Software Engineering and Applications 软件工程与应用, 2014, 3, 161-171
Published Online December 2014 in Hans. http://www.hanspub.org/journal/sea
http://dx.doi.org/10.12677/sea.2014.36019

Hans汉斯

Research of Conflict Detection in Access Control on Information Systems of the Sapiential City

Shengya Han, Guangwei Zhang

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing
Email: shengyah@163.com

Received: Oct. 25th, 2014; revised: Nov. 28th, 2014; accepted: Dec. 4th, 2014

Abstract

In the circumstance of the sapiential city, resource sharing and collaboration are necessary in information systems. The traditional access control technology is faced with new demands and challenges. In this paper, we extend the access control model based on RBAC and lead the element of "application" into this model. Different applications include different roles and permissions so that they can grant a person different roles coming from different applications for the implementations of unified authorizations. Model constraints and parameter configurations are raised in this paper for conflict detection. The detection avoids conflicts effectively and ensures correct authorizations that include basic operations of tree elements, the relationship of roles and permissions and the authorization rules.

Keywords

Sapiential City, Access Control, Conflict Detection

面向智慧城市信息系统的访问控制冲突检测研究

韩圣亚，张光卫

北京邮电大学网络与交换国家重点实验室，北京
Email: shengyah@163.com

收稿日期：2014年10月25日；修回日期：2014年11月28日；录用日期：2014年12月4日

摘　要

在智慧城市的背景下，信息系统之间急需资源共享和协同运作，传统的访问控制技术面临着新的需求和挑战。本文基于RBAC模型进行扩展，引入“应用”这一元素，不同的应用包含不同的角色和权限，通过给人员授予属于不同应用的角色，实现统一授权。同时对新的模型提出了模型约束和模型参数配置以及相应的冲突检测策略，通过检测树级元素基本操作、角色关联权限和授权规则，有效的避免和检测冲突，保证授权的正确性。

关键词

智慧城市，访问控制，冲突检测

1. 引言

智慧城市(Sapiential City)，是把新一代信息技术充分运用在城市的各行各业之中的基于知识社会下一代创新(创新 2.0)的城市信息化高级形态，实现信息化、工业化与城镇化深度融合，有助于缓解“大城市病”，提高城镇化质量，实现精细化和动态管理。智慧城市环境下的信息系统的界定是全面物联、充分整合、激励创新、协同运作等四方面。即智能传感设备将城市公共设施物联成网，物联网与互联网系统完全对接融合，政府、企业在智慧基础设施之上进行科技和业务的创新应用，城市的各个关键系统和参与者进行和谐高效地协作[1]。

智慧城市应用打破了各行业部门间的信息壁垒，实现信息融合和数据共享。智慧城市环境下应用的界定是全面物联、充分整合、协同运作、激励创新等四方面。智慧城市下的信息系统具有以下特点：

- 信息共享。智慧城市的建设不等同于信息化建设，应用系统不是自行其是，而是互联互通。应用系统之间开放数据接口，避免形成大量的信息化孤岛。通过应用系统之间的信息共享，深度挖掘应用自身价值，让应用系统更好地服务于智慧城市建设。
- 信息平台化。将各个应用系统的数据进行信息融合，建立智慧城市公共服务平台，通过大数据处理技术的分析、挖掘、应用、管理，从来自不同应用系统的海量、复杂、实时的信息中发现有用信息、提升应用，真正释放智慧的价值和能量。

作为智慧城市应用系统和公共服务平台的入口，访问控制起着举足轻重的作用。

访问控制(Access Control)防止对任何资源进行未授权的访问，从而使计算机系统在合法的范围内使用。意指，按用户身份及其所归属的某项定义组来限制用户对某些信息项的访问，或限制对某些控制功能的使用的一种技术。在传统的软件系统中，访问控制通最常于用户登录，以验证用户的正确性。

基于角色的访问控制(Role-Based Access Control, RBAC)作为传统访问控制(自主访问[2]，强制访问[3])的有前景的代替受到广泛的关注。在 RBAC 中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限，这就极大地简化了权限的管理。但是，在智慧城市下的多应用之间协同协作的要求下，角色众多，对于角色的管理成为 RBAC 的瓶颈。

智慧城市下，核心是应用，本文对传统的 RBAC 进行了扩展，加入了应用的概念，将角色进行有效的分类，方便了统一授权，对智慧城市下的信息系统协同工作提供了保证。

智慧城市下的信息系统对安全有着更高的要求，这就需要更复杂的冲突检测策略。文献[4]对策略冲突进行了分类，并提出了一种基于着色 Petri 网的 RBAC 策略冲突检测方法，将 RBAC 模型转化为着色 Petri 网模型，利用冲突陷阱进行检测，进而设计而实现了 RBAC 策略冲突检测原型系统，说明了模型的可终结性。文献[5]提出了一种基于语义的冲突检测方法，利用描述逻辑作为逻辑框架构造知识库，对 RBAC 模型及其中的冲突关系进行了形式化的表示和推理，通过检测角色分配和权限授予过程中出现的用户角色冲突、角色权限冲突以及用户权限直接授予冲突，有效的检测冲突。文献[6]针对 RBAC 模型在实施职责分离、最小特权等安全原则时引起的冲突问题，形式化定义 5 种 RBAC 策略冲突类型，分析策略冲突产生的原因，提出一种完整的策略冲突检测算法并进行仿真测试。

本文基于以上研究的基础上，对扩展的 RBAC 模型提出了详细的模型约束和参数，对冲突检测策略进行了详细研究，提出了一种解决办法，并用伪代码进行了形式化表示。

2. 基于智慧城市扩展的 RBAC

智慧城市下，角色众多，为了方便角色的管理和信息系统统一授权和协同工作，在文献[7]-[14]的基础上，对 RBAC 模型进行扩展(见图 1)，引入应用的概念，“应用”元素与软件系统中应用的概念相对应，一般情况下，一个应用解决一个相对独立组织机构的需求。

模型分为两部分：人员模块和应用模块。将人员和角色彻底分离。人员模块包含人员信息，以及人员的归属机构。应用模块对应不同的信息系统，不同的信息系统有不同种类的角色和权限。在智慧城市背景下，资源的种类繁多，不同的应用包含不同的角色和权限。通过给人员授予属于不同应用的角色，实现统一授权。

2.1. 基本元素

扩展的 RBAC 基本元素如下：

1) $\text{Organization} = \{o1, o2, \cdots, on\}$ 是所有机构的集合。在社会生活中，人们为实现某种职能所建立的、由人财物和信息等若干因素有序地联结起来的、相对稳定的社会实体单位的抽象，通常指机关、团体或其他工作单位及其内部组织。

2) $\text{Person} = \{p1, p2, \cdots, pn\}$ 是所有人员的集合。机构内的实体人员及类似实体的抽象，例如：张三。人员属于机构。

3) $\text{Appliaction} = \{app1, app2, \cdots, appn\}$ 是所有应用的集合。应用指得是不同的系统，在智慧城市下的信息系统中，将多个系统进行集成，进行统一授权，将不同的应用中的权限授予不同的操作者。

4) $\text{Role} = \{r1, r2, \cdots, rn\}$ 是所有角色的集合。指在处理特定业务时设定的具有特定工作范围或工作职责，用于解决特定业务问题的实体抽象，例如：系统管理员。角色属于应用，角色包括子角色。

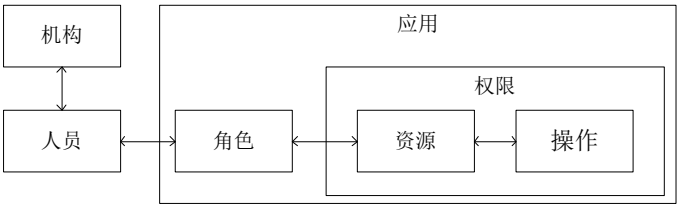

Figure 1. The extended model based on RBAC
图 1. 基于 RBAC 扩展的模型

5) $\text{Resource} = \{\text{rs1}, \text{rs2}, \cdots, \text{rsn}\}$ 是所有资源的集合。可称为资源或对象，是授权客体。资源属于具体的应用，有了应用才有资源。任何需要权限保护的元素都可以作为资源，比如一个页面、一个图片、一个菜单、一个文件或一个按钮等。资源属于应用，资源包含子资源。

6) $\text{Operation} = \{\text{opr1}, \text{opr2}, \cdots, \text{oprn}\}$ 是所有操作的集合。又可称为功能(Function)，权限类型。是访问控制可以执行的最小功能项，被操作者调用或执行。操作和资源的类型有关，资源类型决定相应的操作。

7) $\text{Privilege} \subseteq \text{Operation} \times \text{Resource}$ 是所有权限的集合。权限就是建立操作和资源之间的联系。

8) $\text{AssociateRole} \in \text{Role} \times \text{Privilege}$ 是所有关联角色的集合。关联角色就是建立角色和权限之间的关系。

9) $\text{Grant} \in \text{Person} \times \text{Role}$ 是所有授权的集合。授权就是建立人员和角色之间的关系。

2.2. 模型关系

模型分为两部分：人员模块和应用模块。模型关系图如图 2 所示。人员模块包括机构和人员，机构包含多个子机构，机构可以包含多个人员。应用模块包括应用、资源、操作和角色。应用包含多个资源，资源包含多个子资源，每个资源都有唯一的操作类型，每个资源类型对应多个操作，资源和操作的集合构成权限。应用包含多个角色，角色和权限是多对多的关系。人员和角色是多对多的关系。一个人可以有来自不同应用的很多角色，每个角色可以授予不同的人。

3. 模型约束与参数

本文在文献[4]-[6]研究的基础上，面向智慧城市下的信息系统，为了避免和检测冲突的发生，对模型进行了详细的约束并对模型参数进行配置。

模型约束是指，基于智慧城市扩展的 RBAC 模型，系统静态配置的规则，这些规则和模型关系紧密。

模型参数是指，管理员根据具体的系统环境，为了增加系统的稳定性和健壮性，加快数据交换的能力，自己定义和配置的规则。如果不进行设置，对系统也不会产生严重的影响。

3.1. 模型约束

由于模型的复杂性，对模型提出了三类约束：

1) 对树级元素基本操作的约束

基于 RBAC 扩展的模型中包含了三种树级元素：机构树、角色树和资源树。树级元素的基本操作是指树级节点增加子节点、修改节点和删除节点。

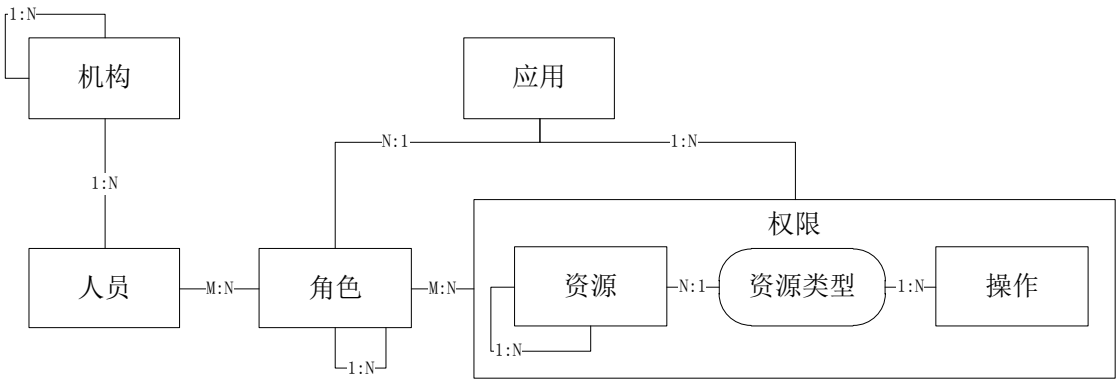

Figure 2. The relationship of new model
图 2. 模型关系

2) 对角色关联权限的约束

角色关联权限包含：角色增加权限、角色修改权限和角色删除权限。

3) 对授权的约束

授权是指给人员授予角色，人员就具备了角色所拥有的权限。由于角色的复杂性，导致了授权过程中可能会发生冲突。

下面对三种约束进行具体的描述：

1) 对树级元素增加子节点的约束。

a) 对于机构和资源树，节点只有基本属性，相互独立，增加子节点没有约束。

b) 对于角色树，角色除了基本属性，还与权限相关联，子角色必须继承父角色的权限。

图 3 表示的是角色继承，子角色 1 和子角色 2 自动继承了父角色的权限 a。

2) 对树级元素删除节点的约束。树级元素删除节点包括两种约束：级联删除和非级联删除。

a) 级联删除：

i) 删除该节点和该节点包含的所有的子节点。

ii) 删除与这些节点关联的关系。

b) 非级联删除：只允许删除叶子节点。

i) 对于资源，如果存在一个角色与该资源相关联，那么不能删除。

ii) 对于角色，如果存在一个人员与该角色相关联，那么不能删除。

iii) 对于人员，删除和人员关联的关系。

3) 对树级元素修改节点的约束。在修改节点的父节点属性时，可能会发生树级环路。为了避免发生树级环路，必须进行相应的检测。图 4 是一个发生环路的机构树。

4) 对角色关联权限的约束。角色关联权限包括角色增加权限、角色修改权限和角色删除权限。由于子角色继承了父角色的权限，所以只允许角色树的叶子节点增加、修改和删除权限。

a) 角色删除权限时，只允许删除其新增的权限，不允许删除继承的权限。

b) 角色修改权限时，只允许修改其新增的权限中资源对应的操作。

c) 角色增加权限时，有 3 个约束条件：

i) 禁止重复权限。避免新增的权限存在于角色已有权限中。

ii) 禁止互斥操作。权限是(资源，操作)的集合。如果要增加的权限中的资源已经存在于该角色中，那么对应的操作不能是互斥操作。图 5 表示角色 A 拥有权限(图片 A，显示)，因为图片的显示的屏蔽是互斥操作，所以不能将(图片 A，屏蔽)与角色 A 进行关联。

iii) 禁止越级关联。如果所增加的资源不存在于该角色中，则该资源要么是根资源；要么是该角色已有资源的子资源。图 6 表示角色 A 拥有(菜单 1，显示)权限，当关联(按钮 2，显示)时，发生了越级关联。因为角色 A 没有关联菜单 2 相关的权限，所以不能越过菜单 2，关联按钮 2。

5) 对授权的约束。授权是指给人员授予角色，人员就具备了角色所拥有的权限。

a) 来自不同应用的角色之间相互独立。

b) 来自相同应用的角色之间可能会有联系。

i) 存在重复权限。

ii) 存在互斥权限——互斥角色。

3.2. 模型参数

模型参数是模型的变量，为了方便系统管理员的管理，对模型的变量进行自定义配置。良好的模型

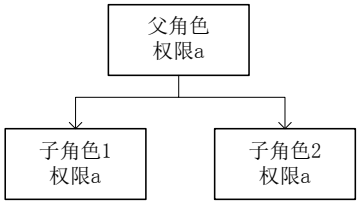

Figure 3. Role blocks
图 3. 角色继承

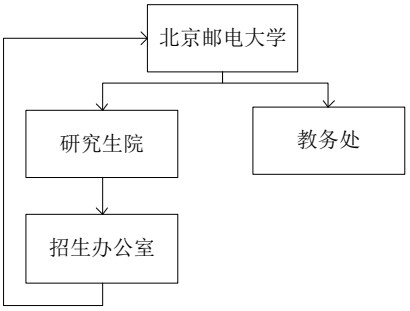

Figure 4. The loop in the tree made of organizations
图 4. 机构树环路

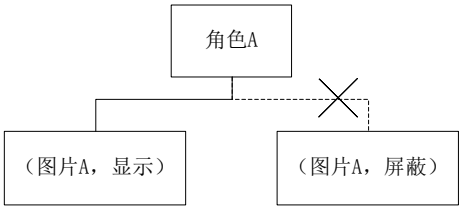

Figure 5. Mutually exclusive operation
图 5. 互斥操作

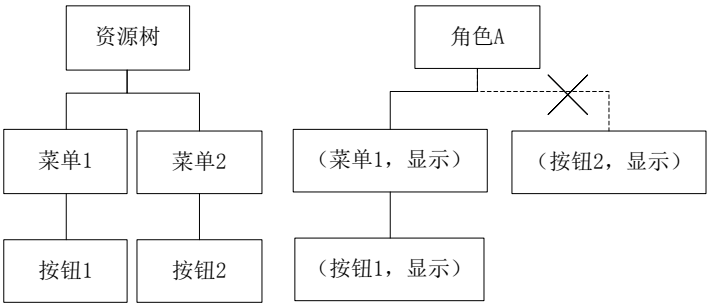

Figure 6. Leapfrog association
图 6. 越级关联

参数配置有利于系统的维护，增加系统的稳定性和健壮性，加快数据交换的能力。模型参数一般存于配置文件，对于模型配置，总结了以下几种类型：

- 树的最大深度。树结构最常用的是递归操作，如果树的深度不可控制，那么递归操作的效率急剧下降，影响系统性能。如果一个操作使树的最大深度超出这个值，那么便回滚这个操作，禁止这个操作。
- 树的祖先数量是否唯一，如果不唯一，那么设置其最大数量。设置这个规则的原因来类似于 Java 中的 Object 类，是否允许所有的资源树和角色树都从一个祖先继承，是要根据具体的应用来考虑的。

- 应用所拥有的最大资源数量和最大角色数量。数量的限制有利于系统的精简和维护。如果一个操作使应用下的资源数量和角色数量超出这个值，那么便回滚这个操作，禁止这个操作。
- 资源类型所拥有的最大操作数量。每个资源类型所拥有的操作数量是有限的，但也不能随便的创建，无意义的重复操作不利于系统的维护。如果创建的操作数量太多，那么禁止创建，不能超出最大值。
- 角色所拥有的最大权限数量。每层角色所拥有的最大权限数量是这个值减去角色树的最大深度，再加上角色所在角色树的层数。这样设置的目的是为了限制角色相关联的权限，并且让每一层子角色都可以最少增加一个权限。
- 授予用户的角色是否唯一，如果不唯一，那么设置其最大数量。根据具体的系统环境，有些应用简单，而有些应用很复杂，用户所授予的每个应用的角色是否唯一来决定这个规则的设置。

4. 冲突检测策略

4.1. 冲突检测

如果按照模型约束和参数的规则进行操作，那么系统不会发生故障，也就没有冲突可言。如果不按照此规则进行操作，那么冲突就发生了，比如将互斥角色授予同一个用户。如果没有相应的检测机制，那么系统就会崩溃。

冲突检测就是进行操作之前，根据一套检测机制来检测该操作是否遵循模型约束和参数配置。

1) 树级元素的冲突检测策略：

a) 创建机构：如果数量超出了用户自定义值，那么禁止创建。如果新建的深度超出用户自定义值，那么禁止创建。

输入：currentNoOfOrg，parentOrg//当前机构数量，父机构信息

输出：result//若发生冲突，返回 false，禁止创建。

```
Begin
Result=true;
Level=0;
If(currentNoOfOrg>=maxNoOfOrg)
    result=false;
p=parentOrg;
while(p!=null)
{
    Level++;
    P=p.parent;
}
If(level>=maxHeightOfOrg)
    result=false;
if(result)
    currentNoOfOrg++;
Return result;
End
```

b) 修改机构：如果修改机构的所属父机构，那么从新的父节点向上遍历，如果遍历到该节点，那么发生环路冲突，禁止修改。

输入：org//要修改的机构信息
输出：result//若发生冲突，返回 false，禁止修改。
Begin
Result=true;
o=org;
p=org.parent;
While(p!=null)
    If(p.id==o.id)
{
    Result=false;
    Return result;
}
Return result;
End
c) 删除机构：如果违反了删除规则，那么禁止删除。
输入：org //需删除的机构信息
输出：result//若发生冲突，返回 false，禁止删除。
Begin
Result=true;
For each o ∈ Organization
    If(o.parent==org)
        Result=false;
Return result;
End
角色树和资源树类似。但是，在删除资源时，还需考虑资源和角色关联关系。在删除角色时，还需考虑角色和人员、资源的关系。
2) 角色关联权限的冲突检测策略：
d) 角色增加权限：
i) 如果角色不是叶子节点，禁止增加。
ii) 如果角色所拥有的权限数量超出用户自定义值，禁止增加权限。
iii) 如果出现重复权限，禁止增加权限。
iv) 如果出现互斥操作，禁止此增加权限。
v) 如果发生了越级关联，禁止增加权限。
输入：role,permission //角色，权限
输出：result//若发生冲突，返回 false，禁止角色增加权限。
Begin
Result=true;
Flag=false;
For each r ∈ Role
    If(r.parent==role)

```
        Return false;
        If(role.noOfPermission>=role_MaxOfPermission)
            Return false;
For each rp ∈ role.permission
    If(rp==permission)
        Return false;
    If(rp.resource==permission.resource&&mutex(rp.operate,permission.operate))
        Return false;
For each rp ∈ role.permission
    If(permission.resource.parent==rp)
        Flag=true;
If(!flag)
    Result=false;
Return result;
End
```

e) 角色修改权限：

i) 如果角色不是叶子节点，禁止修改。

ii) 只允许修改资源对应的操作。

```
输入：role,permission //角色，权限
输出：result//若发生冲突，返回 false，禁止角色修改权限。
Begin
Result=true;
Flag=false;
For each r ∈ Role
    If(r.parent==role)
        Return false;
For each rp ∈ role.permission
    If(rp.resource==permission.resource)
        Flag=true;
If(!flag)
Result=false;
        Return result;
        End
```

f) 角色删除权限：如果角色不是叶子节点，禁止删除。

```
输入：role,permission //角色，权限
输出：result//若发生冲突，返回 false，禁止角色删除权限。
Begin
Result=true;
For each r ∈ Role
    If(r.parent==role)
```

```
        Return false;
    Return result;
    End
```

3) 创建操作的冲突检测策略：如果一个资源类型所对应的操作数量超出了用户自定义的数量上限，那么禁止创建操作。

输入：typeOfResource//资源类型

输出：result//若发生冲突，返回 false，禁止角色删除权限。

```
Begin
Result=true;
sum=0;
For each operate  ∈ Operation
    If(operate.typeOfRs== typeOfResource)
        Sum++;
If(sum>=maxOfResource)
    Result=false;
Return result;
End
```

4) 授权的冲突检测策略：

a) 如果用户拥有的角色超出了用户自定义中用户所拥有的角色最大值，那么禁止授权。

b) 如果出现互斥角色，那么禁止授权

输入：user,role//用户，角色

输出：result//若发生冲突，返回 false，禁止授权。

```
Begin
Result=true;
If(user.noOfRole>=maxOfRole)
    Return false;
For each r  ∈  user.role
    If(mutex(r,role))
        Return false;
Return result;
End
```

4.2. 冲突检测策略比较

与文献[4]-[6]所提出的冲突检测相比，本文提出的冲突检测策略有以下几个特点：

1) 面向智慧城市信息系统，将角色划分到应用下，减少对角色相关冲突的检测，增加了检测效率。传统的检测技术对于角色的检测都是一概而论，当角色数量增大时，对于角色相关的冲突检测十分耗时。

2) 对冲突的检测进行分类，减少检测的繁琐性。传统的检测技术往往采用一种通用的办法检测所有的冲突，本文将检测具体划分为树级元素的检测、角色关联权限的检测、操作的检测和授权的检测，每种冲突的检测具有针对性。

3) 增加了对模型参数的检测。对于一个具体的信息系统，适度的模型参数有利于系统的健壮性和稳

定性，本文对于模型的参数进行了约定，并进行了检测。

参考文献 (References)

[1] 《智慧的城市在中国》白皮书. IBM. http://www.ibm.com/smarterplanet/cn/zh/smarter_cities/overview/

[2] Adleman, L.M. (1994) Molecular computation of solutions to combination problems. *Science*, 266, 1021-1023.

[3] Ding, Y.S., Ron, L.H. and Shao, S.H. (2001) Automatic design of Takagi-Sugeno fuzzy controllers by a new DNA-based evolutionary algorithm. *Acta Automatica Sinica*, **27**, 510-520.

[4] 罗杨, 梁晓艳, 夏春, 吕良双 (2013) 一种基于着色 Petri 网的 RBAC 策略冲突检测方法. *小型微型计算机系统*, **11**, 2487-2490.

[5] 张雷, 向宏, 胡海波 (2011) 基于语义的 RBAC 模型权限冲突检测方法. *计算机工程与应用*, **26**, 74-78.

[6] 程相然, 陈性元, 张斌, 杨艳 (2010) RBAC 策略冲突及其检测算法的研究. *计算机工程*, **18**, 135-137.

[7] Sandhu, R., Coyne, E., Feinstein, H., et al. (1996) Role-based Access Control Model. *IEEE Computer*, **2**, 38-47.

[8] ANSI, INCITS (2004) ANSI/INCITS 359-2004 information technology—Role based access control.

[9] 张晓燕, 张素伟 (2007) 基于 RBAC 的电子政务权限访问控制模块的设计与实现. *计算机工程与设计*, **3**, 680-682.

[10] 彭军 (2006) 统一资源访问控制的研究. 硕士论文, 新疆大学, 乌鲁木齐.

[11] 王娜娜 (2013) 基于 XACML 的访问控制模型的研究与评估优化. 硕士论文, 江苏大学, 镇江.

[12] 张爱娟 (2011) 基于语义的安全规则冲突检测. 2011 *年江苏省人工智能学术会议*, 徐州, 2011 年 10 月 22 日, 67-70.

[13] 韩永征 (2011) 基于约束的委托授权研究. 硕士论文, 重庆大学. 重庆.

[14] 杨化峰 (2013) 神州数码智慧城市发展战略研究. 硕士论文, 河北工业大学, 天津.

汉斯出版社为全球科研工作者搭建开放的网络学术中文交流平台。自2011年创办以来，汉斯一直保持着稳健快速发展。随着国内外知名高校学者的陆续加入，汉斯电子期刊已被450多所大中华地区高校图书馆的电子资源采用，并被中国知网全文收录，被学术界广为认同。

汉斯出版社是国内开源（Open Access）电子期刊模式的先行者，其创办的所有期刊全部开放阅读，即读者可以通过互联网免费获取期刊内容，在非商业性使用的前提下，读者不支付任何费用就可引用、复制、传播期刊的部分或全部内容。