网络安全

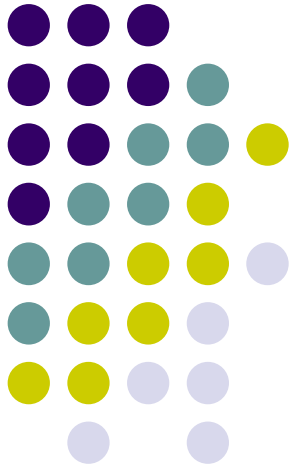
罗　敏

武汉大学计算机学院

mluo@whu.edu.cn

Tel: 13907125177

QQ：5118924



第8章 利用处理程序错误的攻击重点回顾

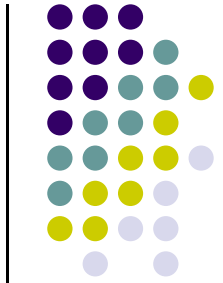
- 系统漏洞及攻防
- Web漏洞及攻防

第9章 访问控制技术

- 本章的主要内容是对入网访问控制、物理隔离、自主访问控制和强制访问控制等技术的实现原理进行了详细的论述，并介绍了一些新型访问控制技术。

第9章 访问控制技术

- 9.1 访问控制技术概述
- 9.2 入网认证
- 9.3 物理隔离措施
- 9.4 自主访问控制
- 9.5 强制访问控制
- 9.6 新型访问控制技术



操作系统安全等级

- 美国国家计算机安全中心(NCSC)于1983年提出并于1985年批准的“可信计算机系统安全评价准则”(TCSEC)。1991年，美国国家计算机安全中心(NCSC)为TCSEC提出可依赖数据库管理系统解释(TDI)。1991年，欧共体发布了“信息技术安全评价准则”（ITSEC）。1993年，加拿大发布了“加拿大可信计算机产品评价准则”（CTCPEC）。同年，美国在对TCSEC进行修改补充并吸收ITSEC优点的基础上，发布了“信息技术安全评价联邦准则”（FC）。1993年6月，上述国家共同起草了一份通用准则（CC），并将CC推广为国际标准。

操作系统安全等级

- 可信计算机系统安全评价准则（TCSEC）标准是计算机系统安全评估的第一个正式标准，将操作系统的安全性划分为A、B、C、D四个等级。
- D类安全等级：D类安全等级只包括D1一个级别。
- C类安全等级：该类安全等级能够提供审慎的保护，并为用户的行动和责任提供审计能力。
- B类安全等级：B类安全等级可分为B1、B2和B3三类。B类系统具有强制性保护功能。
- A类安全等级：A系统的安全级别最高。

第9章 访问控制技术

计算机信息系统中，对信息的安全控制技术有3种：

- 访问控制：访问矩阵模型
- 信息流控制：格式模型（BLP模型、Biba模型、军用安全模型）
- 推理控制：解决数据库应用系统中的信息泄漏问题

第9章 访问控制技术

- 要保证计算机系统实体的安全，必须对计算机系统的访问进行控制
- 访问控制的基本任务
  - 防止非法用户即未授权用户进入系统
  - 合法用户即授权用户对系统资源的非法使用

第9章 访问控制技术

- **主体：**信息系统中用户或进程，系统所有的用户与进程形成主体集合，

- **客体：**系统中被处理、被控制或被访问的对象（如文件、程序、存储器等）

- **访问控制关系：**根据制定的系统安全策略，形成了主体与客体、主体与主体、客体与客体相互间的关系。

关系

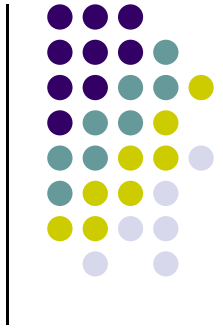
- 有的属于访问性质的
- 有的属于信息的流动问题

- 要实现信息系统安全的目的，需要解决系统中的访问控制问题和信息流控制问题。

基本任务与实现方法

- 基本任务：是保证对客体的所有直接访问都是被认可的。
- 实现：由支持安全策略的执行机制实现

访问控制的有效性



建立在两个前提上

- 用户鉴别与确证
- 信息受保护，不会被非法修改

9.1 访问控制技术概述

- 访问控制是从计算机系统的处理能力方面对信息提供保护
- 它按照事先确定的规则决定主体对客体的访问是否合法
- 当一主体试图非法使用一个未经授权的资源时，访问控制机制将拒绝这一企图，并将这一事件报告给审计跟踪系统
- 审计跟踪系统将给出报警，并记入日志档案

9.1 访问控制技术概述

- 网络的访问主要采用基于争用和定时两种方法
- 基于争用的方法意味着网上所有站点按先来先服务原则争用带宽
- 对网络的访问控制是为了防止非法用户进入系统和合法用户对系统的非法使用
- 访问控制要对访问的申请、批准和撤消的全过程进行有效的控制

9.1 访问控制技术概述

- 访问控制的内容包括
  - 用户身份的识别和认证
  - 对访问的控制
    - 授权 、确定访问权限 、实施访问权限
    - 附加控制
      - 除了对直接的访问进行控制外，还应对信息的流动和推理攻击施加控制
  - 审计跟踪
    - 对用户使用何种系统资源、使用的时间、执行的操作等问题进行完整的记录，以备非法事件发生后能进行有效的追查

9.1 访问控制技术概述

- 访问控制的类型
  - 自主访问控制（DAC）
    - 用户可以按自己的意愿对系统参数做适当的修改，可以决定哪个用户可以访问系统资源
  - 强制访问控制（MAC）
    - 用户和资源都是一个固定的安全属性，系统利用安全属性来决定一个用户是否可以访问某个资源
    - 由于强制访问控制的安全属性是固定的，因此用户或用户程序不能修改安全属性

9.1 自主访问控制

- 自主访问控制
  - 由客体自主地来确定各个主体对它的直接访问权限（又称访问模式）
  - 在自主访问控制下，用户可以按自己的意愿对系统的参数做适当的修改，以决定哪个用户可以访问他们的文件

9.1 自主访问控制

- 自主访问控制
  - Discretionary Access Control,简称DAC
  - 自主访问控制基于对主体或主体所属的主体组的识别来限制对客体的访问，这种控制是自主的
- 自主
  - 是指对其它具有授予某种访问权力的主体能够自主地（可能是间接的）将访问权的某个子集授予其它主体

- 对于通用型商业操作系统，DAC是一种最普遍采用的访问控制手段。
- 包括对文件、目录、通信信道以及设备的访问控制。
- 较完备的和友好的DAC接口，包括对邮箱、消息、I/O设备等客体提供自主访问控制保护。

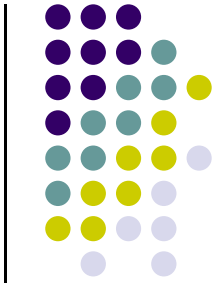
9.1 自主访问控制

- 访问控制矩阵

	O1		Oj		On
S1	**A11**		**A1j**		A1n
. . .	. . .	. . .	. . .	. . .	. . .
Si	**Ai1**		**Aij**		Ain
. . .	. . .	. . .	. . .	. . .	. . .
Sm	Am1		Amj		Amn

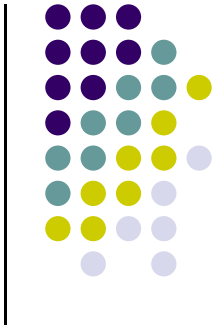
9.1 自主访问控制

- DAC的实现方法
  - 基于行的DAC
    - 权力表（Capabilities List）
    - 前缀表（Profiles）
    - 口令（Password）
  - 基于列的DAC
    - 保护位（Protection Bits）
    - 访问控制表（Access Control List，ACL）



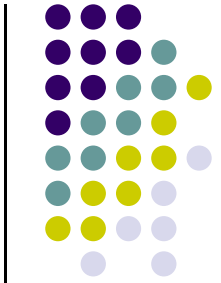
1、权限表机制

- 权限表中存放着主体可访问的每个客体的权限（如读、写、执行等），主体只能按赋予的权限访问客体。
- 由于允许主体把自己的权利转授给其他进程，或从其他进程收回访问权，权限表机制是动态实现的 。



- 利用访问权限表不能实现完备的自主访问控制。
- 实际利用权限表实现自主访问控制的系统并不多。

2、前缀表（profiles）机制



- 前缀表中存放着主体可访问的每个客体的名字和访问权。当主体要访问某个客体时，系统将检查该主体的前缀中是否具有它所请求的访问权。前缀表机制的实现存在以下困难需要解决：

前缀表机制的实现存在以下困难需要解决：

- 主体的前缀表可能很大，增加了系统管理的困难。
- 只能由系统管理员进行修改。这种管理方法有些超出了DAC原则。
- 修改与删除困难。要系统回答“谁对某一客体具有访问权”这样的问题比较困难。但这个问题在安全系统中却是很重要的。

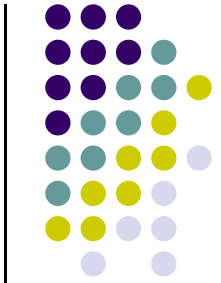
3、口令（password）机制

- 每个客体相应地有一个口令。当主体请求访问一个客体时，必须向系统提供该客体的口令。
- 请注意，这里讲的口令与用户登录进入系统时回答的口令不是一回事。
- 为了安全性起见，一个客体至少要有两个口令，一个用于控制读，一个用于控制写。
- 利用口令机制对客体实施的访问控制是比较麻烦的和脆弱的

口令机制的缺陷

- 1、系统不知谁访问了客体。对客体访问的口令是手工分发的，不需要系统参与
- 2、安全性脆弱。需要把该客体的口令写在程序中，这样很容易造成口令的泄露。
- 3、使用不方便。每个用户需要记忆许多需要访问的客体的口令，很不友好。
- 4、管理麻烦。撤消某用户对某客体的访问权，只能改变该客体的口令，必须通知新口令给其他用户。

基于行的DAC



- 以上三种方式，都是着眼于某一个主体的访问权限，以主体为出发点描述控制信息，因此很难回答对于某一个客体而言，有哪些对它拥有访问的权限。

二、 基于列的访问控制机制

- 这种机制是把每个客体被所在列上的有关主体（即非空矩阵元素所对应的那些行上的主体）访问的控制信息以表的形式附加给该客体，然后依此进行访问控制。它有两种实现形式：保护位方式和访问控制表（ACL）方式，

1、保护位机制

- 保护位对所有主体、主体组以及该客体的拥有者指定了一个访问权限的集合，UNIX中利用了这种机制。
- 在保护位中包含了主体组的名字和拥有者的名字。保护位机制中不包含可访问该客体的各个主体的名字
- 由于保护位的长度有限，用这种机制完全表示访问矩阵实际上是不可能的。

二、访问控制表（ACL）机制

- 在这种机制中，每个客体附带了访问矩阵中可访问它自己的所有主体的访问权限信息表（即ACL表）。该表中的每一项包括主体的身份和对该客体的访问权。
- 如果利用组或通配符的概念，可以使ACL表缩短。
- ACL方式是实现DAC策略的最好方法。

图 **ACL**表的一般结构

客体i	id_1. RW	id_2. RE	id_3. R	……	id_n. E

解决ACL表的长度问题

- 解决的办法是设法缩短ACL表的长度，采用分组与通配符的方法有助于达到该目的。一般而言，一个单位内部工作内容相同的人需要涉及的客体大部分是相同的，把他们分在一个组内作为一个主体对待，可以显著减少系统中主体的数目。再利用通配符手段加快匹配速度，同时也能简化ACL表的内容。通配符用“*”表示，可以代表任意组名或主体标识符。

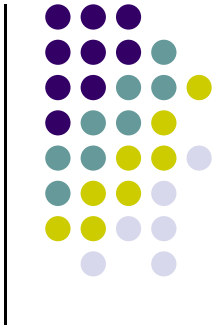
客体 FILE1				
	Liwen.math. REW	***.math.RE**	**zhang. * . R**	*** . * . null**

- 从该ACL表可以看出，属于math组的所有成员对客体FILE1都具有读与执行权；只有liwen 这个人对FILE1有读、写与执行的访问权限。任何组的用户zhang对FILE1只有读访问权，除此以外，对于其他任何组的任何主体对FILE1都没有任何访问权限。

三、 面向过程的访问控制

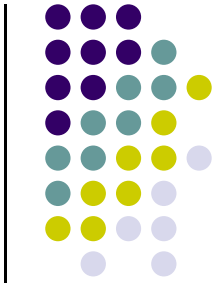
- 面向过程的访问控制是指在主体访问客体的过程中对主体的访问操作进行监视与限制。
- 该访问控制过程实际上是为被保护的客体建立一个保护层，它对外提供一个可信赖的接口，所有对客体的访问都必须通过这个接口才能完成。

- 面向对象技术与抽象数据类型都要求数据隐蔽功能，即数据隐藏在模块内部，这些数据中有的局部于模块内，外界永远不得访问；有的虽然允许外界访问，但必须通过模块接口才能完成。面向过程的保护机制可以实现这种信息隐蔽要求，但要付出执行效率的代价
- 有的系统中实现的保护子系统机制就是面向过程的访问控制的典型例子。



- 在子系统中的数据文件是受保护的对象，子系统中的过程是用来管理受保护对象的，并按用户要求实施对这些客体的访问控制。外部进程只能通过调用管理程序对子系统内部的客体进行访问操作。

- 可以利用多个保护子系统来完成某项作业，这就有可能出现这些子系统互相调用对方内部过程的情况。为了防止调用了不可信程序而对子系统内部的客体造成破坏，各个子系统内部都应该按互相猜疑策略进行防范。



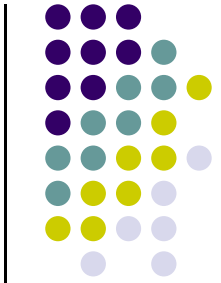
四、　访问许可权与访问操作权

- 在DAC策略下，访问许可（access permission）权和访问操作权是两个有区别的概念。
- 访问操作
- 访问许可

在DAC模式下，有3种控制许可权手段：

- 1、层次型的（hierarchical）文件的控制关系一般都呈树型的层次结构，系统管理员可修改所有文件的ACL表，文件主可以修改自己文件的ACL表。
- 层次型的优点是可以通过选择可信的人担任各级权限管理员，
- 缺点是一个客体可能会有多个主体对它具有控制权，发生问题后存在一个责任问题。

2、属主型的（owner）



- 该类型的访问权控制方式是为每一个客体设置拥有者，一般情况下客体的创建者就是该客体的拥有者。
- 拥有者拥有对自己客体的全部控制权，但无权将该控制权转授给其他主体。

属主型控制方式的安全性

- 属主型控制方式的优点是修改权限的责任明确。
- 如果主体（用户）被调离他处或死亡，系统需要利用某种特权机制来删除该主体拥有的客体。

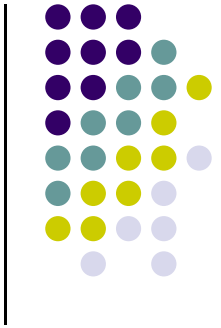
3、自由型的（laissez-faire）

- 客体的拥有者（创建者）可以把对自己客体的许可权转授给其他主体，并且也可以使其他主体拥有这种转授权，而且这种转授能力不受创建者自己的控制。
- 但由于这种许可权（修改权）可能会被转授给不可信的主体，因此这种对访问权修改的控制方式是很不安全的。

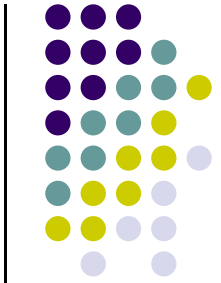
五、实现DAC的实例

- VAX/VMS曾经是非常典型的小型机操作系统，其中采用的支持DAC的文件系统的保护机制是一种很有效的文件安全的保护方法，在许多操作系统中广泛地应用。

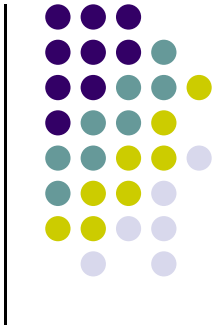
- VAX/VMS提供了两种基本的文件保护机制，一是基于用户识别码（UIC——User Identification Code）的标准保护机制，简称UIC保护机制；另一种是基于访问控制表ACL的保护机制。在VAX/VMS系统中，文件用户被划分为系统（system）类、拥有者（ownner）类、用户组（group）类和所有（world）类等四类，world类包括了前三类的用户。



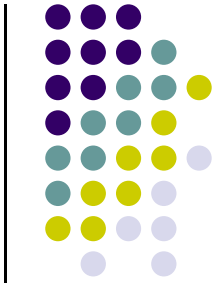
- 系统在用户的授权文件UAF（User Authorize File）中为每一个用户定义一个UIC，UIC由组号与成员号组成，其形式为[group, member]。对系统中的每一个客体也定义UIC和一个保护码，客体的UIC与其拥有者的UIC相同，保护码则表明允许哪些用户类对客体进行访问，以及进行何类访问。



- 下面是一个保护码的示例：
- SYSTEM:rwed ， OWNER:rwed ， GROUP:re，WORLD:e
- 其中r表示读，w表示写，e表示执行，d表示删除。



- 在VAX/VMS系统中按以下步骤控制用户对文件的每一次访问的：
- 1、首先检查文件是否带有访问控制表ACL，如果有，系统就按ACL表控制用户对该文件的访问。



- 2、如果ACL表中没有直接允许或拒绝该用户对该客体进行访问，那么系统就转而根据UIC机制来判决是否允许本次用户的访问。特别是如果ACL表直接拒绝了用户的访问请求，那么系统就仅根据UIC机制中的system与owner域来进一步判断是否允许用户的本次访问。

- 3、如果被访问的客体没有ACL表，系统就直接基于UIC的保护机制判决是否允许用户本次的访问。
- 4、对于拥有某些系统特权的用户可以不受ACL与UIC机制的限制而获得对客体的访问权。这些特权包括GRPPRV（组特权）、SYSPRV（系统特权）、READALL（读特权）以及BYPASS（全权）等特权。

9.1 自主访问控制

- DAC的优点
  - 方便、实用
  - 可由用户自由定制
  - 可扩展性强
- 缺点
  - 允许用户自主地转授访问权，这是系统不安全的隐患。
  - 系统无法区分是用户合法的修改还是木马程序的非法修改；
  - 无法防止木马程序利用共享客体或隐蔽信道传送信息。
  - 无法解决因用户无意（如程序错误、某些误操作等）或不负责任的操作而造成的敏感信息的泄漏问题。

9.1 强制访问控制

- 强制访问控制
  - Mandatory Access Control，简称MAC
  - 用户与文件都有一个固定的安全属性，系统利用安全属性来决定一个用户是否可以访问某个文件
  - 安全属性是强制性的，它是由安全管理员或操作系统根据限定的规则分配的，用户或用户的程序不能修改安全属性

9.1 强制访问控制

- 强制访问控制
  - 如果系统认为具有某一安全属性的用户不适于访问某个文件，那么任何人（包括文件的拥有者）都无法使该用户具有访问文件的能力
  - 强制访问控制是比任意访问控制更强的一种访问控制机制，它可以通过无法回避的访问限制来防止某些对系统的非法入侵
  - 强制访问控制可以防止一个进程生成共享文件，从而防止一个进程通过共享文件把信息从一个进程传送给另一个进程

一、 MAC机制的实现方法

- 最主要的是要做到两条：
- 第一是访问控制策略要符合MAC的原则，

把这些权利交给全系统权利最高和最受信任的安全管理员。

- 第二是对系统中的每一个主体与客体都要加安全标记，使它和主体或客体紧密相连而无法分开。

- 在MAC机制下，创建客体是受严格控制的，这样就可以阻止某个进程通过创建共享文件的方式向其他进程传递信息。
- 用户为某个目的运行的程序，由于他不能修改他自己及其他任何客体的安全属性，因此，即使用户程序中或系统中包含恶意程序（如特洛伊木马），也很难获取与用户程序无关的客体的敏感信息。
- 无法防范用户自己用非计算机手段将自己有权阅读的文件泄漏出去

- 在高安全级（B级及以上）的计算机系统中同时实现MAC机制与DAC机制，是在DAC机制的基础上增加更强的访问控制以达到强制访问控制的目的。

- 主体必须首先通过DAC和MAC的控制检查后，得到允许后方能访问某个客体。客体受到了双重保护，DAC可以防范未经允许的用户对客体的攻击，而MAC不允许随意修改主体、客体的安全属性，提供了一个不可逾越的保护层，因而又可以防范任意用户随意滥用DAC机制转授访问权。

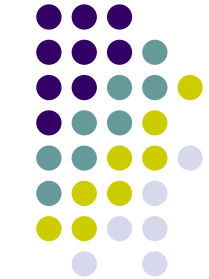
木马窃取敏感文件的方法

- 一是通过修改敏感文件的安全属性（如敏感级别、访问权等）来获取敏感信息。
- 另一种方法是躲在用户程序中的木马利用合法用户读敏感文件的机会，把所访问文件的内容拷贝到入侵者的临时目录下

- 强制访问控制机制比较适合专用目的的计算机系统，如军用计算机系统。因此从B1等级的计算机系统才开始实施这种机制，B2级计算机系统实现更强的MAC控制。

- 但对于通用型操作系统，从对用户友好性出发，一般还是以DAC机制为主，适当增加MAC控制。

二、 支持MAC的措施



- 1、防止恶意程序从外部进入系统。恶意程序从外部进入系统有两种渠道
  - 一是通过软盘、光盘或网络下载等方式，由用户自己“主动地”把未被认证是“纯净”的软件装入到系统中。
  - 二是利用系统存在的漏洞，通过网络攻击等手段把木马类程序装入系统。
- 用户对自己还要加强过程性控制，防止使用别人的木马程序或让木马进入自己的控制目录

- 2、消除利用系统自身的支持而产生木马的可能性。 在MAC机制下，由于系统中有很强的访问控制措施，外来的木马很难顺利工作与达到目的。但是，如果内部某个有不良意图的合法用户利用自己的权限在系统编程工具的支持下，编写藏有木马的程序，并使它在系统中合法的运行，这种情况下木马很难防范。

三、 实现MAC的一些实例

- 1、Multics方案
- Multics文件系统结构也是树型结构。每一个目录和文件都有一个安全级，每个用户也都有一个安全级。用户对文件的访问遵从以下强制访问控制安全策略：
- （1）仅当用户的安全级不低于文件的安全级时，用户才能读该文件；
- （2）仅当用户的安全级不高于文件的安全级时，用户才能写该文件。

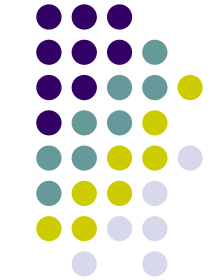
- 在unix系统中有一个专门的共享/TMP目录用于存放临时文件，为了能够让用户可以阅读/TMP目录下的文件，用户的安全级不能低于/TMP目录的安全级。但在Multics方案中，这种情况下是不允许用户在/TMP目录下创建与删除文件的。只有当用户的安全级与/TMP目录的安全级相同时，用户才能既在/TMP目录中阅读文件和进行创建与删除文件的操作。

2、Linus IV方案

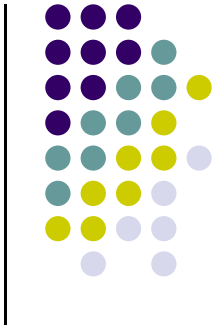
- Linus IV文件系统的访问控制方案基本与Multics的一样。

- 隔离目录（partitioned directory）

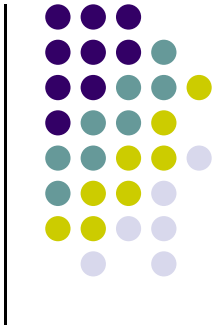
3、安全xenix方案



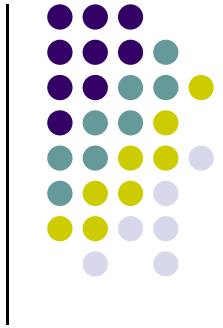
- 该方案中对文件系统的强制访问控制机制类似于Linus IV，它也支持隔离目录机制，此外，该目录还有一个特殊的通配安全级，该安全级与所有用户的安全级都相符。这种目录一般用于虚拟伪设备/dev/null这样的文件，这种文件对所有用户都是可访问的。



- 但在安全xenix方案中对写操作的控制更严格。它要求仅当用户的安全级与文件的安全级相同时，才允许该用户对该文件进行写操作。



- 当用户生成一个文件时，该文件的安全级就与用户的安全级相同。在生成一个目录时，目录的安全级按以下方式处理：所生成的目录名是按它的父目录的安全级分类的，但目录本身的安全级可以高于其父目录的安全级。如果一个目录的安全级高于其父目录，则称该目录为升级目录。



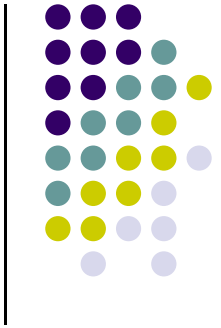
- 使用升级目录有点麻烦，一个用户若要使用升级目录，则需要先退出系统，然后再以升级目录的安全级重新注册进入系统。

4、Tim Thomas方案

- 该方案主要是为了解决安全xenix方案中升级目录所引起的使用不方便，即解决在已经登录进入系统的情况下，如要进入升级目录还要先退出系统，再重新用新安全级注册进入系统的问题。为了解决这种问题，该方案定义了一种新的目录类型。该方案的基本内容如下：

1）让文件名的安全级与文件内容的安全级相同

- 该方案与前面几个方案主要不同点是：文件名的安全级就代表了文件内容的安全级，在一个目录中可以有多个不同安全级别的文件存在，并且允许它们与目录有不同的安全级。



- 但是，对某安全级别的用户在目录中只能看到与自己级别相同或低于自己级别的文件。
- 对于用户所能够看见的文件就可以对其进行读、写、删除的操作，否则就不能。如果用户能够看见一个目录，那么他就可以在DAC机制的控制下创建一个新文件。

2）利用特殊接口实现文件名的隐蔽

- 由于允许一个目录下可包含多种安全级的文件，需要解决的一个问题是如何不让用户看见（即访问）比自己安全级高的文件名。

3）增加了对文件名的访问限制

- 文件的访问策略与安全xenix方案基本相同，但对文件名的访问增加了以下限制：
- （1）仅当用户的安全级不低于文件的安全级时，才能读该文件或文件名；
- （2）仅当用户的安全级与文件的安全级相同时，该用户才能对该文件进行写操作或者更改文件名。删除一个文件名被认为是对该文件的写操作。

9.5 强制访问控制

- Bell-La Padual模型
  - 简单安全规则
    - 仅当主体的敏感级不低于客体敏感级且主体的类别集合包含客体时，才允许该主体读该客体。即主体只能读密级等于或低于它的客体，也就是说主体只能从下读，而不能从上读
  - 星规则
    - 仅当主体的敏感级不高于客体敏感级且客体的类别集合包含主体的类别集合时，才允许该主体写该客体。即主体只能写密级等于或高于它的客体，也就是说主体只能向上写，而不能向下写

9.1 强制访问控制

- Biba模型
  - 简单完整规则
    - 仅当主体的完整级大于等于客体的完整级且主体的类别集合包含客体的类别集时，才允许该主体写该客体。即主体只能向下写，而不能向上写，也就是说主体只能写（修改）完整性级别等于或低于它的客体
  - 完整性制约规则（星规则）
    - 仅当主体的完整级不高于客体完整级且客体的类别集合包含主体的类别集合时，才允许该主体读读客体。即主体只能从上读，而不能从下读

9.2 新型访问控制技术

- 基于角色的访问控制技术
  - RBAC（ Role-Based Access Control）
  - NIST（National Institute of Standard Technology）
- 基于任务的访问控制技术
  - TBAC （Task-Based Access Control）
- 基于组机制的访问控制技术
- …

9.2 RBAC

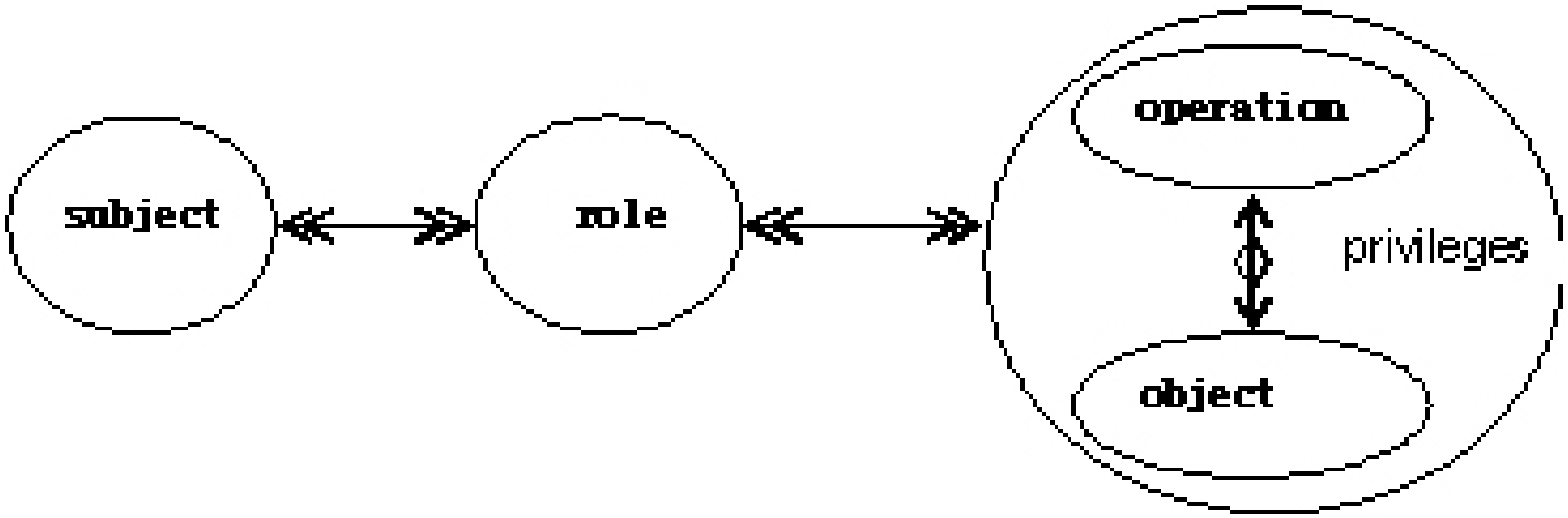


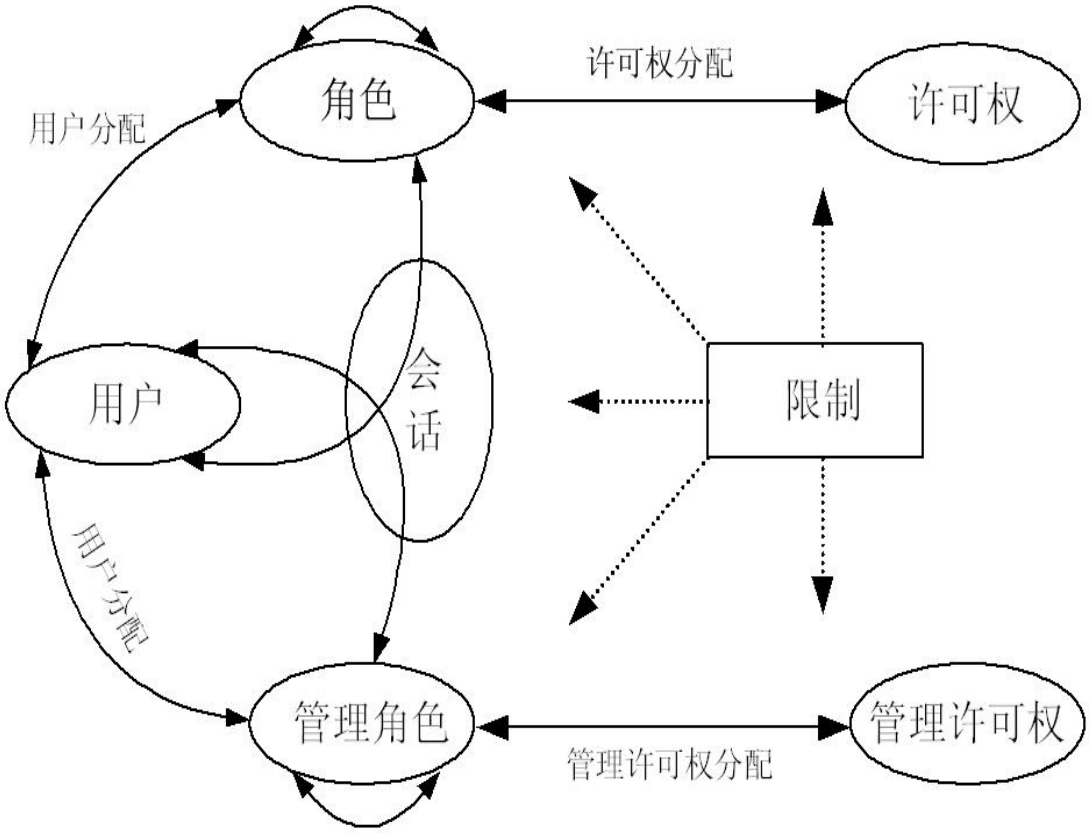
图4.2 RBAC的关系

9.2 新型访问控制技术

- 四种RBAC模型
  - 基本模型$RBAC_0$
  - 角色的层次结构$RBAC_1$
  - 约束模型$RBAC_2$
  - 混合模型$RBAC_3$

9.2 新型访问控制技术

- 基本模型$RBAC_0$
  - 四个基本要素
    - 用户（User）
    - 角色（Role）
    - 会话（Session）
    - 授权（ Permissio



9.2 新型访问控制技术

- 四种RBAC模型
  - 角色的层次结构$RBAC_1$
    - $RBAC_1$的特征是为$RBAC_0$上引入了角色层次的概念
  - 约束模型$RBAC_2$
    - $RBAC_2$除了继承$RBAC_0$的原有特征外，还引入了约束（Constraints）的概念
    - 互斥角色（ Mutually Exclusion Roles）
    - 基数约束（ Cardinality Constraints）
    - 先决条件角色
    - 运行时约束

9.2 新型访问控制技术

- RBAC模型的优点
  - 一种策略无关的访问控制技术
  - 具有自管理的能力
  - 使得安全管理更贴近应用领域的机构或组织的实际情况
- RBAC模型的不足
  - 复杂、不成熟
  - RBAC的策略无关性需要用户自己定义适合本领域的安全策略

9.3 入网认证

- 入网认证即入网访问控制。它为网络访问提供了第一层访问控制
- 入网认证控制哪些用户能够登录到服务器并获得网络资源，也控制准许用户入网的时间和准许他们在哪台工作站入网
- 入网认证实质上就是对用户的身份进行认证

9.3 入网认证

- 身份认证
  - 身份认证过程指的是当用户试图访问资源的时候，系统确定用户的身份是否真实的过程
  - 认证对所有需要安全的服务来说是至关重要的，因为认证是访问控制执行的前提，是判断用户是否有权访问信息的先决条件，同时也为日后追究责任提供不可抵赖的证据

9.3 入网认证

- 身份认证的依据
  - 用户所知道的
    - 密码
  - 用户所拥有的
    - 智能卡
  - 用户的特征
    - 生物学上的属性
  - 根据特定地点（或特定时间）
  - 通过信任的第三方
    - Kerberos , IKE

9.3入网认证

- 身份认证的评价标准
  - 可行性
  - 认证强度
  - 认证粒度
  - 认证数据正确
  - 不同协议间的适应性

9.3 入网认证

- 身份认证的评价标准
  - 可行性
    - 从用户的观点看，认证方法应该提高用户访问应用的效率，减少多余的交互认证过程，提供一次性认证
    - 另外所有用户可访问的资源应该提供友好的界面给用户访问

9.3 入网认证

- 身份认证的评价标准
  - 认证强度
    - 认证强度取决于采用的算法的复杂度以及密钥的长度
    - 采用更复杂的算法，更长的密钥，将能提高系统的认证强度，提高系统的安全性

9.3 入网认证

- 身份认证的评价标准
  - 认证粒度
    - 身份认证只决定是否允许用户进入服务应用。之后如何控制用户访问的内容，以及控制的粒度也是认证系统的重要标志
    - 有些认证系统仅限于判断用户是否具有合法身份，有些则按权限等级划分成几个密级，严格控制用户按照自己所属的密级访问

9.3 入网认证

- 身份认证的评价标准
  - 认证数据正确
    - 消息的接受者能够验证消息的合法性、真实性和完整性
    - 消息的发送者对所发的消息不可抵赖
    - 除了合法的消息发送者外，任何其他人不能伪造合法的消息
    - 当通信双方（或多方）发生争执时，有公正权威的第3方解决纠纷

9.3 入网认证

- 身份认证的评价标准
  - 不同协议间的适应性
    - 认证系统应该对所有协议的应用进行有效的身份识别
    - 除了HTTP以外，安全Email访问（包括认证SMTP、POP或者IMAP）也应该包含在认证系统中

9.3 入网认证

- 口令认证的一般过程
  - 用户名的识别与验证
    - 确定是否存在该用户的信息
  - 用户口令的识别与验证
    - 确定用户输入的口令是否正确
  - 用户帐号的缺省限制检查
    - 确定该用户帐号是否可用，以及能够进行哪些操作、访问哪些资源等用户的权限

9.3 入网认证

- 口令认证
  - 口令认证也称通行字认证，是一种根据已知事物验证身份的方法
- 通行字的选择原则
  - 易记
  - 难以被别人猜中或发现
  - 抗分析能力强
- 需要考虑的方面
  - 选择方法、使用期限、字符长度、分配和管理以及在计算机系统内的保护

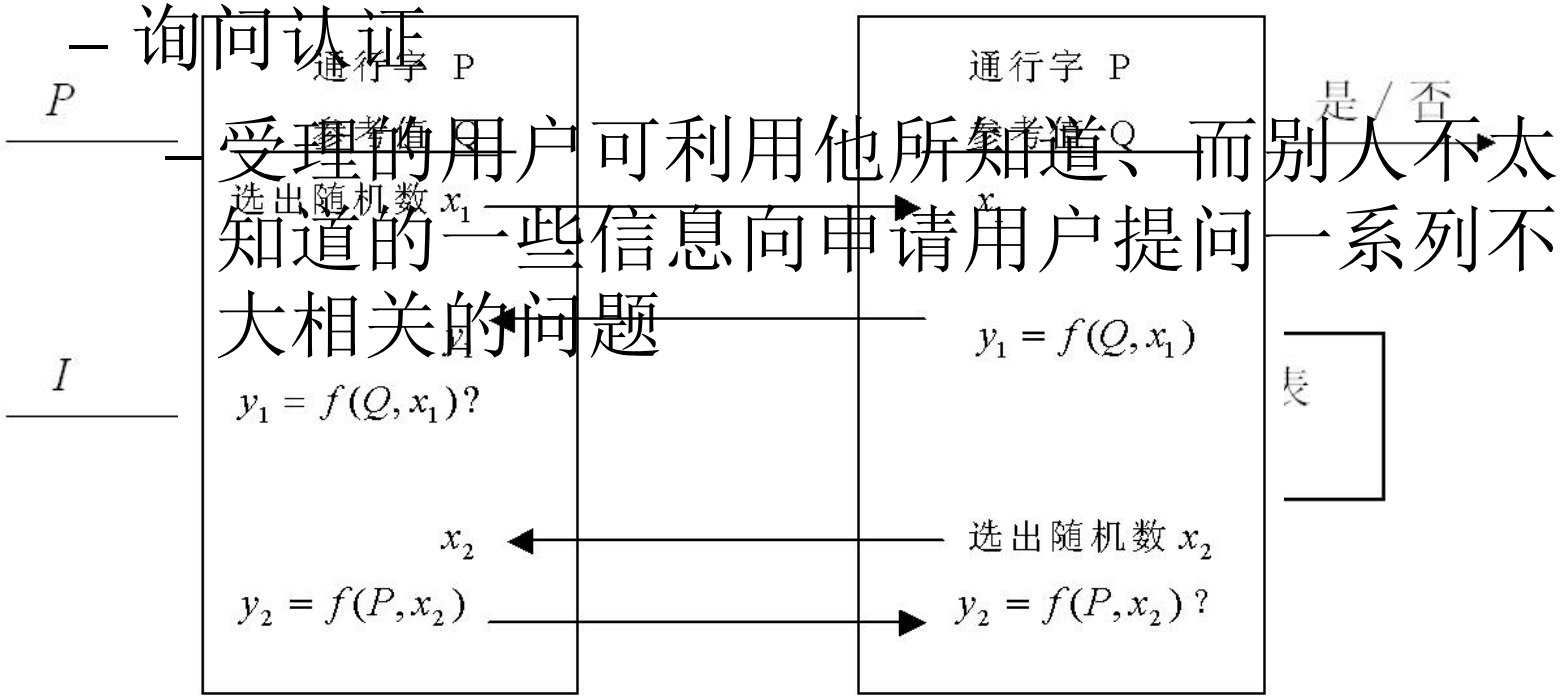
9.3 入网认证

安全性要求	口令认证方案
无	无口令
低	合法用户公用口令
中	每个用户一个单独的口令
高	要求一次一密，或口令分散

* 系统中不存储口令的原文

9.3 入网认证

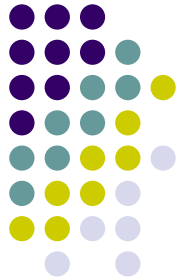
- 认证方式
  - 单向认证
  - 双向认证
    - 询问认证
      - 受理的用户可利用他所知道、而别人不太知道的一些信息向申请用户提问一系列不大相关的问题



一次性口令（OTP：One Time Password）

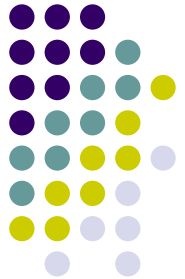
- 在认证口令中插入不确定的因素。不确定因素通常以明文方式传输，使得客户端和服务端都能够识别。用散列函数将口令和不确定因素的结合进行散列，其结果作为认证口令。

不确定因子选择方式



- 口令序列(S/KEY)
- 挑战/回答(CRYPTOCard)
- 时间同步(SecureID)
- 事件同步(Safe Word)

一次性口令的生成方式



- Token Card（硬件卡）
- Soft Token（软件）
- IC卡

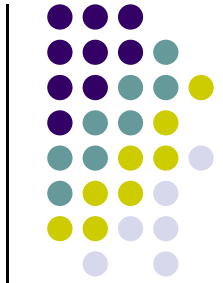
S/Key一次性口令认证系统

- RFC2289 . Alice->Victor
- 1. Alice向出示Victor自己的ID，请求认证。Victor发给Alice一个挑战。挑战（challenge）由一个种子（seed）值和一个迭代值（iteration）组成。
- 2. Alice根据挑战和自己口令生成一个通行证
- 3. Victor收到通行证后，把通行证再散列一次，其值同上次Alice成功登录的口令（或初始口令）比较
- 一定注意: 绝不要让您的计数器减少为0

S/Key协议安全分析

- 当Victor向Alice发出挑战时，Eve窃取下来，假如挑战中的迭代值为m，Eve将迭代值修改为一个较小的整数k，k<m，把修改后的挑战发给Alice。
- Alice根据修改的挑战计算出一个通行证OTP，向Victor提交验证。（Victor验证不通过。）
- Eve窃取出OPT，将OPT散列m-k次，生成通行证，用于冒充Alice进行验证。
- Victor通过验证。Eve冒充成功。
- Victor下一次冒充将窃取的OPT散列m-k-1次，生成通行证也能通过验证。这样可以冒充m-k次。

Kerberos协议



- 问题提出
- 协议

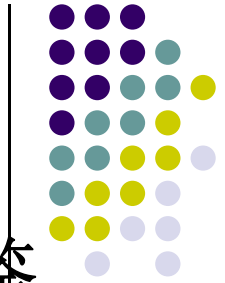
问题

- 在一个开放的分布式网络环境中，用户通过工作站访问服务器上提供的服务。
  - 服务器应能够限制非授权用户的访问并能够认证对服务的请求。
  - 工作站不能够被网络服务所信任其能够正确地认定用户，即工作站存在三种威胁。
    - 一个工作站上一个用户可能冒充另一个用户操作；
    - 一个用户可能改变一个工作站的网络地址，从而冒充另一台工作站工作；
    - 一个用户可能窃听他人的信息交换，并用回放攻击获得对一个服务器的访问权或中断服务器的运行。

Kerberos要解决的问题

- 所有上述问题可以归结为一个非授权用户能够获得其无权访问的服务或数据。
- 不是为每一个服务器构造一个身份认证协议，**Kerberos**提供一个中心认证服务器，提供用户到服务器和服务器到用户的认证服务。
- **Kerberos**采用传统加密算法（无公钥体制）。

Kerberos



- 是美国麻省理工学院（MIT）开发的一种身份鉴别服务。 **http://web.mit.edu/kerberos/**
- **“Kerberos”**的本意是希腊神话中守护地狱之门的守护者。
- **Kerberos**提供了一个集中式的认证服务器结构，认证服务器的功能是实现用户与其访问的服务器间的相互鉴别。
- **Kerberos**建立的是一个实现身份认证的框架结构。
- 其实现采用的是对称密钥加密技术，而未采用公开密钥加密。
- 公开发布的**Kerberos**版本包括版本**4**和版本**5 (RFC1510)**。

信息系统资源保护的动机

- 单用户单机系统。用户资源和文件受到物理上的安全保护；
- 多用户分时系统。操作系统提供基于用户标识的访问控制策略，并用**logon**过程来标识用户。
- **Client/Server**网络结构。由一组工作站和一组分布式或中心式服务器组成。

C/S环境下三种可能的安全方案

- 相信每一个单独的客户工作站可以保证对其用户的识别，并依赖于每一个服务器强制实施一个基于用户标识的安全策略。
- 要求客户端系统将它们自己向服务器作身份认证，但相信客户端系统负责对其用户的识别。
- 要求每一个用户对每一个服务证明其标识身份，同样要求服务器向客户端证明其标识身份。

Kerberos的解决方案

- **Kerberos**支持以上三种策略。
- 在一个分布式的**client/server**体系机构中采用一个或多个**Kerberos**服务器提供一个认证服务。
- 总体方案是提供一个可信第三方的认证服务。

Kerberos系统应满足的要求

- 安全。网络窃听者不能获得必要信息以假冒其它用户；**Kerberos**应足够强壮以至于潜在的敌人无法找到它的弱点连接。
- 可靠。**Kerberos**应高度可靠，并且应借助于一个分布式服务器体系结构，使得一个系统能够备份另一个系统。
- 透明。理想情况下，用户除了要求输入口令以外应感觉不到认证的发生。
- 可伸缩。系统应能够支持大数量的客户和服务器。

Kerberos设计思路

- 基本思路：
  - 使用一个（或一组）独立的认证服务器（**AS — Authentication Server**），来为网络中的用户（C）提供身份认证服务；
  - 认证服务器 **(AS)**，用户口令由 **AS** 保存在数据库中；
  - **AS** 与每个服务器（**V**）共享一个惟一保密密钥（K_v）（已被安全分发）。
- 会话过程：

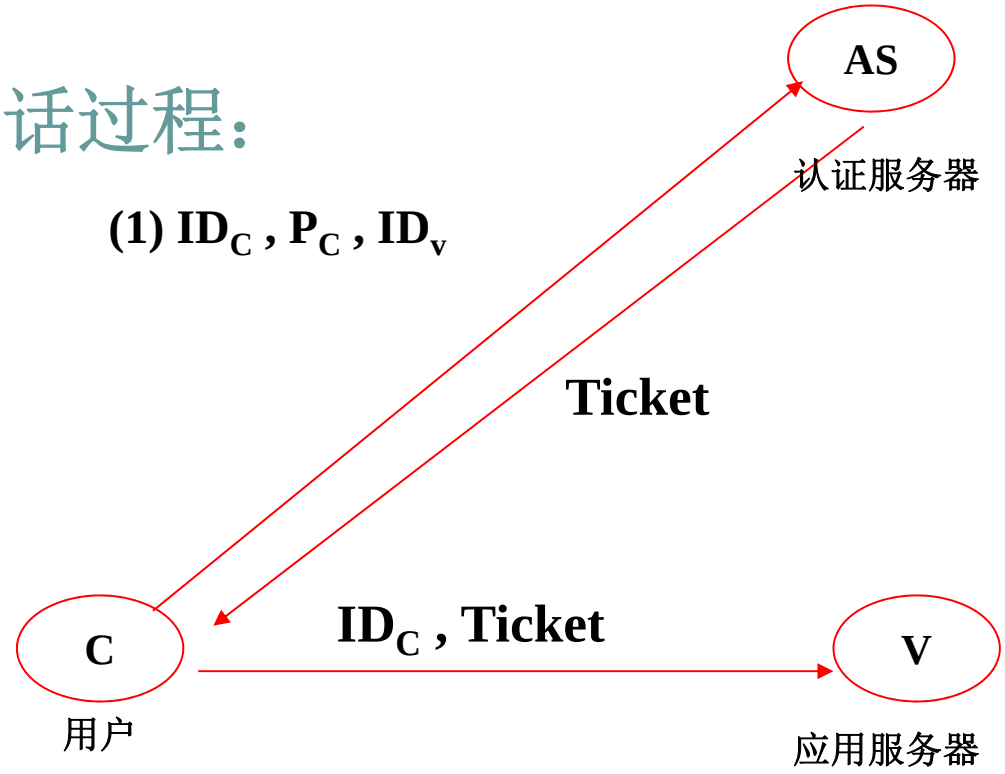
(1) C → AS: $ID_C \| P_C \| ID_v$
(2) AS → C: Ticket
(3) C → V : $ID_C \| Ticket$

- 其中：

$Ticket = E_{Kv}[ID_C \| AD_C \| ID_v]$

Kerberos设计思路（续）

- 会话过程：



搜索数据库看用户是否合法

如果合法，验证用户口令是否正确

如果口令正确，检查是否有权限访问服务器V

用与AS共享密钥解密票据

检查票据中的用户标识与网络地址是否与用户发送的标识及其地址相同

如果相同，票据有效，认证通过

$Ticket = E_{Kv}[ID_C , AD_C , ID_v]$

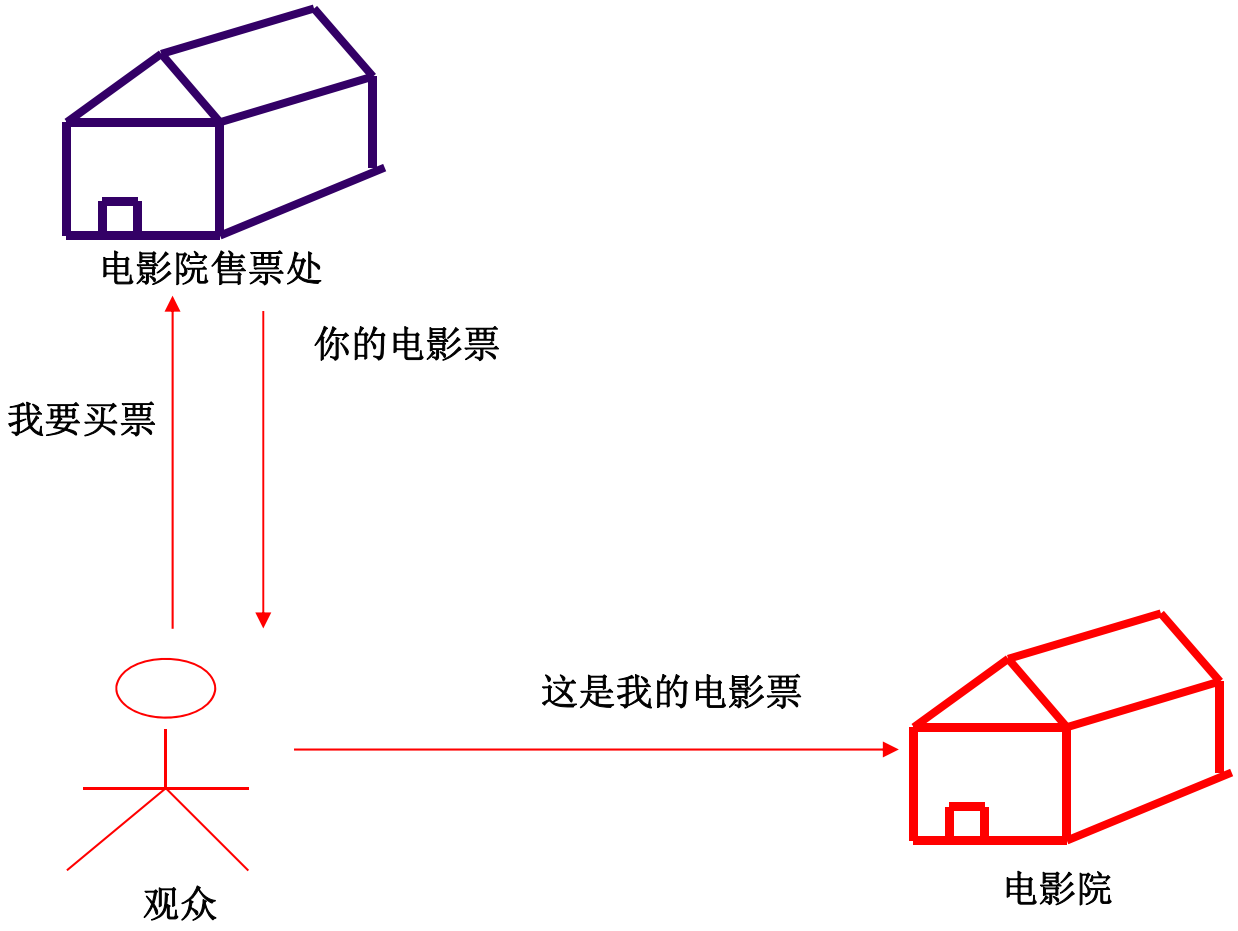
ID_C：用户C的标识

P_C ：用户口令

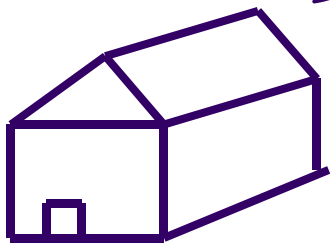
ID_v：服务器标识

AD_C：用户网络地址

Kerberos设计思路（续）



Kerberos设计思路（续）



电影院售票处

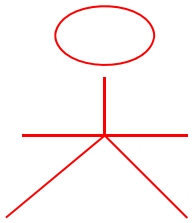
问题：如何买票

答案：出示信用卡卡号和密码

问题之一：信用卡问题

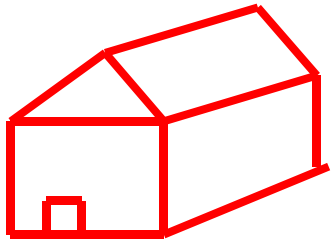
你的电影票

我要买票，
这是我的信
用卡密码



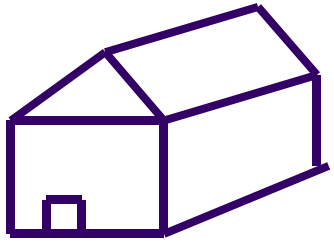
观众

这是我的电影票



电影院

Kerberos设计思路（续）

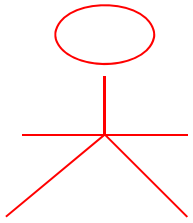


电影院售票处

问题之二：票的有效期问题

我要买票，
这是我的信
用卡密码

你的电影票

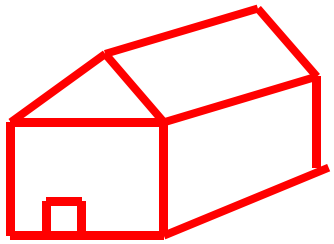


观众

这是我的电影票

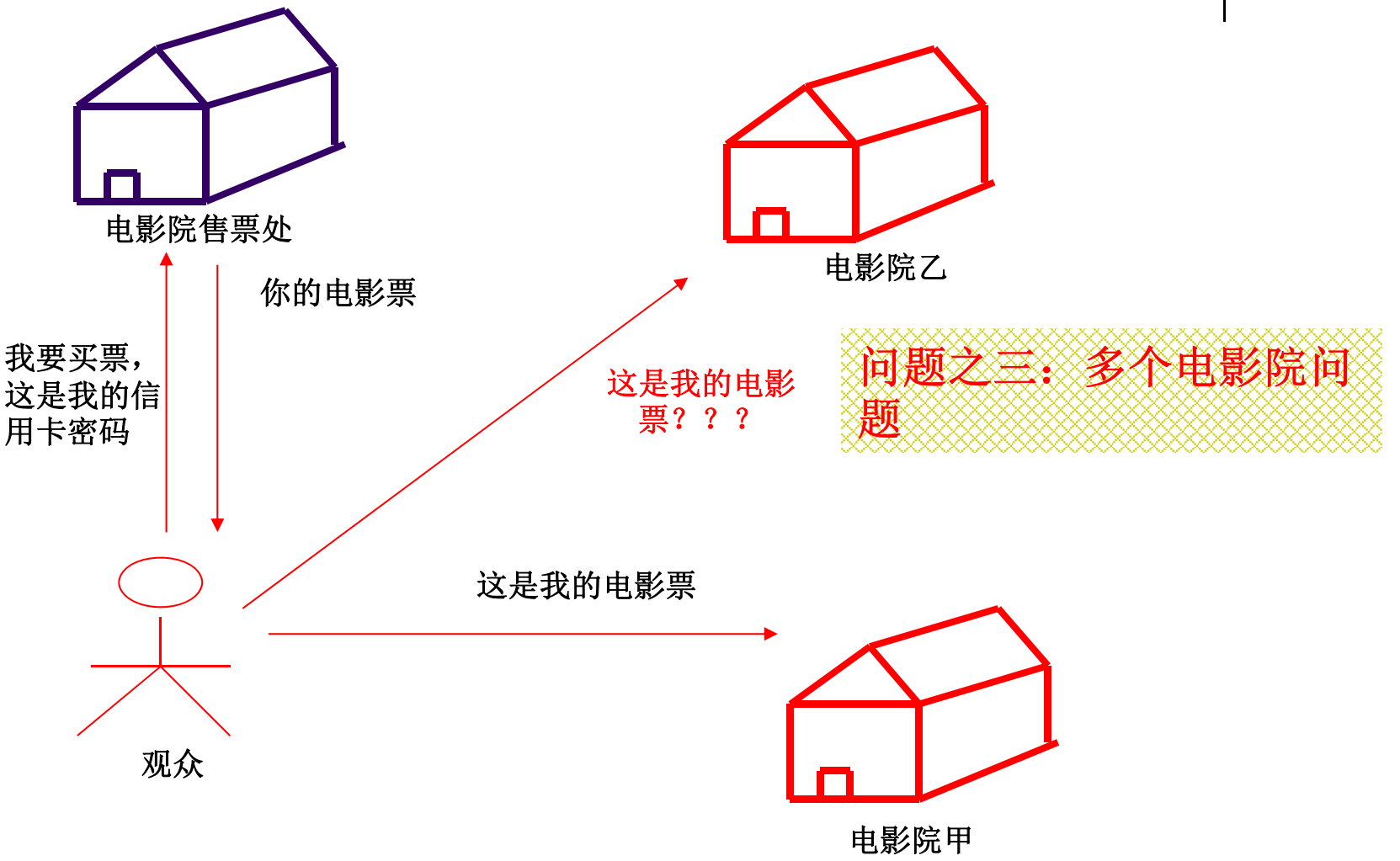
这是我的电影票

这是我的电影票



电影院

Kerberos设计思路（续）



Kerberos设计思路（续）

上述协议问题？

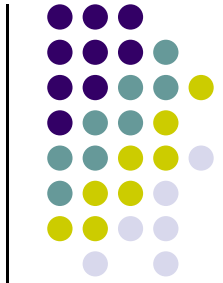
上述协议的问题：

（1）口令明文传送

（2）票据的有效性（多次使用）

（3）访问多个服务器则需多次申请票据（即口令多次使用）

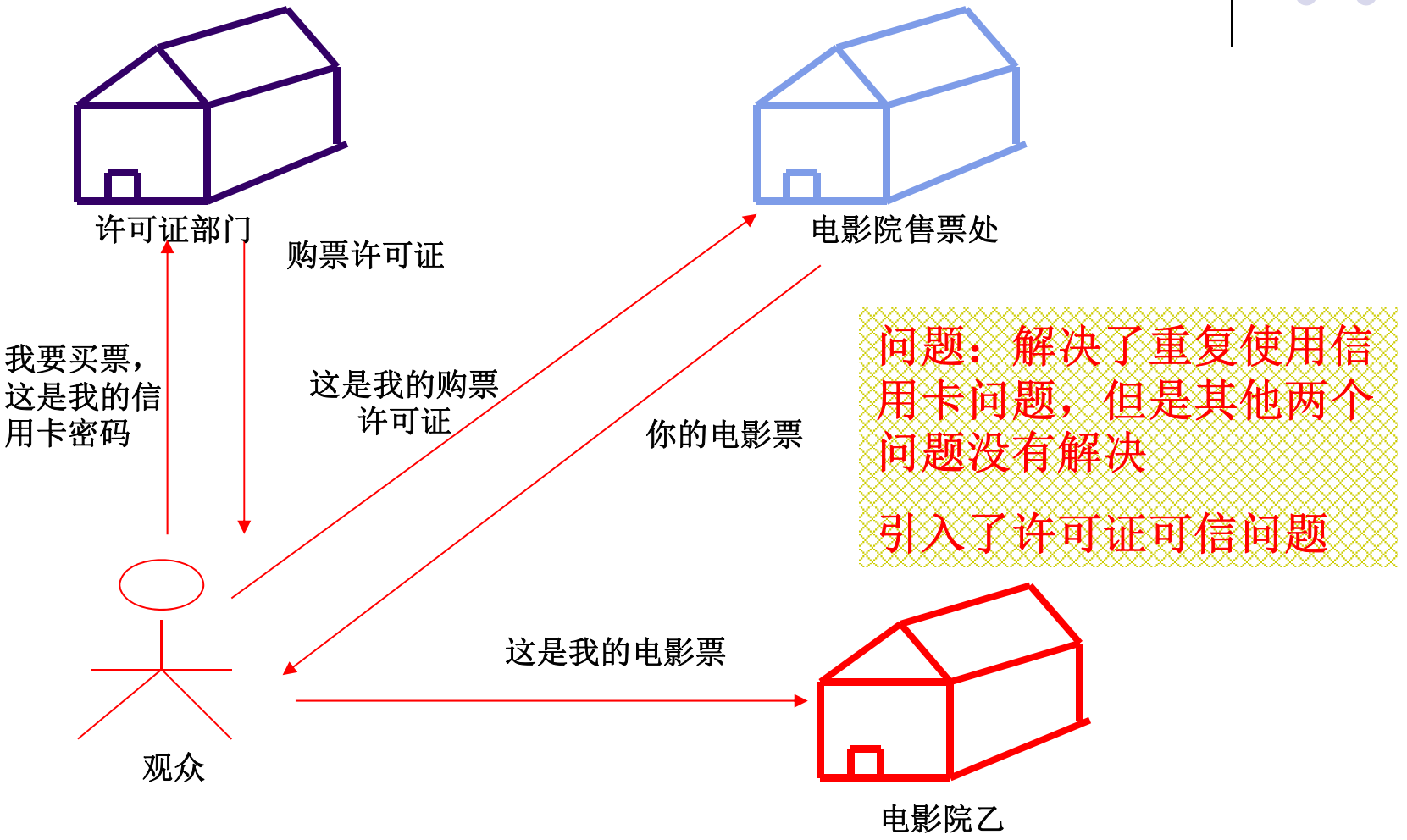
如何解决



Kerberos设计思路（续）

- 问题：
  - 用户希望输入口令的次数最少。
  - 口令以明文传送会被窃听。
- 解决办法
  - 票据重用（**ticket reusable**）。
  - 引入票据许可服务器（**TGS - ticket-granting server**）
    - 用于向用户分发服务器的访问票据；
    - 认证服务器 **AS** 并不直接向客户发放访问应用服务器的票据，而是由 **TGS** 服务器来向客户发放。

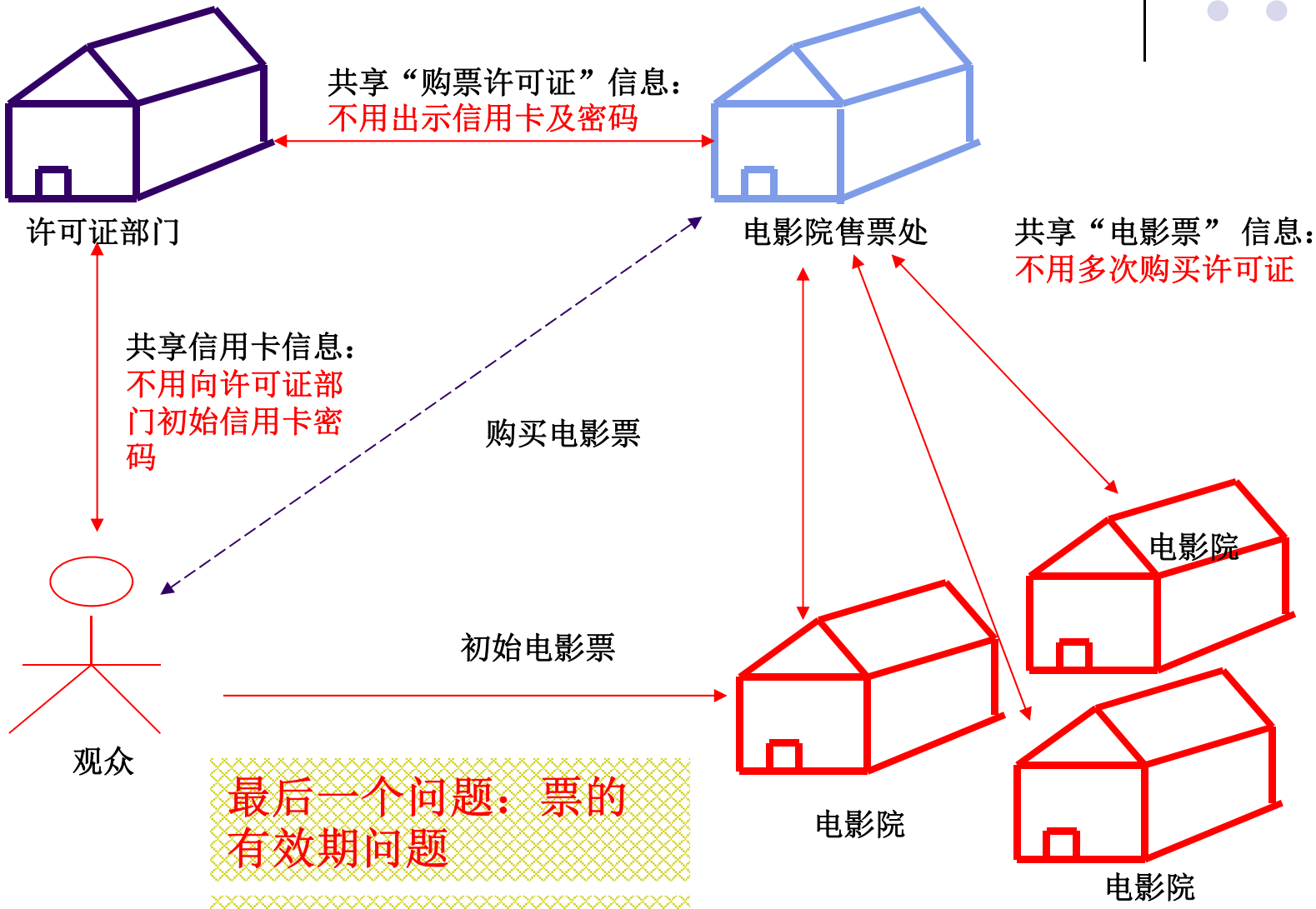
Kerberos设计思路（续）



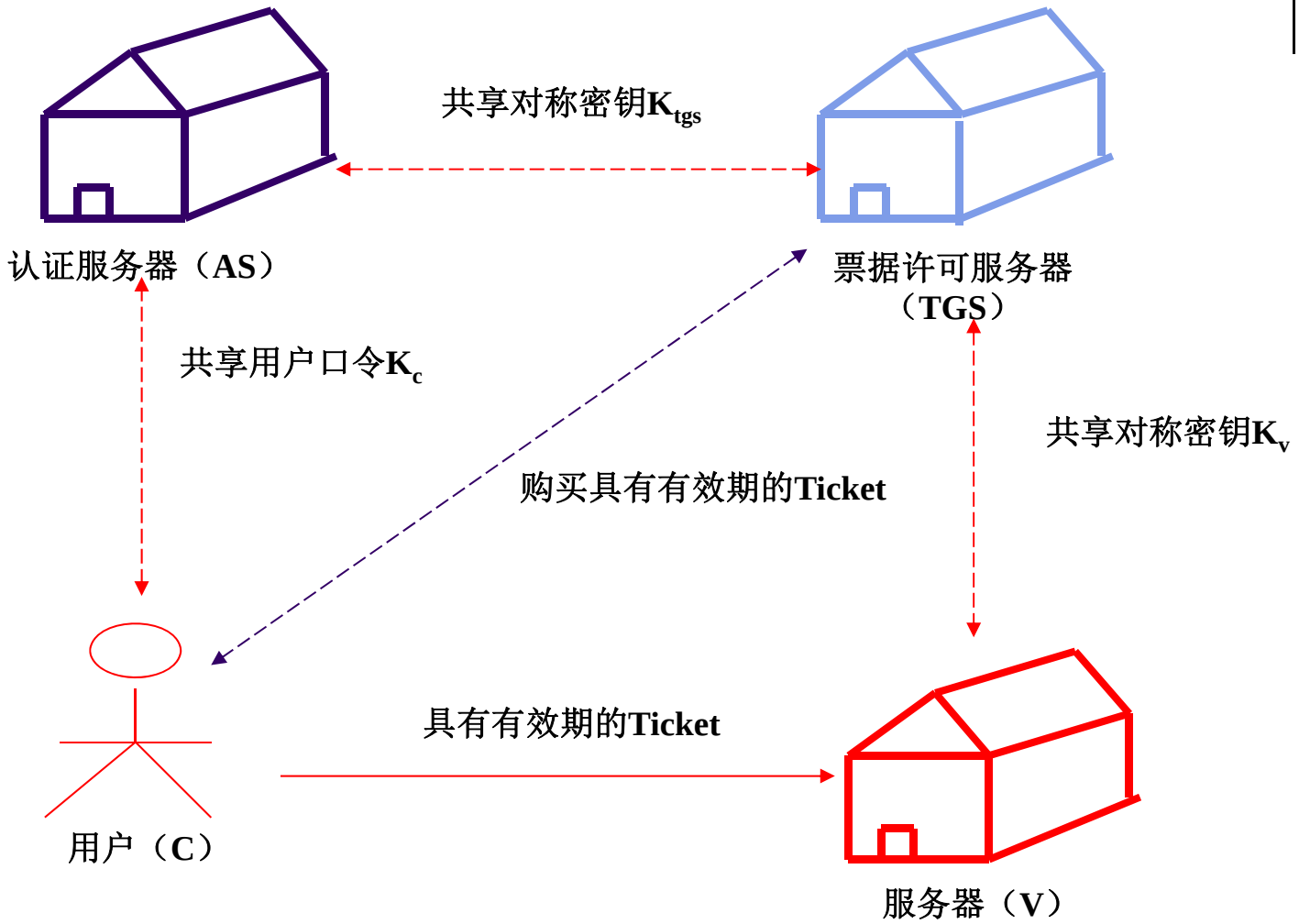
Kerberos的票据

- 两种票据
  - 票据许可票据（**Ticket granting ticket**）
    - 客户访问 **TGS** 服务器需要提供的票据，目的是为了申请某一个应用服务器的“服务许可票据”；
    - 票据许可票据由 **AS** 发放；
    - 用 $\mathbf{Ticket_{tgs}}$ 表示访问 **TGS** 服务器的票据；
    - $\mathbf{Ticket_{tgs}}$ 在用户登录时向 **AS** 申请一次，可多次重复使用；
  - 服务许可票据（**Service granting ticket**）
    - 是客户时需要提供的票据；
    - 用 **TicketV** 表示访问应用服务器 **V** 的票据。

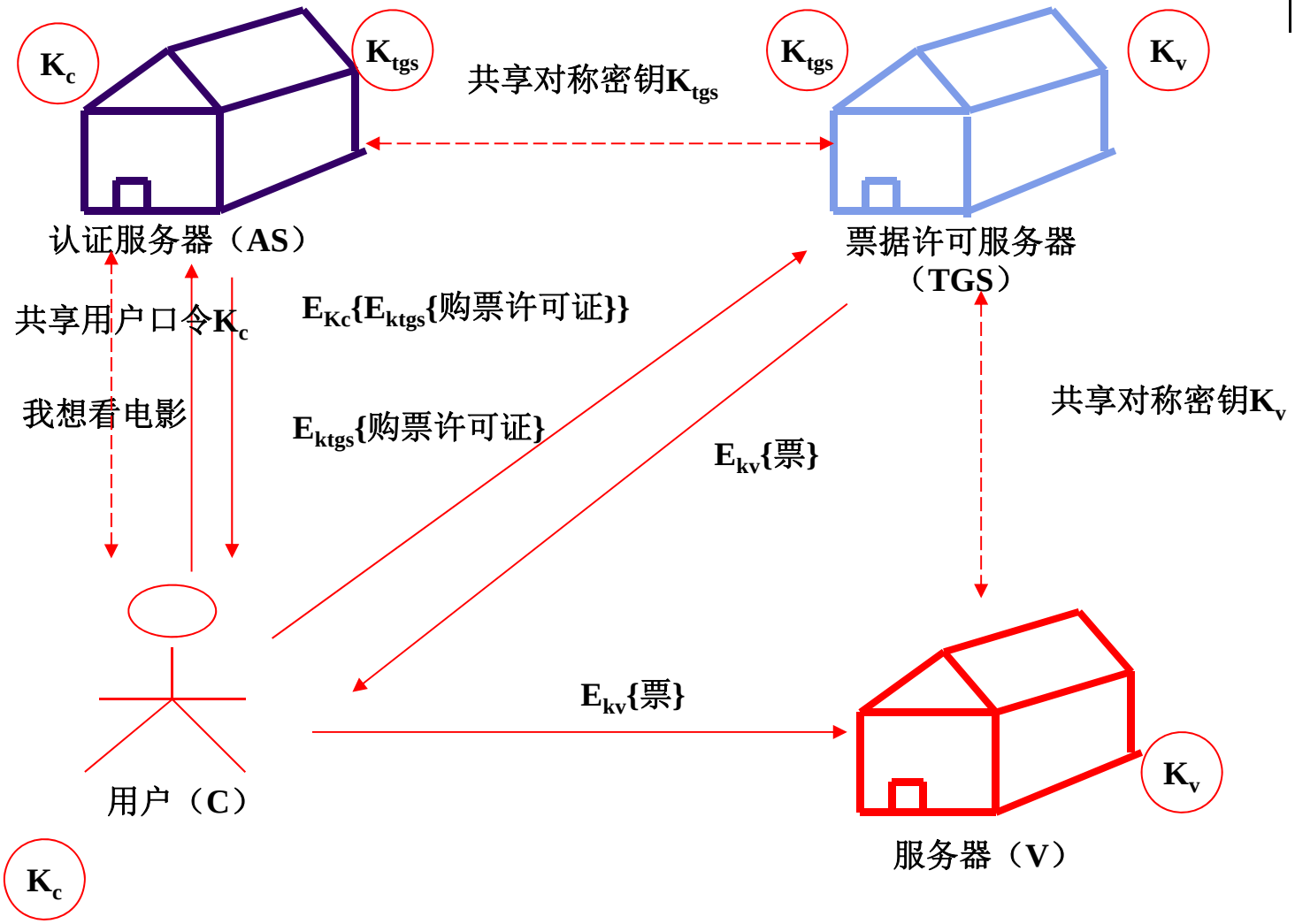
Kerberos设计思路（续）



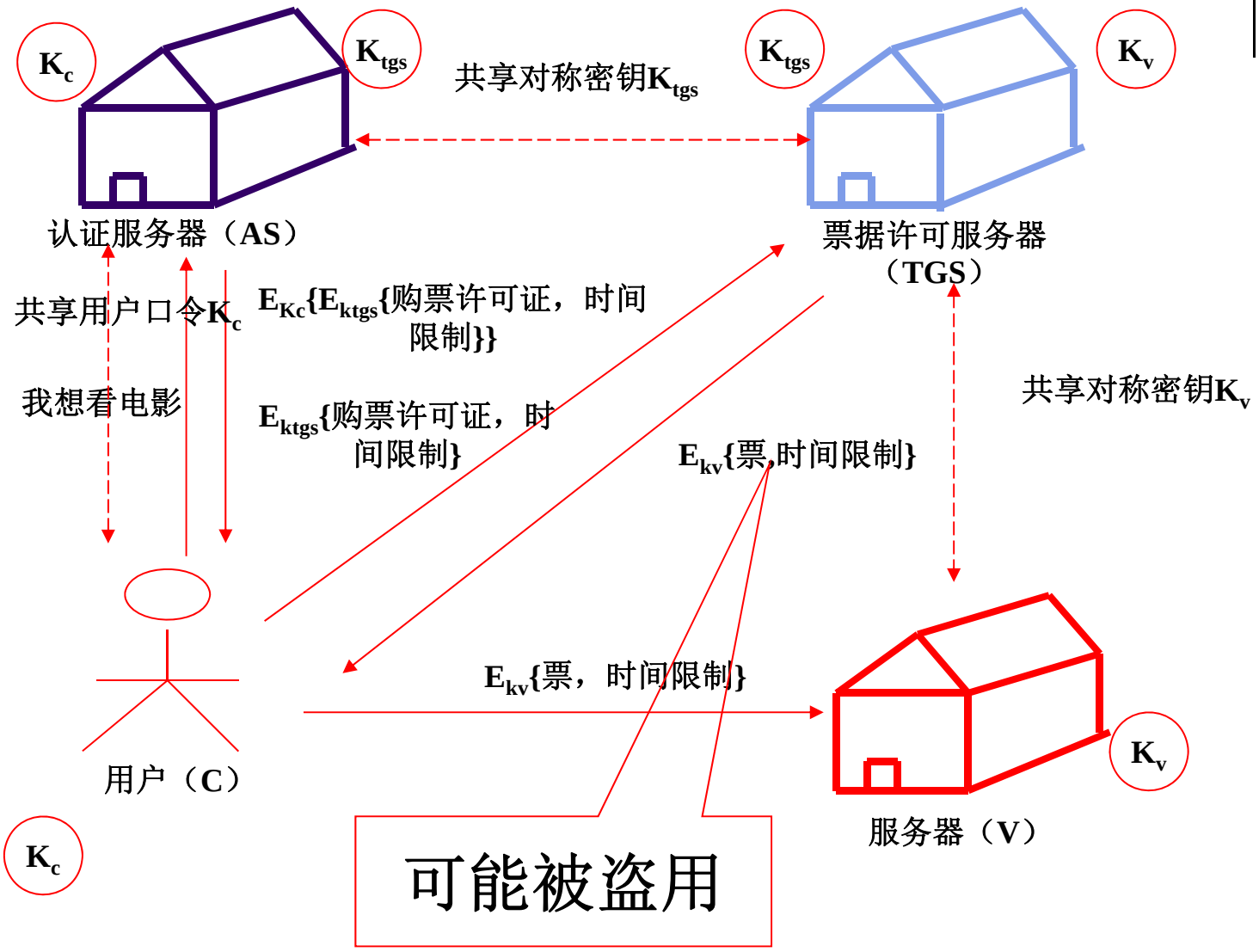
Kerberos设计思路（续）



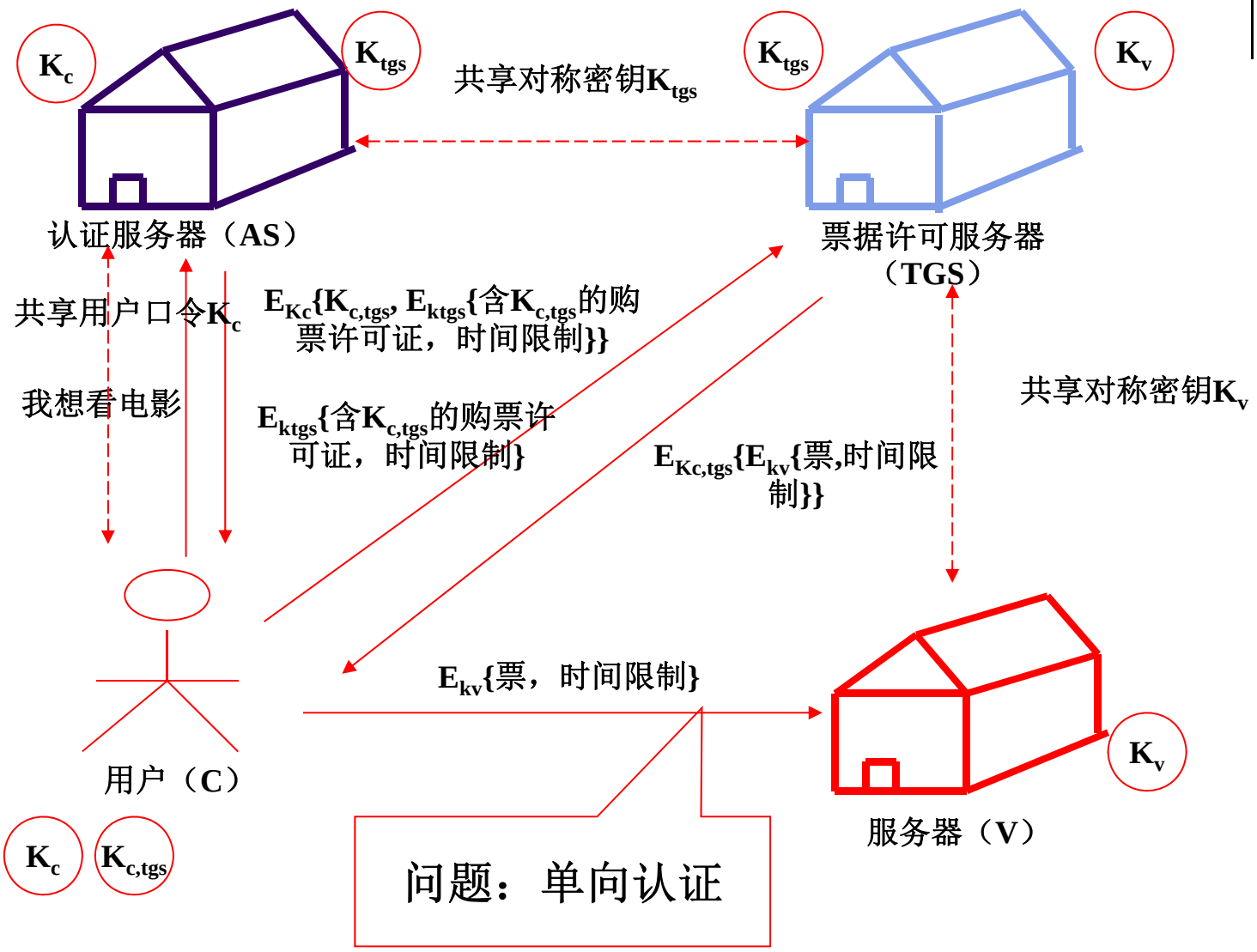
Kerberos设计思路（续）



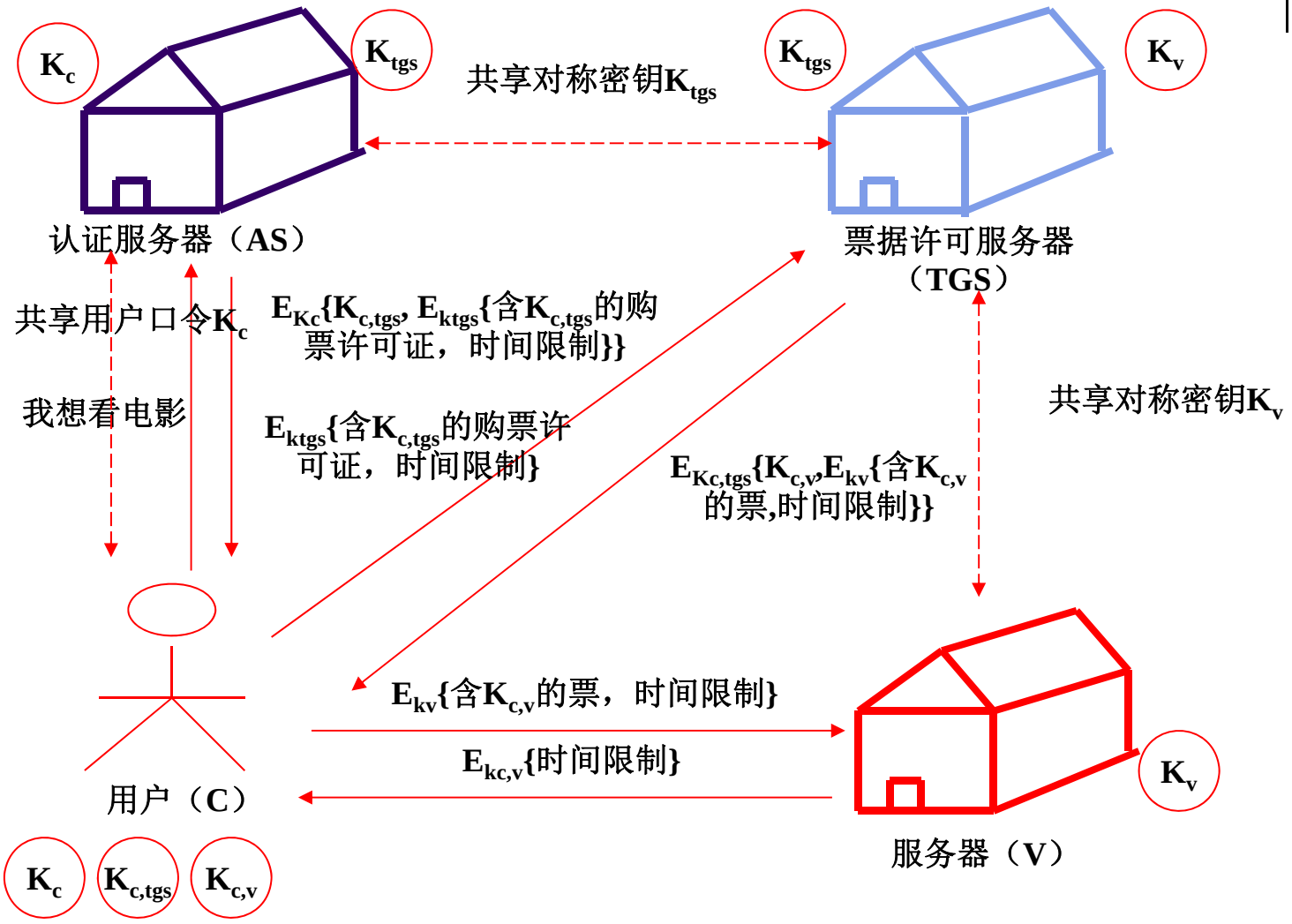
Kerberos设计思路（续）



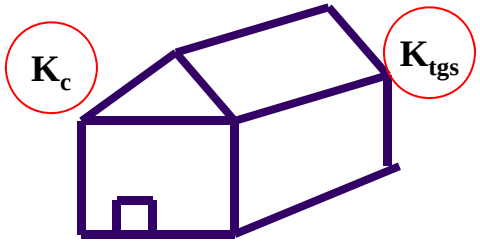
Kerberos设计思路（续）



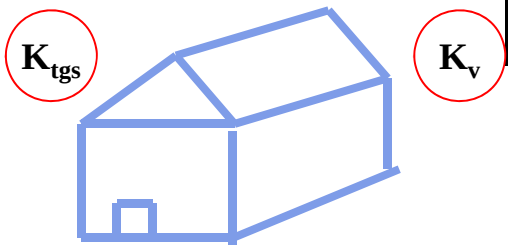
Kerberos设计思路（续）



Kerberos V4协议描述：第一阶段



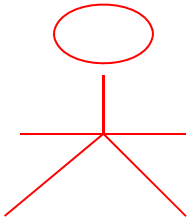
认证服务器（AS）



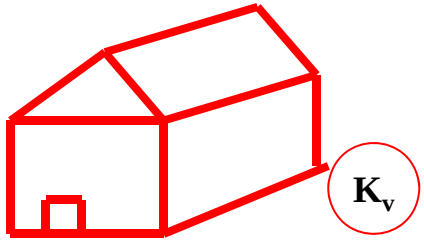
票据许可服务器（TGS）

$E_{Kc}\{K_{c,tgs}, ID_{tgs}, TS_2, LT_2, Ticket_{tgs}\}$

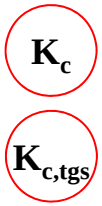
ID_C, ID_{tgs}, TS_1



用户（C）



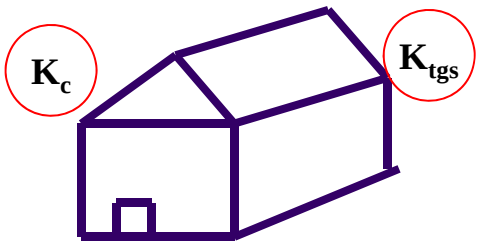
服务器（V）



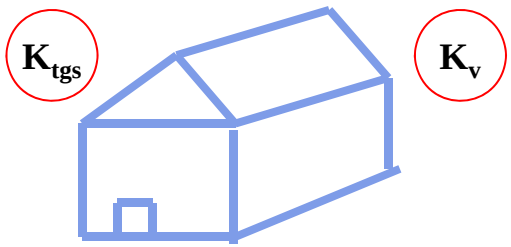
$Ticket_{tgs} = E_{Ktgs}\{K_{C,tgs}, ID_C, AD_C, ID_{tgs}, TS_2, LT_2\}$

AD_C防止何种攻击？

Kerberos V4协议描述：第二阶段



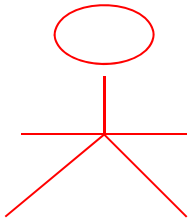
认证服务器（AS）



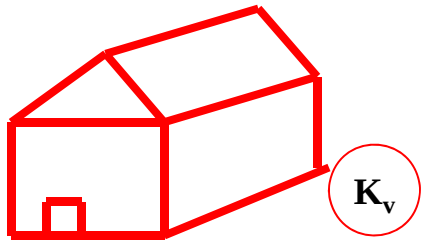
票据许可服务器（TGS）

$ID_V, Ticket_{tgs}, AU_C$

$E_{KC,tgs}\{K_{C,V}, ID_V, TS_4, Ticket_V\}$



用户（C）



服务器（V）

K_c

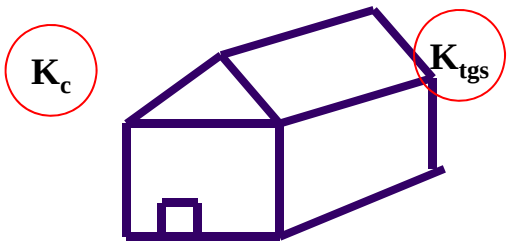
$Ticket_{tgs} = E_{Ktgs}\{K_{C,tgs}, ID_C, AD_C, ID_{tgs}, TS_2, LT_2\}$

$K_{c,tgs}$ $K_{c,v}$

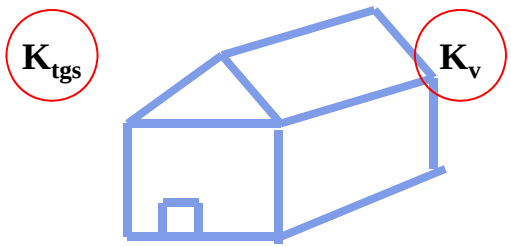
$AU_C = E_{KC,tgs}\{ID_C, AD_C, TS_3\}$

$Ticket_V = E_{KV}\{K_{C,V}, ID_C, AD_C, ID_V, TS_4, LT_4\}$

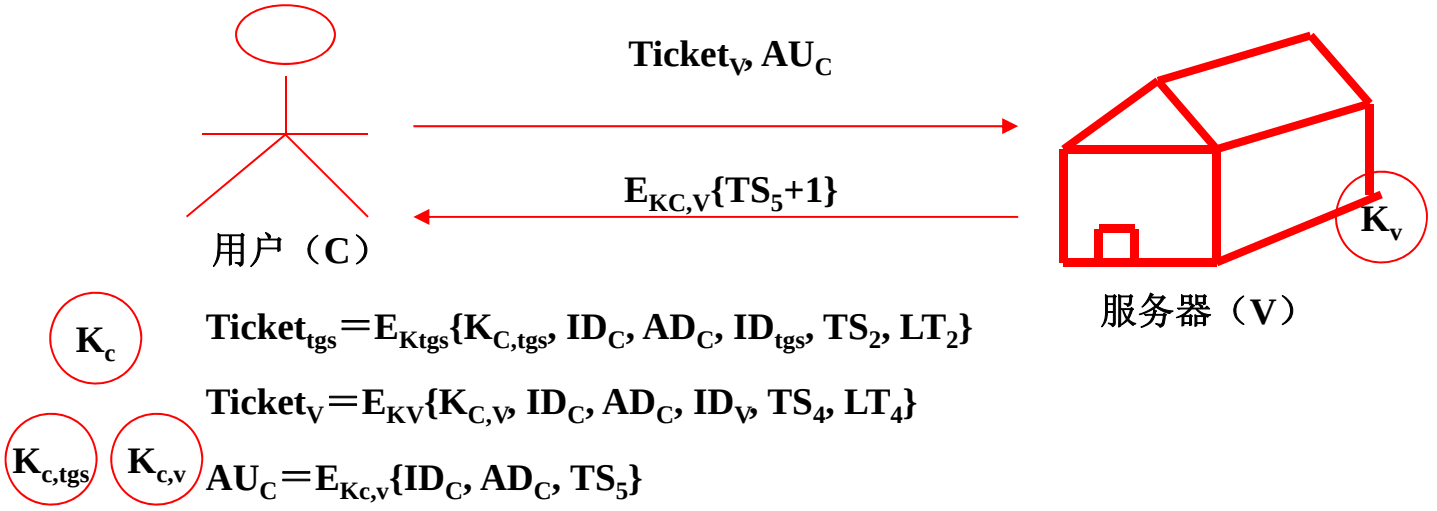
Kerberos V4协议描述：第三阶段



认证服务器（AS）



票据许可服务器（TGS）

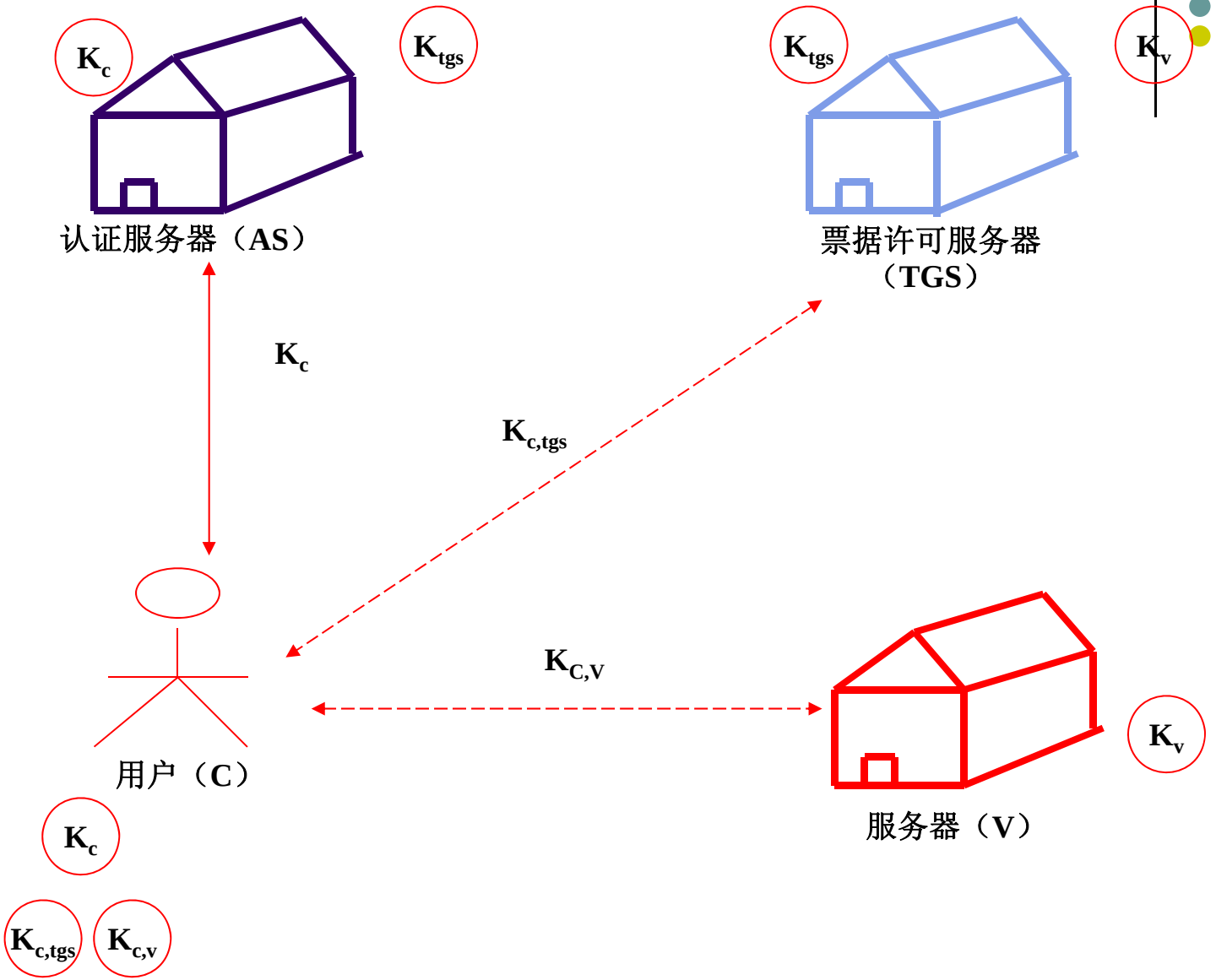


$Ticket_{tgs} = E_{Ktgs}\{K_{C,tgs}, ID_C, AD_C, ID_{tgs}, TS_2, LT_2\}$

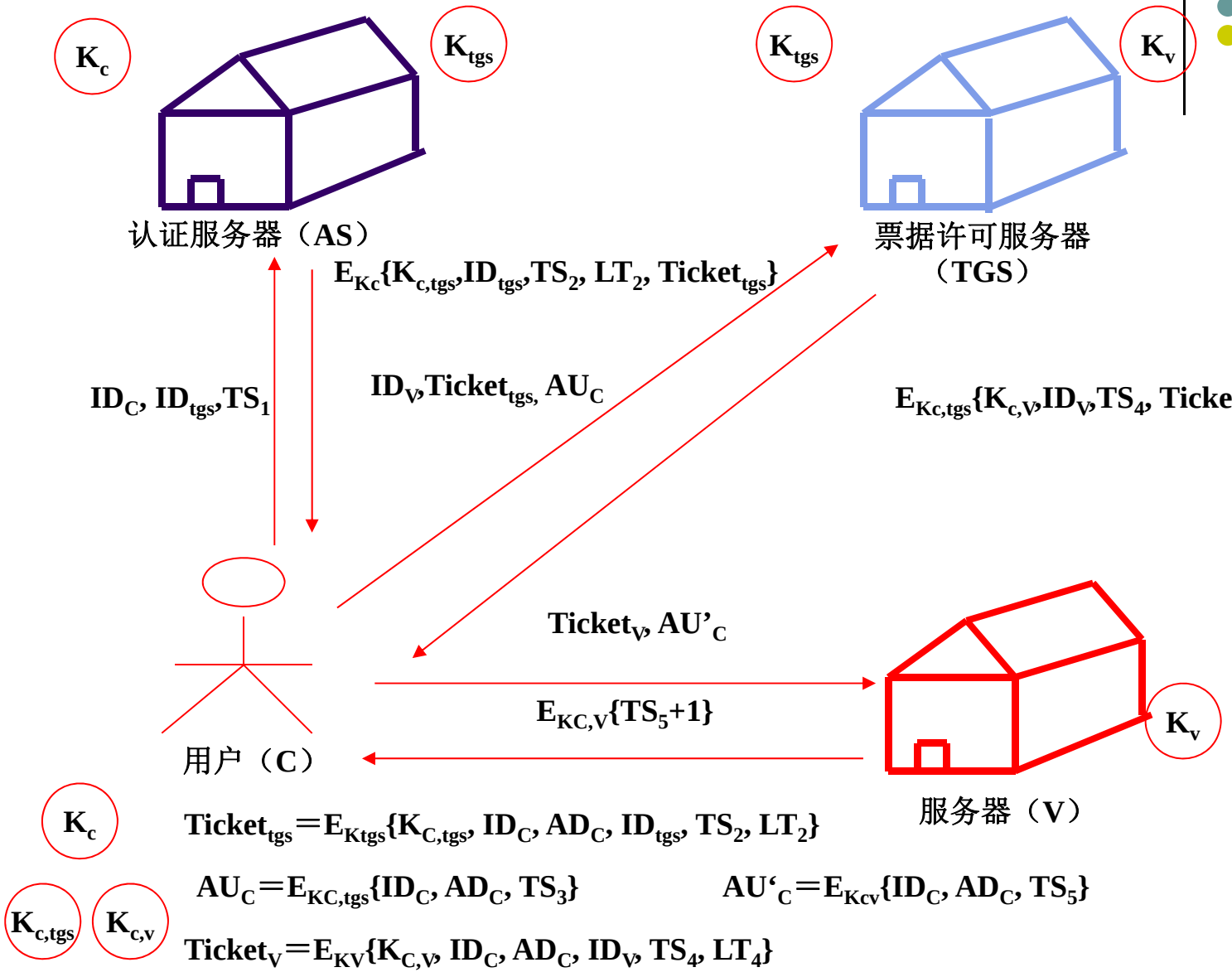
$Ticket_V = E_{KV}\{K_{C,V}, ID_C, AD_C, ID_V, TS_4, LT_4\}$

$AU_C = E_{Kc,v}\{ID_C, AD_C, TS_5\}$

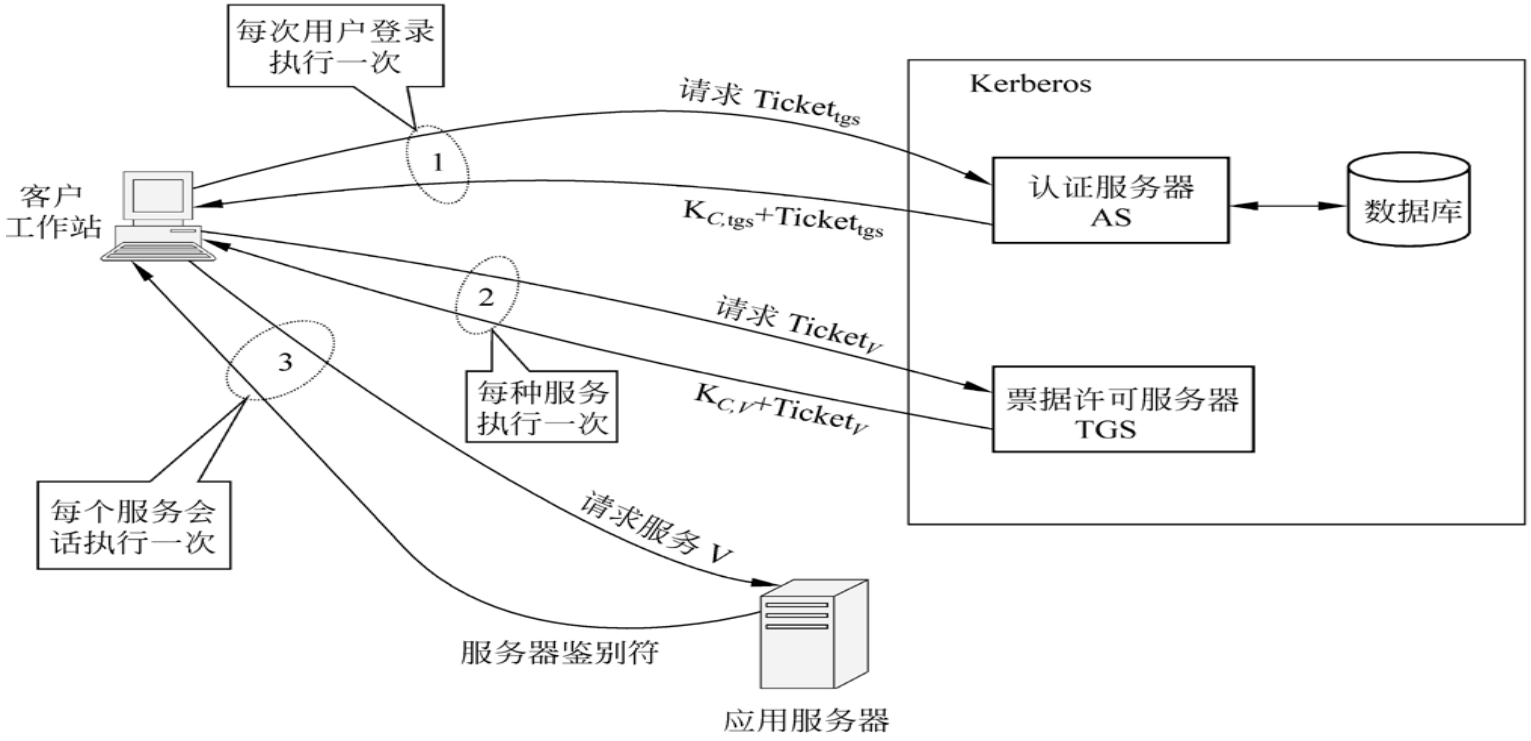
Kerberos V4协议描述：共享密钥及会话密钥



Kerberos设计思路



Kerberos（V4）协议交互过程



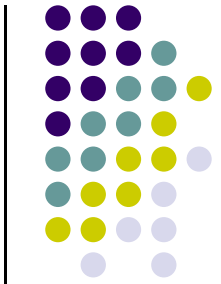
Kerberos（V4）协议的缺陷

- 依赖性
  - 加密系统的依赖性（DES）、对 IP 协议的依赖性和对时间依赖性。
- 字节顺序：没有遵循标准
- 票据有效期
  - 有效期最小为5分钟，最大约为21小时, 往往不能满足要求
- 认证转发能力
  - 不允许签发给一个用户的鉴别证书转发给其他工作站或其他客户使用

Kerberos（V4）协议的缺陷（续）

- 领域间的鉴别
  - 管理起来困难
- 加密操作缺陷
  - 非标准形式的 **DES** 加密（传播密码分组链接 **PCBC**）方式，易受攻击
- 会话密钥
  - 存在着攻击者重放会话报文进行攻击的可能
- 口令攻击
  - 未对口令提供额外的保护，攻击者有机会进行口令攻击

Kerberos（V5）协议的改进

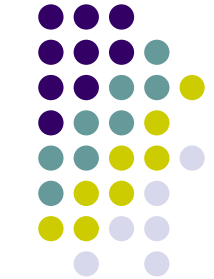


- 加密系统
  - 支持使用任何加密技术。
- 通信协议
  - **IP** 协议外，还提供了对其他协议的支持。
- 报文字节顺序
  - 采用抽象语法表示（**ASN.1**）和基本编码规则（**BER**）来进行规范。

Kerberos（V5）协议的改进（续）

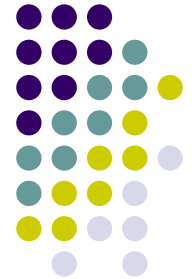
- 票据的有效期
  - 允许任意大小的有效期，有效期定义为一个开始时间和结束时间。
- 鉴别转发能力
- 更有效的方法来解决领域间的认证问题
- 口令攻击
  - 提供了一种预鉴别（**preauthentication**）机制，使口令攻击更加困难。

Kerberos 领域(realm)



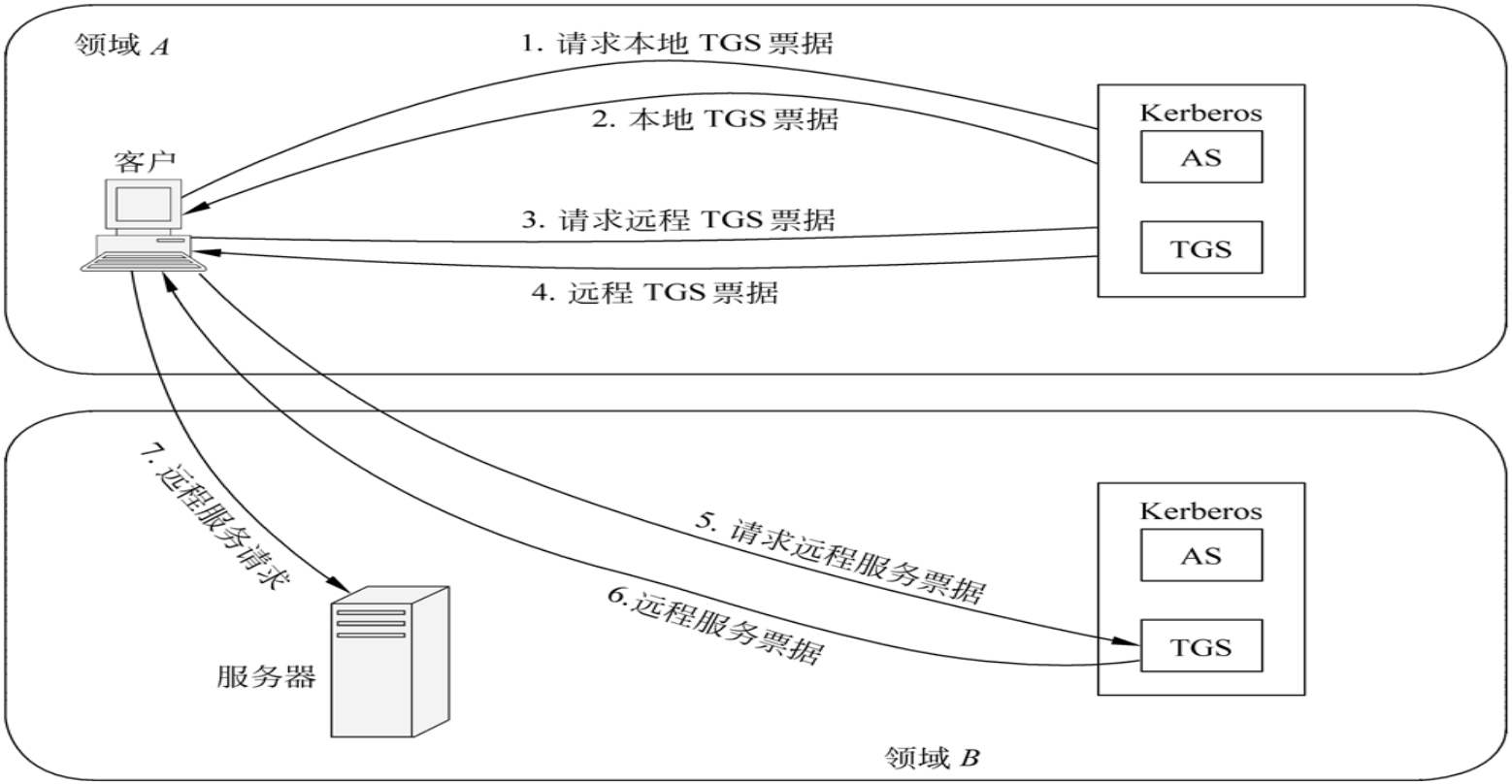
- 构成：一个完整的 **Kerberos** 环境包括一个 **Kerberos** 服务器，一组工作站和一组应用服务器。
- **Kerberos** 服务器数据库中拥有所有参与用户的 **UID** 和口令散列表。
- **Kerberos**服务器必须与每一个服务器之间共享一个保密密钥。
- 所有用户均在 **Kerberos** 服务器上注册。
- 所有服务器均在 **Kerberos** 服务器上注册。
- 领域的划分是根据网络的管理边界来划定的。

Kerberos 领域间的互通



- 跨领域的服务访问
  - 一个用户可能需要访问另一个 **Kerberos** 领域中应用服务器；
  - 一个应用服务器也可以向其他领域中的客户提供网络服务。
- 领域间互通的前提
  - 支持不同领域之间进行用户身份鉴别的机制；
  - 互通领域中的 **Kerberos** 服务器之间必须共享一个密钥；
  - 同时两个 **Kerberos** 服务器也必须进行相互注册。

远程服务访问的认证过程



9.4 物理隔离措施

- 物理隔离
  - 物理隔离技术是一种将内外网络从物理上断开，但保持逻辑连接的网络安全技术
  - 任何时候内外网络都不存在连通的物理连接，同时原有的传输协议必须被中断
  - 逻辑连接指能进行适度的数据交换

9.4 物理隔离措施

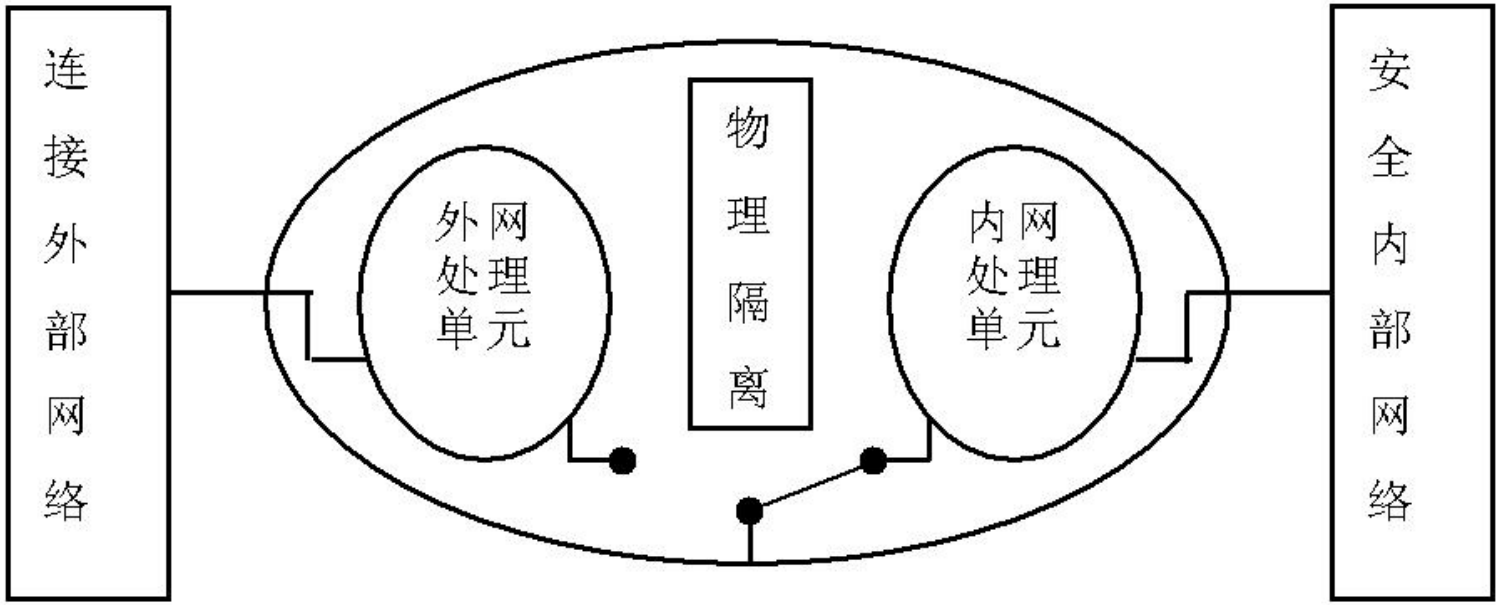
- 1999年12月29日国家保密局发布的《计算机信息系统国际联网保密管理规定》中第二章第六条规定：
- “涉及国家秘密的计算机信息系统，不得直接或间接地与国际互联网或其他公共信息网络相联接，必须进行物理隔离”

9.4 物理隔离措施

- 网络物理隔离方案
  - 客户端的物理隔离
  - 集线器级的物理隔离
  - 服务器端的物理隔离

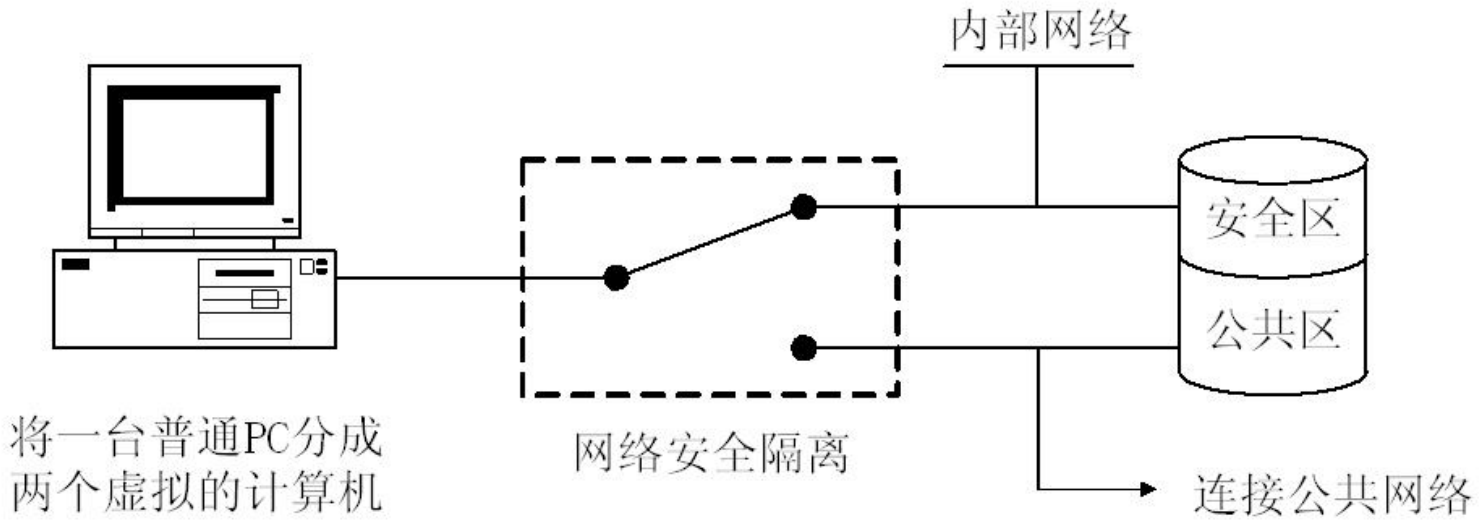
9.4 物理隔离措施

- 网络物理隔离方案
  - 客户端的物理隔离



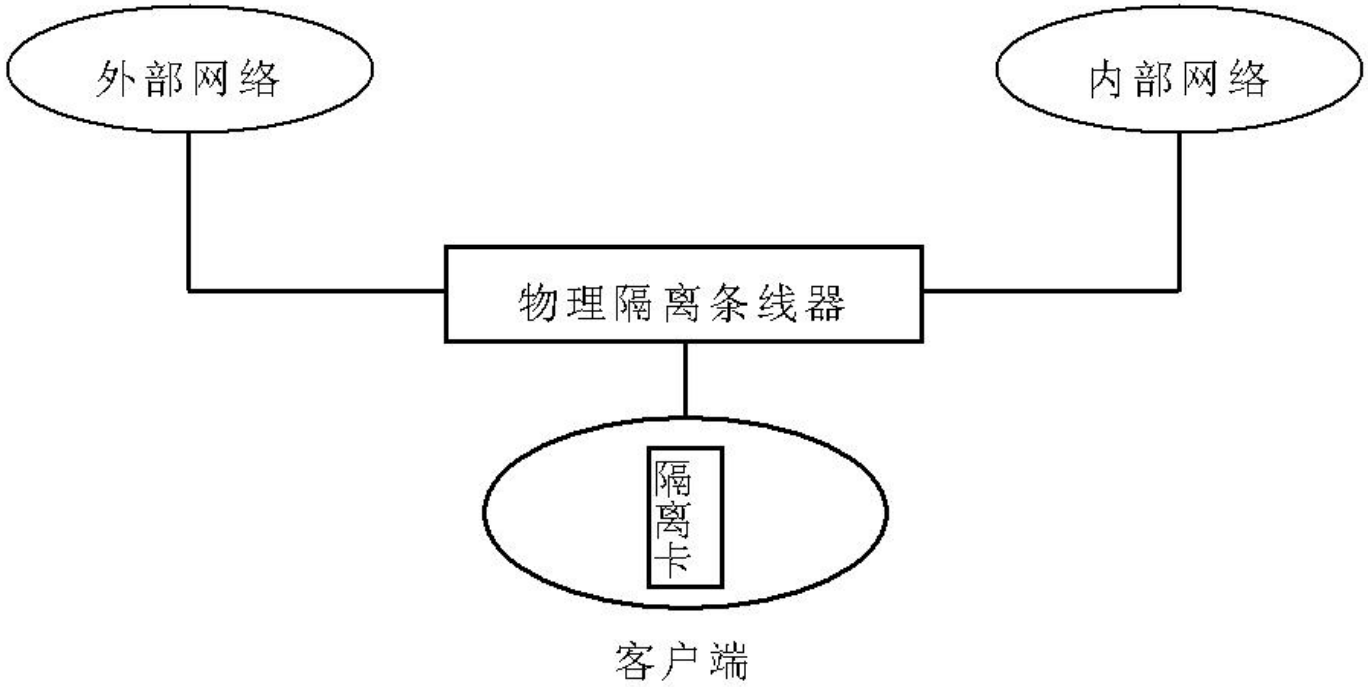
9.4 物理隔离措施

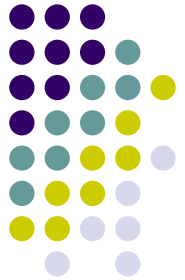
- 网络物理隔离方案
  - 客户端的物理隔离
    - 网络安全隔离卡



9.4 物理隔离措施

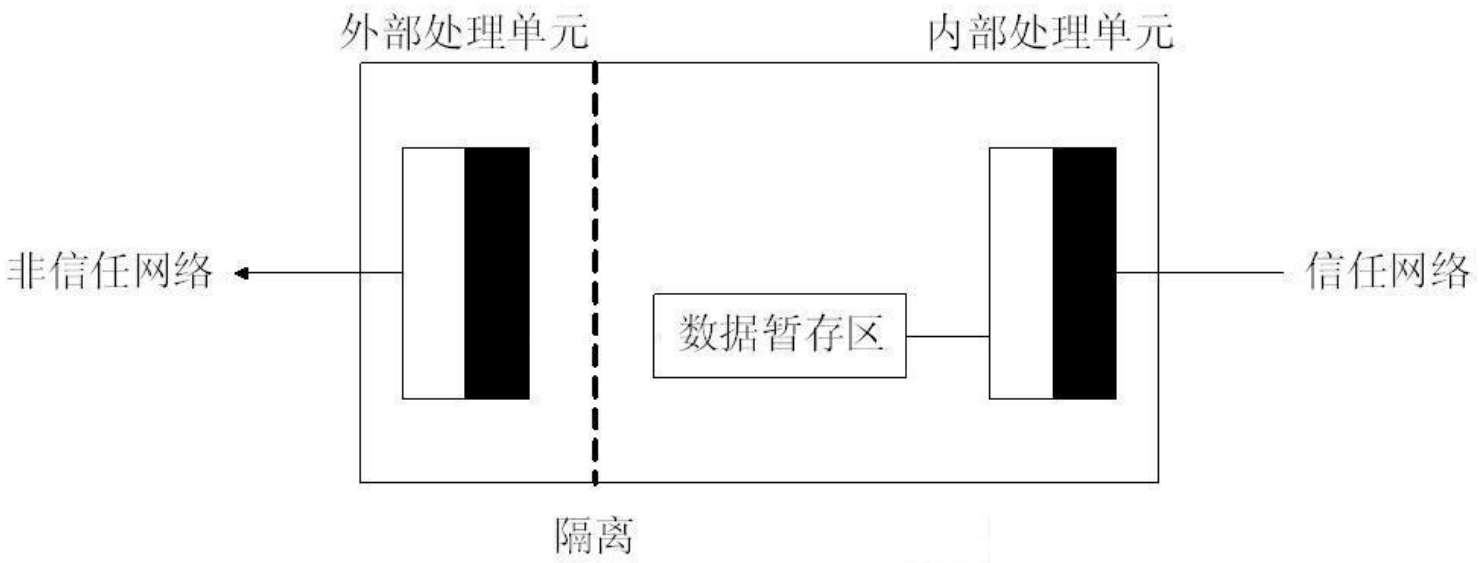
- 网络物理隔离方案
  - 集线器级的物理隔离





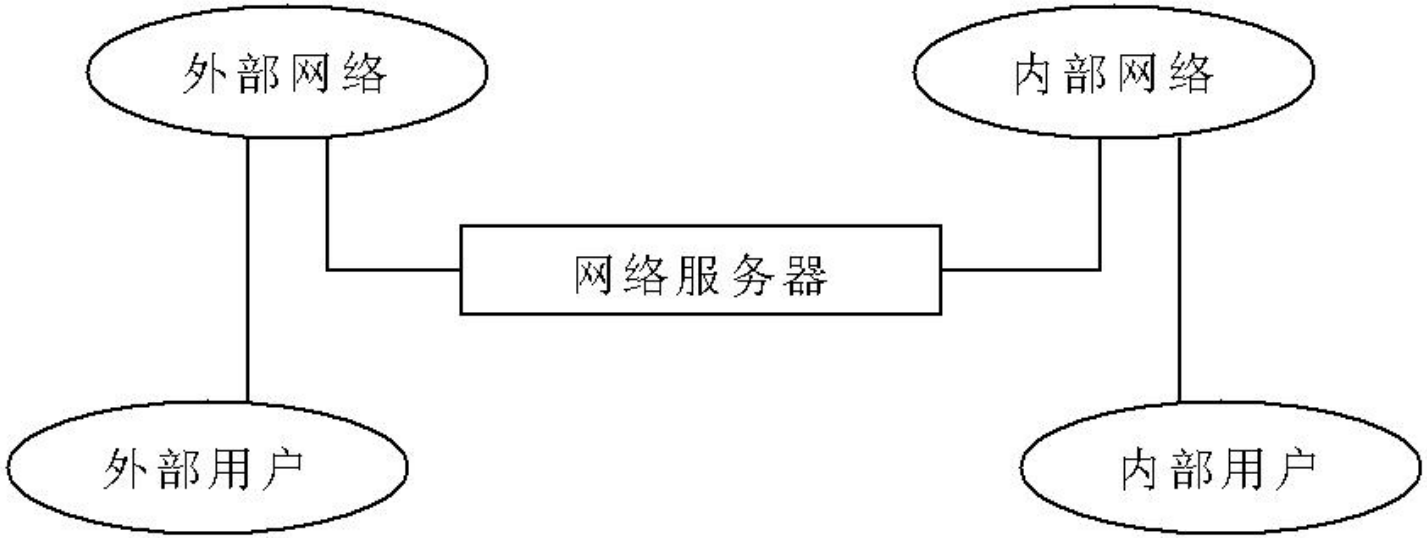
9.4 物理隔离措施

- 网络物理隔离方案
  - 集线器级的物理隔离
    - 物理隔离网闸



9.4 物理隔离措施

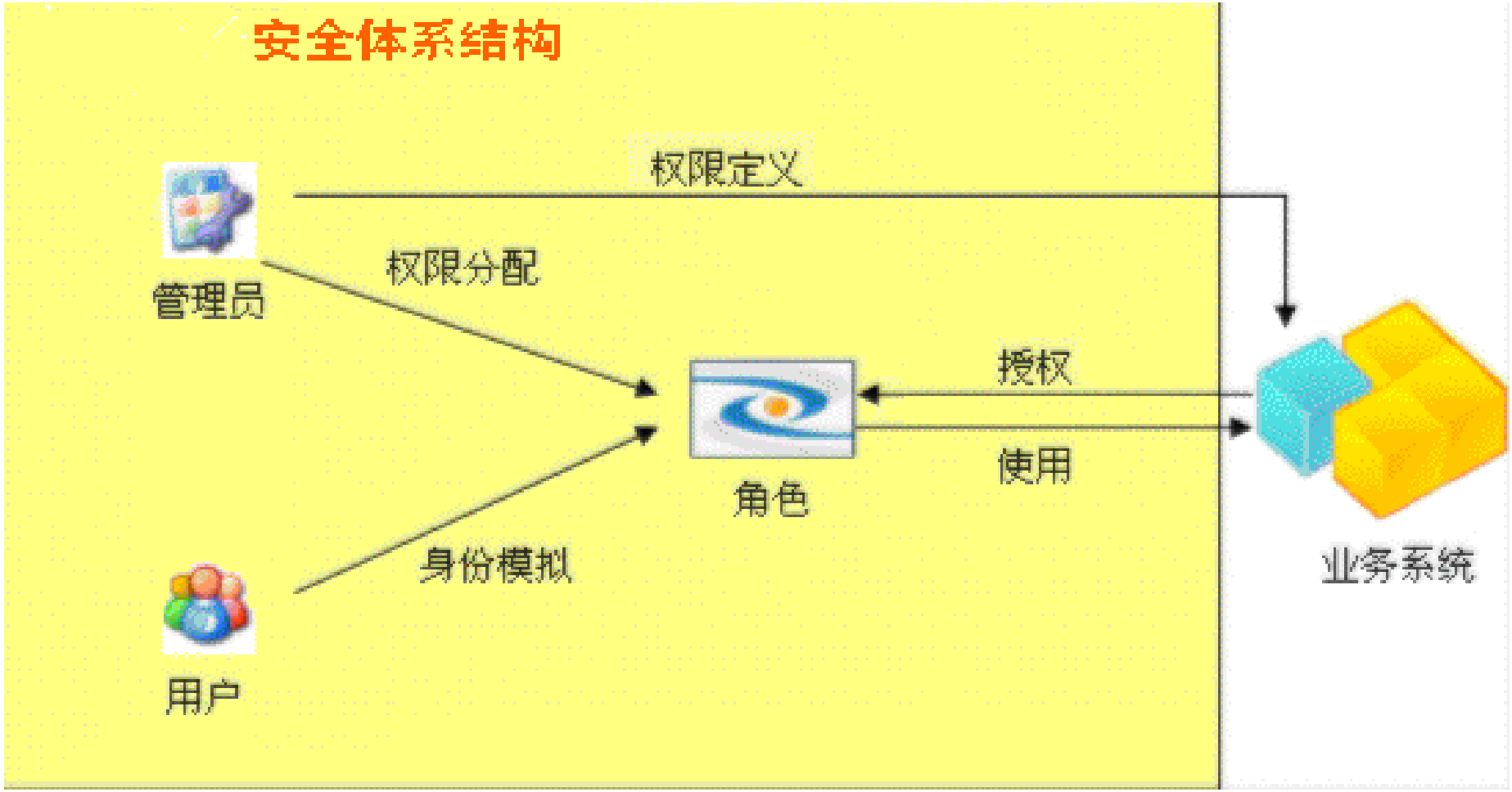
- 网络物理隔离方案
  - 服务器端的物理隔离



9.4 物理隔离措施

- 物理隔离的优点
  - 安全级别高，保障强
  - 易于在现有涉密网上安装
- 物理隔离的未尽之处
  - 资源消耗大
  - 缺乏管理
    - 认证、访问控制、审计、取证
  - 妨碍应用

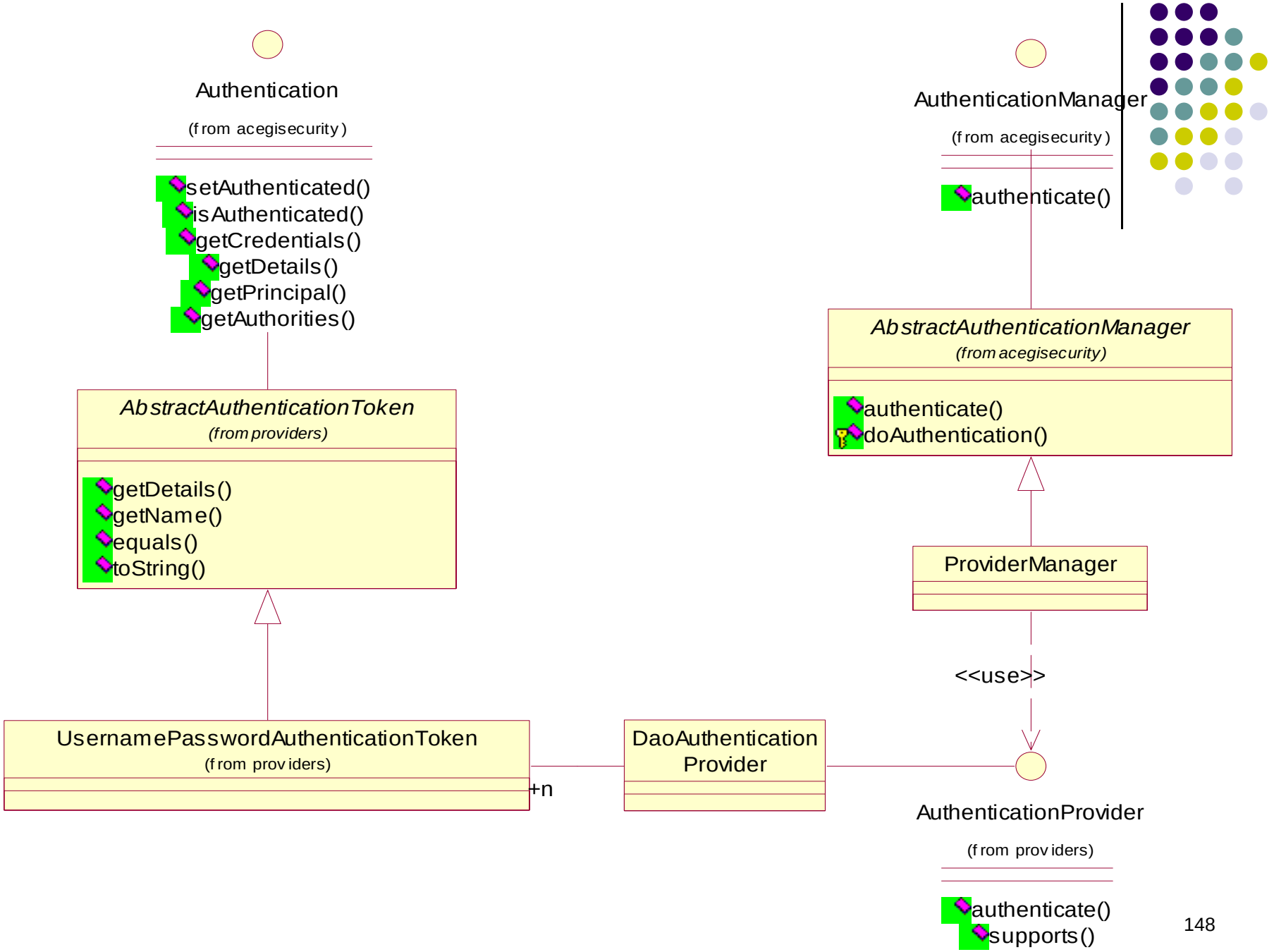
9.5 实例

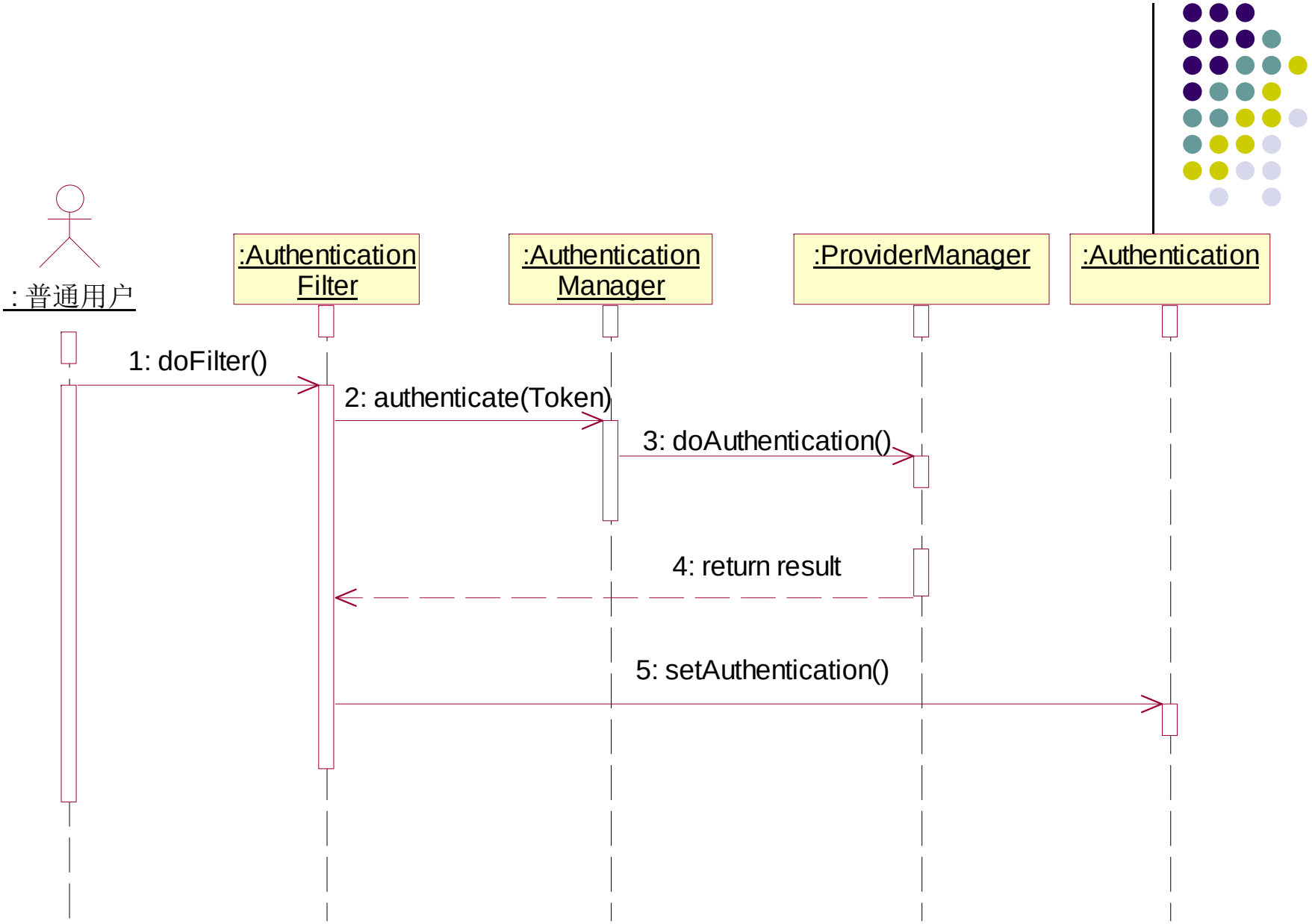


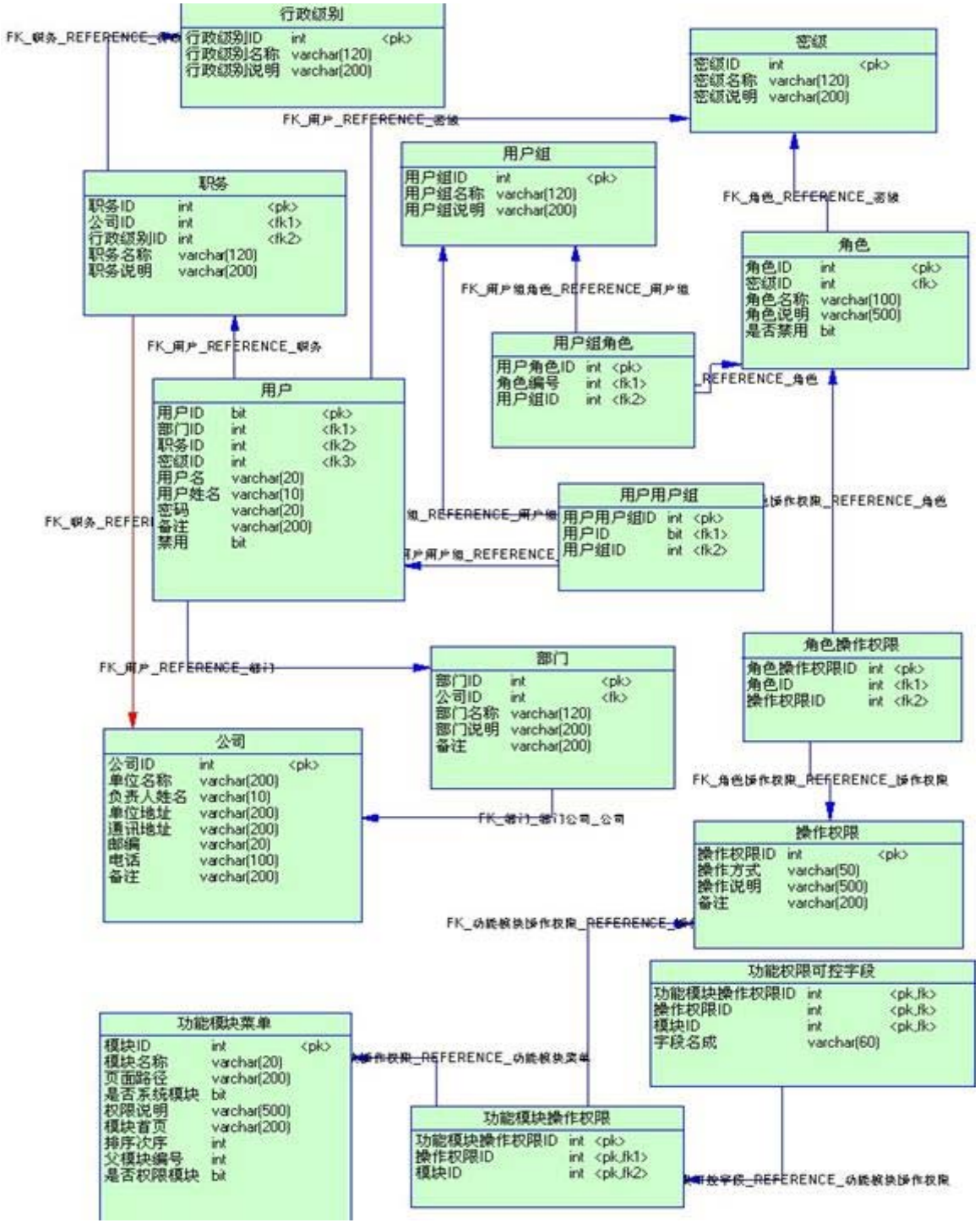
系统运行可分为以下几部分：

- 1、用户认证
- 2、权限拦截
- 3、授权管理配置：包括给资源授权和回收权限。

- 页面级权限、菜单级权限和数据级权限


Authentication
(from acegisecurity)
setAuthenticated()
isAuthenticated()
getCredentials()
getDetails()
getPrincipal()
getAuthorities()
AuthenticationManager
(from acegisecurity)
authenticate()
AbstractAuthenticationManager
(from acegisecurity)
authenticate()
doAuthentication()
AbstractAuthenticationToken
(from providers)
getDetails()
getName()
equals()
toString()
ProviderManager
<<use>>
UsernamePasswordAuthenticationToken
(from providers)
+n
DaoAuthentication Provider
AuthenticationProvider
(from providers)
authenticate()
supports()


: 普通用户
:Authentication Filter
:Authentication Manager
:ProviderManager
:Authentication
1: doFilter()
2: authenticate(Token)
3: doAuthentication()
4: return result
5: setAuthentication()


行政级别
行政级别ID int <pk>
行政级别名称 varchar(120)
行政级别说明 varchar(200)
密级
密级ID int <pk>
密级名称 varchar(120)
密级说明 varchar(200)
FK_用户_REFERENCE_密级
用户组
用户组ID int <pk>
用户组名称 varchar(120)
用户组说明 varchar(200)
FK_角色_REFERENCE_密级
职务
职务ID int <pk>
公司ID int <fk1>
行政级别ID int <fk2>
职务名称 varchar(120)
职务说明 varchar(200)
角色
角色ID int <pk>
密级ID int <fk>
角色名称 varchar(100)
角色说明 varchar(500)
是否禁用 bit
FK_用户组角色_REFERENCE_用户组
FK_用户_REFERENCE_职务
用户组角色
用户角色ID int <pk>
角色编号 int <fk1>
用户组ID int <fk2>
REFERENCE_角色
用户
用户ID bit <pk>
部门ID int <fk1>
职务ID int <fk2>
密级ID int <fk3>
用户名 varchar(20)
用户姓名 varchar(10)
密码 varchar(20)
备注 varchar(200)
禁用 bit
用户用户组
用户用户组ID int <pk>
用户ID bit <fk1>
用户组ID int <fk2>
FK_用户_REFERENCE_部门
部门
部门ID int <pk>
公司ID int <fk>
部门名称 varchar(120)
部门说明 varchar(200)
备注 varchar(200)
角色操作权限
角色操作权限ID int <pk>
角色ID int <fk1>
操作权限ID int <fk2>
公司
公司ID int <pk>
单位名称 varchar(200)
负责人姓名 varchar(10)
单位地址 varchar(200)
通讯地址 varchar(200)
邮编 varchar(20)
电话 varchar(100)
备注 varchar(200)
FK_部门_部门公司_公司
FK_角色操作权限_REFERENCE_操作权限
操作权限
操作权限ID int <pk>
操作方式 varchar(50)
操作说明 varchar(500)
备注 varchar(200)
功能权限可控字段
功能模块操作权限ID int <pk,fk>
操作权限ID int <pk,fk>
模块ID int <pk,fk>
字段名成 varchar(60)
功能模块菜单
模块ID int <pk>
模块名称 varchar(20)
页面路径 varchar(200)
是否系统模块 bit
权限说明 varchar(500)
模块首页 varchar(200)
排序次序 int
父模块编号 int
是否权限模块 bit
功能模块操作权限
功能模块操作权限ID int <pk>
操作权限ID int <pk,fk1>
模块ID int <pk,fk2>

第9章 访问控制技术

- 访问控制技术概述
- 入网认证
- 物理隔离措施
- 自主访问控制
- 强制访问控制
- 新型访问控制技术

第9章 访问控制技术

- 课后习题
  - 简述口令认证技术的认证方法。用哪些方法可以提高口令认证技术的安全性？
  - 网络的物理隔离技术包含哪几方面？它们各自采用了什么样的技术？
  - 什么是基于角色的访问控制技术？它与传统的访问控制技术有什么不同？
  - 简述四种RBAC模型技术。它们各有什么特点？