第7章 安全模型

南京大学计算机系 黄皓教授

2010年12月13日-12月6日

参考文献

I. Simone Fischer-Hübner，IT-Security and Privacy， Lecture Notes in Computer Science 1958.

II. D. Elliott Bell and Leonard J. LaPadula， Secure Computer Systems: Mathematical Foundations。

III. D. Elliott Bell and Leonard J. LaPadula， Secure Computer Systems: Mathematical Model。

IV. K.J. Biba, Integrity Considerations for Secure Computer Systems, USAF Electronic Systems Division, Bedford, Mass., April 1977.

V. David D. Clark, David Il. Wilson, A Comparison of Commercial and MilitarY computer Security Policies， IEEE Symposium on Security and Privacy April 27 - 29, 1987, pp184-194.

VI. D.Brewer, M.Nash, The Chinese Wall Security Policy, Proceedings of the 1989 IEEE Symposium on Security and Privacy, Oakland, May 1989.

Contents

1. Basic Concept

IT-Security

- View: protection of the system, protection from the system;
- Aims: confidentiality, integrity , availability, reliability, functionality, anonymity, pseudonymity, unobservability, unlinkablity;
- Security models
- Security functions: I&A, AC, Audit, Object reuse, reliability of service,
- Security mechanism：password, ACL, cryptography, physical control,etc.

Security Aims

- ***Confidentiality****:* prevention of unauthorised or improper disclosure of data.
- ***Integrity*****:** data continues to be a proper physical and semantic representation of information.
  1. Preventing unauthorised users from making modifications.
  2. Preventing authorised users from making improper modifications
  3. Maintaining internal data consistency (self-consistency of interdependent data).
     - external data consistency (consistency of data with the real-world environment that the data represents).

Security Aims

- ***Availability*:** prevention of the unauthorised withholding of data or resources.
  - timely response
  - fair allocation
  - fault tolerance
  - utility or usability
  - controlled concurrency

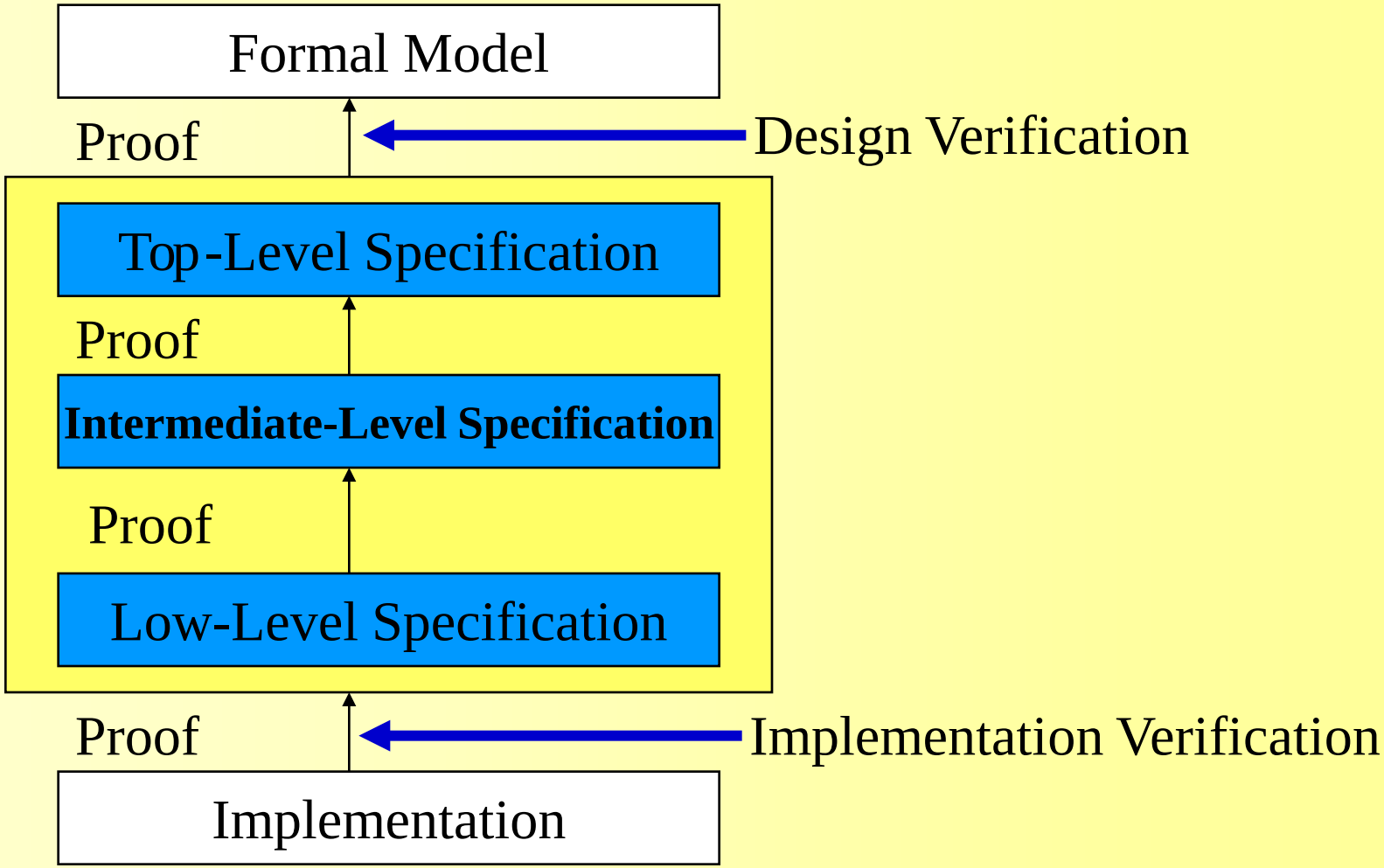
Security Aims

- In the English language, a distinction is made between "**security**" and "**safety**".Important aspects of safety are ***functionality and reliability***
  - **Functionality*:*** the system performs its functions always “as required”.
  - **Reliability**: all functions performed on a system are always equally performed under equal constraints.

Security Aims

- ***Anonymity***
  - Anonymity of a user means that the user may use a resource or service without disclosing the user's identity. For example, electronic cash.
- ***Pseudonymity***
  - Pseudonymity of a user means that the user may use a resource or service without disclosing its user identity, but can still be accountable for that use.
- ***Unobservability***
  - ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used.

Formal System Development Path



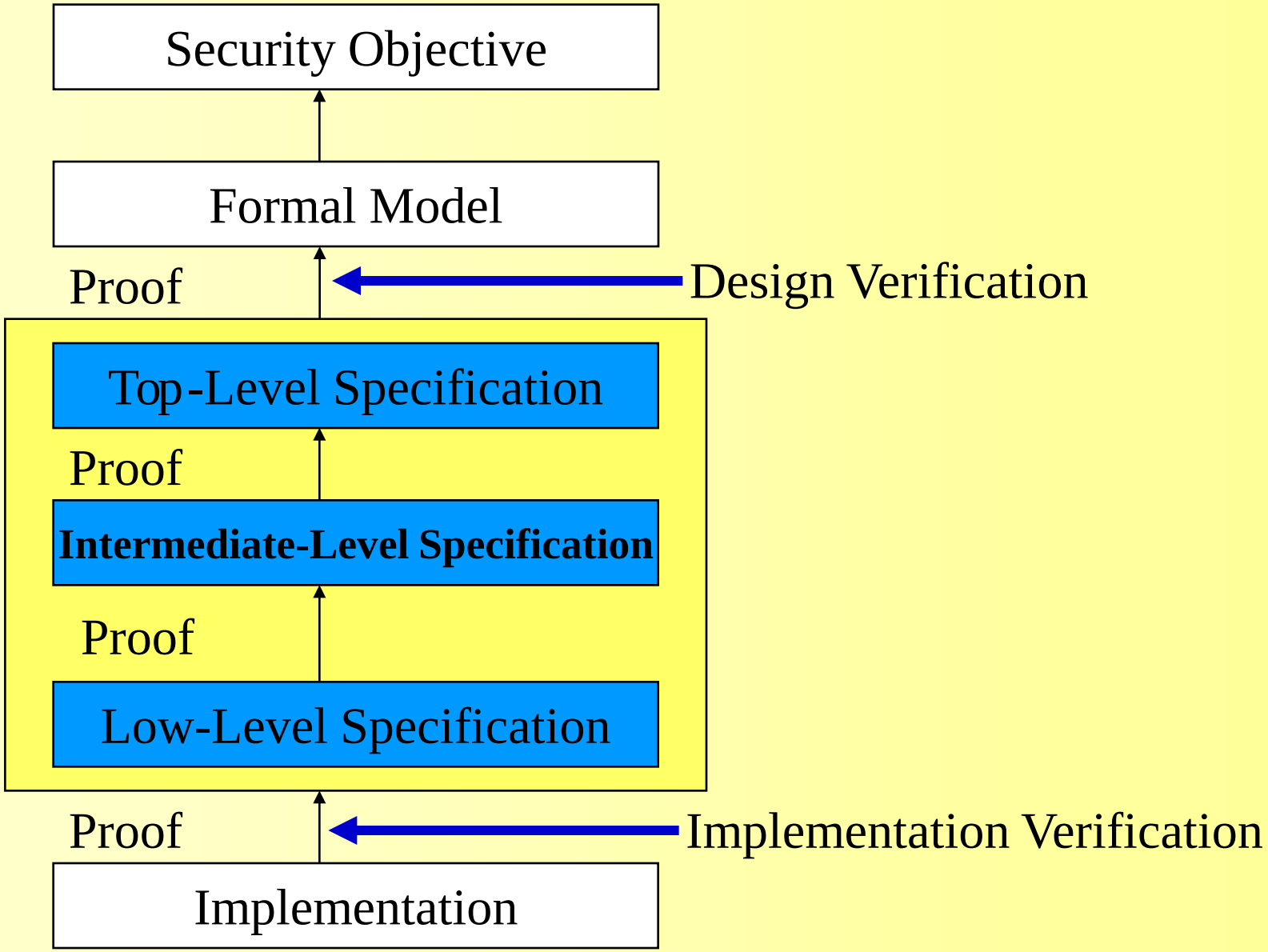
Security Models

- For the design of a secure system, the system´s security policy has to be defined.
- A security policy is a set of rules that regulate how an organisation manages, protects and distributes information.
- A (formal) security model is a (mathematically) precise statement of the security policy.

什么是安全模型?

- 系统的元素
  - 具有行为能力的主体;
  - 不具有行为能力的客体;
- 系统的操作行为
  - 可以执行的命令
- 对系统行为的约束方式
  - 对行为的控制策略

- 模型从抽象层次规定了系统行为和约束行为的方式。
- 模型往往用状态来表示
  - 系统行为所依赖的环境
  - 行为对系统产生的效果


Security Objective
Formal Model
Proof
Design Verification
Top-Level Specification
Proof
Intermediate-Level Specification
Proof
Low-Level Specification
Proof
Implementation Verification
Implementation

Proof the security model enforces the security policy

I. Definition of security-relevant state variables (subjects, objects, security attributes, access rights)

II. Definition of conditions for a secure state (invariants, security properties);

III. Definition of state transition function;

IV. Proof that the functions maintain the secure state;

V. Definition of the initial state;

VI. Proof that the initial state is secure;

M. Gasser, Building a secure computer system, van Nostrand Reinhold, 1988.

Formal system development path

- **Design verification:** proving the consistency between the formal specification of a system and a formal security model.
  - the specification conforms to the functions, invariants, and constraints of the model.
- Implementation verification: proving the consistency between the formal specification and its program implementation.
  - Complete formal implementation verification is not possible with today's technology. —— Simone Fischer-Hübner, IT Security and Privacy, Springer.
  - **Establishing correctness is a matter of belief, not proof.**

2. 访问控制矩阵模型

2.1 HRU 模型

- **A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. Proceedings of the 5th annual SIGOPS Conference, 1975.**

- **Michael A. Harrison and Walter L. Ruzzo (University of California, Berkeley), Jeffrey D. Ullman (Princeton University), Protection in Operating Systems, Communications of the ACM 19(8), pp461-471, Aug. 1976.**

2.1 HRU 模型

- 访问控制矩阵
  - 所有的主体的集合用$S=\{s_1,s_2,\ldots\ldots,s_m\}$ 表示
  - 所有的客体用$O=\{o_1,o_2,\ldots\ldots,o_n\}$表示
  - 所有的权限用R表示
  - 访问矩阵是以主体为行索引、以客体为列索引的矩阵的第i行第j列的矩阵元素$a[s_i,o_j] \subseteq R$ 表示主体s_i对客体o_j拥有的权限。

2.1 HRU 模型

	文件1	文件2	进程1	进程2
进程1	r, w, own	r	r, w, e, own	w
进程2	a	r, own	r	r, w, e, own

- 对于文件客体的r、w、a、own都比较清楚。
- 对于进程的读、写等在不同的系统中可能含义不同。
  - 读：“读”进程可以接收“被读”进程发送的消息。
  - 读：“读”进程可以读取“被读”进程的状态。

2.1 HRU 模型——系统的状态

- 系统的状态
  - 所有的内存、缓存、寄存器、设备的状态等。
- 系统的保护状态
  - 系统状态中涉及安全的子集。
  - 系统的保护状态可以用三元组**(S, O, A)**表示。
  - 假设 P 是所有的保护状态的集合， Q 是 P 的一个子集表示系统的合法状态。
  - 如果系统的安全状态在 Q 中，则系统是安全的；
  - 如果在 P － Q 中，则系统是不安全的。
  - **系统的保护就是将系统控制在状态集Q中。**

2.1 HRU 模型—— 状态的转换

- 当进程执行操作后，保护系统的状态会发生转换。
- 系统的初始状态设为$X_0=\{S_0,O_0,A_0\}$
- 当系统执行一系列操作t1,t2,…后，系统状态依次变成X1,X2,…。记为：

$$X_i \vdash_{t_i} X_{i+1}$$

- 当一个系统从状态x开始，经过一系列的操作后，转换到状态Y，可以记作： $X \vdash^{*} Y$

2.1 HRU 模型—— 基本命令

- **create subject s**

  事前条件$s \notin S$

  事后条件$S'=S \cup \{s\}$, $O'=O \cup \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S, y \in O$, $a'[x,y] = a[x,y]$

- **create object o**

  事前条件$o \notin O$

  事后条件$S'=S$, $O'=O \cup \{o\}$,

  $\forall x \in S'$, $a'[x,o] = \varphi$.

  $\forall x \in S', y \in O$, $a'[x,y] = a[x,y]$

- **destroy subject s**

  事前条件$s \in S$

  事后条件$S'=S \setminus \{s\}$, $O'=O \setminus \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S', y \in O'$, $a'[x,y] = a[x,y]$

- **destroy object o**

  事前条件$o \in O$

  事后条件$S'=S$, $O'=O \setminus \{o\}$,

  $\forall x \in S'$, $a'[x,o] = \varphi$.

  $\forall x \in S', y \in O'$, $a'[x,y] = a[x,y]$

2.1 HRU 模型 —— 基本命令

- **create subject s**

  事前条件$s \notin S$

  事后条件$S'=S \cup \{s\}$, $O'=O \cup \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S, y \in O$, $a'[x,y] = a[x,y]$

- **destroy subject s**

  事前条件$s \in S$

  事后条件$S'=S \setminus \{s\}$, $O'=O \setminus \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S'$, $y \in O'$, $a'[x,y] = a[x,y]$

- **create object o**

  事前条件$o \notin O$

  事后条件$S'=S$, $O'=O \cup \{o\}$,

	o_1	o_2	o_3	s_1	s_2
s_1					
s_2					

	o_1	o_2	o_3	s_1	s	s_2
s_1						
s						
s_2						

  $\forall x \in S'$, $a'[x,o] = \varphi$.

  $\forall x \in S'$, $y \in O'$, $a'[x,y] = a[x,y]$

2.1 HRU 模型 —— 基本命令

- **create subject s**

  事前条件$s \notin S$

  事后条件$S'=S \cup \{s\}$, $O'=O \cup \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S, y \in O$, $a'[x,y] = a[$

- **destroy subject s**

  事前条件$s \in S$

  事后条件$S'=S \setminus \{s\}$, $O'=O \setminus \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S', y \in O'$, $a'[x,y] = a[x,y]$

- **create object o**

  事前条件$o \notin O$

  事后条件$S'=S$, $O'=O \cup \{o\}$,

  $\forall x \in S'$, $a'[x,o] = \varphi$.

$\forall x \in S', y \in O'$, $a'[x,y] = a[x,y]$

φ

	o_1	o_2	o_3	s_1	s_2
s_1					
s_2					

	o_1	o_2	o_3	s_1	s	s_2
s_1						
s	φ	φ	φ	φ	φ	φ
s_2						

系统状态的转换 —— 基本命令

- **create subject s**

  事前条件$s \notin S$

  事后条件$S'=S \cup \{s\}$, $O'=O \cup \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S, y \in O$, $a'[x,y] = a[x,y]$

- **destroy subject s**

  事前条件$s \in S$

  事后条件$S'=S \setminus \{s\}$, $O'=O \setminus \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S'$, $y \in O'$, $a'[x,y] = a[x,y]$

- **create object o**

  事前条件$o \notin O$

  事后条件$S'=S$, $O'=O \cup \{o\}$,

  $\forall x \in S'$, $a'[x,o] = \varphi$.

	o_1	o_2	o_3	s_1	s_2
s_1					
s_2					

	o_1	o_2	o_3	s_1	s	s_2
s_1					φ	
s	φ	φ	φ	φ	φ	φ
s_2					φ	

$\forall x \in S'$, $y \in O'$, $a'[x,y] = a[x,y]$

2.1 HRU 模型 —— 基本命令

- **create subject s**

  事前条件$s \notin S$

  事后条件$S'=S \cup \{s\}$, $O'=O \cup \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S$, $y \in O$, $a'[x,y] = a[x,y]$

- **destroy subject s**

  事前条件$s \in S$

  事后条件$S'=S \setminus \{s\}$, $O'=O \setminus \{s\}$,

  $\forall y \in O'$, $a'[s,y] = \varphi$.

  $\forall x \in S'$, $a'[x,s] = \varphi$.

  $\forall x \in S'$, $y \in O'$, $a'[x,y] = a[x,y]$

- **create object o**

  事前条件$o \notin O$

  事后条件$S'=S$, $O'=O \cup \{o\}$,

  $\forall x \in S'$, $a'[x,o] = \varphi$.

  $\forall x \in S'$, $y \in O$, $a'[x,y] = a[x,y]$

	o_1	o_2	o_3	s_1	s_2
s_1					
s_2					

	o_1	o_2	o_3	s_1	s	s_2
s_1					φ	
s	φ	φ	φ	φ	φ	φ
s_2					φ	

2.1 HRU 模型 —— 基本命令

- **enter r into a[s,o]**

  事前条件$s \in S,\ o \in O$

  事后条件

  - $S'=S,\ O'=O,$
  - $a'[s,o] = a[s,o] \cup \{ r \}$
  - $\forall x \in S', y \in O', (x, y) \neq (s, o) \rightarrow a'[x,y] = a[x,y]$

- **delete r from a[s,o]**
- 事前条件$s \in S, o \in O$
- 事后条件

  $S'=S,\ O'=O,$

  $a'[s,o] = a[s,o] \setminus \{ r \}$

  $\forall x \in S', y \in O', (x, y) \neq (s, o) \rightarrow a'[x,y] = a[x,y]$

2.1 HRU 模型 —— 单步命令

- 系统命令

```
command  create·file(p, f)
        create object  f;
        enter own into a[p,f];
        enter r into a[p,f];
        enter w into a[p,f];
 end


 command spawn·process(p, q)
        create subject q;
        enter own into a[p,q];
        enter r into a[p,q];
        enter w into a[p,q];
        enter r into a[q,p];
        enter w into a[q,p];
 end
```

- 单步命令

系统命令中仅仅包含一条基本命令则称为单步命令。例如:

```
command make·own(p, f)
        enter own into a[p, f];
end
```

2.1 HRU 模型 —— 条件命令

- 条件命令

command (X_1, … X_k)
    if r_1 in (X_{s1}, X_{o1})
    r_2 in (X_{s2}, X_{o2})
    r_m in (X_{sm}, X_{om})
then
    op_1;
    op_2;
    ……
    op_n;
end

- 例

command grant·read ·file
    if own in a[p,f] then
        enter r into a[q,f];
end

2.1 HRU 模型

- 什么样是计算机系统的安全？
  - 不希望的事情不会发生。
  - 主体s 对客体 o不应该具有权限r。
  - 权限 r 不会出现在访问控制矩阵的元素a [s, o]中。
  - “我们不希望的事情是否会发生？” **可判定吗？**

2.1 HRU 模型 —— 可靠性定义

① 当一个基本权限r被加入到原来不存在r的一个访问控制矩阵元素时，这个权限被称为是泄密的。

② 如果一个系统从不泄露权限r，这个系统（包括初始状态s_0）被称为对于权限r是可靠的。如果一个系统会泄露r，则称这个系统对于权限r是不可靠的。

2.1 HRU 模型 —— 两个定理

① 对于给定的单步命令系统，初始状态为s_0，存在一个算法可以判定，对于一个一般意义的权限r是否是可靠的。

② 在一个受保护的系统中，一个给定的状态对于一个基本权限是否是可靠的是不可判定的。

证明也可见：Matt Bishop， 计算机安全学——安全的艺术与科学，电子工业出版社，2005年5月。

2.2 取予(take-grant)模型

1. A K. Jones, R. J. Lipton, and L. Snyder. A Linear Time Algorithm for Deciding Security. In *Proc. 17th Symposium on Foundations of Computer Science*, pages 33–41, Houston, Texas,1976.
2. R. J. Lipton, L. Snyder (*Yale Umverstty),* **A Linear Time Algorithm for Deciding Subject Security,** Journal of the Association for Computing Machinery, Vol.24, No 3, July 1977, pp 455-464.

可以通过建立特定的访问控制规则，可以验证系统是否可靠的。

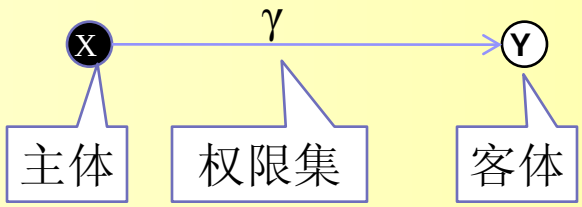
2.2 取予(take-grant)模型

A K. Jones, R. J. Lipton, and L. Snyder. A Linear Time Algorithm for Deciding Security. In *Proc. 17th Symposium on Foundations of Computer Science*, pages 33–41, Houston, Texas,1976.

R. J. Lipton, L. Snyder (*Yale Umverstty),* **A Linear Time Algorithm for Deciding Subject Security,** Journal of the Association for Computing Machinery, Vol.24, No 3, July 1977, pp 455-464.

可以通过建立特定的访问控制规则，可以验证系统是否可靠的。

- 主体对客体具有的权限。



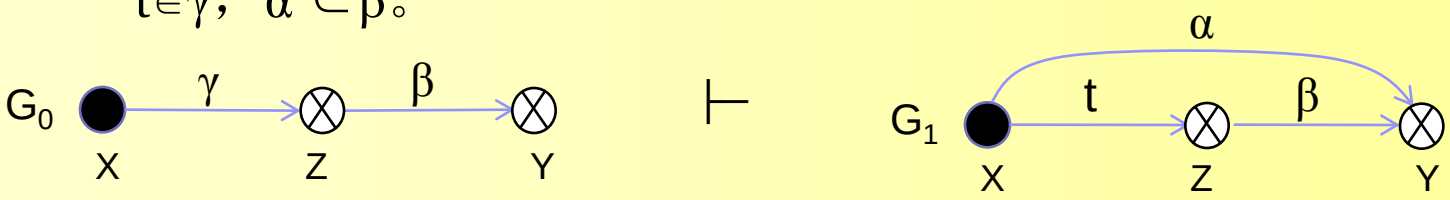
主体	客体	主体和客体
●	○	⊗

t ：取得(take)；　　g：赋予（grant）。

1.2取予(take-grant)模型

- Take规则

t∈γ，α ∈β。

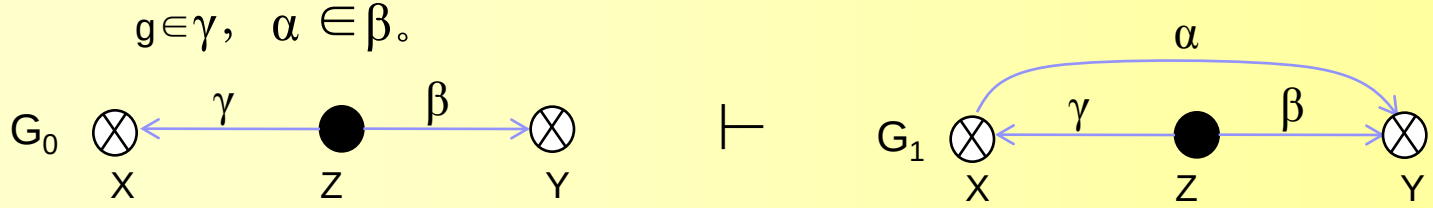


表示成$G_0 \vdash G_1$

“x从z中取得了y的权限α” 。

- Grant规则

g∈γ，α ∈β。

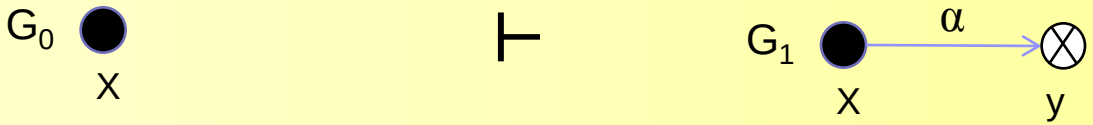


表示成$G_0 \vdash G_1$

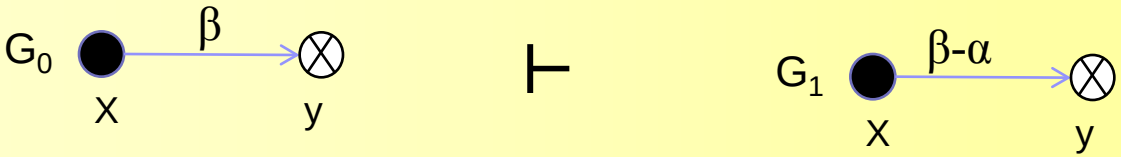
“z赋予x到y的权限α” 。

2.2取予(take-grant)模型

- Create规则：设x是保护图G_0的一个主体，并且α ⊂R。Create规则定义了一个产生新图的G_1方法（1）加入一个新的节点y;(2)加入一条从x到y的标记为α的边。这条规则称为：x通过权限α创建了节点y。

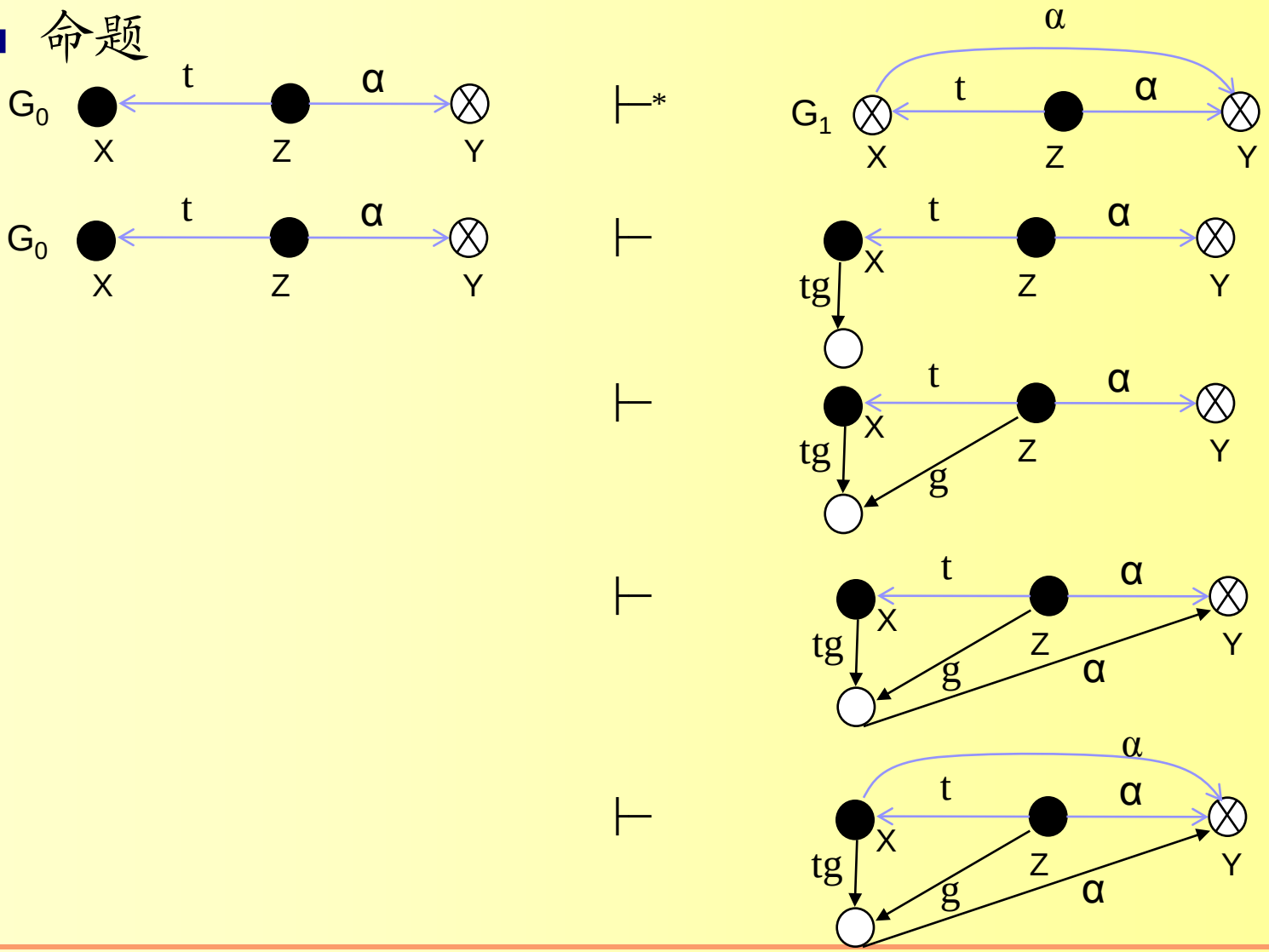


- Remove规则：设x、y是保护图G_0的两个节点，并且x是主体。从x到y存在一条标记为β的边，并且α ⊆ β。Remove规则定义了一个产生新图的G_1方法：新图中边的标记β变为β－α，如果结果为空集，则边要被去掉。

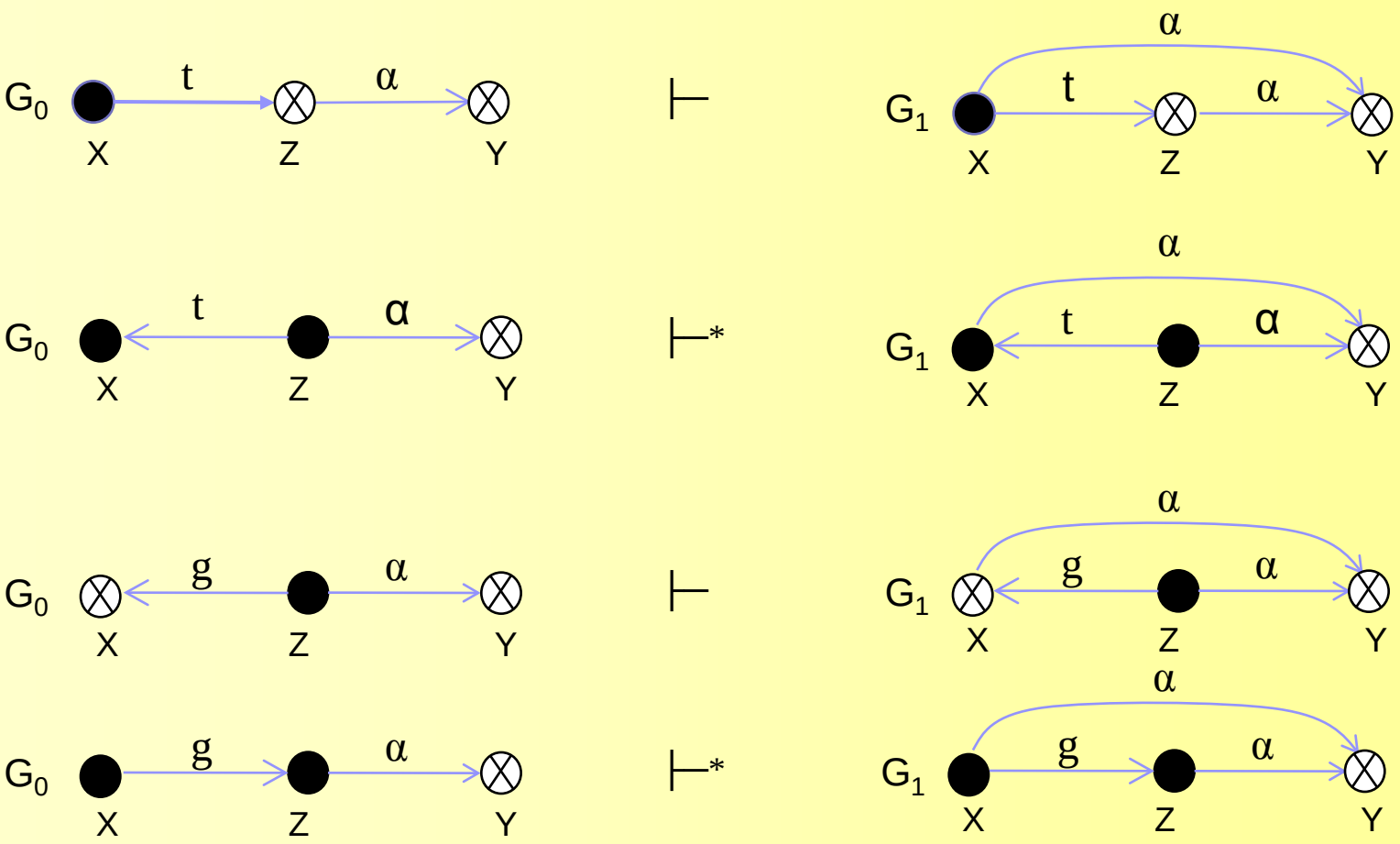


2.2 取予(take-grant)模型

- 命题



2.2取予(take-grant)模型)—— 主体对称性



2.2取予(take-grant)模型 —— 共享模型

- x,y是系统中两个不同实体， 按照系统的规则x能共享y的权限α？

- 设G_0是一个保护图，x,y是G_0中两个不同实体节点， α是权限集合R的一个子集。谓词can·share(α,x,y)的意义是:

  （1） 存在一个保护图的序列G_1，G_2，……,G_n，

  ρ_i, i=1,……， n，利用规则$\rho_{i可以}$实现$G_{i-1} \vdash^* G_i$，

  i=1,……， n;

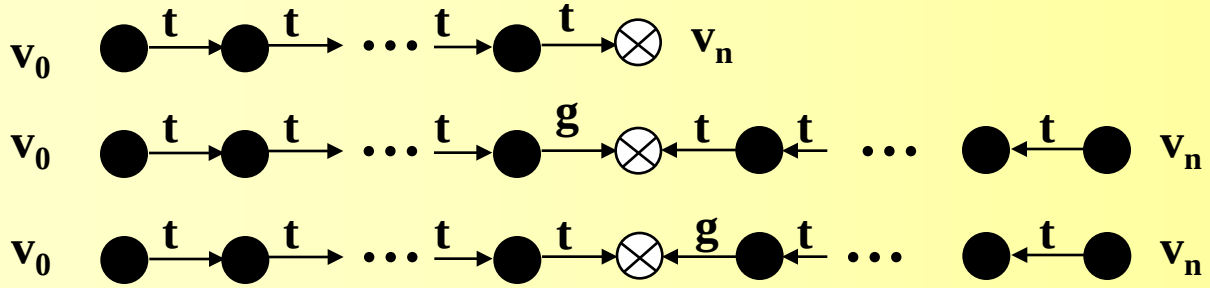
  （2） G_n中存在包含权限子集α的x到y的边;

2.2 取予(take-grant)模型

- 对称性：如果连接x和y的tg路径中的节点都是主体，则Take和Grant规则是对称的。
- 定义：一个岛屿是最大的tg相连的、仅有主体的子图。
- 共享：因为一个岛屿是仅有tg相连的最大子图，可以直接证明任何节点所拥有的任何权限都可以被岛屿中的其他节点所共享。

2.2 取予(take-grant)模型

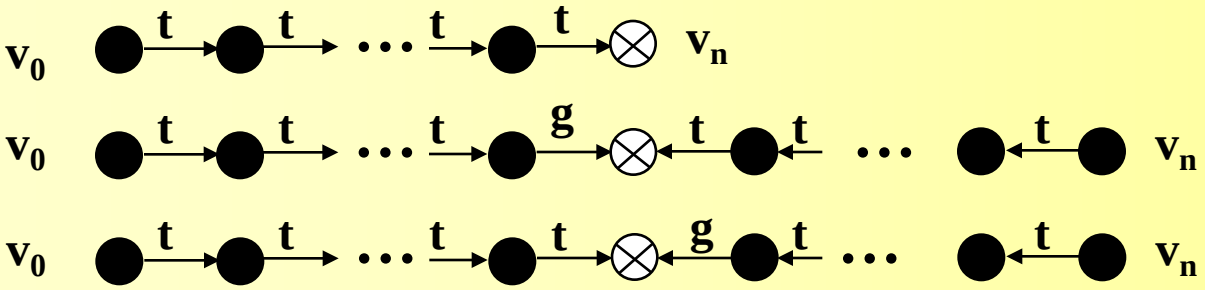
- **岛屿之间的权限转移：**
  - 一个岛屿中的一个主体可以从另一个岛屿中的节点取得权限；或者
  - 一个主体可以赋予权限给一个中间客体，而另一个岛屿中的某个主体可以从该主体申取走这个权限。
- 基于岛屿之间的权限转移的原理和Take和Grant的对称性，可以归纳出岛屿之间权限转移的路径特征。
- 定义：桥是指二个连接端点主体v_0和v_n的tg路径，并且与该路径相关联的字串在集合 $\{\vec{t}^*,\ \vec{t}^*\vec{g}\vec{t}^*,\ \vec{t}^*\vec{g}\overleftarrow{t}^*\}$ 中。



桥的端点都是主体，所以权限可以从桥的一端转移到另一端。

2.2取予(take-grant)模型

- 定义：桥是指二个连接端点主体v_0和v_n的tg路径，并且与该路径相关联的字串在集合 $\{\vec{t}^*,\ \vec{t}^*\vec{g}\overleftarrow{t}^*,\ \vec{t}^*\overleftarrow{g}\overleftarrow{t}^*\}$ 中。

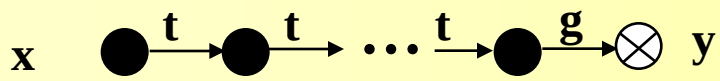


桥的端点都是主体，所以权限可以从桥的一端转移到另一端。

- 定理 谓词subject·can·share(α, x, y, G_0)为真的充要条件是:
  - (1) x, y是主体并且存在一条由x到y的边标记为α；或者
  - (2) 如下条件同时成立:
    - G_0中存在主体s，且存在s到y的标记为α的边;
    - 存在岛屿$I_1, I_2, \ldots, I_n$， x在I_1中，s在I_n中，并且I_j到I_{j+1}（$1 \leqslant j<n$）都存在一座桥。

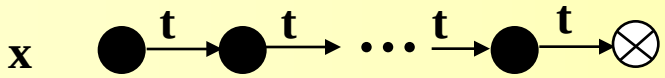
2.2 取予(take-grant)模型

- 当桥的两端是客体时，就不再具有对称性，共享定理不再成立。
- 定义 如果x是一个主体，并且在x，y之间存在一条tg路径具有关联字串定义在集中 $\{\vec{t}*\vec{g}\}\cup\{v\}$ ，则节点x可以**初始扩展到y**。 其中v表示空串。



如果x把它拥有的一个权限**赋予y**，则x可以**初始扩展到y**。
如果y是客体，则x无法获得y拥有的权限。——不对称性！

- 定义 如果x是一个主体，并且在x，y之间存在一条tg路径具有关联字串定义在集中$\{\vec{t}*\}\cup\{v\}$，则节点x可以**最终扩展到y**。其中v表示空串。



如果x从y拥有的权限中**取得任何权限**，则x可以**最终扩展到y**。
如果y是客体，则y无法获得x拥有的权限。——不对称性！

2.2 取予(take-grant)模型

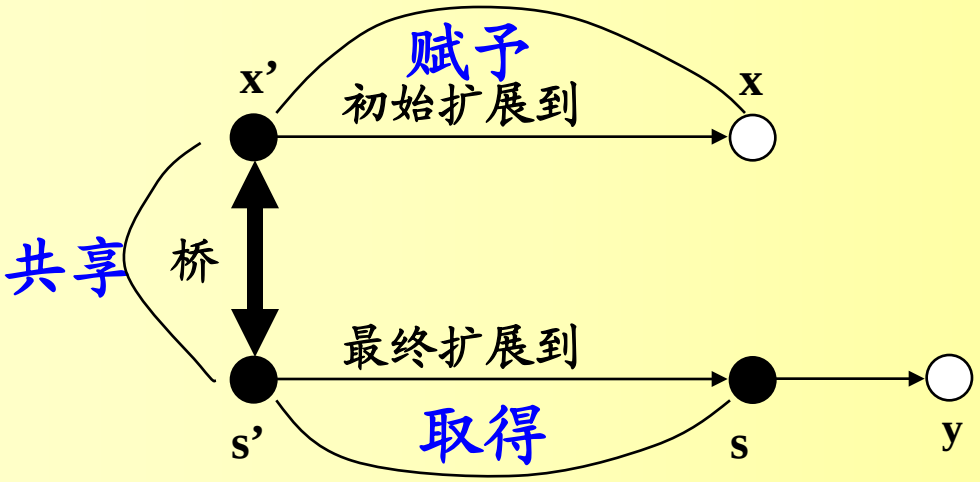
- 定理 谓词**can·share**(α, x, y, G_0)为真的充要条件是：

(1) 存在一条由x到y的标记为α的边；

或者

(2) 如下条件同时成立:

  - G_0中存在节点s，且存在s到y的标记为α的边;
  - 存在一个主体节点x'，且x'=x 或者 x'初始扩展到x;
  - 存在一个主体节点s'，且s'=x 或者 s'最终扩展到s;
  - 存在岛屿$I_1, I_2, \ldots, I_n$， 且x'在I_1中，s'在I_n中，并且I_j到I_{j+1}（$1 \leqslant j<n$）都存在一座桥。



2.2取予(take-grant)模型 —— 共享模型

- 定理：存在一个复杂度O(|V|+|E|)的判定算法，判定谓词can·share是否为真。
- 是否存在一个谓词来表达系统的“安全状态”？
- 在给定的安全状态下，什么样的行为规则可以保障系统可以保持处于安全状态？

- 偷窃模型(steal mode)
  - 设G_0是一个保护图，x,y是G_0中两个不同实体节点，α是权限集合R的一个子集。谓词can·steal(α,x,y)的意义是:

（1） 不存在包含权限子集α的x到y的边;

（2） 存在一个保护图的序列G_1，G_2，……,G_n使得以下条件同时成立:

- G_n中存在包含权限子集α的x到y的边;
- 存在一个状态转换规则序列： ρ_i, i=1,……，n，使得利用规则ρ_i实现 $G_{i-1} \vdash G_i$， i=1,……，n。
- 对于G_{i-1}中的任意节点v和w， $1 \leqslant i<n$，如果在G_0中存在v到y的包含权限α的边，则不是“v将对于y的权限α授予w”形式的规则。 【注：这个条件是要求任何对y拥有α权限的节点不能将这个权限授予其它节点】

- 偷窃模型(steal mode)
  - 设G_0是一个保护图，x,y是G_0中两个不同实体节点， α是权限集合R的一个子集。谓词can·steal(α,x,y)的意义是:

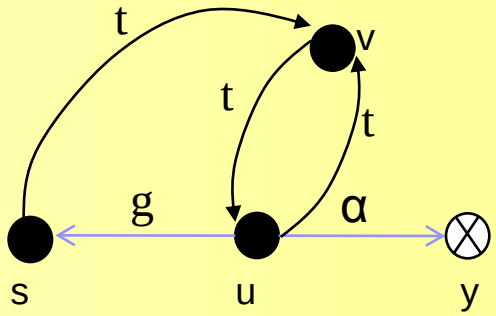
（1） 不存在包含权限子集α的x到y的边;

（2） 存在一个保护图的序列G_1，G_2，……,G_n使得以下条件同时成立:

- G_n中存在包含权限子集α的x到y的边;
- 存在一个状态转换规则序列： ρ_i, i=1,……，n，使得利用规则ρ_i实现 $G_{i-1} \vdash G_i$， i=1,……，n。
- 对于G_{i-1}中的任意节点v和w， $1 \leqslant i<n$，如果在G_0中存在v到y的包含权限α的边，则不是“v将对于y的权限α授予w”形式的规则。 【注：这个条件是要求任何对y拥有α权限的节点不能将这个权限授予其它节点】

- 例如：一个教授拥有学生的成绩单文件，他不会对这个文件的访问权限授予任何其它的主体。 问题是：这个系统中的其它主体可以通过系统的规则获得（窃取）这个文件的访问权限吗?

- u不希望将对y的权限α授予他人;
- u 将其它的权限(对v的t权限)授予s;
- 这时s可以从v取得对u的t权限;
- 这样s就可以取得u的对w的α权限了。

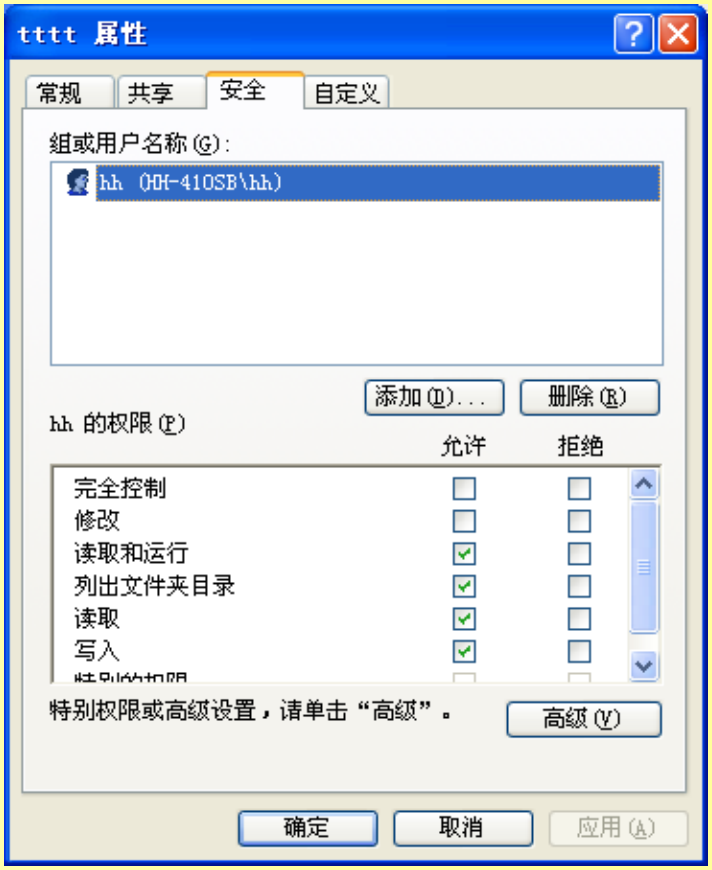


(1) 访问控制抽象 — 权能和访问控制表机制

- **权能模式**
  - 对应访问控制矩阵中的行结构。
  - 面向门票的模式，以主体为中心，阐述每个主体能够访问的所有客体及其访问的方式。
  - 【注意】*现代的属性证书*。
- **访问控制表模式**
  - 与访问控制矩阵中的列结构相对应。
  - 以客体为中心，确定能够访问该客体的所有主体的名单集合。
  - 【例】Windows 的NTFS的权限管理方式。

(1) 访问控制抽象 — 权能和访问控制表机制

- **权能模式**
  - 对应访问控制矩阵中的行结构。
  - 面向门票的模式，以主体为中心，阐述每个主体能够访问的所有客体及其访问的方式。
  - 【注意】*现代的属性证书*。
- **访问控制表模式**
  - 与访问控制矩阵中的列结构相对应。
  - 以客体为中心，确定能够访问该客体的所有主体的名单集合。
  - 【例】Windows 的NTFS的权限管理方式。



取予模型 —— 安全性

- 访问控制矩阵只是表达了主体与客体的权限关系，这样的关系是否满足管理者的要求？如何进行控制能够继续保持满足管理者的要求？
- 是否存在一个谓词来表达系统的“安全状态”？

  安全性需要进行抽象，怎样表达安全目标。
- 在给定的安全状态下，什么样的行为规则可以保障系统可以保持处于安全状态？

  需要研究行为与保护状态的转换的关系，如何对行为进行控制。

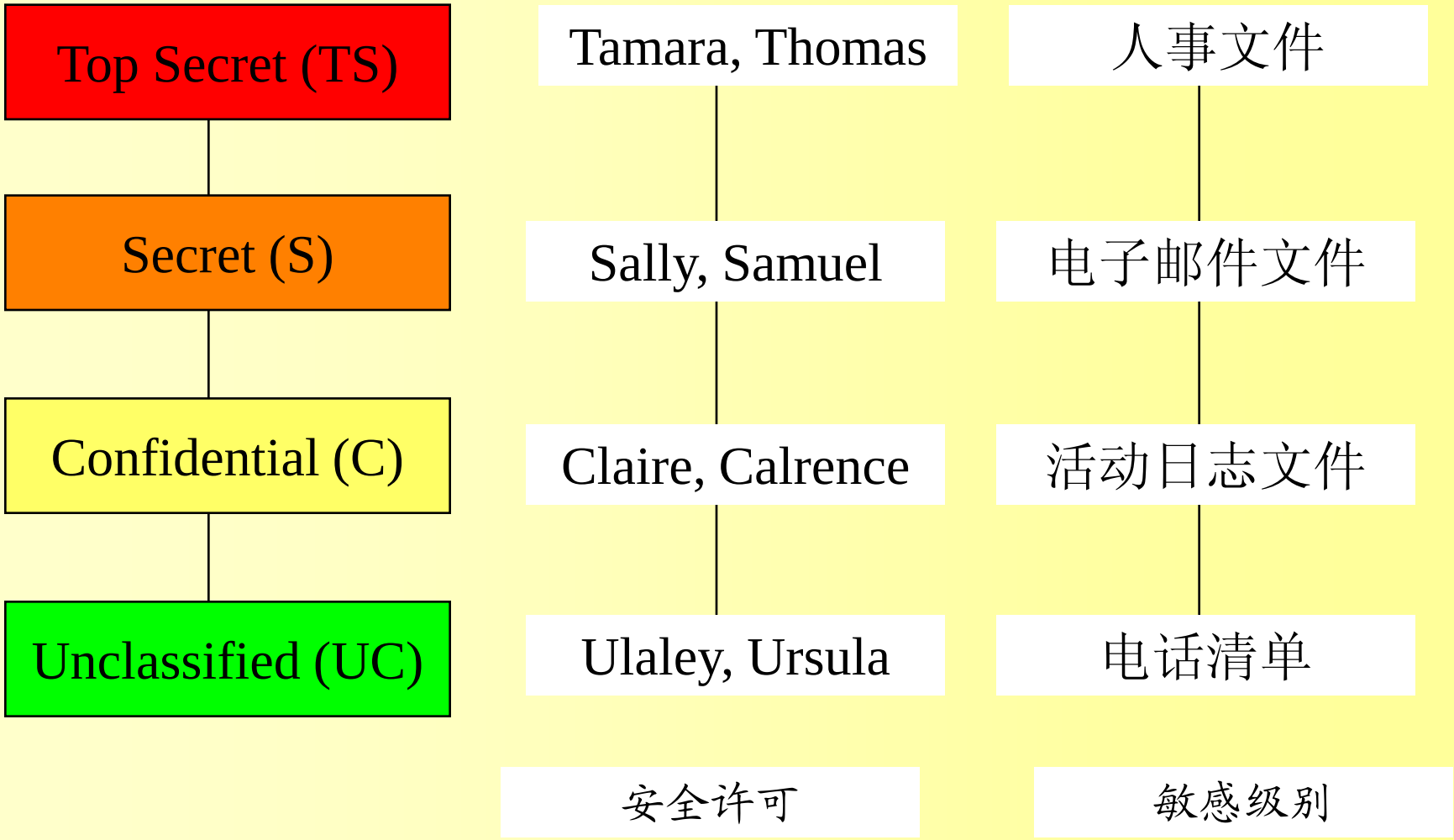
3. Bell-LaPadula 模型

Bell-LaPadula 模型对应军事类型的安全密级分类。

该模型影响了许多其他模型的发展，甚至很大程度上影响了计算机安全技术的发展。

3.1 非形式化描述

- 最简单的保密性分类形式是按照线性(全)排列的安全等级。
- 每一个主体都有一个安全许可(Clearance)。在下图中，Claire的安全许可是C(保密)，Thomas的安全许可是TS(顶级机密)。
- 每个客体都有一个敏感等级，电子邮件文件的敏感等级是S(秘密)，电话清单文件的敏感等级为UC(公开)。
- 当我们同时指主体的许可和客体的密级时，用术语“密级”。Bell－Lapadula安全模型的目的是要防止主体读取安全密级比它的安全许可更高的客体。


Top Secret (TS)
Secret (S)
Confidential (C)
Unclassified (UC)
Tamara, Thomas
Sally, Samuel
Claire, Calrence
Ulaley, Ursula
人事文件
电子邮件文件
活动日志文件
电话清单
安全许可
敏感级别

强制访问控制与自主访问控制

- 自主访问控制
  - 资源的拥有者(创建者)作出决定有哪些主体可以访问这个资源。
  - 由访问控制矩阵表示。
- 强制访问控制
  - 由系统管理员统一制定的策略。

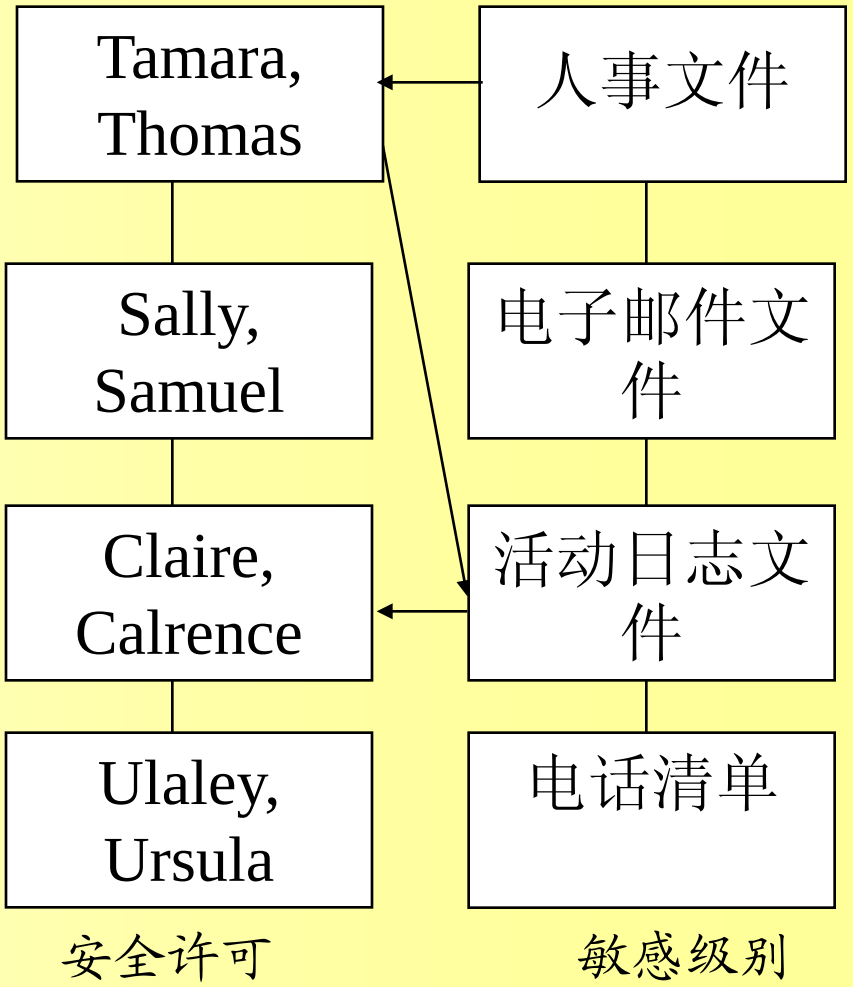
- Bell—LaPadula安全模型中结合了强制型访问控制和自主型访问控制。
- s对o有自主的读(写)访问权
  - 表示与自主访问控制部分相对应的s和o的访问控制矩阵条目中包含一项读(写)的权限。
  - 换句话说，如果没有强制型控制，s就可以读(写)o。

- 设L(s)=l_s是主体s的安全许可，并设L(o)= l_o 是客体o的敏感等级。对于所有安全密级l_i ，i=0，…，k-1，有l_i < l_{i+1} 。
- **简单安全条件** S可以读O，当且仅当l_o < l_s，且s对o具有自主型读权限。
- 例如，在上图中，Claire和Clarence不能读人事文件，但Tamara和Sally可以读活动日志文件(而且，实际上根据Tamara的安全许可，她可以读任何文件在此假设自主型访问控制允许Tamara和Sally的访问)。

- 如果Tamara(Top Secret)决定将人事文件的内容复制到活动日志文件里，并设置适当的自主访问权限，那么Claire(Confidential)就可以读这些人事文件了。这样，Claire可能会读取到具有更高安全等级的文件。

- ***-属性(星号属性** s可以写o，当且仅当 $l_s < l_o$ 。且s对o具有自主型写权限。

- 这时因为活动日志文件的安全密级为C，Tamara的安全许可为TS，所以她不能写活动日志文件。



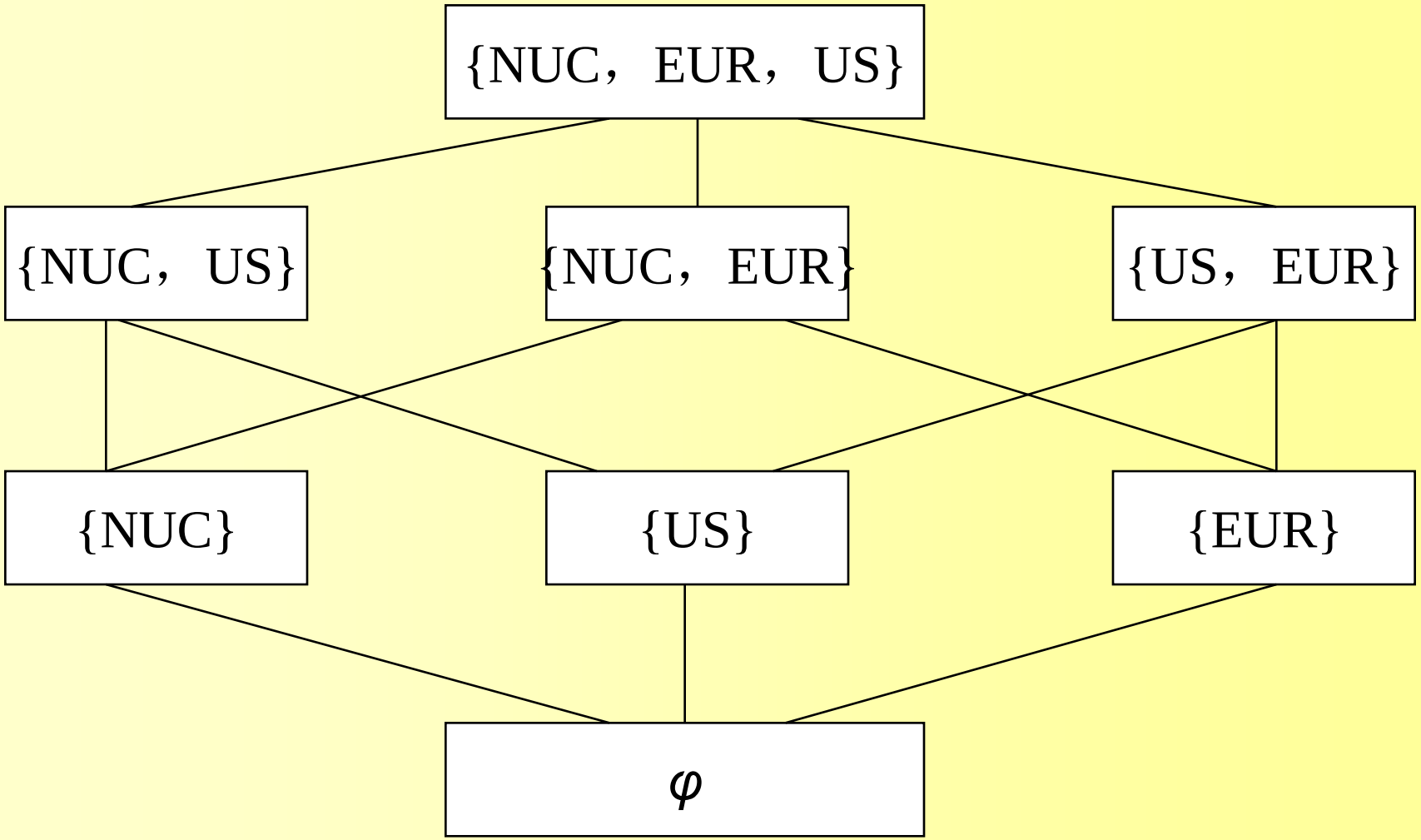
类别

- 通过给每个安全密级增加一套类别，每种类别都描述一种信息，可以将模型进行扩展。属于多个类别的客体拥有所有属于这些类别的信息。
- 这些类别来自于“需要知道’’原则，它规定，除非主体为了完成某些功能而需要读取客体，否则就不能读取这些客体。某人可以访问的类别集合就是类别集合的幂集。
- 例如，如果类别是NUC，EUR和US，那么某人可以访问的类别集合就是以下集合之一：

ϕ(空集)，{NUC}，{EUR}，{US}，

{NUC，EUR}　{EUR，US}　{NUC，Us}　{NUC，EUR，US}

这些类别集合在操作⊆(子集关系)下形成一个格。


{NUC，EUR，US}
{NUC，US}
{NUC，EUR}
{US，EUR}
{NUC}
{US}
{EUR}
φ

- 每个安全级别和类别形成一个安全等级。
- 主体在某安全等级上有安全许可(或归于此安全等级、属于此安全等级)
- 客体处于安全等级的级别(或者属于某个安全等级)。
- 例如，William可能归于等级(SECRET，{EUR})，George归于等级(TOP SECRET，{NUC，US})。一个文档可能会被归于等级(CONFIDENTIAL，{EUR})。

- 类别基于‘‘需要知道’’原则，所以预先假设对类别集合{NUC, US}有访问权的人没有必要访问类别EUR里元素。因此，即便该主体的安全许可高于客体的安全密级，读访问也应该被拒绝。
- 定义一种新的关系来体现安全密级和类别集合的结合。定义 dom (dominates支配)关系。
- 定义5.1 安全等级(L, C)支配安全等级(L', C'), 当且仅当L' ≤ L , C' ⊆ C。
- 关系为安全等级集合上引入了一个格。

- 简单安全条件　S可以读O，当且仅当S dom O，且S对O具有自主型读访问权限。
- *属性　S可以写O，当且仅当O dom S且S对O具有自主写权限。
- 基本安全定理　设系统Σ的某一个初始安全状态为σ_0，T是状态转换的集合。如果T的每个元素都遵守简单安全条件和*－属性，那么对于每个$i \geqslant 0$，状态σ_i都是安全的。

2.2 Bell La Padula 的形式化描述

General Systems

- $S \subseteq X \times Y$
  - The system S is a relation on the abstract sets X and Y.
- $S: X \rightarrow Y$
  - S is a function from X to Y
  - The elements of X: inputs
  - The elements of Y: outputs
  - S expresses a functional input-output relationship.

Example

- Consider a saving account in a bank which compounds interest quarterly
- $b_k = (b_{k-1} + p_k)\cdot(1 + i_k)$ (1.1)
  - b_k : the balance after the computation of interest at the end of the k-th quarter.
  - p_k : the net transaction in the account during the k-th quarter.
  - i_k : the quarterly interest rate at the end of the k-th quarter.

Example (continues)

- A seven-year history of such a savings account is represented by a system
- $S(b_0) \subseteq P \times I \times B$
  - b_0 : the initial balance in the account
  - $P = R^{28}$: the twenty-eight transactions
  - $I = R^{28}$: the twenty-eight quarterly interest rates
  - $B = R^{28}$: the twenty-eight successive balances
- $(p,i,b) \in S(b_0)$ iff (1.1) hold for every k from 1 to 28

Secure Computer Systems

- Problems of security
  - How to guarantee that unauthorized access(by a process) to information(file,program, data) does not occur.

Foundations of a Mathematical Model

- 模型的元素

集合	元素	语义
S	{S1,S2,・・・,Sn}	主体： 进程
O	{O1,O2,・・・,Om}	客体：数据、文件、程序、设备等。
C	{C1,C2,・・・,Cq} {C1 > C2 > ・・・ > Cq	级别：主体的安全许可，客体的敏感级别。
K	{K1,K2,・・・, Kr }	类别：访问权限范围。

Elements of the Model(2)

A	**{r,w,e,a,c}**	访问属性： read, write, append, execute, and control
RA	**{g,r,c,d}**	**请求元素：** g: get, give r: release, rescind c: change, create d: delete
R	$S^+ \times RA \times S^+ \times O \times X$ 其中： $S^+ = S \cup \{\phi\}$ $X = A \cup \{\phi\} \cup F$; 一个请求的元素记为 R_k	请求： inputs, commands, requests for access to objects by subjects
D	**{yes, no, error, ?}** D的元素记为D_m	决定

Elements of the Model(2)

F	$C^S \times C^O \times (PK)^S \times (PK)^O$ an arbitrary element of F is written f = (f1,f2,f3,f4)	classification/need-to-know vectors; f1: subject-classification function f2: object-classification function f3: subject-category function f4: object-category function
X	R^T an arbitrary element of X is written x	request sequences
Y	D^T an arbitrary element of Y is written y	decision sequences

Elements of the Model(4)

M	{M1,M2, • • • , Mc}, $c = (2^5)^{n \cdot m}$; an element of M, say M_k, is an n × m matrix with entries from PA; the (i,j)-entry of M_k shows S_i's access attributes relative to 0_j	access matrices
V	P(S × O × A) × M × F an arbitrary element of V is written v	states
Z	V^T an arbitrary element of z is written z; z^t ∈ z is the t-th state in the state sequence z	state sequences

States of the System

- **A state v ∈ V is a 3-tuple (b, M, f) where**

  (1) b

  - **b ∈ P(S × O × A), indicating a current access set which representing access mode.**
  - **a current access is represent by a triple: (subject, object, access-attribute)**
  - **the current access set b is a set of such triples representing all current accesses.**

（2） M

- **m ∈ M, indicating the entries of the access matrix in the state v;**

Object / Subject		O_j		
S_i		$m_{ij} \in$ **PA**		

（3） f

- **$f \in F$, indicating the clearance level of all subjects, the classification level of all objects, and the categories associated with each subject and object in the state v.**
  - **Clearance or classification usually denoted by unclassified, confidential, secret, top secret**
  - **Category usually denoted by Nuclear, NATO, Crypto, …**
  - **A total security designation is pair:**

    (classification, set of category)

- (class 1, category-set1) dominates (class 2, category-set2)

  if and only if

  Class1 is greater than or equal to class2 and

  category-set1 includes category-set2 as a subset.

State-Transition Relation

- $W \subseteq R \times D \times V \times V$
- The system $\sum(R,D,W,z_0) \subseteq X \times Y \times Z$ is defined by
  - $(x,y,z) \in \sum(R,D,W,z_0)$ if and only if

    $(x_t, y_t, z_t, z_{t-1}) \in W$， for each $t \in T$
  - z_0 is a specified initial state usually of the form (ϕ,M,f).

- 这里W定义了所有可能的状态转换， (x_t, y_t, z_t, z_{t-1})。
- 一个实际的过程不会遍历W中所有的状态转换，而是W的一个子集。
- 系统的参数W，限定了系统中所有的状态变换都在W中。
- 以W为参数的系统 $\sum(R, D, W, z_0)$ ，应该有许多的实例。

SECURITY CONDITION

$(S,O,\underline{x}) \in S \times O \times A$ satisfies the security condition relative to f (SC rel f) iff

(i) $\underline{x} = \underline{e}$ or $\underline{x} = \underline{a}$ or $\underline{x} = \underline{c}$, or

(ii) $(\underline{x} = \underline{r}$ or $\underline{x} = \underline{w})$ and

$f_1(S) \geq f_2(O)$ and

$f_3(S) \supseteq f_4(O))$.

SECURITY CONDITION

- A state $v = (b,M,f) \in V$ is a secure state iff each $(S,O,x) \in b$ satisfies SC rel f.

  A state v is a compromise state (compromise) iff it is not a secure state.

- z is a secure state sequence iff z_t is a secure state for each $t \in T$.

  A state sequence $z \in Z$ has a compromise iff z_t is a compromise for some $t \in T$.

- If $(x,y,z) \in \sum(R,D,W,z_0)$, we call it an a ppearance of the system.

  $\sum(R,D,W,z_0)$ is a s e cure system iff every appearance of $\sum(R,D,W,z_0)$ is secure.

  $\sum(R,D,W,z_0)$ has a compromise iff some appearance of $\sum(R,D,W,z_0)$ has a compromise.

*-PROPERTY

- Let b(s:x,y, ... ,z) denote the set

  { o: o ∈ O and [(s,o,x) ∈ b or (s,o,y) ∈ b or ••• or (s,o,z) ∈ b] }.

- A state v = (b,M,f) ∈ V satisfies *-property iff for each s ∈ S the following proposition is true:

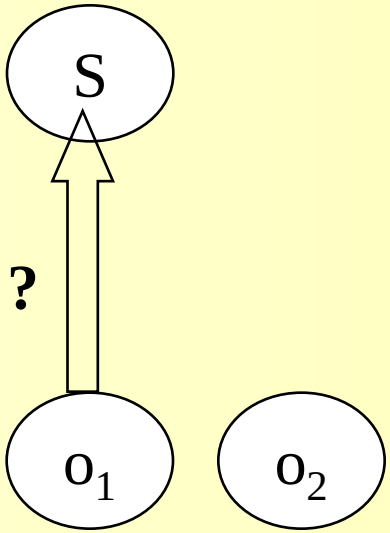
  $[b(s:w,a) \neq \phi$ and $b(s:r,w) \neq \varphi]$ implies

  $[f_2(O_1) \geq f_2(O_2)$ and $f_4(O_1) \supseteq f_4(O_2)$,

  for all O_1 in b(s:w,a), O_2 in b(s:r,w)]

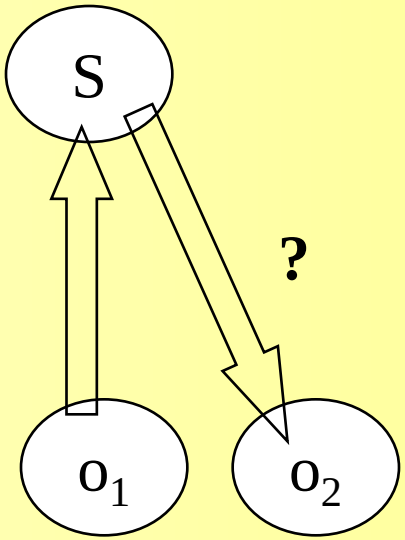
- A state v violates *-property iff v does not satisfy *-property.

- 一次请求就是建立一个通道。
- 这个通道建立后是否造成泄漏？



一个读通道如果满足

SC ref F，则批准请求读权限行为是安全的。



一个写通道如果满足

*-property，则批准写请求行为是安全的。

*-PROPERTY

- A state sequence $z \in Z$ satisfies *-property iff z_t satisfies *-property for each $t \in T$.
- $(x,y,z) \in \Sigma(R,D,W,z_0)$ satisfies *-property iff z satisfies *-property.
- S(R,D,W,z0) satisfies *-property iff every appearance of $\Sigma(R,D,W,z_0)$ satisfies *-property.

Discretionary security property

- A state v = (b,M,f) ∈ V satisfies ds-property if
- A state satisfies the ds-property provided that every current access is permitted bythe current access matrix M:

$$(S_i, O_j, x) \in b \Rightarrow x \in M_{ij}$$

Rules

- A rule is a function r: $R \times V \rightarrow D \times V$.

  given a request and a state, a rule decides a response and a state change.

  一个规则实际上可以定义了一个有限状态机。

Rules

- A rule r is security-preserving iff the proposition

  [[r(Rk,v) = (Dm,v*) and v is secure] implies [v* is secure]]

  holds for all elements (Rk,v) ∈ R × V.

- A rule r is *-property-preserving iff the proposition

  [[r(Rk,v) = (Dm,v*) and v satisfies *-property] implies [v* satisfies *-property]]

  holds for all elements (Rk,v) ∈ R × V.

Rules

- Let $\omega = \{\rho_1, \rho_2, \cdots, \rho_s\}$ be a set of rules relative to R, D, and V.
- The relation $W(\omega)$ is defined by:
  - $(Rk, ?, v, v) \in W(\omega)$ iff $\rho_i(Rk, v) = (?, v)$ for each i, $1 \leqslant i \leqslant s$;
  - $(Rk, error, v, v) \in W(\omega)$ iff there exist i_1, i_2, $1 \leqslant i_1 \leqslant i_2 \leqslant s$ such that $\rho_{i1}(Rk, v) = (?, v^{**})$ and $\rho_{i2}(Rk, v) = (?, v^{*})$ for some v^{*}, $v^{**} \in V$。
  - $(Rk, Dm, v*, v) \in W(\omega)$ iff there exist a unique i, $1 \leqslant i \leqslant s$, such that $\rho_i(Rk, v) = (Dm, v^{*})$ for some $v^{*} \in V$。

Action

- $(Ri,Dj,v^*,v) \in R \times D \times V \times V$ is an action of $\Sigma(R,D,W,z_0)$ iff
  - there is an appearance (x,y,z) of $\Sigma(R,D,W,z_0)$
  - and some $t \in T$ such that

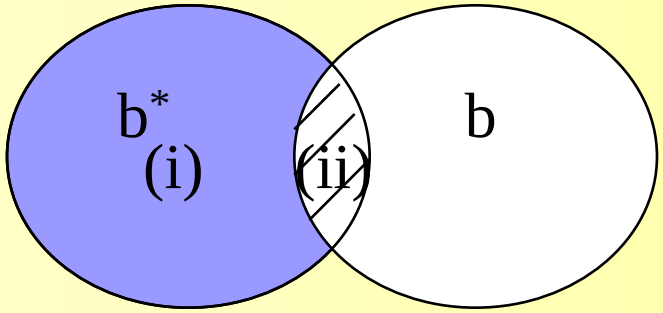
$$(R_i,D_j,v^*,v) = (x_t,y_t,z_t, z_{t-1})$$

Theorems

- Theorem 1 $\Sigma(R,D,W,z_0)$ is secure for any secure state z_0 iff W satisfies the following conditions for every action (R_i, D_j, (b*,M*,f*), (b,M,f)):

  (i) every (S,O,x) ∈ b*- b satisfies SC rel f*;

  (ii) every (S,O,x) ∈ b which does not satisfy SC rel f* is not in b*.



- Theorem 2 Letω be a set of security-preserving rules and z_0 a secure state. Then Σ (R,D,W(ω),z_0) is secure.
- Theorem 3 Let ω be a set of *-property-preserving rules and z_0 a state which satisfies *-property. Then Σ(R,D,W(ω),z_0) satisfies *-property.
- ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

Covert Channels

- Covert Channel: a communications channel that allows transfer of information in a manner that violates the system's security policy.
  - Storage channels: e.g. through operating system messages, file names, etc.
  - Timing channels: e.g. through monitoring system performance
- ***Covert channels are not detected by BLP modeling.***

关于BLP模型的争论

- McLean的†-属性和基本安全定理
  - $b(s:\underline{a}) \neq \varphi \Rightarrow [\forall o \in b(s:\underline{a}) \quad [f_c(s) \text{ dom } f_o(o)]]$
  - $b(s:\underline{w}) \neq \varphi \Rightarrow [\forall o \in b(s:\underline{w}) \quad [f_c(s) = f_o(o)]]$
  - $b(s:\underline{r}) \neq \varphi \Rightarrow [\forall o \in b(s:\underline{r}) \quad [f_o(o) \text{ dom } f_c(s)]]$
- McLean证明了定理：
  - $\sum (R, D, W, z_0)$相对于对S'⊆S任何安全状态z_0满足†-属性，当且仅当对于任何行为(r,d,(b,m,f), ,(b',m',f'))和每个s∈ S, W都满足条件:
  - 对任意的(s,o,a) ∈b-b'满足关于S'的†-属性;
  - (s,o,a) ∈b'中不满足关于S'的†属性的元素都不属于b。

- McLean的基本安全定理：
  - ∑ (R, D, W, z_0)是一个安全系统，当且仅当是一个安全状态，并且W满足关于简单安全属性、 †-安全属性和自主访问控制ds的安全属性。

- 显然McLean的系统Σ (R, D, W, z_0)是不安全的，因为这个系统的规则允许信息向下流动。

- 但是这系统满足基本安全定理。

- Bell 的基本安全定理表明的是如果系统的行为满足给定的条件，则系统的状态的属性能得以保持。
- 系统状态的属性的内容与模型是独立的。
- 如果把McLean的†-属性中的条件

  $b(s:\underline{a}) \neq \phi \Rightarrow [\forall o \in b(s:\underline{a}) \quad [f_c(s) \text{ dom } f_o(o)]]$

  解释为主体的完整性级别(可信程度)高于客体的完整性级别时，主体才可以写客体o，则的确McLean的属性表达了另一个安全目标。

  McLean的基本安全定理也就表明了在什么条件下，系统的系统的安全属性得以保持。

4. Biba 模型

- K.J. Biba, Integrity Considerations for Secure Computer Systems, MITRE Technical Report, MTR-3153, 1975.
  - the first security model to address integrity

完整性威胁问题

- 完整性是如何被破坏的？一个系统如何被说服（或迫使）改变自己的行为？
- 完整性的威胁就是在一个子系统在初始时刻认为不正常的修改行为。
- 完整性的威胁主要考虑两个方面：

来源： 内部和外部。

威胁类型：直接的（显式）和间接的（隐式）。

完整性威胁问题

- 直接的只要考虑访问者（主体）或被访问者（客体）。
- 间接的威胁是非正常地使用了恶意的子系统提供的数据或子程序，它们没有能完成预定的任务，它们涉及到最初的来源和信息转移的路径。

完整性威胁问题

- 外部的直接伤害：另一个人用一把刀直接伤害。
- 外部的间接伤害：另一个人将毒药放在食物中。
- 内部的直接伤害：用到自杀。
- 内部的间接伤害：不保护自己身体。

完整性威胁问题

- 外部的直接伤害：另一个人用一把刀直接伤害。
  - 不让坏人接近
- 外部的间接伤害：另一个人将毒药放在食物中。
  - 不雇佣坏人
- 内部的直接伤害：用到自杀。
  - 证明主体的权能
- 内部的间接伤害：不保护自己身体。
  - 保护自己

完整性威胁问题

- 外部的直接
  - 外部系统恶意地篡改另一个系统的数据或程序。
- 外部的间接
  - 一个外部系统插入恶意的子程序
- 内部的直接
  - 修改自己的代码
- 内部的间接
  - 修改函数的指针

Biba 模型的元素

- S: 主体集合;
- O: 客体的集合
- I: 完整性等级集合;
  - 关系 $< \subseteq I \times I$, $(i_1, i_2) \in <$ 当且仅当i_1完整性等级高于i_2的完整性等级。
- il: $S \cup O \rightarrow I$ 定义主体或客体的完整性等级。

  il (integrity level)。

- leq: $I \times I$ 的一个关系，“less than or equal”
- less: $I \times I$ 的一个关系，“less than”
- min: $2^I \rightarrow I$ I的子集的下界。
- $\underline{o}$: $S \times O$ 上的关系，s $\underline{o}$ o 表示s对o具有读的权限。
- $\underline{m}$: $S \times O$ 上的关系，s $\underline{m}$ o 表示s对o具有修改的权限。
- $\underline{i}$: $S \times S$ 上的关系，s_1 $\underline{m}$ s_2 表示s_1对s_2具有ivoke的权限。

完整性等级的含义

- 等级越高，程序正确执行(或者根据程序输入和程序的执行停止来检测出问题)的可靠性就越高。
- 高等级的数据比低等级的数据具备更高的精确性和可靠性(根据不同的测量方法)。
- 此外，这种模型隐含地融入了“信任”这个概念。事实上，用于衡量完整性等级的术语是“可信度”。例如，一个进程所处等级比某个客体的等级要高，则可以认为进程比该客体更“可信”。

（1）对于主体的下限标记策略 (Low-watermark policy for subjects)

① 一个主体能够持有对任何客体的“observe”访问方式。当主体s执行了对客体o的“observe”操作之后，主体的完整级别被置为访问之前主体和客体的完整级别中较小的：$min(il(s), il(o))$ 。

② 一个主体能够持有对给定客体的“modify”访问方式，仅当此主体的完整级别支配该客体的完整级别。

③ 一个主体能够持有对另一主体的“invoke”访问方式，仅当第一个主体的完整级别支配第二个主体的完整级别。

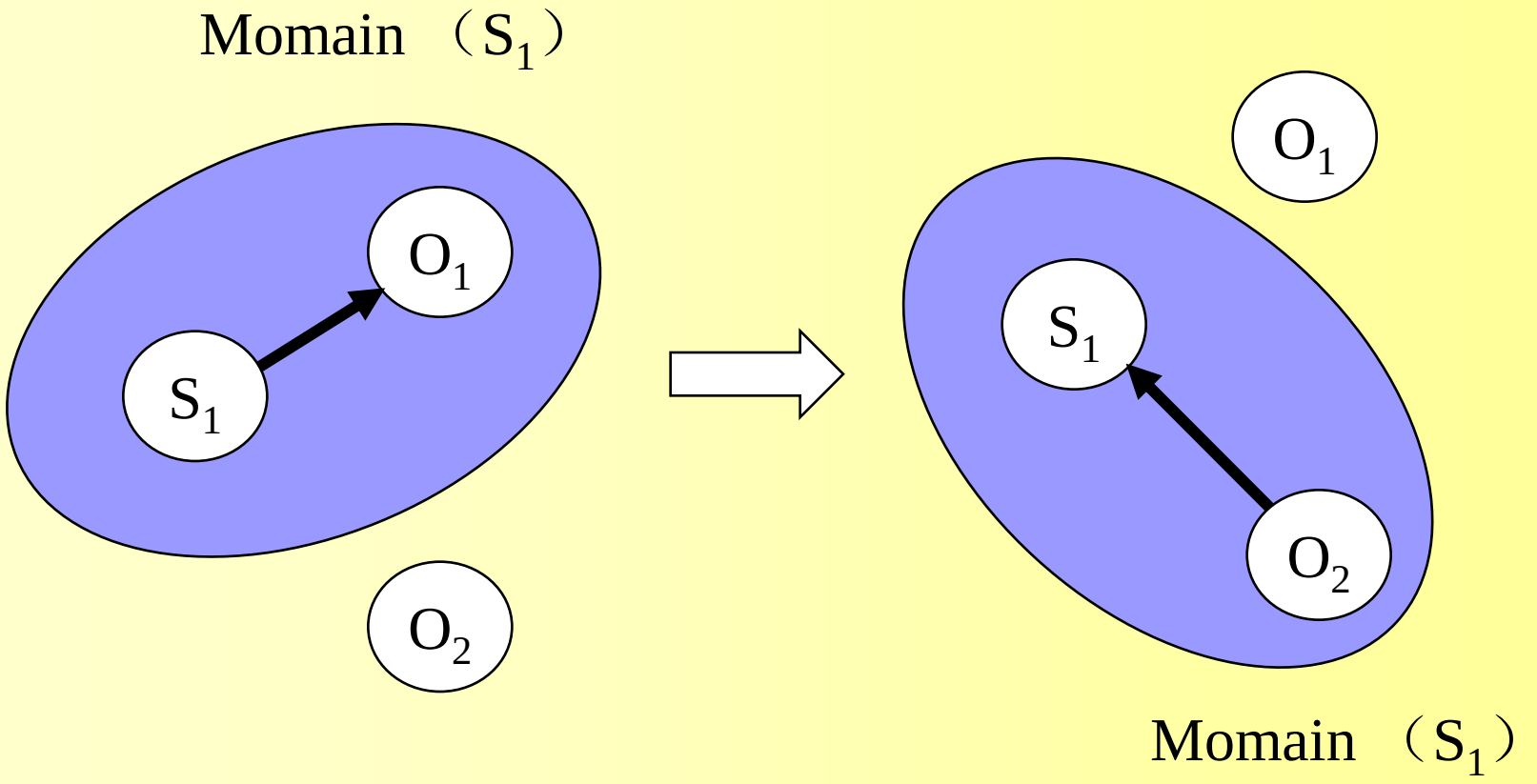
不能间接破坏完整性。

不能直接破坏完整性。

低级别主体不能通过高级别的主体简洁地破坏完整性。

- 由于主体的完整性等级是非递增的，所以，它可能很快就不能访问完整性等级较高的客体了。

s_1在访问o_2后完整性级别下降



- 定义：如果<o_1，o_2，…o_{n+1}>、 <s_1，s_2，…s_n>分别是客体和主体的序列，并且满足：

$$s_i \underline{o}\ o_i \text{ 并且 } s_i\ m\ o_{i+1},\ i=1,\ \ldots n$$

则称<o_1，o_2，…o_{n+1}>和<s_1，s_2，…s_n>为o_1到o_2的一个信息转移路径。

- 定理：如果存在一个o_1到o_{n+1}的一个信息转移路径，并且系统实施下限标记策略，则：

$$o_{n+1}\ \text{leq}\ o_1$$

Low-watermark policy 名称的由来

- 1967年，美国国防科学部组建了计算机安全特别机构，拉开了操作系统安全性研究的序幕。
- 1969年Weissman C. 发表了有关ADEPT-50安全控制的研究成果。ADEPT-50运行于IBM/360硬件平台。
- 高水印模型(high-water-mark model)：
  - 为客体标上敏感级别(sensitivitylevel)属性。
  - 对于读操作，不允许信息的敏感级别高于用户的安全级别clearance)。
  - 对于写操作，在授权情况下，允许使信息从高敏感级别移向低敏感级别。

Weissman C., Security Controls in the ADEPT-50 Time Sharing System， Proceedings of the1069 AFIPS Fall Joint Computer Conference, AFIPS Press, 119-133.

（2） 环策略 (Ring policy)

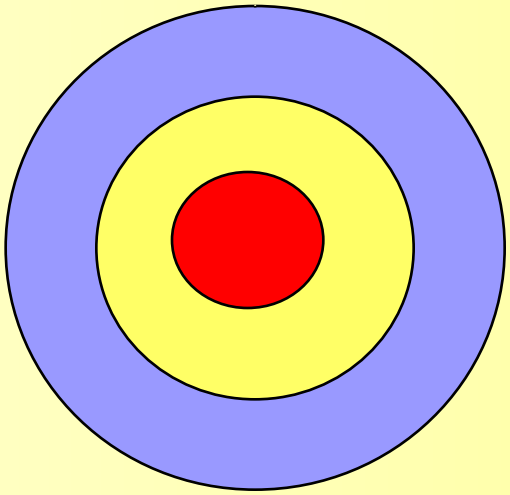
- 一个主体能够持有对给定客体的“modify”访问方式，仅当此主体的完整级别支配该客体的完整级别。
- 一个主体能够持有对另一主体的“invoke”访问方式，仅当第一个主体的完整级别支配第二个主体的完整级别。
- 主体对具有任何完整级别的客体均能够持有“observe”访问方式

- 一个具有高完整性级别主体能够“observe”一个具有较低完整级别的客体
  - 主体必须自己控制净化“observe”的数据，不会通过读入这样的数据而影响其它客体的完整性。
  - 程序的验证必须保证这样的破坏不会发生。

Ring policy 名称的由来

- Named for its similarity to the protection mechanism provided by the Multics hardware.

- M.D. Schroder and J. H. Saltzer, A Hardware Architecture for Implementing Protection Rings, Communications of the ACM, Vol.15, Number 3, March 1972, pp157-170.



(3) Biba 严格完整性策略

- 简单完整性策略 **如果主体s 可以写入客体o，则** $i(s) \geq i(o)$
- 完整性 *-策略：如果主体 s 可以读客体 o，则s能写客体 o' 必须要满足 $i(o) \geq i(o')$
- 调用策略：一个‘不那么可信’的主体 s_1 不能通过调用主体 s_2 来破坏一个客体。主体 s_1 可以调用 s_2 仅当 $i(s_1) \geq i(s_2)$
- **No - ReadDown**
- **No - WriteUp**

Biba模型的应用

- Biba 的访问控制模型被应用于著名的Multics.

5. 第昂(Dion)模型

- Luke C. Dion， **A** Complete Protection Model，IEEE Symposium on Security and Privacy，1981.

安全策略的实施方法

- 隔离
  - 将用户分成不同的组
  - 用户只能与同组的用户通信;
  - 只能影响同组用户的状态信息;
- 分层
  - 将用户分成安全级别
  - 用户只能与同级别用户通信;
  - 只能影响同级别用户的状态信息;
- 信息流控制
  - 控制不同类别间的通信。

- **隔离与分层安全但是不是所期望的**

  因为不同的组、不同的层之间没有共享，需要大量的冗余的软件来支持不同组和层。

  信息流的控制是最合适的。

- **write up: a user can destroy or subtlely modify data at a higher security level.**
  - **It can be limited somewhat by judicial usage of integrity levels, but it cannot be entirely eradicated since many applications will require limited types of write up.**

- **read down: a user can execute a program which reeides at a lower security level. Since the program is constructed and modified at the lower security level, it could be used to modify or destroy highly classified material instead of its purported functionality.**

Security Level and Integrity Level

- **Security Level**
  - All objects and subjects of the system have an associated security level.
  - The set of all possible security levels is partially ordered.
- **Integrity Level**
  - All objects and subjects of the system have an associated integrity level.
  - The set of all possible integrity levels is partially ordered.

- **BLP model prevent:**
  - **a subject reading data residing at a higher security level;**
  - **a subject writing data to a lower security level.**
- **The integrity extension to BLP prevent:**
  - **a subject writing to a higher integrity level;**
  - **a subject reading from a lower integrity level;**

Leads to a stratification.

OBJECT/OBJECT MODEL

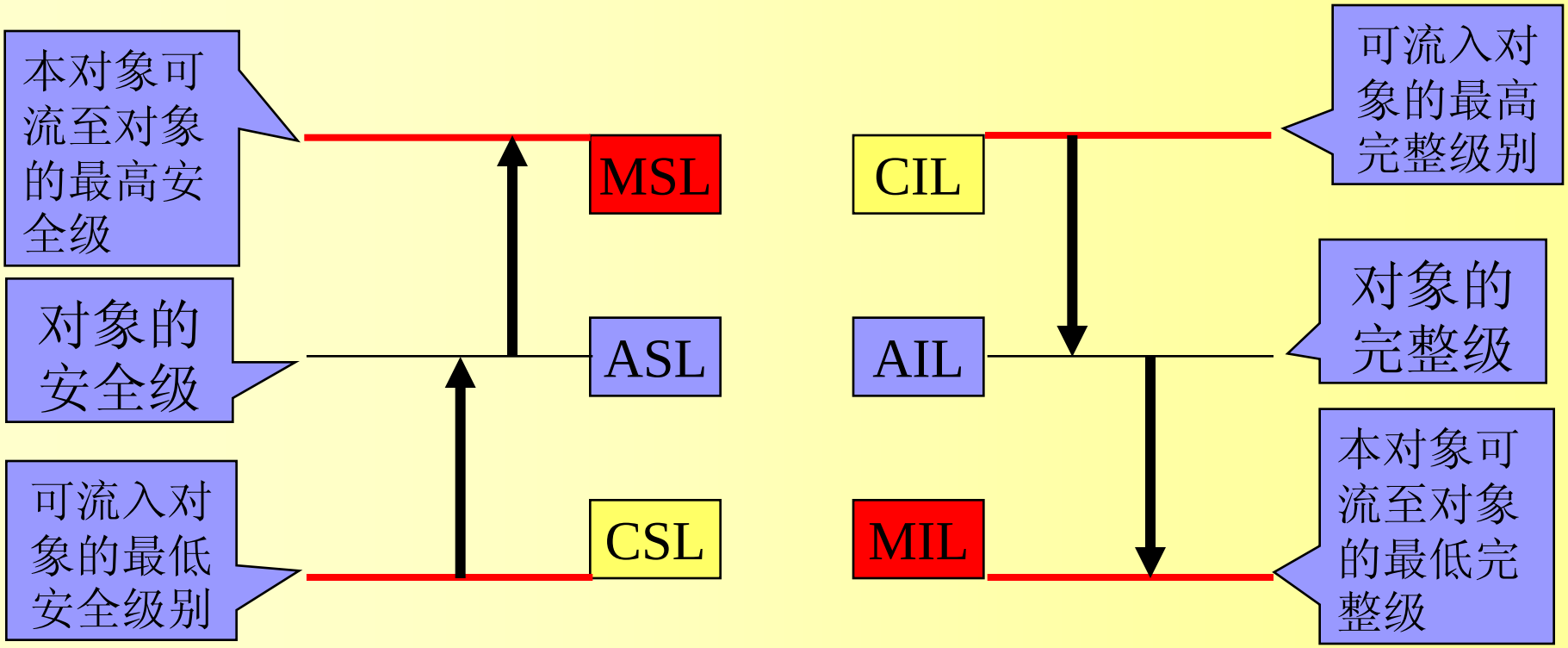
- **Explicit Connection**
  - When a subject wishes to tranafer data between object O_1 and object O_2, it must establish an explicit connection between the objects.
  - *Once* an explicit connection is established between two objects, a subject can then cause data to flow by making the appropriate systemcalls.

MIGKATION AND COKKUPTION

- Each object has three security and three integrity levels
- Migration Level
  - The migration level (ML) of an object is the highest security level (MSL) and the *lowest* integrity level (MIL) to which data in the object may flow.
- Absolute Level
  - The absolute level (AL) of an object is the security level (ASL) and the integrity level (AIL) at which the data in the object is classified.
- Corruption Level
  - The corraptlon level (CL) of an object is the lowest security level (CSL) or the highest integrity level (CIL) from which data may flow into that object.

Additional restrictions

- **The addition of migration and corruption levels to the integrity extended Bell & LaPadula model adds enforced upward (for security) and downward (for integrity) data flow restrictions.**



READ/WRITE LEVELS

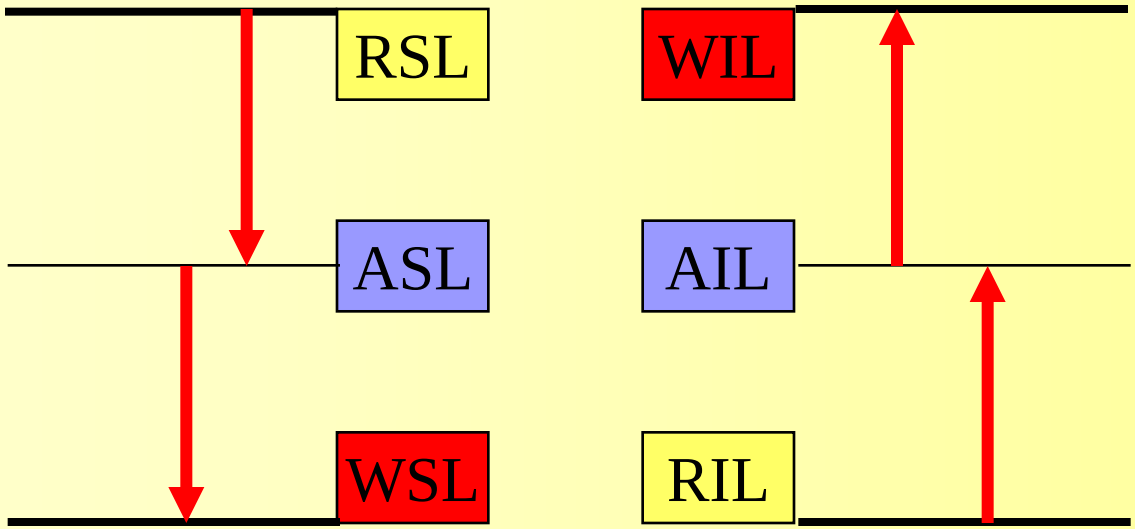
- BLP的系统中，往往必须允许特权的用户可以下写和上读。
- 但是，这样的信任要么完全信任，要么完全不信任。
- Dion模型采用过了受控的信任。

- **Controlled security violation by privileged subjects.**
- **Each subject has three security and integrity levels.**

- **Read Level**
  - The read level (RL) of a subject is the highest security level (RSL) and lowest integrity level (RIL) from which the subject is allowed to read.
- **Absolute Level**
  - The absolute level (AL) of a subject is the security level (ASL) and integrity level (AIL) given the subject upon creation.
- **Write Level**
  - The write level (WL) of a subject is the lowest security level (WSL) and highest integrity level (WIL) to which the subject is allowed to write.

Controled violations

- **If the read security level of a subject is higher than its absolute security level, or if the write security level of a subject is less than its absolute security level, then the subject possesses the ability to perform controlled security violations.**



Integrated view

- **The following inequalities must be true for P to establish a connection from O_1 to O_2:**

$O_1MSL \geqslant O_2MSL$	**(S1)**
$O_1CSL \geqslant O_2CSL$	(S2)
$PRSL \geqslant O_1ASL$	(S3)
$O_2ASL \geqslant PWSL$	(S4)
$PASL \geqslant O_2CSL$	(S5)
$O_1MSL \geqslant PASL$	(S6)
$O_1CIL \leqslant O_2CIL$	(I1)
$O_1MIL \leqslant O_2MIL$	**(I2)**
$PRIL \leqslant 0_1AIL$	**(I3)**
$O_2AIL \leqslant PWIL$	**(I4)**
$PAIL \leqslant O_2CIL$	**(I5)**
$O_1MIL \leqslant PAIL$	**(I6)**

- Migratfon Properties
  - O_1MSL ⩾ O_2MSL
  - O_1MIL ⩽ O_2MIL
- Corruption Properties
  - O_1CSL ⩾ O_2CSL
  - O_1CIL ⩽ O_2CIL
- Security Properties
  - PRSL ⩾ O_1ASL
  - O_2ASL⩾ PWSL

从O_1流向O_2后，最高可能流向O_2MSL，
所以 O_1MSL ⩾ O_2MSL
从O_1流向O_2后，最低可能流向O_2MIL
所以： O_1MIL ⩽ O_2MIL

从O_1流向O_2的，最低可能是O_1CSL，
所以： O_1CSL ⩾ O_2CSL
从O_1流向O_2的，最高可能是O_1CIL，
所以： O_1CIL ⩽ O_2CIL

P读了O_1，写了O_2，
所以： PRSL ⩾ O_1ASL
O_2ASL⩾ PWSL

- Integrity Properties
  - PRIL ⩽O_1AIL
  - O_2AIL ⩽ PWIL
- Write/corruption Properties
  - PASL ⩾O_2CSL
  - PAIL ⩽ O_2CIL
- Read/migration Properties
  - O_1MSL⩾ PASL
  - O_1MIL ⩽PAIL

P读了O_1，写了O_2。

O_2被写了，应该满足左边的O_2的两个corruption级别条件。

O_1迁移了，应该满足左边的条件O_1的两个migration级别条件。

6. Clark-Wilson 模型

- David D. Clark, David Il. Wilson, A Comparison of Commercial and Military computer Security Policies， **IEEE Symposium on Security and Privacy April 27 - 29, 1987, pp184-194.**

Clark-Wilson 的完整性目标

- 防止非授权的用户修改数据
- 防止授权的用户错误(越权)修改数据
- 保持数据的内部和外部一致性

商业环境中的完整性策略

- 商业环境中最关心的是系统数据的完整性和对这些数据操作的完整性。
  - 如果数据满足给定的属性，就称数据处于一个一致性状态。

    例如 存入金额总数D，到昨天为止所有帐户的金额总数YB, 今天到目前为止提取的金额总数W，今天到目前为止所有帐户的金额总数TB，必须满足关系：

    $$D+YB-W=TB$$
  - 事务处理本身的完整性。由谁来检查和验证事务处理是否被正确执行？通过职责分离原则来保证事务处理的完整性。

商业环境中的完整性策略

- 良定义的事务处理 (Well-formed transactions)
  - 在每次操作的前后，都必须满足一致性条件。一个良定义的事务处理是一系列操作，
    - 以确定方式修改数据。
    - 使系统从一个一致性状态转移到另一个一致性状态。
  - 例如，一个储户将钱从一个账号转移到另外一个账号，那么这个事务处理就是转账；在第一个账号中减去金额数，而在第二个账号中增加金额数，这两个操作构成了这个事务处理。每一种操作都可能导致数据处于不一致性状态，但是一个良定义的事务处理必须保证一致性。

商业环境中的完整性策略

- 事务处理本身的完整性— 职责分离原则(sepration of duty)
  - 由谁来检查和验证事务处理是否被正确执行呢?
  - 例如，当一家公司收到一张发票时，采购部门就需要采取若干步骤来支付它。
    - 必须有人曾经请求过某项服务，并确定过支付这项服务的账号。
    - 他必须使发票生效(开出账单的服务是否确实被提供?)。
    - 确认支付服务费用的授权账号被登入借方，且支票已经填好并经过签字。
  - 如果是一个人执行这些所有的步骤，那么他就可以很容易地支付伪造发票了；然而，如果需要两个或两个以上不同的人执行这些步骤，那么只有多人合谋才能欺骗公司。需要至少两个人来处理这个过程就是职责分离原则的一个实例。

商业环境中的完整性策略

- 事务处理本身的完整性— 职责分离原则(sepration of duty)
  - 基于计算机的事务处理也一样，必须有人来检验事务处理是否被正确实现。
  - 职责分离的原则要求检验者和实现者是不同的人。
  - 要使用一个事务处理来破坏数据(或者通过不正当手段改变数据或者将数据置于一个不一致性状态)，就必须有两个不同的人员犯同样的错误，或者他们合谋担保该良定义事务处理已被正确实现。

强制控制

- 鉴别用户
- 为每个良定义的事务处理程序指定权限
- 指定每个用户可以执行的程序
- 审计相关事件
- 采用引用监视器

模型定义

- 两类数据项
  - 约束型数据项CDI (Constrained Data Items) 从属于其完整性控制的数据
  - 非约束型数据项UDIs (Unconstrained Data Items) 不从属于完整性控制的数据
  - *例如，在银行中，账户结算是CDI，因为它们的完整性是银行运作的关键；而当新开账户时，由用户选择的礼物是UDI，因为它们的完整性并‘不是银行运作的关键。CDI集合和UDI集合是模型系统中所有数据集合的划分。 一系列完整性约束(类似于上面讨论的一致性约束)限制着CDI的值。在银行的例子中，前面提出的一致性约束同样也是完整性约束。*

模型定义

- 两组过程。
  - 完整性验证过程IVP (Integrity Verification Procedures)
    - 在IVP执行时，它要检验CDI是否符合完整性约束。如果符合，则称系统处于一个有效状态。
  - 转换过程TP (Transformation Procedures)
    - 它们将系统数据从一个有效状态转换为另一个有效状态，TP实现的是良定义的事务处理。

- 账户结算是CDI。如上描述，检查账户的结算是IVP。
- 存钱、取钱和在账户之间转账则都是TP。
- 为了保证账户的正确管理，在存钱、取钱、转账时，银行的检验人员必须验证银行使用了正确的过程来检验账户结算。
- Clark-Wilson模型使用以下两条证明规则来刻画这些需求

验证与执行

- 验证Certification (C) 和执行enforcement (E) 控制着模型中所有的定义的数据
- 验证由系统管理者、系统的拥有者行使。
- 执行由系统行使。

内部一致性

- 证明规则(CR1)　当任意一个IVP在运行时，它必须保证所有的CDI都处于有效状态。
- 证明规则2(CR2)　对于某些相关联的CDI集合，TP必须将那些CDI从一个有效状态转换到另一个(可能不同的)有效状态。

- CR2定义了一个将一组CDI与一个特定的TP相关联的证明关系。设C是一个证明关系，那么在银行例子中

  (balance，account$_1$)，(balance，account$_2$,)，…，(balance，account$_n$) ∈ C

- CR2隐含指出，如果一个TP没有被证明确可作用于某CDI之上，那么这个TP就可能破坏该CDI。

- 例如，用于银行股票证券投资的TP，可能会破坏账户结算，即使该TP已经通过证明可以作用于股票，因为该TP的行为根本不用考虑银行结算。因此，系统必须阻止TP操作那些它们没有被证明的CDI，这样就导致了下面的实施规则：

- 实施规则 (ER1) 系统必须维护所有的证明关系，且必须保证只有经过证明可以运行该CDI的TP才能操作该CDI。

职责分离原则

- 实施规则2(ER2)

  系统必须将用户与每个TP及一组相关的CDI关联起来。TP可以代表相关用户来访问这些CDI。如果用户没有与特定的TP及CDI相关联，那么这个TP将不能代表那个用户对CDI进行访问。

- 定义一个三元组集合(user，TP，{CDI集})来刻画用户、TP和CDI之间的关系。称这个关系为被允许的关系。当然，这些关系必须经过验证。

- 证明规则3(CR3)

  被允许的关系必须满足职责分离原则所提出的要求。

- 实施规则3(ER3)

  系统必须对每一个试图执行TP的用户进行认证。

- 大部分基于事务处理的系统都要记录每次事务，使得审计人员可以审查这些事务。Clark-Wilson模型仅仅将系统日志看为一种CDI，每一个TP都可以添加这个日志，但是TP不可以重写日志。这就引出以下规则：

- 证明规则4(CR4) 所有的TP必须添加足够多的信息来重构对一个只允许添加的CDI的操作。

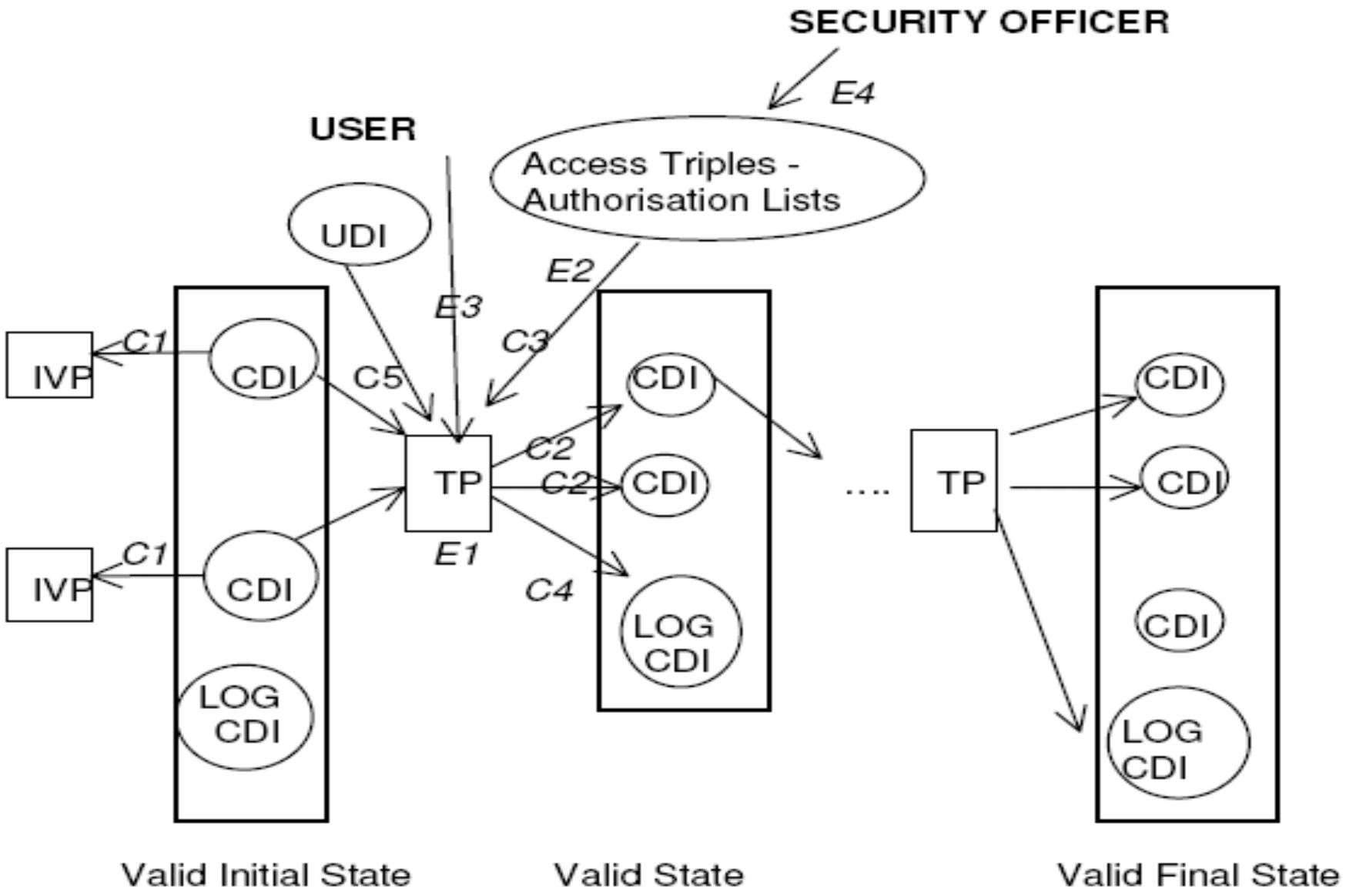
- 当信息输入系统时，信息不一定是可信的或受约束的。例如，当某人将钱存入ATM机时，他不必输入正确的数目。然而当ATM被打开后，在清点现金或支票时，银行工作人员将会检测到差异，然后在他们将在把存款数目输入到一个账户之前修正这种差异。这个例子体现出这个过程：一个UDI(用户声明的存款数目)被检验，必要时进行修正，并在转化为一个CDI(增加到个人账户后的存款数目)之前被证明为正确。Clark—Wilson模型使用第5条证明规则来处理这种情况。

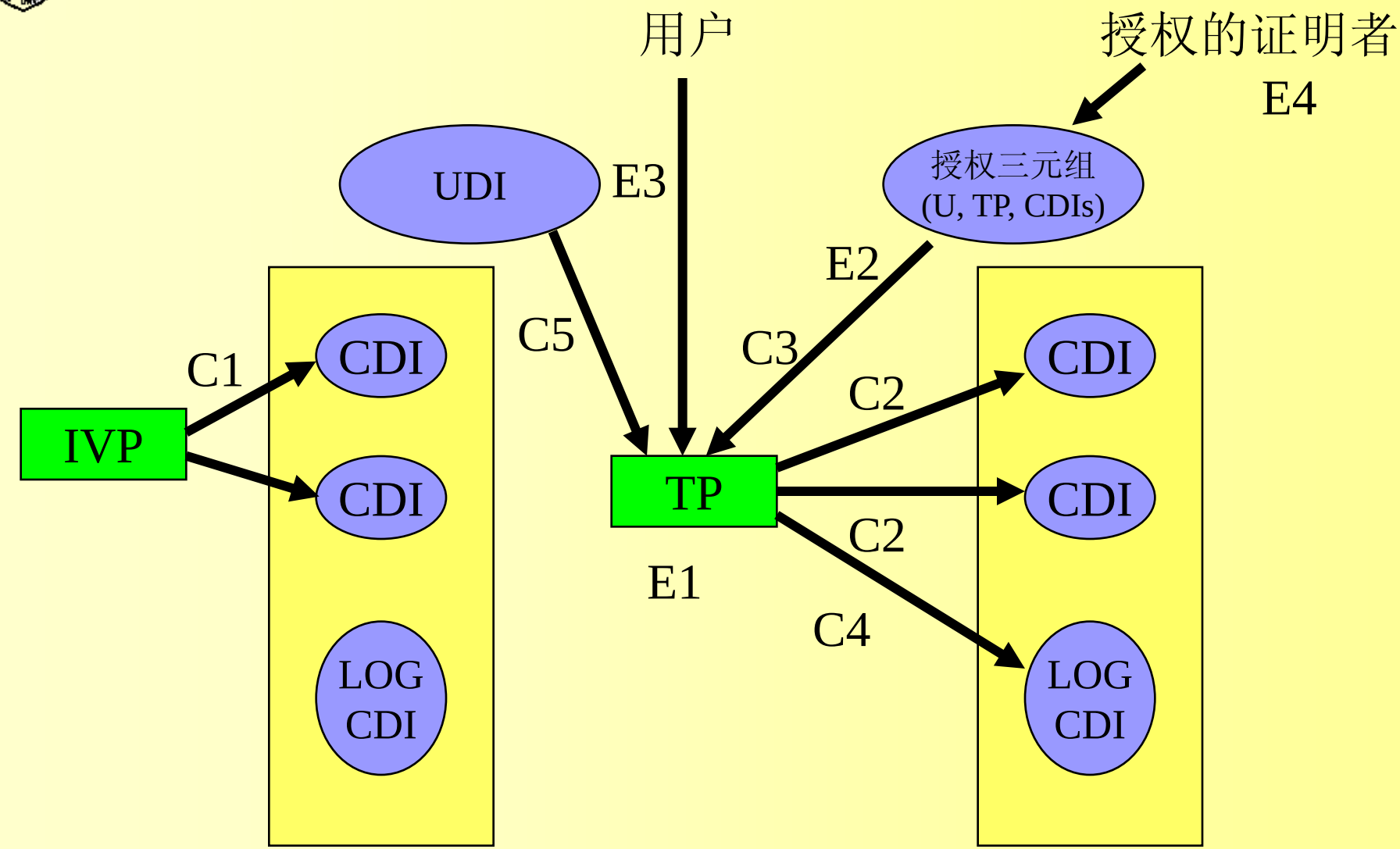
- 证明规则5(CR5)

  - 任何一UDI为输入的TP，对于该UDI的所有可能值，只能执行有效的转换，或者不进行转换。这种转换要么是拒绝该UDI，要么将其转化为一个CDI。

- 如果一个用户能创建TP，并将该TP与他自己和相关的某些实体集合进行关联(如ER3所规定的)，则他就可以操控该TP执行非授权操作。

- 实施规则4(ER4)
  - 只有TP的证明者可以改变与该TP相关的一个实体列表。
  - TP的证明者，或与TP相关的实体的证明者都不会有对该实体的执行许可。


SECURITY OFFICER
E4
USER
Access Triples -
Authorisation Lists
UDI
E2
E3
C3
C1
IVP
CDI
C5
TP
C2
C2
CDI
CDI
....
TP
CDI
CDI
C1
IVP
CDI
E1
C4
LOG
CDI
CDI
LOG
CDI
Valid Initial State
Valid State
Valid Final State


用户
授权的证明者
E4
UDI
E3
授权三元组
(U, TP, CDIs)
E2
C5
C3
C1
CDI
CDI
IVP
TP
C2
CDI
CDI
C2
E1
C4
LOG
CDI
LOG
CDI

7. 中国墙模型

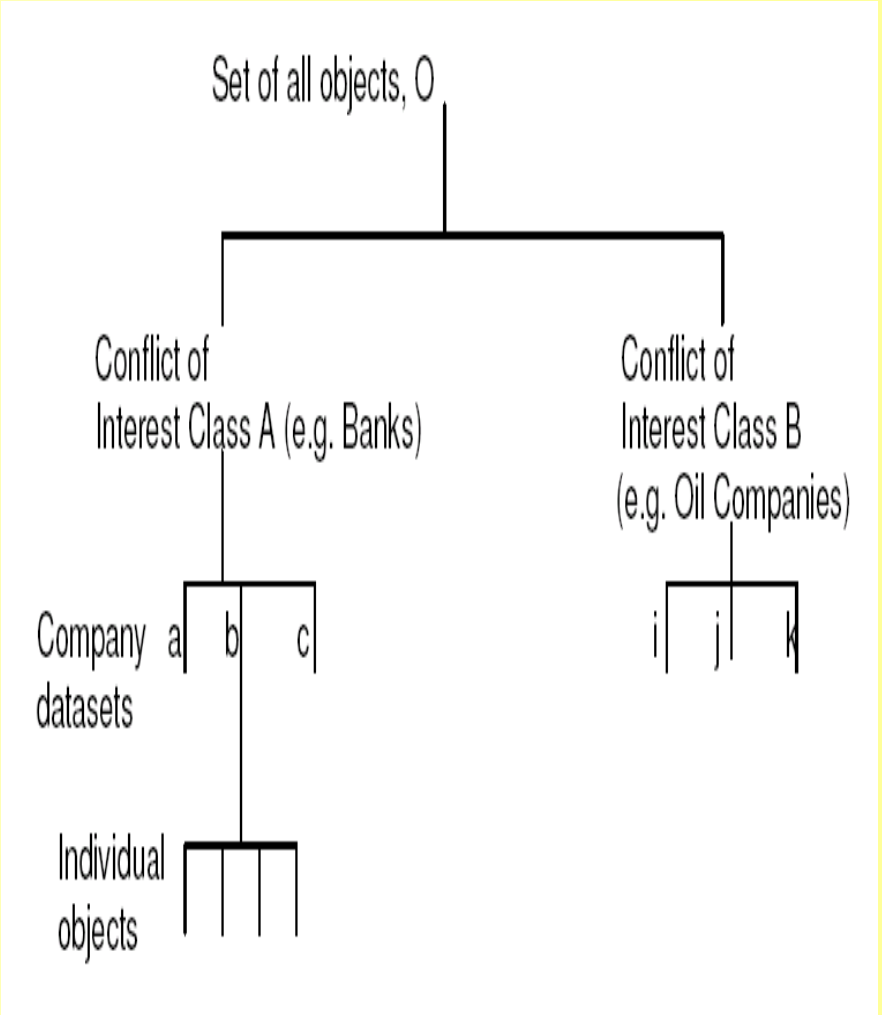
- D.Brewer, M.Nash, The Chinese Wall Security Policy, *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, Oakland, May 1989.

中国墙模型

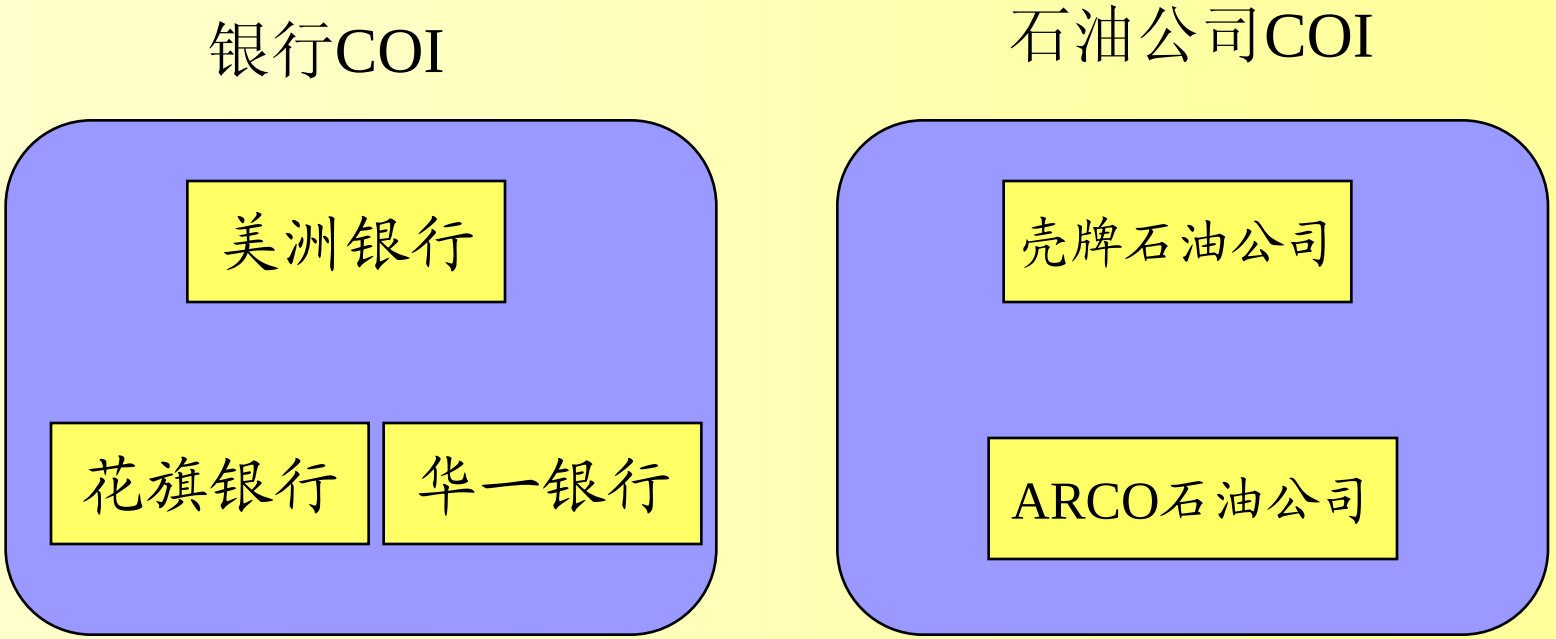
- 中国墙模型是一种同等地考虑保密性与完整性的安全策略模型。
- 该策略模型主要解决商业中的利益冲突问题，其重要性等同于Bell－LaPadula模型在军事中的意义。
- 中国墙模型通常用于股票交易所或者投资公司的经济活动等环境中。在这种意义上，中国墙模型的目标就是防止利益冲突的发生。
  - 当交易员代理两个客户的投资，并且这两个客户的最大利益相互冲突时，交易员就有可能帮助其中一个客户盈利，而导致另一个客户的损失。

非形式化描述

- 数据库的客体是指与某家公司相关的信息条目。
- 公司数据集(CD)包含了与某家公司相关的若干客体。
- 利益冲突(COI)类包含了若干相互竞争的公司的数据集。
- C0I(o)表示包含客体o的COI类
- CD(o)表示包含客体o的公司数据集。



- Anthony有权访问美洲银行的公司数据集(CD)中的客体。因为花旗银行的公司数据集和美洲银行的公司数据集在同一个COI类中，所以Anthony不能获得访问花旗银行CD中的客体的权限。



CW-简单安全属性

- 设想Anthony开始为美洲银行的证券业务工作，然后转而从事花旗银行的证券业务。
- 这种情况下，虽然他在某一时段只接触了银行COI类的一个CD，但是他从美洲银行的业务中所获得的很多信息还是最新的，因此Anthony就可以用这些信息来指导当前他所从事的花旗银行的业务——这就是利益冲突。
- PR(S)表示S曾经读取过的客体集合
- 公司有些数据是可以公开的，如股票交易年报等。中国墙模型不限制这些数据，这些数据称为无害客体。

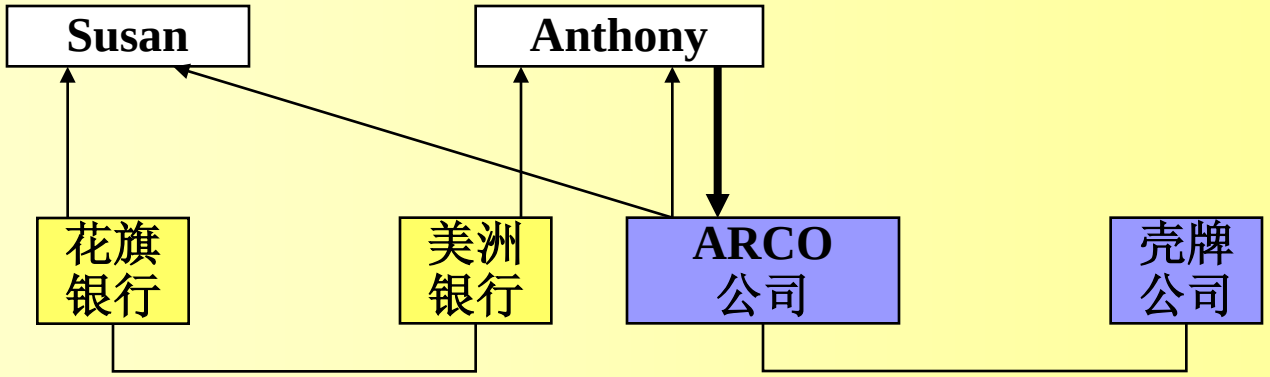
CW -简单安全属性

- CW-简单安全属性 S可以读取O，当且仅当以下任一条件为真:
  - 存在一个O’，它是S曾经访问过的客体，并且CD(O’)=CD(O);
  - 对于所有的客体O’

    $O' \in PR(S) \rightarrow COI(O') \neq COI(O)$
  - O是一个无害客体。

CW *-属性

- 假设Anthony和Susan在同一个交易所工作，Anthony可以读取美洲银行CD中的客体，Susan可以读取花旗银行CD的客体，他们都可以读取ARCO公司CD中的客体。
- 如果Anthony还可以写ARCO公司CD中的客体，那么他就可以读出美洲银行CD中的客体，并将其写到ARCO公司CD的客体中，这样Susan就可以读取这些信息了。此时，Susan可以间接地获得美洲银行CD中的信息，从而导致利益冲突。
- 因此CW简单安全条件必须做出相应扩展，以防止这种情况发生。



CW*-属性

- CW*-属性。主体S能写客体O，当且仅当以下两个条件同时满足：
  - CW-简单安全条件允许S读O;
  - 对于所有有害客体O’，S能读取O’ → CD(O’)=CD(O)。

8. 基于角色的个访问控制

- R.S. Sandhu, Role-Based Access Control, http://www.isse.gmu.edu/faculty/sandhu

8.1 引言

1. 动因与背景
2. RBAC局限性
3. 什么时角色
4. 角色与组

8.1.1动因与背景

- 信息系统中具有大量的用户和大量的权限
- RBAC 的核心
  - 权限与角色进行关联
  - 用户相关的角色
- 角色为特定的工作而创建
- 根据用户的职责和资格指派相应的角色
- 用户容易从一个角色变换到另一个角色
- 权限的变化与角色关系密切

8.1.2 RBAC 的局限

- RBAC没有考虑对操作序列的控制
  - 例如采购过程需要一系列的操作，其中需要对权限进行严格的控制。
- 在RBAC的基础上还可以构建其他的访问控制策略
  - 事务控制
  - 基于任务的授权Task-based authorization controls

8.1.3 什么是角色

- 角色表示了能够干好某项工作的能力，例如医师、药剂师
- 角色体现了作一项工作需要的权限和责任
  - 权限与能力不是一个概念,例如李平具有作许多工作的能力,但是他只能被指派其中一项。
- 角色反映了一项工作的职责

8.1.4 角色与组

- 组是用户的集合
- 角色一方面反映了一个用户的集合，另一方面也反映了权限的集合
- 角色将两者结合了起来

8.2 RBAC 模型

- $RBAC_0$ 基本框架
- $RBAC_1$ 增加了角色的继承关系
- $RBAC_2$ 增加了限制
- $RBAC_3$ 继承了$RBAC_1$和$RBAC_2$

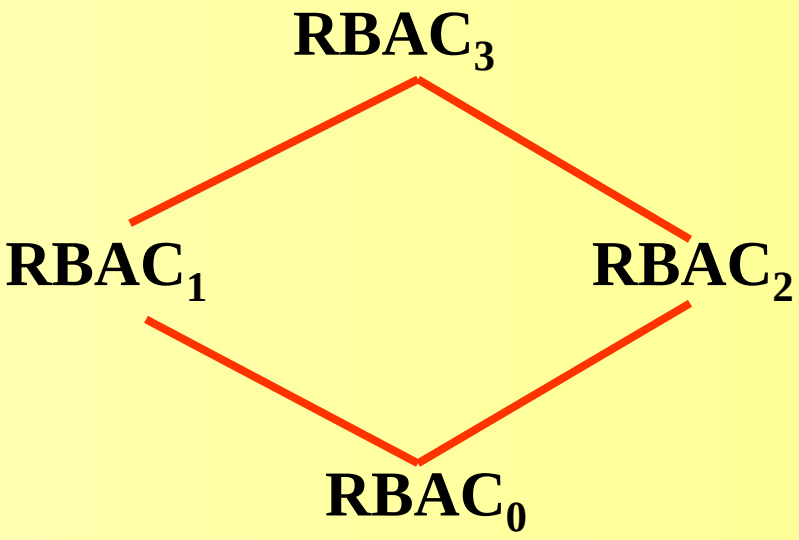


图 1

8.2 $RBAC_0$

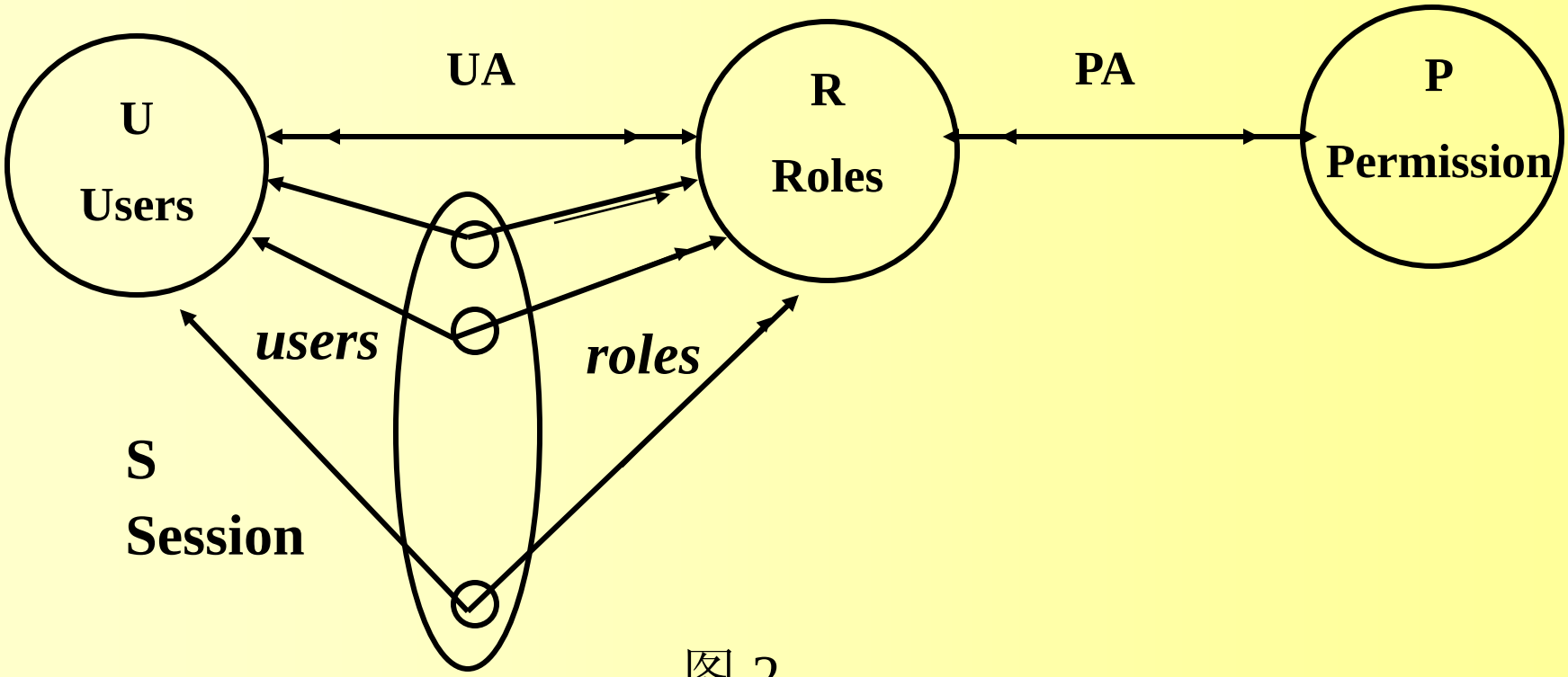


图 2

8.2 $RBAC_0$

- 用户可以是人
- 用户可以是 一个智能体
  - 机器人
  - 软件
  - 计算机
  - 网络中的计算机

8.2 $RBAC_0$

- 角色是一项工作相关的职责和权限

8.2 Base Model: $RBAC_0$

- 权限是在系统内对某个客体的特定的访问方式的权限
- 客体是系统中的数据或其他形式的资源
- 权限可以有不同的形式和粒度
  - 一个权限可能是对整个网络的访问权限
  - 一个权限也可能是对某个文件的访问权限

8.2 Base Model: $RBAC_0$

- session 把一个用户映射到一个或多个角色。在这个session期间这个用户以若干个角色工作。
- 图中的双箭头表示session到角色的多值映射。
- Session到用户是单值映射，并且在session的存活期间不变。

8.2 $RBAC_0$

$RBAC_0$ 的模型组成分部分

- U表示用户，R表示角色， P表示权限
- $PA \subseteq P \times R$, 角色的权限指派
- $UA \subseteq U \times R$, 用户的角色指派
- user: $S \rightarrow U$, session到用户的映射，在session的存活期间不变。
- roles: $S \rightarrow 2^R$, 将session 映射到一组角色

  $roles(s) = \{r \mid (user(si), r) \in UA\}$
- Session s 的权限为

  $\cup_{r \in roles(s)} \{p \mid (p, r) \in 2^A\}$

8.3 $RBAC_1$ — 角色的层次

- $RBAC_1$ 在角色集合中引入了层次结构。
- 角色的层次放映了一个组织中角色之间的职权关系。

8.3 $RBAC_1$ — 角色的层次

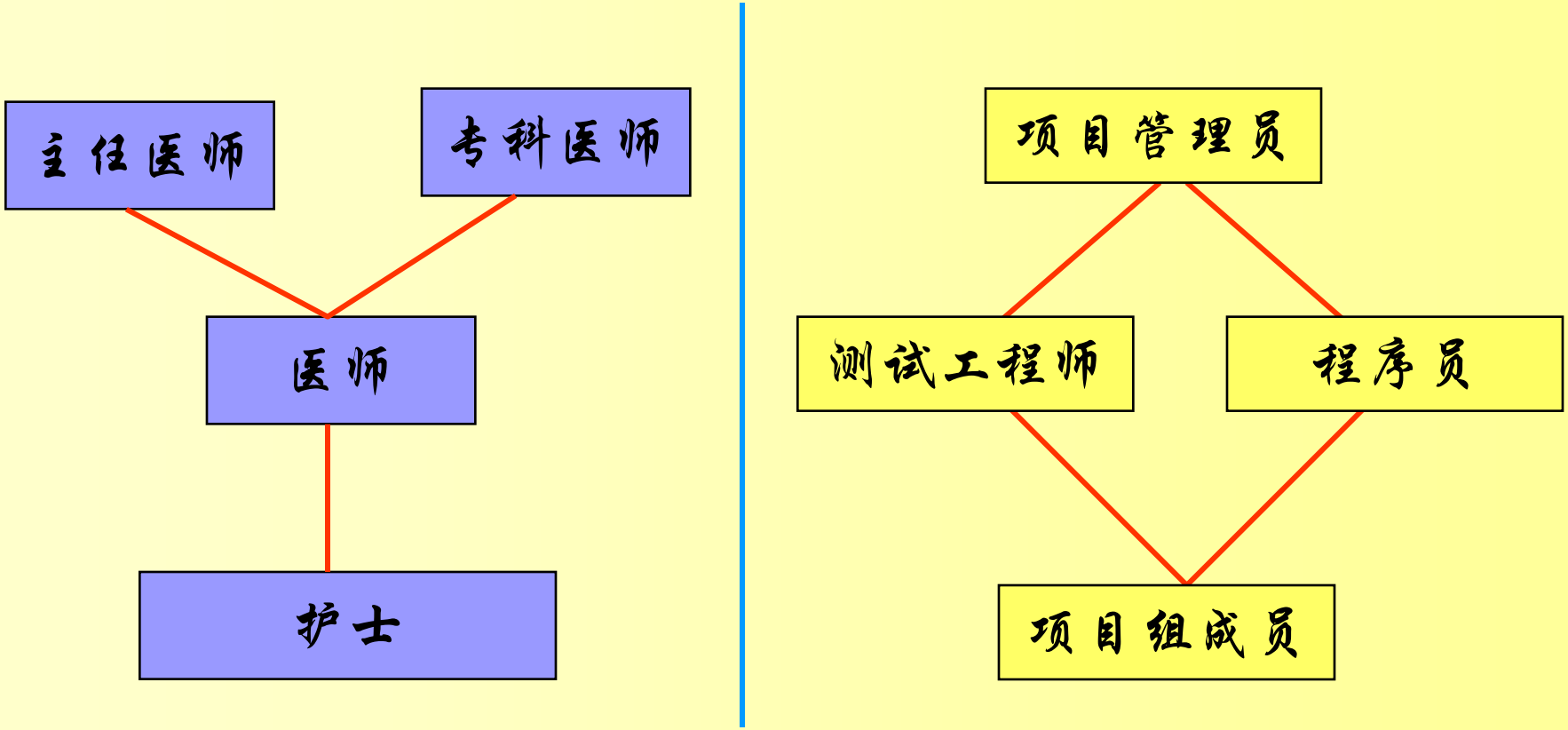


图 3

8.3 $RBAC_1$ — 角色的层次

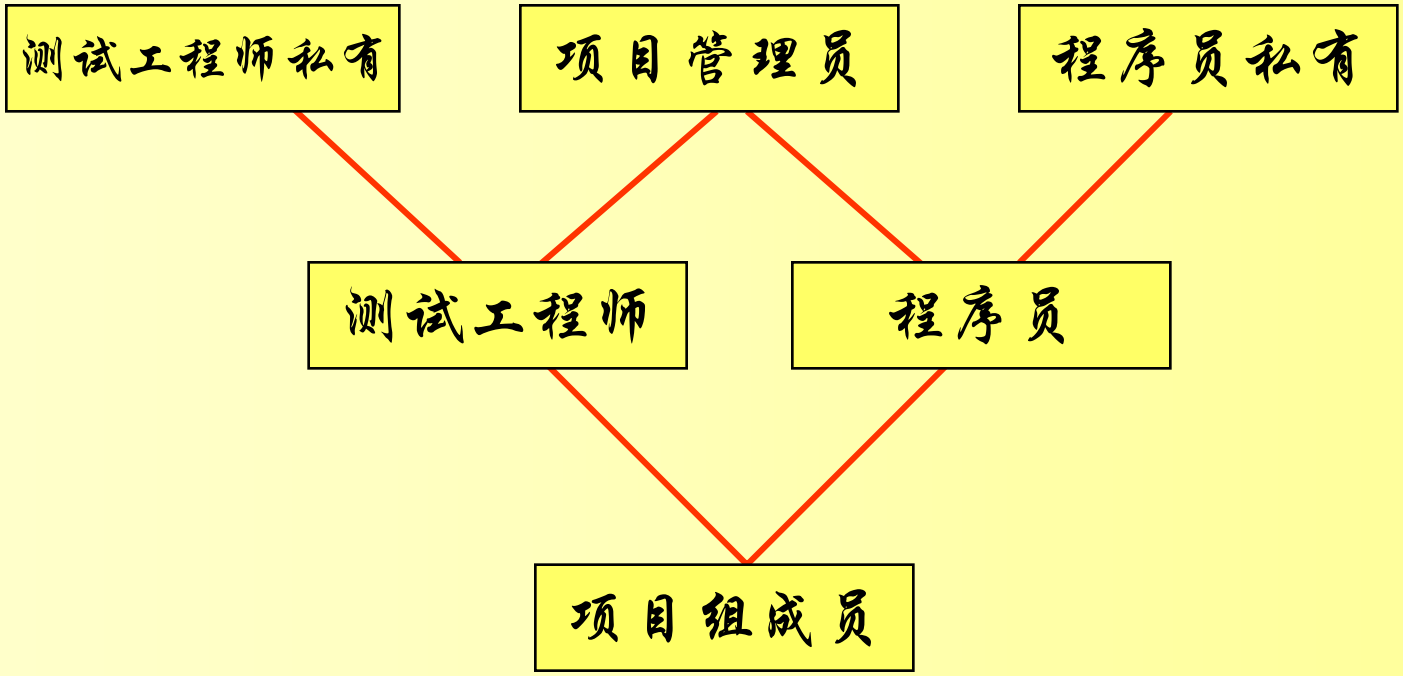


图 4

8.3 $RBAC_1$ — 角色的层次

- **$RBAC_1$ 具有如下组成分部分**
- **U,R,P,S,PA,UA 和 user 与 $RBAC_0$的相同**
- **RH　R×R 是在R上的偏序层次关系≥**
- **roles: $S \rightarrow 2^R$, 修改为**

$$roles(si) = \{r \mid r' \geq r \quad [(user(si), r') \in UA]\}$$

- **Session s 的权限为**

$$session(s) = \cup_{r \in roles(si)} \{p \mid r'' \leq r, \ (p, r'') \in PA\}$$

8.3 $RBAC_1$ — 角色的层次

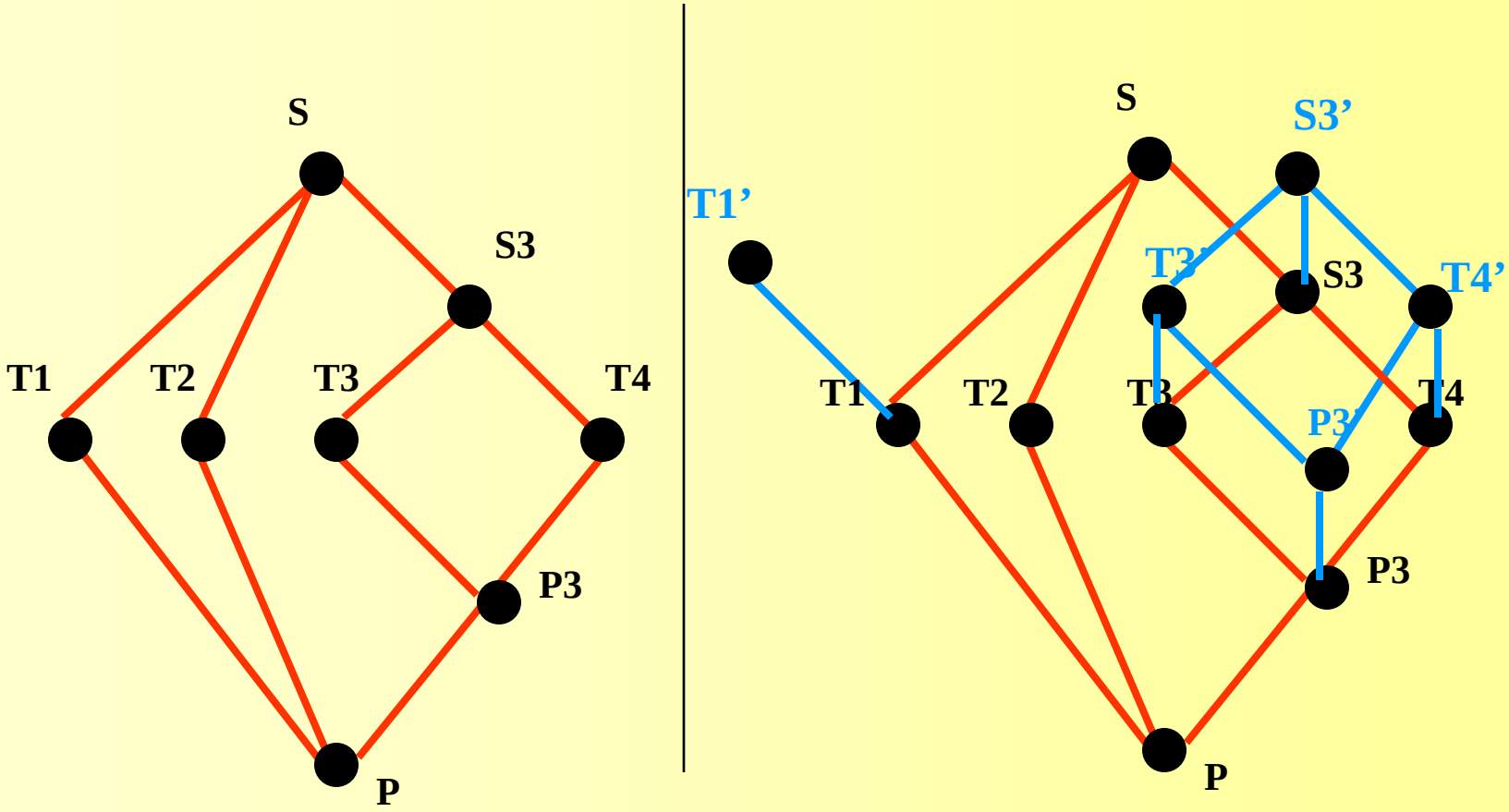


Figure 5

8.4 限制 $RBAC_2$

- 限制是RBAC的主要部分，甚至是RBAC的主要动因。
- 角色互斥限制
  - 在许多组织都有限制一个用户同时指派某些角色，称为权限分离。
  - 例如：采购员和财务人员，会计和出纳，程序员和测试员等。

8.4 Constraints: $RBAC_2$

- 限制条件可以作为组织的高层策略；
- 有了高层的限制条件的控制，底层的策略可以较少的顾虑策略的一致性问题。
- 这样底层的策略可以被代理或采用分布式的方式。
  - 高级安全员设置总体安全框架，限制底层管理员的管理或设置策略的权限。

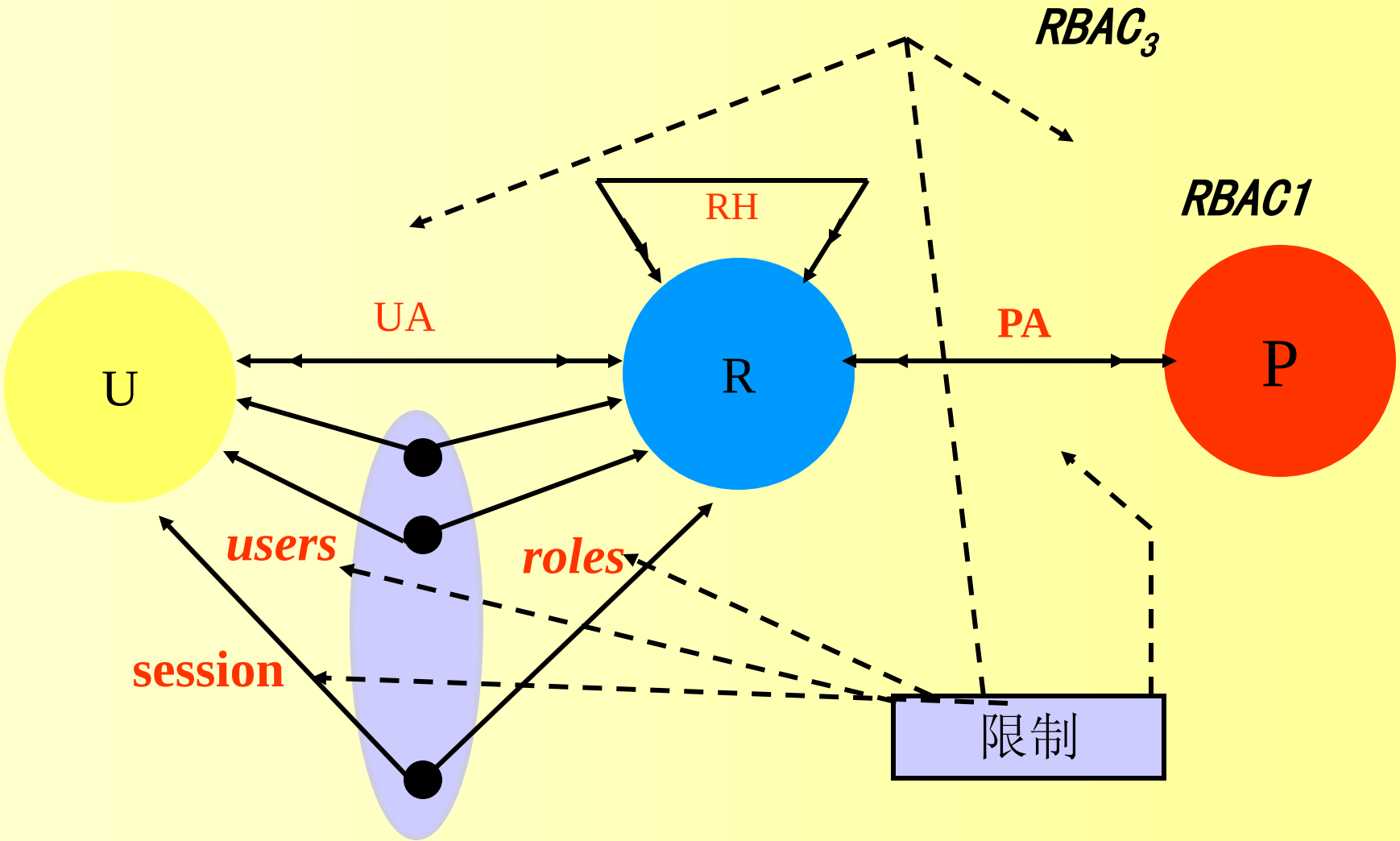
8.4 $RBAC_2$ 限制

- 定义: $RBAC_2$ 在$RBAC_0$ 的组成中删除那些不满足限制条件的元素。

- 互斥限制
  - 会计师不能同时成为出纳。
- 数量限制

- 前提条件限制
  - 项目测试员必须是项目组成员

8.5 $RBAC_3$



9. Workflow Authorisation Model (WAM)

- Simone Fischer-Hubner, IT-Security and Privace, Lecture Notes in Computer Science, Springer, 2001；

9. Workflow Authorisation Model (WAM)

- Workflows represent processes in manufacturing and office environments
  - consist of several well-defined activities —called tasks.
- Authorisation mechanisms have to
  - ensure that these tasks are executed by authorised subjects only;
  - subjects may gain access to required objects only during the execution of the task

Workflow Authorisation Model (WAM)

- synchronising the authorisation flow with the workflow ;
- authorisation is granted only when a tasks starts and revoked as soon as the task finishes;

Model elements

- Subjects：$S = \{s_1, s_2, ...\}$ ；
- objects： $O = \{o_1, o_2, ..\}$,
- objects types $T = \{t_1, t_2, ...\}$;
- privileges set PR;
- time set: $T = \{r \in R \mid r \geq 0\}$;
- time interval set $\{ [t_l, t_u] \in T \times T \mid t_l \leq t_u \}$

Model elements

- A workflow
  - partially ordered set of tasks $\{w_1, w_2, ..., w_n\}$,
- A **task w_i** is defined as: $(OP_i, T_{INi}, T_{OUTi}, [tl_i, tu_i])$
  - OP_i is the set of operations to be performed in w_i;
  - T_{INi} is the set of object types allowed as inputs,
  - T_{OUTi} is the set of object types as expected as outputs,
  - $[tl_i, tu_i]$ is the time interval during which w_i must be executed; specifies temporal constraints stating the lower and upper bounds of the time interval during which a task is allowed to be executed;

Model elements

- A **task-instance w_{insti}** is defined as:

$$(OPER_i, IN_i, OUT_i, [ts_i , tf_i])$$

  - $OPER_i$ is the set of operations performed during the execution of w_i;
  - IN_i is the set of input objects to w_i such that

$$IN_i = \{x \in O \mid T(x) \in T_{INi}\}$$

  - OUTi is the set of output objects from w_i such that

$$OUT_i = \{x \in O \mid T(x) \in T_{OUTi}\}$$

  - and $[ts_i, tf_i]$ is the time interval during which w_i has been executed;

Model elements

- An **authorisation** A_i is defined as a 4-tuple

$$Ai = (s_i, o_i, pr_i, [tb_i, te_i])$$

  - Subject s_i is granted access on object o_i
  - with privilege pr_i
  - at time tb_i
  - and is revoked at time te_i

Model elements

- an **authorisation template of** a task w_i **AT(wi)** is defined as a 3-tuple

$$AT(w_i) = (s_i, (t_i, -), pr_i)$$

  - where $s_i \in S$;
  - $(t_i, -)$ is an *object hole* which can be filled by an object oi of type t_i;
  - pr_i is the privilege to be granted to s_i on object o_i when $(t_i,-)$ is filled by o_i;

s_i is allowed to perform task w_i
and that w_i can only process objects of type t_i
and requires a privilege pr_i on the objects.

Model elements

- A task w_i may have more than one authorisation template attached to it,
  - if there are more than one **type** of object to be processed
  - or more than one **subject** is required to perform the processing;

- authorisation is granted only when a task actually starts and is revoked when the task is completed
- **Authorisation Derivation Rule**
  - task $w_i = (OP_i, IN_i, OUT_i, [tl_i, tu_i])$
  - an authorisation template

    $AT(w_i) = (s_i, (t_i, -), pr_i)$
  - an authorisation $A_i = (s_i, o_i, pr_i, [tb_i, te_i])$ is derived as follows

Grant Rule

- Suppose object $o_i \in T_{INi}$ is sent to w_i at ta_i to start w_i.
- Let the starting time of w_i be ts_i
  - If $ta_i \leqq tu_i$, then $s_i \leftarrow s(AT)$, $pr_i \leftarrow pr(AT)$, $te_i \leqq tu_i$,
  - if $ta_i \leqq tl_i$, then $tb_i \leftarrow tl_i$; otherwise $tb_i \leftarrow ta_i$

Revoke Rule

- Suppose w_i ends at tf_i
  - At time tf_i o_i leaves w_i;
  - IF $tf_i \leqq tu_i$, then $te_i \leftarrow tf_i$;

Example

- The workflow of check processing consists of the three tasks: preparation, approval and issue of the check
  - w1 = ({read request, prepare check}, {request, check}, {check}, [10, 50])
  - w2= ({approve check}, {check}, {check}, [20, 60])
  - w3 = ({issue check}, {check}, {check}, [40,80])

- $AT_1(w_1)$ = (John, (request, -), read), $AT_2(w_1)$ = (John, (check, -), prepare)
- $AT(w_2)$ = (Mary, (check, -), approve)；
- $AT(w_3)$ = (Ken, (check, -), issue)；

10. 不干扰模型

参考文献

I. Matt Bishop, 计算机安全学——安全的艺术与科学，电子工业出版社，2005年5月。

II. John Rushby, Noninterference, Transitivity, and Channel-Control Security Policies, Technical Report CSL-92-02, December 1992.

0. 引言

- 例：假如两个用户要共享一个系统。这两个用户是分开的，每个用户都有一个自己的虚拟机，并且他们不能直接通信。然而，他们根据负载而共享CPU。
- 如果用户Matt(具有SECRET级别)运行一个需要占用CPU时间较长的程序，用户Holly(具有CONFIDENTIAL级别)运行一个需要占用CPU时间较短的程序，则Matt的程序将占用大部分的CPU时间。
- 这就提供了一条可供Matt和Holly通信的隐蔽信道。
- 他们先协商好共同的开始时间和时间片长度(例如，在中午12：00开始，时间片长度为1分钟)。如果要传输比特1，则Matt就在这个时间片运行他的程序；如果要传比特0，Matt就不在这个时间片运行他的程序。
- 这样，每隔一分钟，Holly尝试运行一个程序，如果这个程序可以运行，则表示Matt的程序没有占用CPU，所以要传输的比特是0；如果这个程序在这个时间片内不能运行，则要传输的比特是1。在这个例子中，虽然没有传统意义上的“写”，但信息却从Matt传到Holly，从而违反了Bell-LaPadula模型的约束条件。

1. 状态转换与输出函数

- 首先，将系统视为状态机
  - 一个主体的集合S=$\{s_1, s_2, \cdots\}$
  - 一个状态集合Σ=$\{\sigma_1, \sigma_2, \cdots\cdots\}$
  - 一个输出的集合O=$\{o_1, o_2, \cdots\}$
  - 一个命令的集合Z=$\{z_1, z_2, \cdots\}$
  - 状态转换命令集合C=S×Z
- 定义1
  - 状态转换函数 T: C × Σ → Σ 描述了在状态σ下执行命令c的效果。
  - 输出函数 P: C × Σ →O描述了在状态O下执行命令c的机器输出。初始时，系统处于状态σ_0。

1. 状态转换与输出函数

- 例
  - 假设一种机器具有两个比特的状态信息：H和L(分别代表“高”和“低”)。机器具有两条命令：xor0和 xor1，分别代表把这两个比特与0和1进行异或。
  - 系统有两个用户，Holly(可以读取高级别和低级别的信息)和Lucy(只能读取低级别信息)。系统保存两个比特的状态(H, L)。在这里，我们称这种机器为“两比特机”。
  - 在这个例子中，不管是Holly还是Lucy执行的指令，都可以影响两个状态比特。
  - 状态集合Σ = { (0，0)，(0，1)，(1，0)，(1，1) }，主体集合S={ Holly，Lucy}，命令集合C={xor0, xor1}。 表说明了状态转换函数的输出结果。对于Holly，输出函数输出两个比特，而对于Lucy，仅仅输出L比特。

	输入状态（H, L）			
命令	(0,0)	(0,1)	(1,0)	(1,1)
xor0	(0,0)	(0,1)	(1,0)	(1,1)
xor1	(1,1)	(1,0)	(0,1)	(0,0)

1. 状态转换与输出函数

- $T(c_0, \sigma_0) = \sigma_1$,
- $T(c_{i+1}, \sigma_{i+1}) = T(c_{i+1}, T(c_i, \sigma_i))$
- 因此设C*为C中命令的一个序列集合，那么可以定义T*：$C^* \times \Sigma \rightarrow \Sigma$

  $C_s = c_0, \cdots, c_n$

  $\Rightarrow T^*(C_s, \sigma_i) = T(c_n, T(c_{n-1}, \cdots, T(c_0, \sigma_i) \cdots))$
- 同样，就可以定义一个类似的函数P*：$C^* \times \Sigma \rightarrow O$，这个函数对于始于某个初始状态系统的命令序列，会产生输出序列。

2. 投影

根据以上假设，输出给出了系统活动的一个记录。问题在于某些主体所能看到的输出(和操作)是受限的。在上面的例子中，Holly不应该看到Matt的输出，但是Matt可以看到Holly的输出。通过下述定义使这个概念更为严格：

- **定义2 设$T^*(C_s, \sigma_i)$是某系统的一个命令序列的状态转换，$P^*(C_s, \sigma_i)$是对应的输出。**

  那么proj(s，c_s，σ_i)是在输出$P^*(C_s, \sigma_i)$中的一个子集，它代表主体s被授权可见且与其$P^*(C_s, \sigma_i)$中输出顺序一致的输出集合。

  在这个定义中，每个命令都可以产生一系列输出，但是为了防止主体根据这些信息推导出系统以前的状态信息，因而禁止权限不足的主体看到这些输出。函数proj(s，c_s，σ_i)只是从输出中删除没有授权s可见的输出后，所得到的输出序列。

3. 清除函数

- 定义3 设G⊆S是一个主体集合，A⊆Z是一个命令集合。

  定义$\pi_G(c_s)$为从cs中删除所有符合s∈G的(s, z)元素所得到的子序列。
  定义$\pi_A(c_s)$为从cs中删除所有符合z∈A的(s, z)元素所得到的子序列。
  定义$\pi_{G,A}(c_s)$为从cs中删除所有符合s ∈ G且z ∈ A的(s，z)元素所得到的子序列。

- 清除函数π表示特定命令的执行必须对某些主体是不可见的。对一个命令序列，使用清除函数就输出一个主体可见的命令序列。对于一个特定的系统，其需要的保护域将规定G和A中的成员关系。

3. 清除函数

- 例：在两比特机中，设σ_0 =(0，1)。对于这个状态机，Holly先执行命令xor0，Lucy执行命令xor1，然后Holly 再执行命令xor1。上述命令对所有的状态位都有影响，并且每条命令执行之后都输出两个状态比特。令c_s是序列(Holly，xor0)，(Lucy，xor1)，(Holly，xor1)，则输出就是011001(其中状态比特按顺序记录，每一对比特中High比特总写在前面)。
- Proj(Holly, c_s, σ_0)=011001
- Proj(Lucy, c_s, σ_0)=101
- $\pi_{Lucy}(c_s)= \pi_{Lucy,xor1}(c_s)=$ (Holly，xor0)，(Holly，xor1)
- $\pi_{Holly}(c_s)=$(Lucy, xor1)
- $\pi_{Lucy,xor0}(c_s)=$(Holly, xor0), (Lucy, xor1) ，(Holly，xor1)
- $\pi_{Holly,xor0}(c_s)= \pi_{xor0}(c_s)=$ (Lucy，xor1)，(Holly，xor1)
- $\pi_{Holly,xor1}(c_s)=$ (Holly, xor0), (Lucy, xor1)
- $\pi_{xor1}(c_s)=$(Holly, xor0)

4. 不干涉

- 定义4 设G，G’ ⊆S是两个不同的主体集合，A ⊆Z是一个命令集合。G中的用户在运行A中的命令时不干涉G’中的用户(记为$A, G : I G'$)，当且仅当对于所有由C*中元素组成的序列和所有的 s ∈ G’，满足

$$proj(s, c_s, \sigma_0) = proj(s, c_s, \pi_{G,A}(c_s))。$$

- 例：假设在上面的两比特机中，序列c_s=(Holly，xor0)，(Lucy，xor1)，(Holly，xor1)，其中的命令对所有的状态位都有影响，所有的状态位都将成为每条命令后的输出。设G={Holly}，G’={Lucy}，A=φ。那么$\pi_{Holly}(c_s)$=(Lucy, xor1)，$proj(Lucy, \pi_{Holly}(c_s), \sigma_0)$=0。
- 这就意味着假设{Holly}: | {Lucy}是错误的，因为$proj(Lucy, cs, \sigma_0)$=101 ≠ 0= $proj(Lucy, \pi_{Holly}(c_s), \sigma_0)$。

  直观上，这是可以理解的，因为那些改变H比特的命令也影响了L比特。

5. 安全策略

- 改变上例中的命令集，使得Holly只能改变H比特且Lucy只能改变L比特。考虑序列c_s=(Holly，xor0)，(Lucy，xor1)，(Holly，xor1)。设初始状态是(0，0)，则输出就是$0_H1_Ll_H$，其中下标表示输出的安全等级。

- 令G’={Lucy}，A=φ。

  $\pi_{Holly}(c_s)$=(Lucy, xor1), proj(Lucy, $\pi_{Holly}(c_s)$, σ_0)=1 。 $\pi_{Holly}(c_s)$=(Lucy, xor1), proj(Lucy, $\pi_{Holly}(c_s)$, σ_0)=1。且{Holly}: | {Lucy}成立。

  现在有再次从直观上看，这也是应该的，因为Holly所做的动作对Lucy可以看到的系统部分没有任何影响。

- 定义5 安全策略就是一个由不干涉属性断言所组成的集合。

- **定义5A　r设是D×D上的一个自反关系，那么r就定义了一种安全策略。**
  - 关系r定义了信息的流动方式。如果$d_i r d_j$，则表示信息可以从域d_i流动到d_j，否则就不可以。
  - 因为信息可以在同一个保护域内流动，所以有$d_i r d_i$。
  - 注意这个定义与安全策略的内容无关，它仅仅定义了什么是“策略”。

- **定义3A　设$d \in D$，$c \in C$，$c_s \in C^*$，那么**
  - $\pi'_d(v) = v$,　　如果v是空序列
  - $\pi'_d(c_s c) = \pi'_d(c_s)c$，　　当 dom(c) r d,
  - $\pi'_d(c_s c) = \pi'_d(c_s)$,　　否则

这个定义的意思是：如果c的执行会影响到保护域d，那么c对于d就是“可见”的。否则，对d来说所得命令序列产生的结果就像c没有被执行一样。

- 定义4A　设一个系统有一个域的集合D。那么，如果

$$P^*(c, T^*(c_s, \sigma_0)) = P^*(c, T^*(\pi_d'(c_s), \sigma_0))$$

其中d＝dom(c), 则称该系统在策略r下是不干涉安全的。

- 定义2A 设c∈C且dom(c) ∈D，并用$\sim^{dom(c)}$表示系统X中状态的等价关系。如果

$$\sigma_a \sim^{dom(c)} \sigma_b \text{ 蕴涵 } P(c, \sigma_a) = P(c, \sigma_b)$$

那么$\sim^{dom(c)}$是输出一致的。

对于dom(c)中的主体，如果执行c后的两个状态的输出投影相同，则两个状态是输出一致的。

引理　设$T^*(c_s, \sigma_0) \sim^d T^*(\pi_d'(c_s), \sigma_0)$。那么如果$\sim^d$是输出一致的，则系统X关于策略r是不干涉安全的。

证明　对于任意一个$c \in C$，记$d=dom(c)$。对该引理的假设部分应用定义$\sim^d$，则

$$P^*(c, T^*(c_s, \sigma_0)) = P^*(c, T^*(\pi_d'(c_s), \sigma_0))$$

而这就是关于r的不干涉安全的定义。

- 定义 设r是一种策略，如果dom (c)不干涉d∈D蕴涵$\sigma_0 \sim^d T(c, \sigma_0)$，则称系统X局部满足r。
- 如果在策略r下，命令c对域d没有任何作用，则对当前状态应用命令c，对于域d将显示不出任何效果。如果这句话成立，则这个系统局部满足策略r。

- 定义 设r是一个策略，$d \in D$。如果$\sigma_a \sim^d \sigma_b$蕴涵$T(c, \sigma_a) \sim^d T(c, \sigma_b)$，则系统X在策略r下是传递一致的。
- 传递一致性仅仅意味着对于所有命令c，系统状态对于域d保持等价。

- 定理　展开定理。设r是一个策略，X是一个系统，它满足输出一致性和传递一致性，且系统局部满足策略r。则X关于策略r是不干涉安全的。

- Rushby给出了一个应用展开定理的简单例子，其目标是将这种理论应用到一个使用静态访问控制矩阵的系统中。问题是：给出的条件是否足以得到不干涉安全。
- 系统是由主体和客体组成的。客体分布在存储有数据的内存中。访问控制矩阵控制着数据的读和写。系统处在一个特定的状态中，这个状态把客体的值封装在访问控制矩阵中。

- 特别地，设$L=\{l_1, l_2, \cdots, l_m\}$是在内存或磁盘中的客体的集合，$V=\{v_1, v_2, \cdots, v_n\}$是客体可能用到的值的集合。一般情况下，状态的集合是$\sum=\{\sigma_1, \sigma_2, \cdots, \sigma_k\}$，保护域集合$D=\{d_1, d_2, \cdots, d_j\}$。
- value: $L \times \sum \rightarrow V$
  - 返回系统处在某个给定的状态时存储在给定客体中的值。
- read: $D \rightarrow P(L)$
  - 返回在指定的某个域下可以观察到的客体的集合，其中P(V)代表V的幂集。
- write: $D \rightarrow P(L)$
  - 返回在指定的某个域下可以写的客体的集合。

访问控制需求

对所有的d∈D, 定义状态的等价关系$\sim^d$:

$\sigma_a \sim^d \sigma_b$ *iff* $(\forall\, l_i \in \mathbf{read(d)}\ (\ value(\ l_i, \sigma_a) = value(l_i, \sigma_b)\ ))$

1. 在保护域dom(c)中执行命令c后得到的输出只依赖于dom(c)中的主体可以读的那些值:

$$\sigma_a \sim^{dom(c)} \sigma_b \Rightarrow \quad P(c, \sigma_a) = P(c, \sigma_b)$$

2. 如果命令c改变客体l_i中的值，那么c只能使用在集合read(dom(c))中的客体的值来得到新的值:

$\sigma_a \sim^{dom(c)} \sigma_b$ ***and***

$value(\ l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a)$ ***or*** $value(\ l_i, T(c, \sigma_b)) \neq value(l_i, \sigma_b)$

$\Rightarrow$

$value(\ l_i, T(c, \sigma_a)) = value(l_i, T(c, \sigma_b))$

- 如果在域dom(c)看来，状态σ_a和状态σ_a是等价的，即$\sigma_a \sim^{dom(c)} \sigma_b$，好像应该有

$$value(l_i, T(c, \sigma_a)) = value(l_i, T(c, \sigma_b))$$

- 但是如果$l_i \notin read(dom(c))$, 则可能虽然$\sigma_a \sim^{dom(c)} \sigma_b$仍然有

$$value(l_i, \sigma_a) \neq value(l_i, \sigma_b)$$

而如果c不改变l_i的值，则必然有

$$value(l_i, T(c, \sigma_a)) \neq value(l_i, T(c, \sigma_b))$$

- 因此，当$l_i \notin read(dom(c))$时，应该考虑c改变l_i的值, 即

$$value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a) \text{ or } value(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_a)$$

3. 如果命令c改变客体l_i中的值， 那么dom(c)就对执行c的主体提供了对l_i的写权限：

$$\text{value}(l_i, T(c, \sigma_a)) \neq value(l_i, \sigma_b) \Rightarrow l_i \in \text{write}(\text{dom}(c))$$

定理 设系统X满足以上3个访问控制需求，并且满足

(1) 设u, v ∈ D, 那么u r v ⇒*read* (u) ⊆*read* (v)

(2) l_i ∈ read(u) 且 l_i ∈ write(v) ⇒ v r u

则系统X关于策略r是不干涉安全的。

证明： 见 John Rushby Noninterference, Transitivity, and Channel-Control Security Policies, Technical Report CSL-92-02, December 1992. 的pp14-15 或Matt Bishop, 计算机安全学——安全的艺术与科学

- 条件(1)是说如果u能干涉v，由于v需要从u中读取某些信息，因而在保护域u中可以读的每个客体，也可以在保护域v中读。根据以上假设，如果一个客体在u中不可读而在v中可读，并且u中存在另外一些客体在v中可读，则信息就可以从前一个客体流到另外一个客体中，从而从域u中流出。
- 条件(2)是说如果一个主体可以读一个域v中的客体，且另一个主体可以在u中读同一个客体，那么域v就可以干涉域u。

11. 权限管理基础设施
(Privilege Management Infrastructure, PMI)

PMI 的背景 （1）

- 大型的组织机构的问题
  - 网络结构比较复杂，应用系统比较多。
  - 如果分别对不同的应用系统采用不同的安全策略，则管理将变得越来越复杂，甚至难以控制。
- 访问控制的体系
  - 机构统一权限管理和发布
  - 访问控制系统根据机构发放的权限控制用户访问。
- PMI之前的状况
  - 访问控制，已经十分成熟。
  - 权限的生命周期管理、权限的表达和权限管理方式方面不够成熟。

PMI 的背景 （2）

- 权限管理存在的问题
  - 权限管理混乱：不同应用系统对统一的数据和人力采用不同的权限分配策略和管理方式。
  - 系统不安全因素：不同的权限管理策略产生的安全强度是不同的，这就可能造成机构信息安全管理的漏洞。
  - 权限管理依赖于访问控制应用
  - 权限的发放和控制紧密耦合。权限的拥有者发放权限，而由资源的保护者验证权限。

PMI 的背景 （3）

- 1997年，X.509(V3)中定义了基本的属性证书语法 (属性证书第1版).
- 2000年发布的X.509(V4)中定义了PMI的框架结构，
  - 定义了扩展属性证书语法 (第2版)。
  - 定义了PMI模型
  - 规定了委托路径处理
  - 定义了标准PMI扩展集
  - 并增加了目录服务对象

PMI的定义

- PMI权限管理基础设施或授权管理基础设施，是属性证书、属性权威、属性证书库等部件的集合体，用来实现权限和证书的产生、管理、存储、分发和撤销等功能。
- AA(Attribute Authority)，即属性权威，用来生成并签发属性证书的机构，它负责管理属性证书的整个生命周期。
- AC(Attribute Authority)，即属性证书，对于一个实体权限的绑定是由一个被数字签名了的数据结构。

PKI和PMI之间的主要区别

- 作用不同
  - PMI主要进行授权管理，证明这个用户有什么权限，能干什么。
  - PKI主要进行身份鉴别，证明用户身份。
- 生存期不同
  - 身份证书的生存期较长，身份证书的申请、审核、签发的代价是较高的。
  - 权限信息往往较短。
- 权威者不同

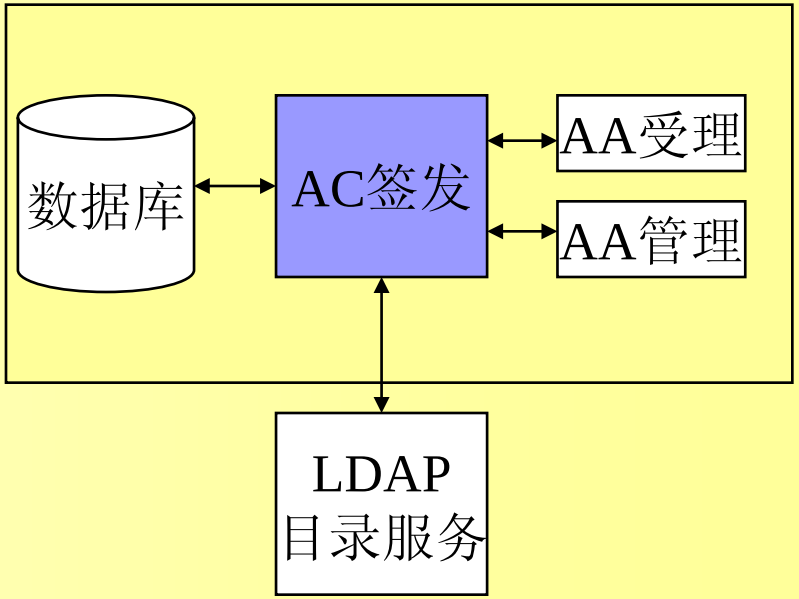
属性权威

- AA (Attribute Authority) 属性权威
  - 用来生成并签发属性证书的机构，它负责管理属性证书的整个生命周期。
- AC签发服务

  AC签发服务是属性权威AA的主体，是以PKI技术为基础，以授权服务为主要任的服务模块，用于签发属性证书。该服务可以由独立的一台服务器提供，也可以是证书签发模块。
- AA受理

  重要用于接受并验证对属性证书的请求，并提供基于属性证书的授权服务。
- AA管理：管理属性权威AA
- 数据库：用于存储用户和资源的基本信息
- LDAP目录服务器

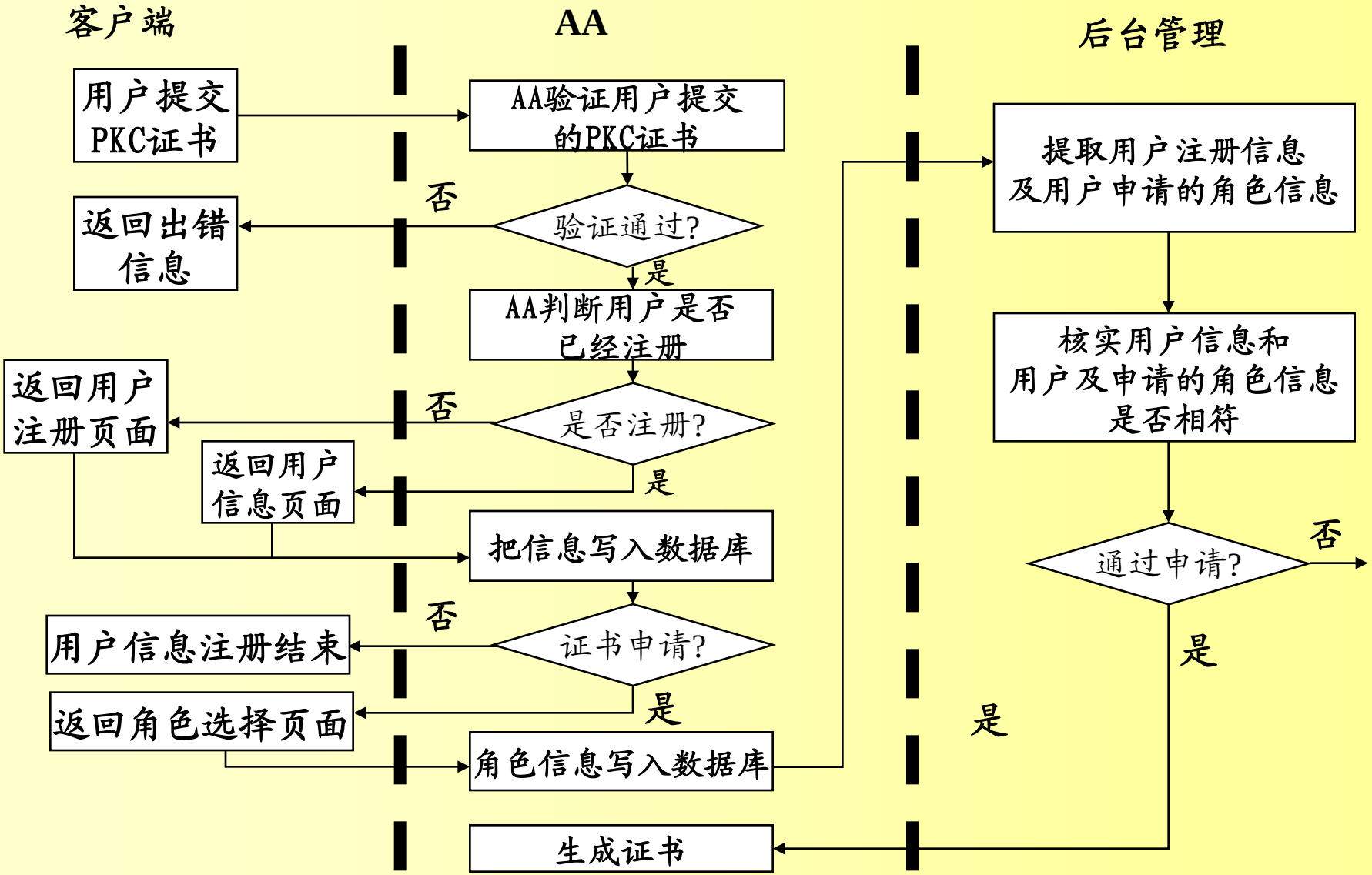
  用于发布PMI用户的属性证书及属性证书的撤销列表ACRL（Attribute Certificate Revocation List）。



属性证书的定义与格式

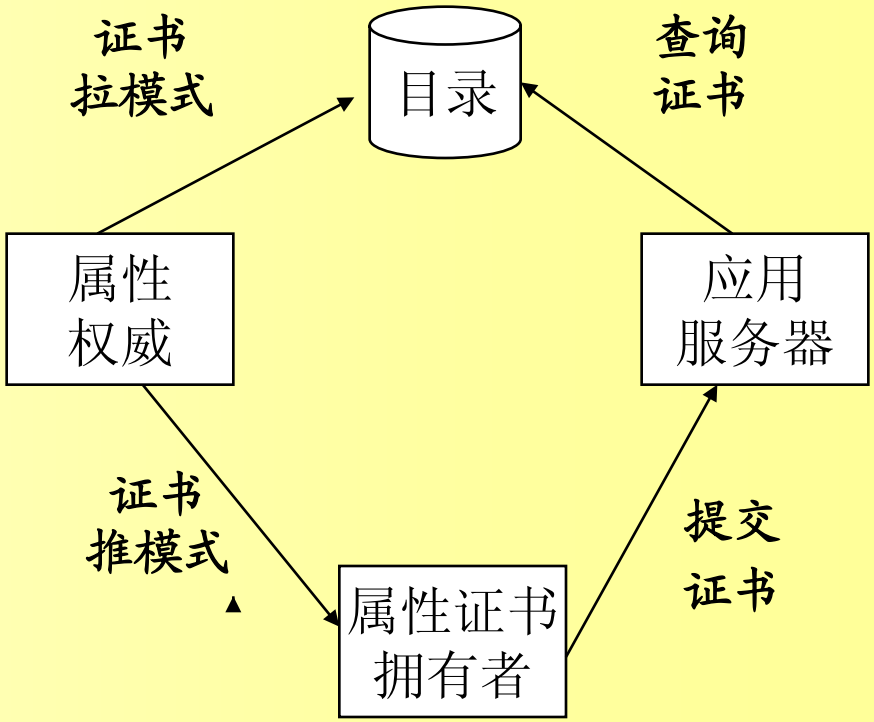
参数	说明
版本号	属性证书的版木号
持有者	属性证书的持有者信息(持有者公钥证书DN和公钥证书颁发者DN的组合)
颁发者	属性证书的颁发者信息(颁发者公钥证书DN)
签名算法	属性证书使用的签名算法
序列号	属性证书的序列号
有效期	属性证书的生效和失效日期
属性	属性证书持有者的属性信息
扩展项	定义诸如证书颁发者密钥标识符、CRL分布点等
签名信息	属性证书签发者对属性证书的签名

属性证书的申请与发布


客户端
AA
后台管理
用户提交PKC证书
AA验证用户提交的PKC证书
提取用户注册信息及用户申请的角色信息
返回出错信息
否
验证通过?
是
AA判断用户是否已经注册
核实用户信息和用户及申请的角色信息是否相符
返回用户注册页面
否
是否注册?
返回用户信息页面
是
否
把信息写入数据库
通过申请?
否
用户信息注册结束
否
证书申请?
是
返回角色选择页面
是
是
角色信息写入数据库
生成证书

属性证书的分发流程

- **第一种是推模式**
  - 当用户要求访问资源时，**由用户自己直接提供其属性证书**，即用户将自己的属性证书“推”给资源服务管理器。
  - 在客户和服务器之间不需要建立新的连接，而且对于服务器来说，这种方式不会带来查找证书的负担，从而减少了开销。
- **第二种是拉模式**
  - **业务授权机构将属性证书发布到目录服务系统**，当用户需用到属性证书时，由服务器从属性证书发放者 (属性权威AA)或存储证书的目录服务系统"拉"回属性证书。这种"拉"模式的一个主要优点在于实现这种模式，不需要对客端及客户-服务器协议做任何改动。



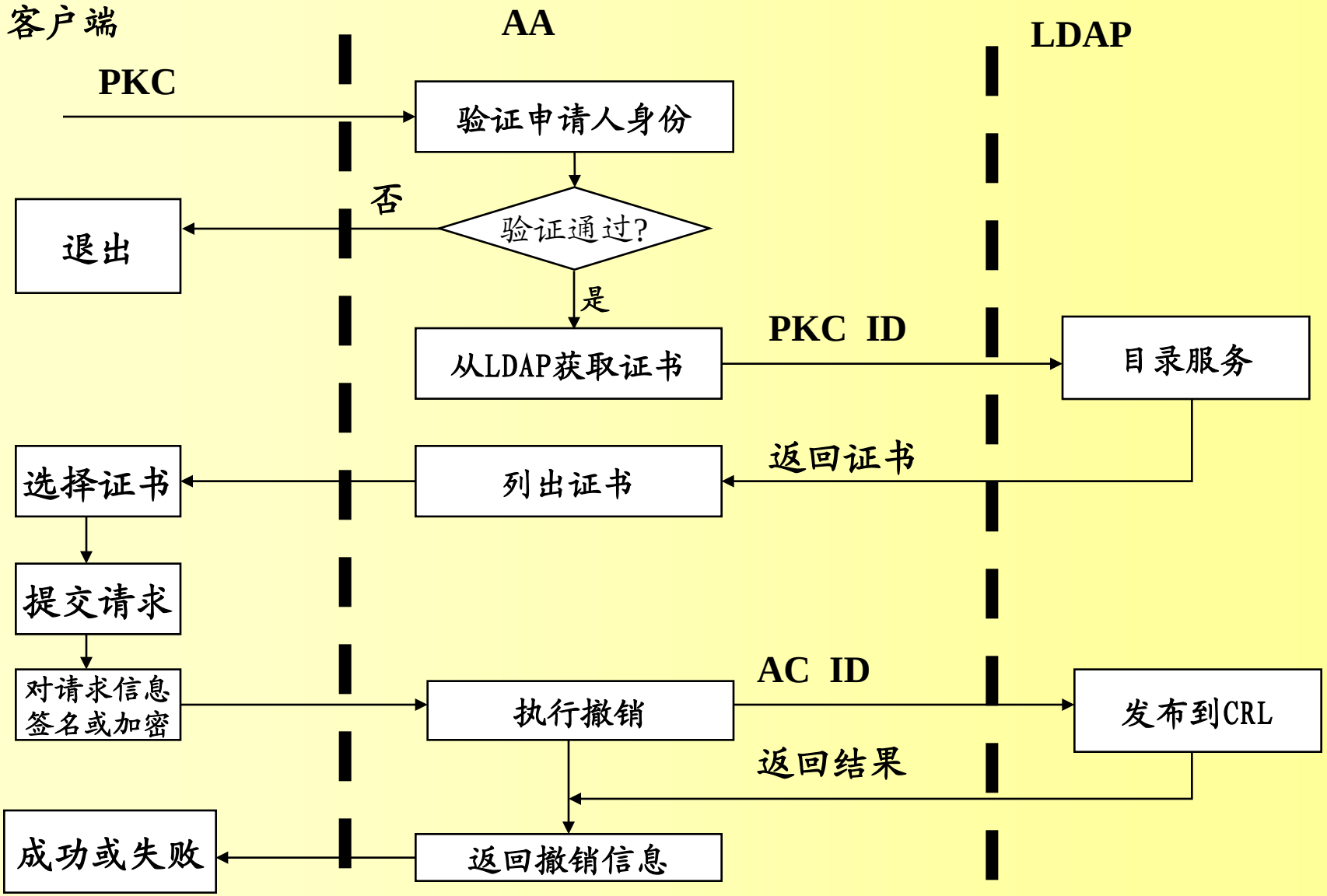
属性证书的基本验证过程

- 验证用户的身份证书
  - 用户提交身份证书，应用系统验证证书的有效性，包括查看是否经过CA的正确签名，并确定签发证书CA的合法性等。
- 获取属性证书
  - 通过身份验证之后，通常便用加密的询问应答模式或其他模式，根据身份获取用户属性证书。
- 验证属性证书，
  - 应用程序获得了属性证书后，验证其合法性，确定发放证书机构的证书签名是否有效，证书是否过期，更重要的是确定用户身份是否与证书声明的身份一致。
- 应用程序检查属性证书中的内容
  - 确定是否允许此用户存取其所需的资源及服务。

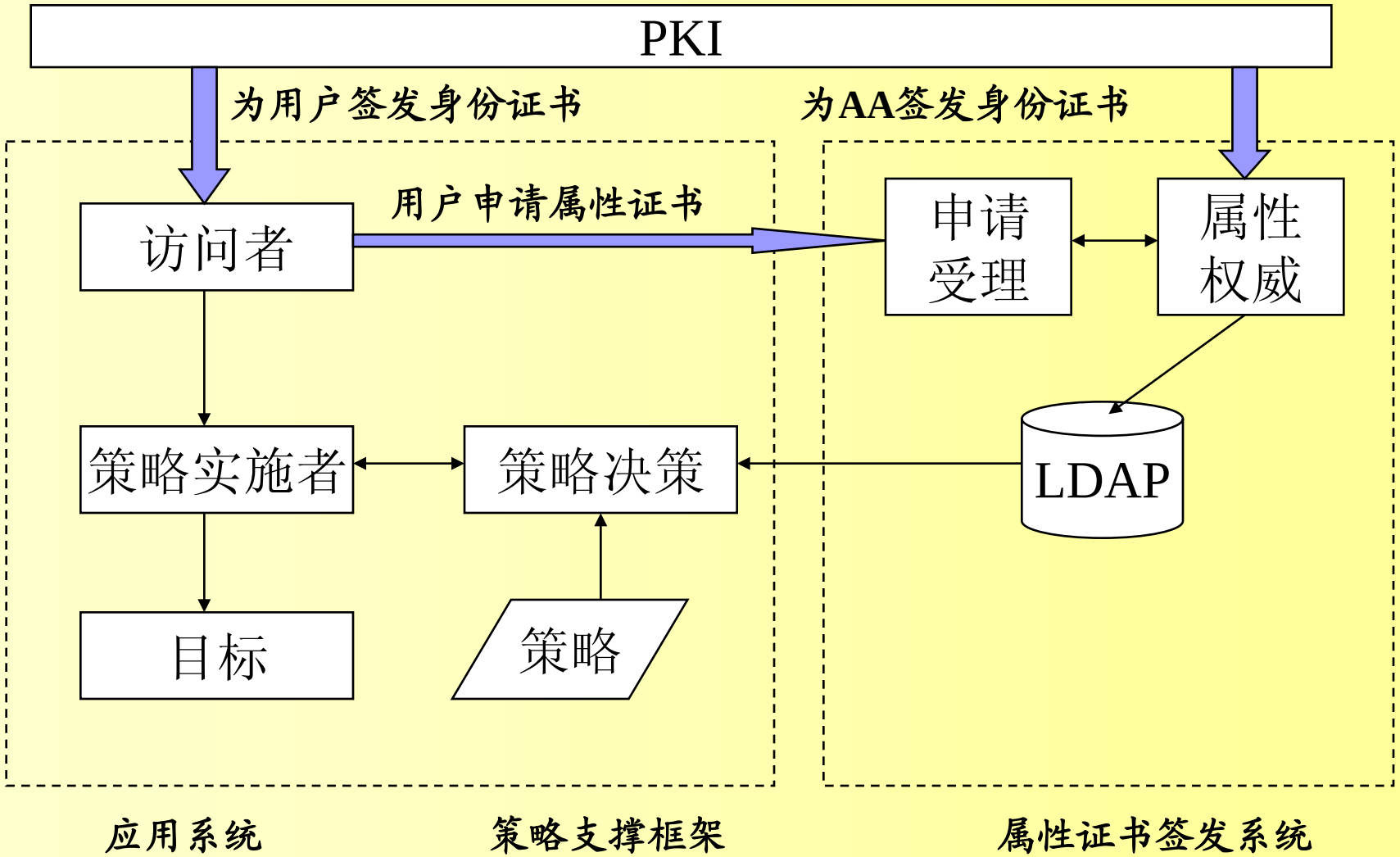
属性证书的特点

- 分立的发行机构
  - 通过一个法定的集中权威机构，来发放身份证书。
  - 通过一个局域的、熟知用户属性的组织来发放属性证书。
  - 一个人可以拥有好几个属性证书，但每一个都会与唯一的身份证书关联，**几个属性证书可以来自不同的机构**。
- 存储介质
  - 用户持有属性证书：建立相应的协议来使用属性证书；用户必须选择与具体应用对应的属性证书；当用户权限改变时需要更新自己的属性证书。
  - 系统托管属性证书：应用系统根据用户的身份直接从属性库中获得用户相应的属性证书；属性证书的使用和变更对用户是透明的；**用户的权限增加立刻就会得到认可**。
  - 权限的生存期通常比较短，相对来说，属性证书的更新是频繁的。由用户自己管理属性证书往往不如由系统托管方便。

属性证书的撤销过程


客户端
AA
LDAP
PKC
验证申请人身份
否
退出
验证通过?
是
PKC ID
从LDAP获取证书
目录服务
返回证书
选择证书
列出证书
提交请求
对请求信息
签名或加密
AC ID
执行撤销
发布到CRL
返回结果
成功或失败
返回撤销信息

PKI / PMI 应用的逻辑结构



12. The Generalised Framework for Access Control (GFAC)

Objectives

- An improved framework for expressing and integrating multiple policy components.
- The main objectives are
  - Make it easy to state, formalise, and analyse diverse access control policies
  - Make it feasible to configure a system with security policies chosen from a vendor provided set of options with confidence that the system's security policy makes sense and will be properly enforced.
  - Construct the model in a manner that allows one to show that it satisfies an accepted definition of each security policy it represents.

GFAC's premise

- **GFAC** is based on the premise
  - all access control policies can be viewed as rules expressed in terms of attributes by authorities;
- **Authorities:** An authorised agent that
  - defines security policies;
  - identifies relevant security information;
  - assigns values to attributes.
- **Attributes**
  - Characteristics or properties of subjects and objects defined within the computer system for access control decision making;
- **Rule:** A set of formalized expressions
  - define the relationships among attributes and other security information for access control decisions in the computer system;
  - reflecting the security policies defined by authority.

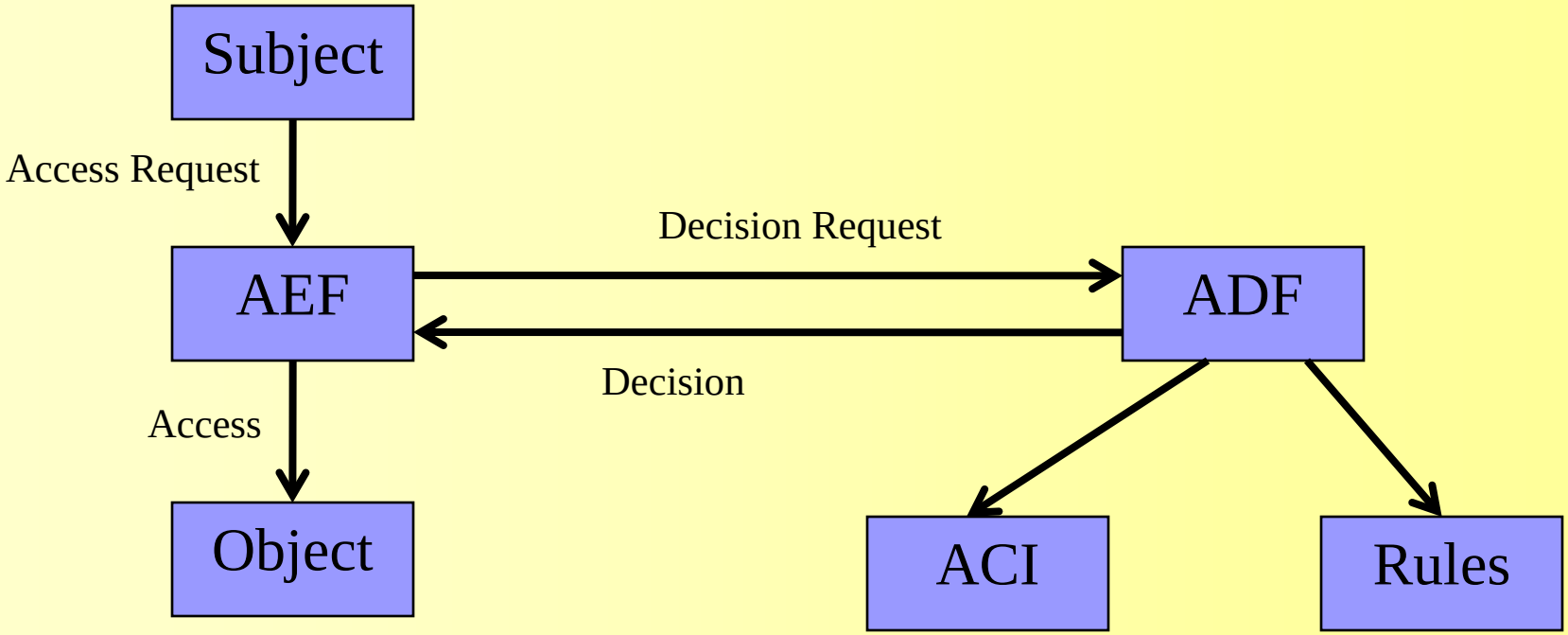
ACI & ACR

- Access control information (ACI)
  - security attributes and other access control data;
- Access control rules (ACR).
  - the rules that implement the trust policies of a system

Adjudication and Enforcement

- The agent that adjudicates access control requests is called access control decision facility (ADF) ;
  - ADF corresponds to the access control rules within the TCB that embody the system´s security policy.
- The agent that enforces the ADF´s decision is called access control enforcement facility (AEF) ;
  - AEF corresponds to the system functions of the trusted computing base (TCB)

Overview of GFAC



Rule-Set Modelling approach

- In traditional security modeling approaches, the security model rules describe both access policy and system behavior ;
- The rule set modelling approach separates the decision criteria (**access rules**) from the state transition (**system operations**)
- AEF corresponds to a model of the system operations, called **state-machine model**
  - abstractly defines the interface of processes to the TCB;
- ADF corresponds to a policy model, called the **rule-set model**
  - defines the security policies of the trusted computer system.