高级网络安全研究与应用——

# 安全防护技术进阶

北京邮电大学

郑康锋 zkfbupt@163.com

伍淳华 wuchunhua@bupt.edu.cn

# 安全防护技术进阶——

# 控制

# 什么是控制？

- 提到控制，你想到了什么？
- 从“控制”想到的问题
  - 控制谁？
  - 控制什么？
  - 怎么控制？

生活中，控制类似的例子？

## 安全防护技术进阶——

# 访问控制

# 什么是访问控制？

- 访问控制（Access Control）指系统对用户身份及其所属的预先定义的策略组限制其使用数据资源能力的手段。通常用于系统管理员控制用户对服务器、目录、文件等网络资源的访问。

- 访问控制是系统保密性、完整性、可用性和合法使用性的重要基础，是网络安全防范和资源保护的关键策略之一，也是主体依据某些控制策略或权限对客体本身或其资源进行的不同授权访问。

- 访问控制的主要目的是限制访问主体对客体的访问，从而保障数据资源在合法范围内得以有效使用和管理。为了达到上述目的，访问控制需要完成两个任务：识别和确认访问系统的用户、决定该用户可以对某一系统资源进行何种类型的访问。

# 什么是访问控制？

- 访问控制包括三个要素：主体、客体和控制策略。
  - （1）主体S（Subject）。是指提出访问资源具体请求。是某一操作动作的发起者，但不一定是动作的执行者，可能是某一用户，也可以是用户启动的进程、服务和设备等。
  - （2）客体O（Object）。是指被访问资源的实体。所有可以被操作的信息、资源、对象都可以是客体。客体可以是信息、文件、记录等集合体，也可以是网络上硬件设施、无限通信中的终端，甚至可以包含另外一个客体。
  - （3）控制策略A（Attribution）。是主体对客体的相关访问规则集合，即属性集合。访问策略体现了一种授权行为，也是客体对主体某些操作行为的默认。

# 访问控制策略

- 安全策略的实施原则：安全策略的制定实施也是围绕主体、客体和安全控制规则集三者之间的关系展开的。

  （1） 最小特权原则：最小特权原则是指主体执行操作时，按照主体所需权利的最小化原则分配给主体权力。最小特权原则的优点是最大限度地限制了主体实施授权行为，可以避免来自突发事件、错误和未授权用主体的危险。也就是说，为了达到一定目的，主体必须执行一定操作，但他只能做他所被允许做的，其它除外。

  （2） 最小泄漏原则：最小泄漏原则是指主体执行任务时，按照主体所需要知道的信息最小化的原则分配给主体权力。

  （3） 多级安全策略：多级安全策略是指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（TS）、秘密（S）、机密（C）、限制（RS）和无级别（U）五级来划分。多级安全策略的优点是避免敏感信息的扩散。具有安全级别的信息资源，只有安全级别比他高的主体才能够访问。

# 访问控制策略

- 多级安全系统必然要将信息资源按照安全属性分级考虑，安全类别有两种类型：
  - 一种是有层次的安全级别（Hierarchical Classification），分为TS，S，C，RS，U五级：绝密级别（Top Secret），秘密级别（Secret），机密级别（Confidential），限制级别（Restricted）和无级别级（Unclassified）；
  - 另一种是无层次的安全级别，不对主体和客体按照安全类别分类，只是给出客体接受访问时可以使用的规则和管理者。

## 访问控制——

# 访问控制模型

# 自主访问控制模型DAC

- 自主访问控制模型（DAC Model，Discretionary Access Control Model）是根据自主访问控制策略建立的一种模型，允许合法用户以用户或用户组的身份访问策略规定的客体，同时阻止非授权用户访问客体，某些用户还可以自主地把自己所拥有的客体的访问权限授予其它用户。

- 自主访问控制又称为任意访问控制。LINUX，UNIX、WindowsNT或是SERVER版本的操作系统都提供自主访问控制的功能。

- 在实现上，首先要对用户的身份进行鉴别，然后就可以按照访问控制列表所赋予用户的权限允许和限制用户使用客体的资源。主体控制权限的修改通常由特权用户或是特权用户（管理员）组实现。

# 自主访问控制模型DAC

- 任意访问控制对用户提供的这种灵活的数据访问方式，使得DAC广泛应用在商业和工业环境中；由于用户可以任意传递权限，那么，没有访问文件File1权限的用户A就能够从有访问权限的用户B那里得到访问权限或是直接获得文件File1；因此，DAC模型提供的安全防护还是相对比较低的，不能给系统提供充分的数据保护。

- 自主访问控制模型的特点是授权的实施主体（1、可以授权的主体；2、管理授权的客体；3、授权组）自主负责赋予和回收其他主体对客体资源的访问权限。DAC模型一般采用访问控制矩阵和访问控制列表来存放不同主体的访问控制信息，从而达到对主体访问权限的限制目的。

# 强制访问控制模型（MAC）

- 强制访问控制模型（MAC Model：Mandatory Access Control Model）一种多级访问控制策略。

- 系统对访问主体和受控对象实行强制访问控制，系统事先给访问主体和受控对象分配不同的安全级别属性，实施访问控制时，系统先对访问主体和受控对象的安全级别属性进行比较，再决定访问主体能否访问该受控对象。

- MAC对访问主体和受控对象标识两个安全标记：一个是具有偏序关系的安全等级标记；另一个是非等级分类标记。当主体s的安全类别为TS，而客体o的安全类别为S时，用偏序关系可以表述为SC(s)≥SC(o)。

# 强制访问控制模型（MAC）

- 考虑到偏序关系，主体对客体的访问主要有四种方式：
  （1）向下读（rd，read down）：主体安全级别高于客体信息资源的安全级别时允许查阅的读操作；
  （2）向上读（ru，read up）：主体安全级别低于客体信息资源的安全级别时允许的读操作；
  （3）向下写（wd，write down）：主体安全级别高于客体信息资源的安全级别时允许执行的动作或是写操作；
  （4）向上写（wu，write up）：主体安全级别低于客体信息资源的安全级别时允许执行的动作或是写操作。

# 强制访问控制模型（MAC）

- MAC模型中的几种主要模型：Lattice模型，Bell-LaPadula模型（BLP Model）和Biba模型（Biba Model）。

- **Lattice模型**

在Lattices模型中，每个资源和用户都服从于一个安全类别。这些安全类别我们称为安全级别，也就是我们在本章开始所描述的五个安全级别，TS，S，C，R，U。在整个安全模型中，信息资源对应一个安全类别，用户所对应的安全级别必须比可以使用的客体资源高才能进行访问。Lattices模型是实现安全分级的系统，这种方案非常适用于需要对信息资源进行明显分类的系统。

# 强制访问控制模型（MAC）

- **Bell-LaPadula模型**

  BLP[Bell and LaPadula，1976]模型是典型的信息保密性多级安全模型，主要应用于军事系统。

- **无上读、无下写**

- Bell-LaPadula模型通常是处理多级安全信息系统的设计基础，客体在处理绝密级数据和秘密级数据时，要防止处理绝密级数据的程序把信息泄露给处理秘密级数据的程序。

- BLP模型的出发点是维护系统的保密性，有效地防止信息泄露，忽略了完整性指标，使非法、越权篡改成为可能。

# 强制访问控制模型（MAC）

- **Biba模型**

  Biba模型[Biba, 1977]在研究BLP模型的特性时发现，BLP模型只解决了信息的保密问题，其在完整性定义存在方面有一定缺陷。

- **禁止向上写，没有向下读**

- Biba模型模仿BLP模型的信息保密性级别，定义了信息完整性级别，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，从而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。

- Biba模型可同时针对有层次的安全级别和无层次的安全种类。

# 基于角色的访问控制模型

- 基于角色的访问控制模型（RBAC Model，Role-based Access Model）的基本思想是将访问许可权分配给一定的角色，用户通过饰演不同的角色获得角色所拥有的访问许可权。

- RBAC从控制主体的角度出发，根据管理中相对稳定的职权和责任来划分角色，将访问权限与角色相联系，这点与传统的MAC和DAC将权限直接授予用户的方式不同；

- 通过给用户分配合适的角色，让用户与访问权限相联系。角色成为访问控制中访问主体和受控对象之间的一座桥梁。

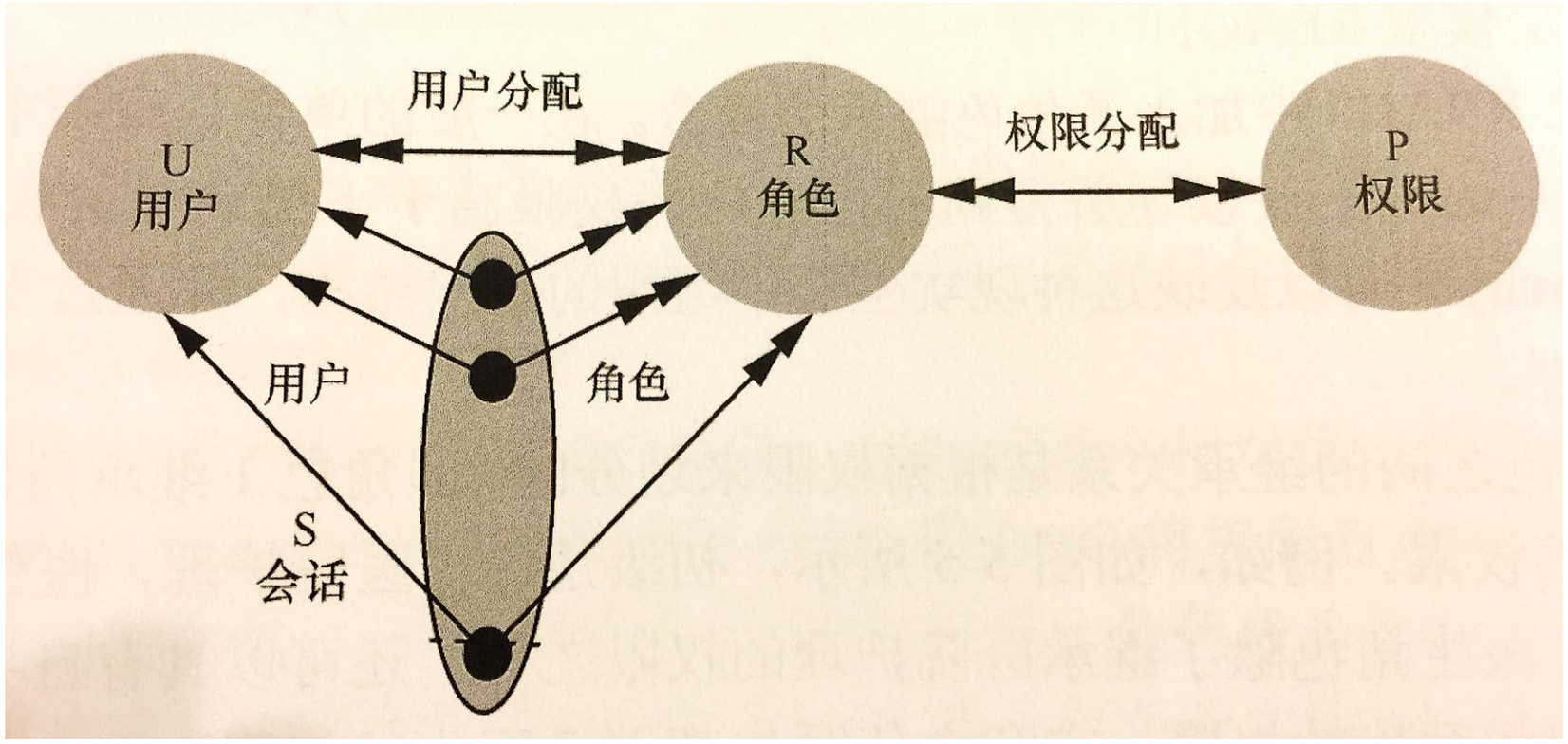
# 基于角色的访问控制模型

- 相比较而言， RBAC是实施面向企业的安全策略的一种有效的访问控制方式，其具有灵活性、方便性和安全性的特点，目前在大型数据库系统的权限管理中得到普遍应用。

- 角色由系统管理员定义，角色成员的增减也只能由系统管理员来执行，即只有系统管理员有权定义和分配角色。用户与客体无直接联系，他只有通过角色才享有该角色所对应的权限，从而访问相应的客体。

- 用户不能自主地将访问权限授给别的用户，这是RBAC与DAC的根本区别所在。RBAC与MAC的区别在于：MAC是基于多级安全需求的，而RBAC则不是。

# 基本模型RBAC0




- 表达了RBAC系统的最小需求，由User,Role,Session（一个用户和一组激活的角色）,Permission构成。
- 用户分配（User Assignment)，用户集合到角色集合的多对多映射。
- 权限分配（Permission Assignment）权限集合到角色集合的多对多映射。

# 角色分层模型RBAC1

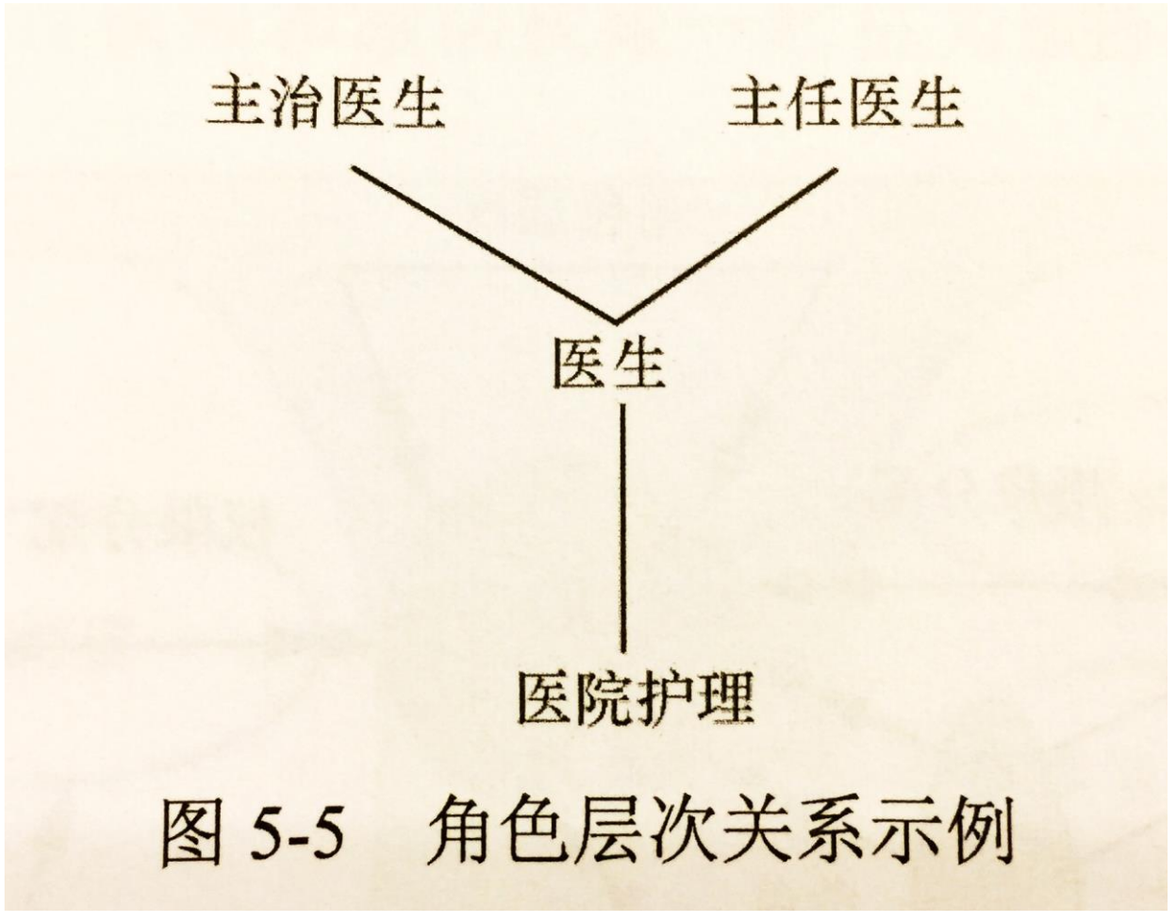


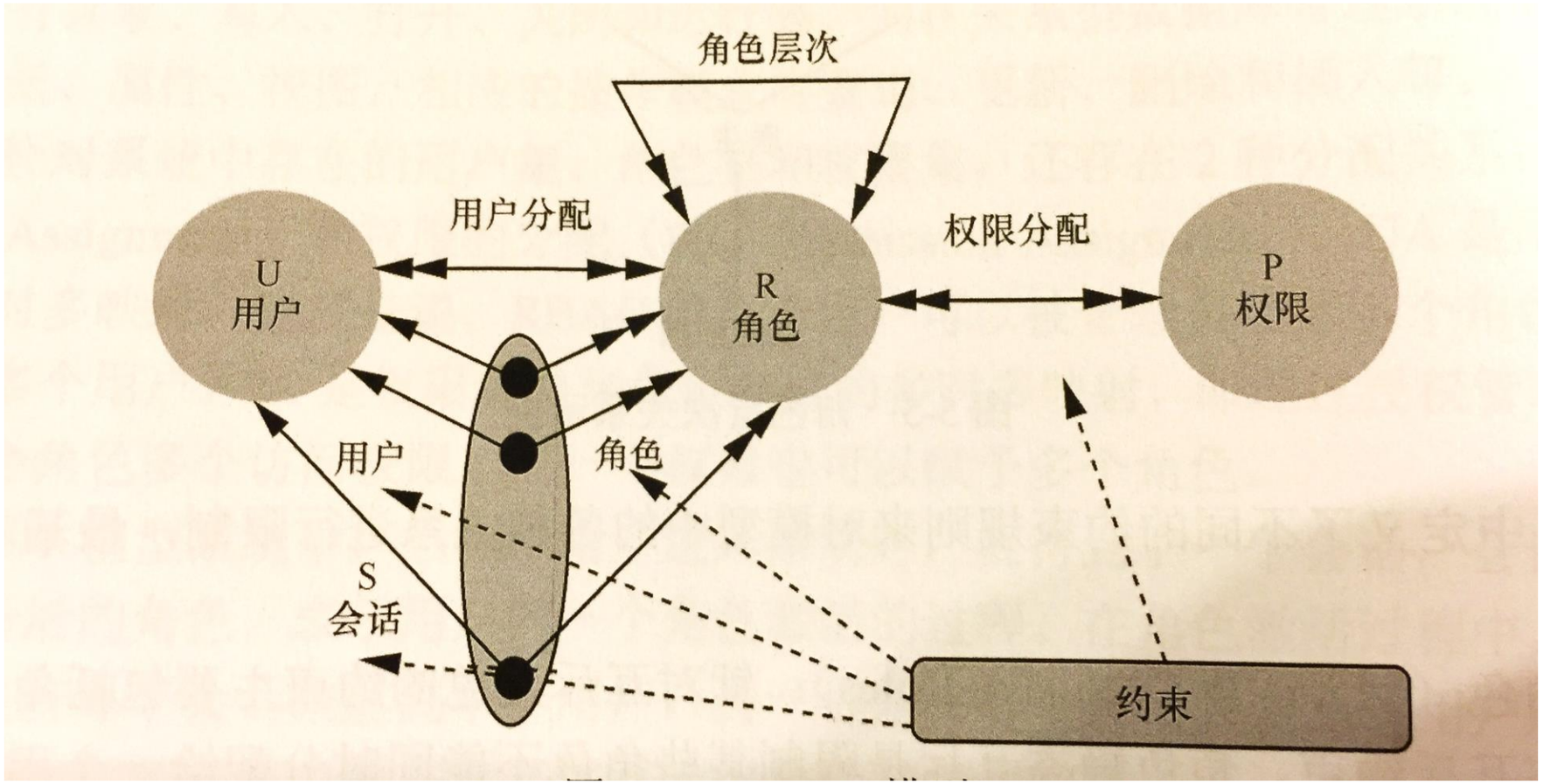

图 5-5　角色层次关系示例

- 在RBAC0基础上加入了角色继承的概念。
- 角色层次（RH，Role Hierarchy）反应层次结构，支持成员和权限继承以方便权限管理。

# 角色约束模型RBAC2

- 在RBAC0基础上引入了约束（Constraints）的概念。
- 约束是角色之间及角色与权限之间的一种限制关系。
- RBAC2定义了不同的约束规则来对各种关系进行限制：
  - 互斥角色：角色静态互斥是限制某些角色不能同时分配给一个用户，角色动态互斥是一个用户开始会话，不能同时激活某些角色。
  - 基数约束：可以限制一个角色可以分配的最大和最小用户数。
  - 先决角色：用户为获得某些高级角色必须先拥有低等级角色。
  - 会话约束：限制仅在特定会话中才允许激活某个角色，还可以限制一个用户在同一时间可以激活的会话数量。
  - 等级约束：限制角色的层次不能超过多少层。

# 统一模型RBAC3




- 在RBAC0基础上加入了角色继承和约束。
- 实际上是把RBAC1和RBAC2结合在了一起，既提供了角色的继承关系，又提供角色之间以及角色与权限之间的限制关系。

# 基于任务的访问控制模型（TBAC）

- 基于任务的访问控制模型（TBAC Model，Task-based Access Control Model）是从应用和企业层角度来解决安全问题，以面向任务的观点，从任务（活动）的角度来建立安全模型和实现安全机制，在任务处理的过程中提供动态实时的安全管理。
- 在TBAC中，对象的访问权限控制并不是静止不变的，而是随着执行任务的上下文环境发生变化。TBAC首要考虑的是在工作流的环境中对信息的保护问题，TBAC是一种上下文相关的访问控制模型。其次，TBAC不仅能对不同工作流实行不同的访问控制策略，而且还能对同一工作流的不同任务实例实行不同的访问控制策略。
- TBAC是基于任务的，是一种基于实例（instance-based）的访问控制模型。
- TBAC模型由工作流、授权结构体、受托人集、许可集四部分组成。

## 访问控制——

# 访问控制实现

# 访问控制表

- 访问控制表（ACLs：Access Control Lists）是以文件为中心建立的访问权限表，简记为ACLs。目前，大多数PC、服务器和主机都使用ACLs作为访问控制的实现机制。

- 访问控制表的优点在于实现简单，任何得到授权的主体都可以有一个访问表，例如授权用户A1的访问控制规则存储在文件File1中，A1的访问规则可以由A1下面的权限表ACLsA1来确定，权限表限定了用户UserA1的访问权限。

# 访问控制矩阵

- 访问控制矩阵（ACM：Access Control Matrix）是通过矩阵形式表示访问控制规则和授权用户权限的方法；也就是说，对每个主体而言，都拥有对哪些客体的哪些访问权限；而对客体而言，又有哪些主体对他可以实施访问；将这种关连关系加以阐述，就形成了控制矩阵。其中，特权用户或特权用户组可以修改主体的访问控制权限。

- 访问控制矩阵的实现很易于理解，但是查找和实现起来有一定的难度，而且，如果用户和文件系统要管理的文件很多，那么控制矩阵将会成几何级数增长，这样对于增长的矩阵而言，会有大量的空余空间。

# 访问控制能力列表

- 能力是访问控制中的一个重要概念，它是指请求访问的发起者所拥有的一个有效标签（ticket），它授权标签表明的持有者可以按照何种访问方式访问特定的客体。访问控制能力表（ACCLs：Access Control Capabilitis Lists）是以用户为中心建立访问权限表，ACCLs的具体实现见图6.3.3。例如，访问控制权限表ACCLsF1表明了授权用户UserA对文件File1的访问权限，UserAF表明了UserA对文件系统的访问控制规则集。因此，ACCLs的实现与ACLs正好相反。定义能力的重要作用在于能力的特殊性，如果赋予哪个主体具有一种能力，事实上是说明了这个主体具有了一定对应的权限。能力的实现有两种方式，传递的和不可传递的。一些能力可以由主体传递给其他主体使用，另一些则不能。能力的传递牵扯到了授权的实现，我们在后面会具体阐述访问控制的授权管理。

# 访问控制安全标签列表

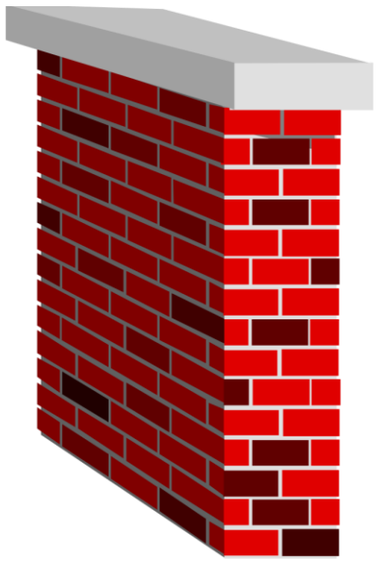
- 安全标签是限制和附属在主体或客体上的一组安全属性信息。安全标签的含义比能力更为广泛和严格，因为它实际上还建立了一个严格的安全等级集合。访问控制标签列表（ACSLLs： Access Control Security Labels Lists）是限定一个用户对一个客体目标访问的安全属性集合。访问控制标签列表的实现示例见图6.3.4，左侧为用户对应的安全级别，右侧为文件系统对应的安全级别。假设请求访问的用户UserA的安全级别为S，那么UserA请求访问文件File2时，由于S<TS，访问会被拒绝；当UserA请求访问文件FileN时，因为S>C，所以允许访问。

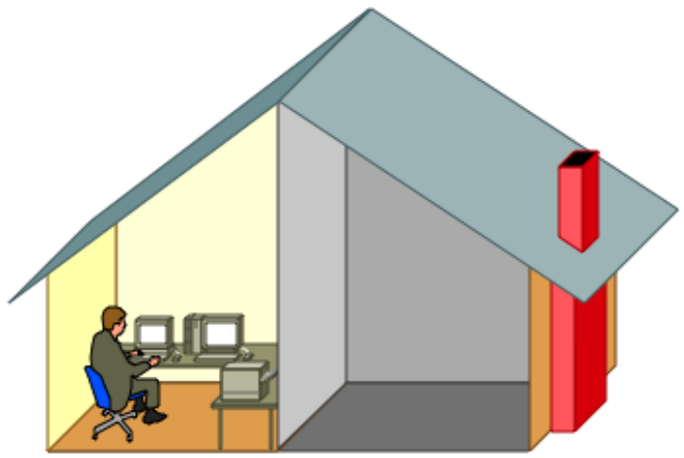
## 访问控制——

# 防火墙——概念

# 什么是防火墙？

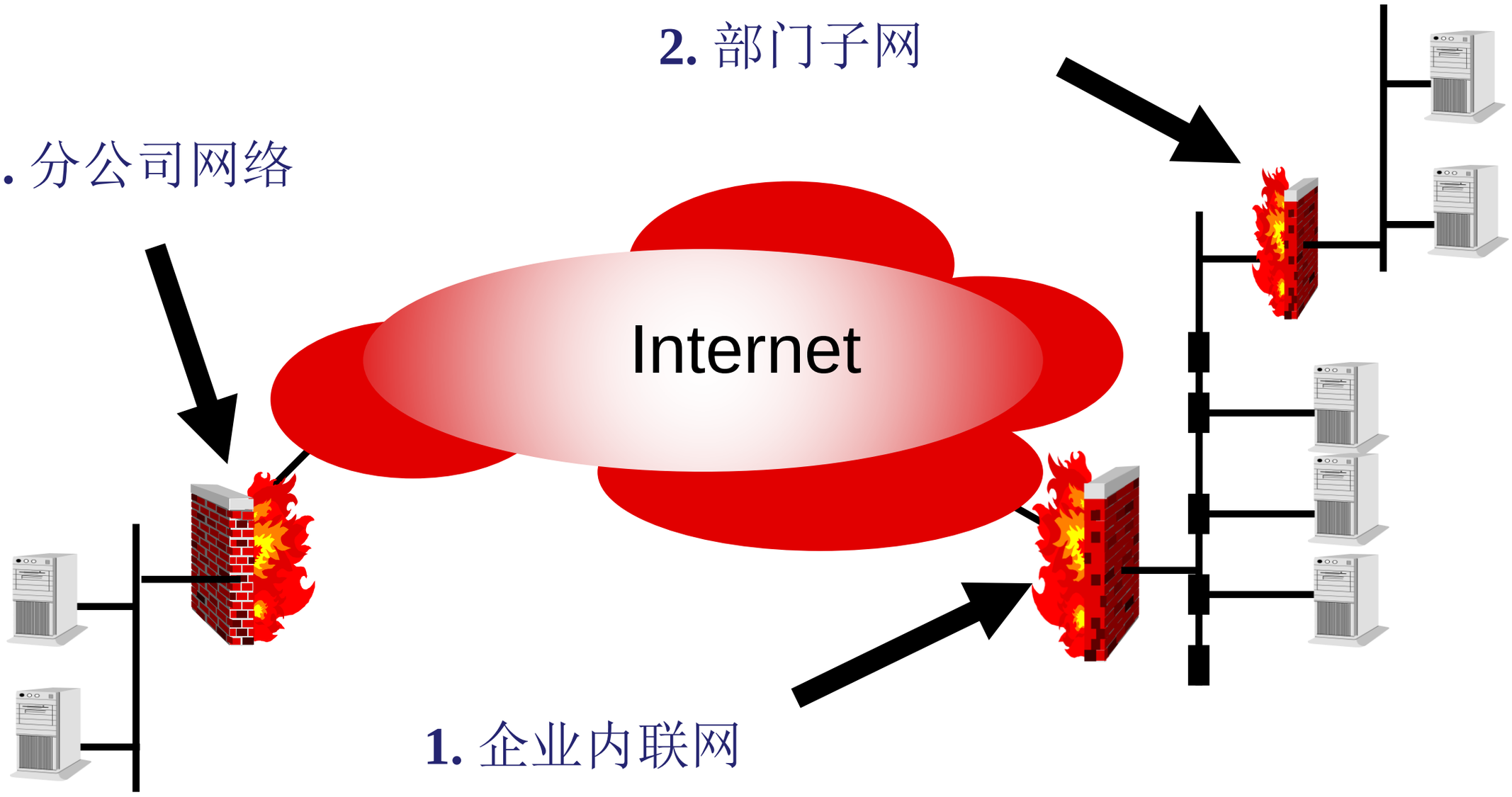
# 防火墙





- 传统的防火墙用来防止火从大厦的一部分传播到另一部分。

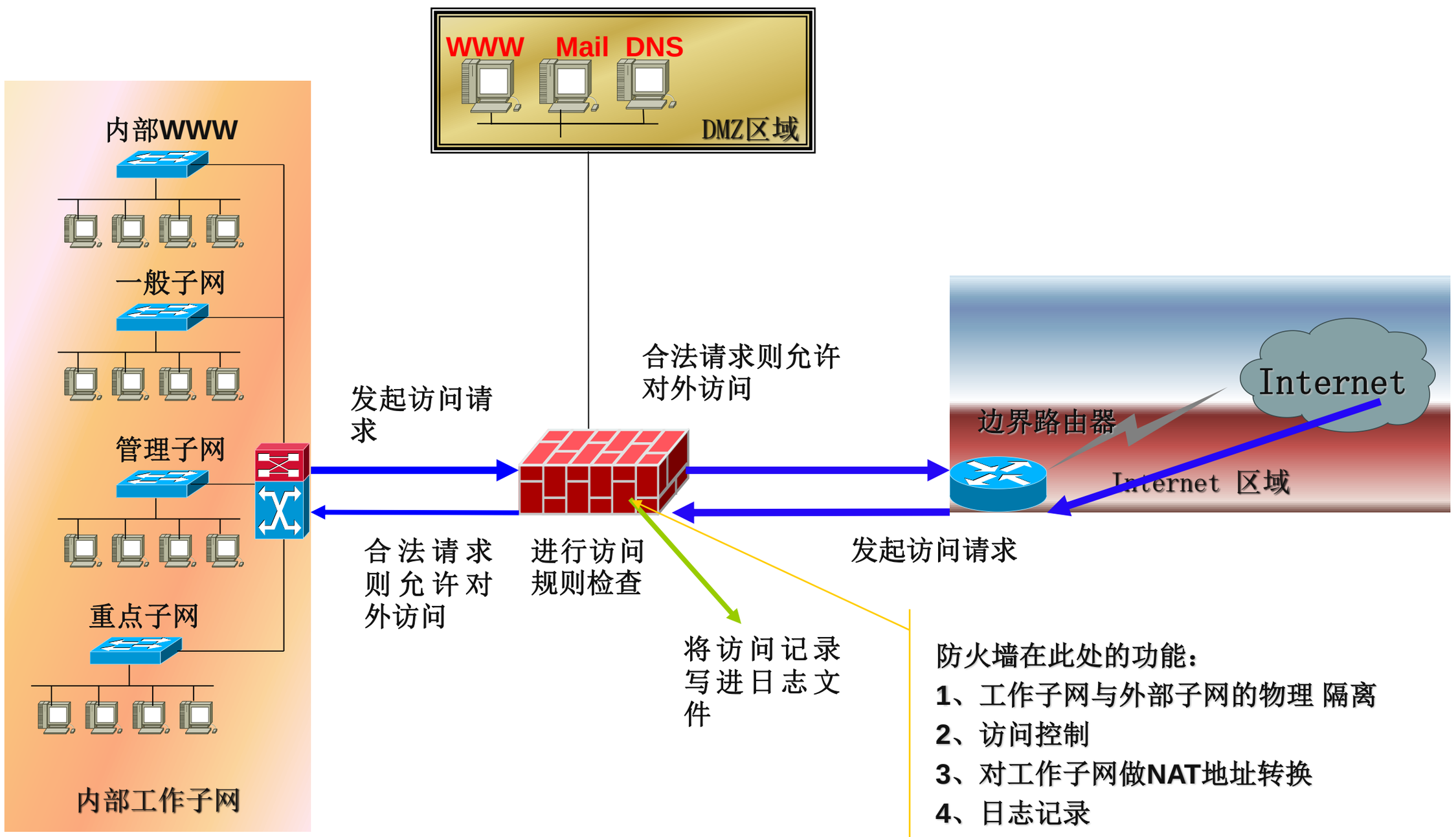
# 防火墙示意图



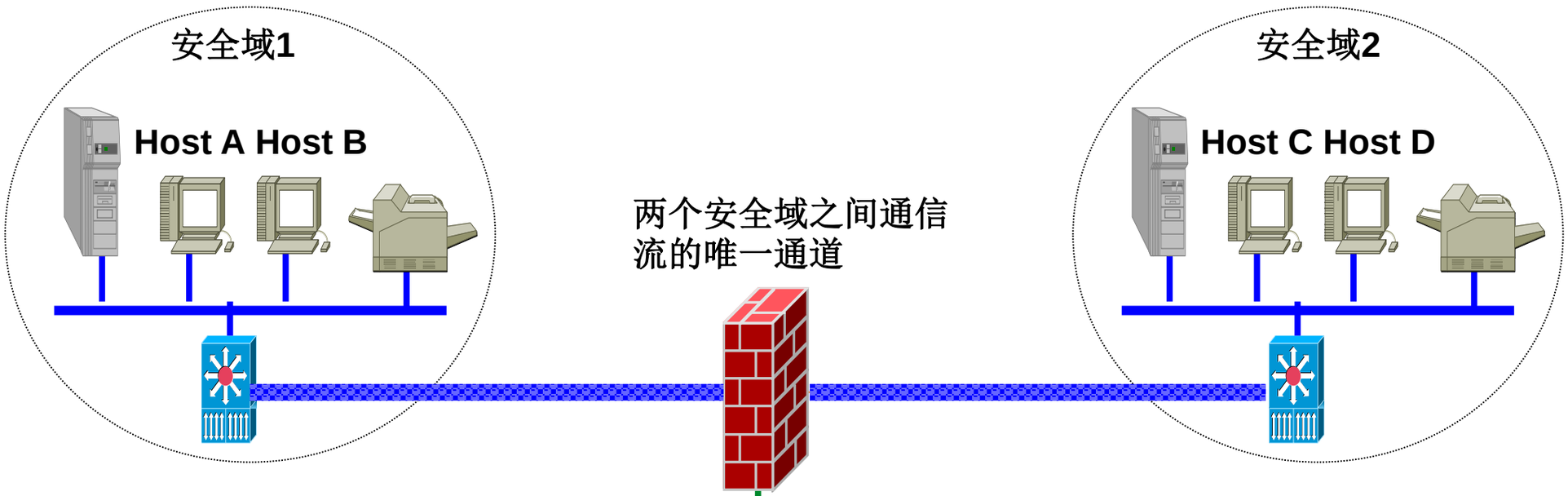
# 防火墙定义

- 防火墙是位于两个(或多个)网络间实施网间访问控制的一组组件的集合。
- 它满足以下条件
  - 所有进出被保护网络的通信必须通过防火墙
  - 所有通过防火墙的通信必须经过安全策略的过滤或者防火墙的授权
  - 防火墙自身应对渗透(peneration)免疫

# 一个典型的防火墙使用形态



# IT 领域使用的防火墙概念




| Source | Destination | Permit | Protocol |
|---|---|---|---|
| Host A | Host C | Pass | TCP |
| Host B | Host C | Block | UDP |

一种高级访问控制设备，置于不同网络安全域之间的一系列部件的组合。

根据访问控制规则决定进出网络的行为

是不同网络安全域间通信流的唯一通道，能根据企业有关的安全政策控制（进出网络的访问行为。

## 访问控制——

# 防火墙——技术

# 防火墙技术

- 网络地址翻译
- 静态包过滤
- 动态包过滤
- 电路级网关
- 应用层网关(代理服务器)
- 状态检查包过滤

# 包过滤防火墙

## 静态包过滤
## 动态包过滤

- 对所接收的每个数据包做允许、拒绝的决定。
- 审查每个数据报以便确定其是否与某一条包过滤规则匹配。

## 优缺点

- 逻辑简单，成本低；对网络性能的影响较小，有较强的透明性，定制策略灵活。
- 允许外部客户和内部主机的直接连接；
- 不提供用户的鉴别机制；
- 仅工作在网络/传输层，提供较低水平的安全性。

## 判断依据

- 基本信息
  - 地址信息：源、目的IP地址
  - 协议信息：数据包协议类型TCP、 UDP、 ICMP 、IGMP等
  - 源、目的端口FTP、 HTTP 、DNS等
- 协议具体信息
  - IP选项源路由、记录路由等
  - TCP选项SYN 、ACK、 FIN 、RST等
  - 其它协议选项ICMP、 ECHO、ICMP、 ECHO 、REPLY等
- 流向及接口信息
  - 数据包流向in或out
  - 数据包流经网络接口eth0 eth1

# 包过滤防火墙原理

| Source | Destination | Permit | Protocol |
|---|---|---|---|
| Host A | Host C | Pass | TCP |
| Host B | Host C | Block | UDP |

控制策略

查找对应的
控制策略

根据策略决定如
何处理该数据包

拆开数据包

数据包

数据包

数据包

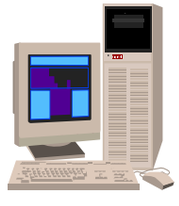
数据包

安全网域

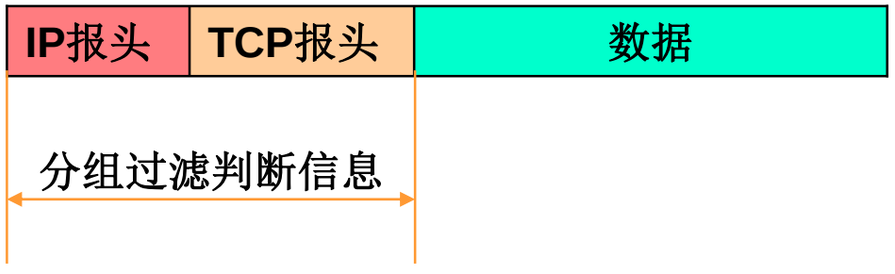
Host C Host D

数据包

数据包



过滤依据主要是TCP/IP报头里面的
信息，不能对应用层数据进行处理



# 包过滤防火墙原理

| Source | Destination | Permit | Protocol |
|---|---|---|---|
| Host A | Host C | Pass | TCP |
| Host B | Host C | Block | UDP |

控制策略

过滤依据主要是TCP/IP报头里面的信息，不能对应用层数据进行处理

查找对应的控制策略

根据策略决定如何处理该数据包

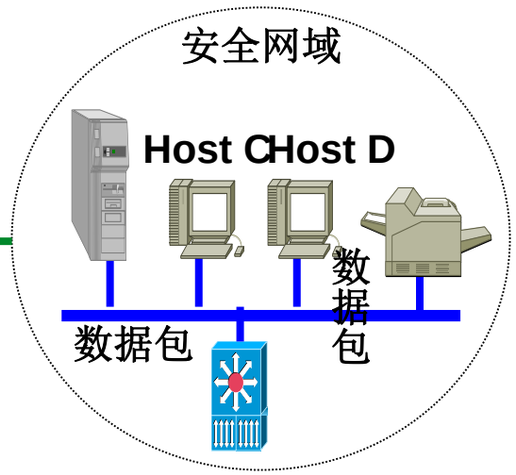
拆开数据包

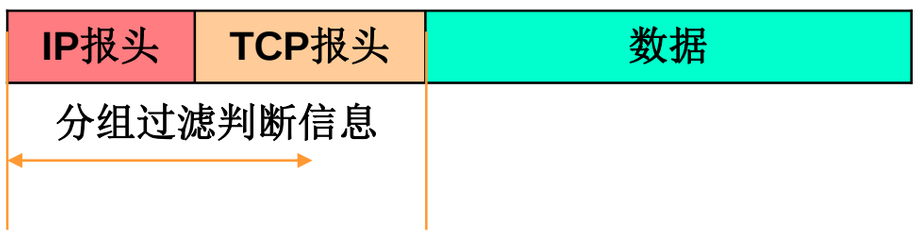
数据包

数据包

数据包

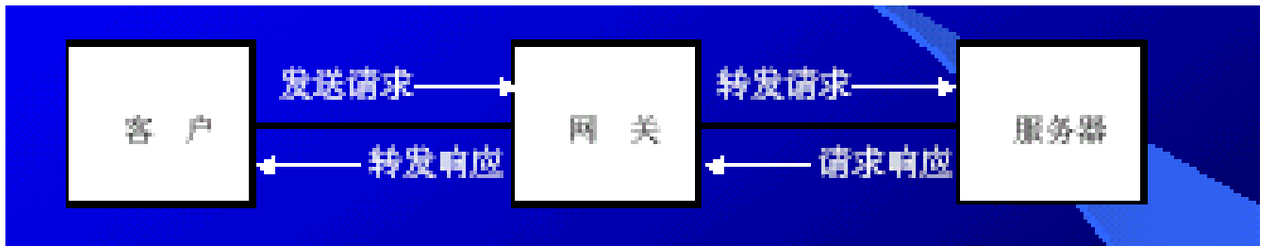
数据包



# 应用网关防火墙

（AGF,Application Gateway Firewall），又称代理防火墙或简称应用网关。




- 应用网关在应用层处理信息
- AGF可以支持多个应用，如E-mail，Web，DNS，Telnet，FTP等

## 缺点

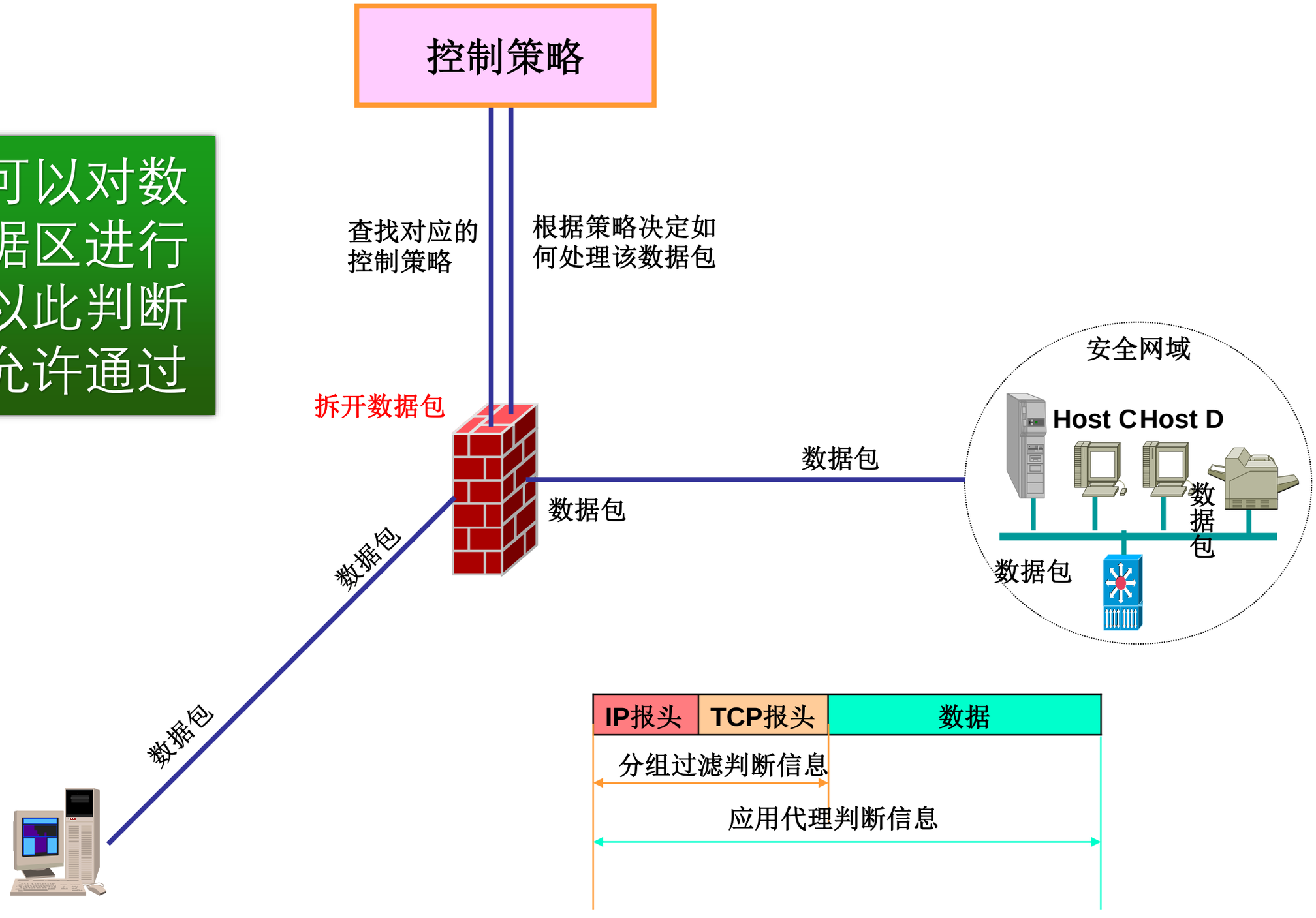
- 代理速度比包过滤慢；
- 代理对用户不透明，给用户的使用带来不便，灵活性不够；
- 这种代理技术需要针对每种协议设置一个不同的代理服务器；
- 有时要求特定的客户端软件。

## 优点

- 不允许内外网主机的直接连接；
- 可以提供比包过滤更详细的日志记录（应用层信息）；
- 可以隐藏内部IP地址；
- 认证用户而非设备；
- 可以为用户提供透明的加密机制；
- 可以与认证、授权等安全手段方便的集成；
- 监控、过滤应用层信息；

# 应用网关

应用代理可以对数据包的数据区进行分析，并以此判断数据是否允许通过



# 状态检测防火墙

- 状态检测防火墙是在动态包过滤防火墙基础上，增加了状态检测机制而形成的。
- 具有连接的跟踪能力。

## 示例

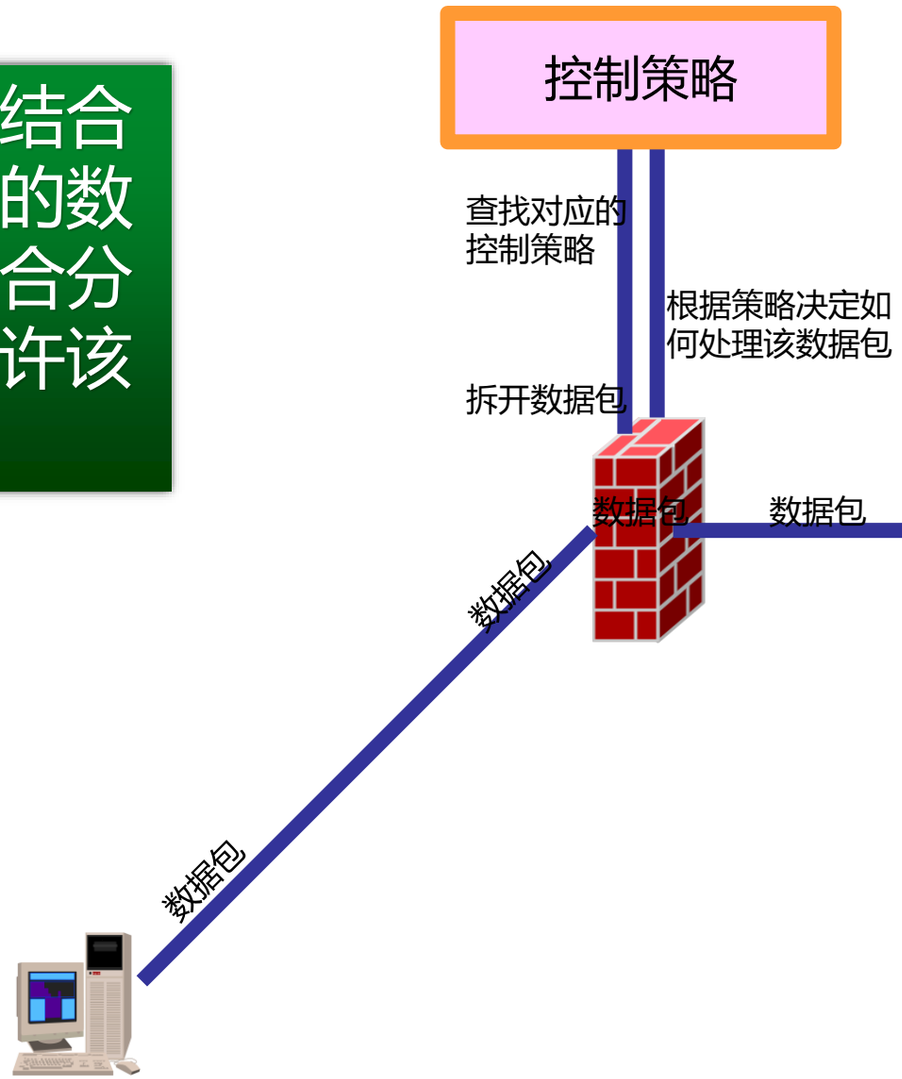
- 以TCP协议为例：所谓的状态检测机制关注的主要问题不再仅是SYN和ACK标志位，或者是来源端口和目标端口，还包括了序号、窗口大小等其它TCP协议信息。

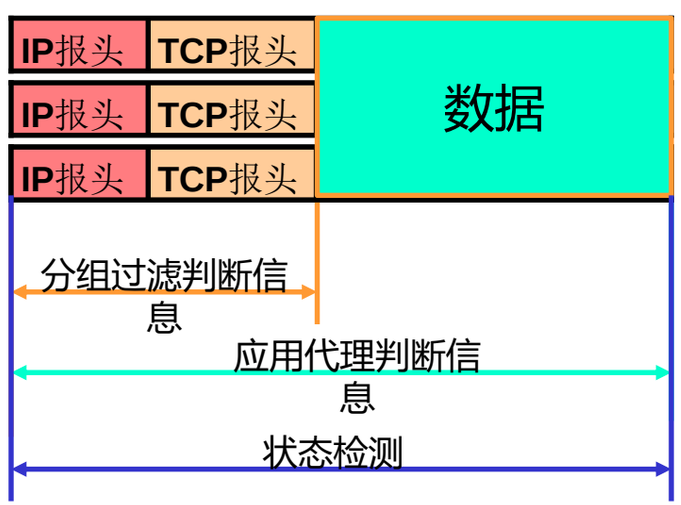
## 优缺点

- 优点：
  - 具备动态包过滤的所有优点，同时具有更高的安全性。
- 缺点：
  - 检测的层次仅限于网络层与传输层，无法对应用层内容进行检测，从而无法抵抗应用层的攻击；
  - 性能比动态包过滤稍差：因为检测更多的内容。

# 状态检测原理

状态检测可以结合前后数据包里的数据信息进行综合分析决定是否允许该包通过



## 访问控制——

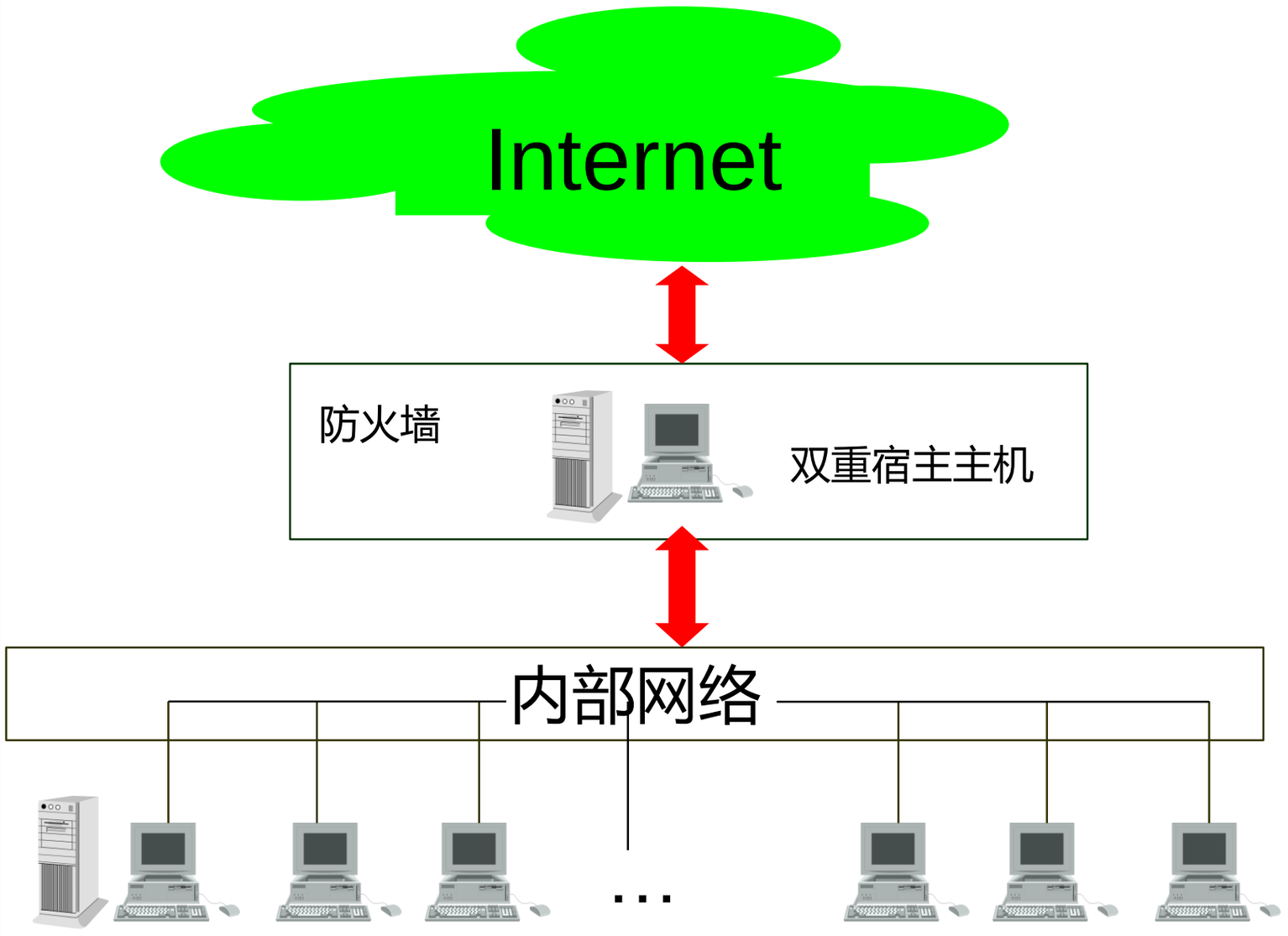
# 防火墙——结构

# 防火墙体系结构

- 双重宿主主机体系结构
- 屏蔽主机体系结构
- 屏蔽子网体系结构

# 双重宿主主机体系结构

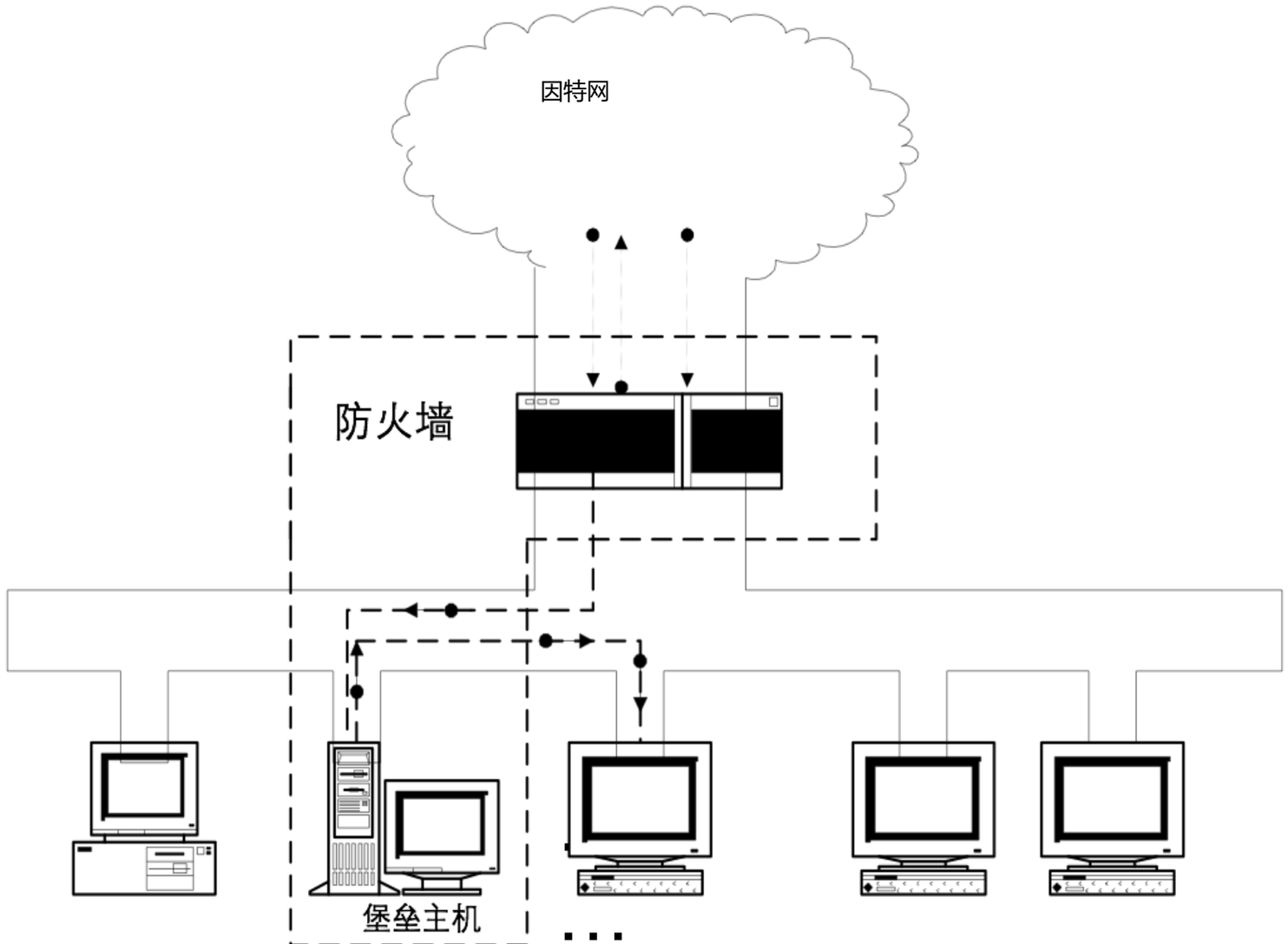
- 双重宿主主机的特性：
  - 安全至关重要（唯一通道），其用户口令控制安全是关键
  - 必须支持很多用户的访问（中转站），其性能非常重要。
- 缺点：双重宿主主机是隔开内外网络的唯一屏障，一旦它被入侵，内部网络便向入侵者敞开大门。



# 屏蔽主机体系结构

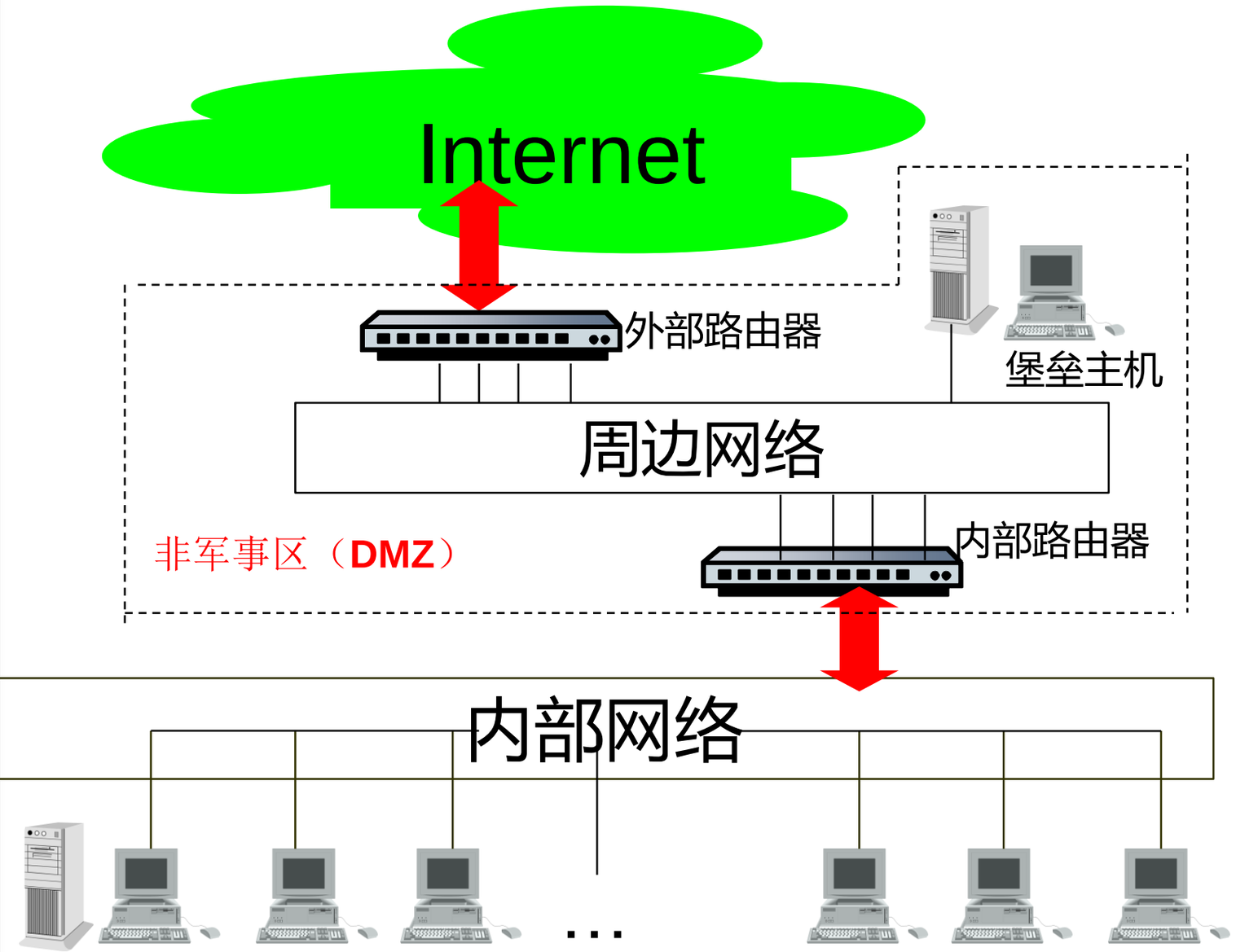
- 典型构成：包过滤路由器 + 堡垒主机。
  - 包过滤路由器保证外部系统对内部网络的操作只能经过堡垒主机
  - 堡垒主机配置在内部网络上，是外部网络主机连接到内部网络主机的桥梁。

- 安全性更高，双重保护：实现了网络层安全（包过滤）和应用层安全
- 缺点：过滤路由器能否正确配置是安全与否的关键。如果路由器被损害，堡垒主机将被穿过。



# 屏蔽子网体系结构

- 堡垒主机位于周边网络，是整个防御体系的核心。
- 外部路由器（访问路由器）
  - 作用：保护周边网络和内部网络不受外部网络的侵犯。
    - 它把入站的数据包路由到堡垒主机。
    - 防止部分IP欺骗，它可分辨出数据包是否真正来自周边网络，而内部路由器不可。
- 内部路由器（阻塞路由器）
  - 作用：保护内部网络不受外部网络和周边网络的侵害，它执行大部分过滤工作。
  - 外部路由器一般与内部路由器应用相同的规则。
- 屏蔽因）



## 访问控制——

# 各类控制实例

# 控制实例一：iptables

- 举三个规则
  - DoS
  - 扫描
  - 漏洞利用

方法：基于攻击特征匹配的原理，准确度高，误报率低，无法检测未知攻击，典型的误用检测技术。

# 控制实例一：iptables

#iptables [-AI 链] [-io 网卡接口] [-p 协议] [-s 源ip/网段] [-d 目标ip/网段] -j [ACCEPT/DROP]

- 参数说明：
  - -AI 链：针对某条链进行规则的“插入”或“添加”-A：新增加一条规则。-I：插入一条规则。
  - 例如原本有四条规则，使用-I插入一条规则后则刚插入的那条变成第一条，原来的四条规则都向后移一位。例如在-I指定了顺序（-I INPUT 3），则插入后，此条规则变成了第三的位置。
  - -io 网络接口：-i:进入口 -o：传出口
  - -p 协议：如:tcp/udp/icmp/all
  - -s 源IP/网段：设置此规则的数据包来源地址，可以是IP，也可以是一个网段
  - -d 目标IP/网段：同-s一样，只是这里指的是目标IP/网段
  - -j:后面接操作，主要的操作有接受（ACCEPT）、丢弃（DROP）、记录（LOG）

# 控制实例一：iptables

- 规则举例1:
  - 来自192.168.1.0/24可接受，但来自来192.168.1.10的所有包丢弃

```
#iptables -A INPUT -i eth0 -s 192.168.1.10 -j DROP
#iptables -A INPUT -i eth0 -s 192.168.1.0/24 -j ACCEPT
```

- 上述这个范例很重要，因为与顺序有关，要先丢弃192.168.1.10后再允许1.0这个网段。

# 控制实例一：iptables

- 规则举例2:
  - 允许端口（upd port 137,138 tcp port 139,445）数据访问。

```
#iptables -A INPUT -i eth0 -p upd --dport 137:138 -j ACCEPT
#iptables -A INPUT -i eth0 -p tcp --dport 139 -j ACCEPT
#iptables -A INPUT -i eth0 -p tcp --dport 445 -j ACCEPT
```

# 控制实例一：iptables

- use this module to avoid various denial of service attacks (DoS) with a faster rate to increase responsiveness

Syn-flood protection:

```
# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
```

Furtive port scanner:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
```

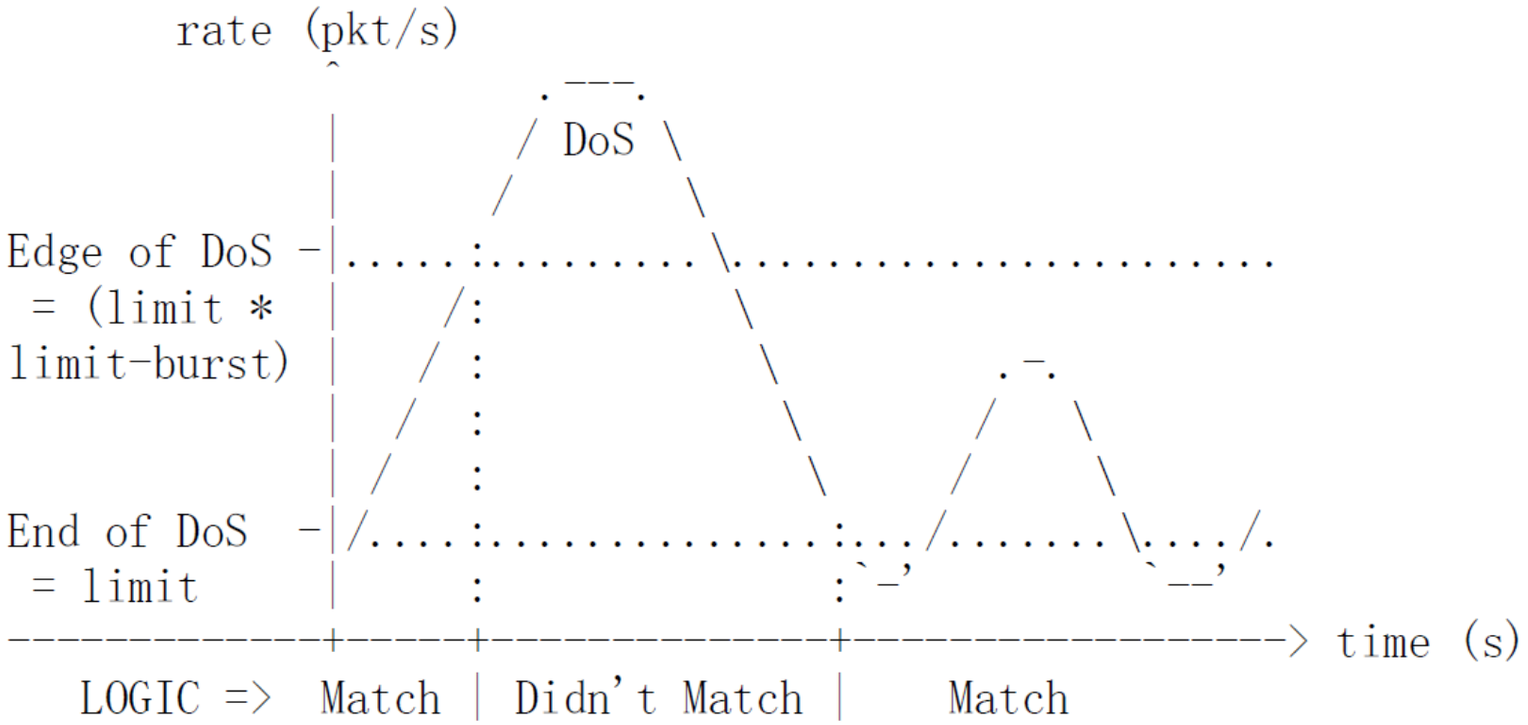
Ping of death:

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

方法：基于攻击特征匹配的原理，准确度高，误报率低，无法检测未知攻击，典型的误用检测技术。

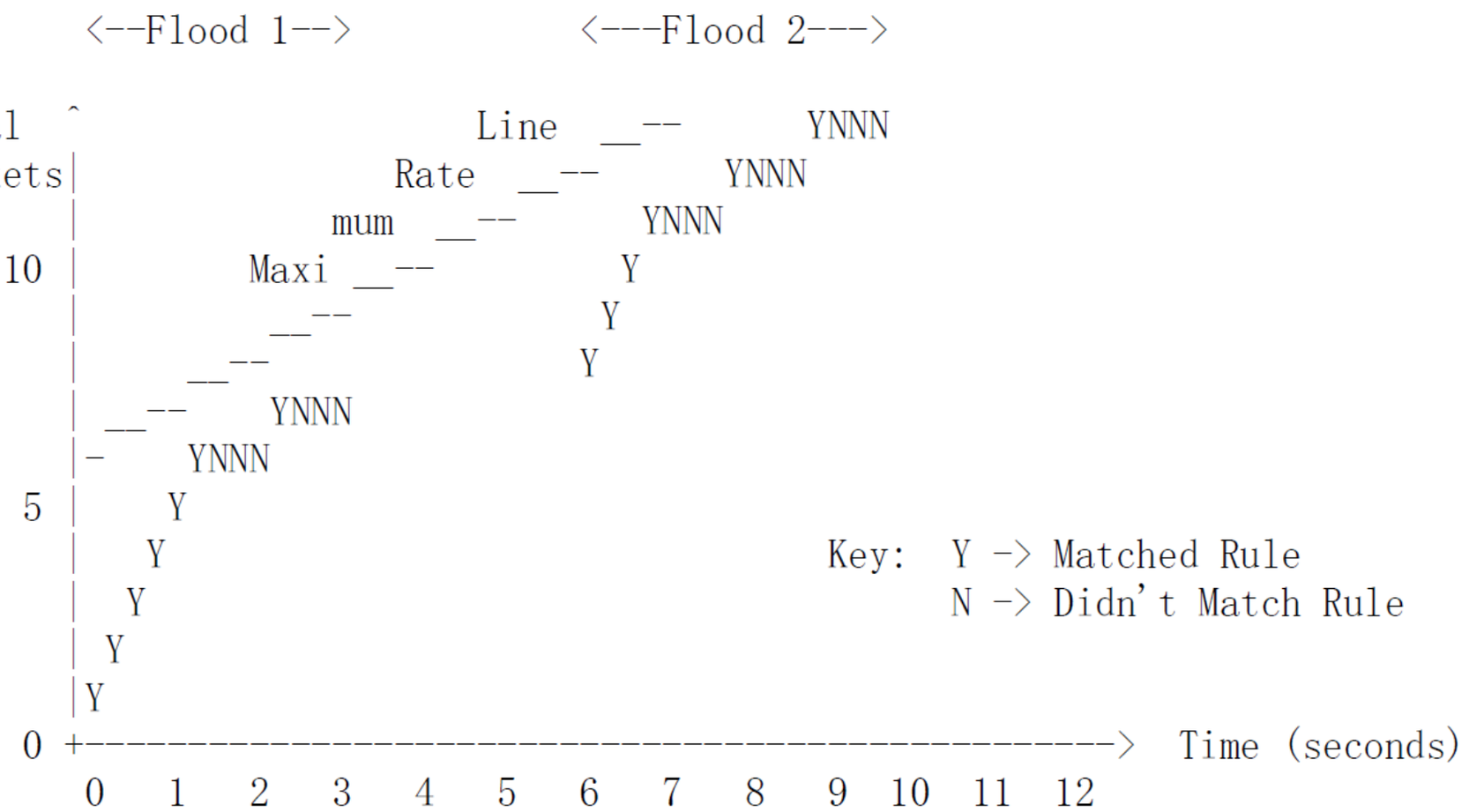
# 控制实例一：iptables

This module works like a "hysteresis door", as shown in the graph below.




Say we say match one packet per second with a five packet burst, but packets start coming in at four per second, for three seconds, then start again in another three seconds.

# 控制实例一：iptables



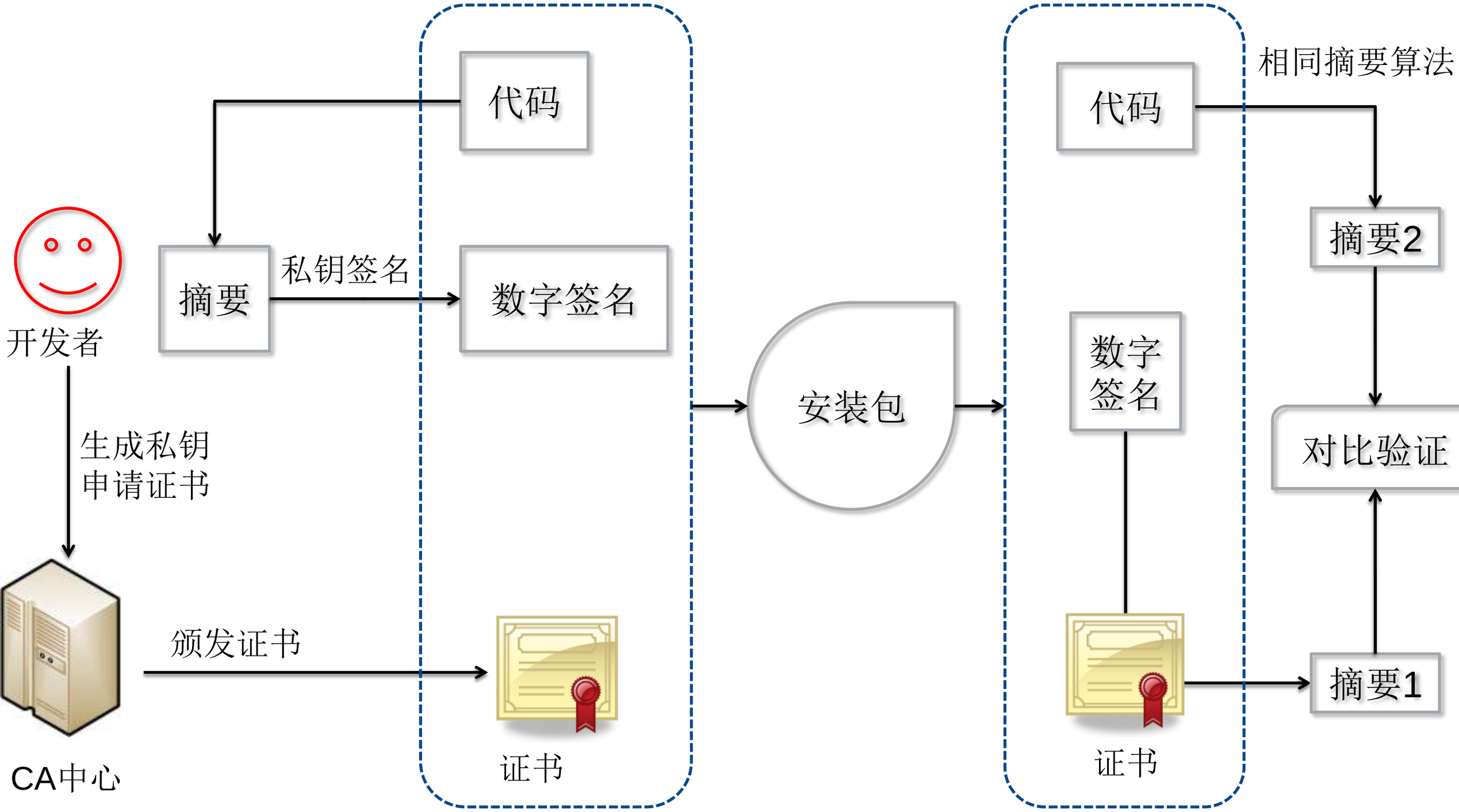

You can see that the first five packets are allowed to exceed the one packet per second, then the limiting kicks in. If there is a pause, another burst is allowed but not past the maximum rate set by the rule (1 packet per second after the burst is used).

# 控制实例二：应用控制

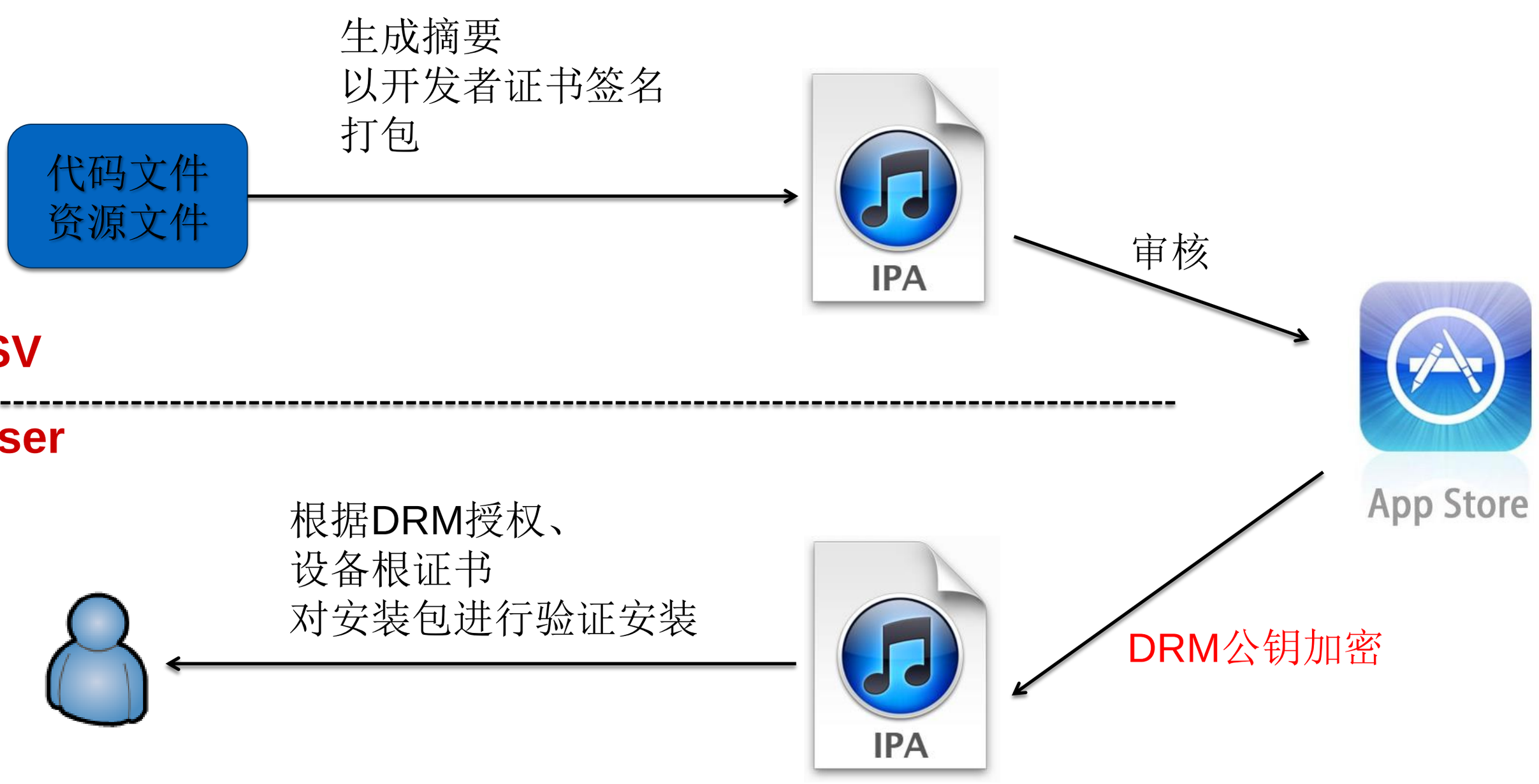
- 手机恶意应用泛滥的重要原因之一就是应用来源问题。
- 可信、可控的应用发布机制是手机系统安全的重要手段。
- 代码签名、DRM（DRM，Digital Rights Management，数字版权管理）是当前常用的控制方法。

**注：应用控制的方法清晰、可行，但考虑到商业、用户习惯等因素，不同系统采取的策略不同。**

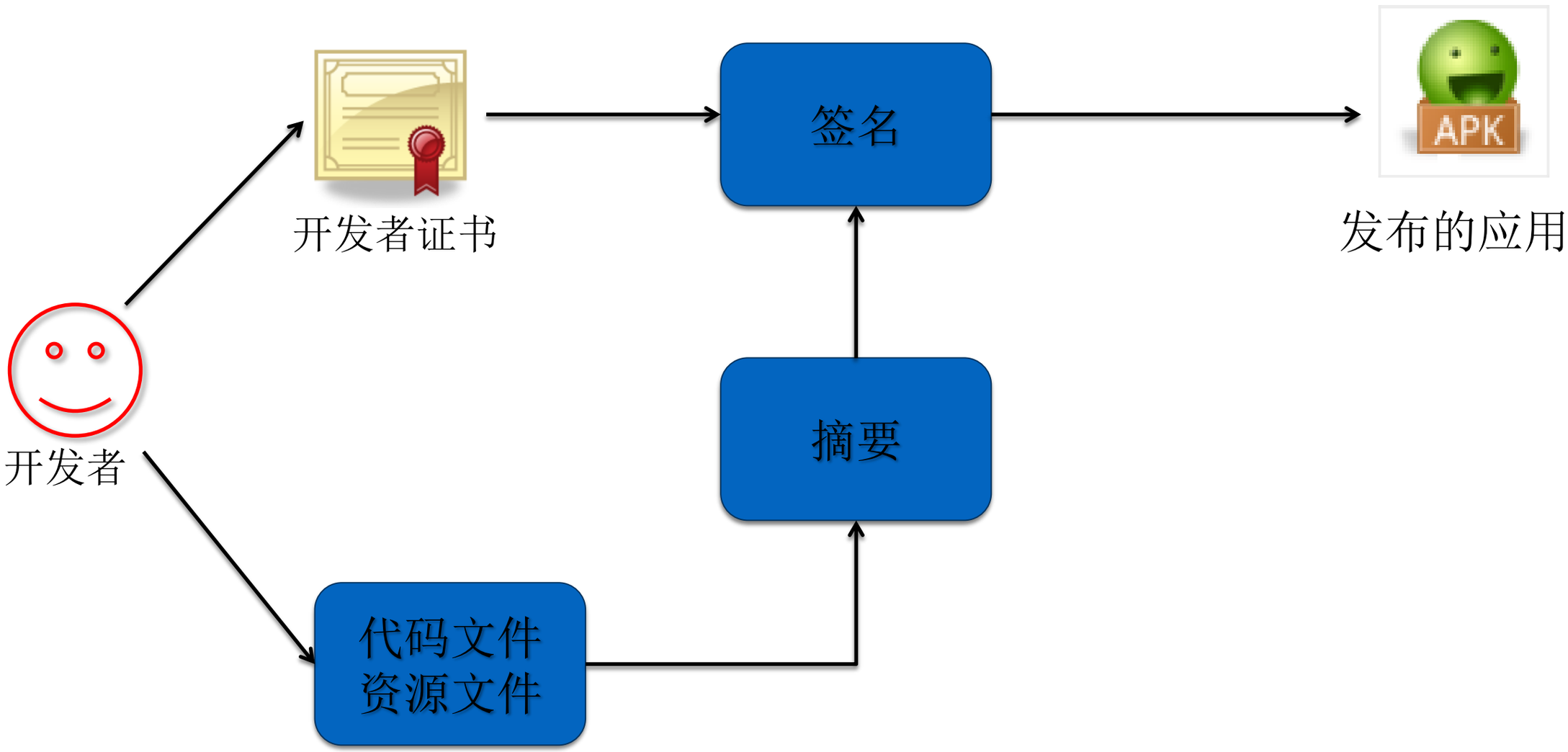
# 代码签名


代码
摘要
私钥签名
数字签名
开发者
生成私钥
申请证书
CA中心
颁发证书
证书
安装包
代码
相同摘要算法
摘要2
数字
签名
对比验证
证书
摘要1

# iOS应用发布流程


生成摘要
以开发者证书签名
打包
代码文件
资源文件
IPA
审核
App Store
ISV
User
根据DRM授权、
设备根证书
对安装包进行验证安装
IPA
DRM公钥加密

# Android签名机制




与其他平台不同，Android平台的开发者可以自行生成开发者证书，不需要通过可信的第三方。

# Windows Phone 应用发布流程

测试
认证
签名

开发应用

.xap安装包

发布

Windows Phone Marketplace

**ISV**

---

**User**

下载

受DRM保护的安装文件

# 控制实例三：单点登陆

- 单点登录（Single Sign On），简称为 SSO，是目前比较流行的企业业务整合的解决方案之一。

- SSO的定义是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。

方法：不同业务系统有各自的身份及认证方法，
将多应用系统的信任集中以实现用户认证的流程简化。

# 控制实例三：单点登陆

- **优势：**
  - 提高用户的效率。用户不再被多次登录困扰，也不需要记住多个 ID 和密码。
  - 提高开发人员的效率。SSO 为开发人员提供了一个通用的身份验证框架。
  - 简化管理。如果应用程序加入了单点登录协议，管理用户帐号的负担就会减轻。
- **问题：**
  - 难以重构。对 SSO 解决方案进行重构来适应现有的应用程序很困难，很耗费时间而且昂贵。
  - 无人看守的桌面。实现 SSO 会减少一些安全风险，但是也增加了其他安全风险。例如，如果用户登录之后离开了他的机器，恶意用户就可以访问他的资源。
  - 单点攻击。在使用单点登录时，所有应用程序使用一个集中的身份验证服务。对于希望实施拒绝服务攻击的黑客，这是一个有吸引力的目标。

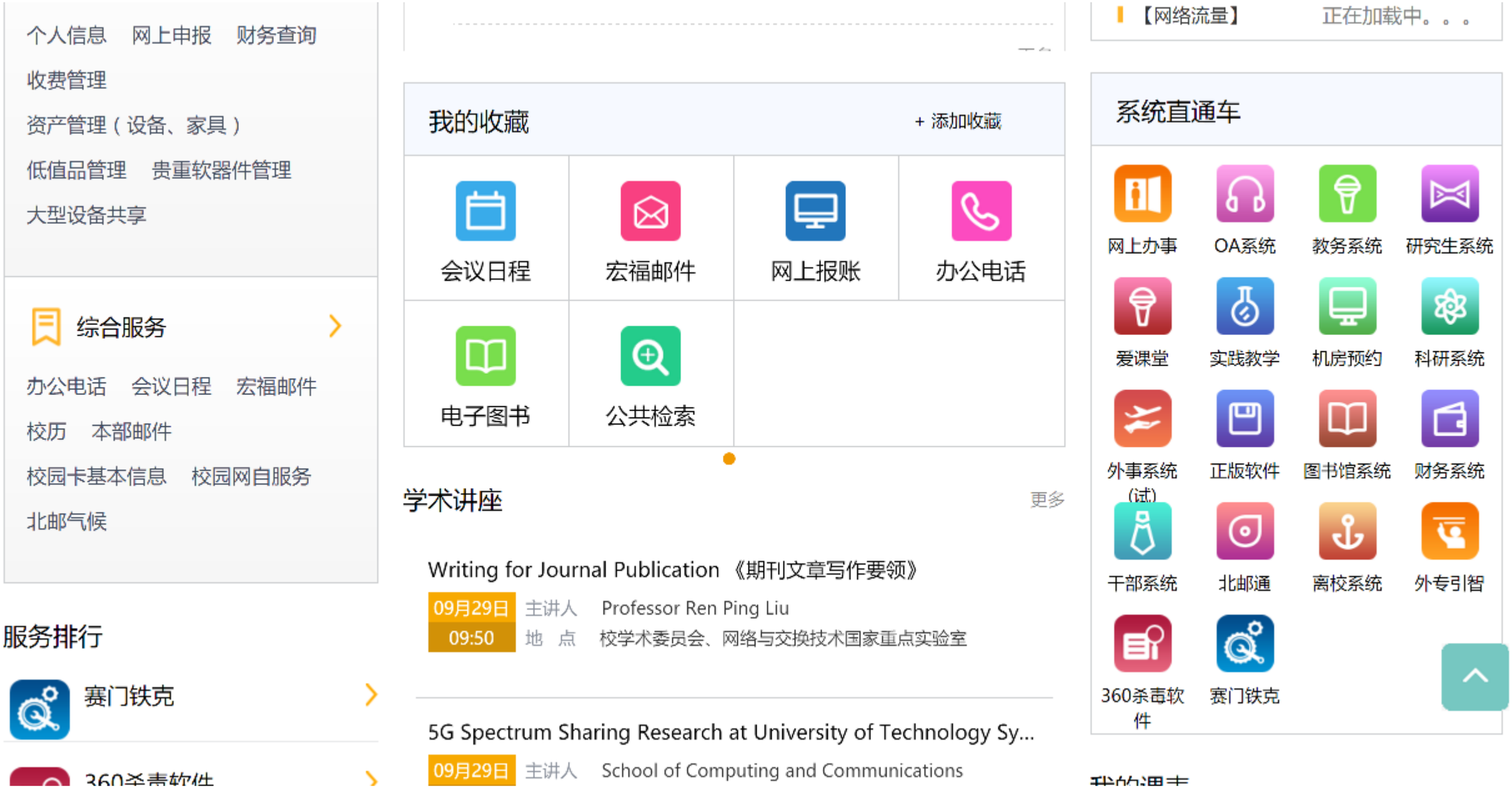
# 控制实例三：单点登陆

- 北邮信息系统为例
  - 教务系统
  - 北邮通
  - 财务系统
  - 图书馆
  - 。。。。

# 控制实例三：单点登陆

- 相关技术应用
  - 身份认证：基于账号、基于身份
  - 权限控制：如教务系统包括教务人员、教师、学生、管理人员
  - 数据库、。。。。

# 控制实例三：单点登陆




- 一次登陆可以访问多类资源

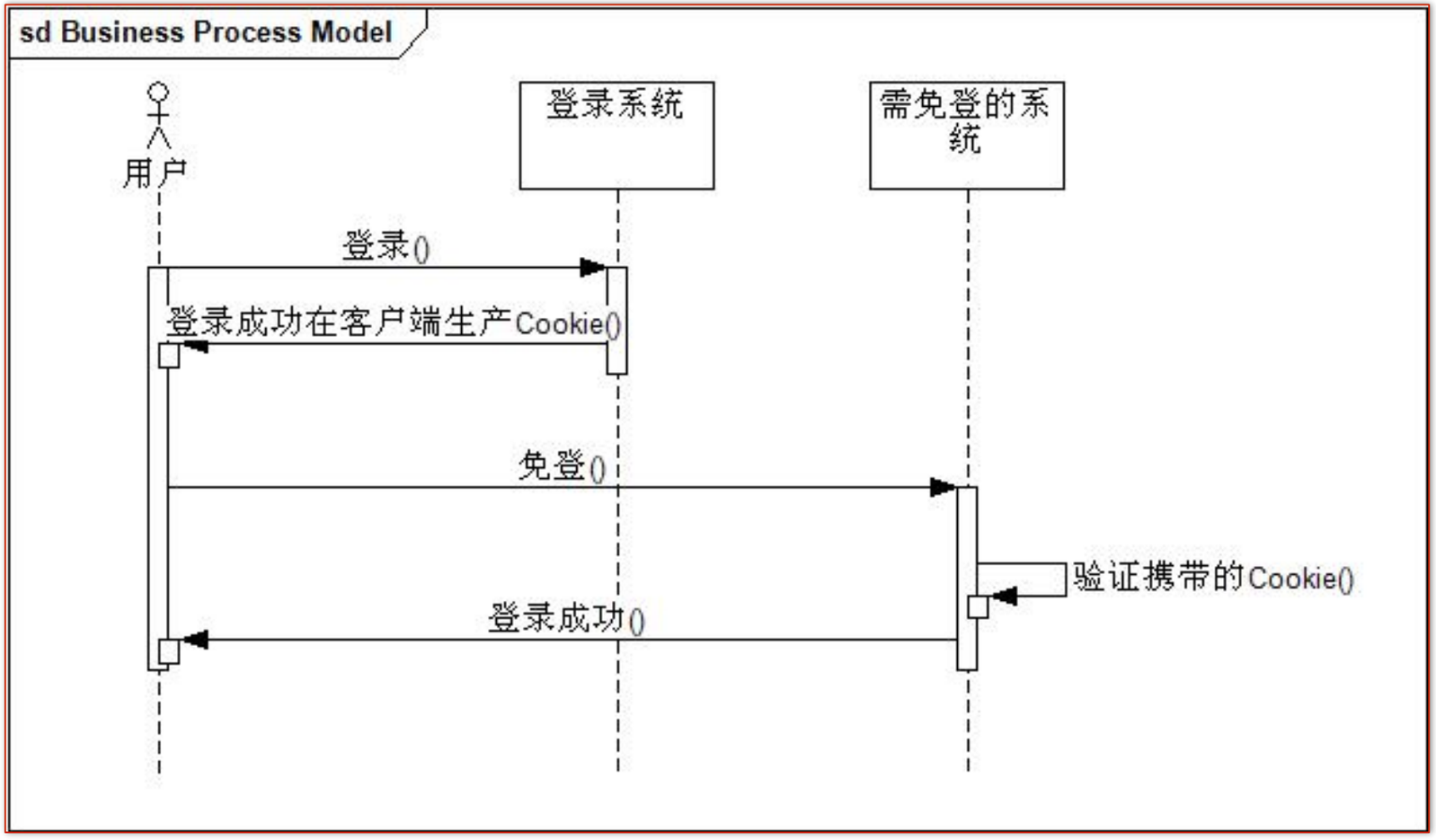
# 控制实例三：单点登陆

- 实现单点登录要解决如何产生和存储信任、如何验证这个信任的有效性，因此要点也就以下两个：
  - 存储信任
  - 验证信任

# 控制实例三：单点登陆

- 以Cookie作为凭证媒介
  - 最简单的单点登录实现方式，是使用cookie作为媒介，存放用户凭证。
  - 用户登录父应用之后，应用返回一个加密的cookie，当用户访问子应用的时候，携带上这个cookie，授权应用解密cookie并进行校验，校验通过则登录当前用户。

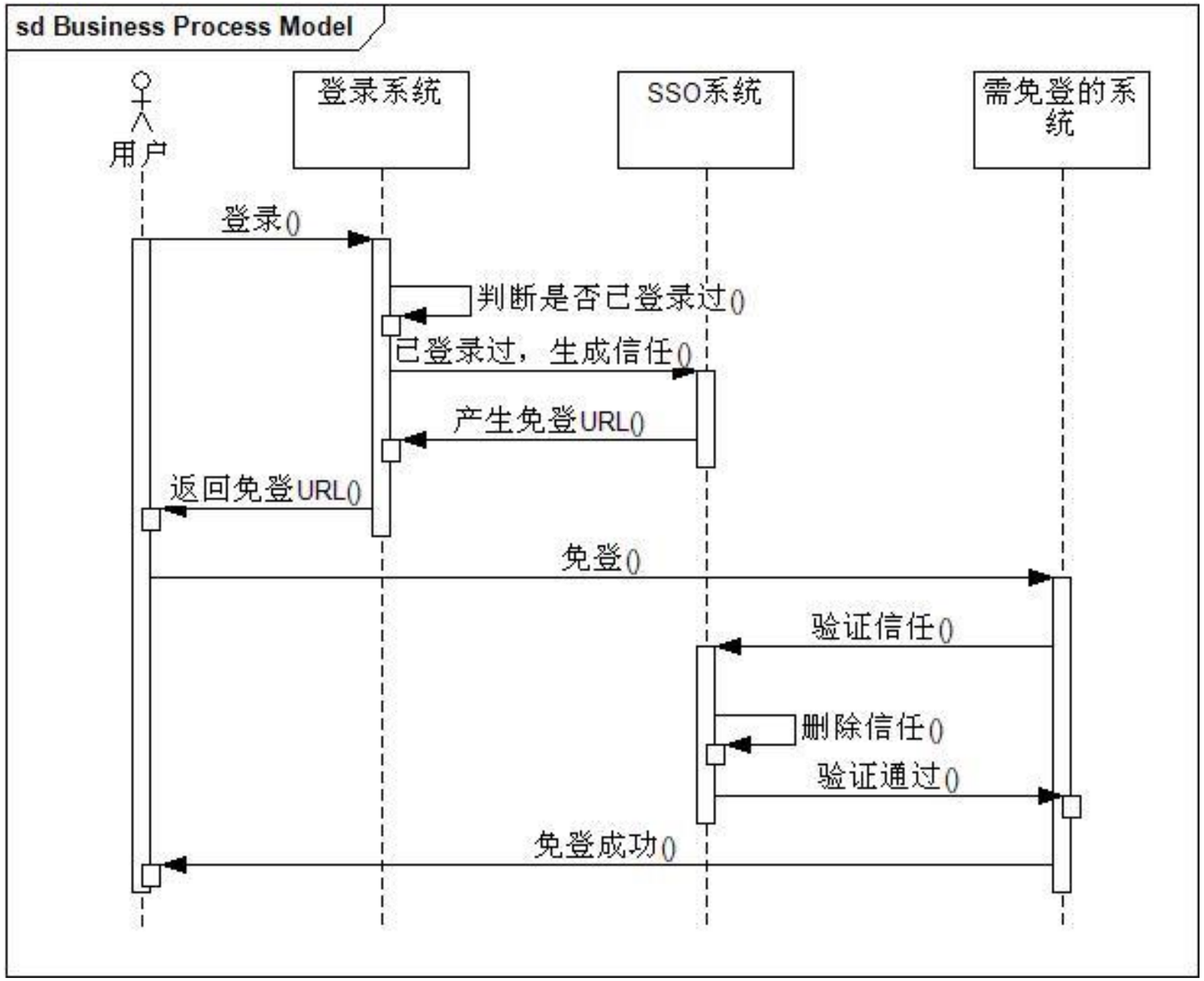
Cookie不安全
不能跨域实现免登




文 / JC_Huang（简书作者）
原文链接：http://www.jianshu.com/p/613e44d4a464

# 控制实例三：单点登陆

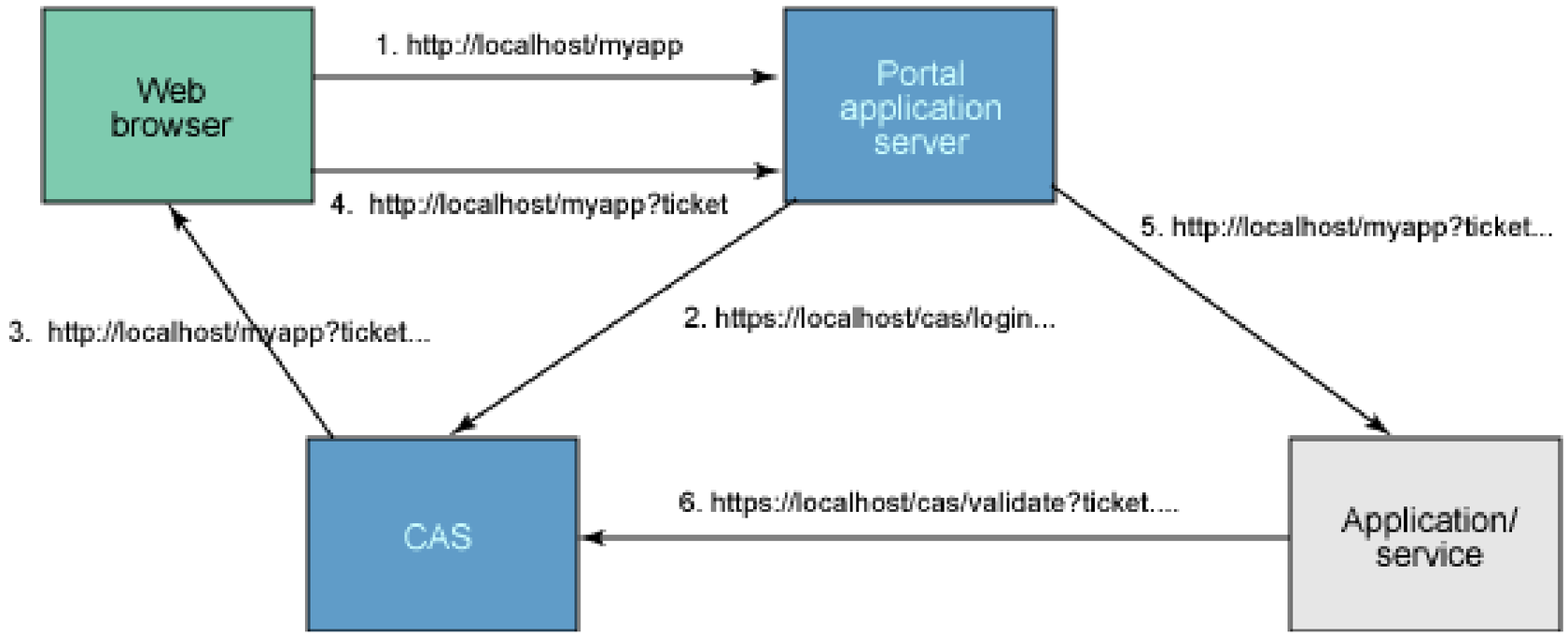
- 通过页面重定向的方式




http://blog.csdn.net/cutesource/article/details/5838693

# CAS（Central Authentication Service）

## Yale University 的 CAS系统




- 用户尝试使用应用程序的 URL 访问应用程序。用户被重定向到 CAS 登录 URL，采用的是 HTTPS 连接，他请求的服务的名称作为参数传递。这时向用户显示一个用户名/密码对话框。
- 用户输入 ID 和密码，CAS 对他进行身份验证。如果身份验证失败，目标应用程序根本不会知道这个用户曾经试图访问它 —— 用户在 CAS 服务器上就被拦住了。
- 如果身份验证成功，CAS 就将用户重定向回目标应用程序，并在 URL 中附加一个称为 ticket 的参数。然后，CAS 尝试创建一个称为 ticket-granting cookie 的内存 cookie。这是为了以后进行自动的重新验证；如果存在这个 cookie，就表示这个用户已经成功地登录了，用户就不需要再次输入他的用户名和密码。
- 然后，应用程序要检查这个 ticket 是否正确，以及是否代表一个有效用户；检查的方法是，打开一个 HTTPS 连接来调用 CAS serviceValidate URL，并作为参数传递 ticket 和服务名称。CAS 检查这个 ticket 是否有效，以及是否与请求的服务相关联。如果检查成功，CAS 就将用户名返回给应用程序。

http://www.ibm.com/developerworks/cn/web/wa-singlesign/

# 控制实例四：边界控制

- 溢出是系统、软件漏洞的重要来源，溢出后的漏洞利用是攻击者利用系统运行原理，实现执行自有代码的常见方法。
- 众多的程序来源，纷杂的编程水平，使溢出难以避免，如何通过系统的限制，减少溢出可能带来的危害，是操作系统重要的安全防护手段。

# ASLR

- Address space layout randomization (ASLR) is a computer security technique involved in preventing exploitation of memory corruption vulnerabilities.

- In order to prevent an attacker from reliably jumping to, for example, a particular exploited function in memory

一种针对缓冲区溢出的安全保护技术，通过对堆、栈、共享库映射等线性区布局的随机化，通过增加攻击者预测目的地址的难度，防止攻击者直接定位攻击代码位置，达到阻止溢出攻击的目的。

ASLR可以有效的降低缓冲区溢出攻击的成功率，Linux、FreeBSD、Windows、IOS等主流操作系统都已采用了该技术。

# DEP

- DEP（Data Execution Prevention，**数据执行保护**），是一套软硬件技术，能够在内存上执行额外检查以帮助防止在系统上运行恶意代码。

- 使用DEP的目的是阻止恶意插入代码的执行，可帮助防止数据当作代码执行，从而有效分离数据与代码。

- 其运行机制是，Windows利用DEP标记只包含数据的内存位置为非可执行(NX)，当应用程序试图从标记为NX的内存位置执行代码时，Windows的DEP逻辑将阻止应用程序这样做，从而达到保护系统防止溢出。

# SafeSEH

- SafeSEH（Image has Safe Exception Handlers），即在调用异常处理函数之前，对要调用的异常处理函数进行一系列的有效性校验，如果发现异常处理函数不可靠（被覆盖了，被篡改了），立即终止异常处理函数的调用。

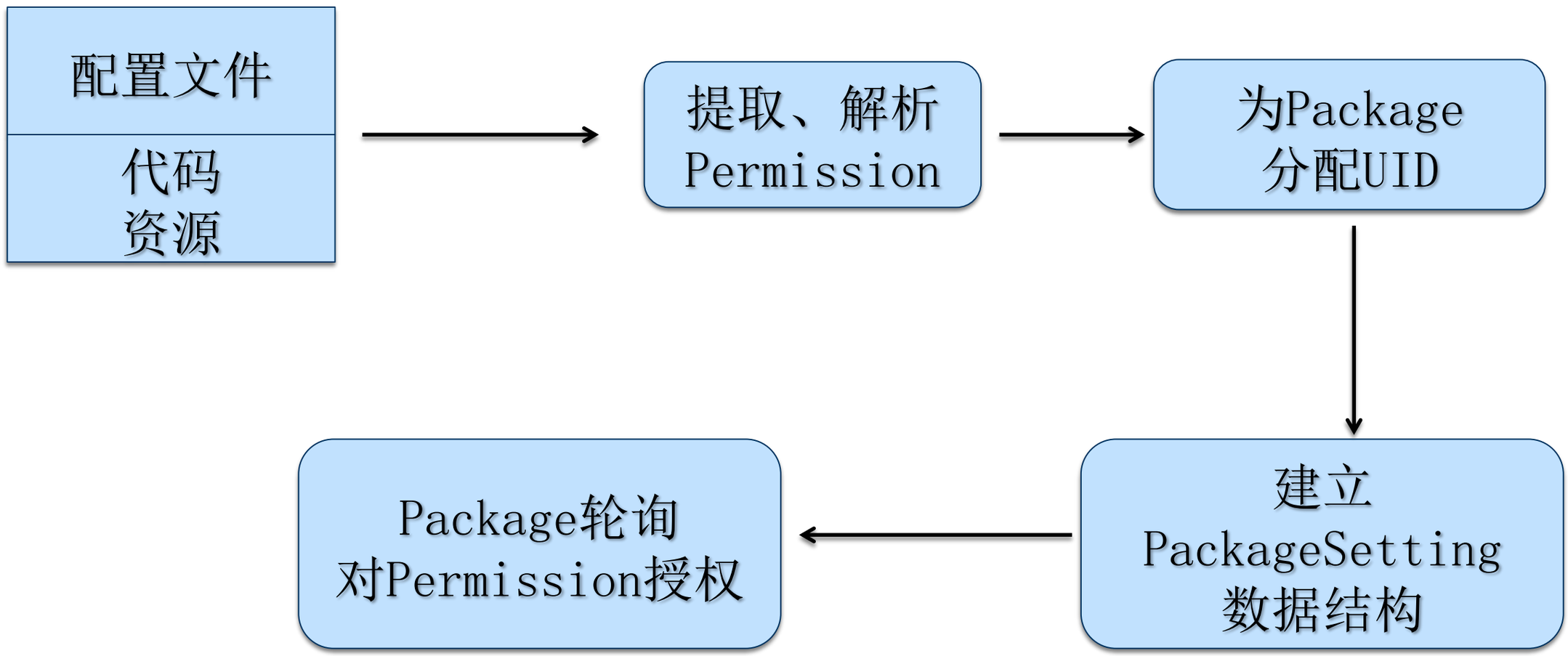
- When /SAFESEH is specified, the linker will only produce an image if it can also produce a table of the image's safe exception handlers. This table specifies for the operating system which exception handlers are valid for the image.

# 控制实例五：手机权限控制

| 值 | 说明 |
| --- | --- |
| android.permission.INTERNET | 访问网络 |
| android.permission.ACCESS_FINE_LOCATION | 通过GPS获取位置 |
| android.permission.BLUETOOTH | 使用蓝牙 |
| android.permission.CALL_PHONE | 拨打电话 |
| android.permission.PROCESS_OUTGOING_CALLS | 获得打出电话状态 |
| android.permission.READ_CONTACTS | 读取联系人 |
| android.permission.READ_SMS | 读取短信 |
| android.permission.RECEIVE_SMS | 接收短信 |
| android.permission.SEND_SMS | 发送短信 |

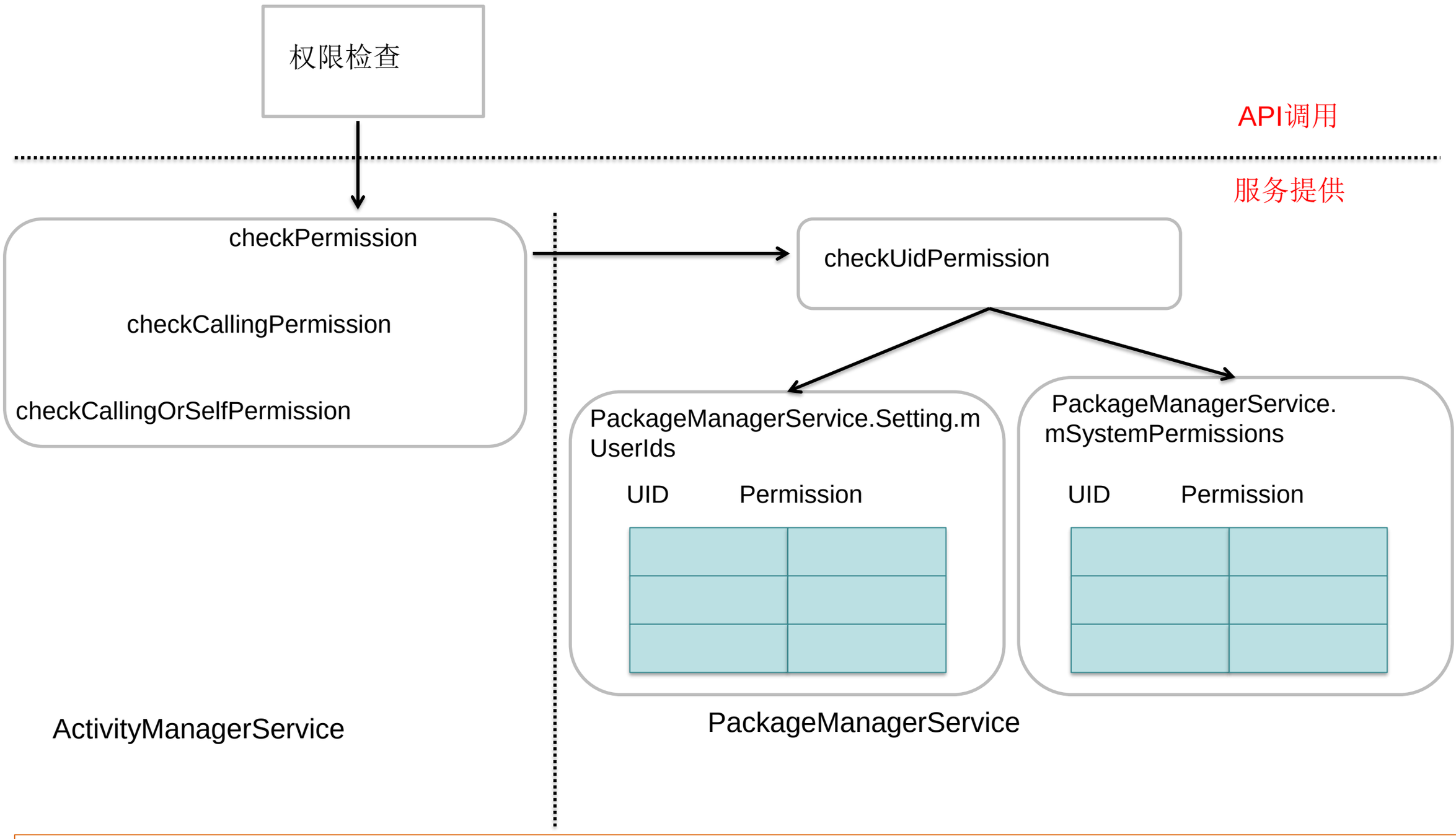
**Android应用权限管理，常见的Permission。**

# 控制实例五：手机权限控制




Android安装过程中的权限授予

# 控制实例五：手机权限控制




Android权限检查流程

# 问题和讨论