ACM Transactions on Information and System Security, 2004, 7(1): 128-174

[3] Sandhu R, Coyne E J. Role based access control models [J]. IEEE Computer, 1996, 29(2): 38-47

[4] http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[5] http://cs.nyu.edu/faculty/davise/ai/datalog.html

[6] 马少平，朱小燕. 人工智能[M]. 北京：清华大学出版社，2004：103-106

[7] 陆钟万. 面向计算机科学的数理逻辑[M]. 北京：科学出版社，2000：73-114

（上接第 102 页）

然块 B_i 中元素的分布是均匀的，即各个元素取值的概率均为 $1/|B_i|$，因此根据式(2)，可以得到 $H(U|V_e=B_i)=\log_2 n_i$。再根据式(3)，可以得到 $H(U|V_e)=\sum_{i=1}^{l}\frac{n_i}{n}H(U|V_e=B_i=\frac{1}{n}\sum_{i=1}^{l}n_i\log_2 n_i)$。

至此，得到了系统面对旁路模板攻击的安全性定量度量方式。

3.3 实例分析

本节中，通过对二进制模幂算法的计时攻击的例子说明形式化分析方法的作用。首先对二进制模幂算法的计时攻击进行定性分析。对密钥长度为 8bits 的二进制模幂实现 RSA 加密算法原型并进行计时分析，采集 10000 个随机密钥对相同明文进行二进制模幂运算的时间（以 CPU 时钟周期数为计时单位），得到密钥不同的汉明重量（Hamming Weight，HW，即密钥二进制比特中 1 的个数）对应的时间关系，如图 2 所示。

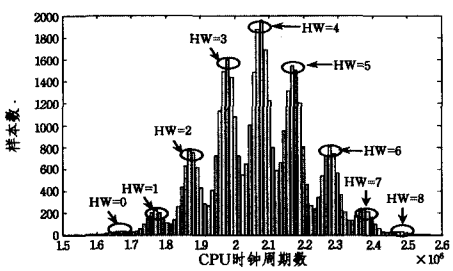

图 2 RSA 二进制模幂运算时间分布与密钥汉明重量关系

实验表明，时间旁路的量值与密钥汉明重量成正比，即 $P_m=\lambda\|D\|+\mu$，其中 P_m 表示测得的时间值，$\|D\|$ 表示数据 D 的汉明重量，即 D 的二进制表示中为 1 的位数，λ 是一个和系统相关的常数，μ 是随机噪声。这样攻击者对于计时观测实验反推密码系统某个时刻处理的秘密信息，只能得到汉明重量相等的等价类。用 $\Psi=\{(a,b)\in\Sigma_K\times\Sigma_K\mid\|a\|=\|b\|\}$ 表示输入秘密信息的汉明重量等价关系，则系统面对计时攻击时满足 Ψ_I/Ψ_O 安全性，即汉明重量等价安全性，而不满足 3.1 节定义的 AI 安全性。

对于定量分析方法，理论上可以认为计时攻击实验将二进制模幂算法的计时攻击的秘密信息输入（8bit）分成了 9 个不同的汉明重量等价类，各汉明重量等价类中元素的个数为 $C_8^i(0\leqslant i\leqslant 8)$，则按照式(4)，二进制模幂算法的计时攻击面对计时模板攻击后的剩余不确定度为

$$H(U|V_e)=\frac{1}{256}\sum_{i=0}^{9}C_8^i\log_2 C_8^i \tag{5}$$

结束语 采用等价关系和等价类划分的方法对旁路攻击中计时攻击进行形式化定性分析，结合信息熵度量方式对计时攻击者能力进行定量评价，体现了形式化分析方法在分析计时攻击敌手能力中的效果。对面向 RSA 二进制模幂运算进行计时攻击的形式化分析实例表明，形式化分析旁路攻击过程的手段使得攻击过程更加直观、确切。因此，将形式化分析方法应用到分析其它的旁路攻击方法，是值得进一步研究的。

参 考 文 献

[1] 邓高明. 基于 Cache 时间特性的密码旁路分析技术研究[D]. 石家庄：军械工程学院，2008

[2] 邓高明，赵强，张鹏，等. 针对密码芯片的电磁频域模板分析攻击[J]. 计算机学报，2009，32(4)：602-610

[3] Standaert F X, Malkin T G, Yung M. A unified framework for the analysis of side-channel key recovery attacks [S]. Version 2.1, February 2008

[4] Micali S, Reyzin L. Physically observable cryptography (extended abstract) [C] // Proceedings of the TCC 2004. LNCS 2951, Springer-Verlag, 2004: 278-296

[5] Kocher P. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems [C] // Advances in Cryptology-CRYPTO'96. LNCS, vol 1109. Springer, 1996: 104-113

[6] Boneh D. Twenty Years of Attacks on the RSA Cryptosystem [M]. Notices of the American Mathematical Society, February 1999

[7] 陈财森. RSA 公钥密码算法的计时攻击研究[D]. 石家庄：军械工程学院，2008

[8] Cachin C. Entropy Measures and Unconditional Security in Cryptography [EB/OL]. ETH Zürich, http://blog.csdn.net/fcxiao/archive/2007/09/15/1786122.aspx, 1997