电子科技大学

University of Electronic Science and Technology of China

第七章

访问控制

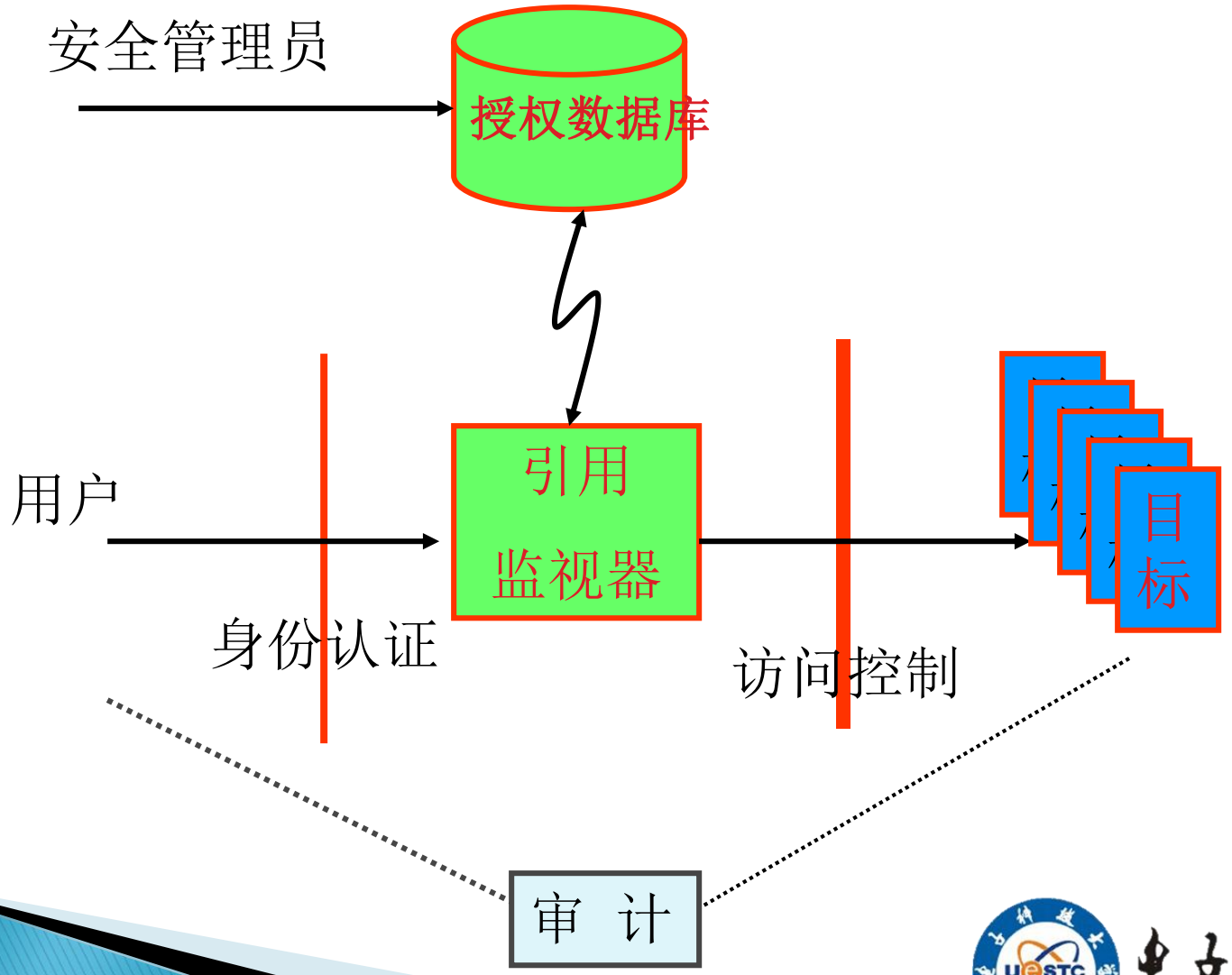
内容

- 访问控制基本概念
- 访问控制模型
- 访问控制的安全策略
- 访问控制实现技术

访问控制的概念

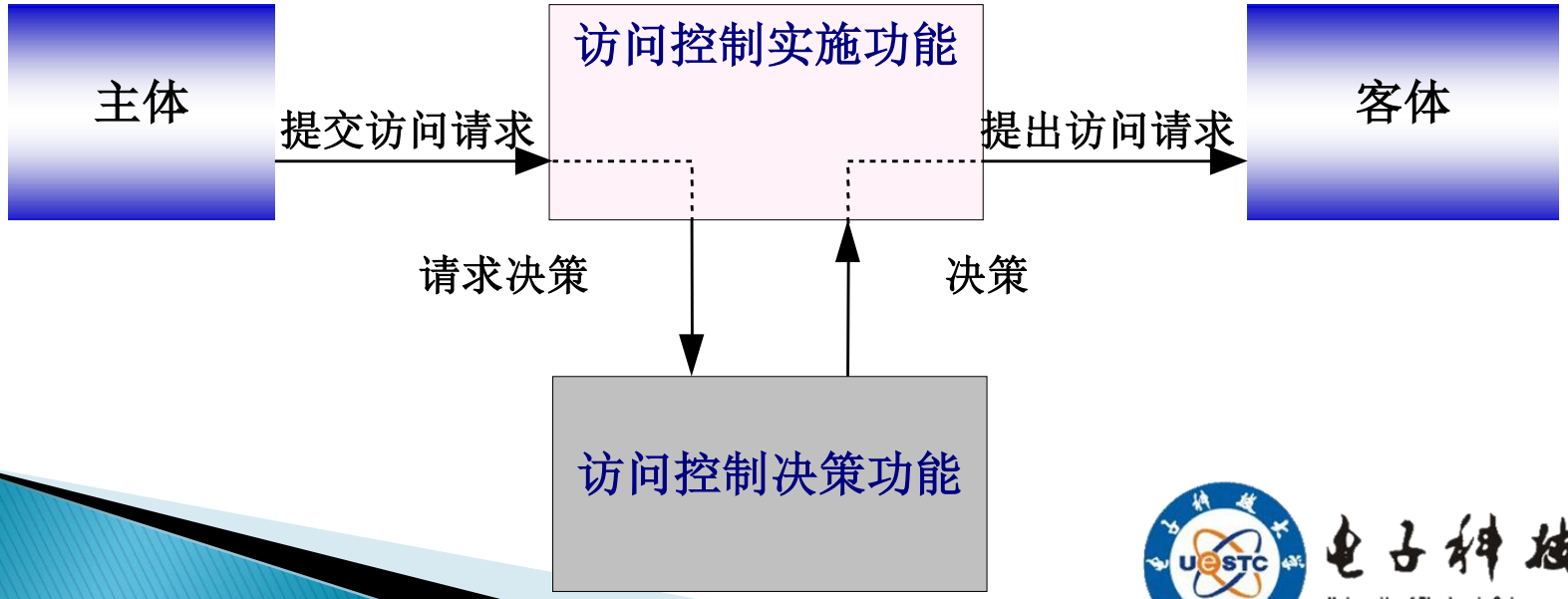
- 网络安全防护的主要**安全策略**之一
- 依据授权规则，对提出的资源访问加以控制。
  - 限制访问主体（用户、进程、服务等）对任何资源（计算资源、通信资源或信息资源）进行**未授权访问**，使计算机系统在合法范围内使用。防止：
    - 非法用户使用
    - 合法用户滥用权限
  - 决定用户能做什么，或代表用户的程序能做什么。

访问控制与其他安全措施的关系模型



访问控制的组成

- 访问控制三要素
  - 主体Subject、客体Object、安全访问策略
- 形式化描述
  - 三元函数 f(s,a,o)

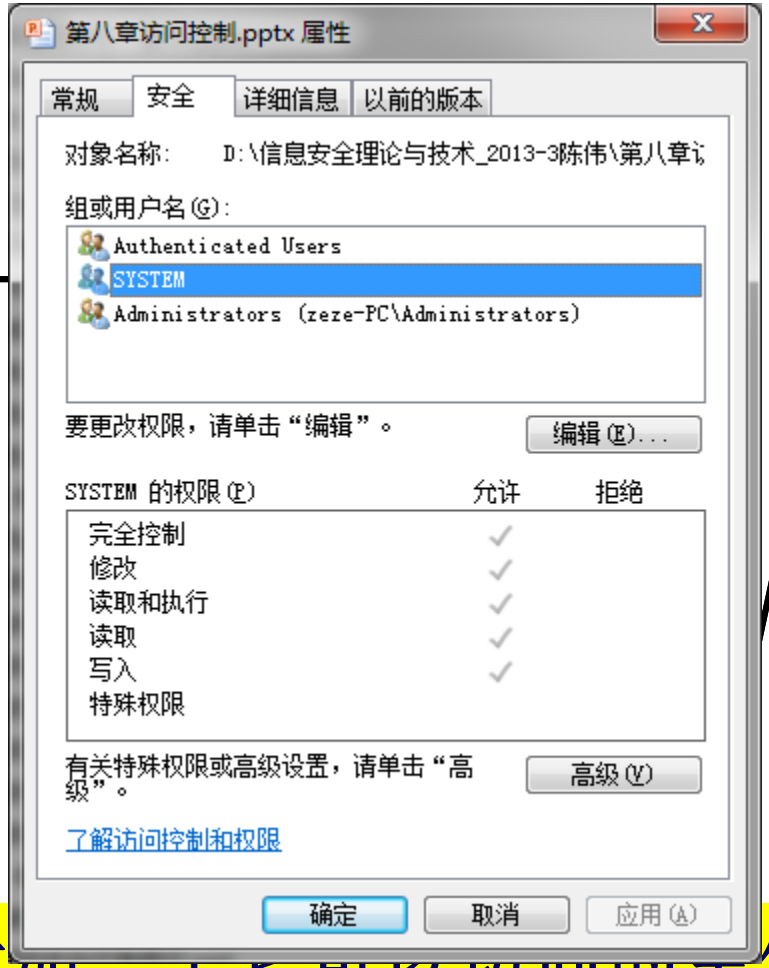


访问控制的一般实现机制和方法

- 一般实现机制
  - 基于访问控制属性：访问控制表/矩阵
  - 基于用户和资源分级（“安全标签”）：多级访问控制
- 常见实现方法
  - 访问控制表ACLs（Access Control Lists)
  - 访问能力表（Capabilities)
  - 访问控制矩阵
  - 授权关系表
  - 访问控制安全标签
  - 其它

访问控制表(ACL)

Obj1



userC

R

W

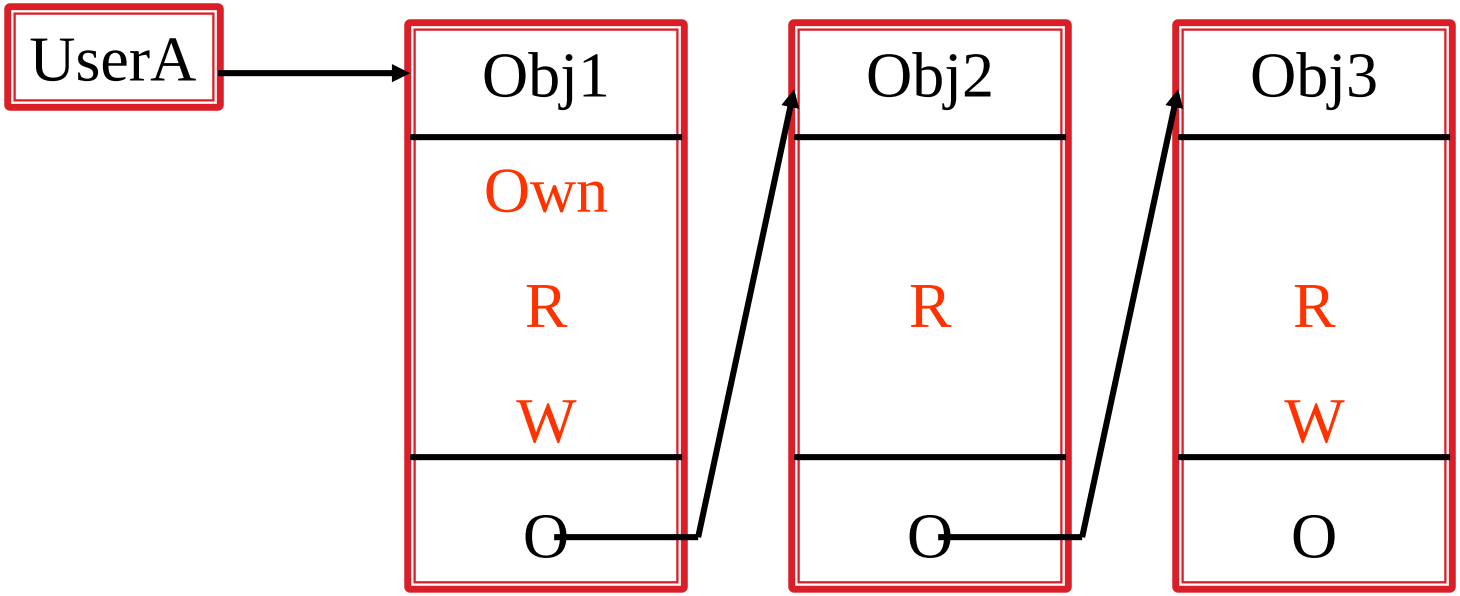
O

每个客体附加一个包含该访问的主体的明细表。

访问控制表ACL

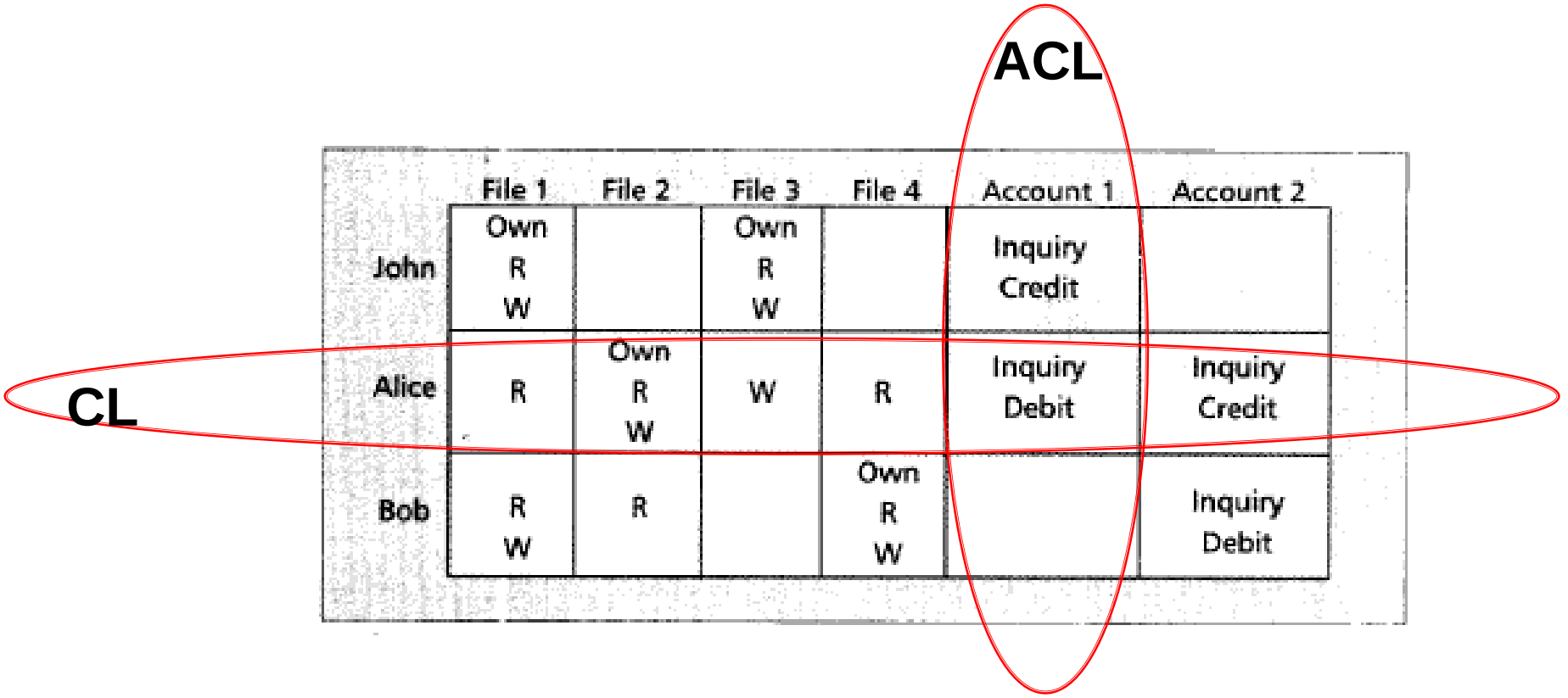
- 访问控制粒度为用户：当用户数量多、管理资源多时，ACL会很庞大。
- 缺省策略：
  - 公开布告板中，所有用户都可以得到所有公开信息
  - 对于特定的用户禁止访问：对于违纪员工，禁止访问内部一些信息。
- 当组织内人员变化（升迁、换岗、招聘、离职）、工作职能发生变化（新增业务）时，ACL的修改变得异常困难。

访问能力表(CL)



每个主体都附加一个该主体可访问的客体的明细表。

访问控制矩阵



授权关系表

UserA	Own	Obj1
UserA	R	Obj1
UserA	W	Obj1
UserB	W	Obj2
UserB	R	Obj2

- 按客体排序：访问控制表
- 按主体排序：访问能力表

访问控制安全标签

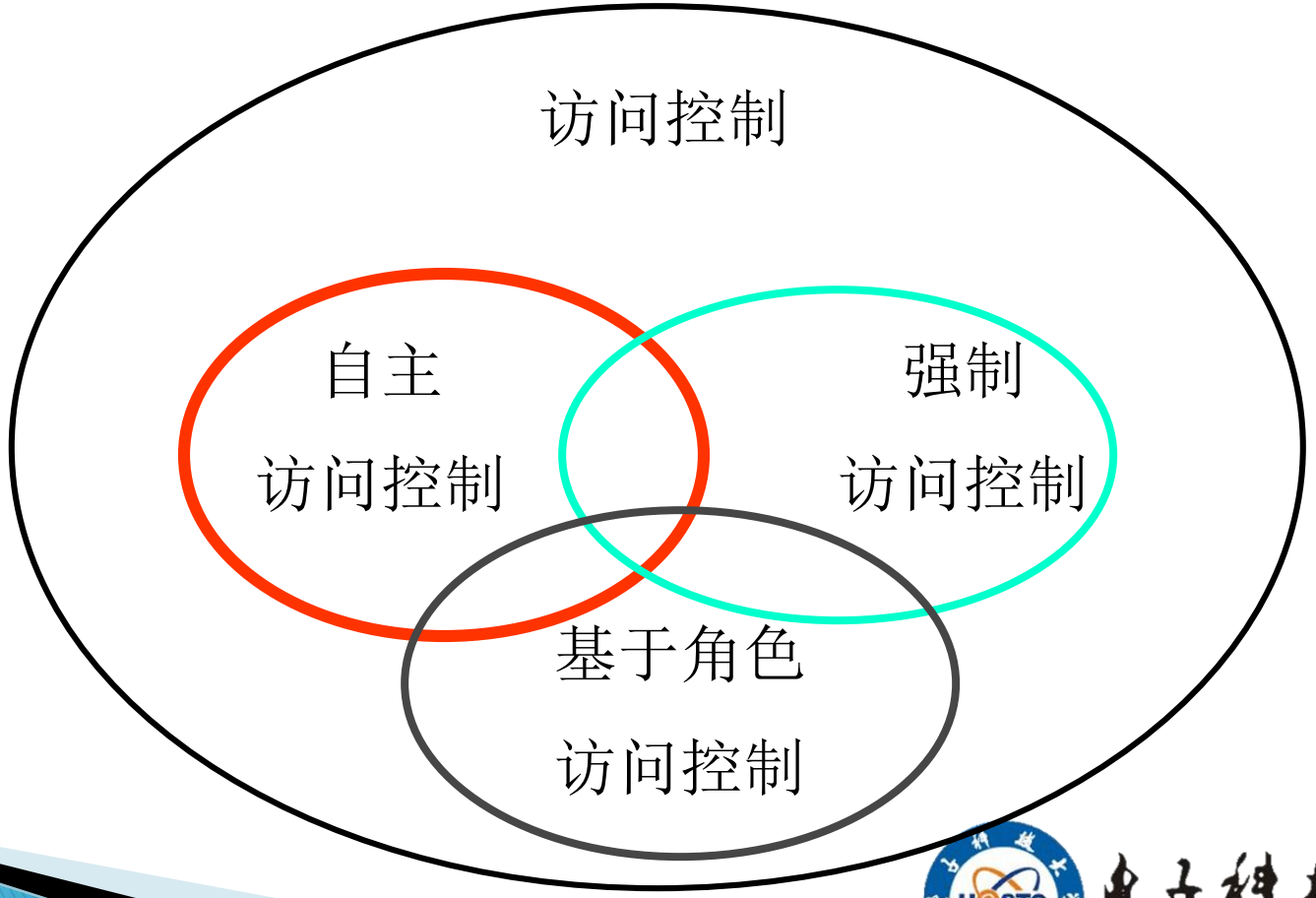
用户	安全级别
用户1	绝密
用户2	机密
……	……
用户n	未分类

客体	安全级别
客体1	绝密
客体2	机密
……	……
客体n	未分类

锁与钥匙

- 客体由一个密钥加密（锁），主体拥有解密密钥（钥匙）
- 同时具有访问控制表与能力表的特征
- 具有动态性

访问控制的一般策略



自主访问控制（DAC）模型

- 客体属主自主管理对客体的访问权限
  - 属主自主负责赋予或回收其他主体对客体资源的访问权限
  - 授权主体可以直接或者间接地向其他主体转让访问权
- Linux权限管理
  - -rwx r-x r-x
  - 主 组 其它
- chmod,chown

自主访问控制

- 特点：
  - 根据主体身份及权限进行决策；
  - 授权主体能自主地将其权限的某个子集授予其它主体；
  - 灵活性高，被大量采用。
- 缺点：
  - 访问权限关系会改变
  - 无法控制信息（权限）的流动
    - Eg，小人书借阅

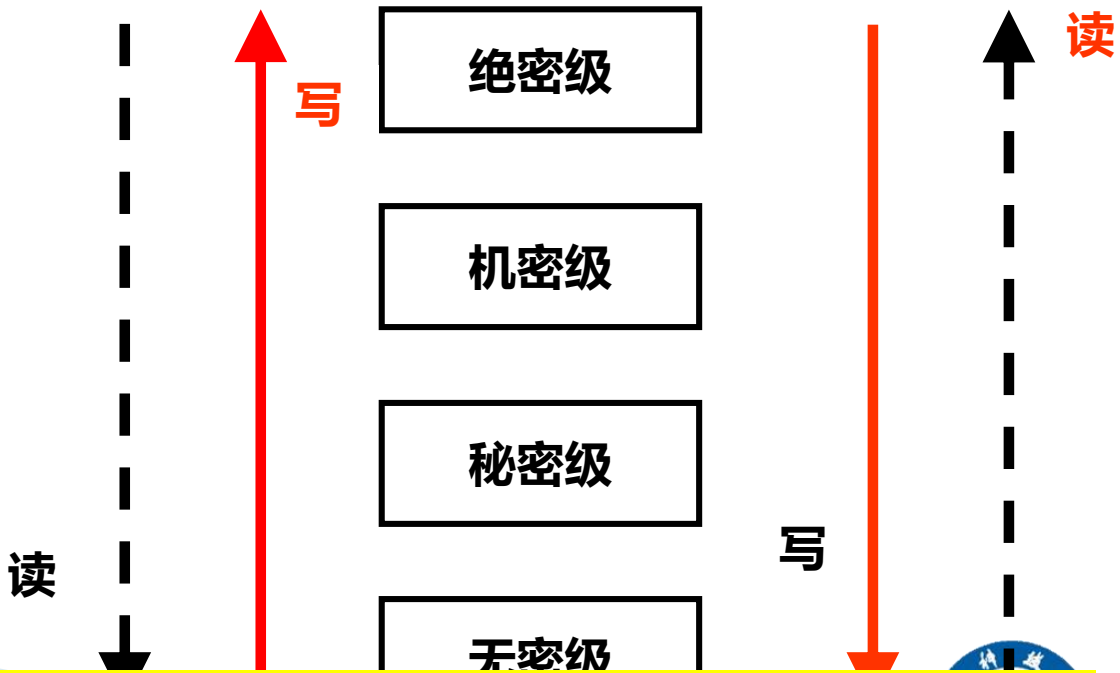
强制访问控制（MAC）模型

- 每个主体和客体分配一个固定的安全级别，只有系统管理员才可以修改
  - 用户：可信任级别
  - 信息：敏感程度
  - 绝密、机密、秘密、无密
- 依据主体和客体的安全级别决定是否允许访问
- 主要用于多层次安全级别的军事应用中

强制访问控制

- 依据主体和客体的安全级别，MAC中主体对客体的访问有四种方式：
  - 下读/上写
  - 上读/下写

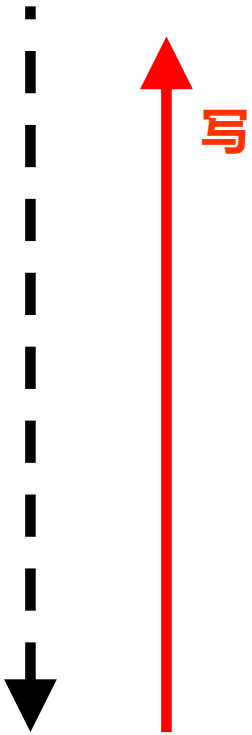
读/写代表信息流动方向，如“生产者/消费者”



“上/下”：方向，有些资料反过来表达

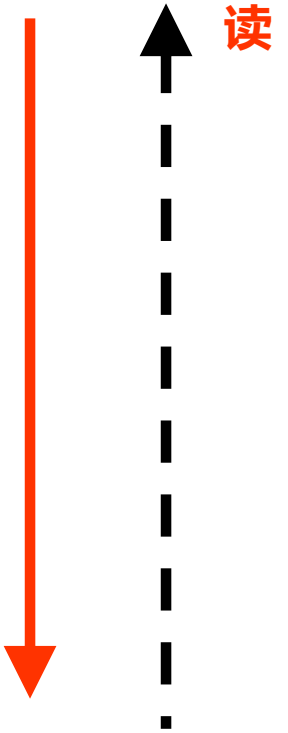
强制访问控制——下读/上写

- 向下读（Read Down，rd）
  - 主体高于客体时允许读
  - 低级别用户不能读高敏感度的信息
- 向上写（Write Up，wr）
  - 主体低于客体时允许写
  - 不允许高敏感度的信息写入低敏感度区域
- 保证数据机密性
  - 信息流只能从低级别流向高级别
  - 如，下级向上级汇报工作或情况



强制访问控制——上读/下写

- 向上读（Read Up，ru）
  - 主体低于客体时允许读操作
  - 低信任级别的用户能够读高敏感度的信息
- 向下写（Write Down，wd）
  - 主体高于客体时允许写操作
  - 允许高敏感度的信息写入低敏感度区域
- 保证数据完整性
  - 信息从高级别流向低级别
  - 如,上级像下级下发文件、精神

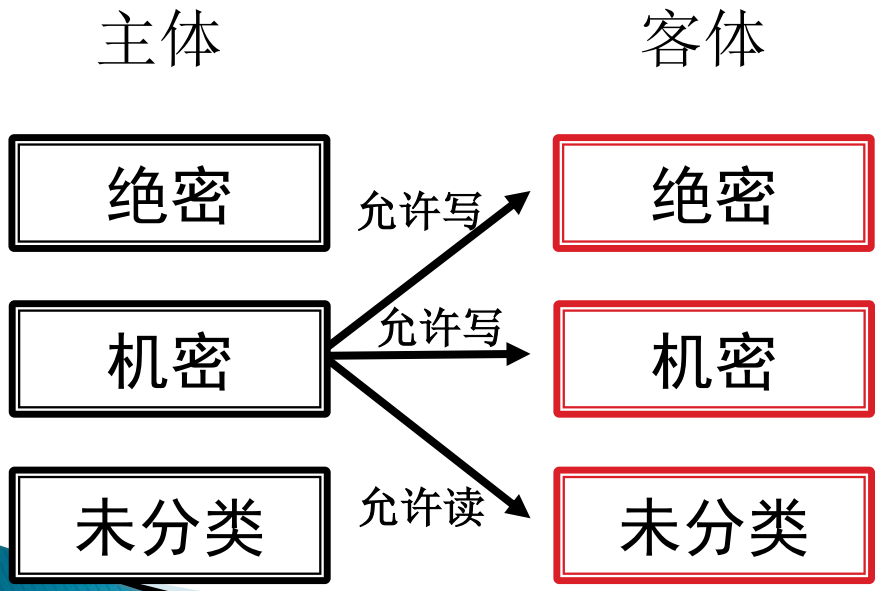


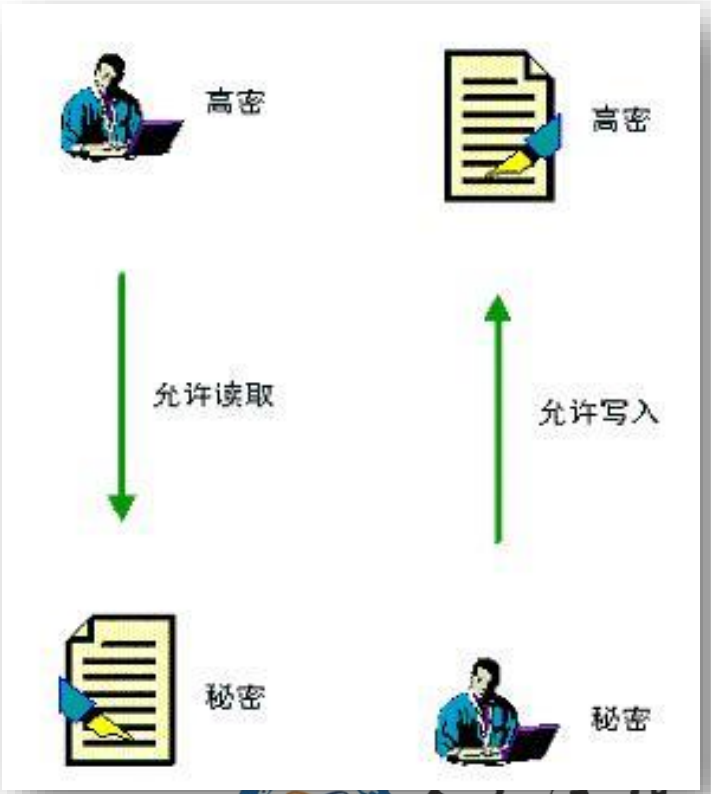
常见的强制访问控制模型

- 通过访问方式组合
  - Bell-Lapadula(BLP)模型
  - Lattice 模型
  - Biba

Bell－LaPadula(BLP)模型

- 下读/上写 (不上读/不下写)
- 保证机密性





BLP应用：防火墙

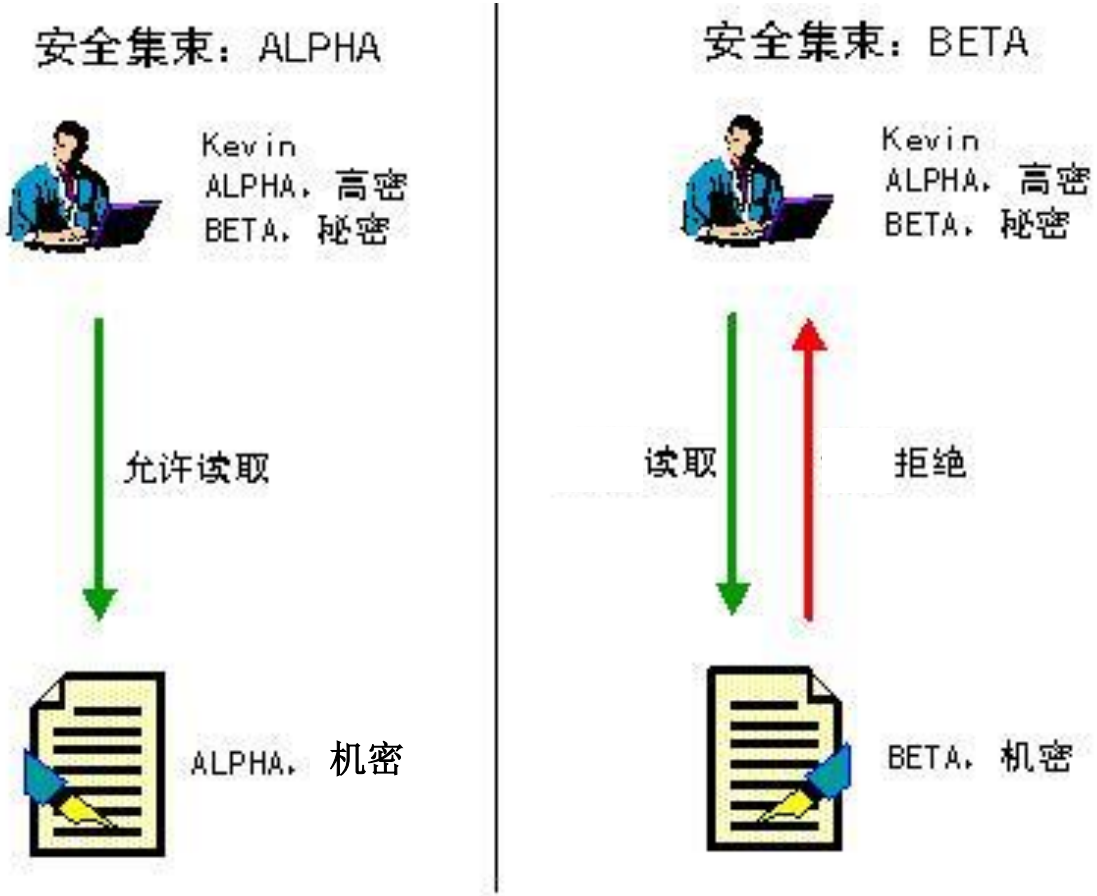
- 安全级别
  - 内部网络：机密
  - 外部Internet：公开
- 隔离内外部网络——单向访问机制
  - 不上读：阻止Internet访问内部网络，仅允许由内向外发起的数据流通过
  - 不下写：不允许敏感数据从内部网络流向Internet

Lattice模型

- 本质上同BLP模型相同，通过划分安全边界对BLP模型扩充。
  - 将用户和资源进行分类，形成安全集束（部门，组织等）。
  - 安全集束内的主体和客体划分安全等级，
    - 一个主体可同时从属于多个安全集束，
    - 一个客体仅能位于一个安全集束。
  - 下读上写：前提条件是各对象位于相同安全集束。

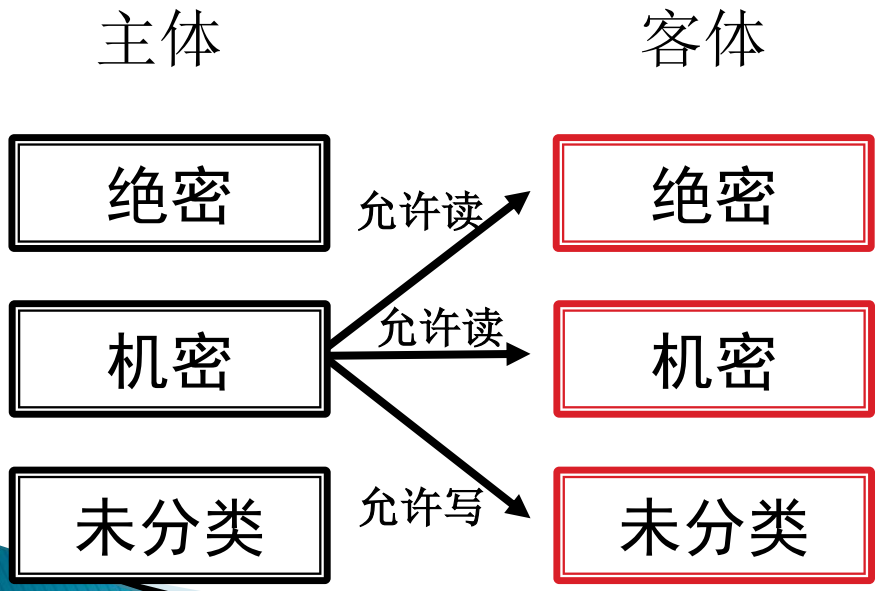
Lattice模型

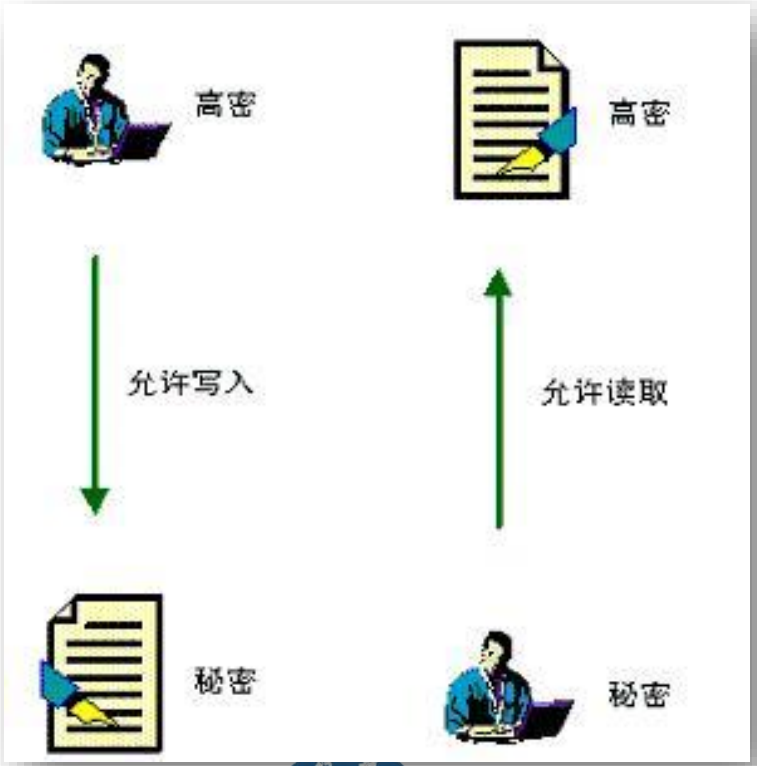
- 在不同安全集束间控制信息的流动，而不仅是垂直检验其敏感级别。



Biba模型

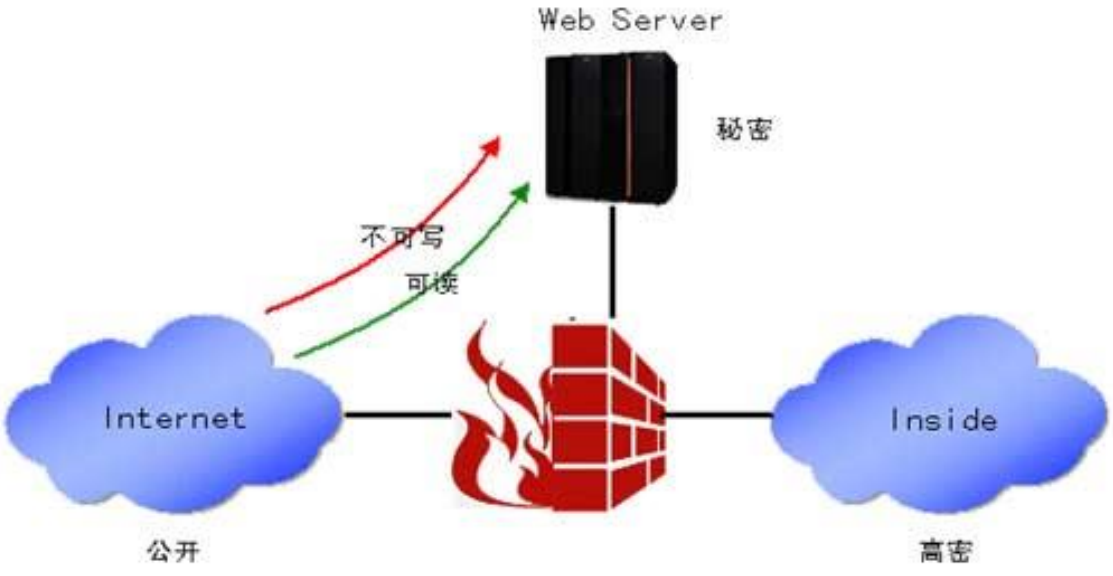
- 上读/下写（不下读/不上写）
- 保证完整性





Biba应用：Web服务器

- 安全级别
  - Web服务器上资源安全级别为“秘密”
  - Internet用户级别为“公开”
- 上读/不上写，保障Web数据完整性
  - Internet上的用户只能读取服务器上的数据而不能更改它



MAC优缺点

- 优点
  - 严格，便于控制和管理
  - 特别适用于军事领域
- 缺点
  - 不灵活
  - 用户、资源多时配置工作量大

自主/强制访问控制策略的问题

- 自主式太弱
  - 配置的粒度小；配置的工作量大，效率低
  - 无法控制信息（权限）的流动
- 强制式太强
  - 配置的粒度大；缺乏灵活性
- 二者工作量大，不便管理
  - 例：1000主体访问10000客体，须1000万次配置。如每次配置需1秒，每天工作8小时，就需
  - 10，000，000/（3600*8）=347.2天

基于角色的访问控制策略

- RBAC(Role Based Access Control)与现代的应用环境相结合的产物
- 起源于UNIX系统或别的操作系统中组的概念

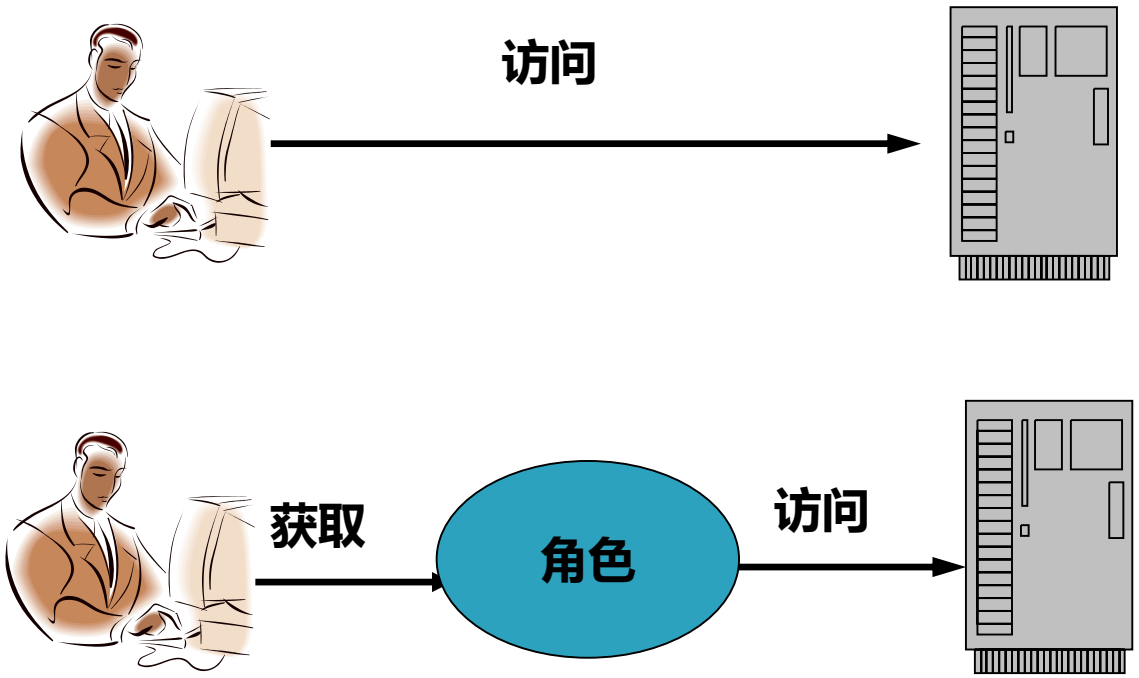
基于组的策略

- 一组用户对一个目标具有同样的访问许可。
- 实际使用时
  - 先定义组的成员；
    - 用户的集合　$G=\{s_1, s_2, s_3 \ldots\}$
  - 对用户组授权；
    - 把访问权限分配给一个用户组；
  - 组的成员可以改变。

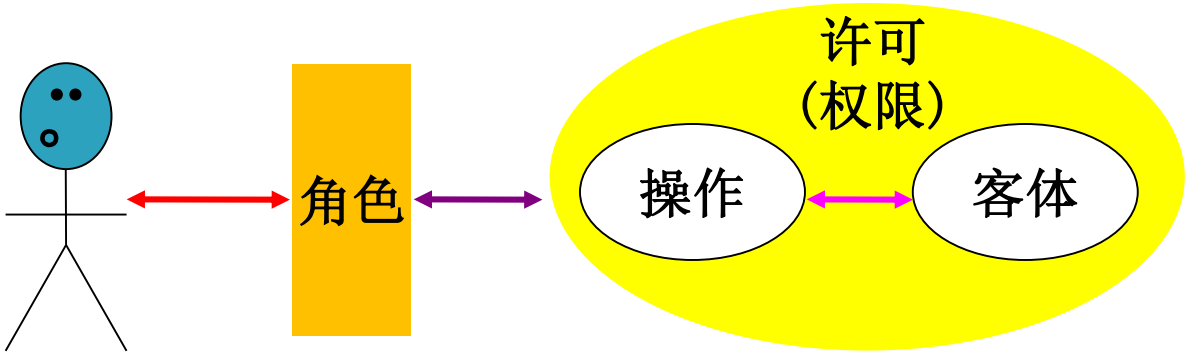
基于角色的访问控制(RBAC)

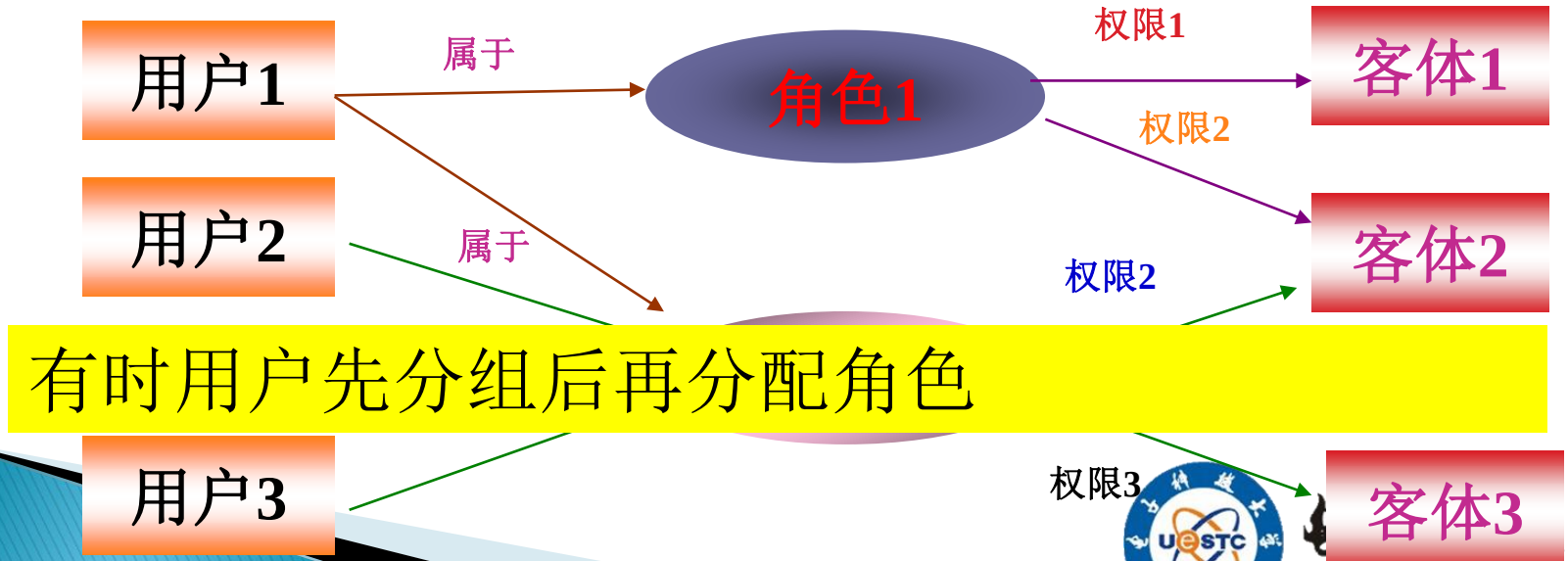
- 角色（Role）
  - 用户组及其许可（完成一项任务必须访问的资源及相应操作权限）的集合，R={(a1,o1), (a2,o2), (a3,o3)…}
  - 角色：用户组＋许可（资源-权限）集
- 授权管理：
  - 根据任务需要定义角色
  - 为角色分配许可——（资源和操作权限）
  - 给一个用户指定一个角色

RBAC与传统访问控制的差别



基于角色的访问控制模型





用户、角色、许可的关系（三多）

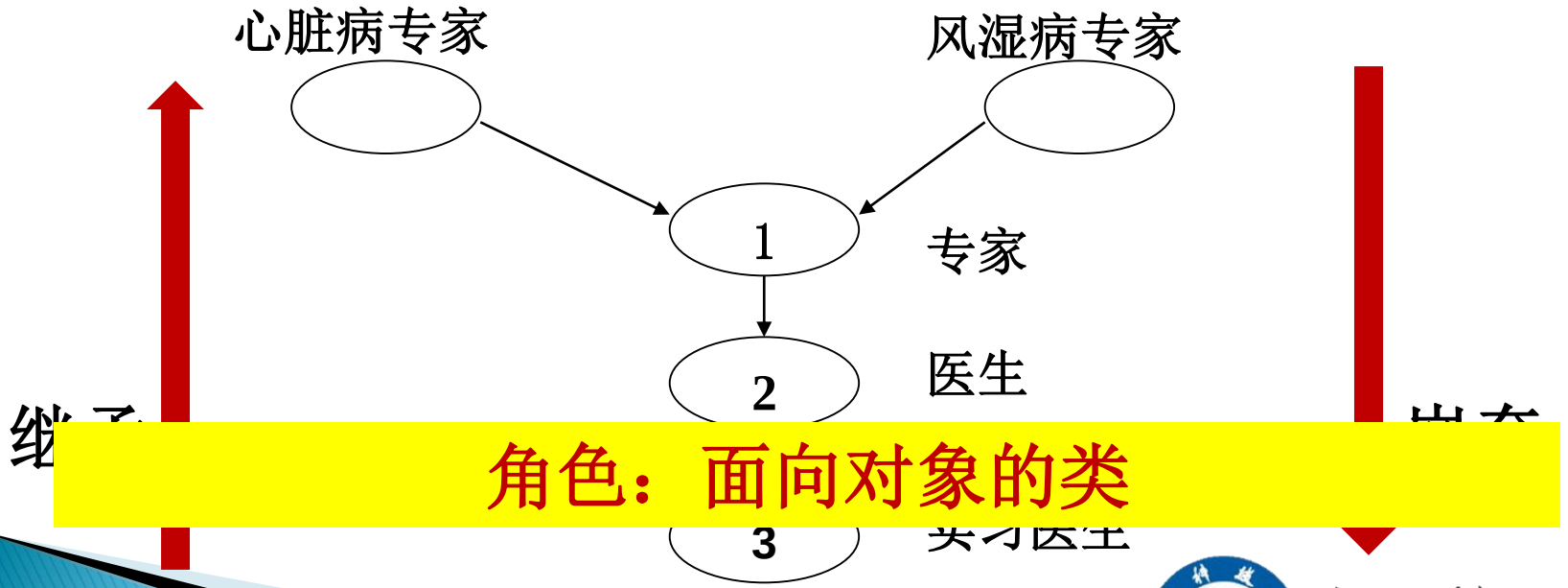
- 用户、角色多对多
  - 一个用户可经授权而拥有多个角色
  - 一个角色可授予多个用户
- 角色、许可多对多
  - 每个角色可拥有多种许可（权限）
  - 每个许可也可授权给多个不同的角色
- 许可=操作+客体，操作、客体多对多
  - 每个操作可施加于多个客体（受控对象）
  - 每个客体也可以接受多个操作。

基于角色的访问控制实例——银行

- 角色：
  - 出纳员、分行管理者、顾客、系统管理者和审计员。
- 授权管理：
  - 出纳员：允许修改顾客帐号记录；
  - 分行管理者：允许修改顾客帐号记录，创建和终止帐号；
  - 顾客：允许查询自己的帐号；
  - 系统管理者：允许开关系统，但不允许读或修改用户的帐号信息；
  - 审计员：允许读系统中任何数据，但不允许修改任何事情。

角色继承

- 避免角色权限重复设置
  - 角色有自己许可，还能继承其他角色的许可。
- 可用祖先关系来表示继承。
  - 角色2是角色1的“子角色”，继承了角色1的部分许可



角色：面向对象的类

角色管理——分配与授权

- 系统管理员
  - 设定角色
  - 设定权限
  - 为用户分配角色、取消角色等
  - 为角色分配权限、撤销权限等

角色激活

- 用户：
  - 静态概念
- 会话：
  - 动态概念
  - 用户的一个活跃进程，代表用户与系统交互。
- 会话构成用户到角色的映射：
  - 会话中分配角色
  - 会话激活用户授权角色集的某个子集——活跃角色集。

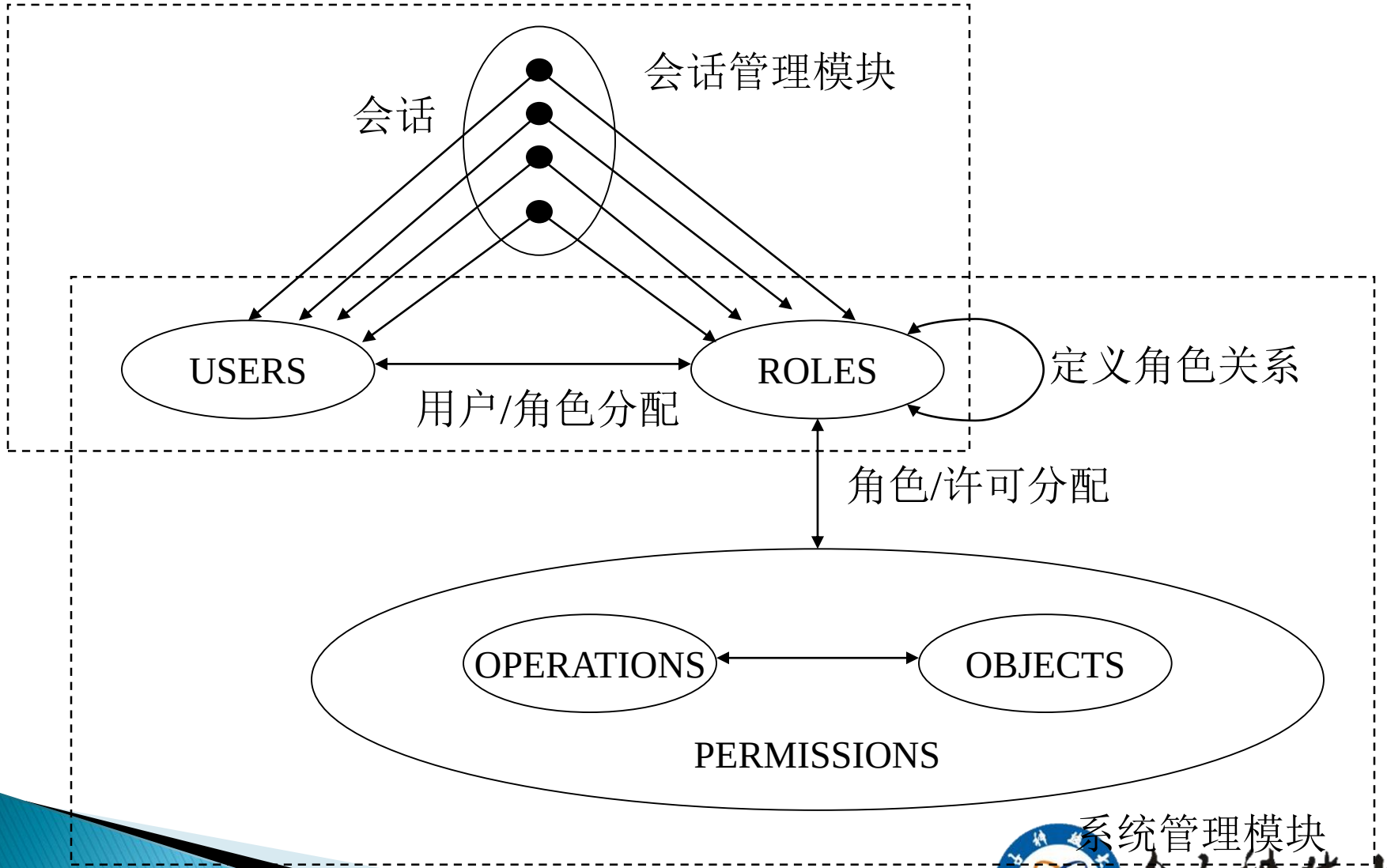
角色限制

- 角色互斥：
  - 对于某些特定的操作集，某一个用户不可能同时独立地完成所有这些操作。
- 角色互斥：
  - 静态角色互斥
  - 动态角色互斥
- 角色基数限制：
  - 创建角色时，要指定角色的基数。在一个特定的时间段内，有一些角色只能由一定人数的用户占用。
- eg. 杀人游戏

RBAC特点

- 通过角色定义、分配和设置，可描述复杂、灵活的安全策略
- 通过角色分层映射组织结构
- 容易实现最小特权原则
- 能够满足职责分离原则——角色互斥
- 岗位上的用户数通过角色基数约束

RBAC系统结构



RBAC系统的运行步骤

- ①用户登录：
  - 身份认证
- ②检索授权角色集
  - 会话管理模块从RBAC数据库检索用户授权角色集并送回用户。
- ③选择活跃角色集
  - 选择本次会话活跃角色集，其间会话管理模块维持动态角色互斥。
- ④创建会话
  - 体现授权，菜单、按钮
- ⑤会话过程中，系统管理员若要更改角色或许可
  - 在此会话结束后
  - 或终止此会话立即进行。

策略组合

- MAC+RBAC

本章可选平时考核题目

- 其他访问控制技术或策略
- 基于属性的访问控制