分布计算环境

北京邮电大学计算机学院

Chapter 2

分布式系统的基本原理

主要内容

- 体系结构
- 进程
- 通信
- 命名
- 一致性和复制
- 容错
- 安全

容错主要内容

- 分布式容错模型
- 可靠的客户服务器通信
- 进程的恢复
- 可靠的分组通信
- 分布式提交
- 恢复处理

基本概念（1）

- 可靠/可信系统（Dependable, Trustworthy）
  - 可用性（availability）
    - 在任何给定时刻能正确操作的概率
    - 可用性= (平均正常工作时间/ (平均正常工作时间+ 平均修复时间))
  - 可靠性（Reliablity）
    - 在给定时间间隔内能正确操作的概率
  - 安全性（Safty）
    - 含义1：临时失效不会造成灾难
    - 含义2：系统或数据不会被破坏、更改、泄露
  - 可维护性（Maintainability）
    - 系统发生故障后进行修复的难易程度

基本概念（2）

- 失效（fail, failure）、失败
  - 一个系统不能兑现它的承诺（提供服务）
- 差错（error）：
  - 导致系统失效的系统状态
    - ➔如一些数据包在到达接收方时与原来的不一致了
- 故障（fault）：
  - 导致差错发生的原因
    - ➔如传输介质坏了
    - ➔如无线传输时的恶劣天气

基本概念（3）

- 故障类型
  - 短暂型(transient):出现一次，再也不出现
    - ➔例：一只鸟从微波电波中飞过
  - 间歇型(intermittent)：消失后，再重复出现
    - ➔例：接触不良
  - 永久型(permanent)：一直存在
    - ➔例：文件损坏
- 容错（fault tolerance）
  - 即使发生故障，系统仍能提供服务

失效（失败）模型

失效类型	描述
崩溃性失效	服务器停止。但在停止前一直正确工作
遗漏性失效 　接收遗漏 　发送遗漏	服务器不能响应到来的请求 服务器不能接收到来的消息 服务器不能发送消息
定时性失效	服务器的响应超出规定的时间间隔
响应性失效 　值失效 　状态变迁失效	服务器的响应不正确 响应的值是错误的 服务器偏离正确的控制流
任意性失效	服务器在任意的时刻产生任意的响应

其它相关术语

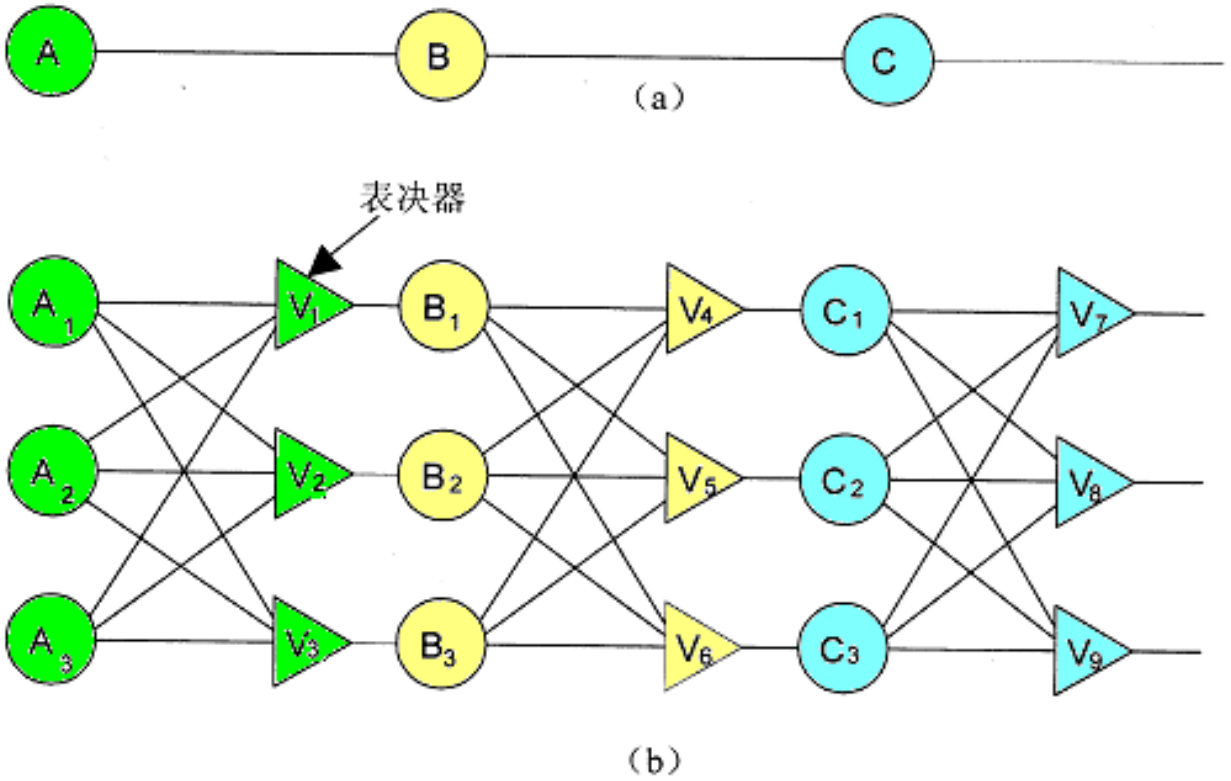
- 失校即停 (fail-stop)：一般指崩溃性失效。不通知失效。良性的、易检测。
- 失效沉默(fail-silent)：可能是崩溃，也可能是性能故障引起。不通知失效
- 失效安全(fail-safe)：产生不确定响应，但响应易被识别，不产生恶果。
- 拜占庭(Byzantine)失效：恶意的、难检测。
  - ➔ 拜占庭帝国（330-1453）

基于冗余的失效屏蔽技术

- 冗余类型
  - 信息冗余：如，海明码
  - 时间冗余：如，重发，重做
  - 物理冗余：
    - 软件:如复制进程
    - 硬件：如复制电路、网卡
  - 信息冗余和物理冗余都属于空间冗余

基于冗余的失效屏蔽技术（续）

- 三模冗余方法(TMR,Triple Modular Redundancy)
  - 三路表决器(voter)：三路输入，一路输出
  - 可屏蔽一路错误（任意性失效）



主要内容

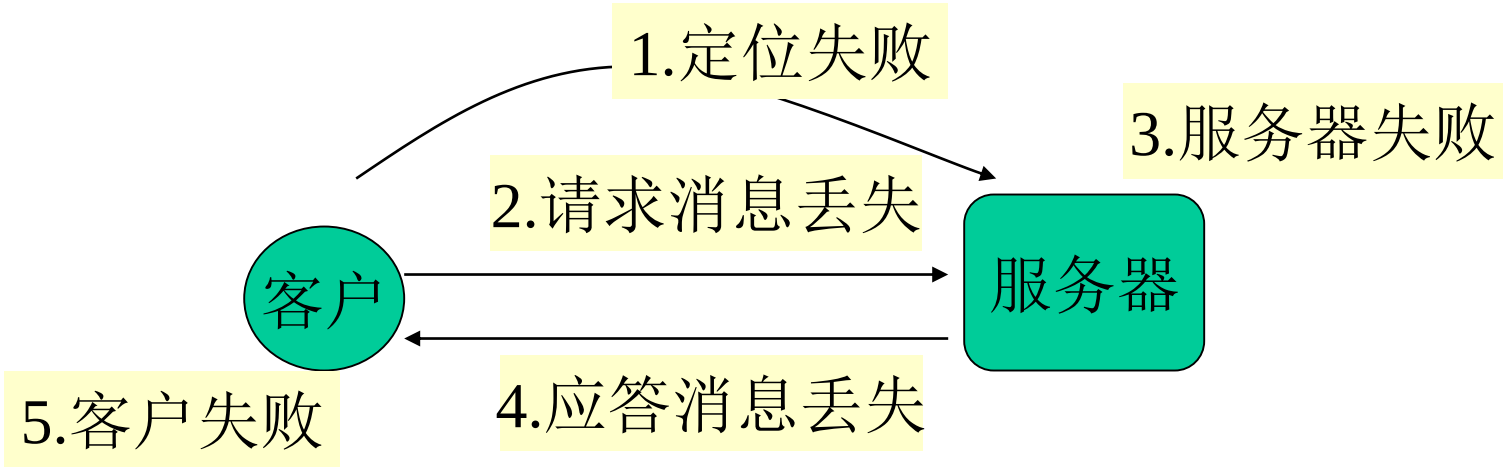
- 分布式容错模型
- 可靠的客户服务器通信
- 进程的恢复
- 可靠的分组通信
- 分布式提交
- 恢复处理

可靠的客户服务器通信

- 点到点通信
  - 可靠通信：防止通信失效
  - 遗漏型失效：消息丢失
    - ➔解决策略：利用可靠的传输协议，如TCP协议。确认和重新传输
  - 连接崩溃失效：连接中断
    - ➔不能屏蔽，需重建连接
    - ➔解决策略：抛出异常，通知客户进程

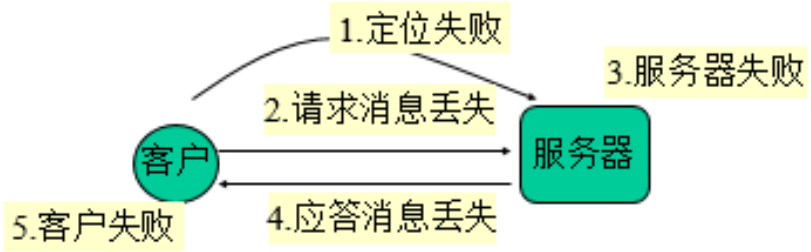
RPC失效(1)

- RPC失效
  - 5种失效情况



RPC失效(2)

- 1、客户不能定位服务器
  - 可能服务器发生故障
  - 可能服务器被修改，客户存根（stub）与新的服务器存根不匹配
  - 解决策略：
    - 抛出异常信号SIG-NOSERVER，然后由编写的信号处理程序做相应处理。
    - 没有透明性：需要编写异常处理程序

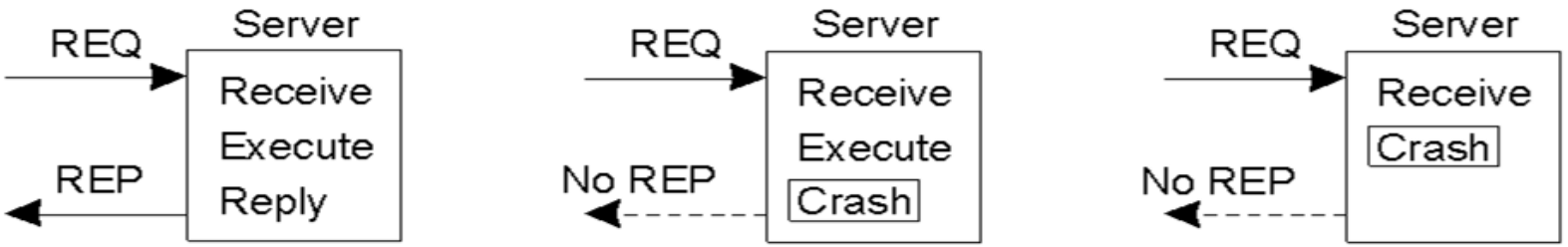


- 2、丢失请求消息
  - 解决策略：
    - 客户发现超时，重发请求
    - 频繁丢失，一般认为“不能定位服务器”

RPC失效(3)

3、服务器崩溃

- 崩溃情况



（a）正常情况；　（b）在执行后崩溃　（c）在执行前崩溃

- 解决策略：
  - 至少一次语义：重发
  - 最多一次语义：丢弃
  - 听之任之，什么都不保证
  - 恰好一次语义：很难：客户端无法知道什么情况下崩溃的

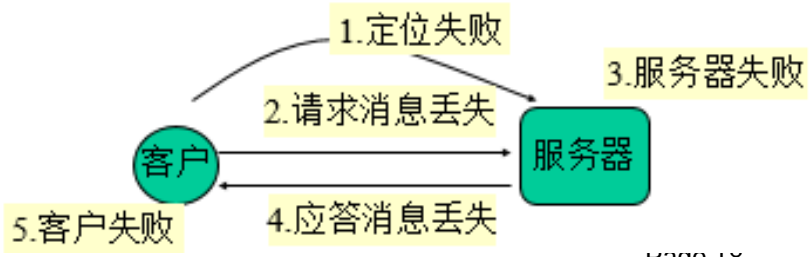
RPC失效(4)

- 举例：打印文本。在打印服务器失效时，客户和服务器的策略组合
  - M(发送完成消息，即下表的ACK)；P(打印)；C(崩溃)

客户	服务器					
	策略 M -> P			策略 P -> M		
重发策略	MPC	MC(P)	C(MP)	PMC	PC(M)	C(PM)
总是重发请求	DUP	OK	OK	DUP	DUP	OK
不重发	OK	ϕ	ϕ	OK	OK	ϕ
当收到ACK时，重发	DUP	OK	ϕ	DUP	OK	ϕ
当没收到ACK时，重发	OK	ϕ	OK	OK	DUP	OK

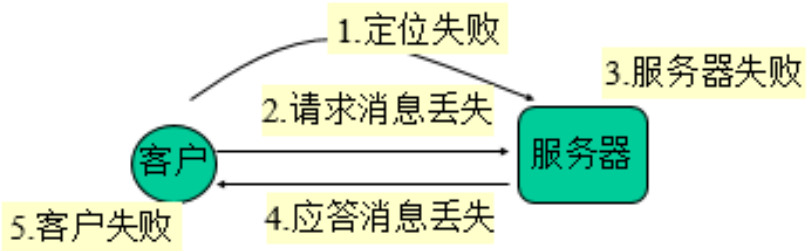
RPC失效(5)

- 4. 丢失应答消息
  - 解决策略
    - ➔定时器，超时检测，重发请求
- 问题：重复操作
  - 解决策略
    - ➔构造幂等性操作（idempotent）：可以安全地重复多次而不会造成任何损害的操作
    - ➔非幂等操作
      - 序号
      - 标志：区分原始消息和重发消息



RPC失效(6)

- 5. 客户崩溃
  - 孤儿进程问题：没有关注该进程计算结果的进程存在。
    - ➔造成资源浪费；造成结果混淆
  - 解决策略：
    - ➔过期法：设置时间量T。如果超过T，则撤销客户的请求（不一定是孤儿）
    - ➔……



主要内容

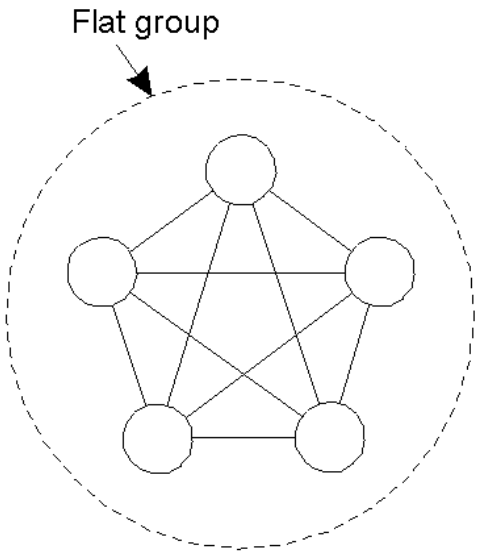
- 分布式容错模型
- 可靠的客户服务器通信
- 进程的恢复
- 可靠的分组通信
- 分布式提交
- 恢复处理

进程恢复：平等组与等级组

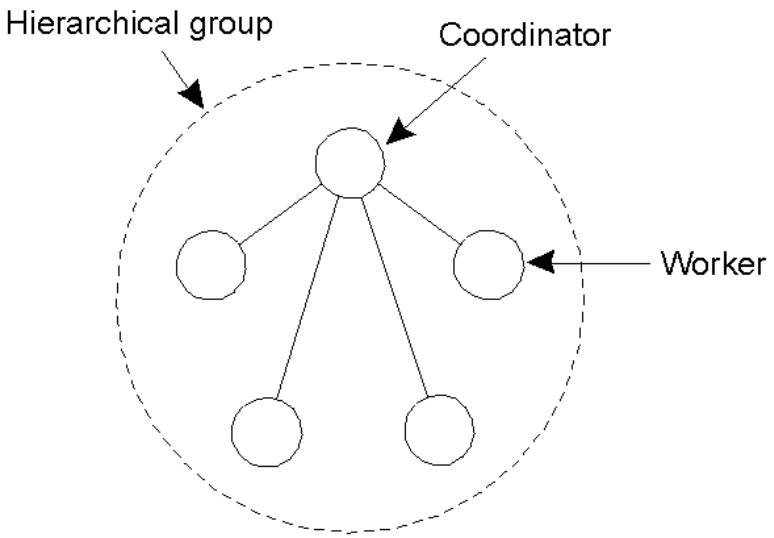
- 为防止进程失败，把进程复制到组
- 当消息发送到组时，组中所有成员都接收它，一个进程失败，其他进程可以接管它
- 进程组是动态的

a) 平等组：无单点失效；增加延迟和开销

b) 简单等级组：单点失效；



(a)



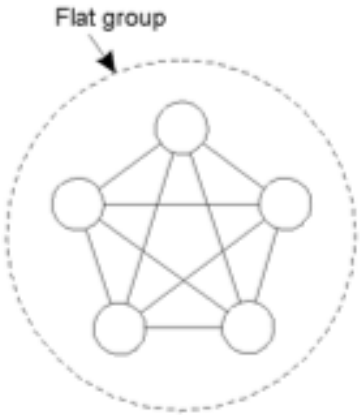
(b)

故障掩盖和复制

- 复制进程，用一个容错的进程组来代替一个脆弱的进程
- 需要多少复制?
  - 如果系统能经受K个组件的故障而且能满足规范的要求,被称为K容错的
  - 如果组件是失败沉默的，具有K+1个组件即可
  - 如果组件发生拜占庭失效（ Byzantine failure ），继续错误运行，则至少需要2K+1个组件才能获得K容错
  - 拜占庭失效：在非失败沉默模型下，一个有故障的进程可能会对其它进程发出干扰消息，从而影响这些进程的正常工作
    - ➔拜占庭失效是所有失效类型中最严重的

故障系统的协定问题

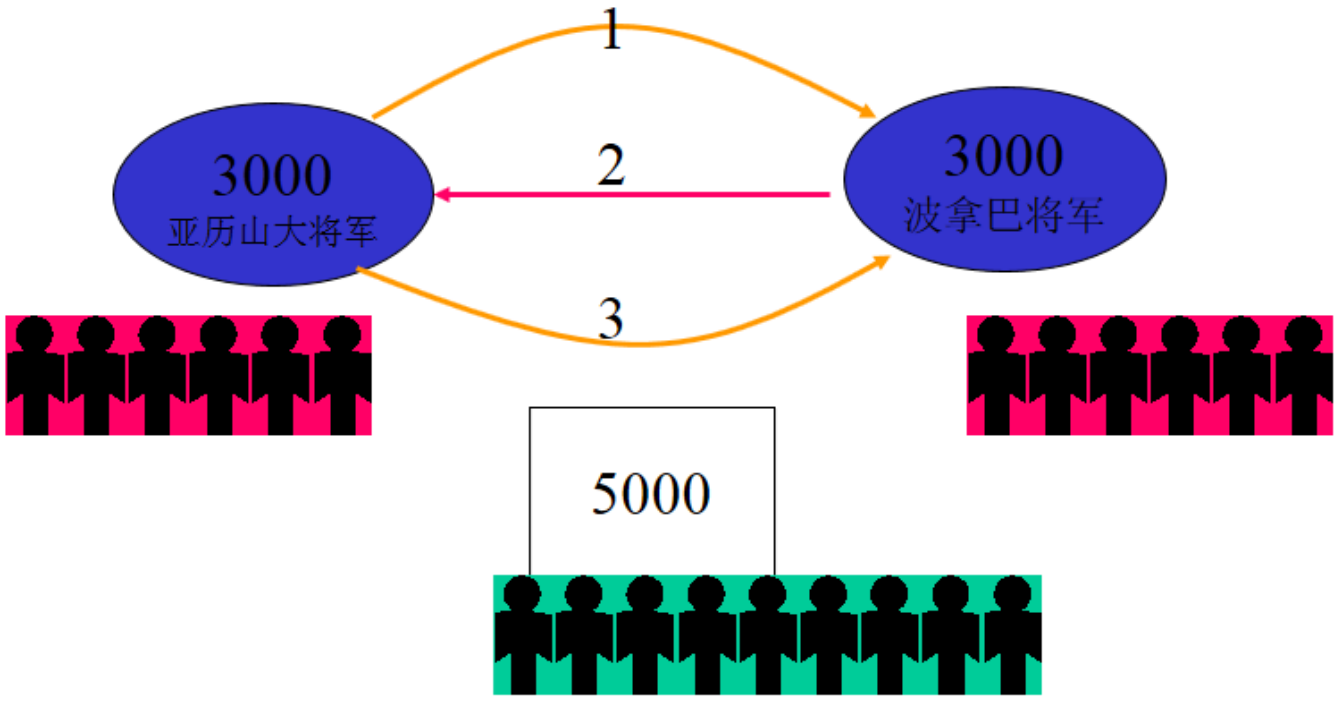
- 协定(共识，agreement)
  - 对某些问题的一致意见。如：选择一个协调者，是否提交事务，在工作者之间划分任务
- 分布式协定算法
  - 在有限的步骤内，所有非故障进程达成协定（共识）



什么情况下可达成协定呢？

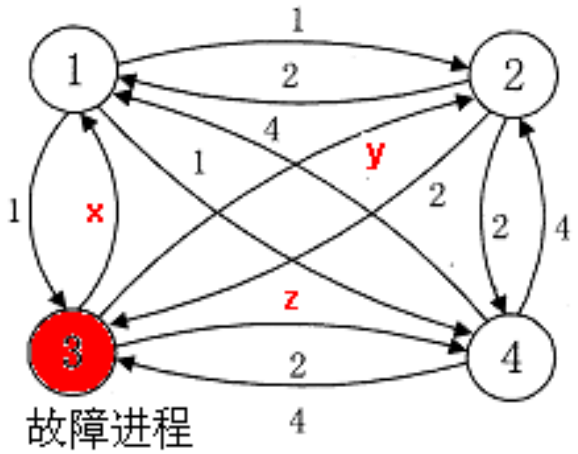
分布式协定与通信

- 对于不可靠通信，即使进程是可靠的，也不可能达成协定
- 两军问题（two-army）
  - 3000+3000 > 5000



分布式协定与进程

- 拜占庭将军协定问题
  - 假设通信是可靠的，但进程可能是不可靠的
    - ➔3个忠诚将军，1个叛变将军
  - 不管叛徒做什么，算法必须保证所有忠诚的将军达成相同的行动计划：完成协定，达成共识
- Lamport递归算法
  - 共4步：(a)对外报告(b)收集向量(c)报告向量(d)生成结果向量：（1，2，未知，4）



(a)

1 Got(1, 2, x, 4)
2 Got(1, 2, y, 4)
3 Got(1, 2, 3, 4)
4 Got(1, 2, z, 4)

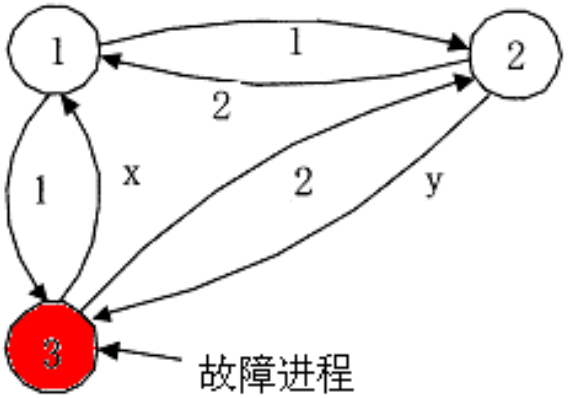
(b)

1 Got	(1, 2, y, 4)
	(a, b, c, d)
	(1, 2, z, 4)
2 Got	(1, 2, x, 4)
	(e, f, g, h)
	(1, 2, z, 4)
4 Got	(1, 2, x, 4)
	(1, 2, y, 4)
	(i, j, k, l)

(c)

拜占庭将军问题：2忠1叛

- 若三个将军中，有两个忠诚将军，一个叛变将军，则不能判断出哪个将军叛变。



(a)

1 Got(1, 2, x)
2 Got(1, 2, y)
3 Got(1, 2, 3)

(b)

1 Got (1, 2, y)
(a, b, c)
2 Got (1, 2, x)
(d, e, f)

(c)

- 若要有m个进程出错的系统实现协同一致，最少要有2m＋1个正常进程。进程总数为3m+1。
- 需超过2/3多数，才能达成协定（共识）

故障检测

- 进程故障检测
  - 主动式方法，发送 “Are you alive?”消息
    - ➔常用方法， ping操作
  - 被动式方法，等待发来的故障消息
  - 超时机制，在规定时间内作出响应，否则，为故障

主要内容

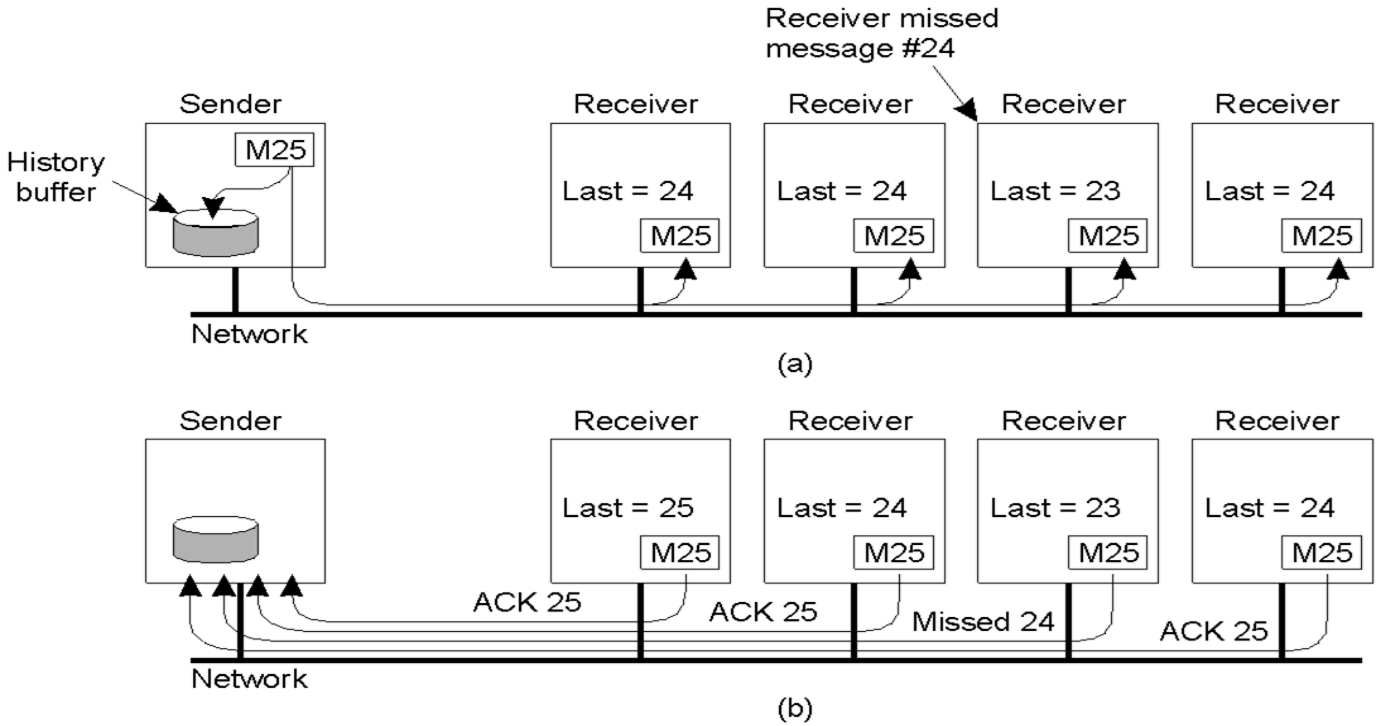
- 分布式容错模型
- 可靠的客户服务器通信
- 进程的恢复
- 可靠的分组通信
- 分布式提交
- 恢复处理

可靠的分组通信

- 可靠多播:发送到一个进程组的消息被传递到该组的每个成员
- 基本的可靠多播方法：假定所有的接收者已知而且假定不会失败的简单可靠多播方法
  - ➔接收方数量不多
- 可靠多播的可扩展性：接收方数量比较多
- 原子多播：实现存在进程失败的情况下的可靠多播：保证所有正常组员都按照相同的顺序接收到消息
  - ➔如果通信期间有进程加入
  - ➔如果通信期间一个（发送）进程崩溃

例：基本的可靠多播模式

- 假定所有的接收者已知而且假定不会失败的简单可靠多播方法
  a) 消息传播：记录顺序号
  b) 报告反馈：如果丢失，返回NACK，重新发送

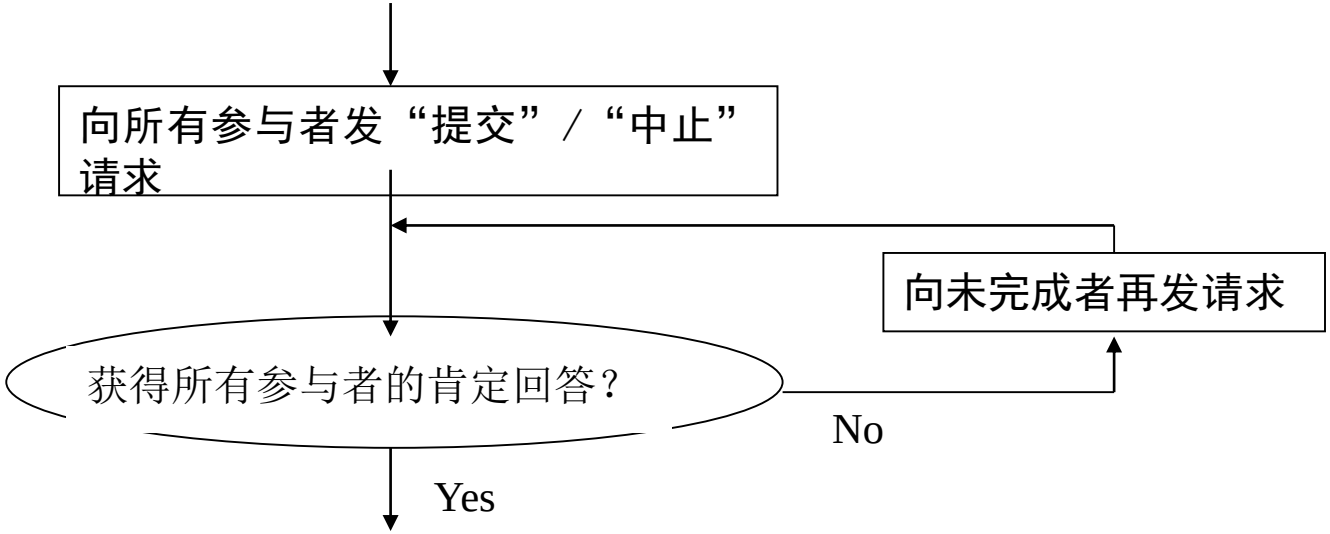


主要内容

- 分布式容错模型
- 可靠的客户服务器通信
- 进程的恢复
- 可靠的分组通信
- 分布式提交
- 恢复处理

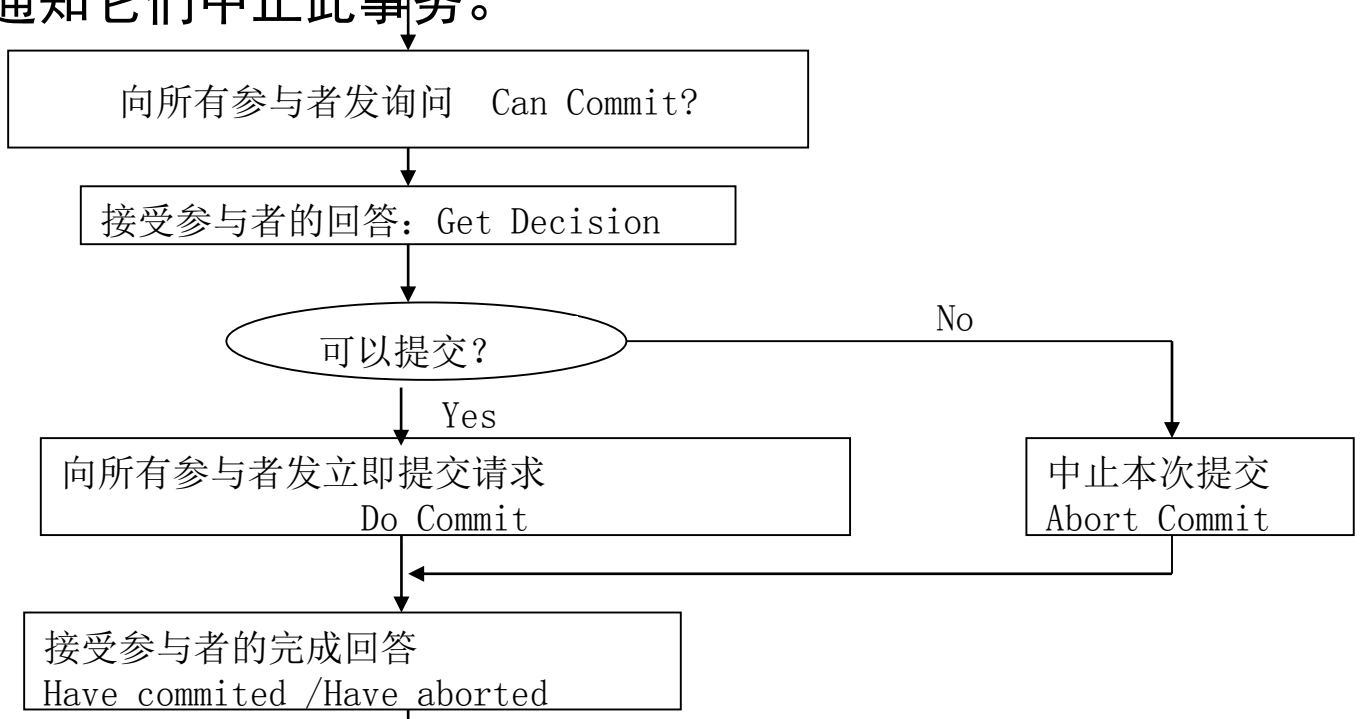
单阶段提交

- 分布式提交：具有原子性：要使一个操作被进程组中每一个进程都执行或都不执行。通常使用协调者
- 单阶段提交中，协调者向事务所有的参与者发出提交或者中止请求，并不断重复该请求，直到所有参与者报告它们执行了该请求
- 问题：如果遇到某一问题，导致某一参与者无法完成提交，本协议就无法实现



两阶段提交2PC

- 简单、实用、可靠，成为事实上的工业标准。将提交分成两个阶段：
- 第一阶段（表决阶段），事务的协调者询问各个参与者是否可以提交，此时，各个参与者将回答消息发给协调者。协调者根据收到的消息，看是否可以真正提交
- 第二阶段（完成阶段），如果可以提交，则通知各参与者立即执行提交，否则，通知它们中止此事务。



主要内容

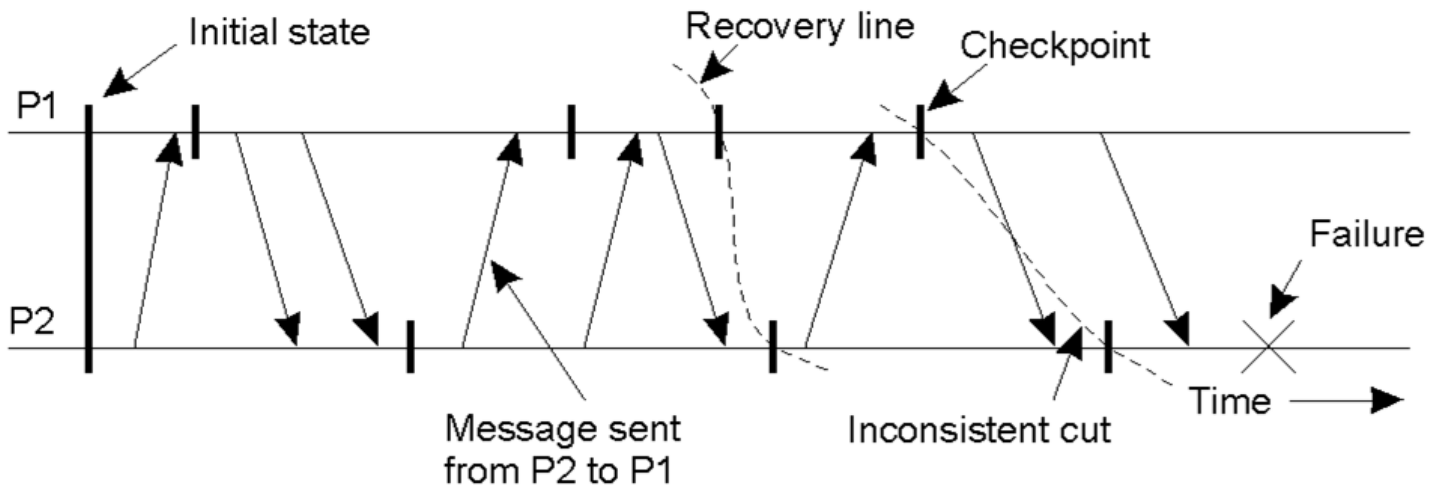
- 分布式容错模型
- 可靠的客户服务器通信
- 进程的恢复
- 可靠的分组通信
- 分布式提交
- 恢复处理

恢复处理

- 目的: 使系统从错误状态到正确状态
- 类型：
  - 向后恢复(backward recovery)：使系统返回到上一个正确状态：需要在必要时记录系统状态
    - ➔恢复状态一般开销较大
    - ➔恢复后不能保证不发生类似错误
  - 向前恢复(forward recovery)：使系统前进到一个正确的新状态：需要预知会发生什么错误
- 检查点技术(checkpoint)
- 消息日志技术（logging）
  - 基于发送者也可基于接收者写日志

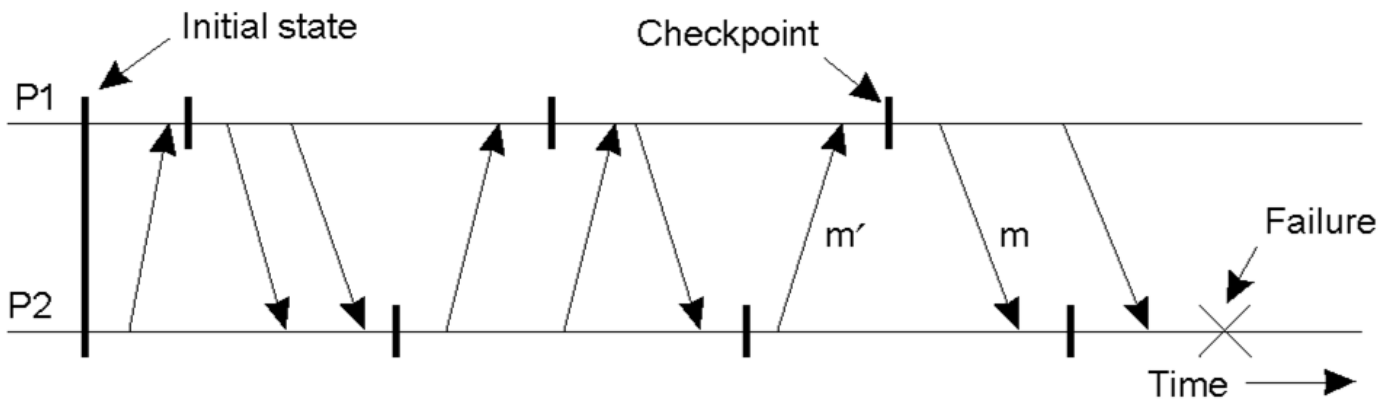
检查点

- 每次记录系统的当前状态时，称为设置一个检查点
- 分布式快照（snapshot）
  - 一致的全局状态：若P收Q发，如果进程P记录了一条消息的接收，那么Q就要记录该条消息的发送
- 恢复线路
  - 最近的分布式快照，最近的一致性割集（cut）



独立检查点

- 独立检查点
  - 每个进程的检查点是相互独立的
- 问题：多米诺效应
  - 局部状态没有形成分布式快照，导致级联回滚（cascaded rollback）过程
  - 举例：只有m,m'的接受记录，没有发送记录
- 有相应解决算法



协作式检查点

- 非阻塞式算法
  - 有相应的分布式快照算法
- 两阶段阻塞式算法
  - CHECKPOINT_REQUEST：
    - ➔协调者发送命令，所有进程写局部检查点，向协调者返回ACK消息
  - CHECKPOINT_DONE：
    - ➔当协调者收到所有的ACK后，发送命令，所有进程继续

消息日志

- 基本思想
  - 减少检查点的个数
  - 如果消息的传送可以重放(replay)，则可取得全局一致性状态，而不必从稳存恢复
- 分段确定性模型(piecewise deterministic model)，假定：
  - 每个进程在一连串的间隔中执行，有先后次序，是确定性的。每个间隔是可重放的
  - 每个间隔以一个非确定性事件开始（如一个消息的接收），以下一个非确定形事件发生而结束。
  - 如果以相同的非确定性事件作为开始进行重放，那么一个间隔的重放可以具有确定的结果
  - 如果记录所有非确定性事件，则全部执行可重放。

容错小结

- 故障模型：故障类型、三模容错
- 客户服务器容错问题：RPC
- 进程容错技术：冗余容错、两军问题、拜占庭将军问题
- 组通信容错问题：可靠多播
- 分布式提交：两阶段提交
- 恢复方法：分布式日志、消息日志

主要内容

- 体系结构
- 进程
- 通信
- 命名
- 一致性和复制
- 容错
- 安全

主要内容

- 安全性概述
- 安全通道
- 访问控制
- 安全性管理

主要内容

- 安全性概述
  - 安全威胁、安全机制
  - 设计问题
  - 加密
- 安全通道
- 访问控制
- 安全性管理

基本概念

- 一个安全的计算机系统，必须具有以下两个属性：
  - 机密性(Confidentiality)：指计算机系统的一种属性，系统凭借此属性使得信息只向授权用户公开
  - 完整性(Integrity)：指对系统资源的变更只能以授权的方式进行，变更包括
    - ➔不适当的变更应该是可以察觉的并可以恢复的

安全性威胁类型

- 考虑计算机系统中安全性的另一角度是保护该系统所提供的数据和服务不受到安全威胁（Security Threat）
- 安全威胁分类：
  - 拦截/窃听(Interception)：指一个未经授权的用户获得了对一项服务或数据的访问的情况
  - 中断(Interruption)：指服务或者数据变得难以获得、不能使用、被破坏等
  - 篡改(Modification)：指对数据未经授权的改变或者篡改一项服务使其不再遵循其原有的规范
  - 伪造(Fabrication)：指产生通常不存在的附加数据或活动的情况

安全策略和安全机制

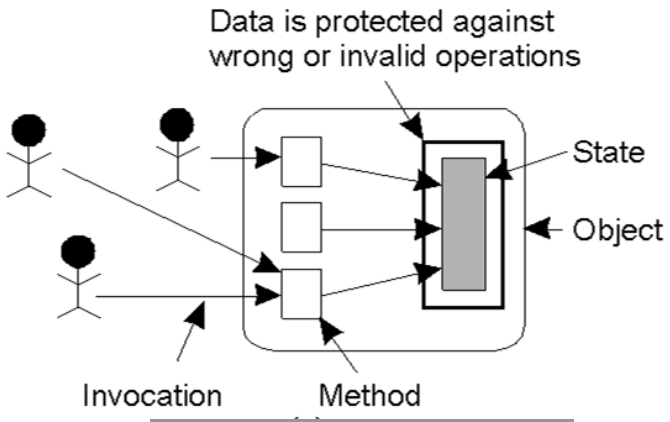
- 安全策略：准确地描述系统中的实体能够采取的行为以及禁止采取的行为
- 安全机制：安全策略通过安全机制来实施
  - 加密(Encryption)：是计算机安全的基础，加密是将数据转换为攻击者不能理解的形式
  - 认证(Authentication)：用于检验用户、客户、服务器等所声明的身份
  - 授权(Authorization)：授予客户执行所请求操作的权限
  - 审计(Auditing)：用于跟踪客户的访问内容以及访问方式

设计问题

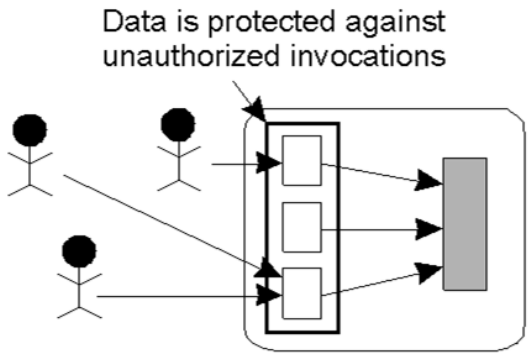
- 安全系统实现时需要考虑的设计问题：
  - 控制的焦点
  - 安全机制的分层
  - 安全机制的分布
  - 简单性

控制的焦点

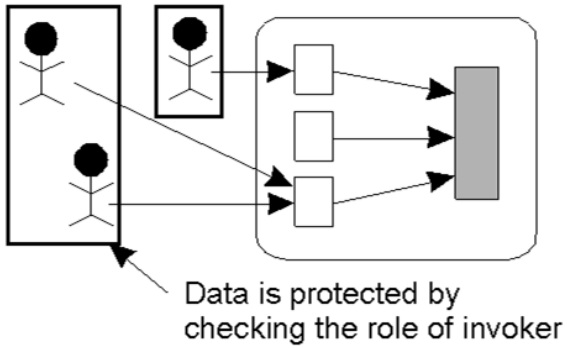
- 防止安全性威胁的三种途径



防止无效操作



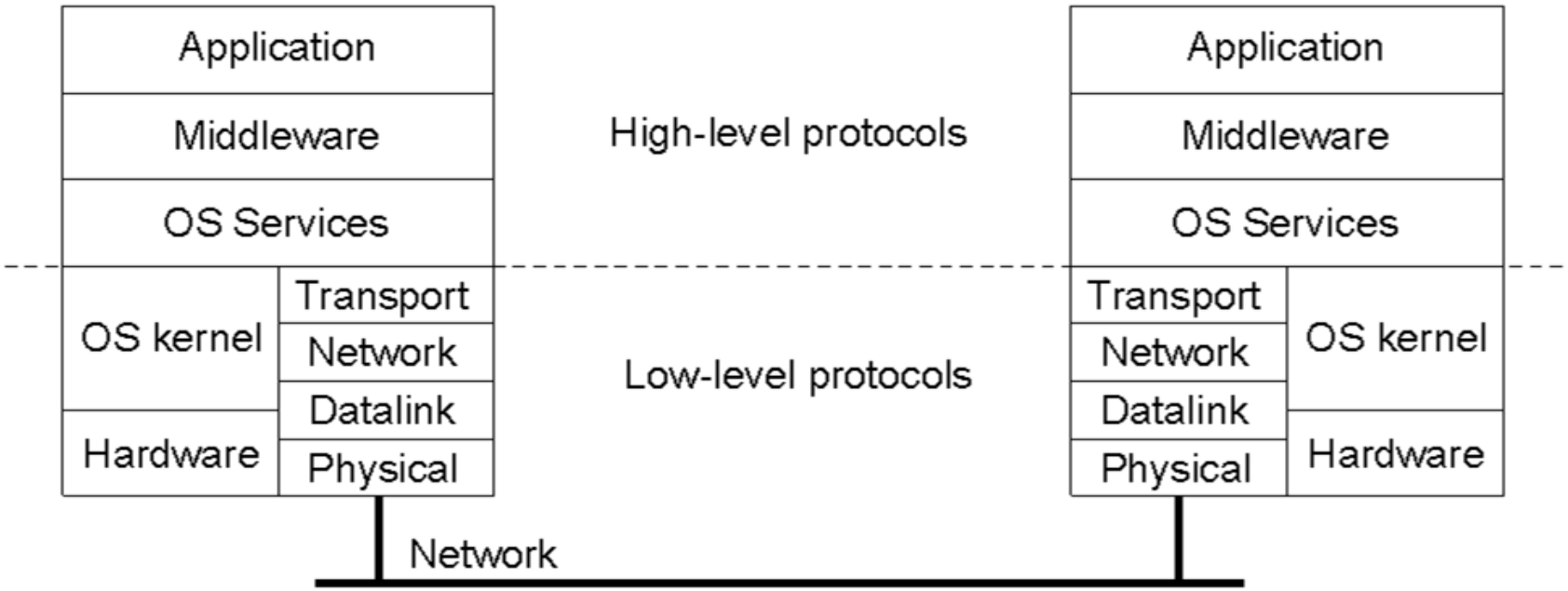
防止未授权调用



防止未授权用户

安全机制的分层结构

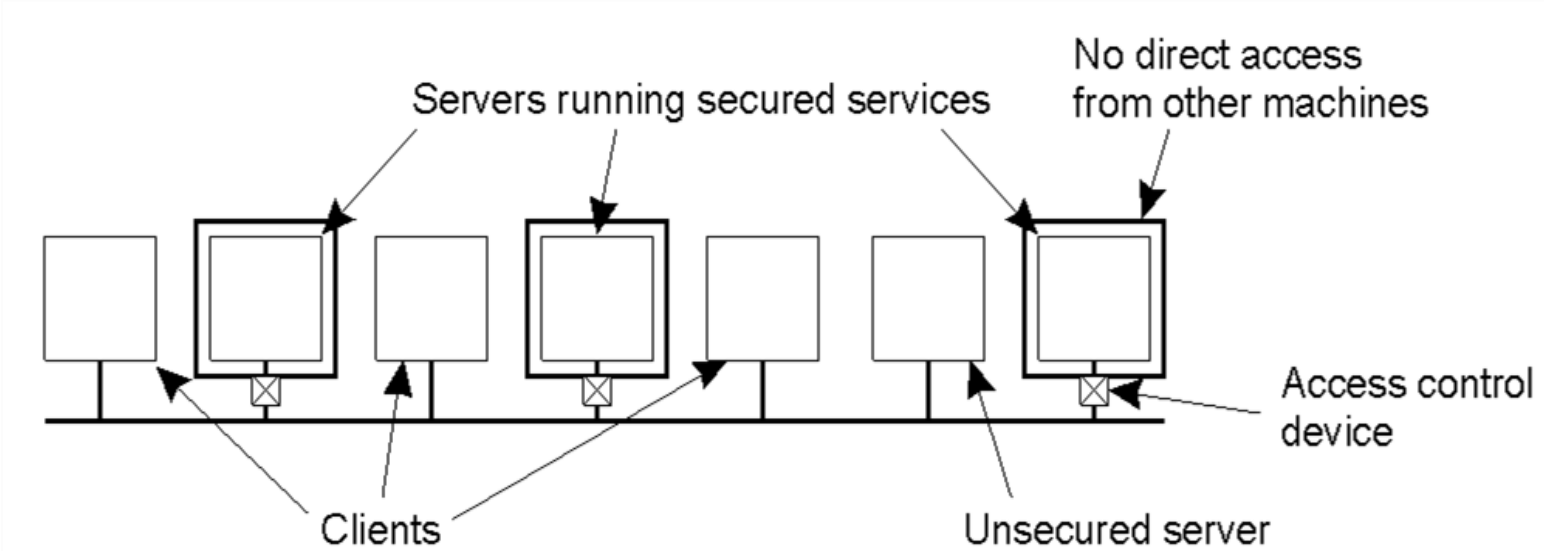
- 分布式系统的分层逻辑结构



安全机制放在哪一层，取决于客户对特定层中服务的安全程度所具有的信任度

安全机制的分布

- 应用于安全分布式系统的 RISSC原理：Reduced Interfaces for Secure System Components



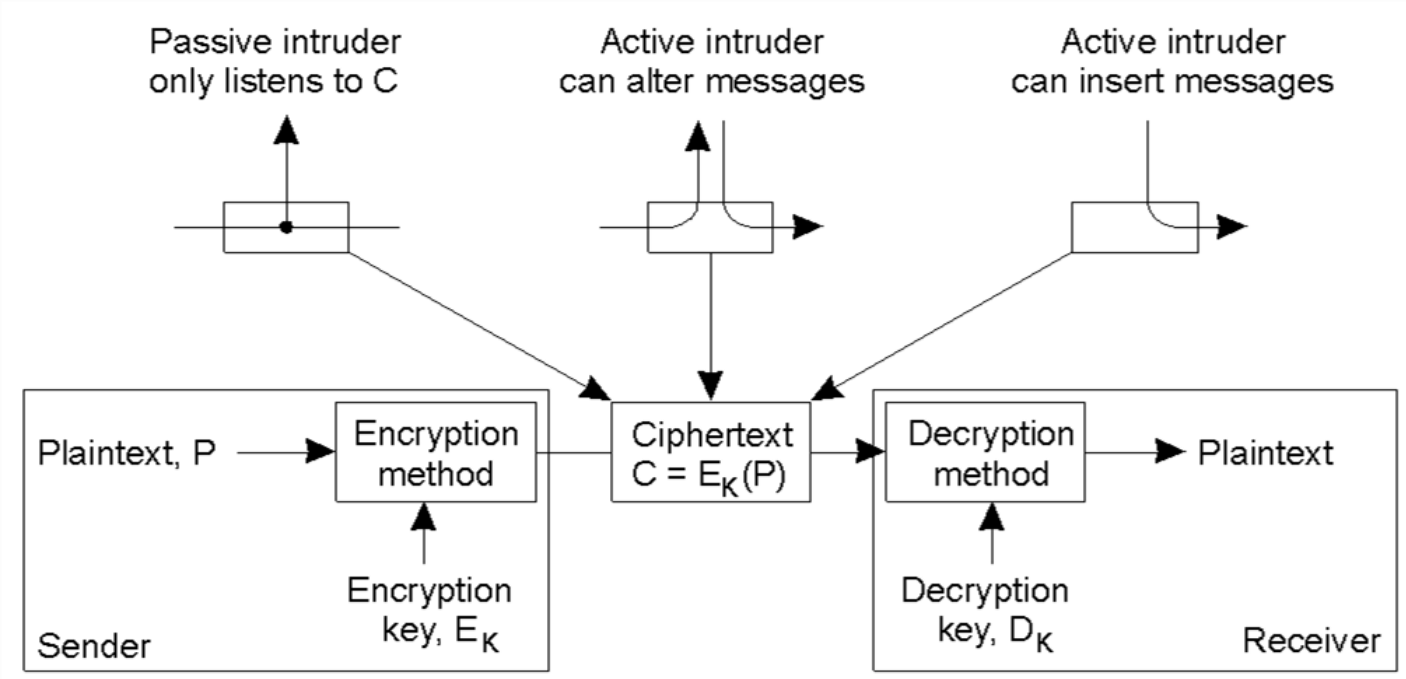
任何安全性非常关键的服务器都放置在一台单独的机器上，防止客户及应用程序直接访问关键服务

简单性

- 安全本身的复杂性和重要性，使得尽管很多安全服务都是高度自动化且对用户隐藏的，但用户常常需要了解服务的工作方式以便其投入信任
  - 如用户级身份认证需要用户对加密密钥的概念的理解以及对诸如证书这样机制的认识
- 如果一个系统的设计者可以使用一些易于理解且工作可靠的简单机制，设计和实现工作都将会比较容易

安全的技术基础：加密 (1)

- 通信中的入侵者和窃听者



加密(2)

- 对称加密系统：使用相同的密钥进行一个消息的加密和解密

$$P = D_K(E_K(P))$$

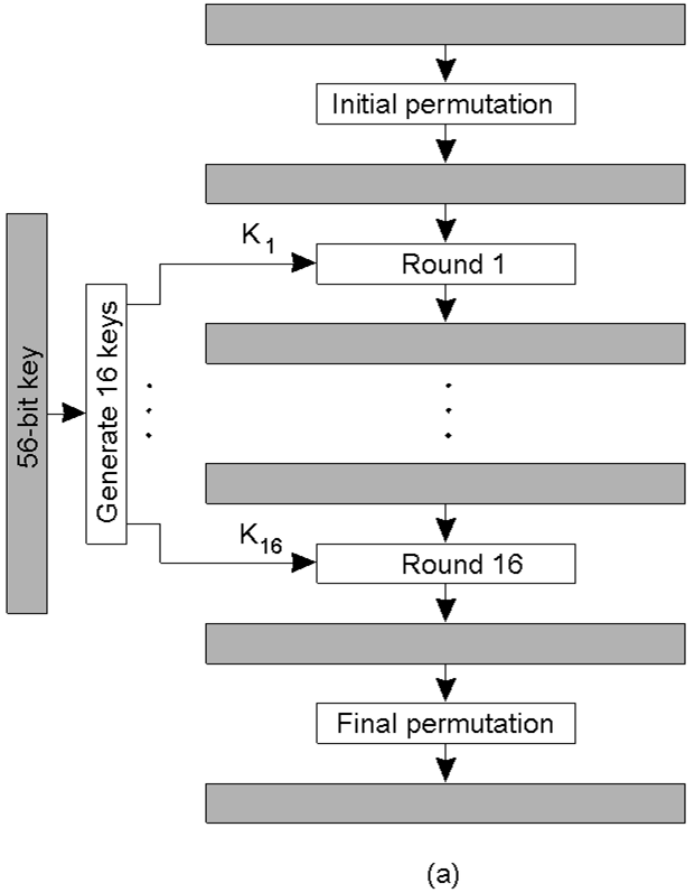
- 非对称加密系统（也叫公钥系统）：加密和解密所使用的密钥不同，但两个密钥一起构成唯一的一对

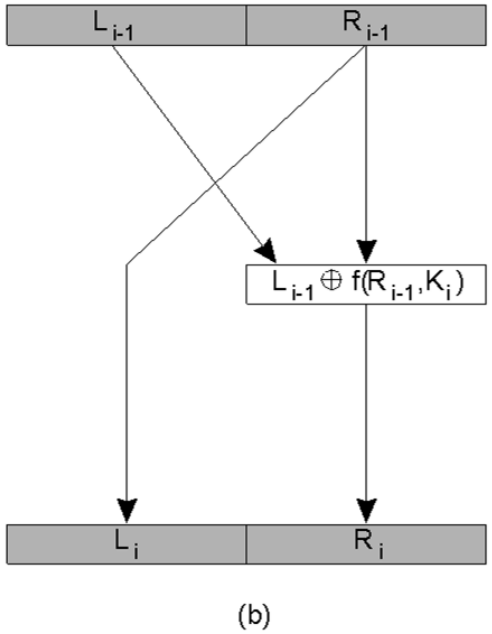
$$P = D_{K_D}(E_{k_E}(P))$$

- 符号定义

Notation	Description
$K_{A,B}$	A和B共享的密钥
K_A^+	A的公钥
K_A^-	A的私钥

对称加密系统: DES





（a）DES原理

（b）一轮加密

公钥加密系统: RSA

- 基本出发点：没有任何已知的方法能够有效地找到大数的质因子
- 生成私钥和公钥的4个步骤:

1. 选择非常大的两个质数, p 和 q
2. 计算$n = p \times q$ 和$z = (p - 1) \times (q - 1)$
3. 选择数字 d ， d是z 的质因数
4. 计算数字 e，使得 $e \times d = 1 \bmod z$

- E和d，其中一个用作解密码，另一个用作加密码
  - $c_i = m_i^e \pmod n$
  - $m_i = c_i^d \pmod n$

RSA VS 私钥系统？

- 使用RSA加密消息比DES慢大约100~1000倍
- 所以，许多加密系统使用RSA来以安全方式交换共享密钥，但很少用它来加密数据。
  - 有机结合二者，取长补短

主要内容

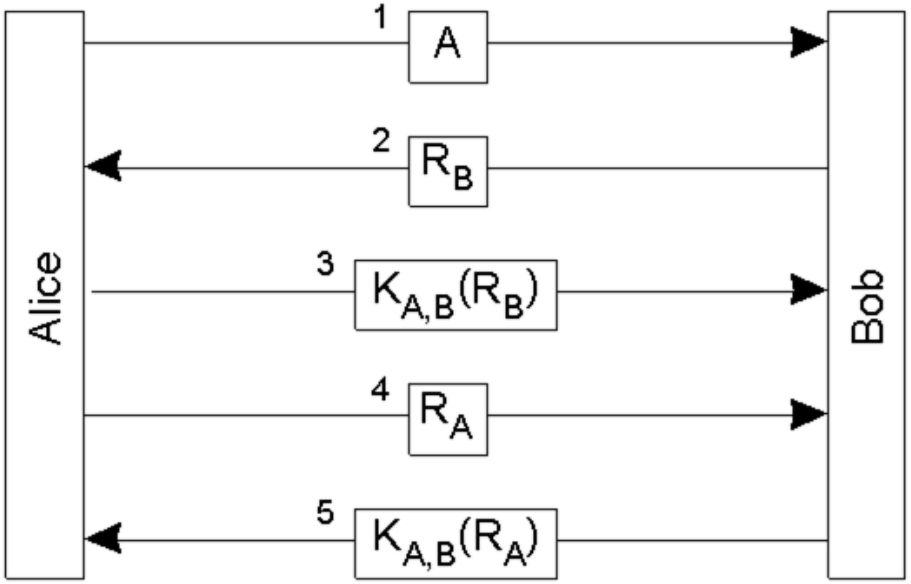
- 安全性概述
- 安全通道
  - 身份认证
  - 消息的完整性和机密性
- 访问控制
- 安全性管理

安全通道

- 保护发送者和接收者之间的安全通信：通信方正确（身份认证）、通信的数据保持机密性和完整性
  - 使用对称性密钥和公钥密码系统

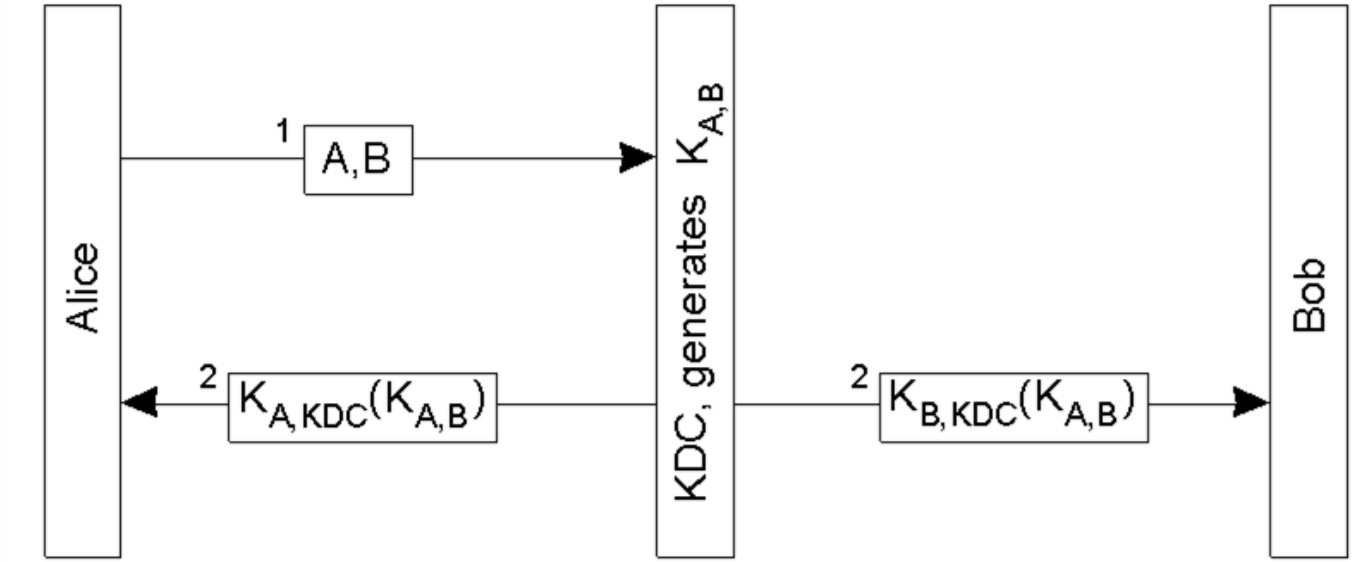
基于共享密钥的身份认证

- 质询-响应协议(challenge-response)
- 使用五个消息的基于共享密钥的身份认证
  - $K_{A,B}$：A,B的共享密钥



使用密钥发布中心的身份认证

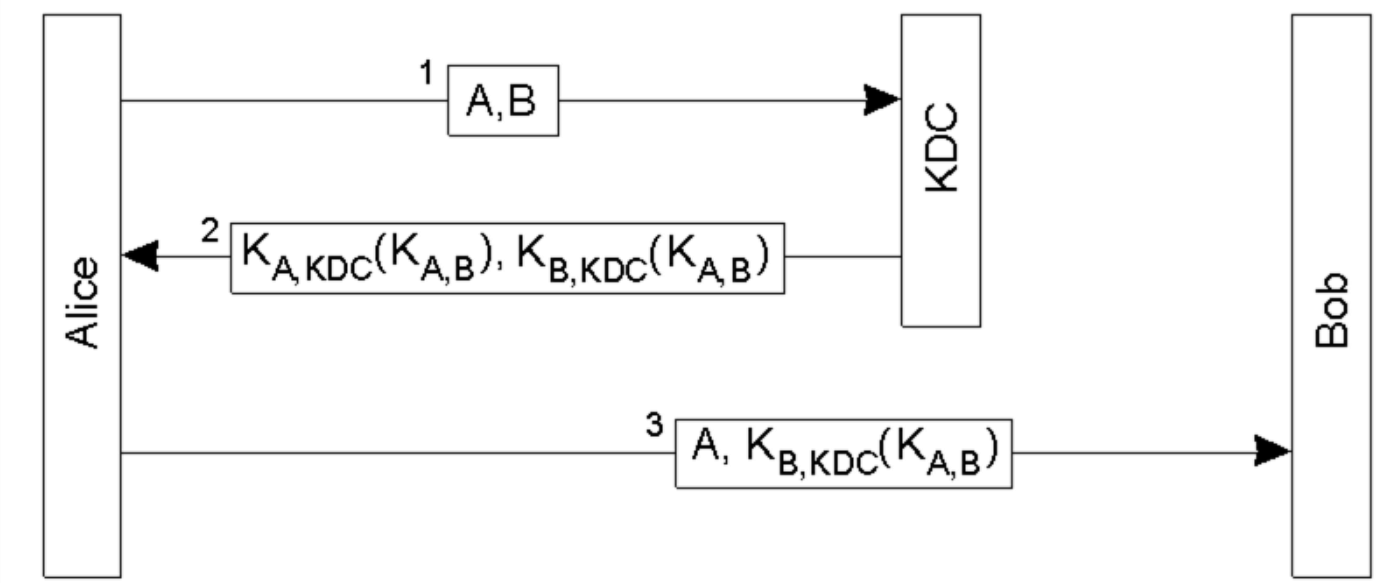
- 前面介绍的方法扩展性较差
- 使用KDC的原理：每个客户只需要知道自己与KDC间的秘钥



A可能在B接收到来自KDC的共享密钥之前就开始建立安全通道怎么办？

使用票据

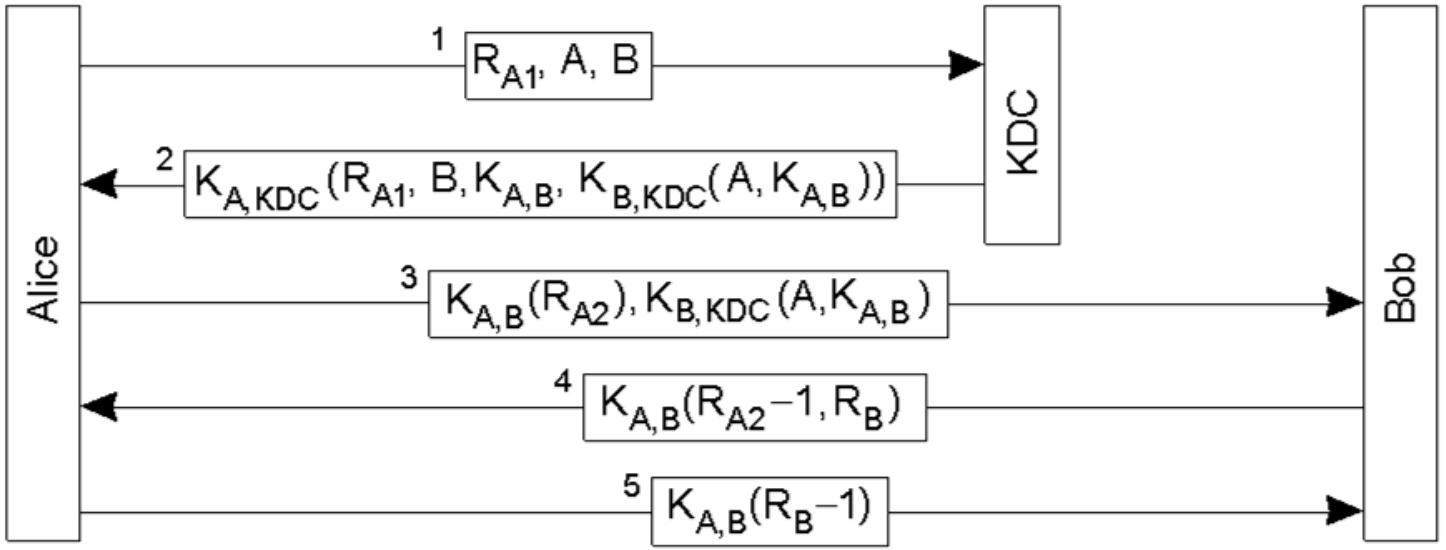
- 使用ticket，使得Alice建立一个与Bob的连接通道.



如果Chuck获取了一个老的消息2，并得到了B的老的与KDC间的密钥？

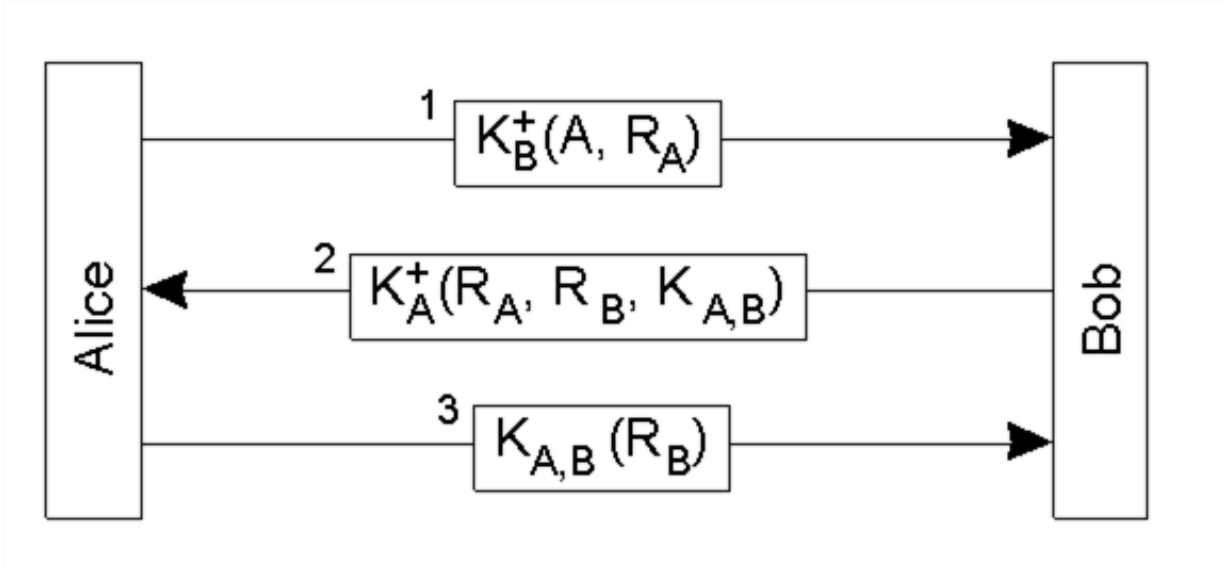
Needham-Schroeder鉴别协议

- nonce：只使用一次的随机数



使用公钥密码的身份认证

- 在公钥密码系统中，相互鉴别

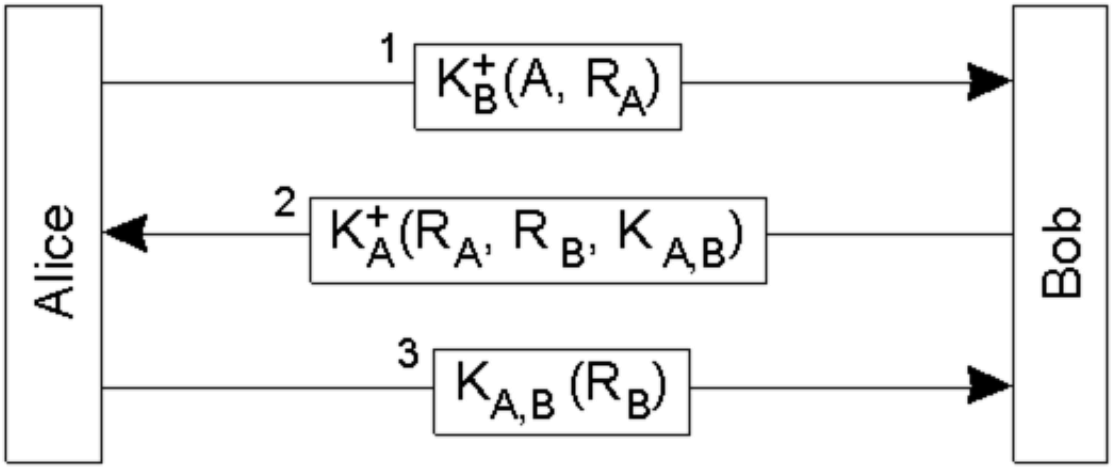


消息的完整性和机密性

- 除身份认证外，安全通道还应该保证消息的机密性和完整性
  - 机密性：通过加密实现
    - ➔常使用会话密钥
  - 完整性：常通过数字签名保证

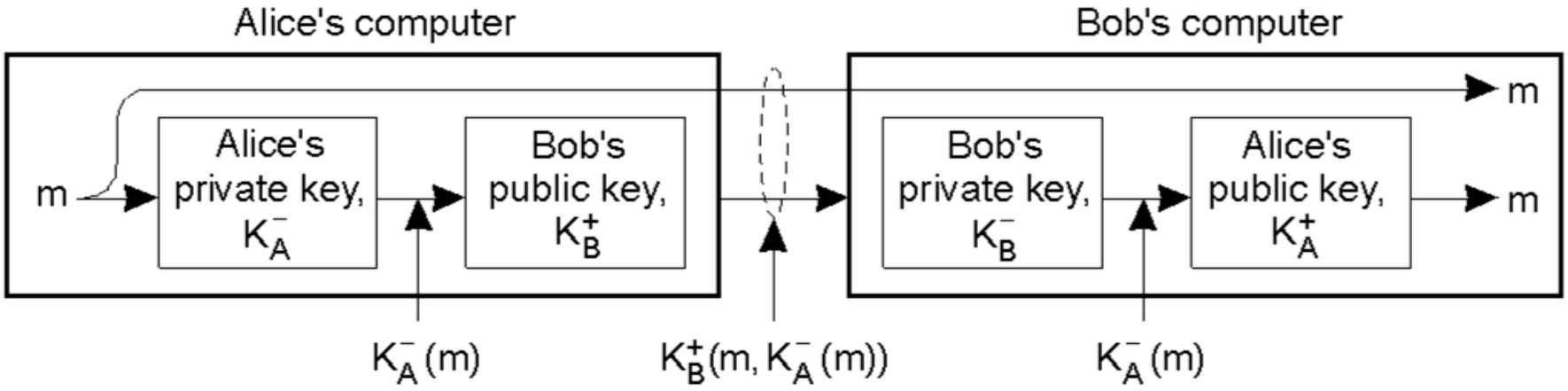
会话密钥

- 在身份认证阶段完成以后，安全通道各方一般使用唯一的共享会话密钥以实现机密性，而不再使用该通道时丢弃该会话密钥
  - 保证身份认证密钥的安全
  - 防止重放攻击
  - 隔离会话密钥受损带来的危害



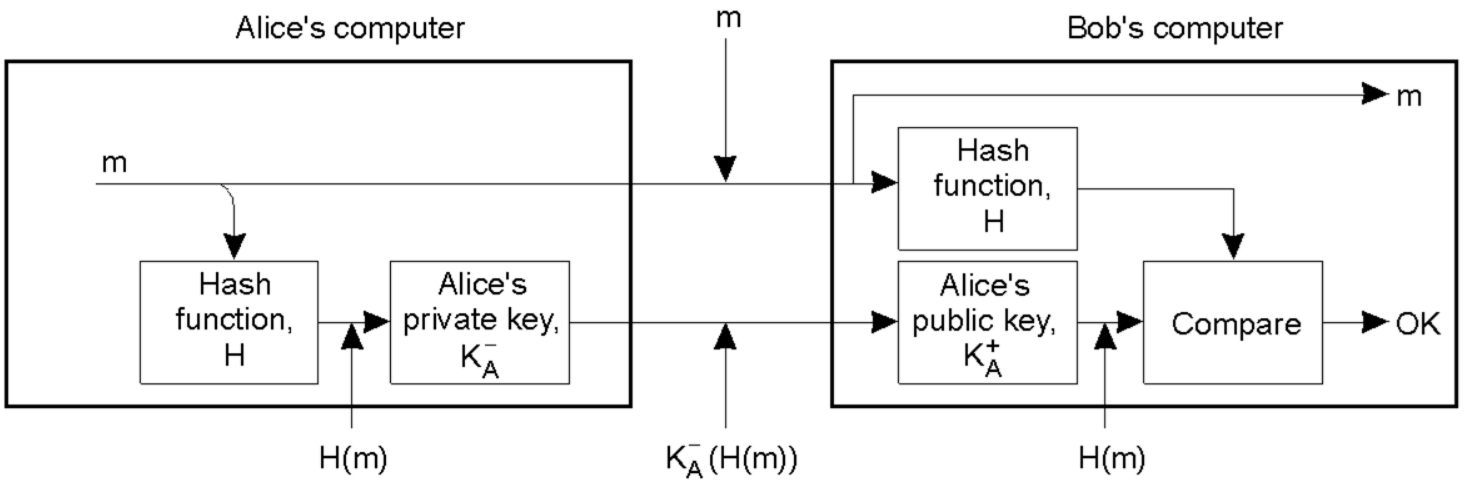
数字签名 (1)

- 消息的完整性通常超出安全通道的传输范畴
  - 如A向B报价某商品：B修改报价、A否认报价
- 使用公钥密码对一个消息进行数字签名



数字签名(2)

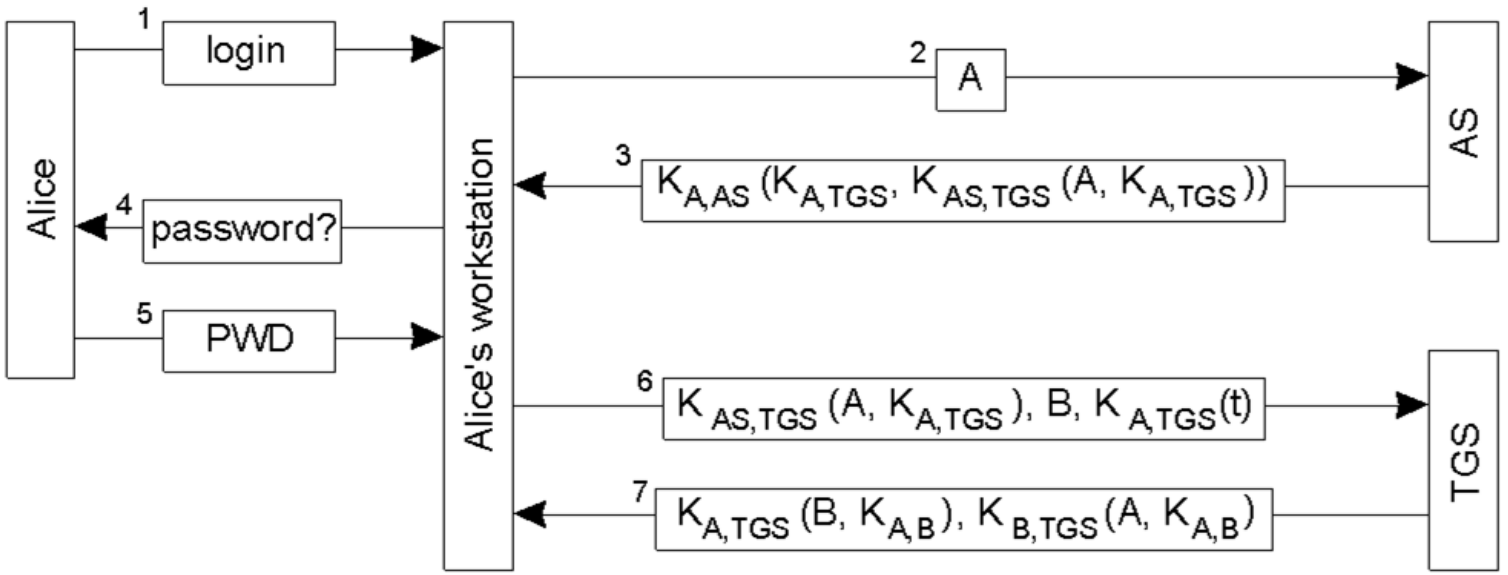
- 使用消息摘要message digest对一个消息进行数字签名



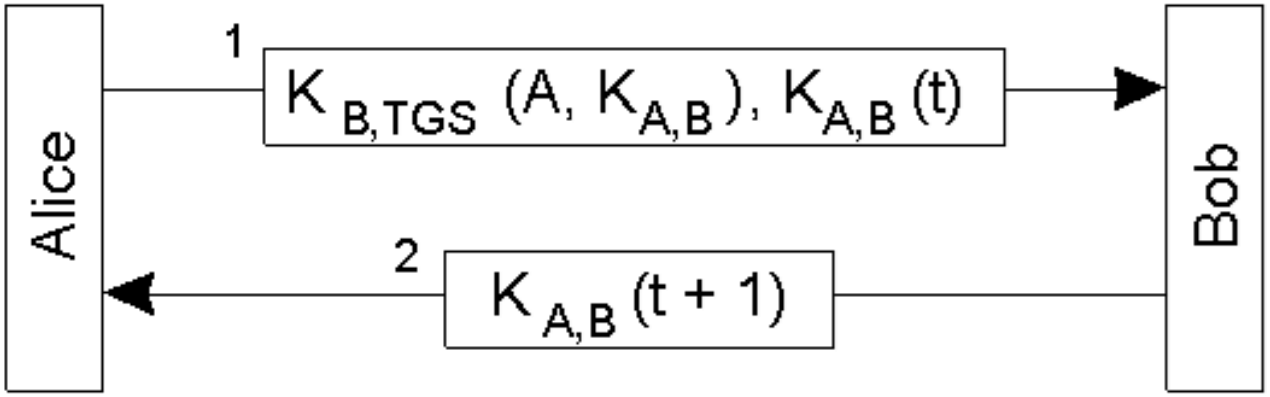
例：Kerberos系统

- 可看成一个安全服务系统，该系统帮助客户和服务器建立安全通道，安全是基于共享密钥的
- 两个主要组件：
  - 身份认证服务器AS：负责处理来自用户的登陆请求，进行身份认证并提供与TGS建立安全通道的密钥
  - 票据授予服务TGS：分发称为票据的特殊消息，用于使服务器确信该客户正是其所声称的那个客户，并建立客户与服务器间的安全通道

例：Kerberos的身份认证



例：Kerberos中建立安全通道



主要内容

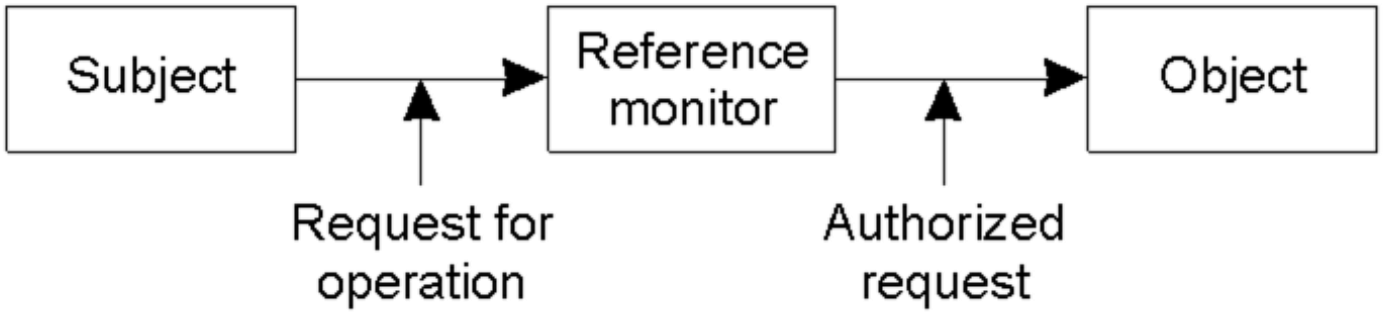
- 安全性概述
- 安全通道
- 访问控制
  - 常见问题
  - 防火墙
  - 拒绝服务
- 安全性管理

访问控制

- 访问控制：
  - 对访问权限的验证
- 授权：
  - 授予访问权限
- 这两个词通常以可互换的方式使用

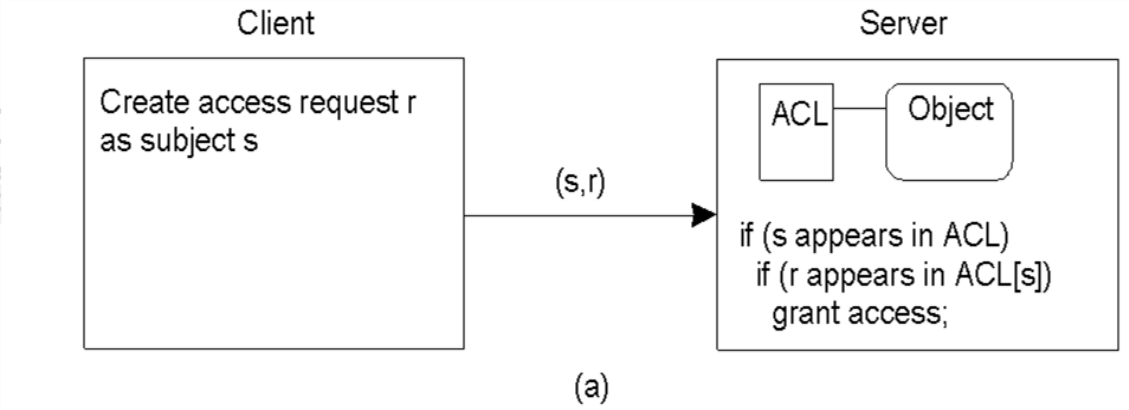
访问控制问题

- 访问控制的一般模型
  - 主体：发出请求
  - 客体：被访问的对象
  - 引用监控器：记录主体作了什么，决定是否允许主体执行对客体的操作



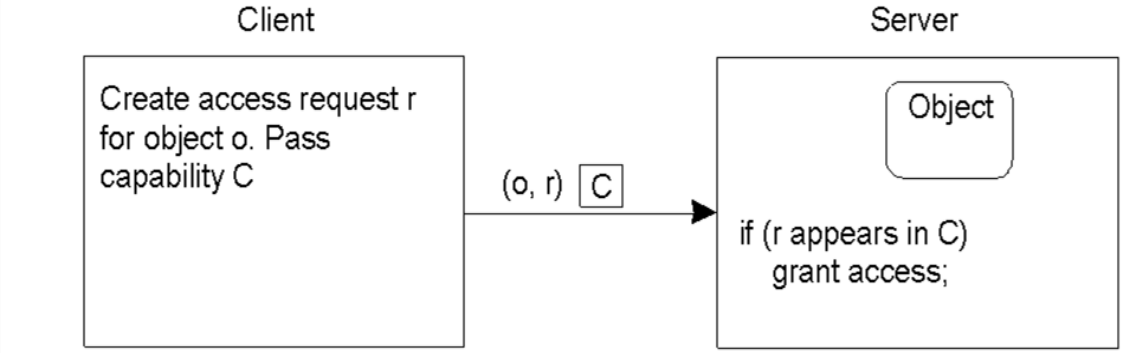
访问控制矩阵和权能

- 在保护对象上访问控制矩阵ACL 和权能的比较



(a)

使用 ACL

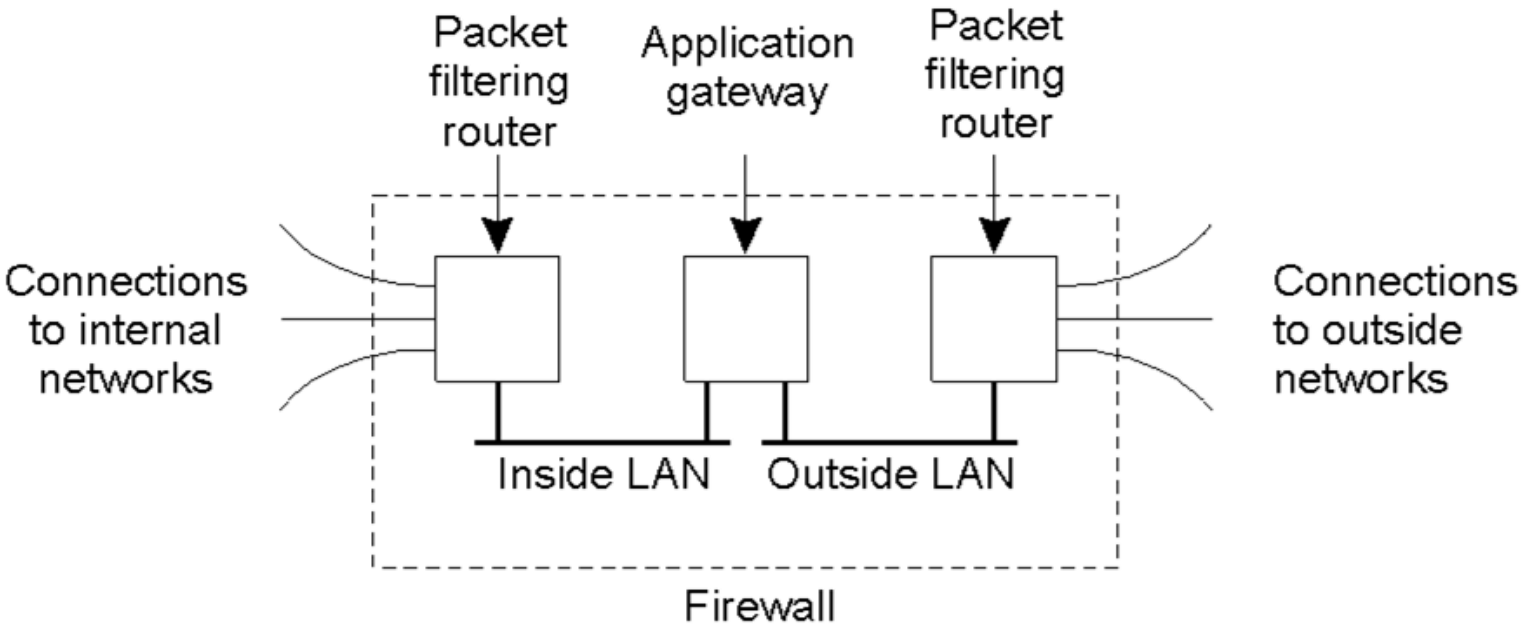


(b)

使用 capabilities.

防火墙

- 防火墙常见实现方法



拒绝服务

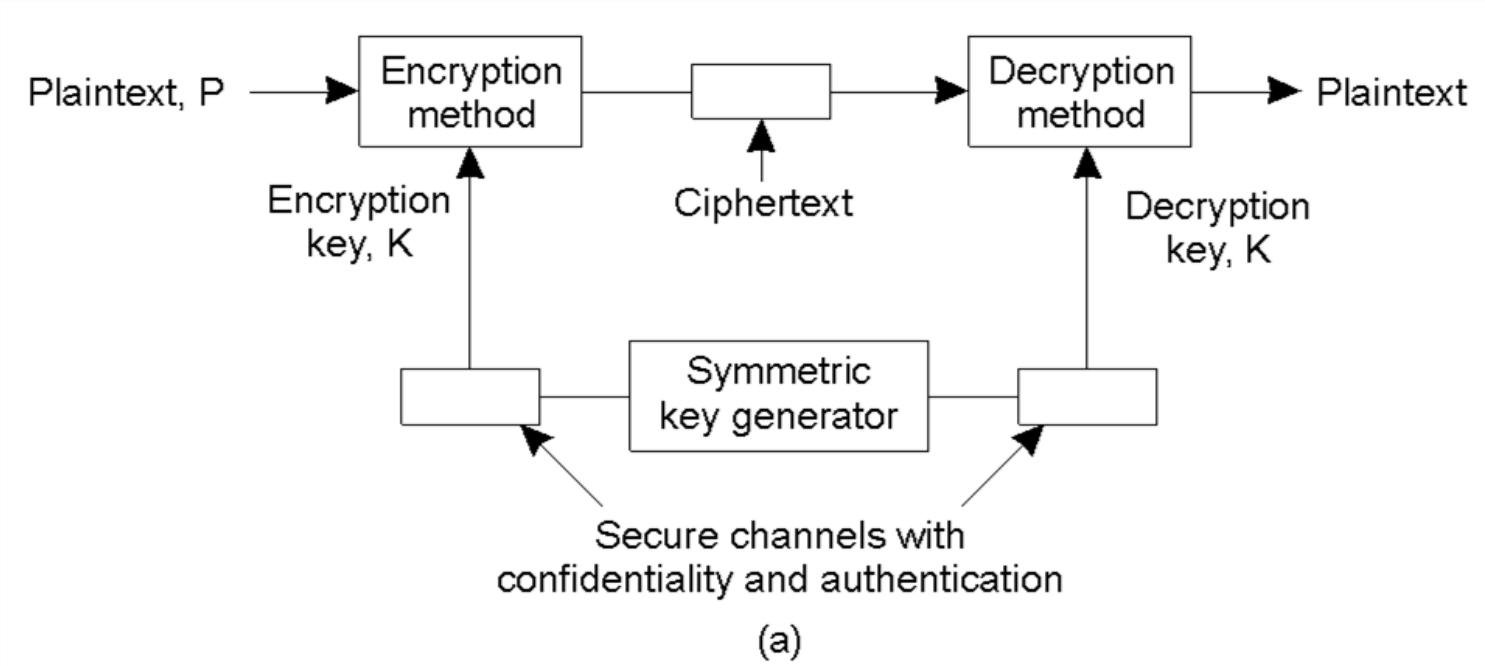
- 访问控制技术是为了确保只有被授权的客户访问资源
- 拒绝服务攻击是恶意地阻止已授权用户访问资源
  - 带宽耗竭的攻击
  - 资源耗竭的攻击
- 没有特别好的方法用于阻止DDos，新的攻击方法不断出现，需要部署新的方法
- 不断监视网络流量是比较好的方法

主要内容

- 安全性概述
- 安全通道
- 访问控制
- 安全性管理
  - 密钥管理
  - 授权管理

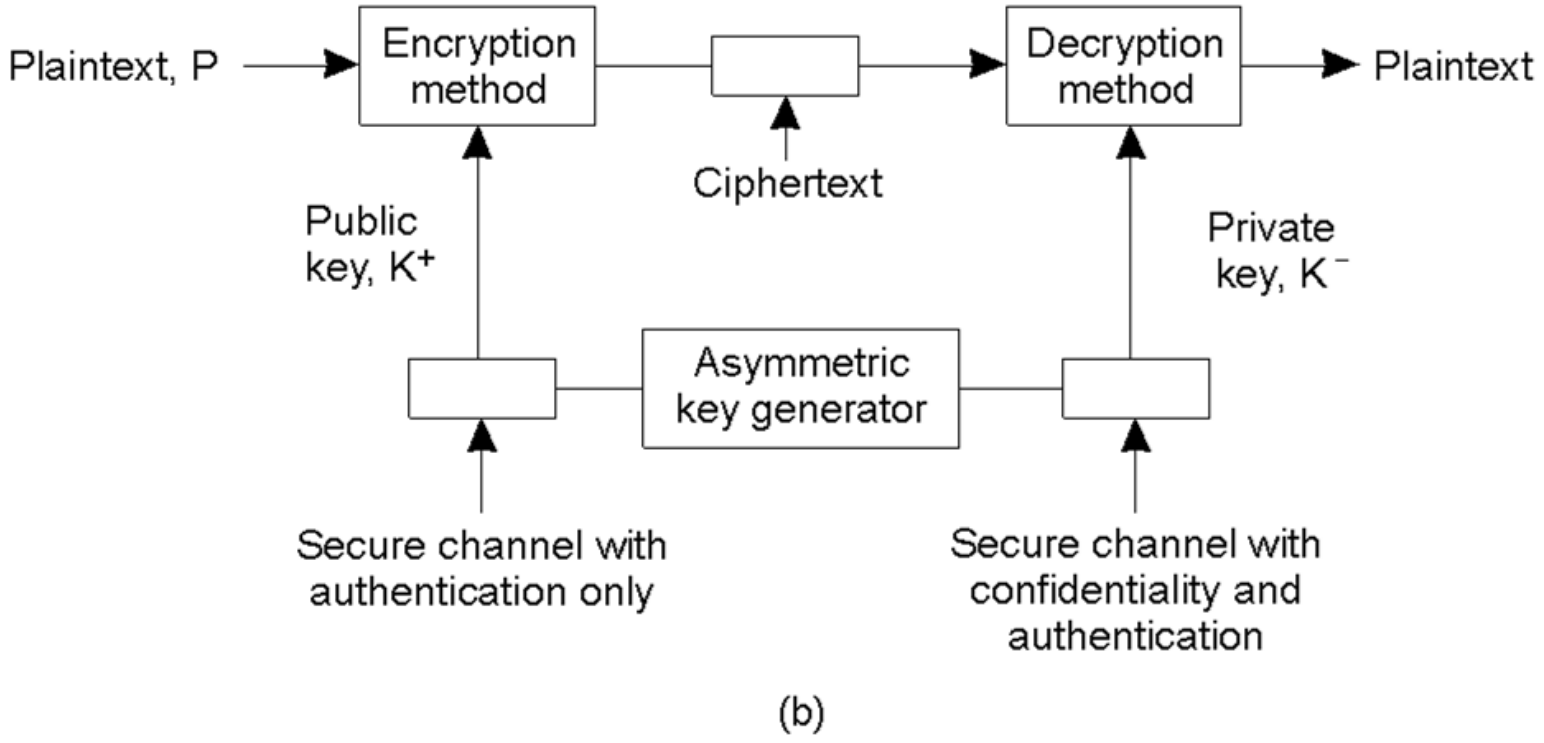
密钥的分发 (1)

- 共享密钥的分发：



密钥的发布(2)

- 公钥私钥发布



公钥证书

- 公钥分发可通过使用公钥证书实现
- 公钥证书由一个公钥与一个标识该公钥所关联的实体的字符串共同组成
- 该公钥和标识符共同由认证机构签发，因此此签名也置于该证书上。
- 公钥证书可以被吊销
  - 证书吊销表
  - 设置生命期

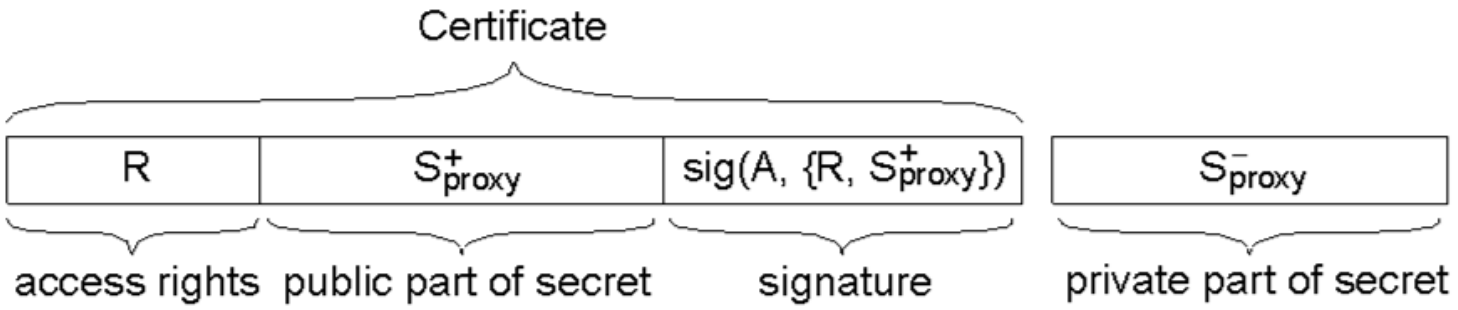
授权管理：权能

- 权能是对于指定资源的一种不可伪造的数据结构，它确切指定它的拥有者关于该资源的访问权限
- Amoeba系统中的权能
  - 早期的一种基于对象的分布式系统

48 bits	24 bits	8 bits	48 bits
Server port	Object	Rights	Check

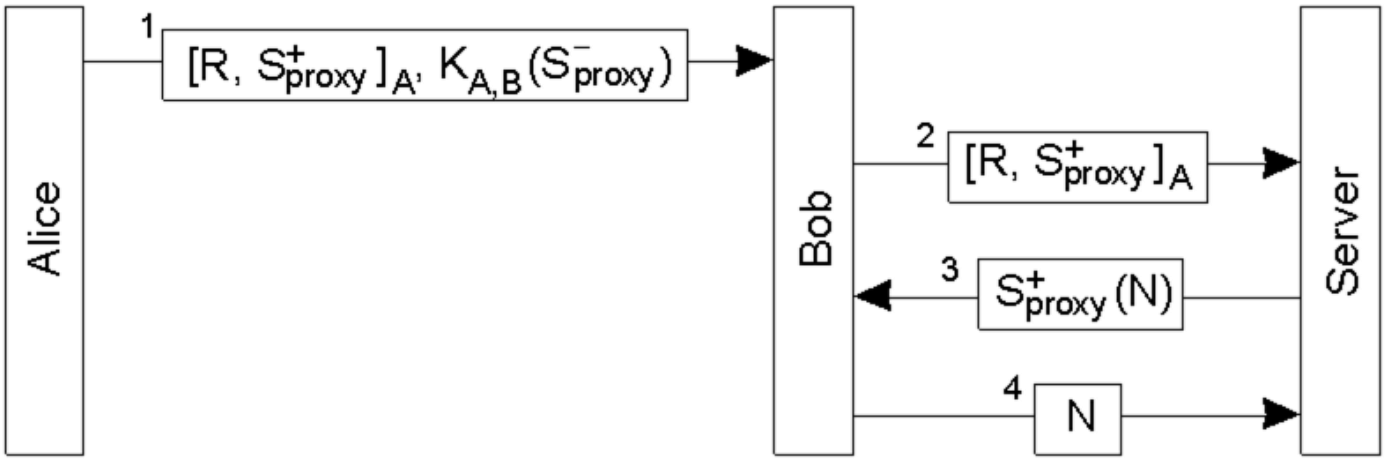
委派Delegation (1)

- 委派：
  - 将访问权限从一个（用户）进程传递给另一个进程
    - ➔A说“B具有权限R”
    - ➔A说“此证书的持有者具有权限R”
- 代理（proxy）：使其拥有者按照授予的权限操作的令牌
- 用作delegation的通用代理结构：



Delegation (2)

- 使用代理 去delegate和证明访问权的拥有权



安全小 结

- 安全性概述：安全威胁、安全机制、加密
- 安全通道：身份认证、会话密钥、数字签名
- 访问控制：访问控制矩阵、权能、防火墙、拒绝服务
- 安全管理：密钥管理、授权管理