访问控制

1. 访问控制系统需要包括哪4个要素
2. DAC，MAC模型中授权分别由谁决定

LOGO



课外阅读：

- N. Li, J. Byun, and E. Bertino, “A Critique of the ANSI Standard on Role-Based Access Control,” IEEE Security & Privacy, Nov. 2007, pp. 41-49
- D.F. Ferraiolo, R. Kuhn, R. Sandhu (2007), "RBAC Standard Rationale: comments on a Critique of the ANSI Standard on Role Based Access Control', *IEEE Security & Privacy*, vol. 5, no. 6 (Nov/Dec 2007), pp. 51-53
- From ABAC to ZBAC: the Evolution of Access Control Models,” tech. report HPL-2009-30, HP Labs, 21 Feb. 2009
- D.R. Kuhn, E.J. Coyne, T.R. Weil, "Adding Attributes to Role Based Access Control", *IEEE Computer*, vol. 43, no. 6 (June, 2010), pp. 79-81
- Eric Yuan, Jin Tong, Attribute Based Access Control, A New Access Control Approach for Service Oriented Architectures (SOA), New Challenges for Access Control Workshop, April 27, 2005
- **Guide to Attribute Based Access Control (ABAC) Definition and Considerations**
  SP 800-162 FAQ

1. 访问控制概述

重要性与特点：

- 最基本的安全服务
- 与业务系统密切相关，随着业务与应用发展出现了新的模型

Re
vu
co

fur
B. L

访问控制构成

•1.授权（Authorization）：规定可对该资源执行的操作/权限（与系统相关，如

）。

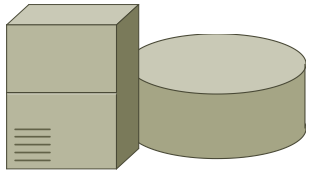
(subject; object; operation)

➢2.策略policy

策略

身份认证

访问控制



审计

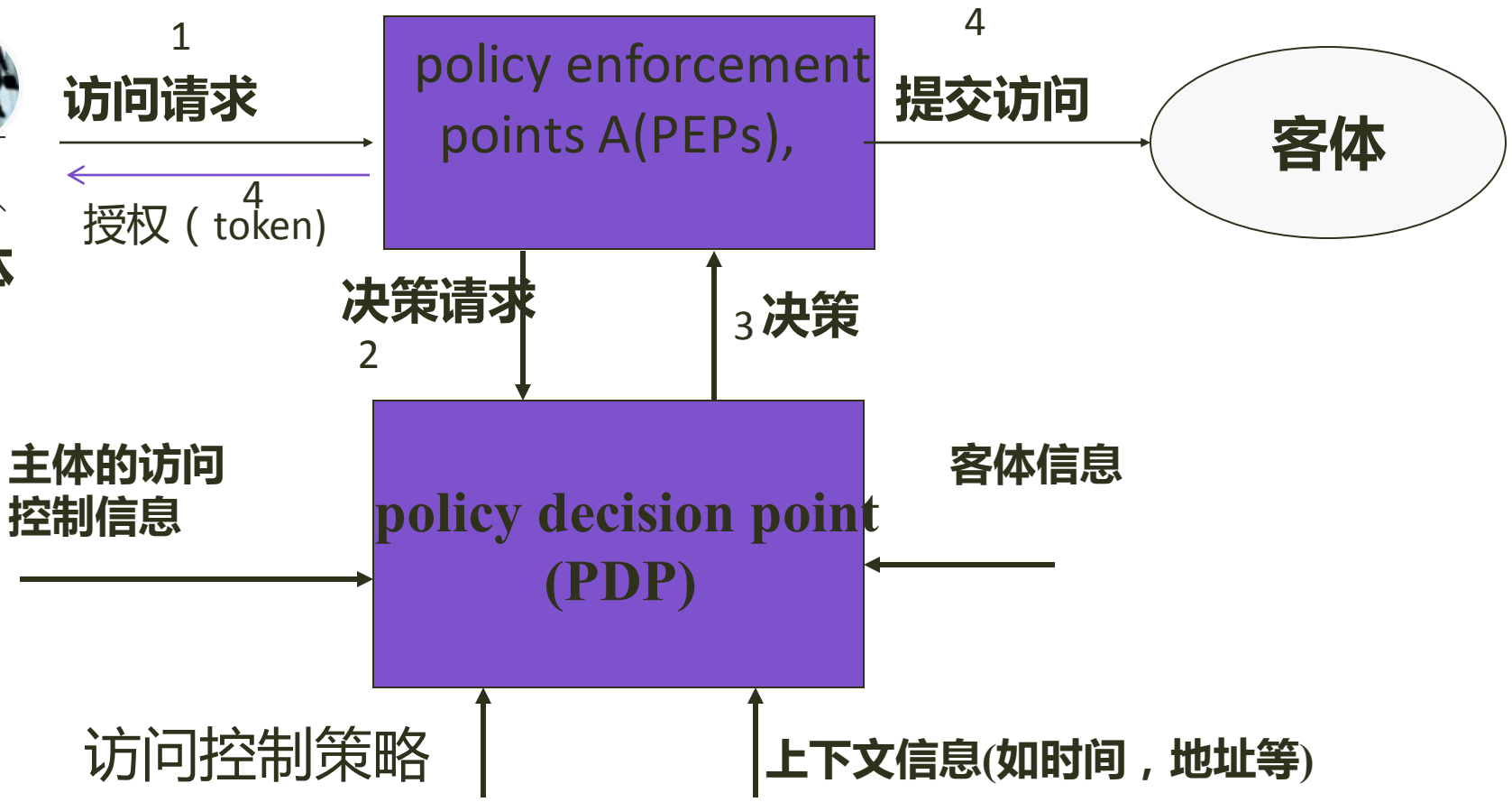
Log

3.主体（Subject）：或称为发起者(Initiator)，是一个主动的、可以访问资源的实体（users、processes、tasks,etc.）。可组织成group，roles等

•3.客体（Object）：又称作目标（target），规定需要保护的资源（所有可供访问的软、硬件资源、数据、信息等）。

2.访问控制模型

主体


1
访问请求
4
授权（token)
policy enforcement points A(PEPs),
4
提交访问
客体
决策请求
2
3 决策
主体的访问控制信息
policy decision point (PDP)
客体信息
访问控制策略
上下文信息(如时间，地址等)



2.1访问控制模型发展

- **业务发展：**
  - 单机的操作系统访问控制
  - 封闭的企业应用
  - 跨域的企业应用
  - 开放业务环境应用
  - 服务聚合
  - …

2.2 基本模型：Ownership and Administration

谁来决定授权？

- 自主访问控制（discretionary Access control），又称任意访问控制，选择性访问控制，它允许用户可以自主地在系统中规定谁可以存取它的资源实体。

  所谓自主，是指具有授与某种访问权力的主体（用户）能够自己决定是否将访问权限授予其他的主体。

- 强制访问控制(Mandatory Access control），指用户的权限和文件（客体）的安全属性都是固定的，由系统决定一个用户对某个文件能否实行访问。

  所谓“强制”，是指安全属性由系统管理员人为设置，或由操作系统自动进行设置，用户不能修改这些属性。 （在高安全级别的系统中使用，B1级以上）

2.2.1DAC实现结构：访问控制矩阵

- 访问控制机制可以用一个三元组来表示（S,O,A）
  - 主体的集合 $S=\{s_1,s_2,\ldots,s_m\}$
  - 客体的集合 $O=\{o_1,o_2,\ldots,o_n\}$
  - 所有操作的集合 $A=\{R, W, E,\ldots\}$
  - 访问控制矩阵 $M= S \times O \rightarrow 2^A$

$$M=(a_{so})_{s\in S,\, o\in O},\ a_{so}\subset A$$

$$M=\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0n} \\ a_{10} & a_{11} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m0} & a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

对于任意$s_i \in S, o_j \in O$,存在$a_{ij} \in M$决定s_i对o_j允许的操作。比如$a_{ij}=\{R,W\}, a_{lk}=\phi$

访问控制矩阵

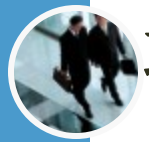
$$M=\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0n} \\ a_{10} & a_{11} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m0} & a_{m1} & \cdots & a_{mn} \end{bmatrix}=\begin{bmatrix} S_1 \\ S_2 \\ \cdots \\ S_m \end{bmatrix}=\begin{bmatrix} O_1 & O_2 & \cdots & O_n \end{bmatrix}$$

- 由于M常为稀疏矩阵，所以常用：
  - **访问控制表（Access Control List ）:每个客体附加一个可以访问它的主体的明细表。**矩阵的第j列O_j表示客体o_j允许的所有主体操作，即为o_j的ACL。
  - **目录表:每个主体附加一个该主体可访问的客体的目录表：**矩阵的i行S_i表示了主体s_i对所有客体的操作权限，即为主体s_i的目录表或能力表(Capability List)

DAC特点

- **由个体（creator）决定权限。**
- **授权基于主体和客体的标识/身份（identity）。**
- **其自主性为用户提供了极大的灵活性，适合于小规模的系统和应用。**
- 访问控制矩阵可扩展性差。
- 无法控制信息流动，信息在移动过程中其访问权限关系会被改变。如用户A可将其对目标O的访问权限传递给用户B(copy，so A is the creator of the new data)，从而使不具备对O访问权限的B可访问O。
- **无法防止特洛伊木马攻击（Trojan horse attack)**

2.2.2强制访问控制



➢强制——安全属性（级别）由系统管理员人为设置，或由操作系统自动地按照严格的安全策略与规则进行设置，用户和他们的进程不能修改这些属性。

如：Clearance ，classification , sensitivity

Unclassified< confidential< secret< top secret

普密<秘密<机密<绝密

➢强制访问控制——指访问发生前，系统通过比较主体和客体的安全级别来决定主体能否以他所希望的模式访问一个客体。

强制型访问控制（续）

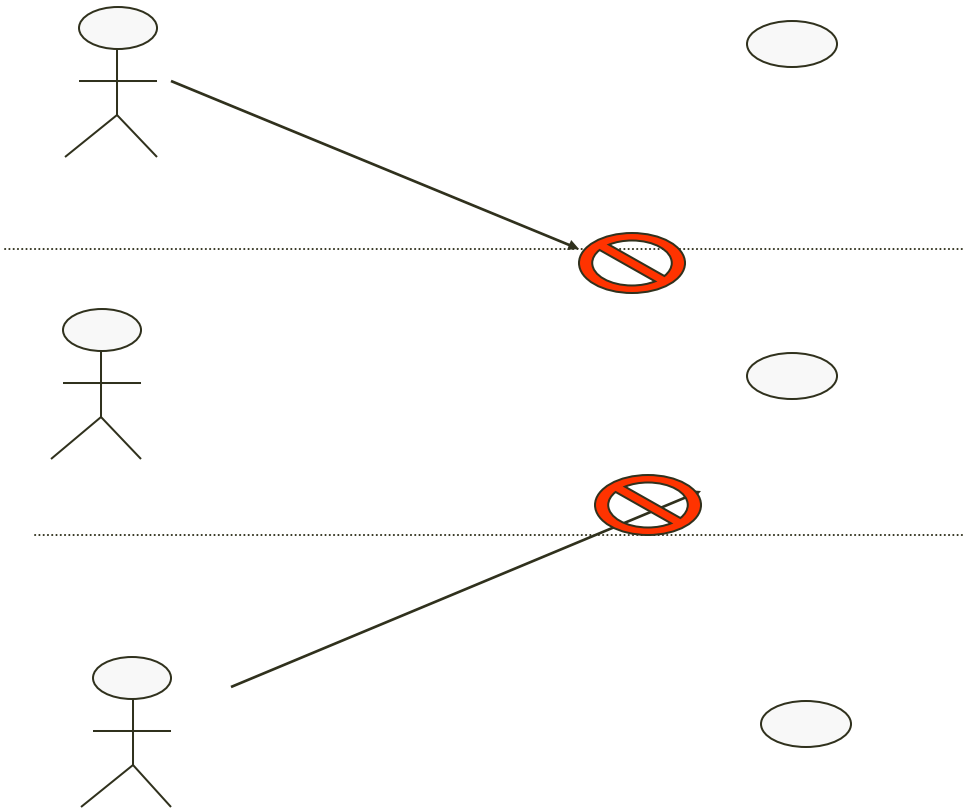
- 最典型的MAC

BLP（ Bell and LaPadula）模型

- 禁止向下写：

  如果用户的级别比要写的客体级别高，则该操作是不允许的

- 禁止向上读：
  - 如果主体的级别比要读的客体级别低，则该操作是不允许的。





MAC的问题

- 限制了高安全级别用户向非敏感客体写数据的合理要求
- 高安全级别的主体拥有的数据永远不能被低安全级别的主体访问，降低了系统的可用性。
- 不能同时实现系统对机密性和完整性（不可篡改）的要求
- 过于偏重保密性，对其它方面如系统连续工作能力、授权的可管理性等考虑不足，造成管理不便，灵活性差。
- 比较适合与等级划分严格的行业
- 当存在covert channel（隐密信道）时，这种访问准则会被破坏。 两类： timing and storage covert channels

Biba模型则具有不允许向下读、向上写的特点，可以有效地保护数据的完整性



3.1 Chinese Wall

- Introduced by Brewer and Nash in 1989 (BN model)
- It dynamically establishes the access rights of a user based on what the user has already access
- The motivation for this work was to avoid that sensitive information concerning a company be disclosed to competitor companies (Conflict-of-Interest (CoI) classes)through the work of financial consultants



Chinese Wall Policy

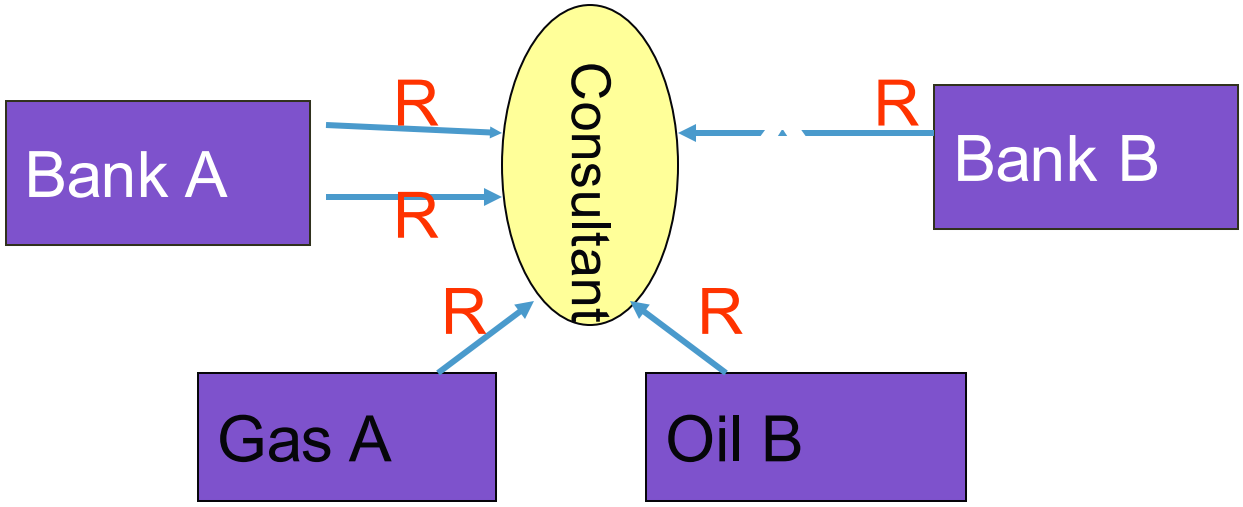
- ***Subjects*: Active entities accessing protected objects**
- ***Objects*: Data organized according to 3 levels**
  - Information
  - DataSet
  - Conflict-of-Interest (CoI) classes
- ***Access Rules***
  - Read rule
  - Write rule

Data Classification

3.2 Read Rule

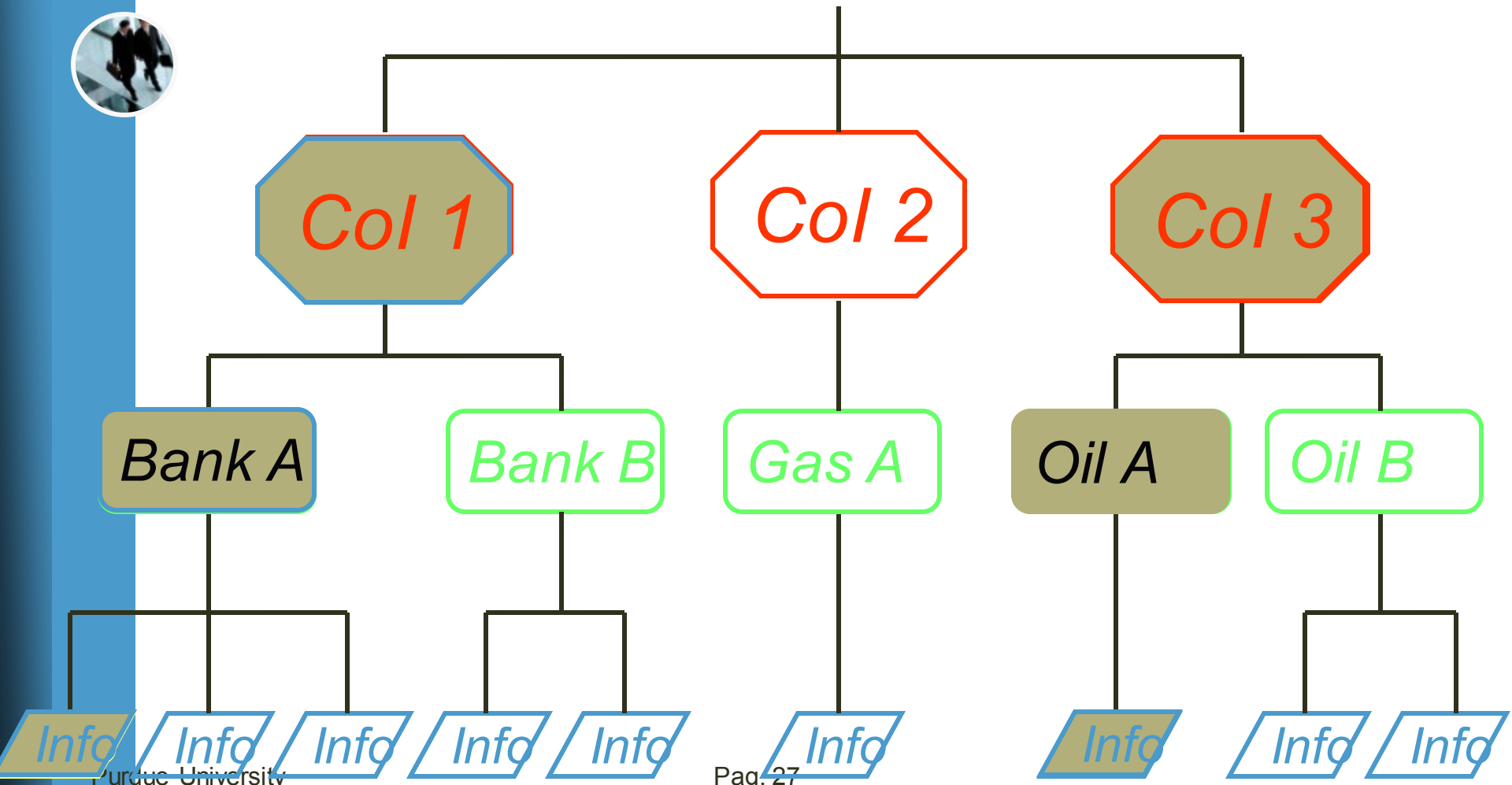
Read Rule: A subject S can read an object O if :

- O is in the same Dataset as an object already accessed by S OR
- O belongs to a CoI from which S has not yet accessed any information



Read Rule

- John





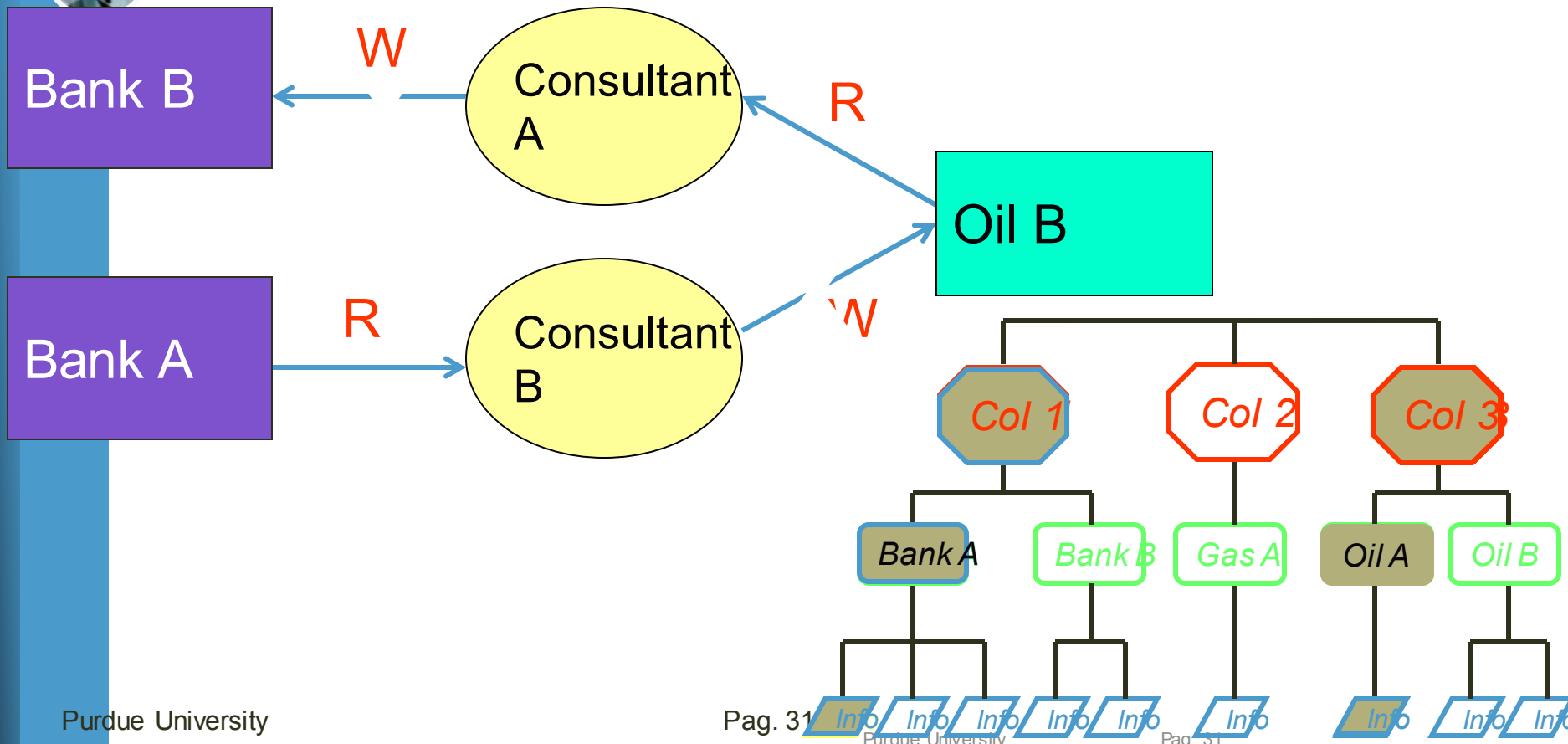
Comparison with Bell-LaPadula

- The Chinese Wall Policy is a **combination of free choice and mandatory control**
- Initially a subject is free to access any object it wishes
- Once the initial choice is made, a *Chinese Wall* is created for that user around the dataset to which the object belongs
- Note also that a Chinese Wall can be combined with DAC policies

3.3 Write Rule

Write Rule: A subject S can write an object O if:

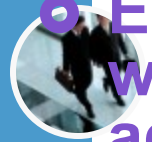
- S can read O according to the Read Rule AND
- No object has been read by S which is in a different company dataset to the one on which write is performed



Purdue University
Pag. 31

4.1 RBAC: Motivations

In enterprise multiuser systems: a large (but structured) number of users



- End users often do not own the information for which they are allowed access. The corporation or agency is the actual owner of data objects
- Control is often based on **employee functions** rather than data ownership
- One challenging problem in managing large systems is **the complexity of security administration**
  - Whenever the number of subjects and objects is high, the number of authorizations can become extremely large
  - Moreover, if the user population is highly dynamic, the number of grant and revoke operations to be performed can become very difficult to manage
- RBAC has been proposed as an alternative approach to DAC and MAC both to simplify the task of access control management and to directly support **function-based** access control



4.2 RBAC （Role based access control)

基于角色的访问控制模型

- 目的：解决访问控制管理的复杂性
- 原理：将访问权限分配给角色，用户担任一定的角色，从而具有角色对应的权限
- 假设：用户变化频繁，角色相对稳定
- 标准： American national standard for information technology – role based access control. ANSI INCITS 359-2004, February 2004

参考
http:/



4.3 不同RBAC模型

- **NIST Model：**
  - Core RBAC – also called Flat RBAC
  - Hierarchical RBAC
  - Constrained RBAC
  - Symmetric RBAC

RBAC96模型分为四个层次:
RBAC0、RBAC1、RBAC2和RBAC3，其中RBAC0是基本模型，RBAC1添加了角色的层次关系，RBAC2添加了约束，RBAC3是RBAC1和RBAC2的结合。

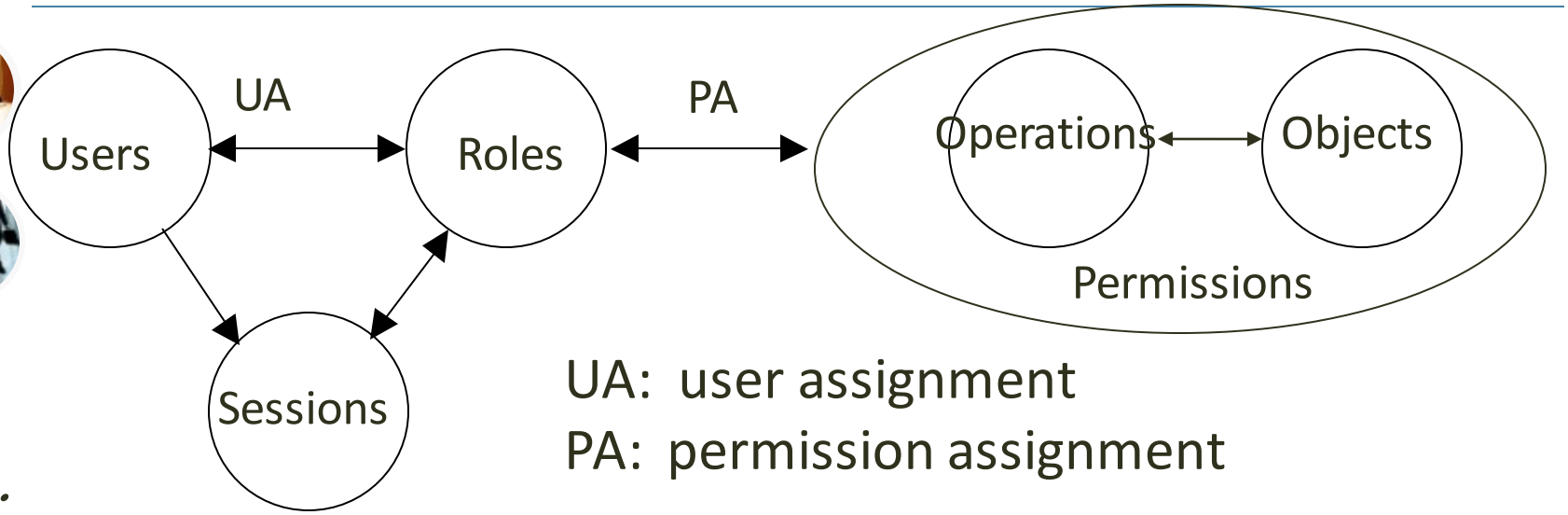
Ravi Sandhu, Edward Coyne, Hal Feinstein, and Charles Youman. Role-based access control models. IEEE Computer, 29(2):38{47, Feb. 1996.

还有各种演变模型

Level	Name	RBAC Functional Capabilities
1	**Flat RBAC** (see Figure 1)	• users acquire permissions through roles • must support many-to-many user-role assignment • must support many-to-many permission-role assignment • must support user-role assignment review • users can use permissions of multiple roles simultaneously
2	**IIierarchical RBAC** (see Figure 2)	Flat RBAC + • must support role hierarchy (partial order) • **level 2a** requires support for arbitrary hierarchies • **level 2b** denotes support for limited hierarchies
3	**Constrained RBAC** (see Figures 6 and 7)	Hierarchical RBAC + • must enforce separation of duties (SOD) • **level 3a** requires support for arbitrary hierarchies • **level 3b** denotes support for limited hierarchies
4	**Symmetric RBAC** (see Figures 9 and 10)	Constrained RBAC + • must support permission-role review with performance effectively comparable to user-role review • **level 4a** requires support for arbitrary hierarchies • **level 4b** denotes support for limited hierarchies

Ferraiolo D, Sandhu R, Gavrila S, Kuhn D, Chandramouli R(2001) Proposed NIST standard for role-based access control. ACM Trans Inf Syst Secur 4(3):224–274

4.3.1 Core RBAC - Permissions



UA: user assignment
PA: permission assignment

Set :

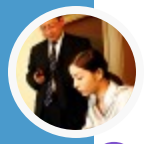
USERS, *ROLES*, *Object*, *Operation*, *Permission(Privilege)* = $2^{(OPS \times OBS)}$

Session

UA $\subseteq$ *USERS x ROLES*, a many-to-many mapping user-to-role assignment relation

PA $\subseteq$ *PRMS x ROLES*, a many-to-many mapping permission-to-role assignment relation

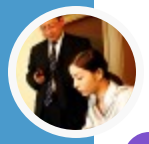
Session , "*a mapping between a user and an activated subset of roles that are assigned to the user*"


Core RBAC - Sessions

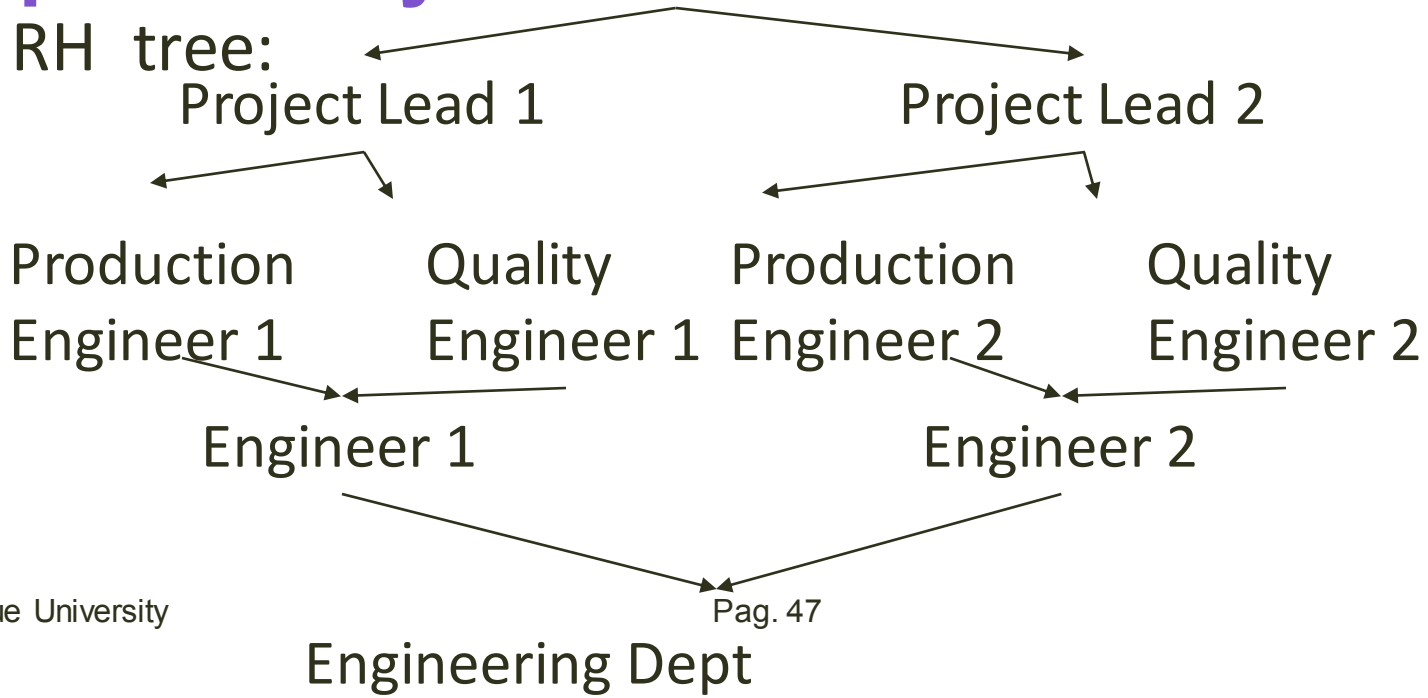
- **The notion of session is quite abstract – it is defined as "*a mapping between a user and an activated subset of roles that are assigned to the user*"**
- **Basic distinction:**
  - *Single-role activation (SRA)* Only one role can be activated
  - *Multi-role activation (MRA)* Multiple roles can be activated in one session, and dynamic separation of duty constraints may be used to restrict concurrent activation of some roles
  - There are trade-offs between the use of these two types of session

4.3.2Hierarchical RBAC - Motivations


- Role hierarchies are a natural means for **structuring roles** to reflect an organization's line of authority and responsibility

Example of RH tree:





Hierarchical RBAC - Model

- ***RH*** $\subseteq$ ***ROLES x ROLES*** **it is a relation defined on the set of roles in the system**
- **It has to be irreflexive and acyclic. We refer to this relation as *dominance relation;* if** $(r_i, r_j) \in RH$ **we say that** r_i **dominates** r_j
- **We also define a partial order** $\geq$ **which is the reflexive and transitive closure of *RH*.**



Hierarchical RBAC - Semantics

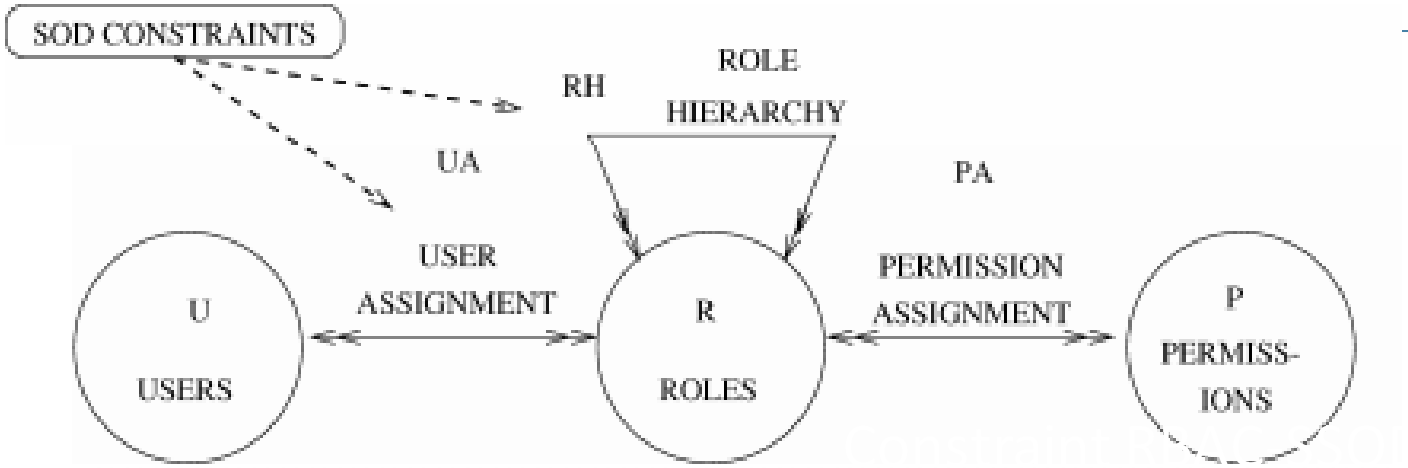
- User Inheritance (*UI*): All users authorized for a role *r* are also authorized for any role *r'* where $r \geq r'$
- Permission Inheritance (*PI*): A role *r* is authorized for all permissions for which any role *r'*, such that $r \geq r'$, is authorized
- Activation Inheritance (*AI*): Activating a role *r* automatically activates all roles *r'*, such that $r \geq r'$. This semantics can be used only if MRA sessions are used

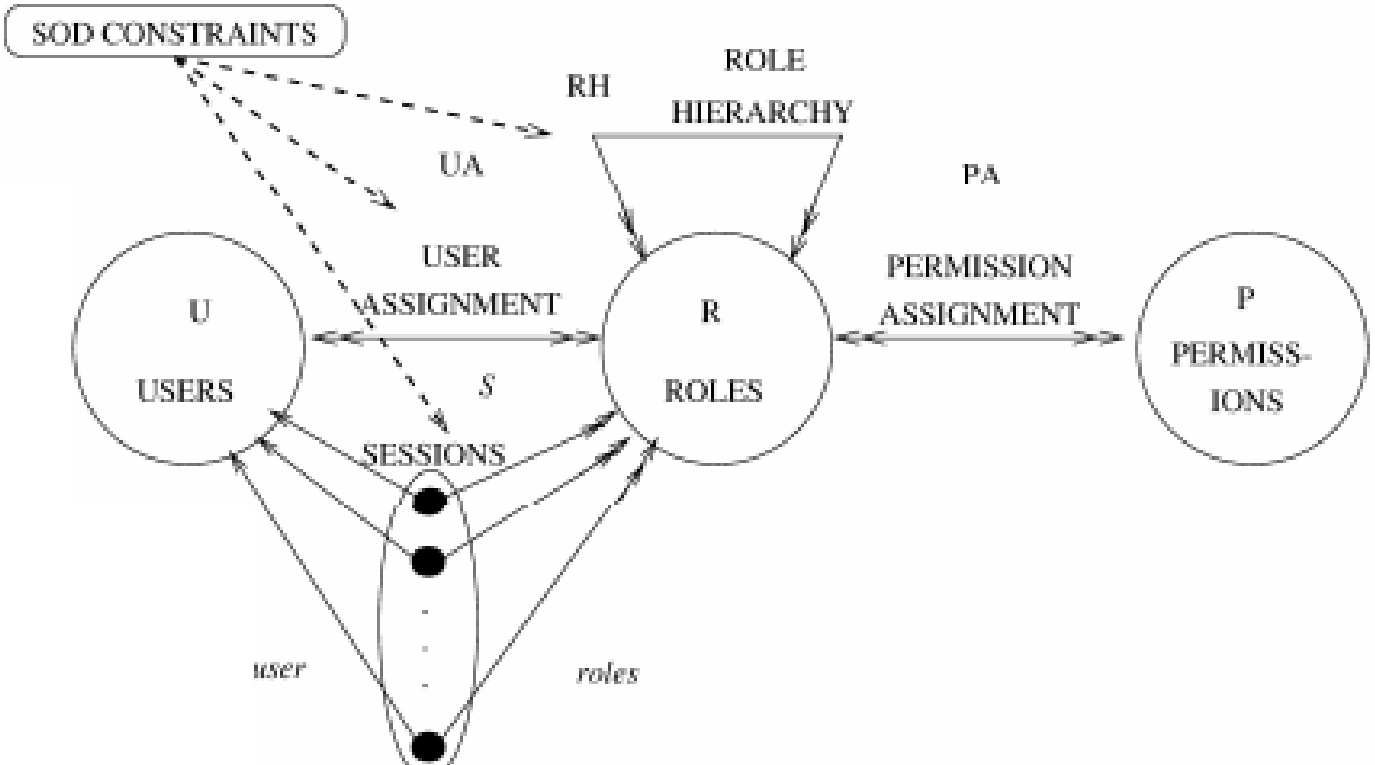


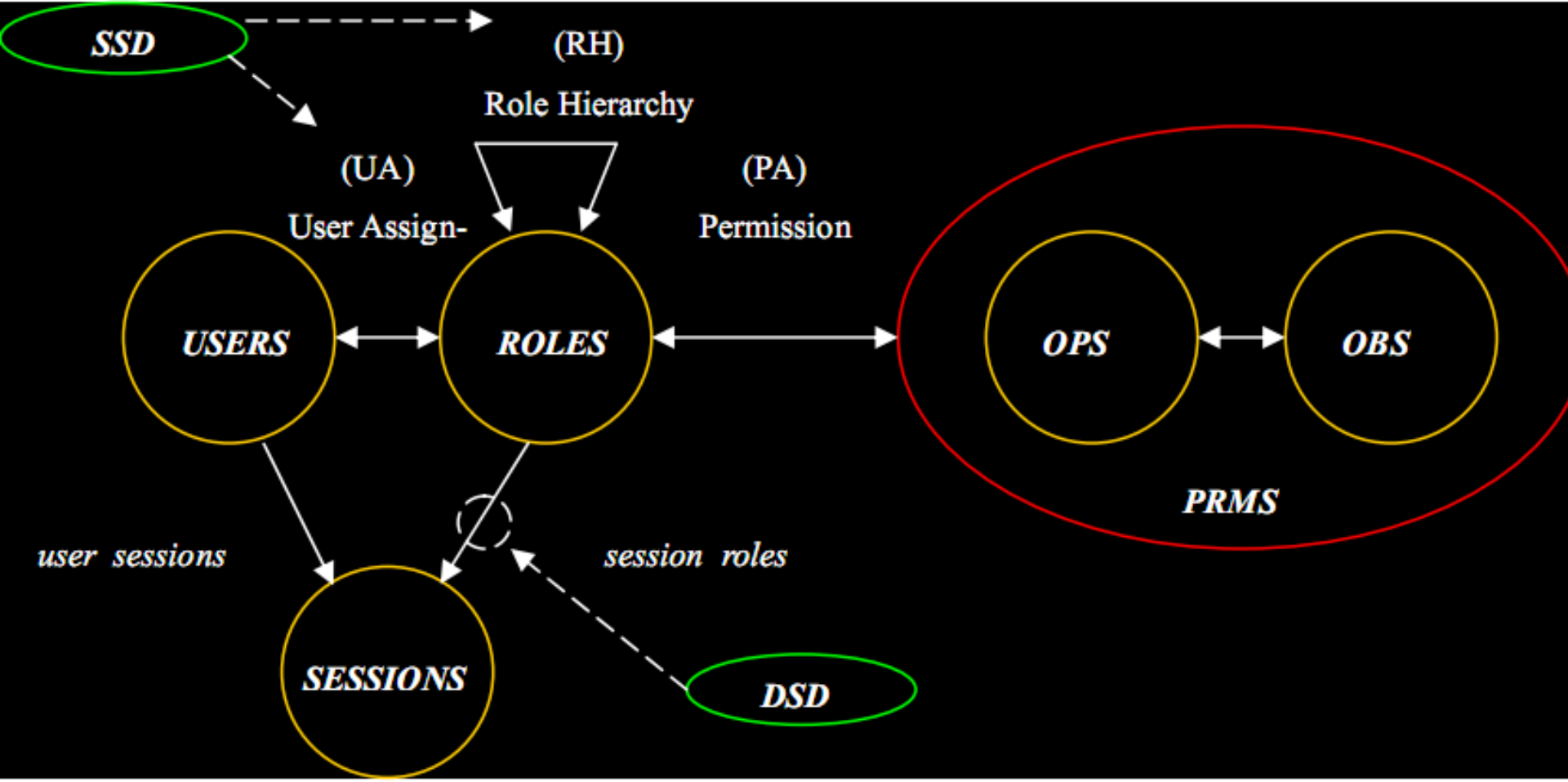
4.3.3 Constrained RBAC

- **Constrained RBAC is an RBAC model with the capability of supporting *Separation of Duties* policies**
- **Definition:**
  - ANSI: “Dividing responsibility for sensitive information so that no individual acting alone can compromise the security of the data processing system”
  - The U.S. Office of Management and Budget’s Circular A-123: “Key duties and responsibilities in authorizing, processing, recording, and reviewing official agency transactions should be separated among individuals”.
- **Two main categories:**
  - Static SoD （based on user-role assignment)
  - Dynamic SoD (based on role activation)

4.4 RBAC 小结


SOD CONSTRAINTS
RH
ROLE
HIERARCHY
UA
PA
USER
ASSIGNMENT
PERMISSION
ASSIGNMENT
U
USERS
R
ROLES
P
PERMISS-
IONS


SOD CONSTRAINTS
RH
ROLE
HIERARCHY
UA
PA
USER
ASSIGNMENT
PERMISSION
ASSIGNMENT
U
USERS
S
R
ROLES
P
PERMISS-
IONS
SESSIONS
user
roles



1. 模型中有了UA完成用户到角色的映射，为什么还需要定义session，session的作用是什么
2. 图中SSD和DSD是什么，分别举一个业务实例，说明SSD和DSD的意义
3. 在图中的RH（Role Hierarchy）,角色增加了一个新的语义，或者说角色间有了一种新的关系，是：



Other models based on RBAC (examples)

- **TRBAC （ Temporal RBAC ） express temporal constraints**
  - Time = 19:00 Alice DayDoctor deny
  - Time = 9:00 Alice DayDoctor permit
- **GTRBAC**
  - extends TRBAC by introducing temporal conditions on:
    - User-role assignments
    - Role-permission assignments

“ther
“the



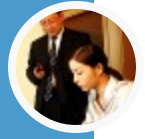
Context aware RBAC

- Context: temporal, spatial, environment…
- A context constraint specifies that certain conditions must be fulfilled to permit the execution of a particular task
- RBAC supports the definition of context constraints on various parts of an RBAC model

RBAC的特点

- **RBAC的优势**
  - 便于授权管理，便于权限划分
  - 策略与访问控制模型分离
  - 操作系统、数据库中广泛的支持
- **RBAC的问题**
  - 角色限定了权限，难以实现细粒度的访问权限管理
  - 必须预先知道用户信息，配置用户到角色的分配
  - 前提：角色数目有限，角色的权限稳定

5. ABAC(Attribute based access control)



- Utilize(possibly dynamic) **properties (Attributes)of** subjects and objects as the basis for authorization
- **Rules** specify conditions under which access is granted or denied.
- No standard for ABAC currently

5.1 ABAC Policy Formulation

1. *S*, *R*, and *E* are subjects, resources, and environments;
2. SA_k ($1 \le k \le K$), RA_m ($1 \le m \le M$), and EA_n ($1 \le n \le N$) are the pre-defined **attributes** for subjects, resources, and environments;
3. *ATTR(s)*, *ATTR(r)*, and *ATTR(e)* are attribute assignment relations for subject *s*, resource *r*, and environment *e*, respectively:

$$ATTR(s) \subseteq SA_1 \times SA_2 \times \ldots \times SA_K$$

$$ATTR(r) \subseteq RA_1 \times RA_2 \times \ldots \times RA_M$$

$$ATTR(e) \subseteq EA_1 \times EA_2 \times \ldots \times EA_N$$

ABAC Policy Formulation (Cont'd

- In the most general form, a ***Policy Rule*** that decides on whether a subject *s* can access a resource *r* in a particular environment *e*, is a Boolean function of *s*, *r*, and *e*'s attributes:

$$Rule\ X : can\_access(s, r, e) \leftarrow$$
$$f(ATTR(s), ATTR(r), ATTR(e))$$

Policy Rule Examples

- **Modeling conventional RBAC rules:**
  - "User with role 'Manager' may access the 'ApprovePurchase' web service"

$$Rule\ 1: can\_access(s, r, e) \leftarrow Role(s) = 'Manager' \wedge Name(r) = 'ApprovePurchase'$$

- **Modeling richer access control semantics**
  - "A resource may only be accessed by its owners"

$$Rule\ 2: can\_access(s, r, e) \leftarrow UserID(s) = OwnerID(r)$$

- **Modeling mandatory acc control**
  - "Classified files can be accessed by users with equal or higher clearance"

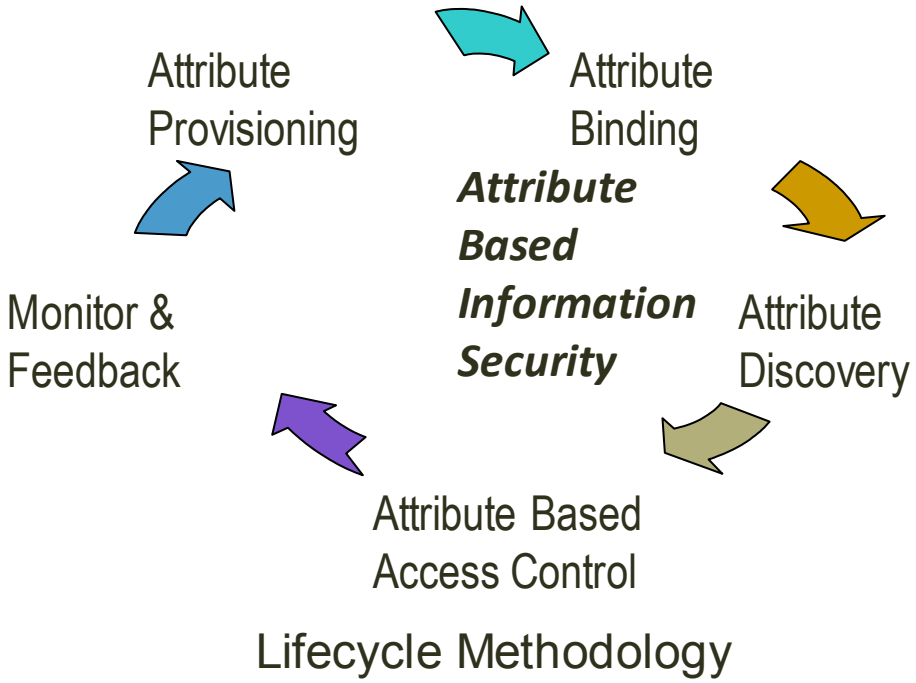
$$Rule\ 3: can\_access(s, r, e) \leftarrow Clearance(s) \geq Classification(r)$$

5.2 Attribute Management

How attributes are managed throughout their life cycle:

- Attribute definition "provisioning";
- Cryptographic mechanisms to "bind" attributes to subjects and objects ;

(who is attribute authority)



Lifecycle Methodology

GENI: ABAC policies are a combination of X.509v3 identity certificates and X.509v2 attribute certificates.

Shibboleth: is an attribute authority service developed by the Internet2 community for cross-organization identity federation;

- Discovery mechanisms for attribute definitions and attribute ssignments;
- A "feedback loop" through which attribute usage can be monitored and audited



Policy Evaluation

- Given attribute assignments, the **evaluation of policy rules** may be boiled down to the evaluation of **First Order Logic** expressions, or its simpler, computationally more attractive subsets (e.g. **Description Logic, Horn Logic**)
- Natural marriage with Semantic Web technologies for attribute and policy representations
  - E.g., attribute assignments as OWL axioms
  - Existing inference engines / reasoners may be readily leveraged
- Implementation aspects: E.g., **complexity**



5.3 ABAC vs RBAC:
Online Entertainment Store Example

- **Inspired by [Al-Kahtani & Sandhu 2003]:**
  - Streams movies to customers for a flat monthly fee
- **Access Control Requirements**
  - Basic requirement: access control is based on user's age and the movies' content ratings (R, PG-13, G)
  - Advanced requirement: Suppose the store introduces membership classes (Premium, Regular) and would like to enforce a new policy that only Premium users can view New Releases
- **Using RBAC:**
  - Basic policy: Need to create roles such as *Adult*, *Juvenile*, *Child*
  - Advanced policy: Roles and permissions need to be doubled (e.g., *Adult Premium*, *Adult Regular*, …)
  - I.e., given K subject attributes, total roles could be:
  - UA with unknown user

$$\prod_{1}^{k} Range(SA_k)$$

Online Entertainment Store Example



- **Using ABAC:**
  - Basic policy: No need to define explicit roles

$$R1: can\_access(u, m, e) \leftarrow$$
$$(Age(u) \geq 21 \wedge Rating(m) \in \{R, PG13, G\}) \quad \vee$$
$$(21 \geq Age(u) \geq 13 \wedge Rating(m) \in \{PG13, G)\} \quad \vee$$
$$(Age(u) < 13 \wedge Rating(m) \in \{G\})$$

  - Advanced policy: Only need to add a second rule and combine the two

$$R2: can\_access(u, m, e) \leftarrow$$
$$(MemberType(u) = 'Premium') \quad \vee$$
$$(MemberType(u) = 'Regular' \wedge$$
$$MovieType(m) \notin \{'NewRelease'\})$$
$$R3: can\_access(u, m, e) \leftarrow \quad R1 \wedge R2$$



Comparison with (Pure) RBAC

- **Inherent limitation in RBAC is the single dimension of roles**
  - Finer-grained access control policies often involve multiple subject and object attributes
  - As more attributes are involved, number of roles and permissions needed to encode these attributes will grow exponentially, thereby making User Assignments and Permission Assignments difficult to manage
- **RBAC usually needs centralized management of user-to-role and permission-to-role assignments**
  - Not well suited for a highly distributed environment
  - Even more difficult when subject and resource belong to different security domains!

Comparison with (Pure) RBAC



- **RBAC doesn't consider environment attributes explicitly**
  - E.g., continuing with the previous example, suppose an additional requirement states "Regular customers in general may not watch new releases, but may be allowed in promotional periods"
  - In ABAC, a new rule can be easily added, involving an environment attribute *CurrentDate(e)*
- **RBAC doesn't handle MAC**
  - In ABAC, security labels can be treated naturally as attributes

ABAC advantages

- **The ABAC model brings many advantages over traditiona identity or role based models**

- Intuitive to model and manage real-world access control policies
- More flexible and more powerful to represent complex, fine grained access control semantics, which is especially suitable for the dynamic SOA / web services environment
- Management of security information is spread over a number of Attribute and Policy Authorities, possibly across organizational boundaries – suitable for large-scale information sharing
- Reduces overall system complexity, allowing different system components (user directory, service registry, policy server, etc.) to focus on their respective administrative tasks

disadvantages

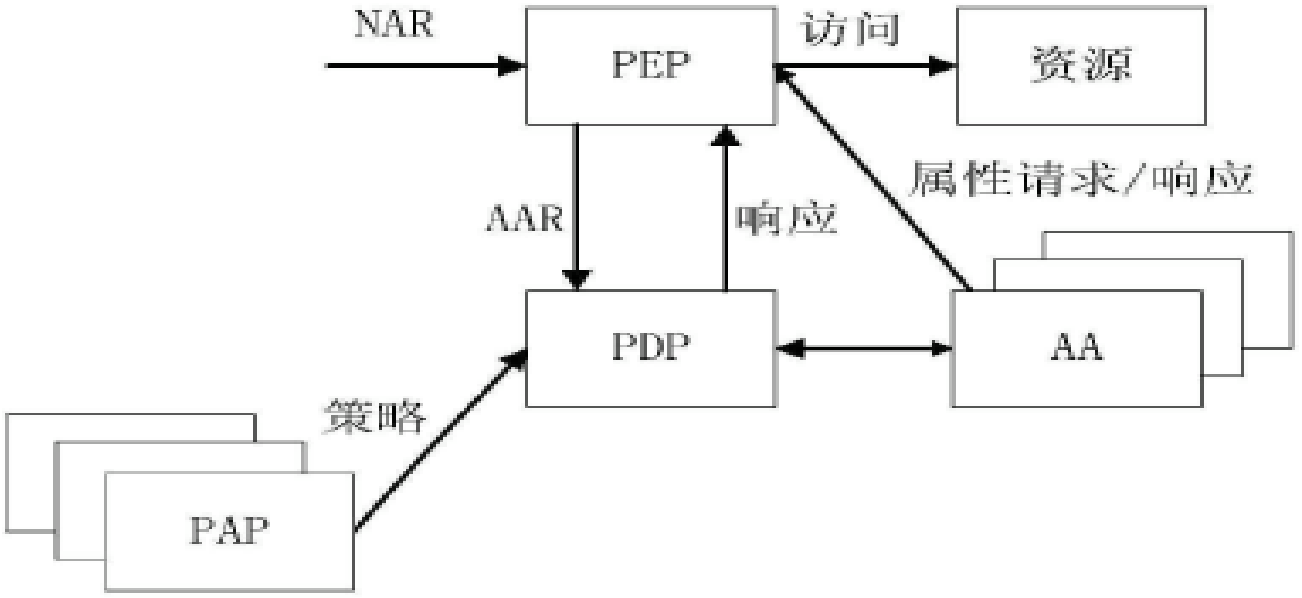
- **Scalability:**
  - for n Boolean attributes or conditions using attributes, there are 2^n possible combinations
  - ABAC is easy to set up, but analyzing or changing user permissions can be problematic.

- **many aspects of the entire attribute management "life cycle" needs to be considered, such as attribute provisioning, binding, discovery, and feedback loop**
  - All are potential research and engineering topics to be explored
- **Policy management**
- **No standard model**



ABAC practice and attribute authority

- Akenti: attributes signed by unrelated stakeholders from different domains.
- PERMIS: E C PERMIS project, is a role-based access control infrastructure based on X.509 PMI and using X.509 attribute certificates
- Shibboleth: attribute authority service developed bythe Internet2 community for cross-organization identity federation, make access decisions based on attributes
- VOMS: European Data Grid and DataTAG projects, , runs in a virtual organization, manages authorization information about its own members, and supplies this information as a kind of attribute certificate
- GENI: http://groups.geni.net/geni/wiki/TIEDABACModel



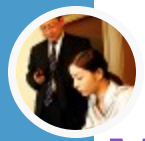
NAR:原始访问请求　AA：属性权威　AAR：基于属性访问请求

PEP：策略执行点　PDP：策略决策点　PAP：策略管理点

图2.7 ABAC框架示意图

Fig2.7 ABAC framework

6.1 UMA(User-Managed Access)



- WEB开放业务环境特点及访问控制需求
  - 资源和服务的多样性
  - 资源共享
  - PUB/SUB业务模式
  - 开放业务环境

- Publish/Share resources in open WEB environment
- 如何解决个人资源发布和共享中的访问控制问题

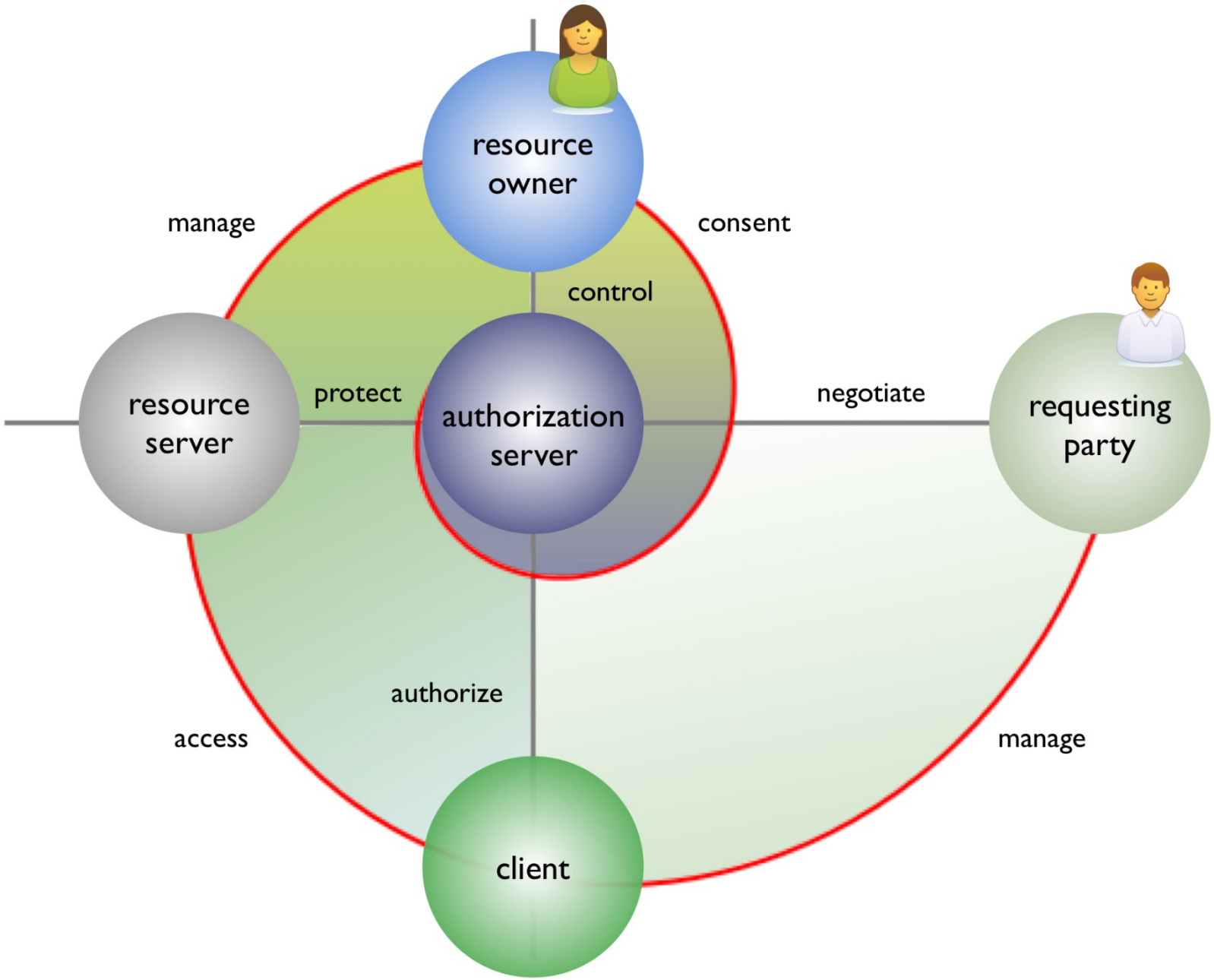


UMA(User-Managed Access)

- In highly dynamic and open web environment, access control **should not be based solely on preliminary identification and authentication of requesters**; servers may not know possible identities of clients accessing data in advance.
- **Access control** should be externalized from web applications and provided **as a dedicated online service** without the real-time presence of the resource

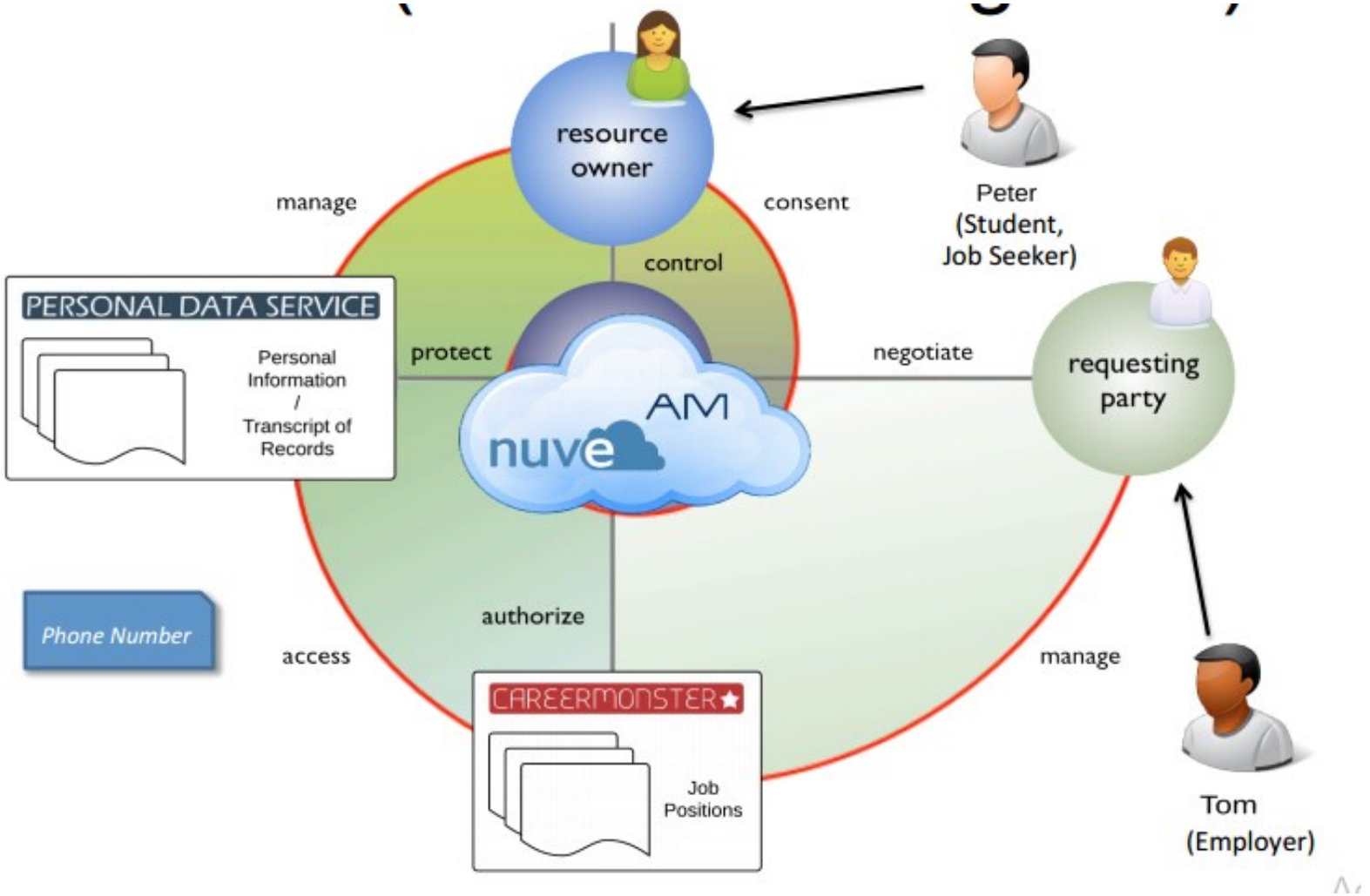


- The User-Managed Access (UMA) core protocol provides a method **based on OAuth 2.0** for **users to control access** to their protected resources, residing on any number of host sites, through a single **authorization manager (AM)** that governs access decisions based on **user policy**.

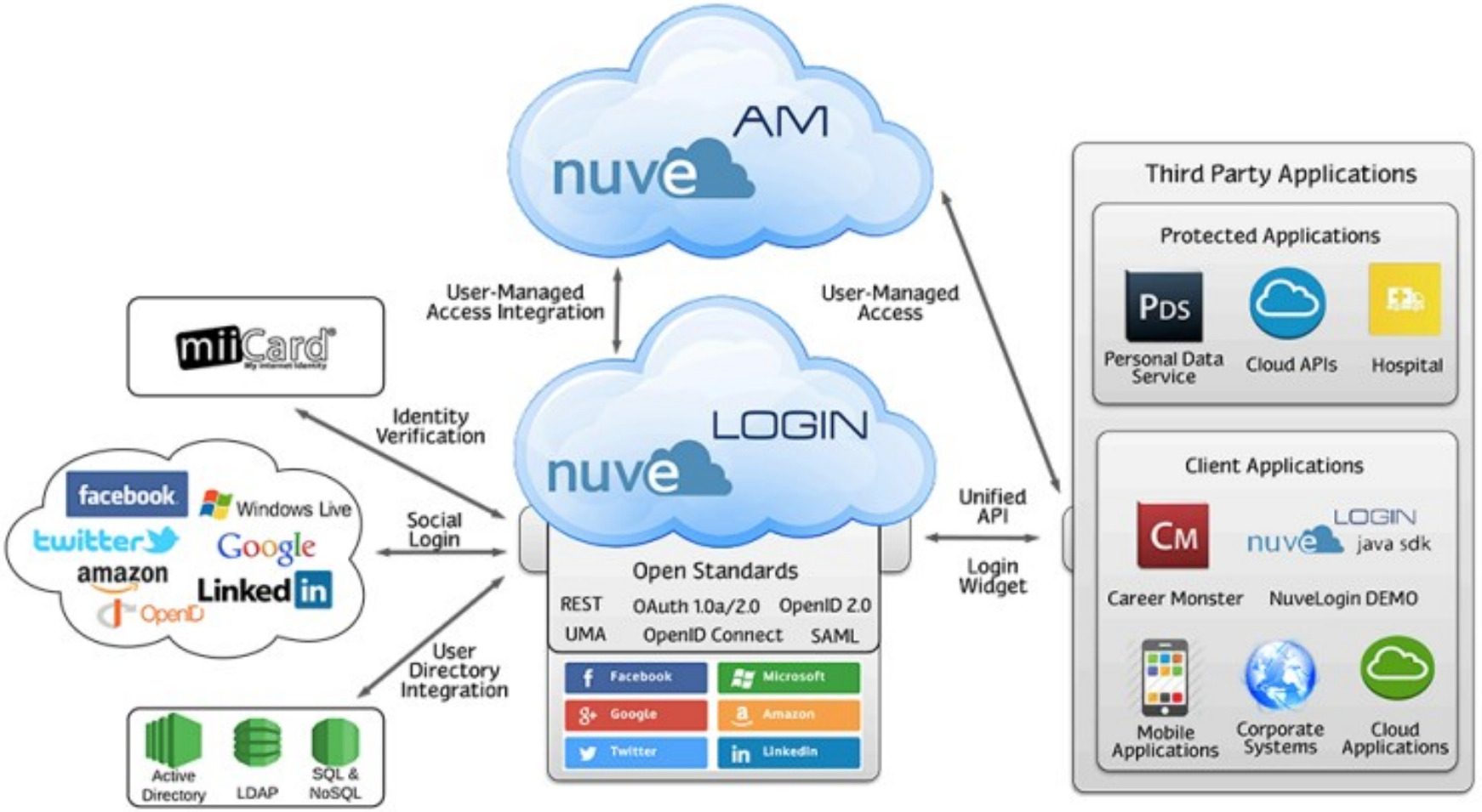

resource owner
manage
consent
control
resource server
protect
authorization server
negotiate
requesting party
authorize
access
manage
client



Scenario:


resource owner
manage
consent
Peter (Student, Job Seeker)
control
PERSONAL DATA SERVICE
Personal Information / Transcript of Records
protect
negotiate
requesting party
AM
nuve
Phone Number
authorize
access
manage
CAREERMONSTER
Job Positions
Tom (Employer)

UMA & IDM



cloudidentity

6.2 Usage Control (UCON) model

- Attribute based model
- Authorization, obligation, condition
- enhances traditional access control models in two novel aspects:
  - mutability of attributes
  - continuity of an access decision

continuity of an access decision means that the decision to allow the access to an object is made not only before the access but also continuously when the access is in progress.

Park,R Sandhu.Towards Usage Control Models:Beyond Traditional Access Control[C].Proceedings of the 7th ACM Symposium on Access Control Models and Technologies,2002.



6.3属性基加密 (Attribute-Based Encryption

ABE通过密码学算法保证只有具有一定属性特征的接收者才可恢复正确的明文

- 密钥策略KP-ABE(Key-Policy Attribute-Based Encryption)
  - 用树结构描述属性，可表达与、或逻辑的访问控制策略,在KeyGen算法中将密钥与属性树关联, Decrypt算法中只有属性符合访问控制策略所定义的属性树时才能正确解密
- 密文策略CP-ABE(Ciphertext-Policy Attribute-Based Encryption)
  - 在Encrypt算法生成的密文中定义访问控制策略要求的属性树结构



7. 总结：访问控制技术

- **Discretionary Access Control (DAC)**
- **Mandatory Access Control (MAC)**
- **Role Based Access Control (RBAC)**
- **Attribute Based Access Contorl(ABAC)**
- **Other topics:**
  - Usage Control (UCON) model
  - Context aware access control
  - Trust Management (TM) and Digital Rights Management (DRM)