第3讲

信息安全模型

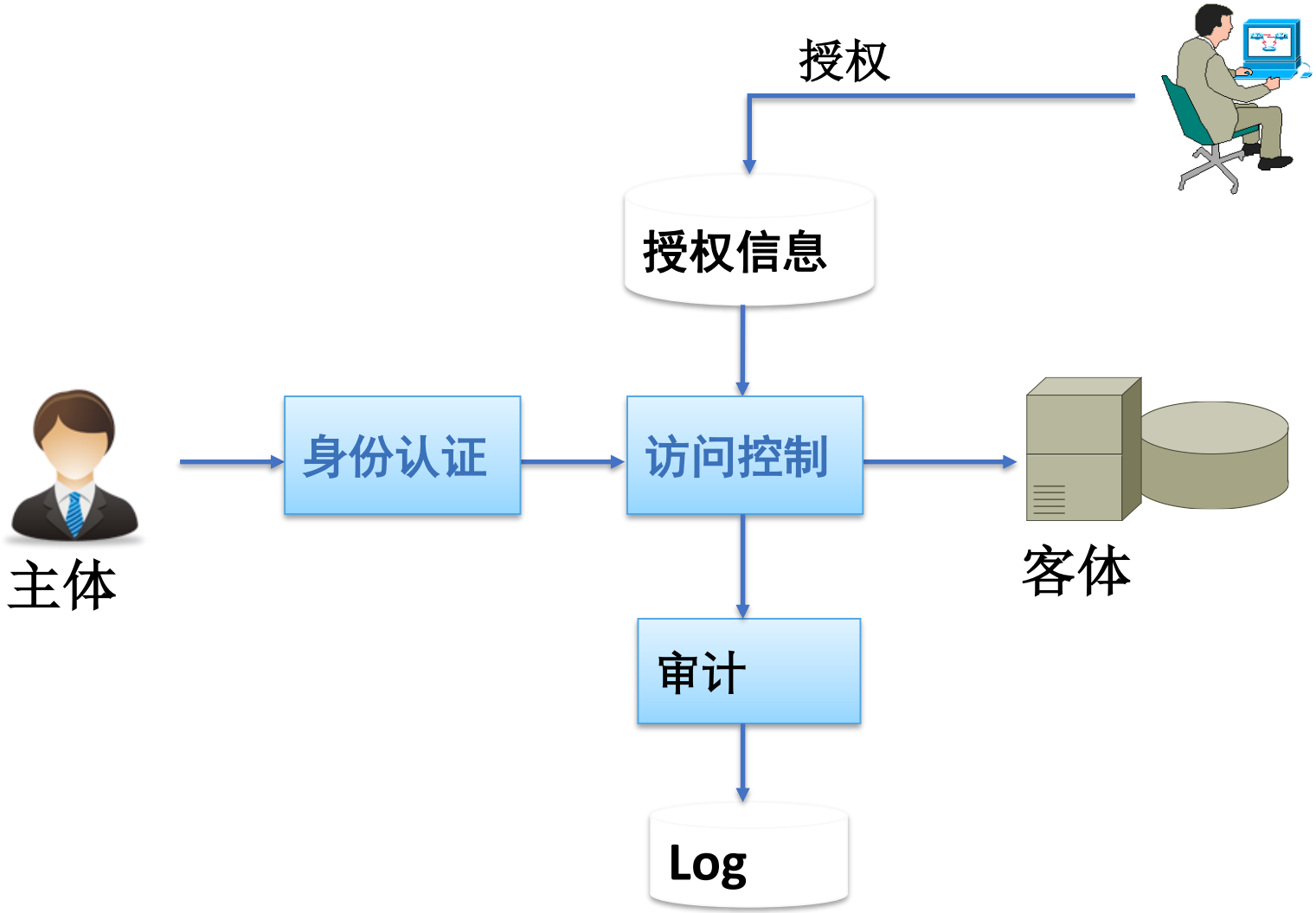
大纲

- 经典安全模型
- 自主访问控制
- 强制访问控制
- 基于角色的访问控制
- 完整性度量模型

经典安全模型

- James P.Anderson在1972年提出的引用监控器（The Reference Monitor）的概念是经典安全模型的最初雏形

经典安全模型



经典安全模型

- 谁可以进入系统?
  - **Authentication (身份认证)**
- 用户可以做什么?
  - **Authorization (授权控制)**
- 用户做了什么?
  - **Audit (审计)**

AAA (3A)

访问控制策略的实施原则

- **最小特权原则**
  - 当主体执行操作时，按照主体所需特权的最小化原则分配给主体权力。
- **最小泄漏原则**
  - 当主体执行任务时，按照主体所需知道的信息的最小化的原则分配给主体权力。
- **多级安全策略**
  - 主体和客体间的数据流向和权限控制按照安全级别的绝密、秘密、机密、限制和无级别这五个级别来划分。优点是可避免敏感信息的扩散。

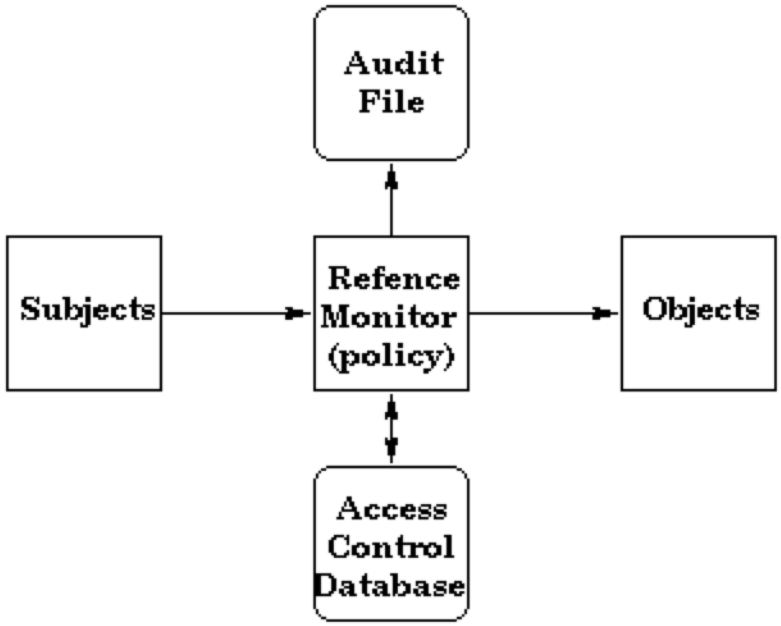
访问控制策略的实施原则

- **最小特权原则**
  - 当主体执行操作时，按照主体所需权力的最小化原则分配给主体权力。
- 如：综合教务管理系统中，教员查询学生信息权限，不需授予更新权限

- 最小泄漏原则
  - 当主体执行任务时，按照主体所需知道的信息最小化的原则分配给主体权力。
  - 例：综合教务管理系统中教员只能查询所授班次学生的信息

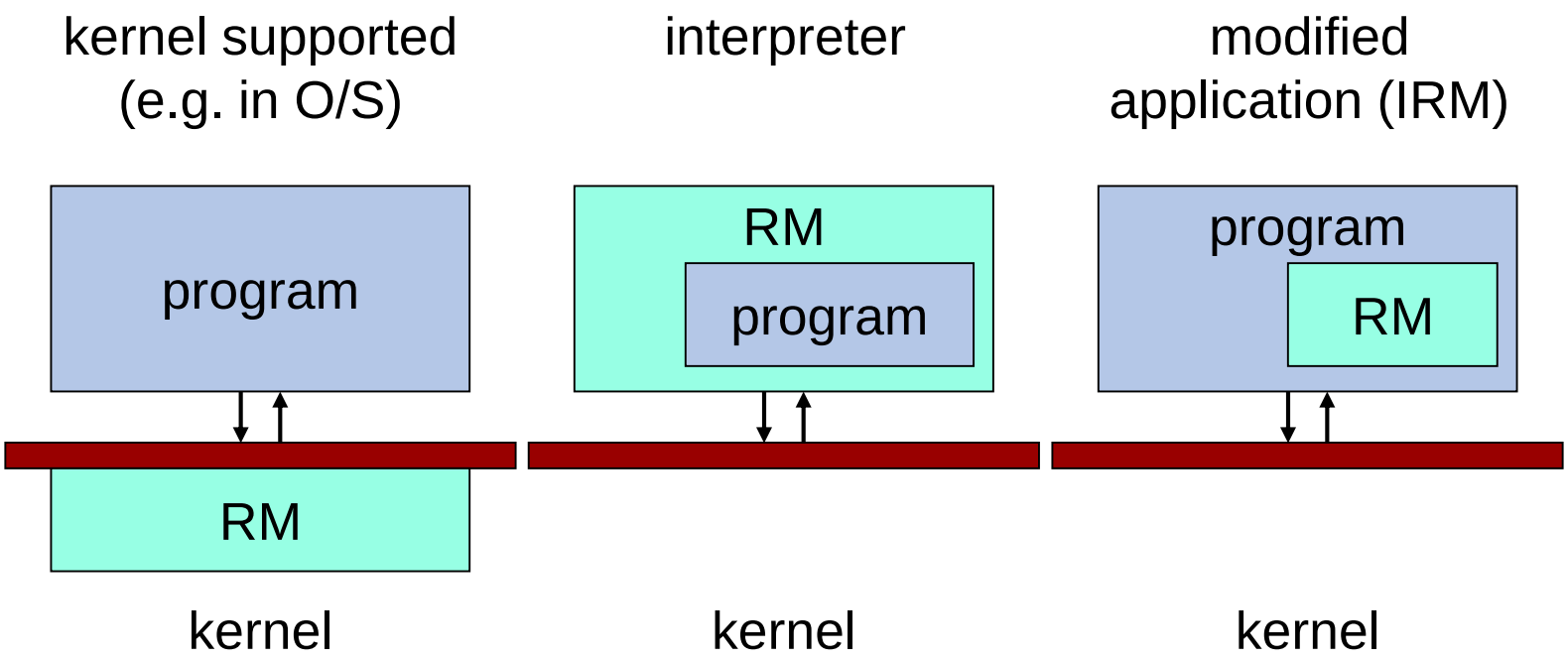
The Reference Monitor

- 一个抽象的保护模型
  - ✓有时相当接近于实现
  - ✓例如：SELinux (Flask 架构)
- 在实践中应提供:
  - ✓完全仲裁(Complete mediation)
  - ✓不可篡改(Be tamper proof)
  - ✓可证明(Be small enough to understand , i.e., verify)



- Idea: 计算机系统一般巨大而复杂，与安全有关的部分往往很小，把涉及安全的部分提取出来，以便可以理解/验证它。

RM – Design Choices



Trusted Computing Base (TCB)

- 计算机系统内的全部保护机制---包括硬件、固件和软件---它们的组合负责执行安全策略；
- TCB由一个或多个组件组成，它们共同对一个产品或系统执行统一的安全策略；
- TCB正确执行安全策略的能力，完全取决于TCB内的机制以及系统管理人员对安全策略相关参数的正确输入。

自主访问控制与安全策略

- **定义：** 如果作为客体的拥有者的用户个体可以通过设置访问控制属性来准许或拒绝对该客体的访问，那么这样的访问控制称为自主访问控制。

- **定义：** 如果普通用户个体能够参与一个安全策略的策略逻辑的定义或安全属性的分配，则这样的安全策略称为**自主安全策略**。

自主访问控制

- **自主访问控制的实现机制**
  - **访问控制矩阵（Access Control Matrix）**
  - **访问控制列表（ Access Control Lists）**
  - **访问控制能力列表(Access Control Capabilities Lists,ACCLs)**

访问控制矩阵

- 访问控制矩阵（Access Control Matrix）
  - 是最初实现访问控制机制的概念模型
  - 它利用二维矩阵规定了任意主体和任意客体间的访问权限。

主体＼客体	文件 1	文件 2	文件 3
Alice	*R*、*W*、*Own*	*R*	*R*、*W*
Bob	*R*	*R*、*W*、*Own*	
John	*R*、*W*	*R*	*R*、*W*、*Own*

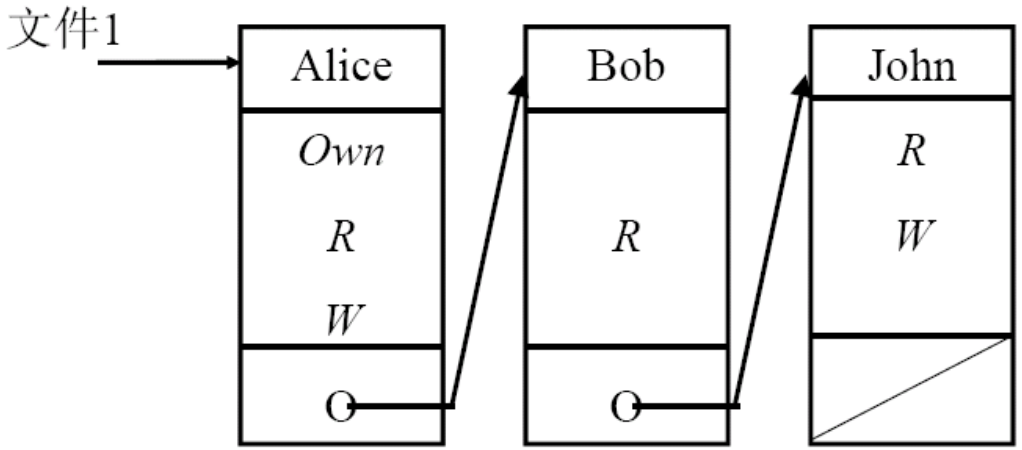
— 文件1: { (Alice, rxo) (Bob, r) (John, rx) }

— 文件2: { (Alice, r) (Bob, rwo) (John, r) }

— 文件3: { (Alice, rw) (John, rwo) }

访问控制表

- 访问控制表（ACL，Access Control List）是以文件（客体）为中心建立的访问权限表。
- 优点：能够很容易的判断出对于特定客体的授权访问，哪些主体可以访问并有哪些访问权限。



访问控制能力表

主体＼客体	文件 1	文件 2	文件 3
Alice	*R*、*W*、*Own*	*R*	*R*、*W*
Bob	*R*	*R*、*W*、*Own*	
John	*R*、*W*	*R*	*R*、*W*、*Own*

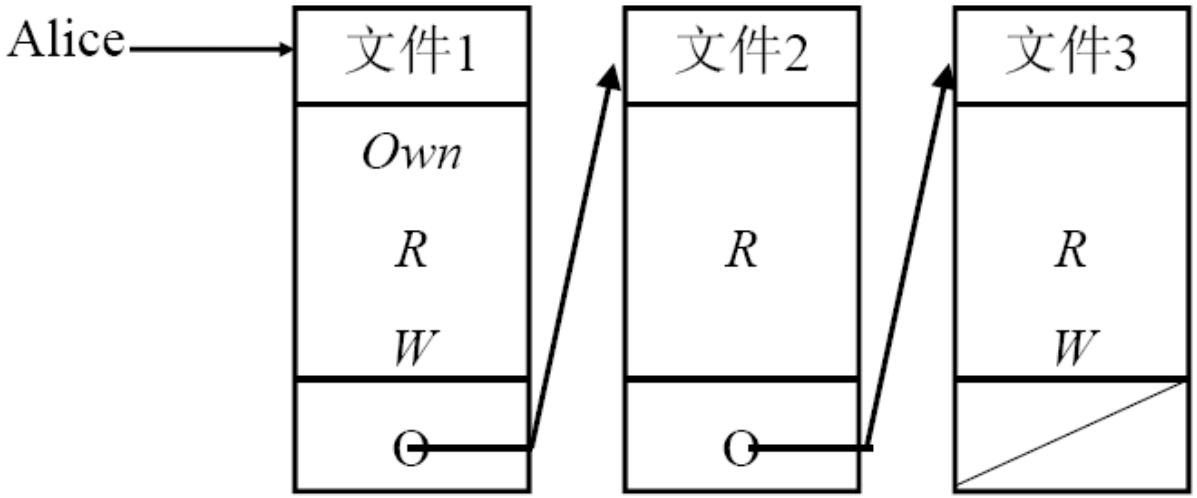
— Alice: { (文件1, rwo) (文件2, r) (文件3, rw) }

— Bob: { (文件1, r) (文件2, rwo) }

— John: { (文件1, rw) (文件2, r) (文件3, rwo) }

访问控制能力表

- 访问控制能力表（ACCL，Access Control Capabilities List）
  - 是以用户（主体）为中心建立的访问权限表。
  - 为每个主体附加一个该主体能够访问的客体明细表

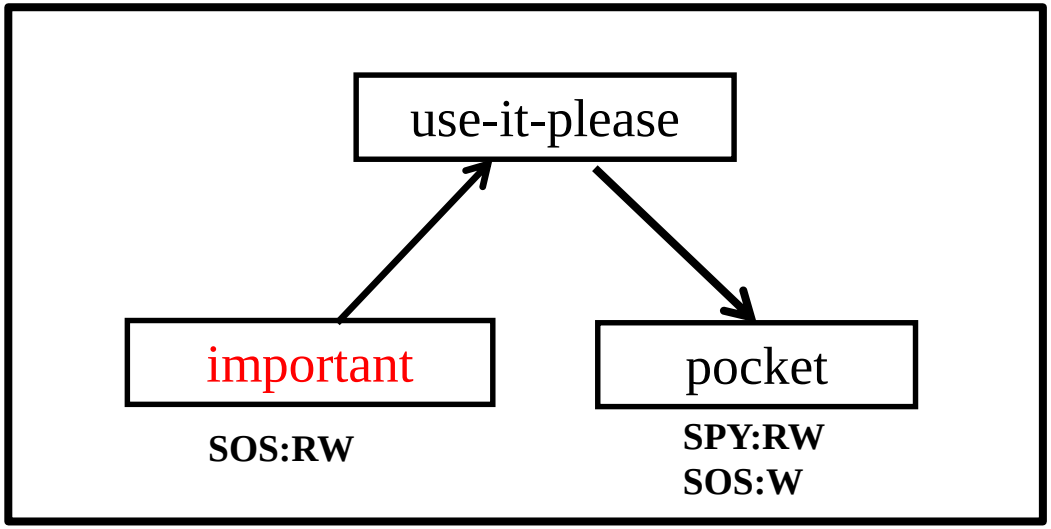


自主访问控制问题

- DAC提供的安全保护容易被非法用户绕过
  - 例如，若某用户A有权访问文件F，而用户B无权访问F，则一旦A获取F后再传送给B，则B也可访问F。
  - 原因：在自主访问控制策略中，用户在获得文件的访问后，并没有限制对该文件信息的操作，即并没有控制数据信息的分发。

优点：高度的灵活性、简单性、个性化权限分配
缺点：安全性低、难以维护、权限扩散、难以撤销权限、审计困难

自主访问控制问题



（1）SOS将重要信息放在important文件中，权限为只有自己可读写

（2）SPY是一个恶意攻击者，试图读取important.doc文件内容，他首先准备好一个文件pocket.doc，并将其权限设置成为SOS:w，SPY:rw

（3）SPY设计一个有用的程序use_it_please，该程序除了有用部分，还包含一个木马

（4）当诱使SOS下载并运行该程序时，木马会将important.doc中的信息写入pocket.doc文件，这样SPY就窃取了important.doc的内容

自主访问控制问题

- 没有用户与主体分离
- 没有控制信息流动

当用户被信任遵守访问限制时，代表他的主体（进程）却没有

- 恶意代码问题，即特洛伊木马。**自主访问控制无法防范特洛伊木马**

- 应该对主体（进程）自身可以执行的操作实施限制
- 强制访问控制策略提供了一种通过使用标签来强制执行信息流控制的方法

自主访问控制问题

- DAC的优缺点
  - 优点：授权机制比较灵活
  - 缺点：存在较大的安全隐患，不能对系统资源提供充分保护，尤其不能抵御特洛伊木马的攻击

强制访问控制与安全策略

- **定义：**如果只有系统才能控制对客体的访问，而用户个体不能改变这种控制，那么这样的访问控制称为强制访问控制。

- **定义：** 如果一个安全策略的策略逻辑的定义与安全属性的分配只能由系统安全策略管理员进行控制，则该安全策略称为强制安全策略。

强制访问控制

- 强制访问控制(Mandatory Access Control)，简称MAC
  - 在强制访问控制中，每个主体及客体都被赋予一定的安全级别(包括安全域)，系统通过比较主体和客体的安全级别来决定主体是否可以访问该客体。
  - 如，绝密级，机密级，秘密级，无密级。
  - 通过安全标签实现单向信息流通模式。
  - 系统“强制”主体服从访问控制策略。

普通用户不能改变自身或任何客体的安全级别，即不允许普通用户确定访问权限，只有系统管理员可以确定用户的访问权限。

强制访问控制实现机制-安全标签

- 安全标签是定义在目标上的一组安全属性信息项。在访问控制中，一个安全标签隶属于一个用户、一个目标、一个访问请求或传输中的一个访问控制信息。
- 最通常的用途是支持多级访问控制策略。
  在处理一个访问请求时，目标环境比较请求上的标签和目标上的标签，应用策略规则（如Bell Lapadula规则）决定是允许还是拒绝访问。

强制访问控制

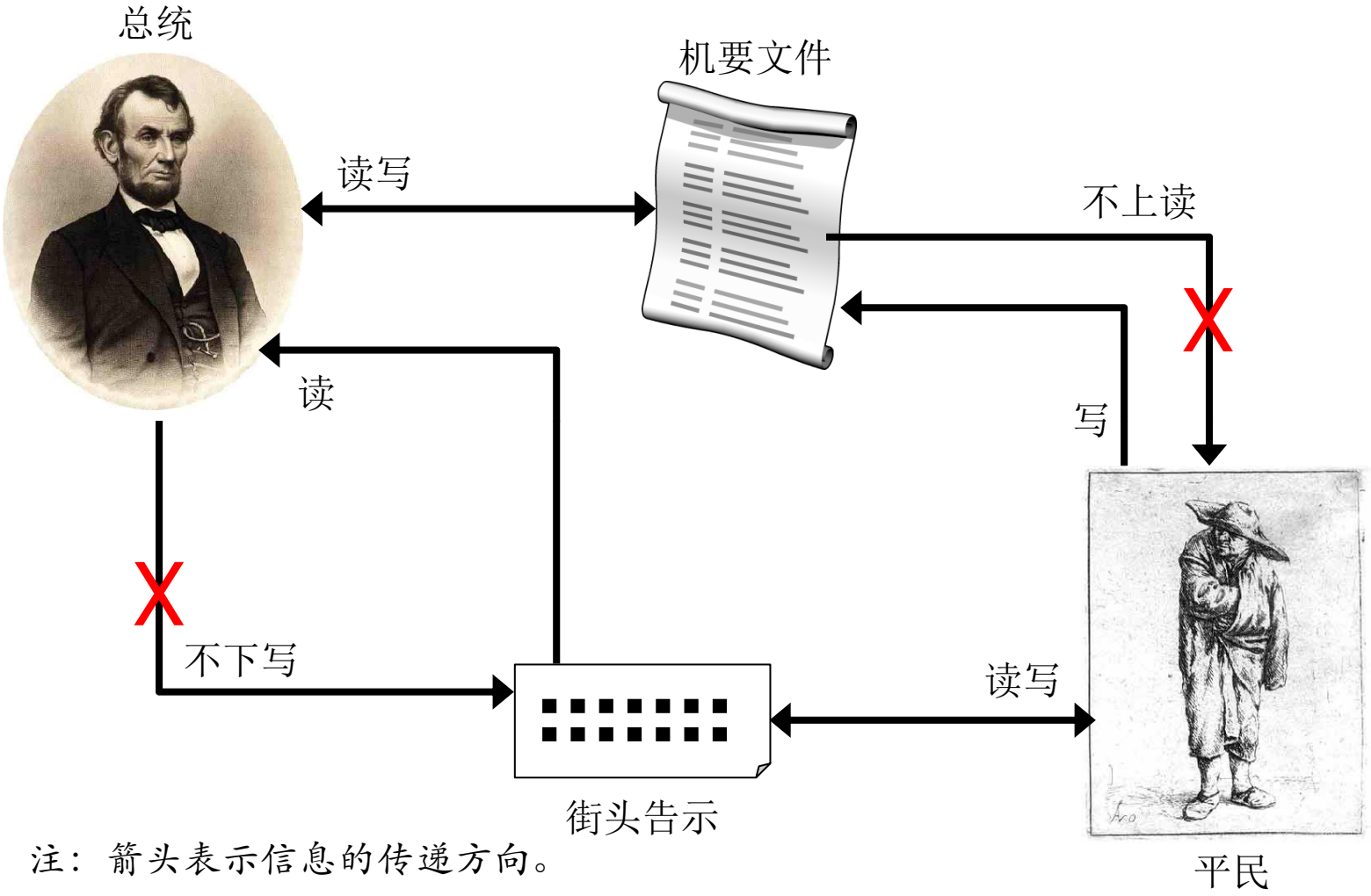
- BLP模型
- Biba模型
- Clark-Wilson模型
- Chinese Wall模型
- TE模型

Bell-LaPadula模型

- 1973年，David Bell和Len Lapadula提出了第一个也是最著名的安全策略模型，Bell-LaPadula安全模型，简称BLP模型。

- BLP模型是遵守军事安全策略的多级安全模型。

BLP模型现实意义



注：箭头表示信息的传递方向。

“不上读，不下写”规定

① 总统可以读和写机要文件，因为他的地位匹配它的敏感性；

② 总统可以读街头告示，因为他的地位高于它的敏感性（可下读）；

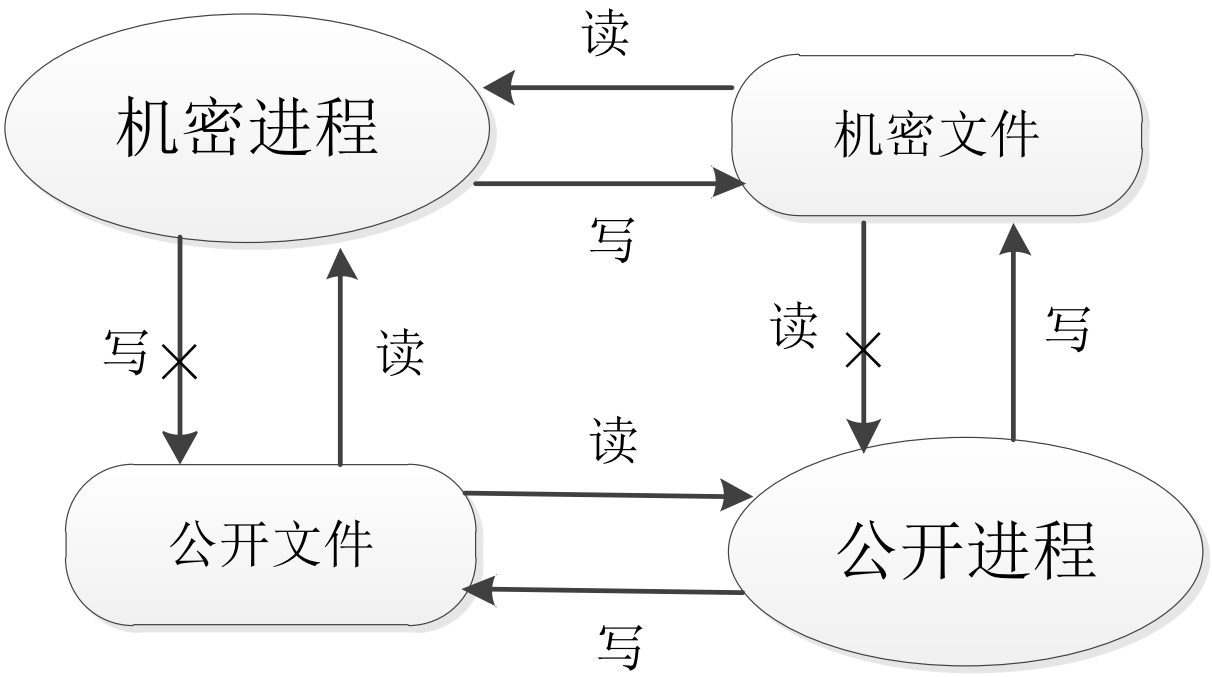
③ 总统不能写街头告示，因为他的地位高于它的敏感性（不下写）；

④ 平民可以读和写街头告示，因为他的地位匹配它的敏感性；

⑤ 平民可以写机要文件，因为他的地位低于它的敏感性（可上写）；

⑥ 平民不能读机要文件，因为他的地位低于它的敏感性（不上读）。

BLP模型



举例

security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Ulaley can only read Telephone Lists

BLP模型的构成

- BLP模型的核心内容由简单安全特性（ss-特性）、星号安全特性（*-特性）、自主安全特性（ds-特性）和一个基本安全定理构成。

简单安全特性

- （ss-特性）如果（*Sub, Obj, r*）是当前访问，那么一定有：

$$f(Sub) \geqslant f(Obj)$$

其中，*f* 表示安全级别，$\geqslant$表示支配关系

星号安全特性

- （*-特性）在任意状态，如果（*Sub, Obj, Acc*）是当前访问，那么一定有：

(1) 若*Acc*是*r*，则：$f_{-c}(Sub) \geqslant f(Obj)$

(2) 若*Acc*是*a*，则：$f(Obj) \geqslant f_{-c}(Sub)$

(3) 若*Acc*是*w*，则：$f(Obj) = f_{-c}(Sub)$

其中，f表示安全级别，f_{-c}表示当前安全级别，$\geqslant$表示支配关系。

r是只读，e是可执行但不可读不可写，w是可读可写，a是只可写

Attribute Elements

The set A ={r, e, w, a} is used in the model for access mode designation with the following meanings:

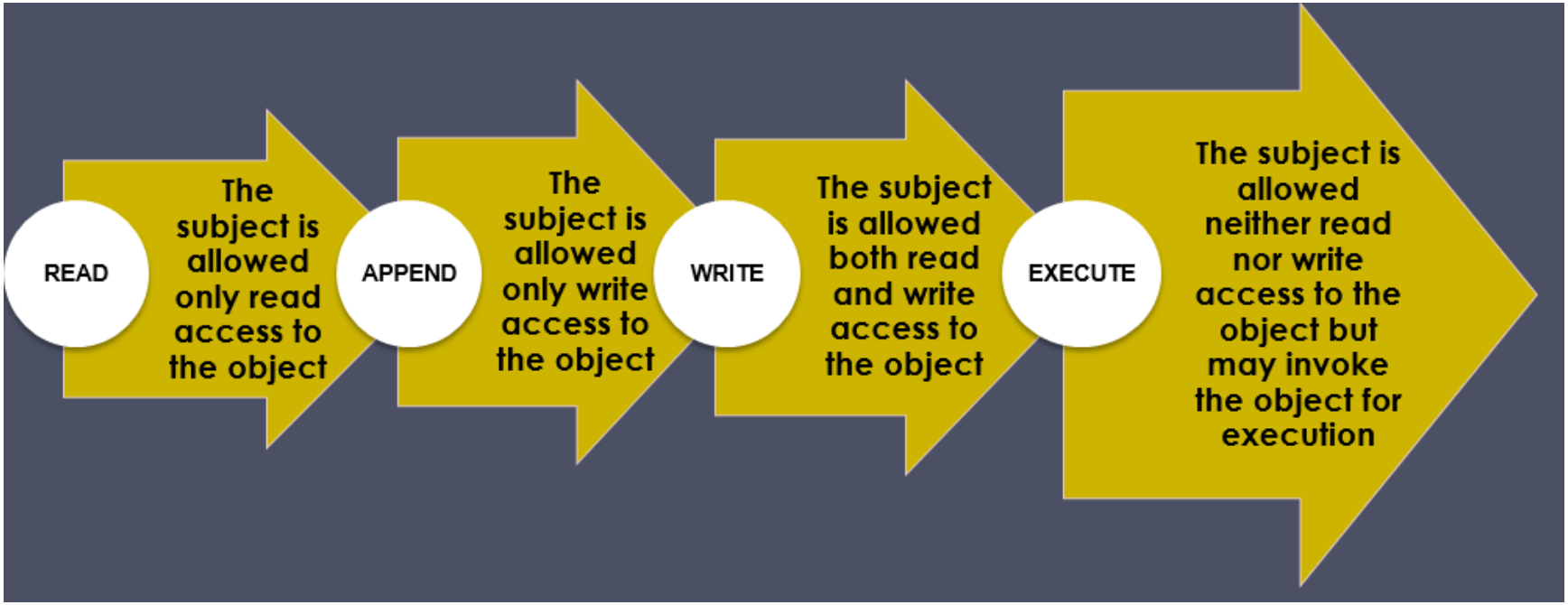
r--read; observe only

e--execute; neither observation nor alteration

w--write; observe and alter

a--append; alter only.

Access Privileges



BLP模型

- 简单安全特性：是指任何一个主体不能读安全级别高于它的安全级别的客体，即不能“向上读”；
- 星号安全特性：“*”特性是指任何一个主体不能写安全级别低于它的安全级别的客体，即不能“向下写”。

上述两个特性是BLP模型的两条主要性质，约束了所允许的访问操作。在访问一个客体时，两个特性都将被验证。

自主安全特性

- （ds-特性）如果（*Sub-i, Obj-i, Acc*）是当前访问，那么，*Acc*一定在访问控制矩阵*M*的元素M_{ij}中。

- ds-特性---处理自主访问控制

基本安全定理

- **基本安全定理：** 如果系统状态的每一次变化都满足ss-特性、*-特性和ds-特性的要求，那么，在系统的整个状态变化过程中，系统的安全性一定不会被破坏。

- 符合状态机模型，即如果我们开始处于一个安全状态，并且所有的状态转换是安全的，那么系统将是安全的。

安全级别定义

- 在BLP模型中，实体*E*的安全级别*level*(*E*)是一个二元组(*L*, *C*)，其中，*L*是等级分类，可用整数表示，*C*是非等级类别，由集合表示。

- 举例:
  - (Top Secret, { NUC, EUR, ASI })
  - (Confidential, { EUR, ASI })
  - (Secret, { NUC, ASI })

支配关系定义

- 设在BLP模型中，实体*E1*和*E2*的安全级别分别为：

  $level(E_1) = (L_1, C_1)$，$level(E_2) = (L_2, C_2)$，

  $\geqslant$表示支配关系，那么：

$$level(E_1) \geqslant level(E_2)$$

当且仅当：

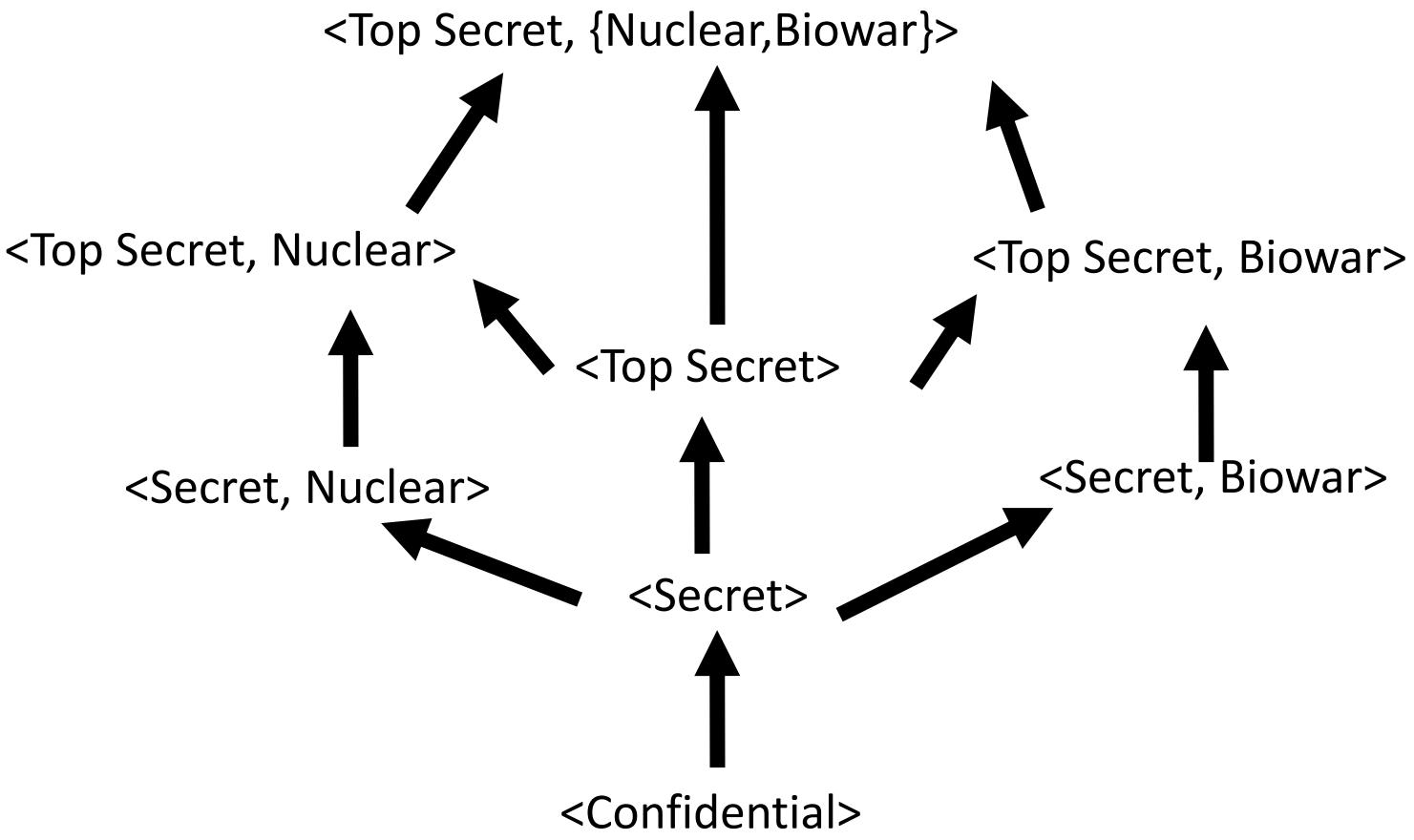
$$L_1 \geqslant L_2 \text{ 且 } C_1 \supseteq C_2$$

支配关系

- $(L, C) \geqslant (L', C')$ iff $L' \leq L$ and $C' \subseteq C$
- 举例
  - (Top Secret, {NUC, ASI}) $\geqslant$ (Secret, {NUC})
  - (Secret, {NUC, EUR}) $\geqslant$ (Confidential,{NUC, EUR})
  - (Top Secret, {NUC}) $\neg \geqslant$(Confidential, {EUR})
- 支配关系构成lattice格
  - *lub*(*Lattice*) = (*max*(*L*), *C*)
  - *Glb*(*Lattice*) = (*min*(*L*), $\varnothing$)

支配关系

- 不上读，不下写


<Top Secret, {Nuclear,Biowar}>
<Top Secret, Nuclear>
<Top Secret, Biowar>
<Top Secret>
<Secret, Nuclear>
<Secret, Biowar>
<Secret>
<Confidential>

支配关系--当前级别

- 问题:
  - Colonel 有(Secret, {NUC, EUR}) 安全级别
  - Major 有(Secret, {EUR})安全级别
    - Major can talk to Colonel (“上写” or “下读”)
    - Colonel cannot talk to major (“上读” or “下写”)

支配关系--当前级别

- 为主体定义最大级别和当前级别
  - *maxlevel*(*s*) ⩾ *curlevel*(*s*)
- 举例
  - Treat Major as an object (Colonel is writing to him/her)
  - Colonel has *maxlevel* (Secret, { NUC, EUR })
  - Colonel sets *curlevel* to (Secret, { EUR })
  - Now *L*(Major) ⩾ *curlevel*(Colonel)
    - Colonel can write to Major without violating “no writes down”

支配关系--当前级别

- 问题：
  - 考虑一个系统中有 s_1, s_2, o_1, o_2
    - $L(s_1) = L_C(s_1) = L(o_1) = high$
    - $L(s_2) = L_C(s_2) = L(o_2) = low$
  - 以及以下执行：
    - s_1 读 o_1
    - 然后改变当前级别到low，获得 o_2的写权限，写入 o_2
  - 每个状态都是安全的，但是存在非法信息流动
- 方法：
  - tranquility principle: 主体不能将当前级别改变到目前所读的最高级别之下

BLP模型

- 讨论：BLP模型是否可以解决特洛伊木马攻击

BLP模型

- 特洛伊木马攻击: “拥有Top Secret的主体(Subject) 读取Top Secret 的信息, 然后将其写入Unclassified 文件(Object)中”
- More generally:
  - S 读 O_1， 然后写 O_2， where $L(O_2) < L(O_1)$， and regardless of L(S)
- **证明:**
  - S 读O_1, 那么， $L(O_1) \leqslant L(S)$
  - S 写 O_2, 那么， $L(S) \leqslant L(O_2)$
  - 所以， $L(O_1) \leqslant L(S) \leqslant L(O_2)$
  - 即， $L(O_1) \leqslant L(O_2)$
  - 但结合 $L(O_2) < L(O_1)$, 有， $L(O_1) < L(O_1)$
  - 矛盾

BLP模型的不足

- 可信主体不受*特性约束，访问权限太大，不符合最小特权原则
- BLP模型注重保密性控制，而缺少完整性控制，不能控制“向上写”，不能有效限制隐通道.
- 仅能处理单级客体，缺乏处理多级客体
- 不支持系统运行时动态调节安全级的机制

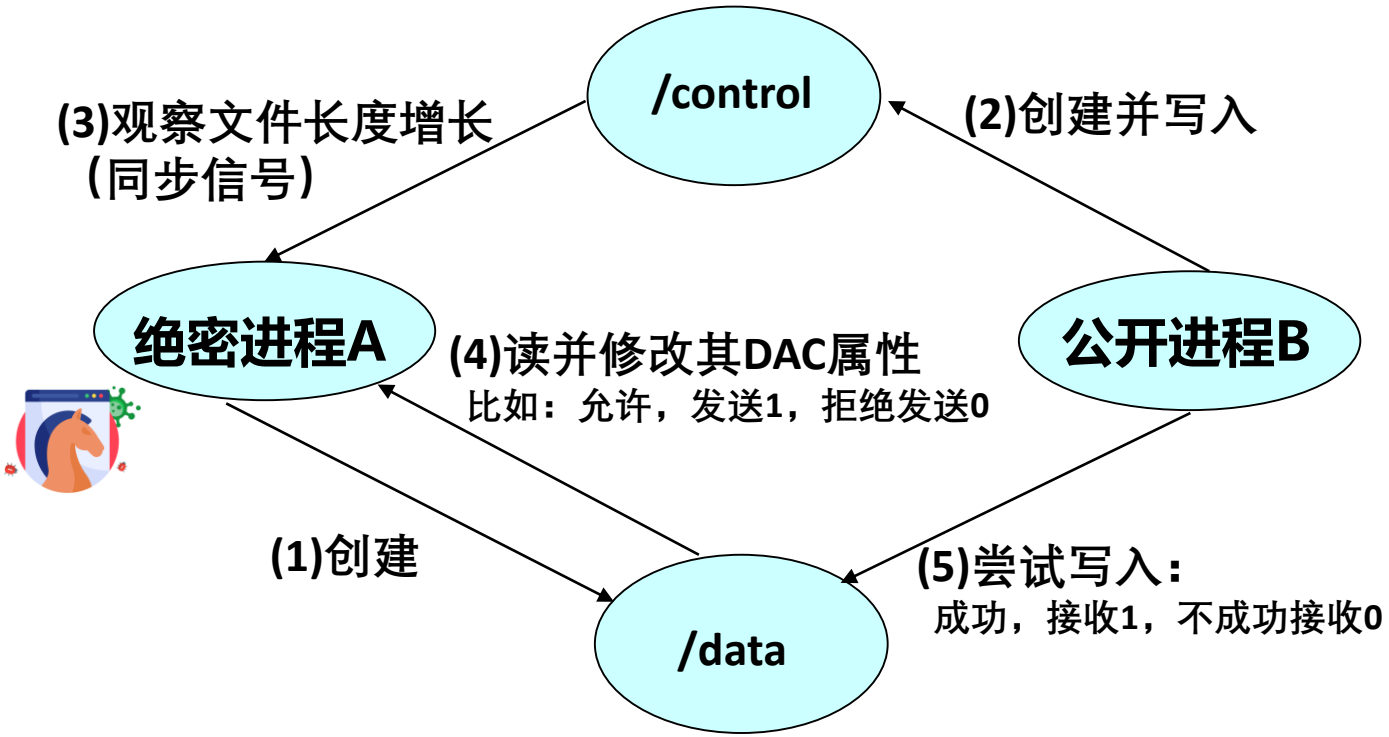
公开通道(Overt, Explicit Channels)vs.隐蔽通道(Covert Channels)

- BLP的安全目标
  - 高安全级别信息无法流向低安全级别的用户
- 通过公开通道（例如，读/写对象）的非法信息流被BLP阻止
- 但是，通过隐蔽通道的非法信息流仍然可能发生
  - 基于系统资源使用的通信通道，而这些通道通常不用于系统中的主体（进程）之间的通信

“向上写”导致的隐蔽通道示例

- 假定在一个系统中，“向上写”是允许的，如果系统中的文件/data的安全级支配进程B的安全级，即进程B对文件/data有写权限而没有读权限，进程B可以写打开、关闭文件/data。
- 因此，每当进程B为写而打开文件/data时，总返回一个是否成功打开文件的标志信息。这个标志信息就是一个隐蔽通道，它可以导致信息从高安全级流向低安全级。即，可以用来向进程B传递它本不能存取的信息。

隐蔽通道场景描述


/control
(3)观察文件长度增长
（同步信号）
(2)创建并写入
绝密进程A
(4)读并修改其DAC属性
比如：允许，发送1，拒绝发送0
公开进程B
(1)创建
/data
(5)尝试写入：
成功，接收1，不成功接收0

BLP Example

- Two users: Carla (student) and Dirk (teacher)
  - Carla (Class: s)
  - Dirk (Class: T); can also login as a students thus (Class: s)
- A student has a lower security clearance
- A teacher has a higher security clearance

BLP Example

Dirk creates f1; Carla creates f2

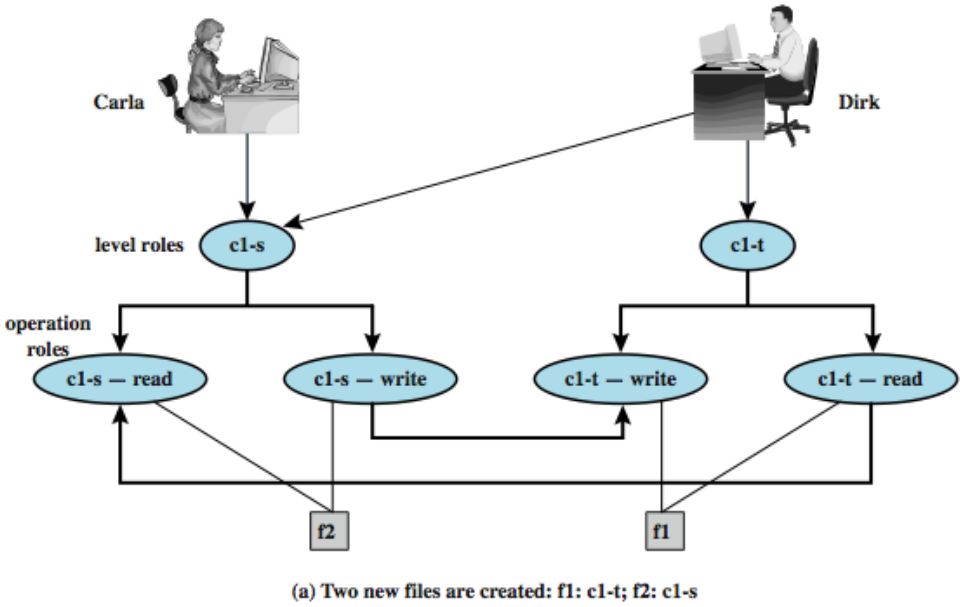
Carla can read/write f2

Carla cannot read f1

Dirk can read/write f1

Dirk can read f2

Dirk can read/write f2 only as a *stu*

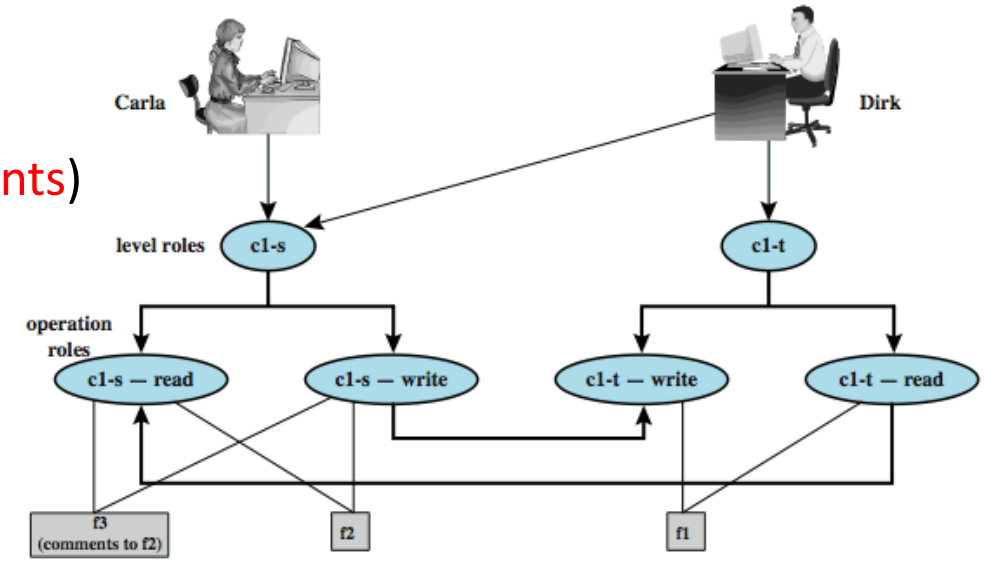


(a) Two new files are created: f1: c1-t; f2: c1-s

Dirk reads f2; want to create f3 (comments)

Dirk signs in as a *stu* (so Carla can read)

As a *teacher*, Dirk cannot create a file at *stu* classification

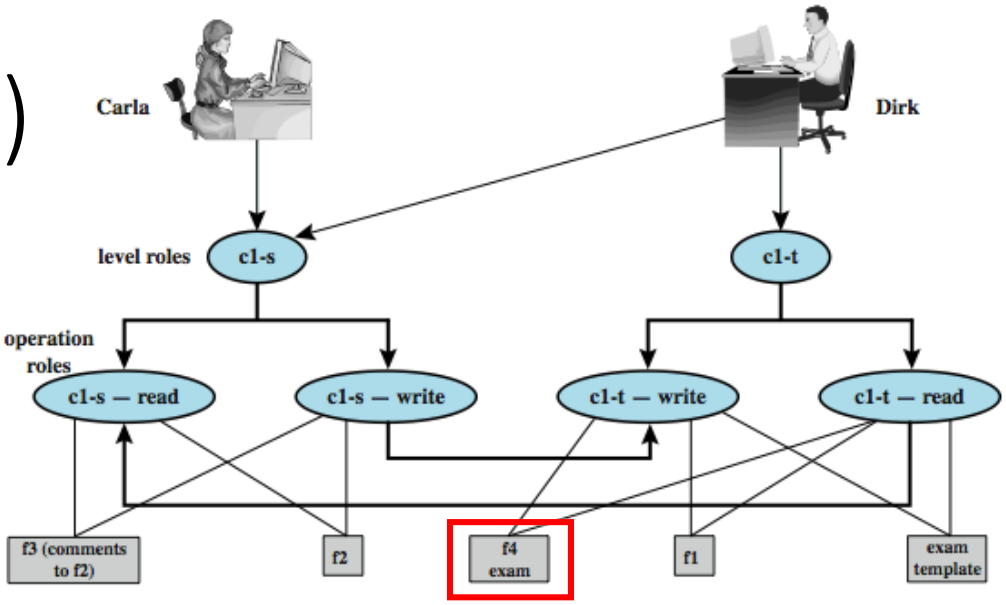


(b) A third file is added: f3: c1-s

BLP Example (cont.)

Dirk as a *teacher* creates f4 (exam)

(Must log in as a *teacher* to read template)

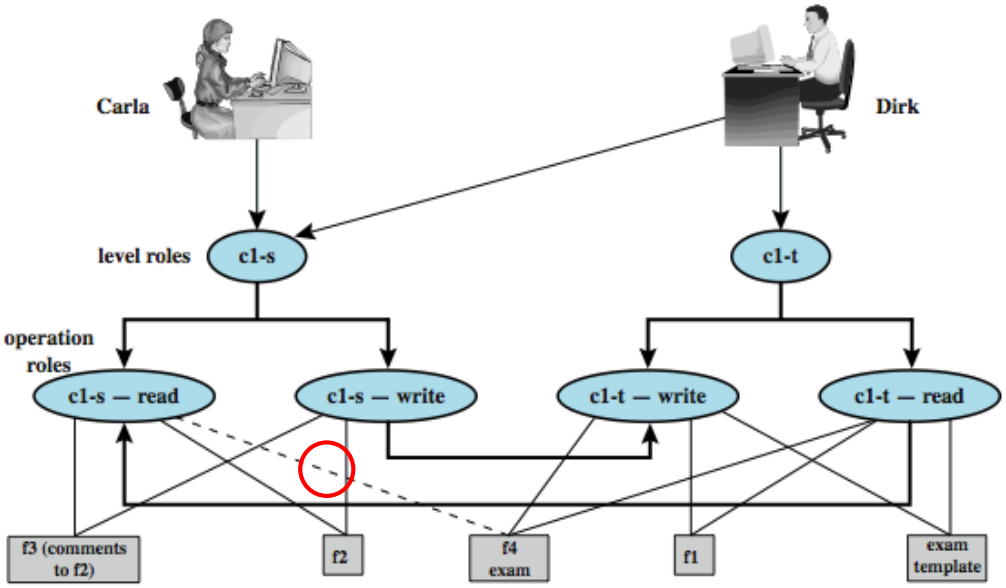


(c) An exam is created based on an existing template: f4: c1-t

Dirk wants to give Carla access to read f4

Dirk cannot do that;

An *admin* downgrades f4 (c1-t) class to c1-s



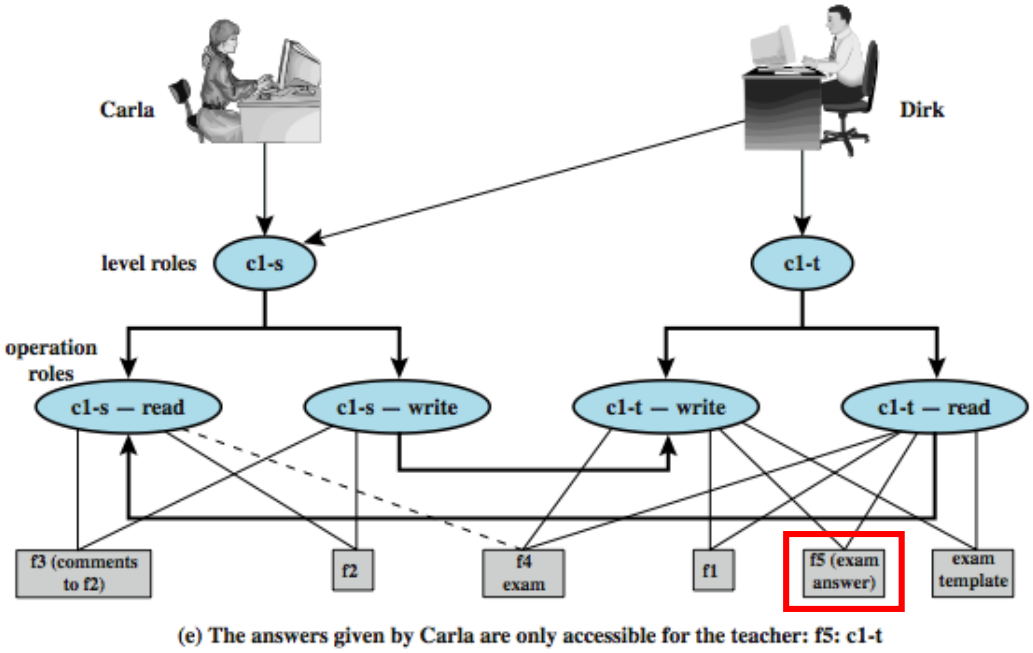
(d) Carla, as student, is permitted acess to the exam: f4: c1-s

BLP Example (cont.)

Carla writes answers to f5 (c1-t)

-- An example of write up

Dirk can read f5



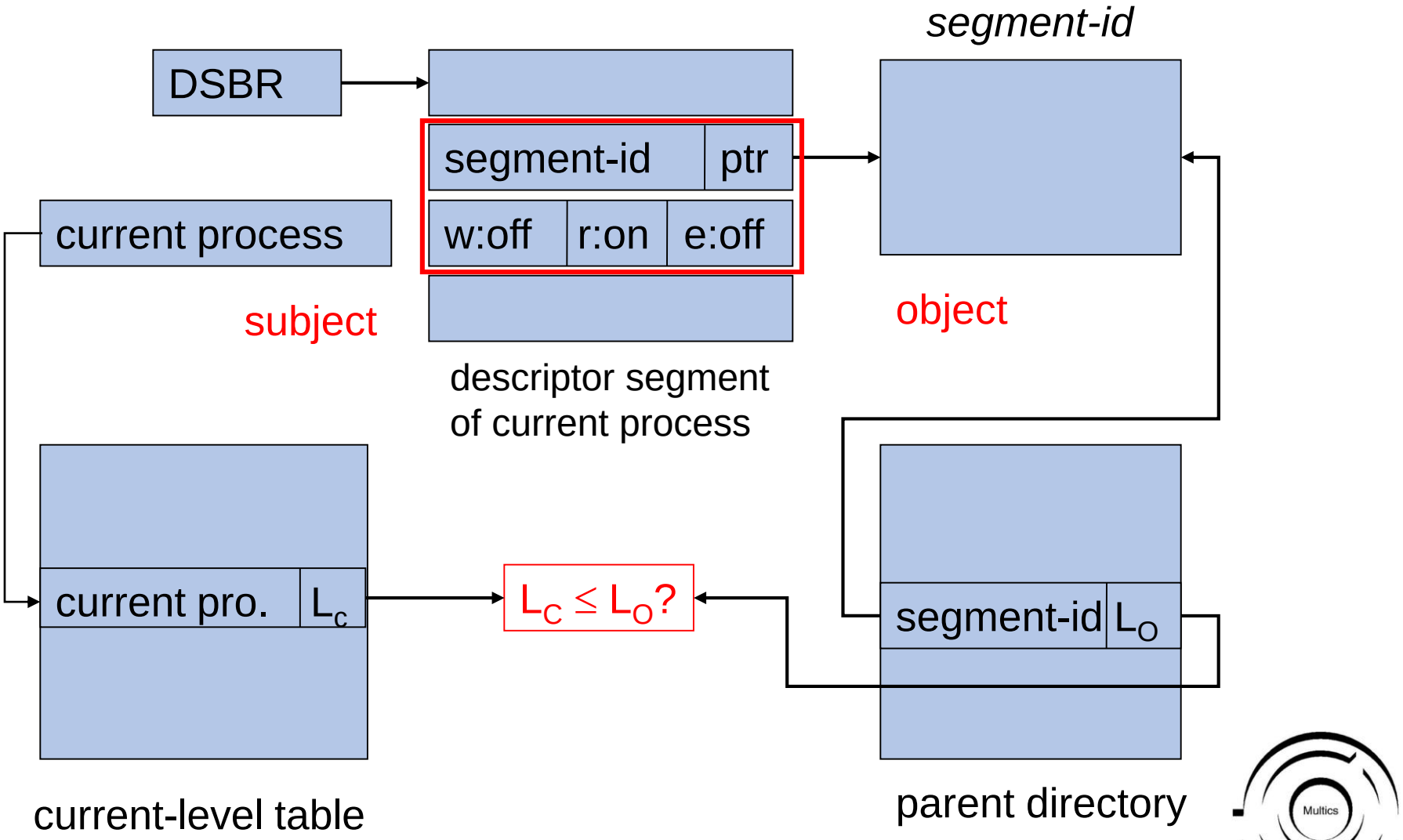
(e) The answers given by Carla are only accessible for the teacher: f5: c1-t

BLP in OSes

- SELinux [open source release by NSA 2000]
- TrustedBSD [2000], 影响了 iOS 和 OS X

BLP in Oses---Multics OS

- read r
- execute e, r
- read & write w
- write a



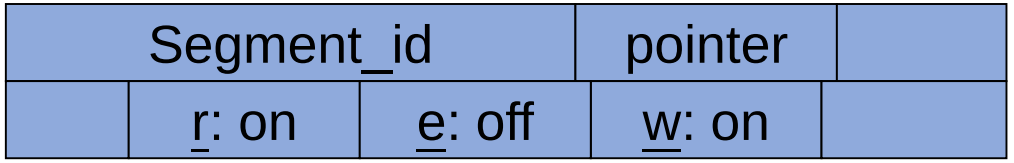
BLP in Oses---Multics OS

- Subjects in Multics are processes. Each subject has a descriptor segment containing information about the process
- The security level of subjects are kept in a process level table and a current-level table.
- The active segment table records all active processes; only active processes have access to an object.

BLP in Oses---Multics OS

- For each object the subject currently has access to, there is a segment descriptor word (SDW) in the subject's descriptor segment.
- The SDW contains the name of the object, a pointer to the object, and flags for read, execute, and write access.
- Objects are memory segments, I/O devices, ...

segment descriptor word (SDW)



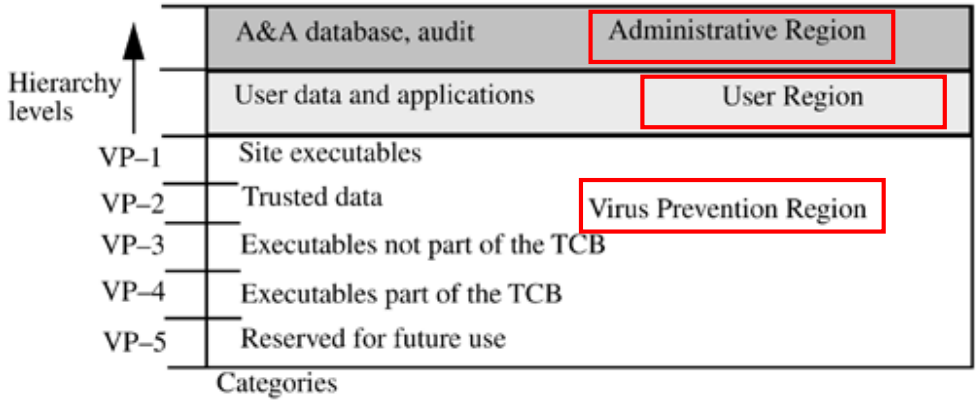
BLP in Oses---Multics OS

- Multics的星号安全特性
- For any SDW in the descriptor segment of an active process, the current level of the process
  - dominates the level of the segment if the read or execute flags are on and the write flag is off,
  - is dominated by the level of the segment if the read flag is off and the write flag is on,
  - is equal to the level of the segment if the read flag is on and the write flag is on.

BLP in Oses---Multics OS

- Conditions for get-read
  - The OS has to check whether
    - the ACL of segment-id, stored in the segment's parent directory, lists process-id with read permission,
    - the security level of process-id dominates the security level of segment-id,
    - process-id is a trusted subject, or the current security level of process-id dominates the security level of segment-id.
  - If all three conditions are met, access is permitted and a SDW in the descriptor segment of process-id is added/updated.

BLP in Oses--- DG/UX



DG/UX [1985]

- Three regions: Virus Prevention ⊑ User Region ⊑ Administrative Region
- Writing up is prohibited (不可上写，完整性)
- User Region
- Virus Prevention: executables that implement the system
  - Can't be written by users (不可下写)
  - Can be executed
- Administrative Region: authorization & authentication database (assigns labels), audit logs
  - Can't be read by users (不可上读)

Biba模型

- 1977年，Biba等人提出了第一个完整性安全模型---Biba模型，它基本是BLP模型的对偶，不同的只是它对系统中的每个主体和客体均分配一个完整性级别（integrity level）。
- 通过信息完整性级别的定义，在信息流向的定义方面不允许从级别低的进程到级别高的进程，也就是说用户只能向比自己安全级别低的客体写入信息。
- 防止非法用户创建安全级别高的客体信息，避免越权、篡改等行为的产生。

什么是系统的完整性?

- *Attempt* 1: Critical data not changed (e.g., TCB).
- *Attempt* 2: Critical data changed only in "correct ways"
  - E.g., in DB, integrity constraints are used for consistency
- *Attempt* 3: Critical data changed only through certain "trusted programs" (e.g., /bin/passwd)
- *Attempt 4*: Critical data changed only as intended by authorized users

Biba模型

- 客体的完整性级别：用于描述对一个客体中所包含信息的信任度“trustworthiness”。比如：过路人的报告和专家组的报告。记为：i(o)
- 主体的完整性级别：用于衡量主体产生/处理信息能力上的信心“confidence”。比如：经鉴定软件产品和网上自由下载的软件。记为：i(s)
- 完整性级别和可信度有密切的关系，完整级别越高，意味着可信度越高。

- 三种访问模式：observe（read), modify（write), invoke（execute）

Biba模型

- 完整性级别高的实体对完整性低的实体具有完全的支配性
- 反之，如果一个实体对另一个实体具有完全的控制权，说明前者完整性级别更高，这里的实体既可以是主体也可以是客体。

完整性级别绝对支配关系

- 设i_1和i_2是任意两个完整性级别，如果完整性级别为i_2的实体比完整性级别为i_1的实体具有更高的完整性，则称完整性级别i_2绝对支配完整性级别i_1，记为：

$$i_1 \prec i_2$$

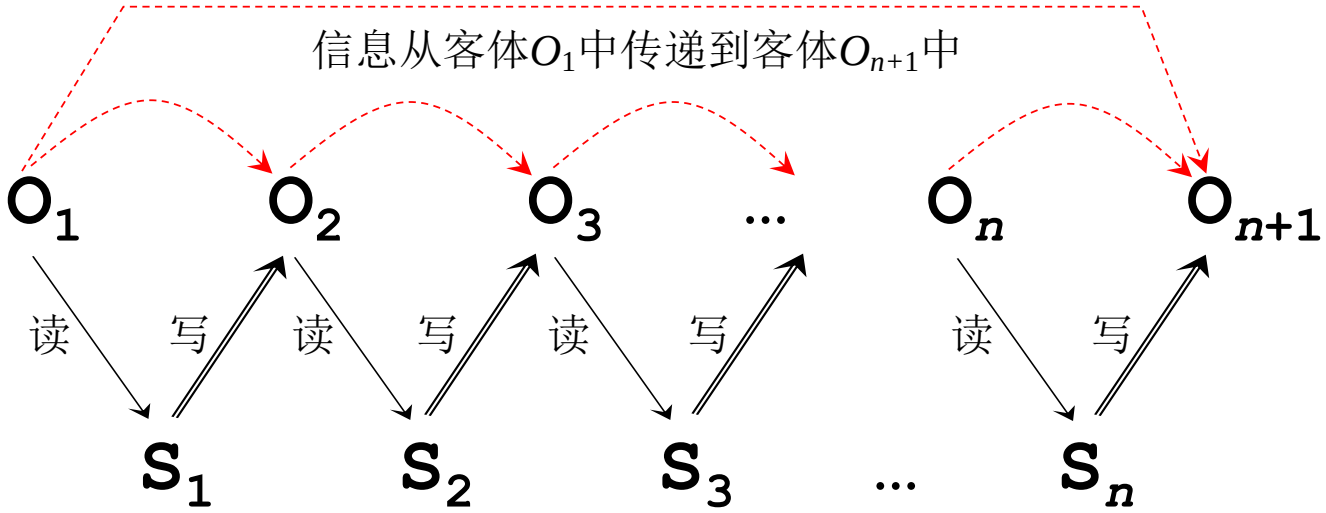
完整性级别支配关系

- 设i_1和i_2是任意两个完整性级别，如果i_2绝对支配i_1，或者，i_2与i_1相同，则称i_2支配i_1，记为：

$$i_1 \leqslant i_2$$

信息传递路径定义

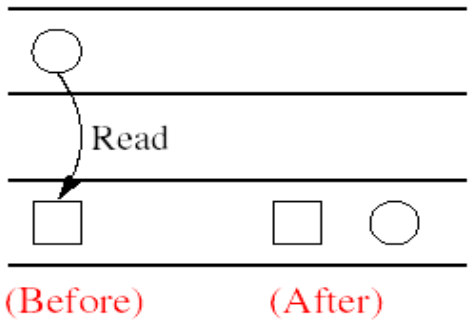
- 一个信息传递路径是一个客体序列$O_1, O_2, \ldots, O_{n+1}$和一个对应的主体序列$S_1, S_2, \ldots, S_n$，其中，对于所有的$i$（$1 \leqslant i \leqslant n$），有$S_i \underline{r} O_i$和$S_i \underline{w} O_{i+1}$。



通过连续读取和写入，信息可以从O_1流向O_{n+1}

主体低水标(Low-Water-Mark)规则

- 设S是任意主体，O是任意客体，$i_{min} = min(i(S), i(O))$，那么，不管完整性级别如何，$S$都可以读$O$，但是，“读”操作实施后，主体$S$的完整性级别被调整为$i_{min}$。



- 当且仅当 $i(O) \leqslant i(S)$ 时，主体S可以写客体O。

即：任意主体可以读任意完整性级别的客体，但是如果主体读完整性级别比自己低的客体时，主体的完整性级别降低为客体完整性级别，否则，主体的完整性级别保持不变

意义：当主体需要读完整性级别低的客体，说明这个主体对低可信度的信息有依赖性，那对该主体的信任程度因此降低，于是把它的完整性级别降低到客体的完整性级别，防止这个主体进行非法修改行为。

主体低水标规则

- 写规则旨在防止直接的非法修改行为的发生。即，主体和客体的完整性级别标记能直接反映出该修改是不合理的；
- 读规则旨在防止间接的非法修改行为的发生。即：主体和客体原有的完整性级别标记并未反映出该修改的不合理，但从实际的完整性看，该修改已经不合理。
  - 例如：由于接受了低完整性的信息，因受感染的缘故，主体的完整性实际上已经降低，但原有的完整性级别标记没有反映出来。

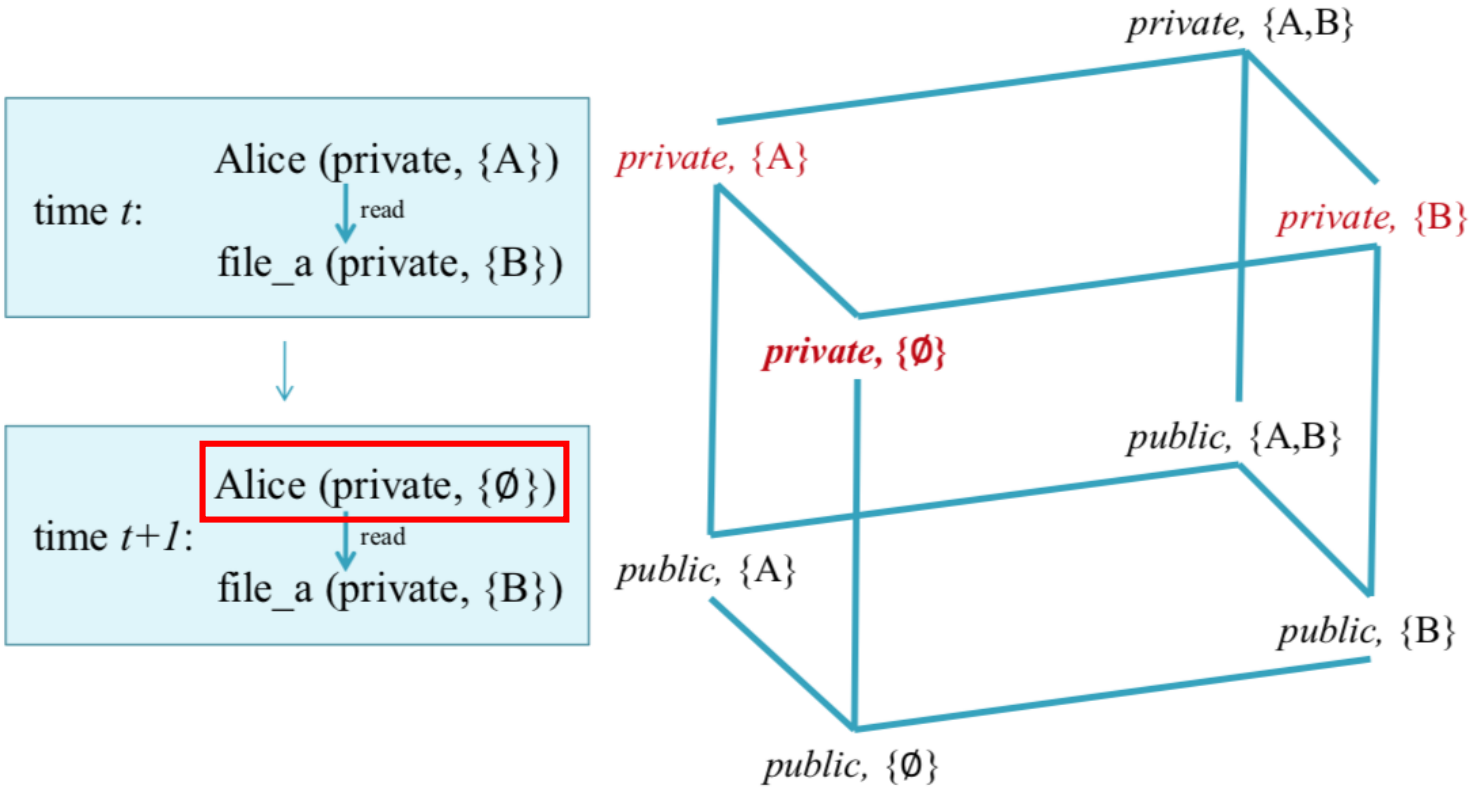
主体低水标的信息传递

- 在Biba主体低水标模型的控制下，如果系统中存在一个从客体O_1到客体O_{n+1}的信息传递路径，那么，对于任意的k（$1 \leqslant k \leqslant n$），必有 $i(O_{k+1}) \leqslant i(O_1)$。

- **信息不会从低完整性级别的客体传向高完整性级别的客体。**

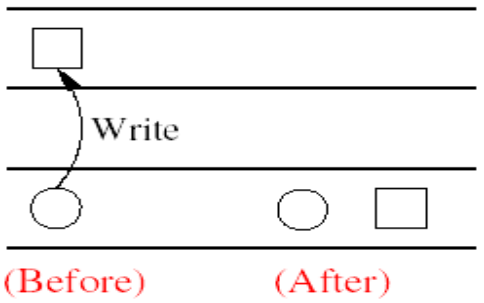
- 问题：随着系统运行，主体的完整性水平下降
  - 任何主体很快将都无法访问高完整性级别的客体

举例---主体低水标规则


time t:
Alice (private, {A})
read
file_a (private, {B})
time t+1:
Alice (private, {∅})
read
file_a (private, {B})
private, {A,B}
private, {A}
private, {B}
private, {∅}
public, {A,B}
public, {A}
public, {B}
public, {∅}

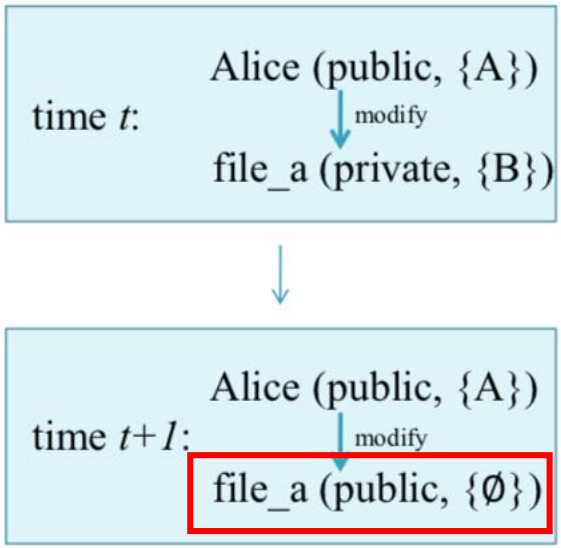
客体低水标规则

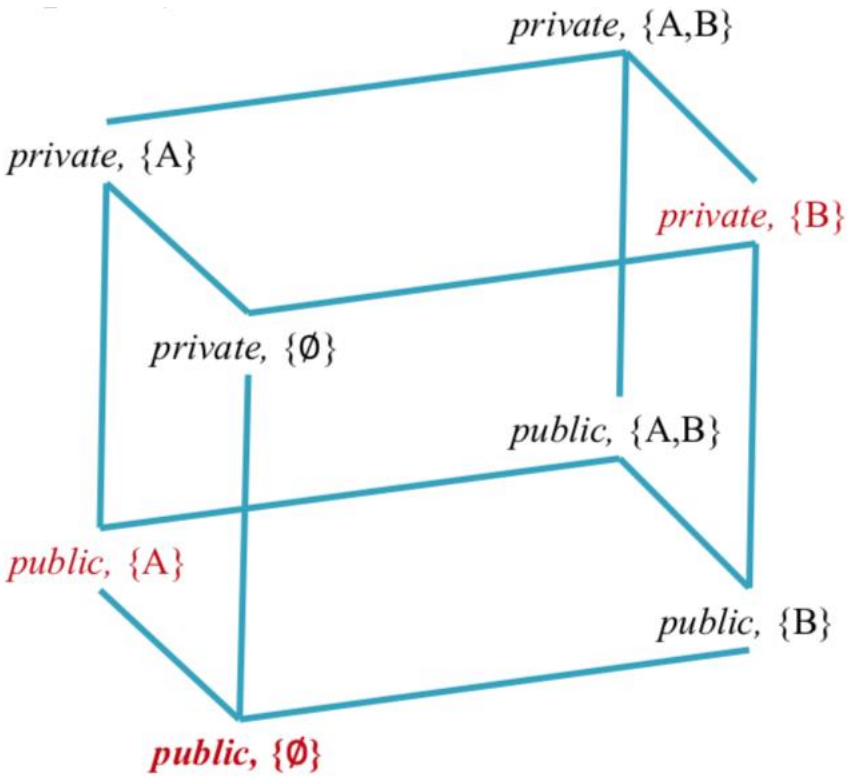
- 当且仅当 $i(S) \leqslant i(O)$ 时，主体S可以读客体O。
- 设S是任意主体，O是任意客体，$i_{min} = min(i(S), i(O))$，那么，不管完整性级别如何，$S$都可以写$O$，但是，“写”操作实施后，客体$O$的完整性级别被调整为 i_{min}。



客体的完整性级别会因主体的污染而降低。

举例---客体低水标规则


time t:
Alice (public, {A})
modify
file_a (private, {B})
time t+1:
Alice (public, {A})
modify
file_a (public, {∅})


private, {A,B}
private, {A}
private, {B}
private, {∅}
public, {A,B}
public, {A}
public, {B}
public, {∅}

低水标完整性审计规则

- 设S是任意主体，O是任意客体，$i_{min} = min(i(S), i(O))$，那么，不管完整性级别如何，$S$都可以读$O$，但是，“读”操作实施后，主体$S$的完整性级别被调整为$i_{min}$。
- 设S是任意主体，O是任意客体，$i_{min} = min(i(S), i(O))$，那么，不管完整性级别如何，$S$都可以写$O$，但是，“写”操作实施后，客体$O$的完整性级别被调整为$i_{min}$。

追踪，但不防止污染

类似于软件安全中的污点（Tainting）概念

环规则

- 不管完整性级别如何，任何主体都可以读任何客体。
- 当且仅当 $i(O) \leqslant i(S)$ 时，主体S可以写客体O。

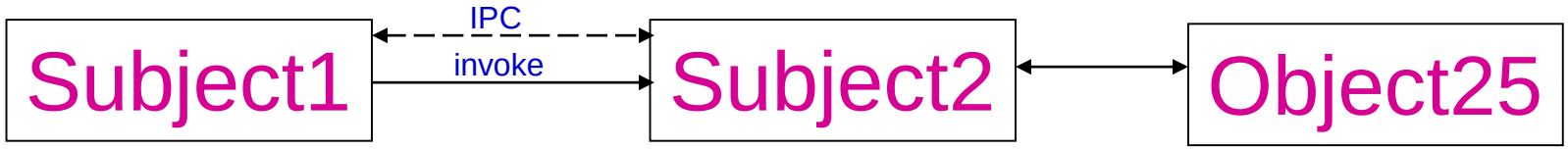
主体被信任可正确地处理低安全级别的输入。（与主体低水标规则的区别）

严格完整性规则

- 当且仅当 $i(S) \leqslant i(O)$ 时，主体S可以读客体O（不下读）。
- 当且仅当 $i(O) \leqslant i(S)$ 时，主体S可以写客体O（不上写）。

实施了严格完整性模型的规则，就隐含地实施了低水标模型的规则；因此，严格完整性模型也同时防止对实体进行直接的或间接的非法修改

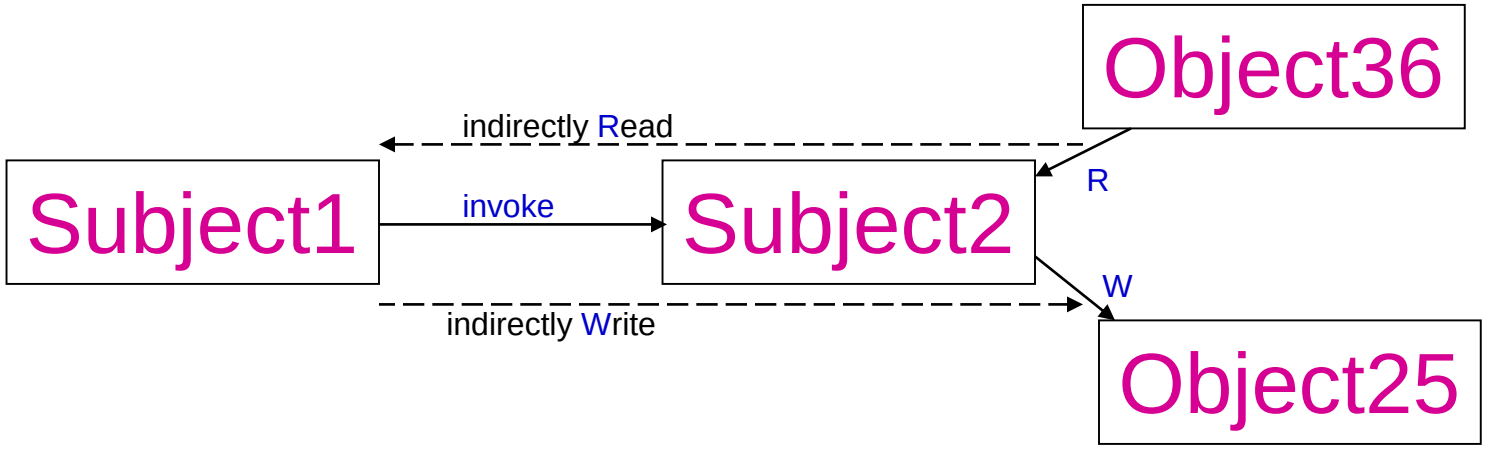
调用/执行



主体运行并与另一个进程（另一个主体）通信的权限

从而通过调用软件工具（主体）间接访问其它客体。

调用/执行规则



- 两种选择
  - Invocation Property---调用低完整性级别的主体
  - Controlled Invocation (Ring Invocation) ---调用高完整性级别的主体

调用/执行规则

- Invocation Property – 主体 S_1 只能调用在其自身完整性级别或之下的另一个主体 S_2，即 $i(S_2) \leqslant i(S_1)$
  - 举例：管理员调用工具来更改普通用户的密码。
  - 动机：否则，“dirty”工具可能会使用clean工具污染干净的客体。即，防止滥用可信程序。
    - 阻止的示例：防止由于防火墙而无法连接到网络的病毒（低），调用IE浏览器（高）（作为命令行参数传输到IE），进而为它构建连接以危害服务器。

调用/执行规则

- Controlled Invocation – 主体 S_1 只能调用在其自身完整性级别或之上的另一个主体 S_2，即 $i(S_1) \leqslant i(S_2)$
  - 低级主体应该只能通过可信/认证的更高级别的进程/工具访问高级客体。 高级工具需要执行所有的一致性检查，以确保客体保持clean。
  - 举例：普通用户通过密码更改工具更改自己的密码，需要控制为仅能更该用户自己的密码，而不是/etc/shadow中的其它任何密码。
  - 动机：被调用的进程必须至少与调用进程一样可信（防止非特洛伊木马进程调用特洛伊木马）。
    - 阻止的示例：阻止管理员使用没有验证其安全性/完整性的第三方管理工具

Biba and BLP

- Biba的严格完整性策略是BLP机密性策略的对偶。
- Biba和BLP模型相结合。

  基于Biba模型和BLP模型的相似性，Biba模型可以比较容易的与BLP模型的相结合用以形成集机密性和完整性于一体的综合性安全模型。比如：在格（Lattice）模型中实现了机密格和完整格的结合。

完整性与机密性

- 高机密性、低完整性
  - Information collected by spy.
- 高完整性、低机密性
  - Code, configuration data, time, public key, root certs, etc.

完整性与机密性

机密性	完整性
Control reading; preserved if confidential *info* is not read	Control writing; preserved if important *obj* is not changed
For subjects who need to read, control writing after reading is sufficient, no need to trust them	For subjects who need to write, has to trust them, control reading before writing is not sufficient

Integrity requires trust in subjects!

完整性与机密性

- Confidentiality violation: leak a secret
  - CAN be prevented even if I tell the secret to a person I do not trust, so long as I can lock the person up AFTERWARDS to prevent further leakage
    - The person cannot leak confidential info without talking
- Integrity violation: follow a wrong instruction
  - CANNOT be prevented if I follow instruction from an person I do not trust even if I lock the person up BEFOREHAND to prevent the person from receiving any malicious instruction
    - The person can invent malicious instruction without outside input

完整性与机密性的区别

- For confidentiality, controlling reading & writing is sufficient
  - theoretically, no subject needs to be trusted for confidentiality;
  - however, one does need trusted subjects in BLP to make system realistic
- For integrity, controlling reading & writing is insufficient
  - one has to trust all subjects who can write to critical data

Biba完整性保护的应用

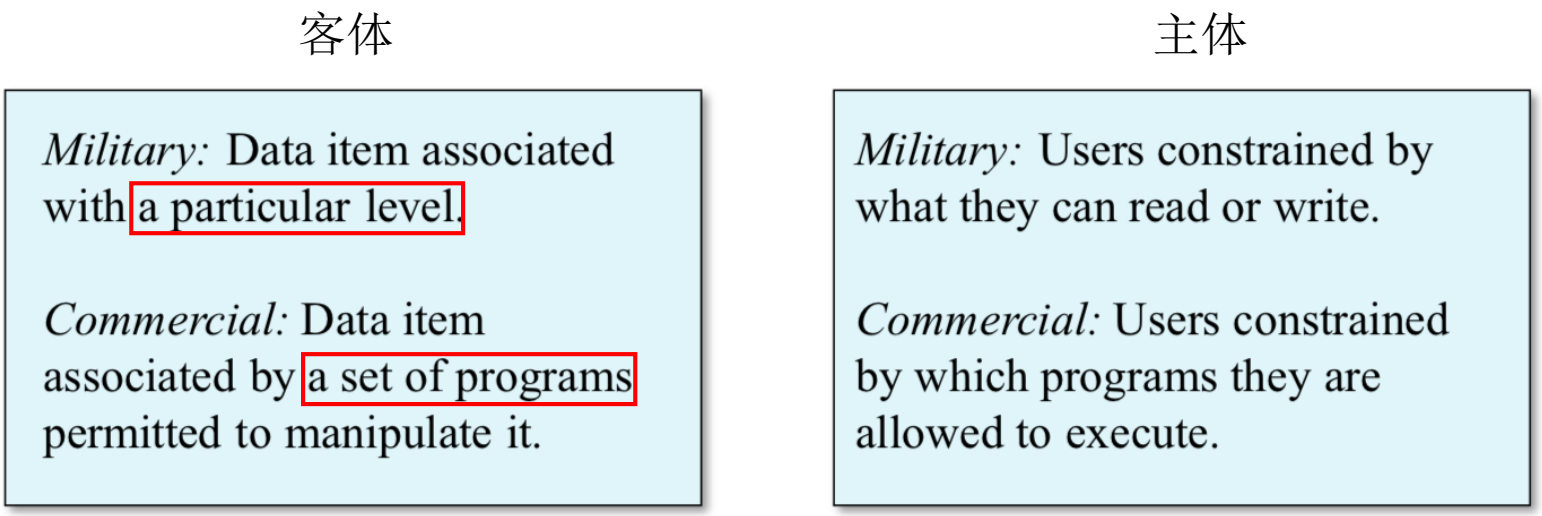
- Windows的强制完整性控制 (since Vista)
  - Uses four integrity levels: *Low*, *Medium*, *High*, and *System*
  - Each process is assigned a level, which limit resources it can access
  - Processes started by normal users have *Medium*
  - Elevated processes have *High*
    - Through the User Account Control feature
  - Some processes run as *Low*, such as IE in protected mode
  - Reading and writing do not change the integrity level
    - 环规则（Ring policy）

Biba完整性保护的应用

- Fully implemented in FreeBSD 5.0.
  - as a kernel extension
  - the integrity levels are defined for subjects and objects in a configuration file.
  - support for both hierarchical and non-hierarchical labeling of all system objects with integrity levels
  - supports the strict enforcement of information flow to prevent the corruption of high integrity objects by low integrity processes.

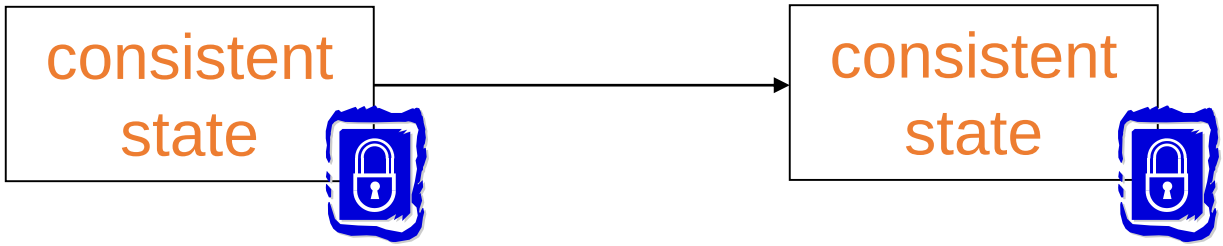
Clark-Wilson 模型

- 面向事务，既要确保数据的完整性，又要保证事务的完整性。
- 应用具有不同的强制访问控制需求：

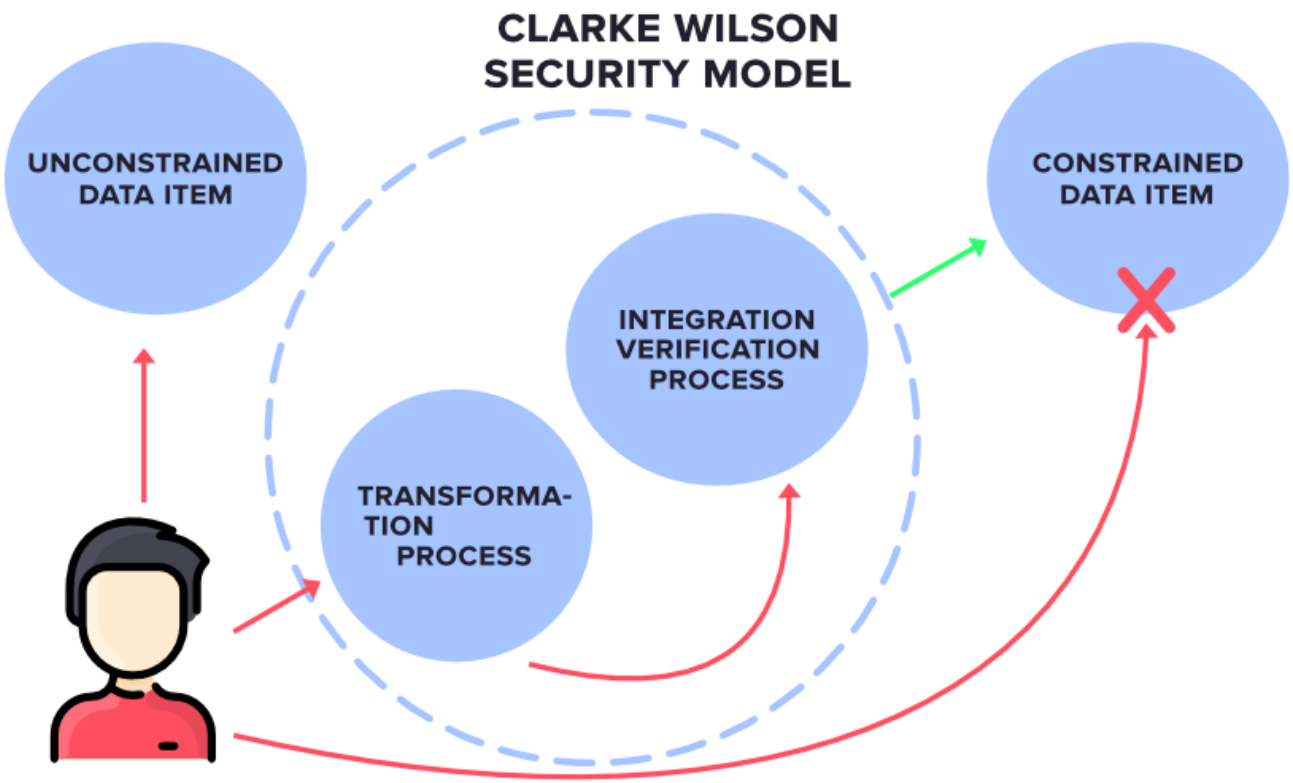


Clark-Wilson 模型

- 两个重要概念
  - 良构事务（Well-formed Transaction）
    - 用户只能通过可信程序（即良构事务）操作数据
    - 良构事务可以将数据从一个一致性状态转换到另一个一致性状态
  - 职责分离-用户只能使用某一组程序
    - 如果用户能创建一个良构事务，他可能不被允许运行它
    - 用户必须合作来操作数据



Clark-Wilson 模型



Clark-Wilson 模型

- 一致性---如果满足某些给定性质，那么数据是一致的
- 举例: Bank
  - Objective: today's deposits - today's withdrawals + yesterday's balance = today's balance
  - Policy level 1: transactions must meet this objective
  - Policy level 2: users execute only those transactions
  - Policy level 3: certifiers must ensure users do so
  - Policy level 4: logs will monitor that certifiers are doing their job!

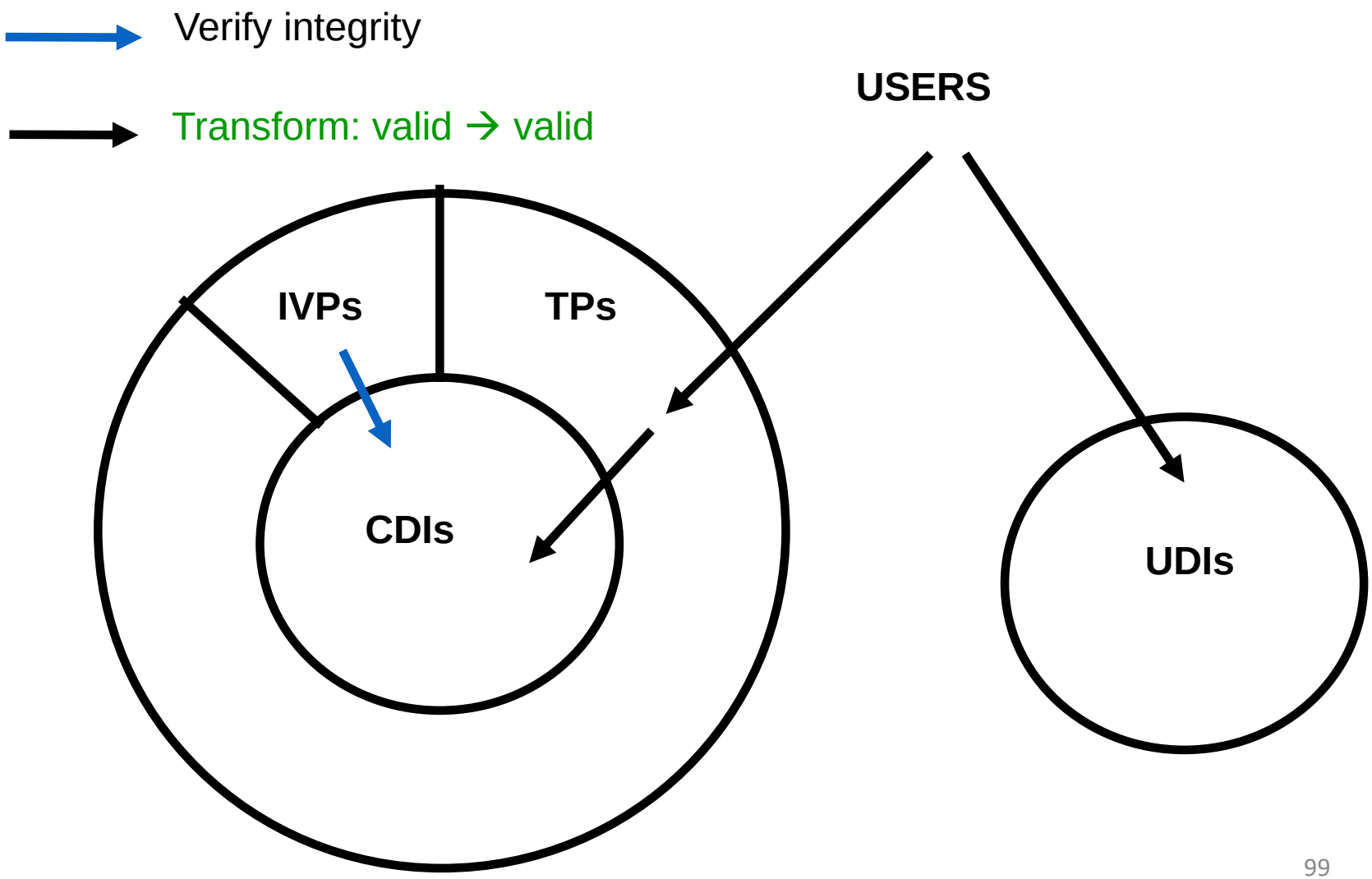
Clark-Wilson 模型

- 关键贡献：层次结构减少了对特殊可信主体的依赖
  - Certifiers will enforce users to run only good transactions
  - logs will in turn monitor certifiers
- But who will then monitor log auditors (logs)?
  - *Trust* is always needed （根信任）

模型的要素

- **Users** Active agents
- **CDIs** Constrained Data Items
  - 需要完整性的数据
- **UDIs** Unconstrained Data Items
  - 不需要完整性的数据
- **TPs** Transformation Procedures
  - like commands in Access Control Matrices, but for debit, credit
- **IVPs** Integrity Verification Procedures
  - 定期运行以检查CDI的完整性

模型的要素


Verify integrity
Transform: valid → valid
USERS
IVPs
TPs
CDIs
UDIs

Clark-Wilson 模型

- C-W模型采取两类措施来支持系统完整性
  - 系统实施的措施 （实施规则，ER）
  - 用于证明系统实施的有效性的措施 （证明规则，CR）

规则

Certification Rules （证明规则）

CR1 IVPs verify CDI integrity

CR2 TPs preserve CDI integrity

CR3 Separation of duties for ER2

CR4 TPs write to log

CR5 TPs upgrade UDIs to CDIs

Enforcement Rules （实施规则）

ER1 CDIs changed only by authorized TP

ER2 TP run only by authorized users

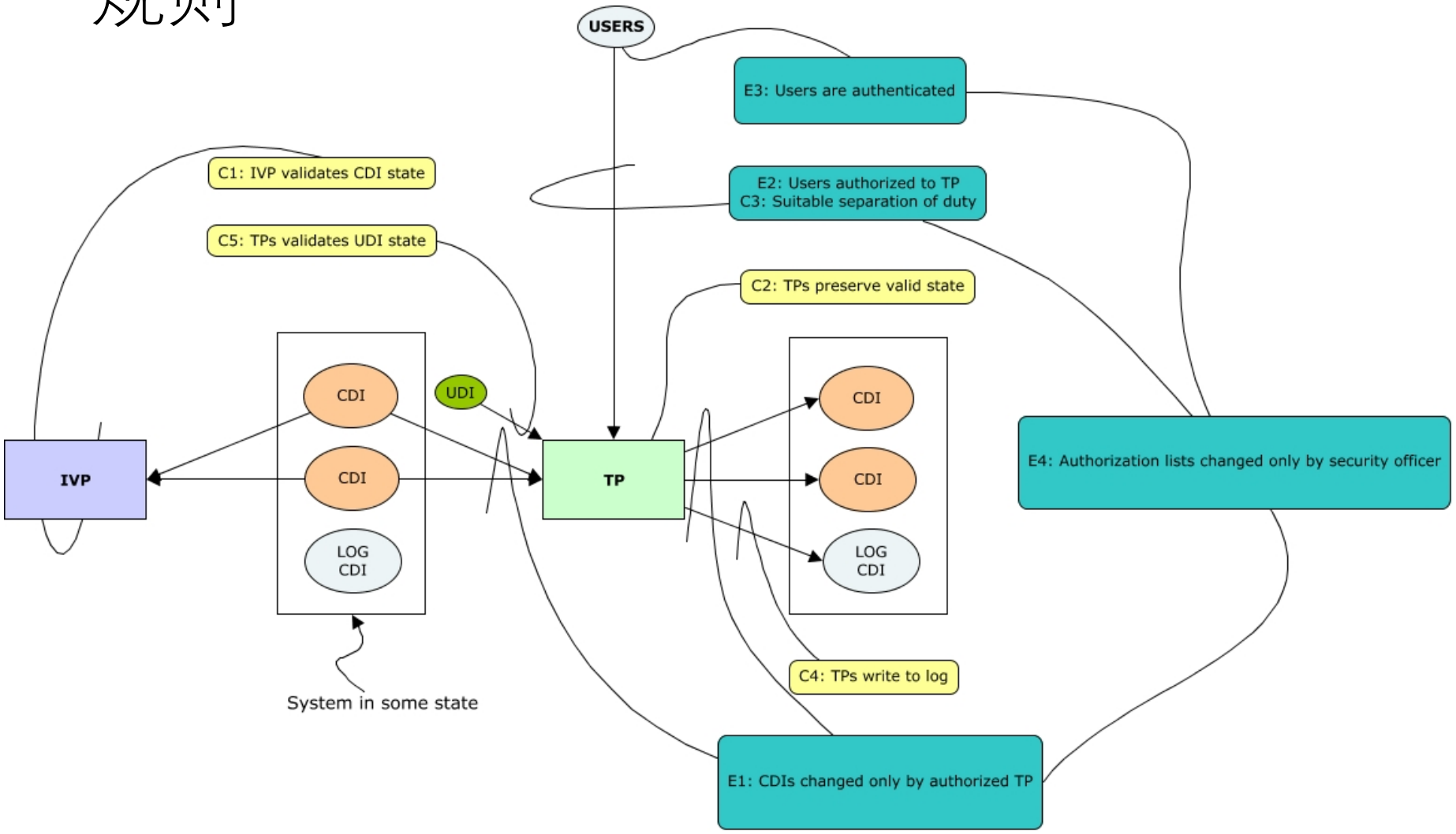
ER3 Users are authenticated

ER4 Authorizations changed only by certifiers

CR1 规定完整性验证流程（IVPs）必须验证控制数据项（CDI）的完整性，保证数据未被未授权修改。
CR2 强调可信进程（TPs）有责任维护CDI的完整性，确保数据处理过程中数据保持准确和完整。
CR3 提出对于实施规则ER2的职责分离要求，意味着执行关键任务的角色应该由不同的个体或实体来担任，以提高安全性。
CR4 要求可信进程必须将操作记录在日志中，以便于事后审计和监控。
CR5 说明可信进程应将未控制数据项（UDIs）升级为控制数据项（CDIs），以应用更严格的安全措施。

ER1 规定只有授权的可信进程才能更改CDI，确保数据更改的合法性和安全性。
ER2 要求可信进程只能由授权用户执行，限制了能触发关键操作的人员范围。
ER3 强调用户必须经过身份验证，以确保只有合法用户能够访问系统和资源。
ER4 规定授权的变更只能由认证者进行，确保权限和访问级别的变更得到严格控制。

规则


USERS
E3: Users are authenticated
C1: IVP validates CDI state
E2: Users authorized to TP
C3: Suitable separation of duty
C5: TPs validates UDI state
C2: TPs preserve valid state
CDI
UDI
CDI
LOG
CDI
IVP
TP
CDI
CDI
LOG
CDI
E4: Authorization lists changed only by security officer
C4: TPs write to log
System in some state
E1: CDIs changed only by authorized TP

Certification Rules 1,2,3

CR1　当任何IVP运行时，它必须确保所有CDIs都处于有效状态

CR2　对于一个相关的CDIs集合，一个TP必须将处于有效状态的那些CDIs转换为（可能不同的）有效状态

- A relation certified associates a set of CDIs with a particular TP
  - Say ($before_1$, $after_1$), ($before_2$, $after_2$) … ($before_n$, $after_n$)
- Example: TP-- withdraw money, CDIs-- accounts, in bank example

CR3　允许关系(allowed relations (user, TP, CDIs))必须符合职责分离原则（ separation of duty, SoD）的要求

SoD: The principle that says different duties that may result in compromising integrity must not be permitted to be executed by the same process, subject, or entity

Certification Rules 4

CR4　所有的TP都向一个只能以添加(append)方式写log，以便能够重现TP的操作过程

- Because the auditor needs to be able to determine what happened during reviews of transactions

实际欺诈的例子:

Hastings银行的一名银行职员注意到，他们的系统没有审核地址的变化。于是他把一位女士的地址改成了自己的地址，签发了一张信用卡和密码，然后又改回了这位女士的真实地址。他从她那里偷了8600英镑。

当这位女士投诉时，她没有被相信，银行坚持认为他们的系统是安全的，而且这些提款一定是她的错（她因疏忽，让别人拿走了她的卡和密码）。她不得不支付。

https://www.cl.cam.ac.uk/~rja14/Papers/SE-09.pdf

Certification Rules 5

CR5　TP以两种方式之一处理 UDI

(1): 要么拒绝该UDI

(2): 要么将转换为CDI

- 例子： 在一家银行，在键盘上输入的存款金额是UDI。TP必须在使用数字之前验证它们（使它们成为CDI）；如果验证失败，TP拒绝UDI

Enforcement Rules 1 and 2

ER1　系统必须维护证明关系(certified relations (TP, CDI))，并且必须确保只有经过证明能在一个CDI上运行的TPs才能操纵该CDI

ER2　系统必须将User与每个TP和一组CDI相关联(allowed relation (user, TP, CDIs))。 TP可以代表相关User访问这些CDI，但无法代表与该TP和CDI无关的User访问该CDI

Enforcement Rules 3 and 4

ER3　系统必须对试图执行TP的每个User进行身份认证

- Authentication not required before use of the system, but is required before manipulation of CDIs

ER4　只有有权给某TP作证明的主体才能修改该TP与相关实体（如CDIs）之间的关系；并且，有权给某实体（TP或CDI）作证明的主体不能拥有与该实体相关的执行权限。

- Enforces separation of duty with respect to certified and allowed relations

与 BLP的区别

- 一个数据项并不与一个特定的安全级别相关联，而是与一组TPs可信进程相关联

- 一个用户没有被赋予对数据项的读/写权限，而是被赋予执行某些程序的权限

与 Biba的比较

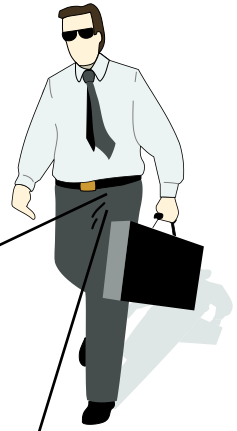
- Biba lacks the procedures and requirements on identifying subjects as trusted

- Clark-Wilson focuses on how to ensure that programs can be trusted

Chinese Wall模型

- Brewer-Nash, 1989
- 主要目标：防止利益冲突(conflicts of interest)
- 动态职责分离（SoD）：
  - 主体可以自由选择要访问数据，但选择会影响后续权限
  - 主体后续访问中，权限取决于其当前的和已访问的数据：
    - 举例：一旦你为百事可乐工作，你就无法为可口可乐工作！

Chinese Wall- 直接信息流



conflict of interest class 1

Yahoo

file1, file2

Google

file3, file5

max 1 of each

conflict of interest class 2

HSBC

file6

RBS

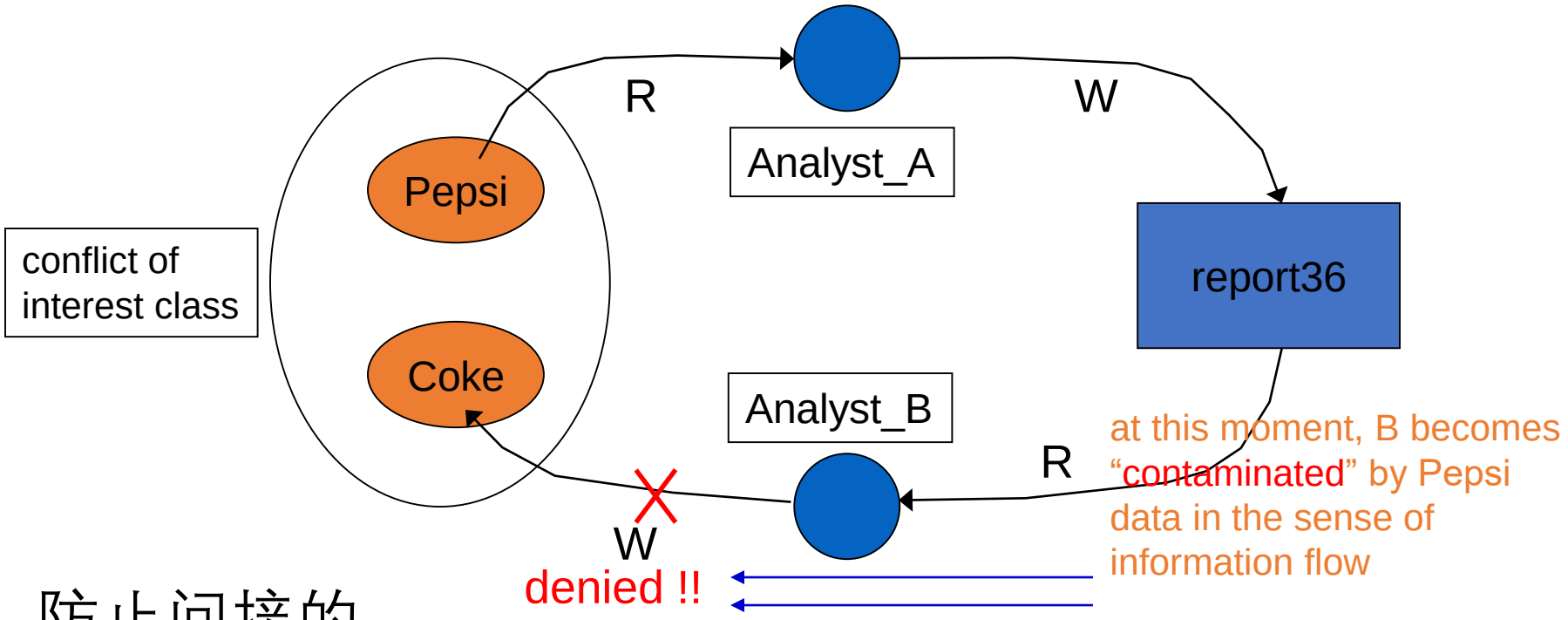
file7

防止直接的
信息泄漏!

Chinese Wall–间接信息流- Write Rule

Write access granted if no other object can be read that:

- Belongs to a competing company dataset
- Contains un-sanitized information

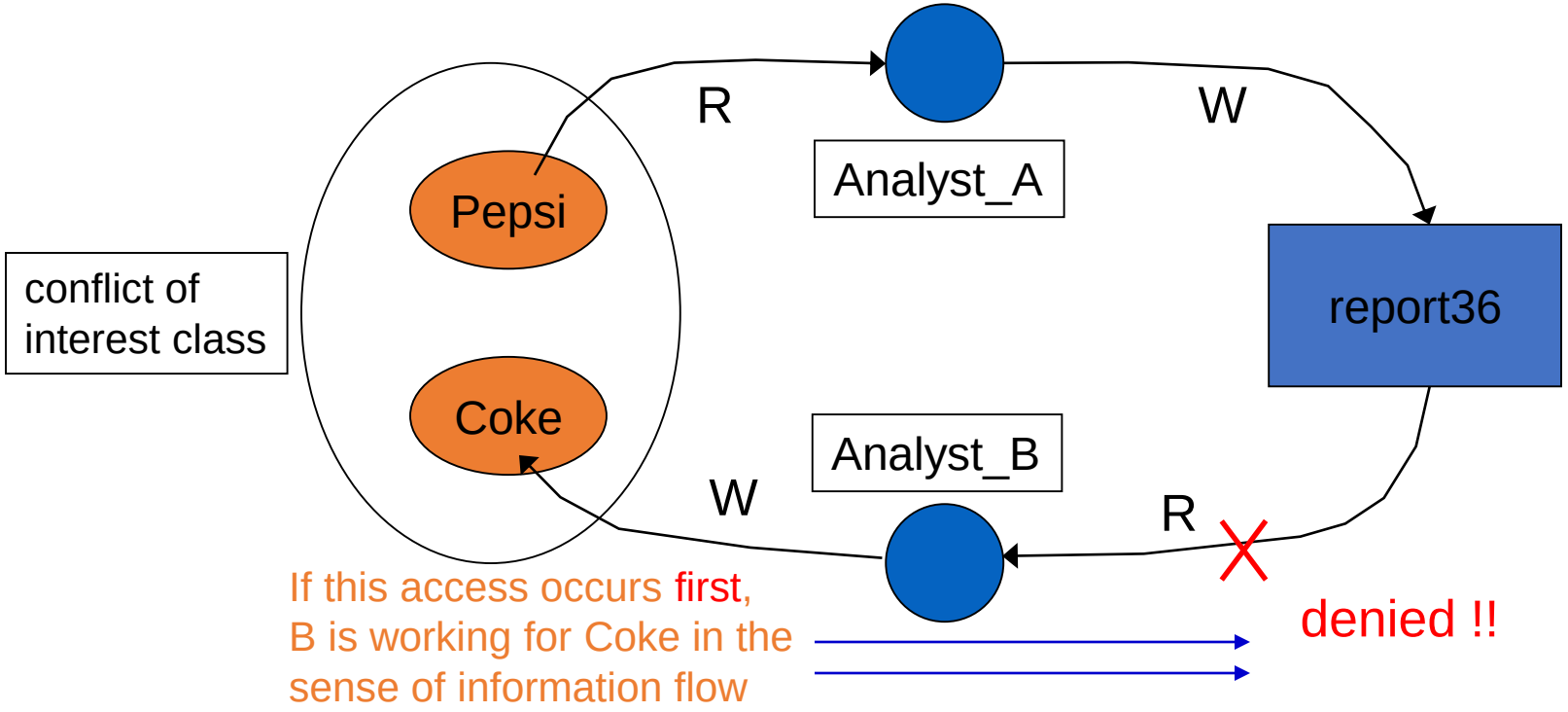


防止间接的信息泄漏!

Chinese Wall –间接信息流- Read Rule

Read access granted if no other object can be written that:

- Belongs to a competing company dataset
- Contains un-sanitized information



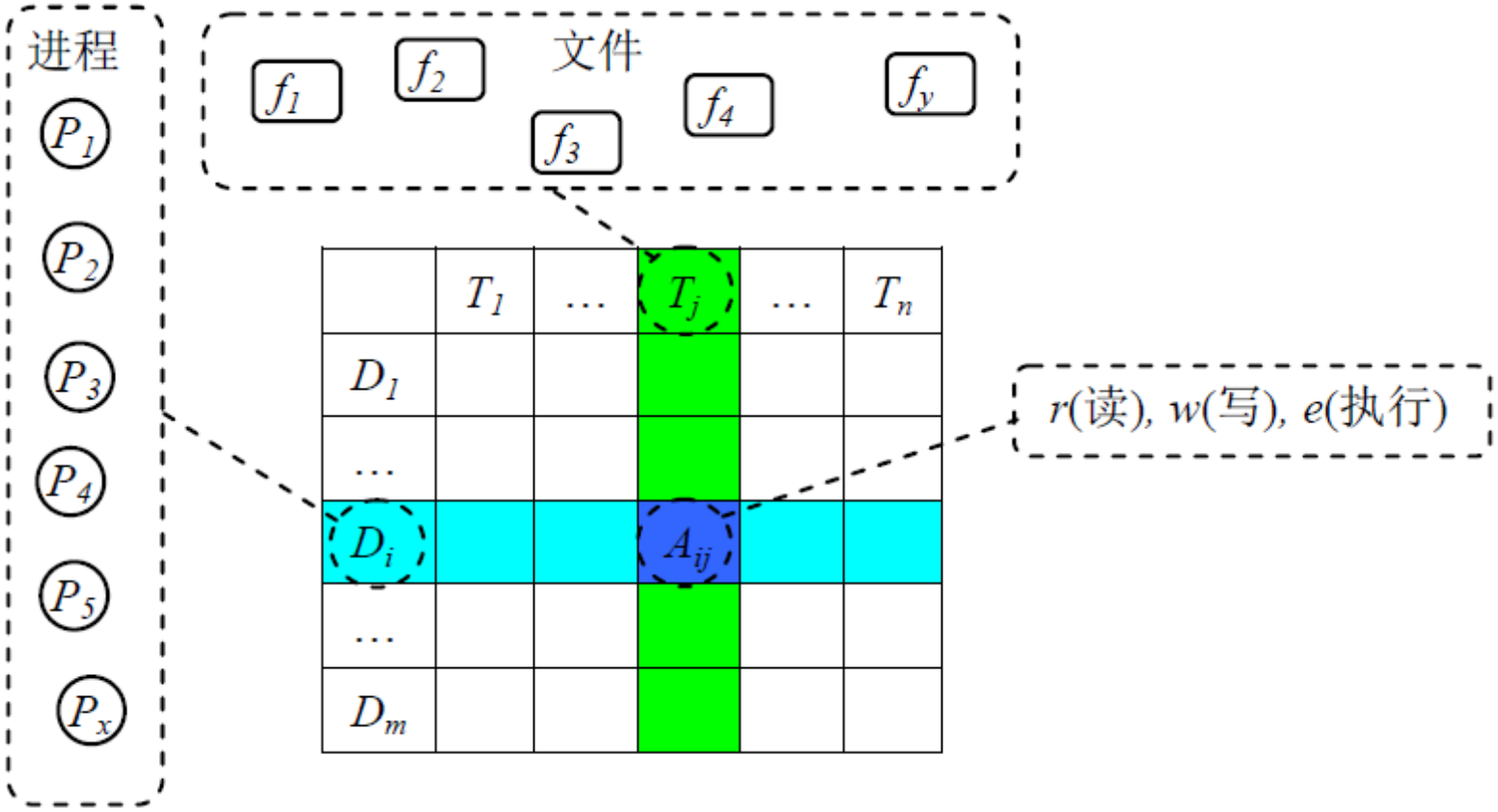
TE模型

- 类型实施模型 TE （Type Enforcement）
- 域类实施模型 DTE （Domain and Type Enforcement）

TE策略描述

- TE将系统视为一个主动实体（主体）的集合和一个被动实体（客体）的集合。
  - （1）每个主体有一个属性–域，每个客体有一个属性–类型，这样所有的主体被划分到若干个域中，所有的客体被划分到若干个类型中。
  - （2）“域定义表”(Domain Definition Table, DDT)，描述各个域对不同类型客体的访问权限。
  - （3）“域交互表”(Domain Interaction Table, DIT)，描述各个域之间的许可访问模式（如创建、发信号、切换）。

域定义表DDT



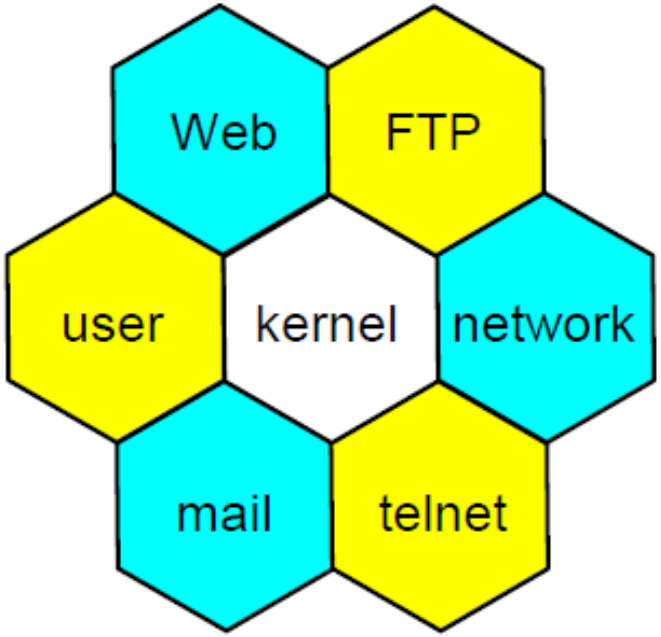
域间作用表DIT

- DIT是用于描述主体对主体的访问权限的二维表，该表的行与列都与域相对应，行与列的交叉点的元素表示行中的域对列中的域的访问权限
- 权限：发信号、创建进程、杀死进程

TE模型

- 为什么TE模型是强制访问控制?
  - 基于域和类型而不是用户标识，进行访问控制
  - DDT和DIT由管理员确定

TE模型应用



TE模型实现的应用隔离

TE模型的不足

- 访问控制权限的配置比较复杂
- 二维表结构无法反映系统的内在关系
  - 目录、父子进程等的层次关系
- 控制策略的定义需要从零开始
  - TE只规定了访问控制框架，没有提供访问控制规则

DTE模型的特点

- 使用高级语言描述访问控制策略
  - 提供DTE语言，用于取代TE模型的二维表，描述安全属性和访问控制配置；
- 采用隐含方式表示文件安全属性
  - 在系统运行期间，利用内在的客体层次结构关系简明的表示文件的安全属性，以摆脱对存储在物理介质上的文件属性的依赖。

DTEL语言的主要功能

- 类型描述
- 类型赋值
- 域描述
- 初始域设定

类型描述语句

```
type unix_t,specs_t,budget_t,rates_t;
```

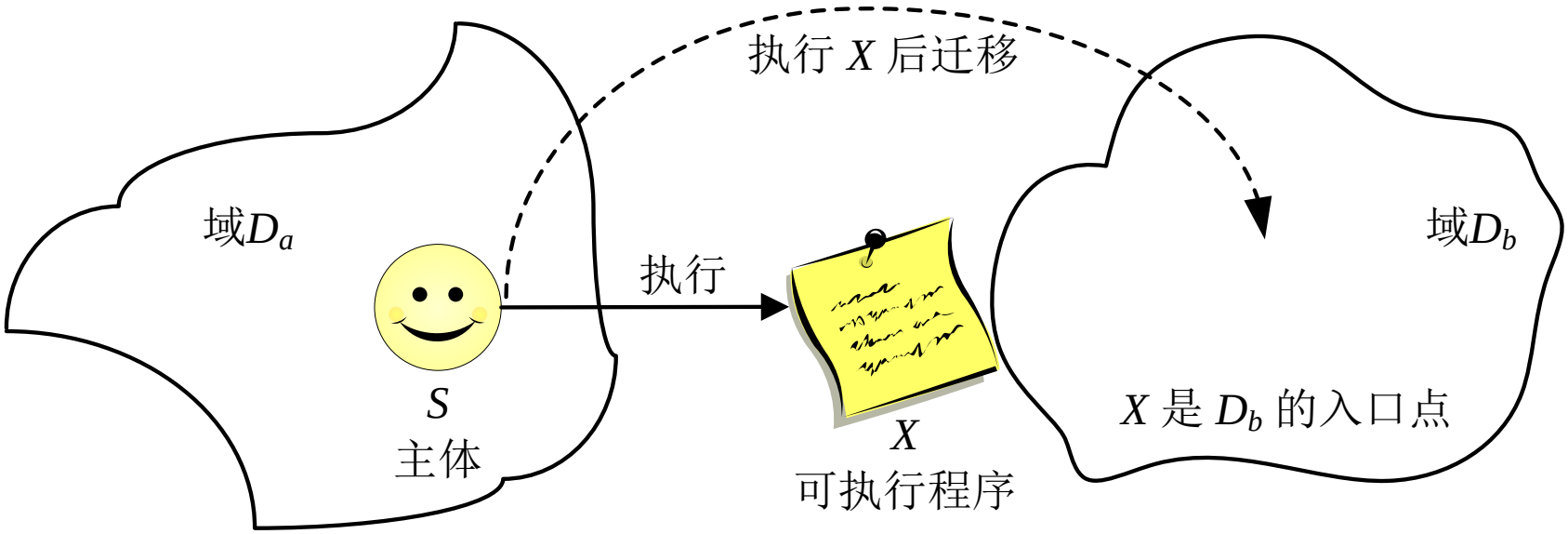
类型赋值语句

```
assign -r -s unix_t     /;
assign -r -s specs_t    /subd/specs;
assign -r -s budget_t   /subd/budget;
assign -r -s rates_t    /subd/rates;
```

访问权限

- 域对类型：r、w、x、d
- 域对域：exec、auto

域的入口点



域描述语句

```
#define DEF (/bin/sh),(/bin/csh),(rxd->unix_t)
domain engineer_d  = DEF,(rwd->specs_t);
domain project_d   = DEF,(rwd->budget_t),(rd->rates_t);
domain accounting_d= DEF,(rd->budget_t),(rwd->rates_t);
```

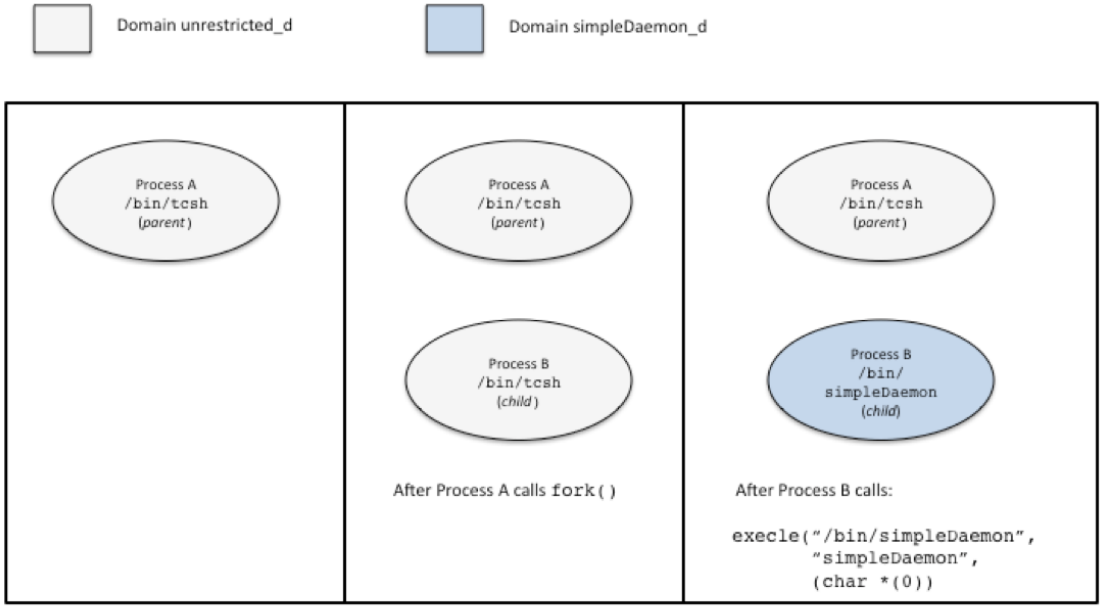
域描述语句

```
domain system_d= (/etc/init),(rwxd->unix_t),(auto->login_d);
domain login_d = (/bin/login),(rwxd->unix_t),
                 (exec->engineer_d,project_d,accounting_d);
initial_domain = system_d;
```

https://acv.cs.mtu.edu/dteTutorial.htm

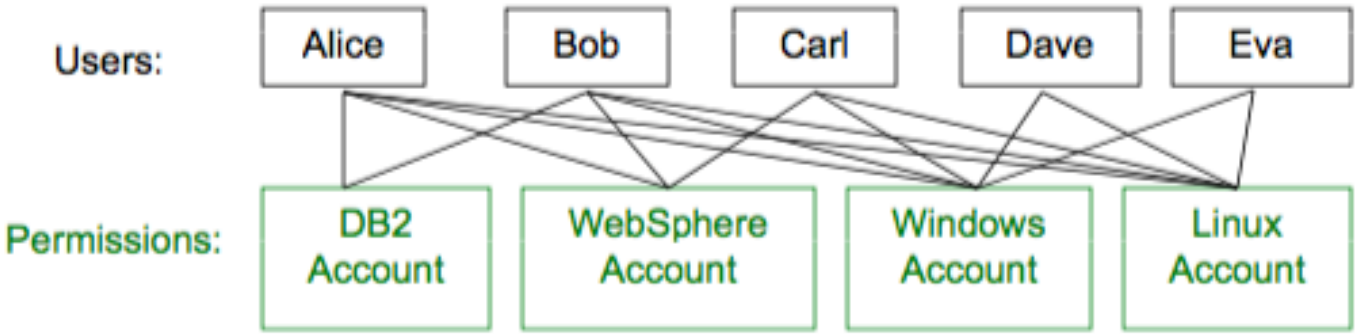
DTE---举例

```
01    type unrestricted_t, simpleDaemon_t;
02
03    domain unrestricted_d = (/sbin/init),
04                            (cdrwx->unrestricted_t),
05                            (drwx->simpleDaemon_t),
06                            (auto->simpleDaemon_d);
07
08    domain simpleDaemon_d = (/bin/simpleDaemon),
09                            (rw->simpleDaemon_t);
10
11    initial_domain = unrestricted_d;
12
13    assign -r unrestricted_t /;
14    assign    simpleDaemon_t /etc/simpleDaemon;
```



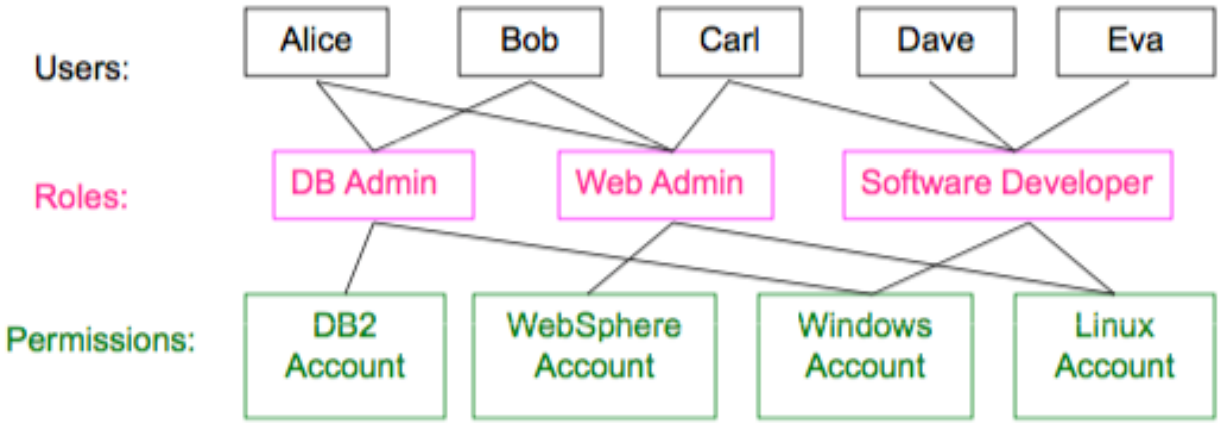
基于角色的访问控制(RBAC)

- 安全管理的复杂性
  - 对于大量的主体和客体，数量授权可以变得非常大
  - 对于动态用户群体，要执行的授予和撤销操作的数量可能变得非常难以管理



基于角色的访问控制

- 组织基于角色进行操作
  - 角色：新的抽象层
- RBAC为组织中的角色分配权限，而不是直接向用户分配权限
- 使用角色，管理的关系就更少了
  - 可能是从O(mn)到O(m+n)，其中m是用户的数量，n是权限的数量。



基于角色的访问控制

- 角色更加稳定
  - 用户可以很容易地从被一个角色重新分配给另一个角色
  - 当新的应用程序和系统被合并时，角色可以被授予新的权限，并且可以根据需要撤销角色的权限
  - 分配给角色的权限相对缓慢地变化
- 让管理员在现有角色中授予和撤消用户成员资格，而不必授权他们创建新角色或更改角色权限。
  - 将角色分配给用户需要的技巧比给角色分配权限要少

组vs.角色

- 区别
  - 用户集合vs. 用户和权限集合
  - 角色可以被激活和停用，组不能
    - 可以使用组通过负授权以阻止访问
    - 角色可以停用以实现最小特权(least privilege)
- 角色可以枚举其具有的权限
  - 角色与函数关联
- 角色形成层次结构

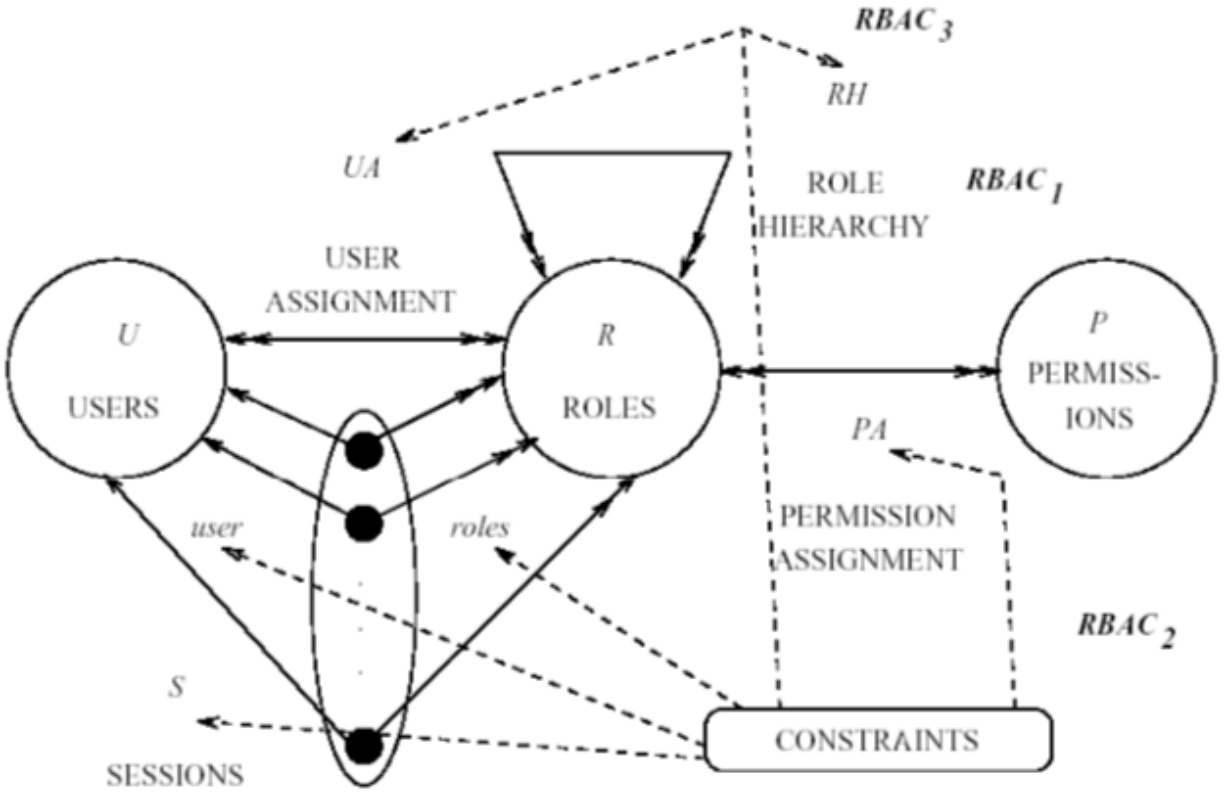
基于角色的访问控制 （RBAC）

- 简化授权管理
  - 主体-角色-客体 vs. 主体-客体
  - 为各种功能创建角色
  - 用户根据职责分配角色
- 表达组织策略
  - 职责分离（SOD，Separation of Duty）
    - 定义不能由同一用户执行的冲突角色
  - 权力下放 （Delegation of authority）
- 支持
  - 最小特权
  - SOD
  - 数据抽象

RBAC-基本概念

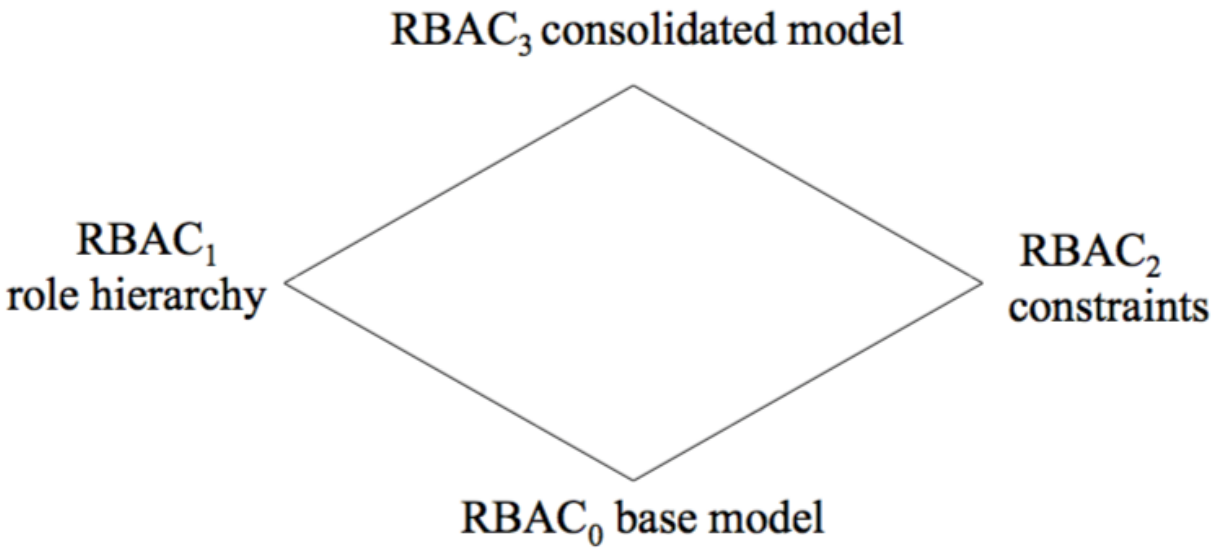
- 用户(user)、权限(permission)、角色(role)
- 权限分配: role-Permission
- 用户分配: user-role
- 会话: session, 动态激活的角色子集

RBAC模型

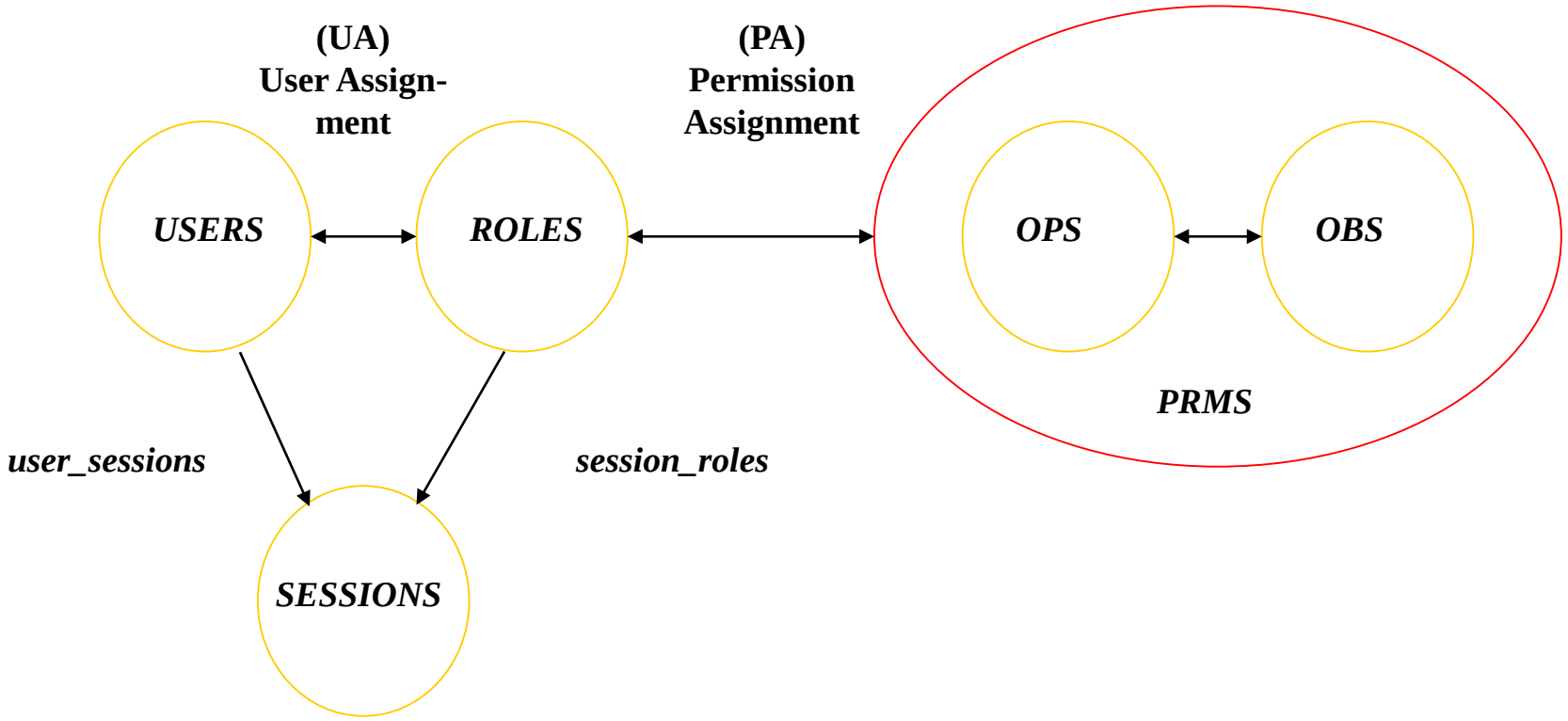


R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. *Role-based Access Control Models. IEEE Computer*, 29(2):38--47, February 1996

RBAC模型


RBAC3 consolidated model
RBAC1
role hierarchy
RBAC2
constraints
RBAC0 base model

$RBAC_0$


(UA)
User Assign-
ment
(PA)
Permission
Assignment
USERS
ROLES
OPS
OBS
PRMS
user_sessions
session_roles
SESSIONS

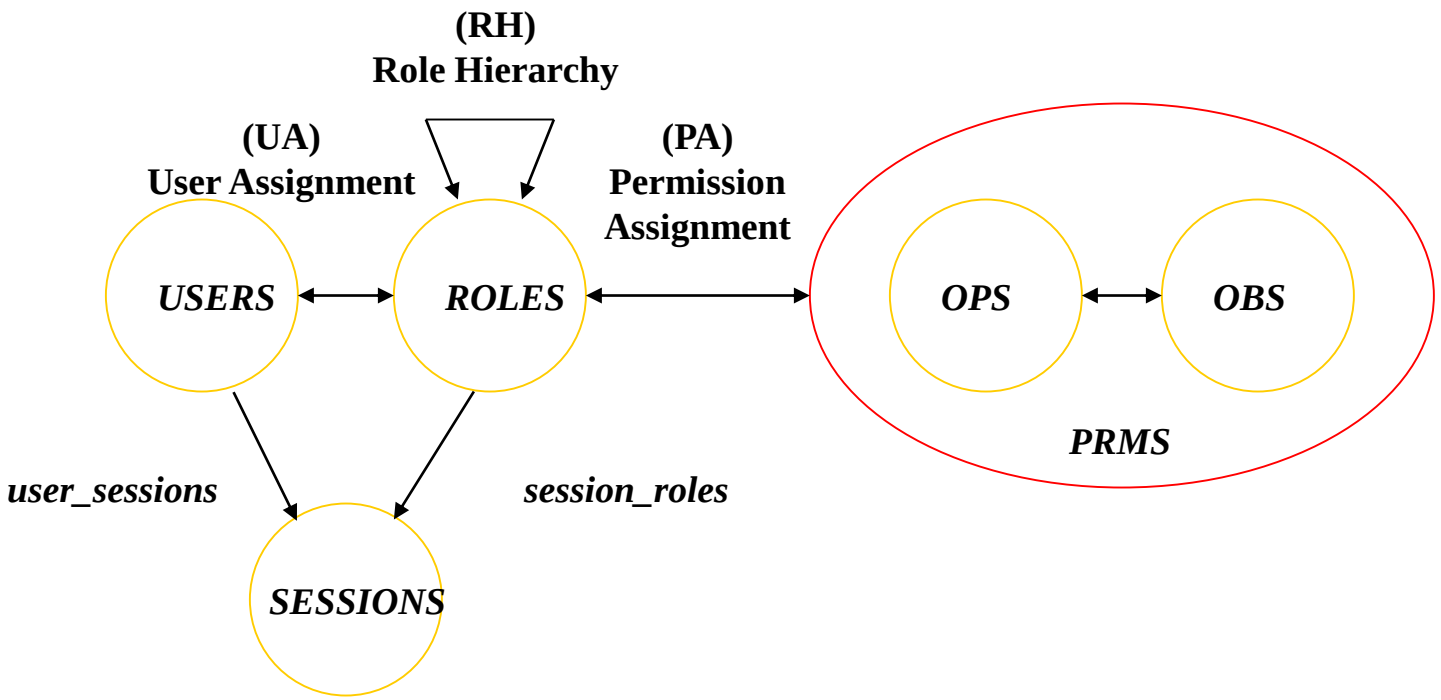
$RBAC_0$

- UA: 用户分配
  - 多对多
- PA: 权限分配
  - 多对多
- 会话:用户映射到多个角色
  - 多个角色可以同时激活
  - 权限：所有角色的权限的合并
  - 每个会话与单个用户相关联
  - 用户可以同时拥有多个会话

$RBAC_0$

- 权限仅适用于数据和资源对象
- 管理权限：修改u，r，s，p
- 会话(session)：在用户的控制下
  - 激活任何允许的角色子集
  - 改变会话中的角色

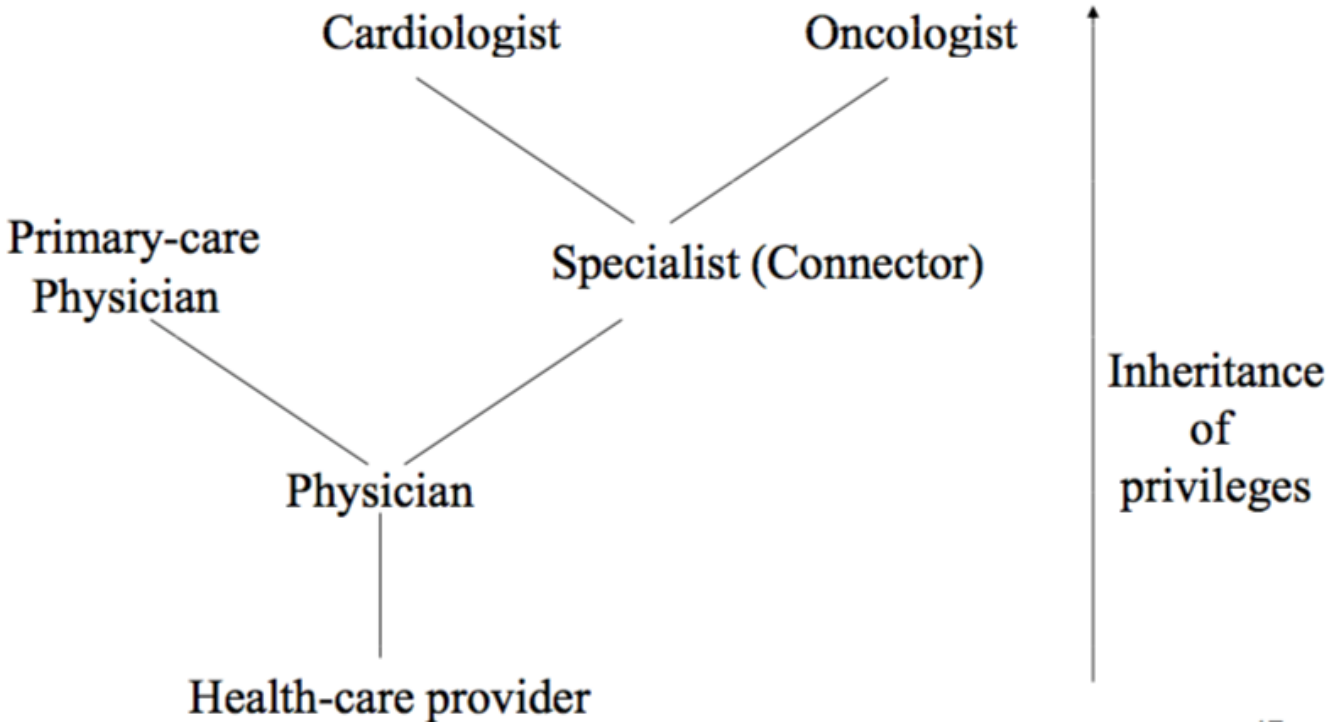
$RBAC_1$--$RBAC_0$ + Role Hierarchy



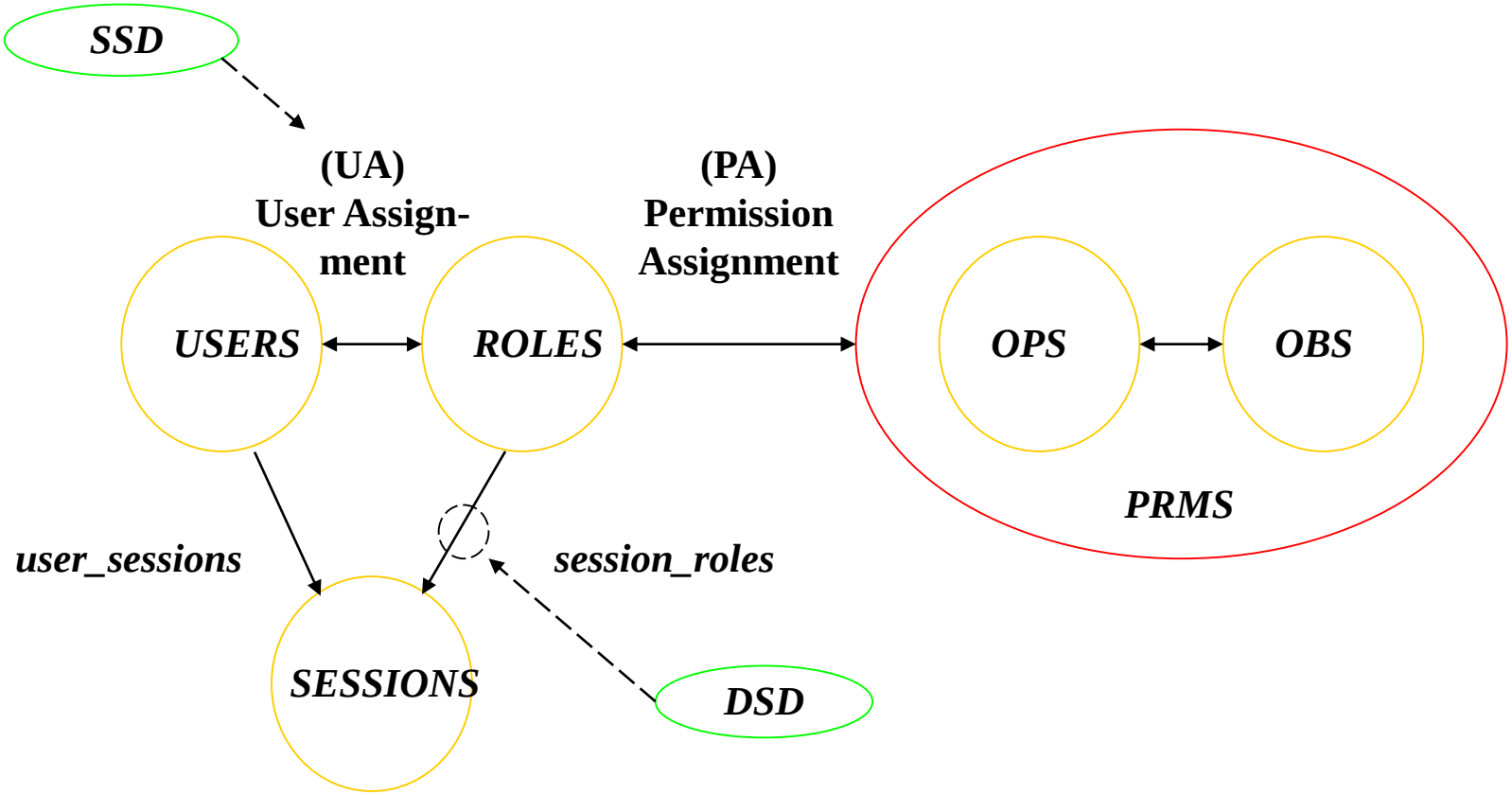
$RBAC_1$--$RBAC_0$ + Role Hierarchy

- 角色层次结构，用于构造角色以反映组织的职权范围和职责
- 从初级角色（下）到高级角色的权限继承（上）
- 偏序关系

$RBAC_1$--角色层次


Cardiologist
Oncologist
Primary-care Physician
Specialist (Connector)
Physician
Health-care provider
Inheritance of privileges

$RBAC_2$—$RBAC_0$ + Constraints



$RBAC_2$—$RBAC_0$ + Constraints

- 执行高层次的组织政策
  - 相互分离的角色：职责分离（SOD，Separation of duties）
    - UA：同一用户不能兼任客户经理和采购经理
    - 违规只会因合谋而引起
  - 权限分配的约束
    - PA：签发支票的权限不能同时分配给会计和采购经理（限制高权限的分发）
  - 基数：
    - 一个角色可以有的最多成员数量
    - 每个用户的最大角色数
    - 在执行最小数目时的问题？
    - 也适用于PA
  - 其他：限制运行时的角色数量（每个session）或基于历史或先决条件
    - 例如，如果用户被分配project角色，用户只能被分配给testing role；
    - 如果读取目录的权限被分配给角色，则将读取文件的权限也将被分配给该角色

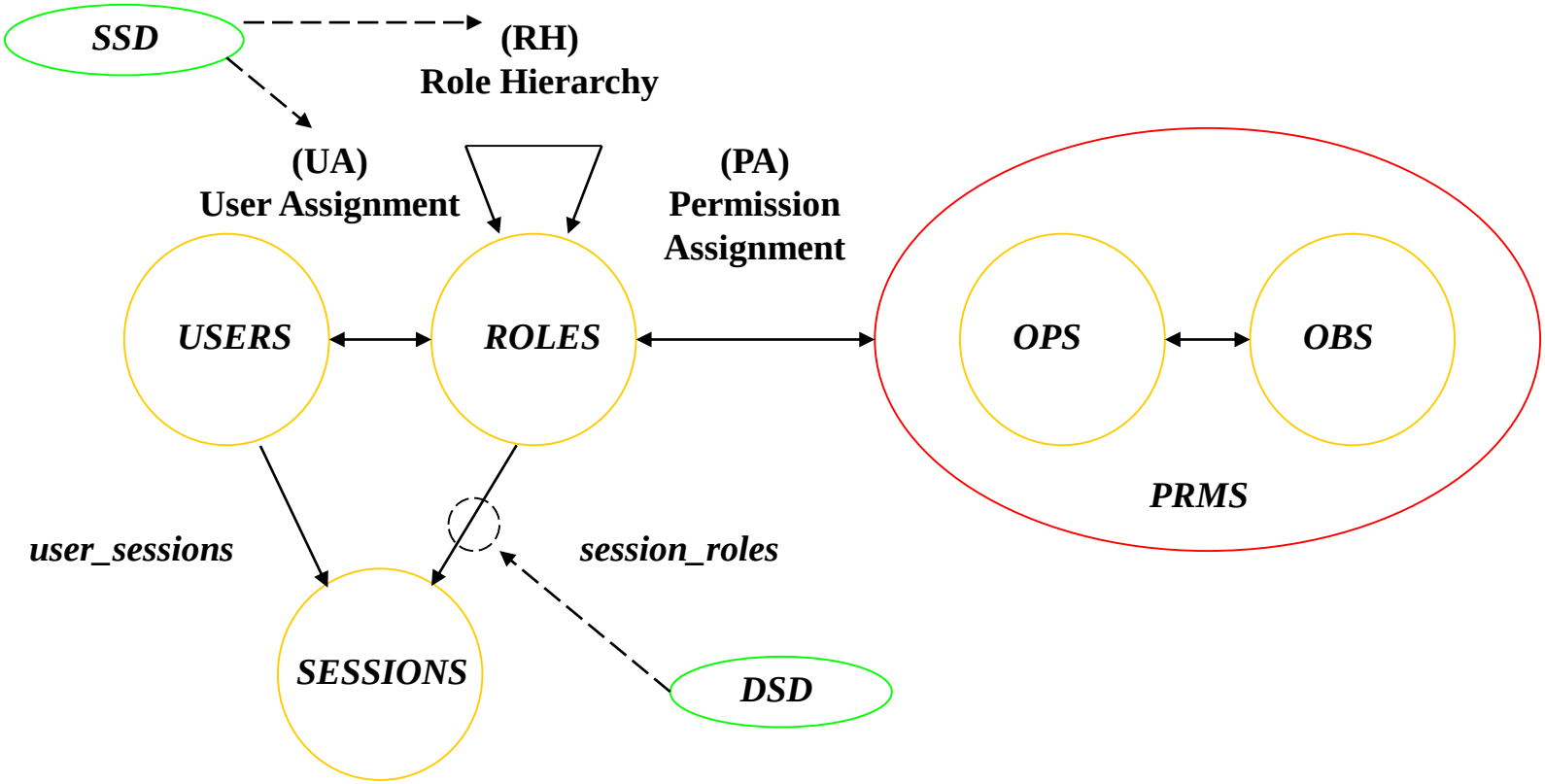
$RBAC_2$--静态SoD约束

- 静态SOD对角色的集合进行了限制
- 在一组*m*角色中，没有用户分配给*t*或多个角色
- 防止某人被授权使用太多的角色
- 可以根据分配给每个角色的用户强制执行这些约束

$RBAC_2$—动态SoD约束

- 这些约束限制了用户可以在单个会话中激活的角色数量
- 约束的例子：
  - 用户不能从每个用户会话的角色集合中激活t或者更多角色
  - 如果用户在会话中使用角色r_1，则他/她不能在同一会话中使用角色r_2
    - 如果用户在一个角色中终止一个会话，并使用另一个角色登录，该怎么办？
- 执行这些角色需要保持用户对会话中角色的访问历史记录

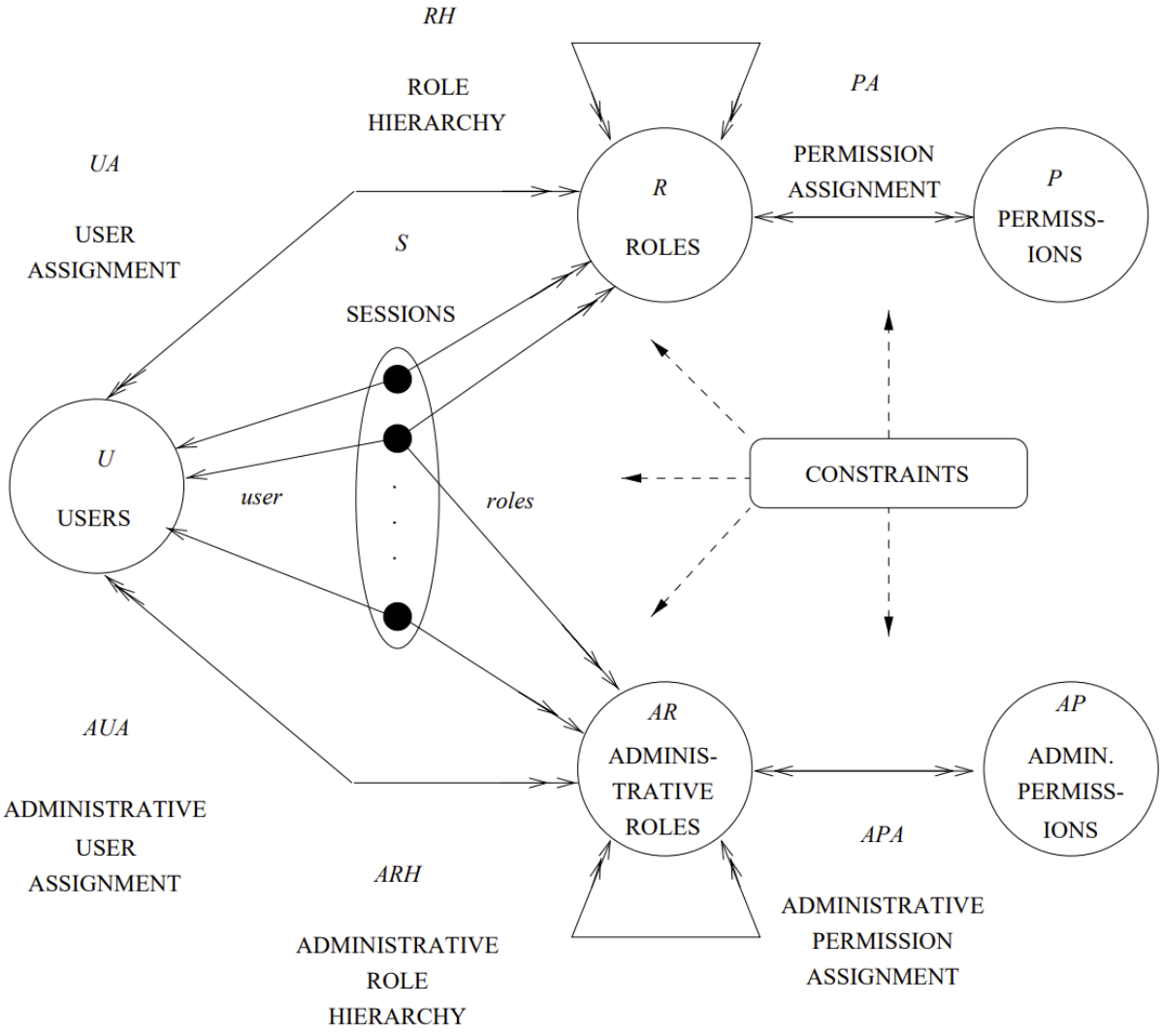
$RBAC_3$—$RBAC_1$ + $RBAC_2$



$RBAC_3$—$RBAC_1$ + $RBAC_2$

- 约束可以适用于角色层次
  - 例如 2个或更多角色不能具有普通的高级/初级角色
  - 例如：限制特定角色可能拥有的高级/低级角色的数量
- RH与约束之间的相互作用
  - 例如：Programmer和Tester是相互排斥的。 Project supervisor继承两组权限。 如何处理?
  - 例如，基数约束，基数约束只适用于直接成员，还是继承成员?

RBAC Models (+ Administrative Roles)



RBAC System and Administrative Functional Specification

- Administrative Operations
  - Create, Delete, Maintain elements and relations
- Administrative Reviews
  - Query operations
- System Level Functions
  - Creation of user sessions
  - Role activation/deactivation
  - Constraint enforcement
  - Access Decision Calculation

莫科尔树模型

- 完整性度量模型
- 针对的问题
  - 设计一个算法，使得对于任意一个数据项D_i（$1 \le i \le n$），该算法能够快速验证该数据项的完整性，并且，占用较少的内存空间。

$$D = D_1 || D_2 || \ldots || D_n$$

递归函数 $f(i,j,D)$

(1) $f(i,i,D) = h(D_i)$

(2) $f(i,j,D) = h(\,f(i,\,(i+j-1)/2,\,D)\,\|\,f((i+j+1)/2,\,j,\,D))$

i 和 j 是自然数（$1 \le i \le j \le n$）

举例

- 假设：$D=D_1||D_2||\ldots||D_8$
- 已知：哈希函数为h，$f(1,8,D)$是已知且正确的，试给出运用递归函数$f(i,j,D)$验证数据项D_5的完整性的过程。

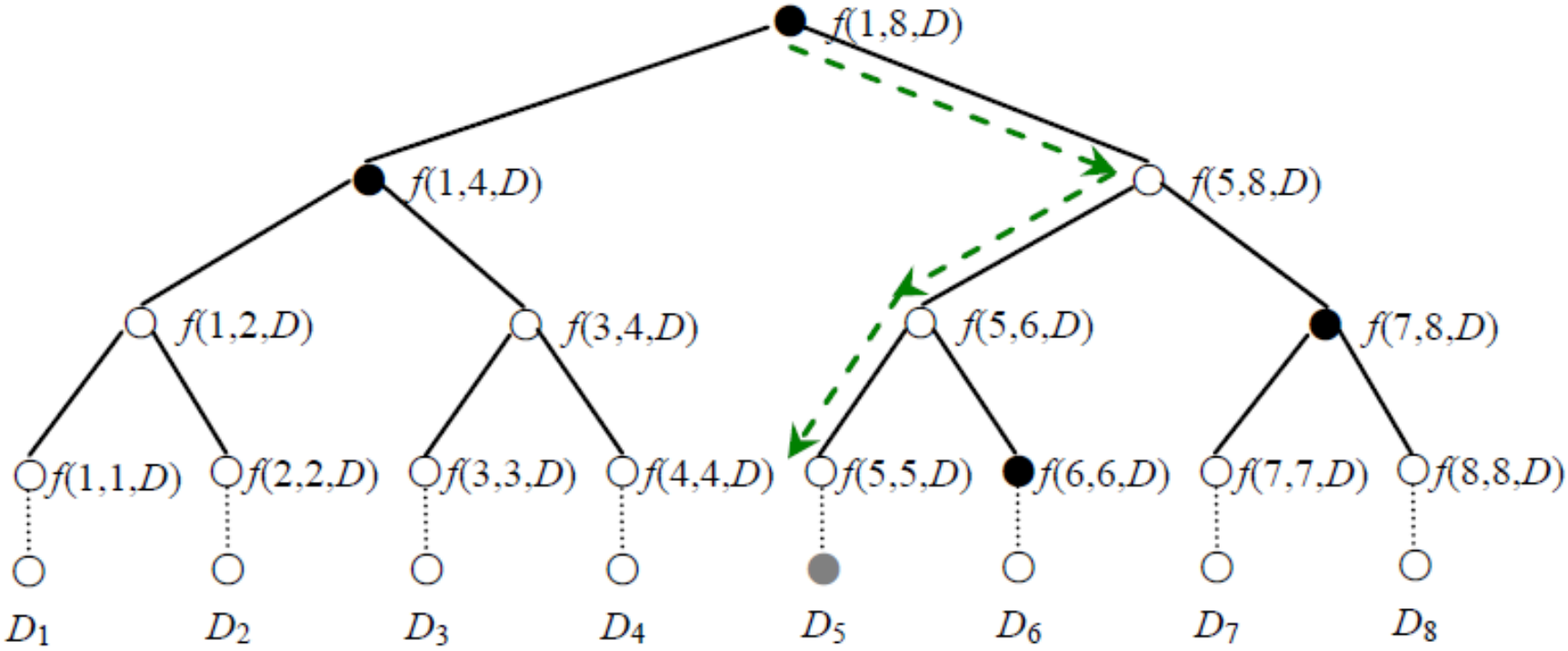
$$f(1,8,D)=h(f(1,4,D)\,\|\,f(5,8,D))$$

$$f(5,8,D)=h(f(5,6,D)\,\|\,f(7,8,D))$$

$$f(5,6,D)=h(f(5,5,D)\,\|\,f(6,6,D))$$

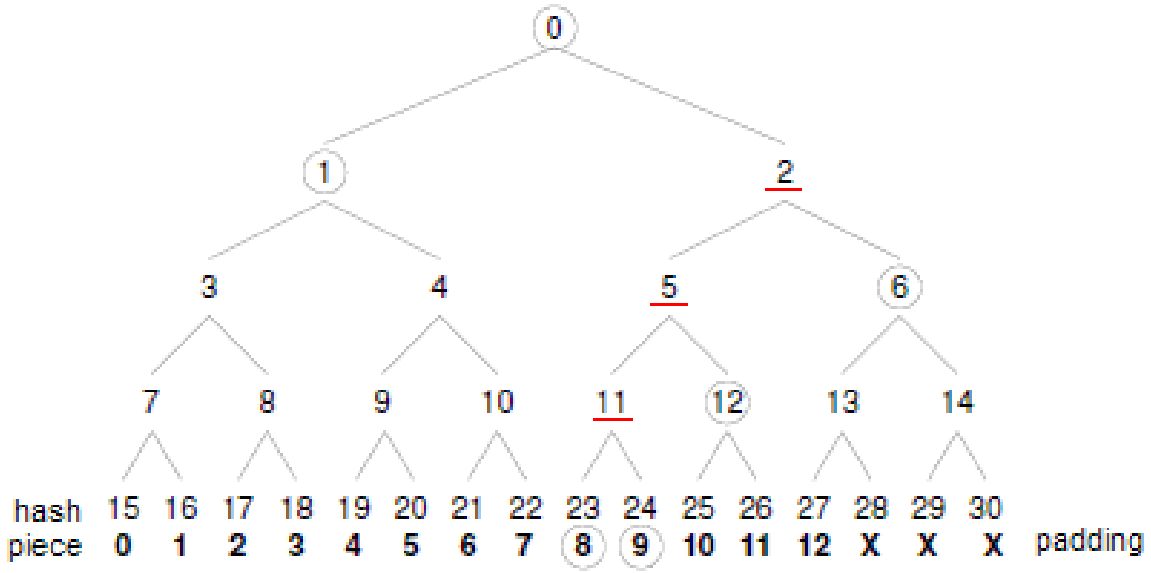
$$f(5,5,D)=h(D_5)$$

完整性验证路径

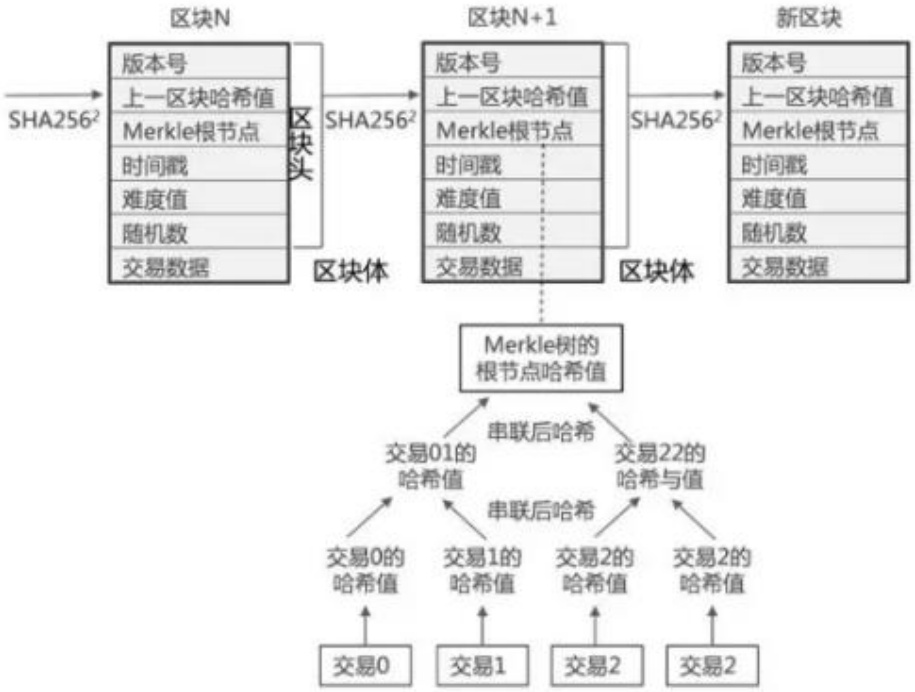


$f(1,8,D)$、$f(1,4,D)$、$f(7,8,D)$、$f(6,6,D)$

莫科尔树的应用



http://blog.csdn.net/houzengjiang/article/details/8635392



思考

- 分析BLP存在的隐蔽通道问题的原理。
- 说明C-W模型的实施规则是从哪几个方面进行完整性访问控制的？它的证明模型是从哪几个方面确保完整性访问控制的有效性的？
- 分析访问控制矩阵和域定义表的相同和不同之处。
- 莫科尔树模型的目标是以较少的内存开销实现较快的数据完整性验证，请以其某种应用为例，分析它是如何实现这一目标的。

BLP模型中的隐蔽通道问题源于它主要关注于防止信息的非法流动，但并未完全阻止所有未授权信息传递的可能性。在BLP中，即使数据项的读取和写入权限得到了控制，两个进程仍可能通过共享访问某些数据对象来传递信息，比如一个进程修改了数据对象的状态，另一个进程随后读取这个状态变化，从而实现隐蔽通信。

C-W模型的实施规则主要从职责分离、数据标签和完整性子集等方面进行完整性访问控制。它要求不同任务由不同个体执行，数据项和操作都带有标签，且定义了数据的完整性约束。证明规则则通过数据项完整性的验证、操作的完整性保障以及访问控制的完整性审查来确保控制的有效性。

访问控制矩阵与域定义表相比，访问控制矩阵是一个列出了系统中所有主体对所有客体访问权限的二维表，而域定义表则定义了主体可以访问的客体集合以及客体可被哪些主体访问的规则。两者的相同之处在于都定义了主体与客体之间的访问关系，不同之处在于它们的表示方式和应用侧重点。

莫科尔树模型通过构建哈希值的树状结构来实现数据完整性的快速验证。**在文件系统或分布式数据库等应用中**，莫科尔树将数据分割成多个区块，计算每个区块的哈希值，并构建一个树状结构，其中每个父节点的哈希值基于其子节点的哈希值。这种结构允许通过比较根哈希值来快速验证数据集的完整性，因为任何对叶节点数据块的篡改都会最终影响根哈希值，从而被检测出来。莫科尔树模型的优势在于它以较少的内存开销实现了快速的数据完整性检查。