doi:10.3969/j.issn.1000-436x.2013.z1.018

基于动态情景网关的系统协同访问控制模型

郭树行，张禹
（中央财经大学 信息学院，北京 100081）

摘 要：为适应网构环境下应用系统主客体间访问集中控制的需要，提出了一种基于动态情景网关的系统协同访问控制模型 DSGAC。首先，给出了网构应用系统中访问的动态情景要素构成，从多视角定义了情景要素；其次，提供动态情景状态机的概念，并给出了动态情景机约束下的系统协同访问网关模型 DSGAC，支持情景状态演算、规则演算；最后，实际案例应用验证表明 DSGAC 模型在应用系统的应用可行性，并归纳了其与现有访问控制模型的相对创新性。

关键词：情景网关；系统协同；访问控制；情景演算；状态机
中图分类号：TP302.1 **文献标识码**：A **文章编号**：1000-436X(2013)Z1-0142-06

Dynamic situation gateway based system cooperation access gatel model

GUO Shu-hang, ZHANG Yu
(School of Information, Central University of Finance and Economics, Beijing 100081, China)

Abstract: In order to adapt the centralized access control between the application system subject and object in the environment of network configurations, a dynamic situation based system cooperation access gatel model — DSGAC was proposed. To begin with, the constitution of dynamic situational factors of access controls in the application systems of network configurations was analyzed, defining the factor of situation from muti-perspective. Furthermore, the concept of dynamic situational finite state machine was provided and a system cooperation access gatel model supporting calculations and rules of the situation was presented under the constraint of dynamic situations. Finally, the real case application shows the feasibility and validation of the DSGAC model between the application systems. Additionally, the relative innovativeness between DSGAC model and existing access control models was summerized.

Key words: situation gatel; system cooperation; access control; situation calculation; finite state machine

1 引言

近年来，随着企业应用架构的建设，系统间安全访问问题受到广泛关注。这其中协同访问控制技术是解决安全交互的关键，访问控制机制对保护各自系统私有资源十分重要。传统的访问控制模型主要有自主访问控制模型（DAC)、强制访问控制模型(MAC)和基于角色的访问控制模型（RBAC)[1~3]、基于静态情景的访问控制模型（SAC）。但是，这4种访问控制，主要适用于单个系统自身的访问控制。目前我国大部分企业正在快速推进信息化的顶层设计，主要搞好企业的统一应用架构。应用架构指明了系统间通信交互中心访问控制需求，上述4种安全访问控制模型均存在一定的弊端，具有一定的局限性。近年来，访问控制模型研究多属于对DAC、MAC与RBAC、SAC不同程度的扩展。诸如，

收稿日期：2013-07-02
基金项目：国家自然科学基金资助项目（61103198, 61100112, 60970143）；教育部科学技术重点项目基金（109016）；教育部人文社会科学研究青年基金资助项目（11YJCZH006）；中央财经大学学科建设基金资助项目（CUFEIE201107）
Foundation Items: The National Natural Science Foundation of China (61103198, 61100112, 60970143); The Key Project of Ministry of Education(109016); MOE (ministry of education in China) Project of Humanities and Social Sciences(11YJCZH006); The Construction Fund of Subjects of Central University of Finance and Economics(CUFEIE201107)

基于时态的角色访问控制模型(GTRBAC)[4]、基于任务的访问控制模型(TBAC)[5]、面向服务的访问控制模型(SOAC)[6,7]、基于属性的访问控制模型(ABAC)[8~10]、状态机模型[11]等。但是，这些模型不能对权限动态分配、前置需求差异、访问约束与条件等情景因素统一考虑。例如基于角色的访问控制(RBAC)被认为是对传统的自主访问控制(DAC)和强制访问控制(MAC)的革新与补充，但是改进后的DAC、MAC与RBAC模型也只能实现单个系统内部安全访问规则的动态性。

研究企业应用架构中系统协同访问控制，必须考虑系统协同访问受控因素的多样性。为此，提出对权限、前置需求差异、访问约束与条件等动态情景因素进行统一建模，并结合通信网关模型提出了一种动态情景网关。动态情景网关的主要使命是根据实时、动态要求进行情景演算，分为情景构造与情景推演 2 个环节。情景构造，根据应用系统环境交互控制所需要协议规则，定义情景状态；情景推演是指利用情景状态推理，判定系统主体对系统客体的访问许可。基于动态情景网关的系统协同访问控制模型，是以动态交互环境下应用系统主客体间交互为研究对象提出的控制模型，称为 DSGAC。

2　概念定义

实际中的访问控制模型是指利用主体、客体与权力 3 个实体所构造的访问控制协议（Agreements），也即是主体访问客体时遵循的一致性约定。提出在这 3 个核心实体基础引入了第 4 个核心概念实体，即情景（Situation）。

定义 1　客体（Object）：包括实现对主体协同请求做出响应的系统对象，简称响应客体。

定义 2　主体（Subject）：包括发起对客体交互请求的系统对象，简称请求主体。

定义 3　权力（Right）：是指请求主体所拥有的对响应客体访问的能力。

定义 4　动态情景：在一定时间内影响主体访问客体的不同权限、使用约束、前置需求与使用条件的集合。通过此 4 个核心实体，表达情景访问控制协议。

定义 5　协同权限：表示客体可以被主体利用的方法，诸如结算、支付、认证、点播、查询等。

定义 6　协同约束：表示主体访问客体时应满足的外部限制要素，诸如请求主体应遵循的时空边界、网络位置等。

定义 7　协同需求：表示请求主体访问响应客体需要时请求主体事先需要满足的要求。诸如主体自身所属类型、主体自身的应用系统类型限制等。

定义 8　协同条件：表示响应客体被请求主体访问时响应客体自身应具备的前提条件。诸如状态、时间等。

情景演算主要依据是协同权限属性(permission attributes)、需求属性(requirement attributes)、约束属性(constraint attributes)和条件属性。

在 DSGAC 模型中，访问控制验证是基于属性值动态逻辑推理。DSGAC 模型访问控制的动态性在于适应这些属性值的动态变化。DSGAC 访问控制模型如下图 1 所示。

3　情景构造

情景构造指出为了控制应用系统主体对应用系统客体进行访问而进行配置管理。说明如表 1 所示。

表 1　　**情景构造操作字典**

op	操作说明
情景初始	$Init(S)\mapsto S_0$
情景更新	$Update(S_{i-1},S_i)$，其中 S_{i-1},S_i 两者权限、使用约束、前置需求与使用条件有区别
情景授权	$Deploy(Subject_x,S_i,Object_y)$
情景撤权	$Destroy(Subject_x,S_i,Object_y)$
情景代理	$Delegate(Subject_x,Subject_w),t\in(t_m,t_n)$
情景转发	$Derive(Subject_x,Subject_w),t\in(t_m,t_n)$

情景初始：$Init(S)\mapsto S_0=(S_{\text{Permission0}},S_{\text{Requirement0}},S_{\text{Constraint0}},S_{\text{Condition0}})$，是指为了访问控制的需要，构造访问控制的情景状态定义。

情景更新：$Update(S_{i-1},S_i)=(Update(S_{\text{Permission}_{i-1}})\oplus S_{\text{Requirement}_{i-1}}\oplus S_{\text{Constraint}_{i-1}}\oplus S_{\text{Condition}_{i-1}})$，其中 $S_i\neq S_{i-1}$，指为了访问控制的需要，更新访问控制的情景状态。

情景授权：$Deploy(Subject_x,S_i,Object_y)$，授权主体 x 在状态 i 下对客体 y 的访问许可。在情景授权下，有 $Subject_x\xrightarrow{S_i}Access(Object_y)\Rightarrow\text{true}$，表示主体 x 在状态 i 下对客体 y 的访问时验证为 True。

情景撤权：$Destroy(Subject_x,S_i,Object_y)$，撤销主体 x 在状态 i 下对客体 y 的访问许可。在情景撤

权后， $Subject_x \xrightarrow{S_i} Access(Object_y) \Rightarrow \text{False}$ 。表示若无代理许可，主体 x 在状态 i 下对客体 y 的访问时验证为 False。

情景代理： $Delegate(Subject_x, Subject_w)$, $t \in (t_m, t_n)$，授权主体 w 代理主体 x 的访问许可且主体 x 不再拥有相应许可，有效期为 (t_m, t_n)。在情景代理下，有 $Subject_w \xrightarrow{S_i} Access(Object_y) \Rightarrow \text{true}, t \in (t_m, t_n)$，表示主体 w 在状态 i 下对客体 y 的访问时验证为 True；但此时， $Subject_x \xrightarrow{S_i} Access(Object_y) \Rightarrow \text{false}, t \in (t_m, t_n)$。

情景转发： $Derive(Subject_x, Subject_w), t \in (t_m, t_n)$，授权主体 w 代理主体 x 的访问许可且主体 x 同时拥有相应许可，有效期为 (t_m, t_n)。在情景代理下，有 $Subject_w \xrightarrow{S_i} Access(Object_y) \Rightarrow \text{true}, t \in (t_m, t_n)$，表示主体 w 在状态 i 下对客体 y 的访问时验证为 True；但此时， $Subject_x \xrightarrow{S_i} Access(Object_y) \Rightarrow \text{true}, t \in (t_m, t_n)$。

4 情景推演

DSGAC 验证模型表示某一时刻验证应用系统主体对应用系统客体访问控制的逻辑推理，称为 DSGAC 情景演算。提出可以采取 2 种推演模式，即状态推演和规则推演。2 种推演方式并不分离开来，具有较强的关联性，具体过程如图 2 所示。

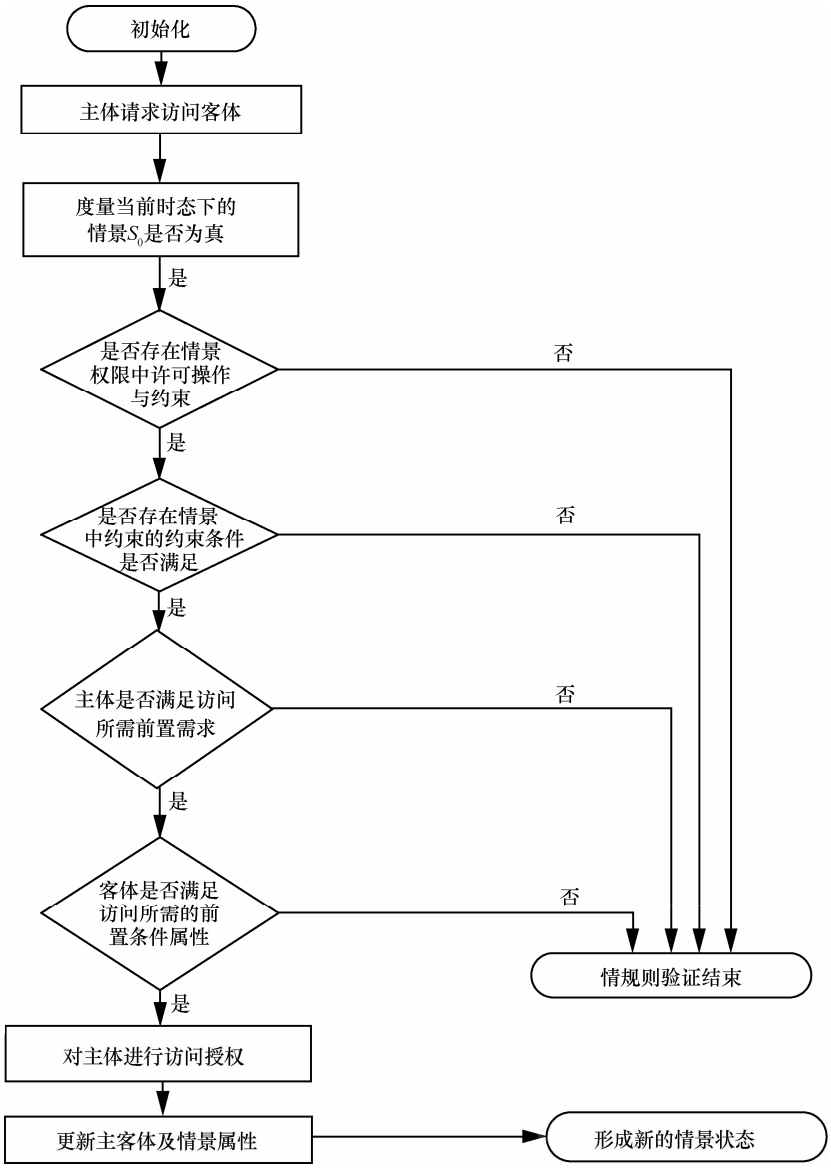

图 2 基于 DSGAC 的访问控制算法

状态推演可以用逻辑谓词描述情景状态中权限、使用约束、前置需求与使用条件内容是否组合为真；规则流则可以用逻辑规则来描述并进行演算。当存在 $S_{\text{Permission}}$ 中许可的操作、主体 $Subject_x$ 满足 $S_{\text{Requirement}}, S_{\text{Constraint}}$ 中所约定前置需求和使用约束、客体 $Object_y$ 满足 $S_{\text{Condition}}$ 中所指定的条件。

可表示为：

$$S_{\text{Permission}_i} \bigcap S_{\text{Requirement}_i} \bigcap S_{\text{Constrain t}_i} \bigcap S_{\text{Condition}_i} \to \text{true} \mapsto (S_i \to \text{True})$$

采用独立的情景状态演算，可以充分体现情景元素在访问控制的核心作用。用来表明主体访问客体时上下文环境中是否可以使待验证情景状态为真。

规则推演可以用于验证在通用环境主体、情景状态、通用环境客体 3 者之间的已构造情景的规则。当存在一种情景状态 S_i 属于已定义的情景状态空间 S，某一主体 $Subject_x$ 已授权、代理或被转让这种情景状态下对客体 $Object_y$ 的访问许可，则主体 $Subject_x$ 访问客体 $Object_y$ 为真。可以表示为

$$\exists S_i \in S, Access(Subject_x, Object_y) = \text{true} \rightleftarrows \begin{cases} Deploy(Subject_x, S_i, Object_y) \\ \cup Delegate(Subject_w, Subject_x), t \in (t_m, t_n) \\ \cup Derive(Subject_w, Subject_x), t \in (t_m, t_n) \end{cases}$$

情景演算方法则是从情景状态和情景规则 2 个方面组合演算。条件 $t \in (t_m, t_n)$ 反映对主体对客体的访问权限进行时态的控制，确保 DSGAC 模型中访问控制的实时性、准确性和高效性。

算法的形式化描述如下：

Phase 1：
//Initializing Situation
Init S_0($S_{\text{Pemission0}}$,$S_{\text{Requirement0}}$,$S_{\text{Constraint0}}$,$S_{\text{Condition0}}$)
Phase 2:
//Subject Request Sending
Send subjectRequest
Phase 3：
//Situation operating and updating
if subjectAttr($S_{\text{Pemission0}}$) then
if subjectAttr($S_{\text{Constraint0}}$) then
if subjectAttr($S_{\text{Requirement0}}$) then
Deploy(Subject,S_0,Object)
Destroy(Subject,S_0,Object)
Delegate(Subject0,Subject1)
subjectAttr ← Update(subjectAttr)
S_0 ← Update(S_0)
end if
else Deny(subjectRequest)
end if
else Deny(subjectRequest)
end if
else Deny(subjectRequest)

5　实验验证

5.1　实验设计

本文以一个实例来验证基于动态情景网关的系统协同访问控制模型（DSGAC）的系统授权和情景更新过程。为验证 DSGAC 模型的可行性，通过采取获取多个主体对客体操作的系统协同访问过程的日志记录的方法进行分析。该日志详细记录了主体的属性、权利和请求访问的客体属性以及访问过程中情景的权限、约束、条件和需求，最终还包括了访问过程中情景中主客体交互过程中产生的动作。日志记录充分反映了在 DSGAC 模型中，主客体应用系统间访问控制的详细过程及记录。

图 3 给出应用架构中系统协同访问的日志收集方式，主体系统通过发送请求产生消息流，日志服务区抓取日志记录并记录到本地的存储设备中。日志记录了不同主体对不同客体的请求以及在系统处理请求前后主客体属性以及情景状态的变化。

实验日志的数据如图 4 所示，通过统计分析可以说本文所提出的 DSGAC 模型最终在案例企业全程物流安全应用系统建设项目中得以应用。该系统实现了从仓储、运输、终端销售到企业管理等关键业务环节的系统无缝接合。实际中，提出需要达到全程安全访问可控与监控、关键环节访问确认、非法操作即时准确预警等安全生产的工作要求。结合自身业务特点，构建出了具有行业特色的能源运输的通用环境安全网关。在本实例中，主体的操作请求分为查询、修改、删除、权限代理以及权限撤销，情景状态随情景演算结果与时间同步更新。针对主体不同的请求，系统记录了情景在处理请求之前与之后的操作和状态的变化。日志充分反映了 DSGAC 模型情景推演的过程，也验证了状态推演与规则推演的结果，从侧面体现了 DSGAC 模型的实时性、灵活性和动态更新的特性。

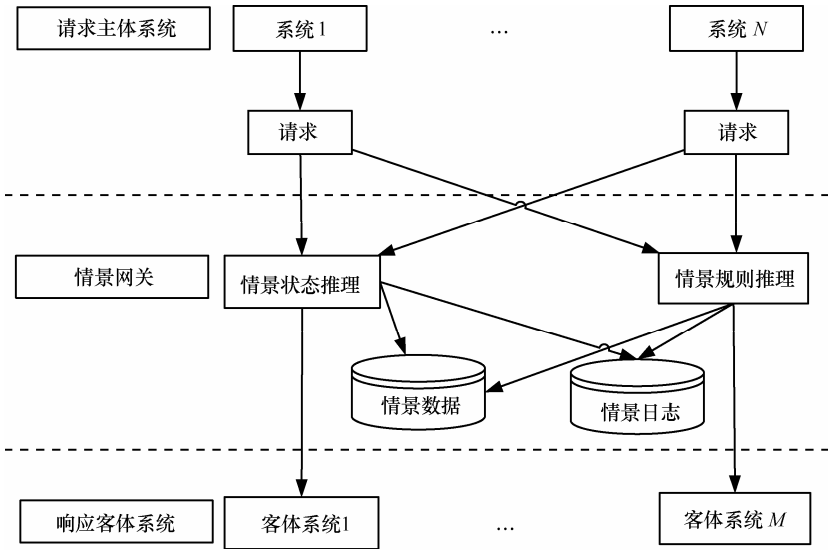

图 3　基于情景的协同访问应用架构

主体属性	客体属性	请求	情景状态	主体属性更新	客体属性更新	情景更新
Subjec2t(query,mo...	Object1(query,mod...	Query	Deploy (Subject2...	Subject2(querry,m...	Object1(querr...	Update (S0, S1)
Subject1(query,de...	Object2(query,del...	Query	Deploy (Subject1...	Subject1(query,de...	Object2(query...	Update (S1, S2)
Subject0(query,mo...	Object1(query,mod...	Modify	Deploy (Subject0...	Subject0(query,mo...	Object1(query...	Update (S2, S3)
Subject2(query,de...	Object3(query,del...	Query	Deploy (Subject2...	Subject2(query,de...	Object3(query...	Update (S3, S4)
Subject1(query,mo...	Object3(query,mod...	Modify	Deploy (Subject1...	Subject1(query,mo...	Object3(query...	Update (S4, S5)
Subject0(query,mo...	Object1(query,mod...	Delete	Deploy (Subject0...	Subject0()	Object1()	Update (S5, S6)
Subject2(query,de...	Object3(query,del...	Query	Deploy (Subject2...	Subject2(query,de...	Object3(query...	Update (S6, S7)
Subject0()	Object1()	Query	S7	Subject0()	Object1()	S7
Subject0(query,mo...	Object2(query,mod...	Modify	Deploy (Subject0...	Subject0(query,mo...	Object2(query...	Update (S7, S8)
Subject1(query,mo...	Object3(query,mod...	Query	Deploy (Subject1...	Subject1(query,mo...	Object3(query...	Update (S8, S9)
Subject2(query,mo...	Object2(query,mod...	Modify	Deploy (Subject2...	Subject2(query,mo...	Object2(query...	Update (S9, S10)
Subject0()	Object1()	Query	S10	Subject0()	Object1()	S10
Subject0(query,mo...	Object3(query,mod...	Delegate	Delegate (Subject...	Subject1(query,mo...	Object3(query...	Update (S10, S11)
Subject2(query)	Object1(query)	Modify	S11	Subject2()	Object1()	S11
Subject1(query,mo...	Object3(query,mod...	Modify	Deploy (Subject1...	Subject1(query,mo...	Object3(query...	Update (S11, S12)
Subject2(query,de...	Object3(query,del...	Modify	S12	Subject2(query,de...	Object3(query...	S12
Subject0(query,mo...	Object2(query,mod...	Query	Deploy (Subject0...	Subject0(query,mo...	Object2(query...	Update (S12, S13)
Subject0(query,mo...	Object3(query,mod...	Query	Deploy (Subject1...	Subject1(query,mo...	Object3(query...	Update (S13, S14)
Subject2(query,mo...	Object2(query,mod...	Delete	S14	Subject2(query,mo...	Object2(query...	Update (S14, S15)
Subject1(query,mo...	Object3(query,mod...	Delegate	Delegate (Subject...	Subject2(query,mo...	Object3(query...	Update (S15, S16)
Subject1(query,mo...	Object3(query,mod...	Delete	S16	Subject1()	Object3()	Update (S16, S17)
Subject2(query,mo...	Object2(query,mod...	Query	Deploy (Subject2...	Subject2(query,mo...	Object2(query...	Update (S17, S18)
Subject0(query,mo...	Object3(query,mod...	Query	S18	Subject0()	Object3()	S18
Subject0(query,mo...	Object2(query,mod...	Destroy	Destroy(Subject1,...	Subject1()	Object2()	Update (S18, S19)
Subject0()	Object1()	Query	S19	Subject0()	Object1()	S19
Subject2(query,mo...	Object2(query,mod...	Query	Deploy (Subject2...	Subject2(query,mo...	Object2(query...	Update (S19, S20)
Subject1()	Object3()	Modify	S20	Subject1()	Object3()	S20
Subject0(query,mo...	Object2(query,mod...	Destroy	Destroy(Subject3,...	Subject2()	Object2()	Update (S20, S21)
Subject0()	Object3()	Query	S21	Subject0()	Object3()	S21

图 4　情景日志数据示例

5.2　实验结论

DSGAC 模型把主客体应用系统之间协同访问引入了情景感知安全层，统一访问控制处理，提高了通用环境中应用系统之间的协同访问控制。DAC、MAC、SAC 与 RBAC 模型一般在授权过程中通过引入条件实现一定程度的权限动态性，不能对系统间外部交互进行统一访问控制。企业应用架构指明了系统间通信交互中心访问控制需求，上述 4 种安全访问控制模型均具有一定的局限性。

DSGAC 模型定位于使用情景演算方法支持对应用系统客体的访问控制，提出以情景状态演算和情景规则演算相结合的组合推演模式。

DSGAC 模型具有以下创新性。

1) 支持访问控制策略动态更新

提出情景模型，可通过权限、前需、约束与条件 4 个方面，动态定义访问控制的策略。在 DSGAC 模型中情景作为访问控制的依据，支持通过状态演算机制与规则演算机制 2 种方式推演。

2) 支持工作流模型

DSGAC 模型通过引入情景的概念，具有对工

作流的支持。在流程步骤中，利用 DSGAC 访问控制模型,可领域定义情景状态或情景规则。通过建立情景与任务之间的关系，是支持工作流任务访问控制的差异化、动态化、模式化。避免为任务直接授权带来的权限难以维护等弊端。

3) 支持系统间协同访问安全控制

DSGAC 模型能够更好地支持系统协同访问控制，突破了传统访问控制模型的局限性。更好地满足了企业信息化中集中访问控制能力建设的需要。

4) 安全管理的灵活性、便捷性

DSGAC 模型采用情景的概念使访问控制粒度更加适合人的思维，采用情景封装访问控制影响因素,可以实现更加便捷的安全管理。

6 结束语

网构环境中主体、客体系统交互访问多样化，促使协同环境中客体访问安全成为了一个亟待研究解决的问题，为此提出了一种基于动态情景网关的系统协同访问控制模型 DSGAC。DSGA 更好地满足了应用系统间通信架构建设的需要，应用系统间通信安全打造了协同访问控制理论模型。采用情景综合表达影响访问许可多种因素，情景模型是一个更加全面访问控制模型,使 DSGAC 模型更具有通用性,可以适用于权限动态控制、约束控制、前置需求控制与条件控制系统。DSGAC 模型有效地加强了系统自身安全性和协同访问控制的灵活性。

参考文献：

[1] SANDHU R, COYNE E J, FEINSTEIN H L, *et al*. Role-based access control models[J]. IEEE Computer, 1996, 29(2):38- 47.

[2] SANDHU R, MUNAWER Q. The ABRAC99 model for role based administration of roles[J]. ACM Transactions on Information and System Security, 1999, 2(1):105-135.

[3] PANG C Y, HANSEN D, MAEDER A. Managing RBAC States with transitive relations[A]. Proceedings of the 2nd ACM Sym-posium on Information, Computer and Communications Security(ASIACCS'07)[C]. Singapore, 2007.139-148.

[4] JOSHI J B D, BERTINO E, LATIF U, *et al*. A generalized temporal role-based access control model[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(1):04-23.

[5] 王小明, 赵宗涛. 基于角色的时态对象存取控制模型[J]. 电子学报, 2005, 09, 33(9):1634-1638.
WANG X M, ZHAO Z T. Role-based access control model of temporal object[J]. Acta Electronica Sinica, 2005, 33(9): 1634-1638.

[6] 许峰, 赖海光, 黄皓等. 面向服务的角色访问控制技术研究[J]. 计算机学报, 2005, 28(4):686-693.
XU F, LAI H G, HUANG H, *et al*. Service-oriented role-based access control[J]. Chinese Journal of Computers, 2005, 28(4):686-693.

[7] 徐伟, 魏峻, 李京. 面向服务的工作流访问控制模型研究[J]. 计算机研究与发展, 2005, 42(8):1369-1375.
XU W, WEI J, LI J. A service-oriented workflow access control model[J]. Journal of Computer Research and Development, 2005, 42(8):1369-1375.

[8] 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, 29(4):90-98.
LI X F, FENG D G, CHEN Z W, *et al*. Model for attribute based access control[J]. Journal on Communications, 2008, 29(4):90-98.

[9] KERN A, WALHORN C. Rule support for role based access control[A]. Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies[C]. Stockholm, Swden, 2005. 38-130.

[10] YUAN E, TONG J. Attributed based access control(ABAC)for Web service[A]. Proceedings of the 2005 IEEE International Conference on Web Services[C]. Orlando FL, USA, 2005. 561-569.

[11] DOUGHERTY D J, FISLER K, KRISHNAMURTHI S. Specifying and reasoning about dynamic access-control policies[A]. 3rd International Joint Conferenceon Automated Reasoning(IJCAR)[C]. Seattle, USA, 2006. 632-646.

作者简介：

郭树行（1978-），男，河北沧州人，博士，中央财经大学讲师，主要研究方向为软件工程、项目管理、企业信息化。

张禹（1991-），男，北京人，中央财经大学学生，主要研究方向为信息管理。