第九章　分布式安全

- 分布式安全简介
- 安全通道
- 访问控制
- 安全管理
- 实例: **Kerberos**

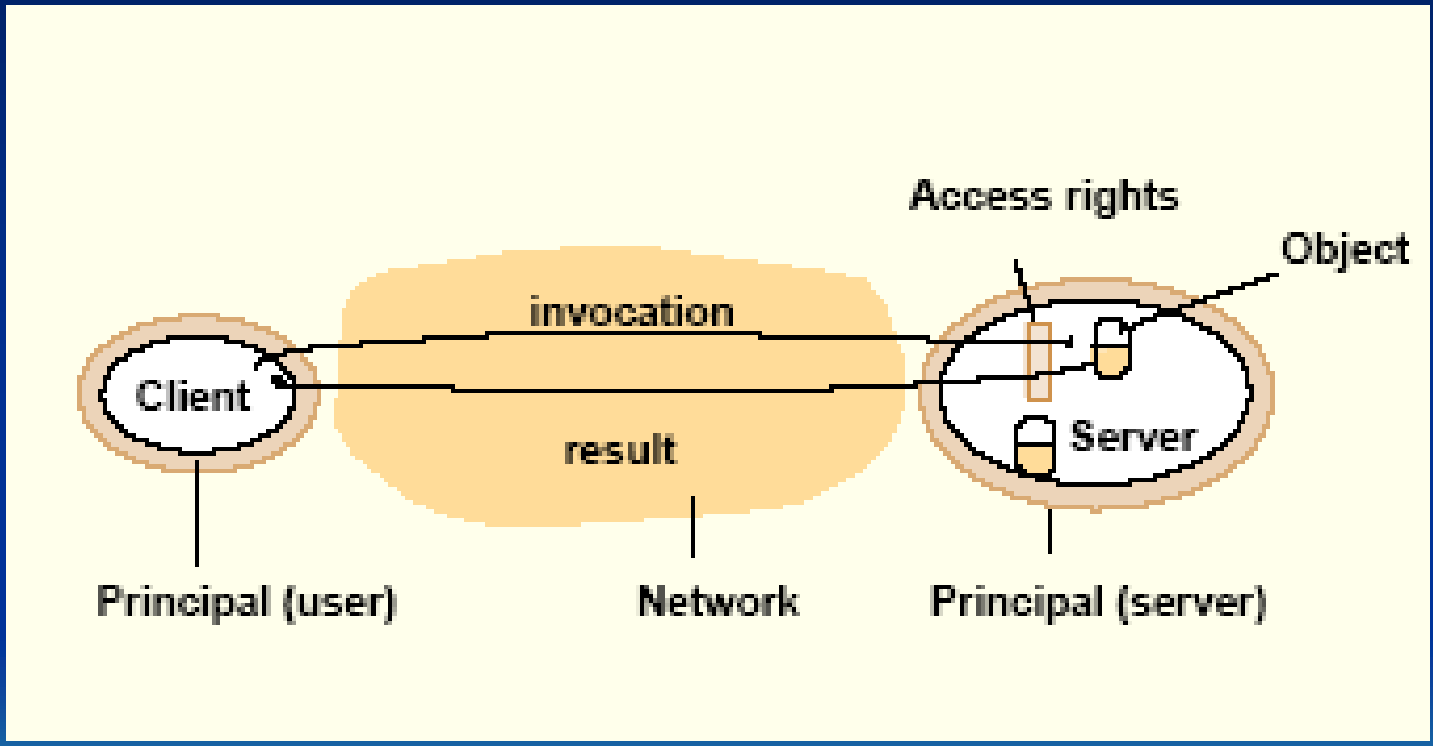
分布式安全简介

- 可用性（**Availability**）：任何给定的时刻都能及时工作
- 可靠性（**Reliability**）：系统可以无故障的持续运行
- 安全性（**Safety**）：系统偶然出现故障能正常操作而不会造成任何灾难
- 可维护性（**Maintainability**）：发生故障的系统被恢复的难易程度
- 机密性:系统将信息只向授权用户公开
- 完整性:对系统资源的更改只能以授权方式进行

安全模型

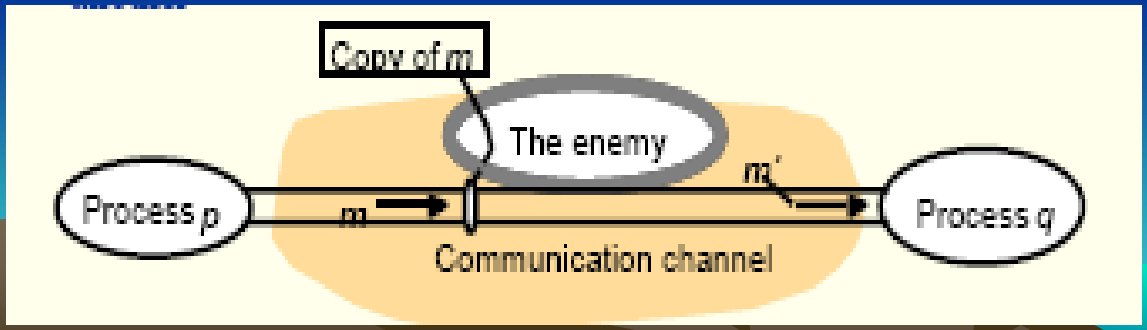
- 分布式的安全
  - 进程
  - 通信通道
  - 对象
- 保护对象
  - 访问权(Access rights): who is allowed to perform the operations of an object
  - 授权用户(Principal): the authority who has some rights on the object

安全模型


Access rights
Object
invocation
Client
result
Server
Principal (user)
Network
Principal (server)

安全威胁

- 对进程的威胁
  - 对服务器： 使用假身份欺骗
  - 对客户：使客户收到假结果：窃取客户口令
- 对通信通道的威胁
  - 复制、改变或插入消息
  - 保存和重放：从账户转账

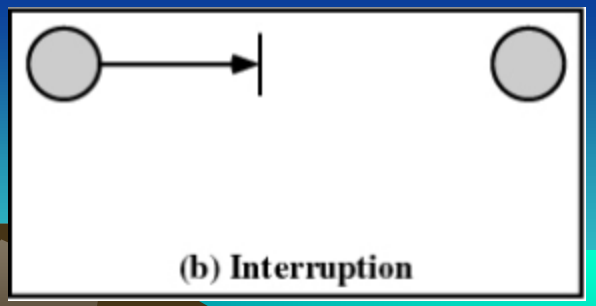


对通信通道的安全威胁类型

- 切断（Interruption）
- 侦听（Interception）
- 更改（Modification）
- 伪造（Fabrication）

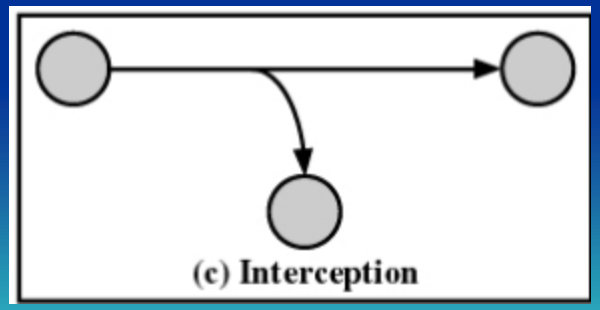
威胁的类型

- 切断（Interruption）
  - 系统中的资源被破坏而变得不可获得或不能用
  - 对可用性的攻击
  - 破坏硬件
  - 切断通信线
  - 使文件系统失效



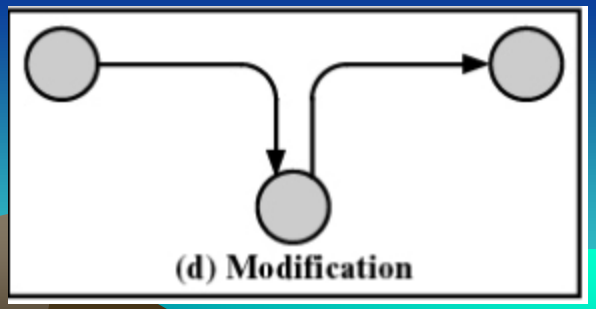
威胁的类型

- 侦听（Interception）
  - 未授权者获得资源的访问
  - 对机密性的攻击
  - 搭线获得对资源的访问,非法复制文件或程序



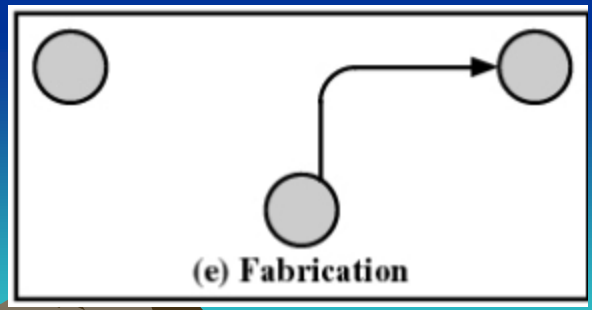
威胁的类型

- 更改（Modification）
  - 未授权者不仅获得资源的访问，还进行更改
  - 对完整性的攻击
  - 改变数据文件的数值
  - 改变程序使其执行发生变化
  - 修改网络中传输消息的内容



威胁的类型

- 伪造（Fabrication）
  - 未授权者将伪造的对象加入系统中
  - 对可靠性的攻击
  - 将假消息放到网络中
  - 在文件中添加记录

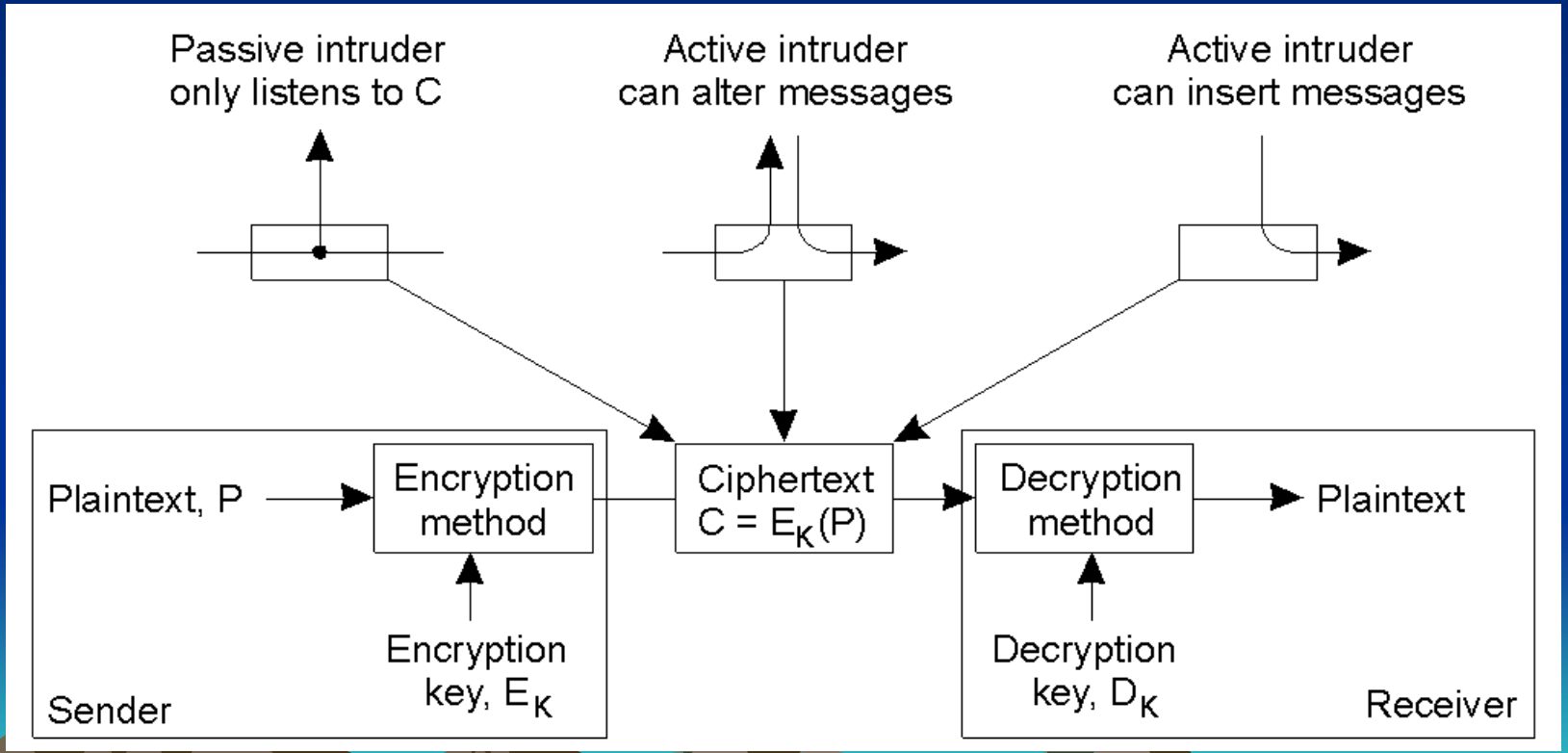


安全的机制

- 加密（Encryption）：将数据转换为攻击者不能理解的形式，保证机密性
- 身份验证（Authentication）：用于检验用户、客户、服务器等所声明的身份
- 授权（Authorization）：身份验证后，检查是否给予该客户执行所请求操作的权限
- 审计（Auditing）：跟踪每个客户的访问内容以及访问方式，用于对于分析安全破坏的方式以及随后采取措施防范

加密 Cryptography (1)

- 加密技术是分布式系统的基本措施
- 加密解密通过以密钥K为参数的加密算法实现



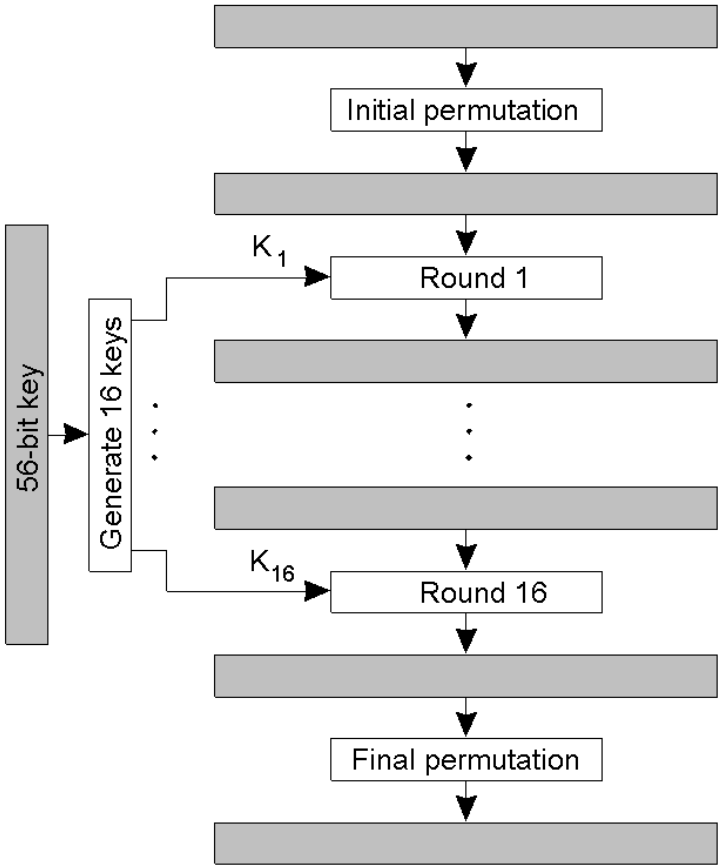
通信中的入侵者和窃听者

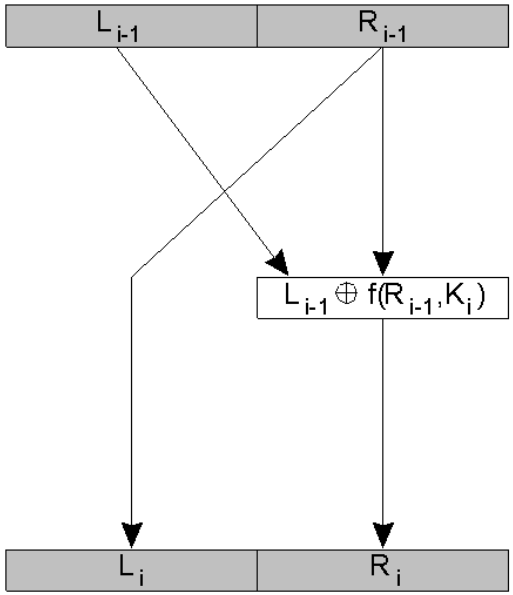
加密 (2)

- 对称加密系统：加密与解密密钥相同，即$P=D_K(E_K(P))$,也称为共享密钥系统
- 非对称加密系统：加密与解密密钥不同(一个公开、一个保密)，但构成唯一的一对，即$P=D_{KD}(E_{KE}(P))$,也称为公钥系统
- 本章使用的符号

符号	描述
$K_{A,B}$	A B共享的密钥
K_A^+	A的公钥
K_A^-	A的私钥

对称加密系统
Symmetric Cryptosystems: DES (1)



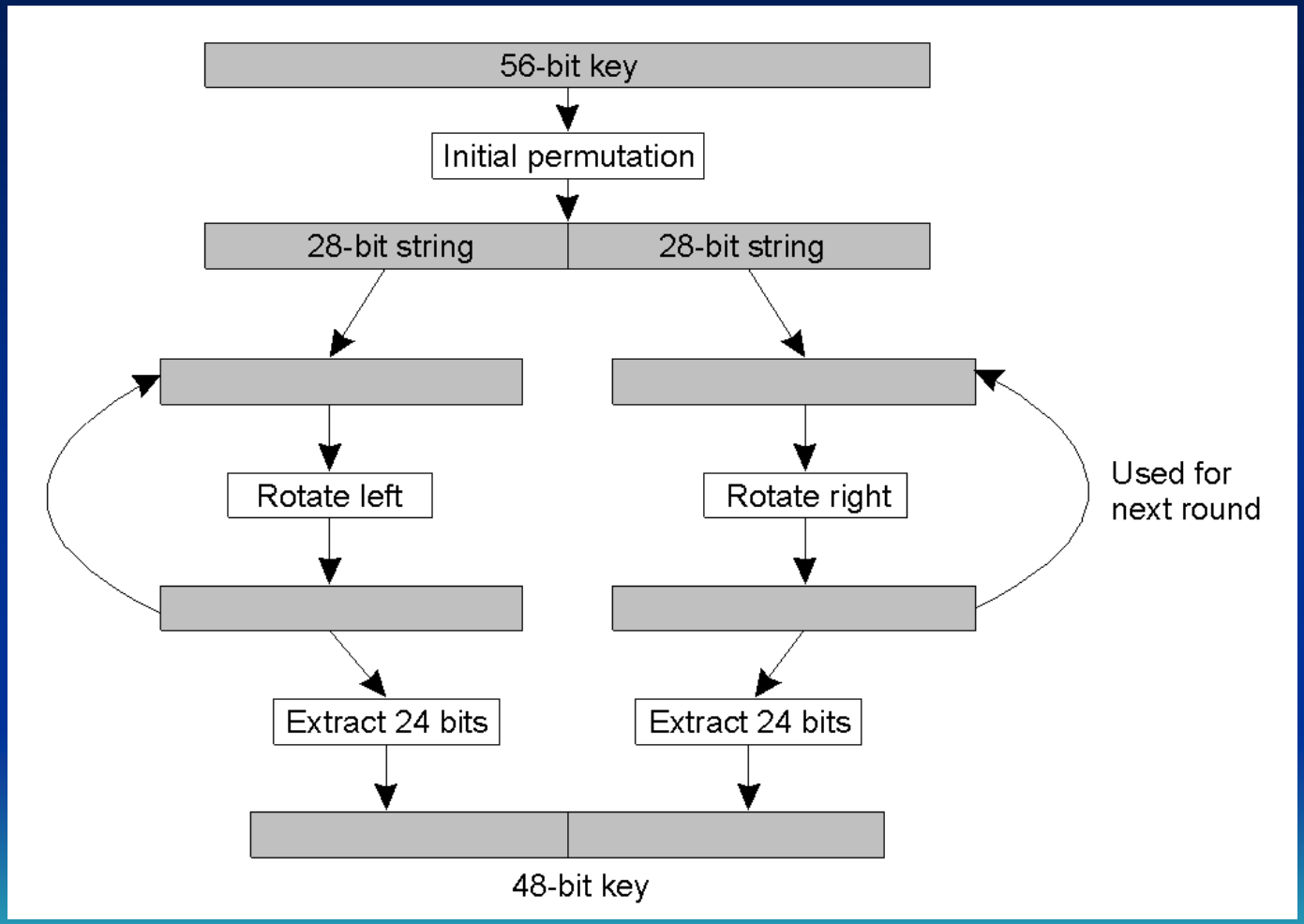


- 64位数据块
- 16轮转换
- 每轮48位密钥
- 来源于56位主密钥
- 切碎函数 f

a) DES原理

b) 一个加密轮次的概要

对称加密系统: DES (2)



DES中每轮密钥生成的细节

对称加密系统: DES (3)

- DES的原理非常简单，但使用分析的方法难以破解该算法
- 原因是其设计的基本原理从未被清楚地解释过
- 多年以来DES一直作为标准加密技术使用，但正被128位的 Rijndael 算法取代

公钥加密系统
Public-Key Cryptosystems: RSA

- 每个整数都可以写成素数的乘积，如 **2100=2*2*3*5*5*7**
- **RSA**的安全性来自于以下事实：没有任何已知的方法可以有效地找到大数的素因子
- 产生私钥和公钥的四个步骤：
  1. 选择两个非常大的素数 ***p*** 和 ***q***
  2. 计算 $n = p \times q$ 和 $z = (p - 1) \times (q - 1)$
  3. 选择 ***z*** 的一个相关素数***d***
  4. 计算 ***e*** 使得（$e \times d) \bmod z = 1$
  5. 若用整数**X**表示明文，整数**Y**表示密文**(X，Y**均小于**n)**，则加解密运算为：

     加密：$Y=X^e \bmod n$

     解密：$X=Y^d \bmod n$

     其中的**d**和**n**也互素、**e**和**n**是公钥、**d**和**n**是私钥。

公钥加密系统

- RSA的安全性依赖于大数的因子分解
- 加密消息比DES慢100-1000倍，只用于加密较短的信息，如共享密钥。
- 下表给出了在计算机每一微秒做一次操作的假定下分解不同大小的N所需要的时间：

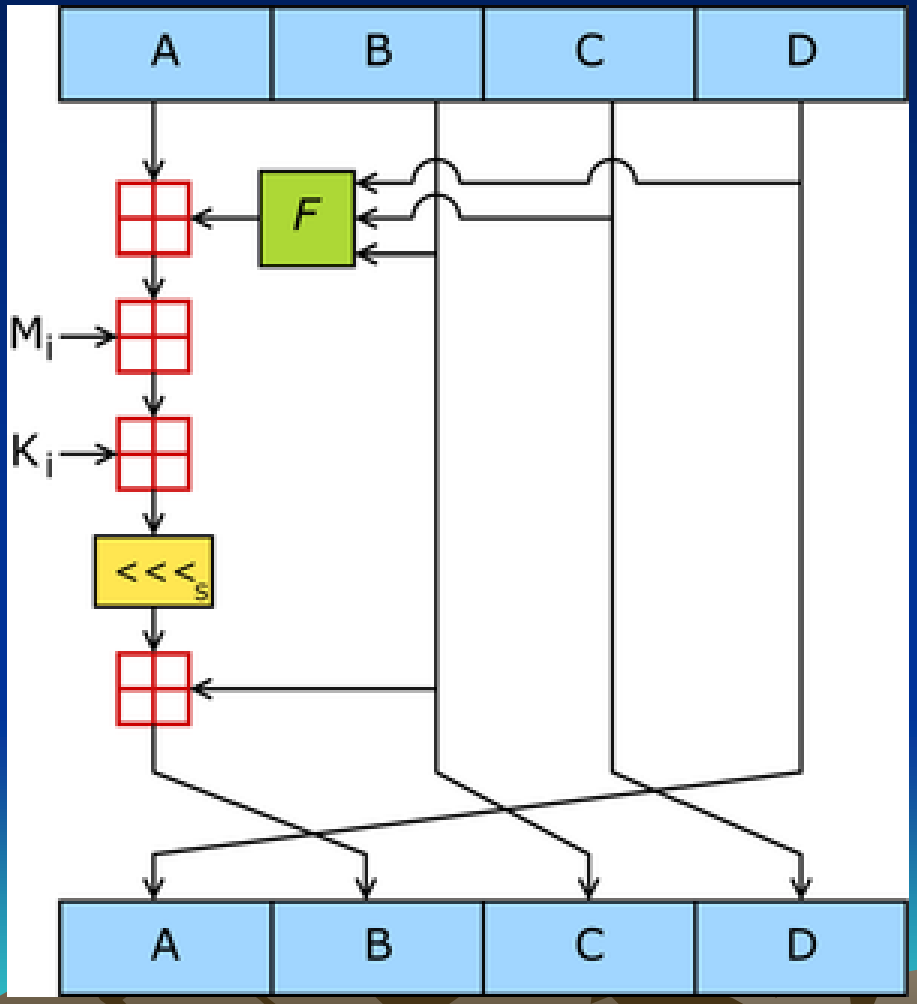
N的十进位数	时间
50	3.9小时
75	104天
100	4年
200	3.8×1015年

MD5（Message-Digest Algorithm 5 ）

MD5是计算机安全领域广泛使用的一种散列函数，用以提供消息的完整性保护，具有以下特点：

- 压缩性：任意长度的数据，算出的MD5值长度都是固定的。
- 容易计算：从原数据计算出MD5值很容易。
- 抗修改性：对原数据进行任何改动，哪怕只修改1个字节，所得到的MD5值都有很大区别。
- 弱抗碰撞：已知原数据和其MD5值，想找到一个具有相同MD5值的数据（即伪造数据）是非常困难的。
- 强抗碰撞：想找到两个不同的数据，使它们具有相同的MD5值，是非常困难的。

MD5 原理



- 一个**MD5**运算由类似的**64**次循环构成，分成**4**组**16**次。
- ***F*** 一个非线性函数。***Mi*** 表示一个 **32-bits** 的输入数据，***Ki***表示一个 **32-bits** 常数，用来完成每次不同的计算。
- 主循环有四轮，每轮循环都很相似。第一轮进行**16**次操作。每次操作对**a**、**b**、**c**和**d**中的其中三个作一次非线性函数运算，然后将所得结果加上第四个变量，文本的一个子分组和一个常数。再将所得结果向左环移一个不定的数，并加上**a**、**b**、**c**或**d**中之一。最后用该结果取代**a**、**b**、**c**或**d**中之一。

安全通道

安全通道：使客户与服务器之间的通信保持安全，免受对消息的窃听、修改和伪造的攻击

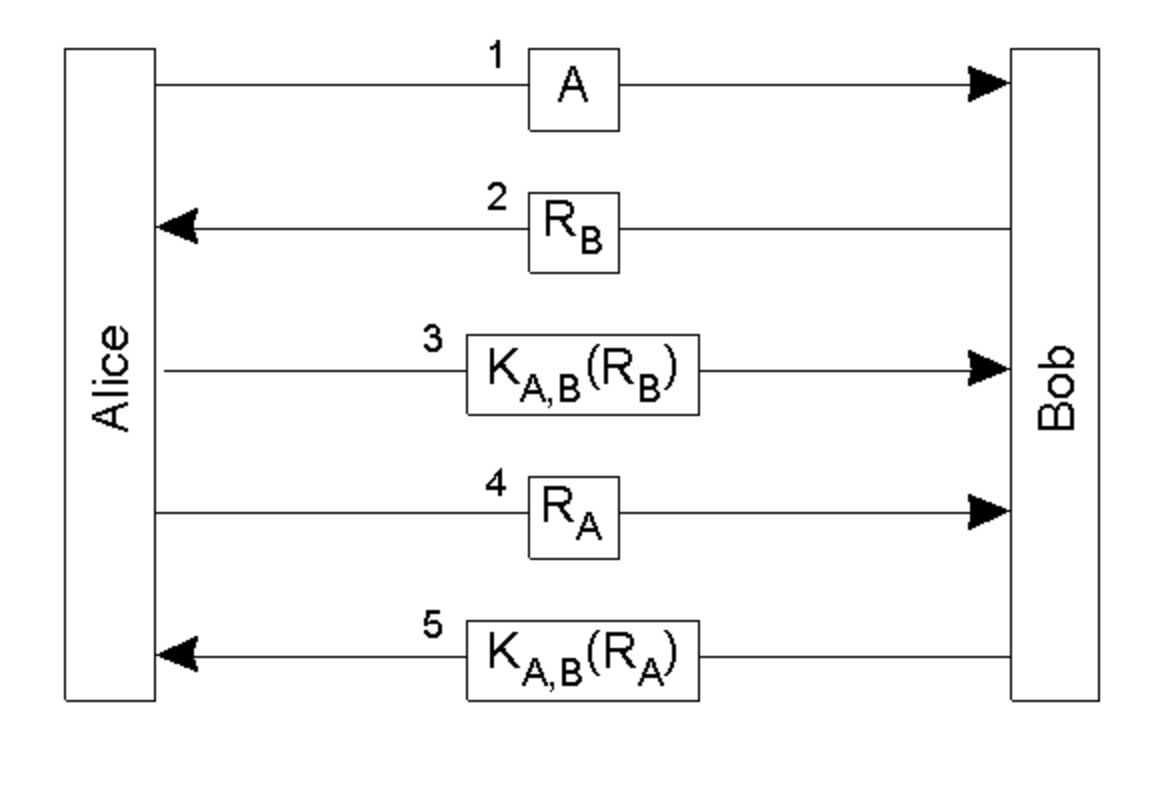
- 身份验证：通信双方需要验证身份
- 消息的完整性和机密性：消息未受到窃听、修改和伪造的攻击

身份验证（Authentication）

- 基于共享密钥的身份验证
- 使用密钥发布中心的身份验证
- 使用公钥加密的身份验证

身份验证 (1)

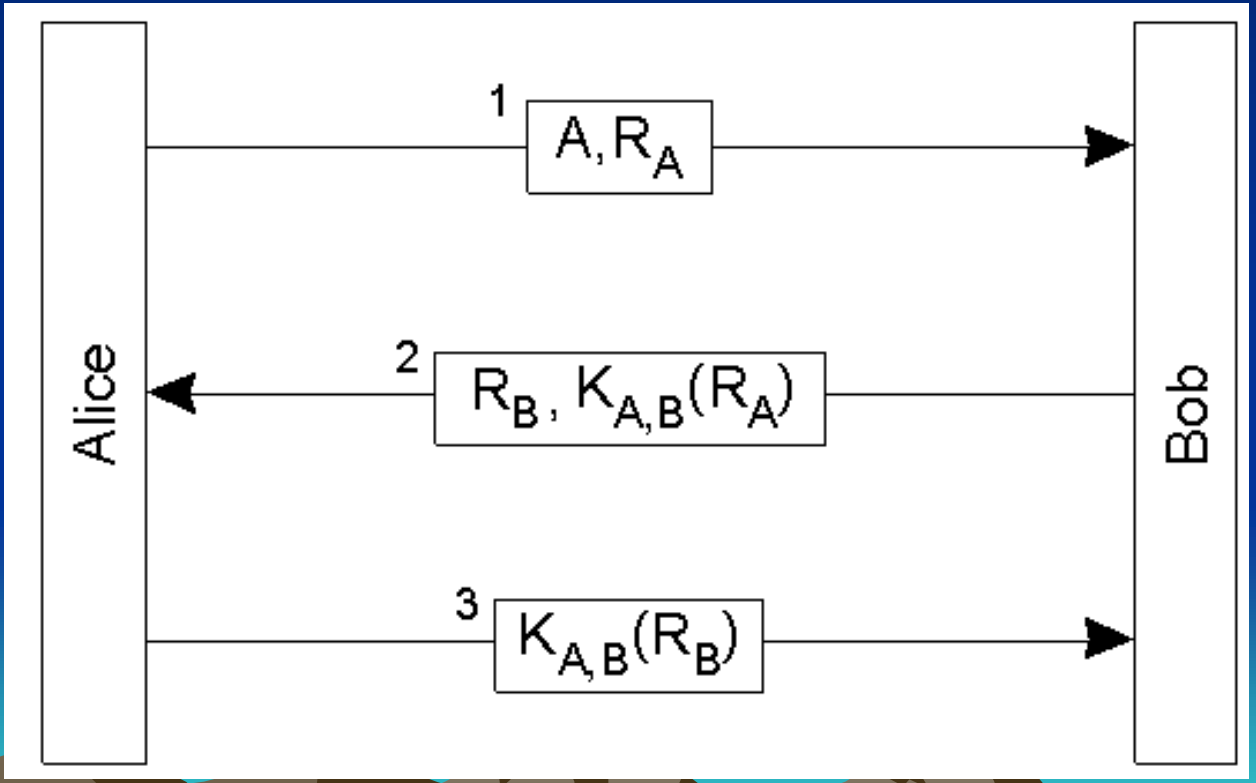
质询-响应协议：一方向另一方质询一个响应，只有对方知道共享密钥时才能给予正确的响应



基于共享密钥的身份验证

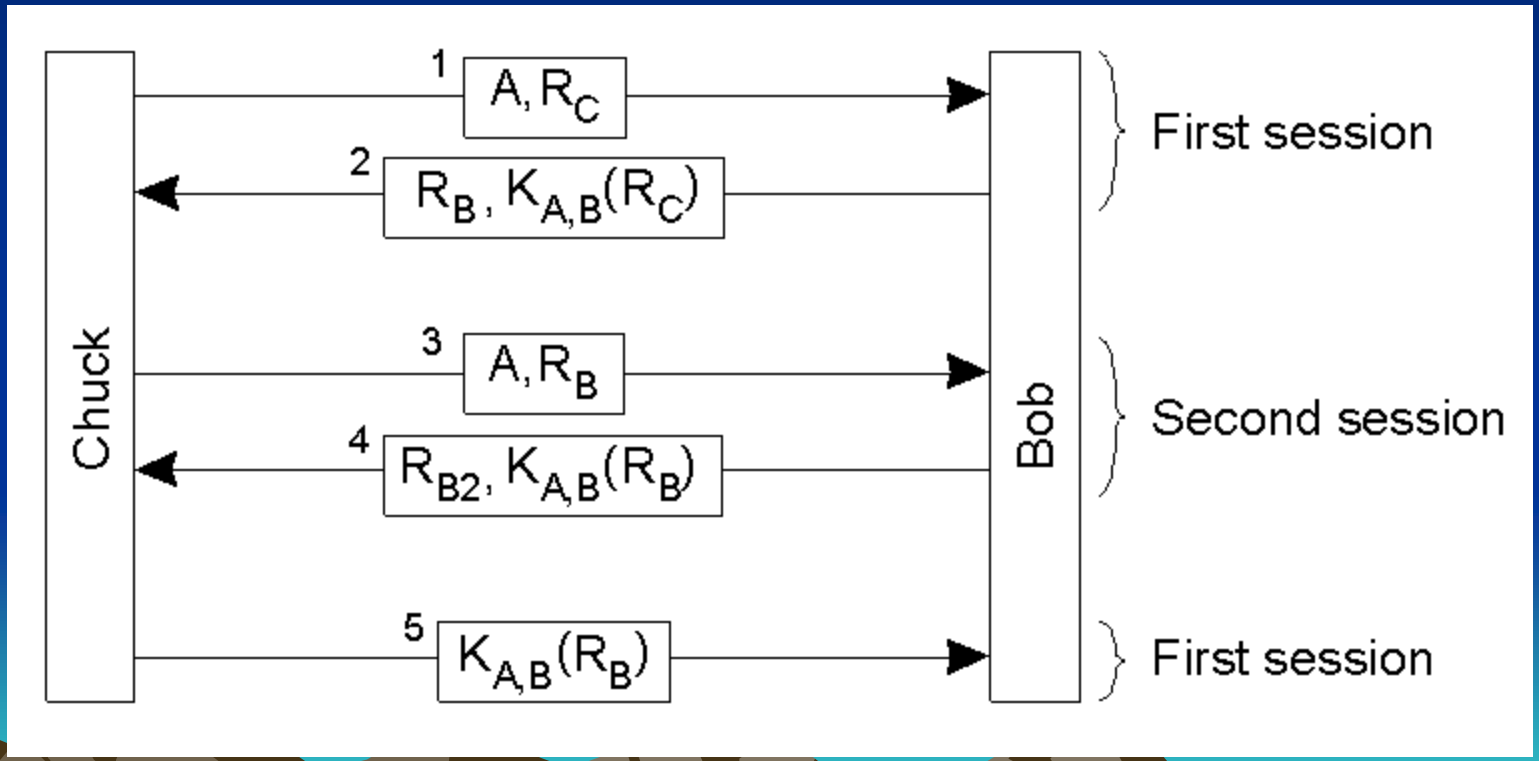
身份验证 (2)

- 基于共享密钥的身份验证，用三个消息代替五个



身份验证 (3)

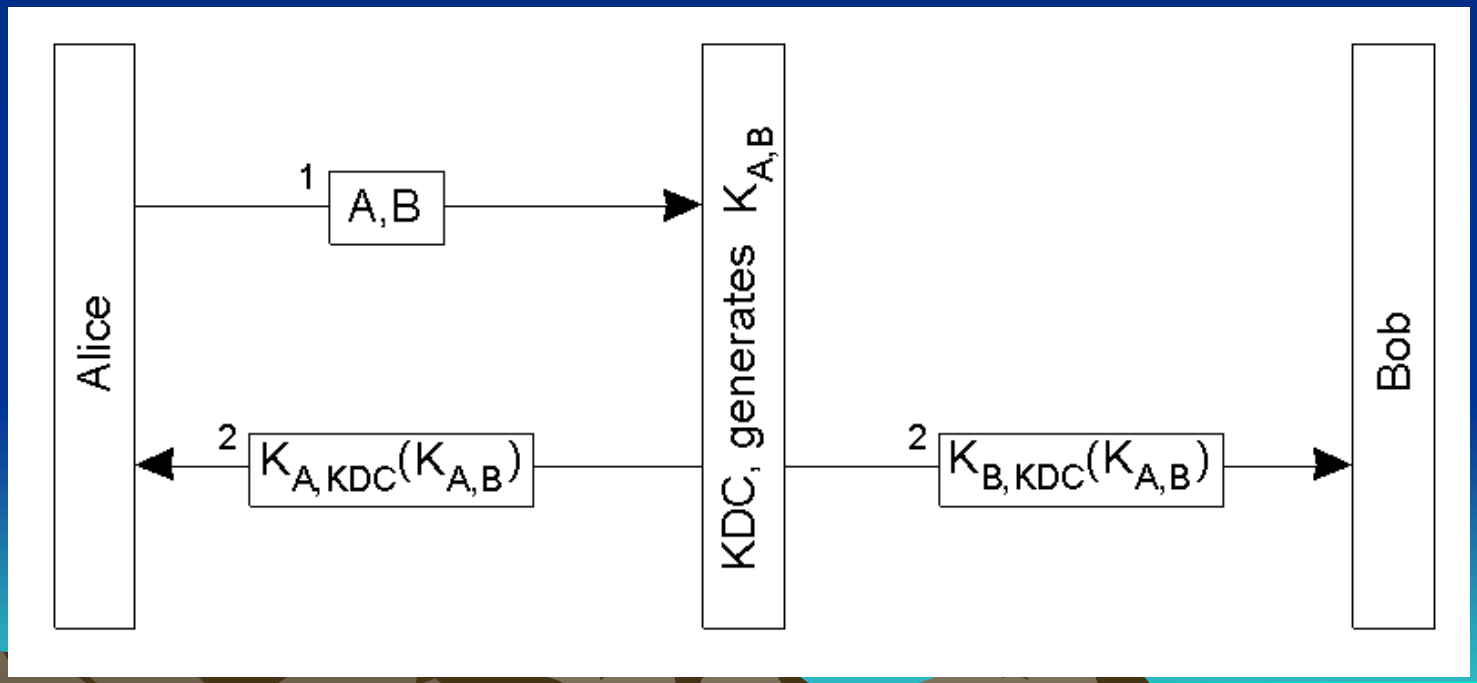
原因：协议的双方在两个不同方向都使用相同的质询
解决：协议的双方永远使用不同的质询



反射攻击（reflection attack）

使用密钥发布中心的身份验证 (1)

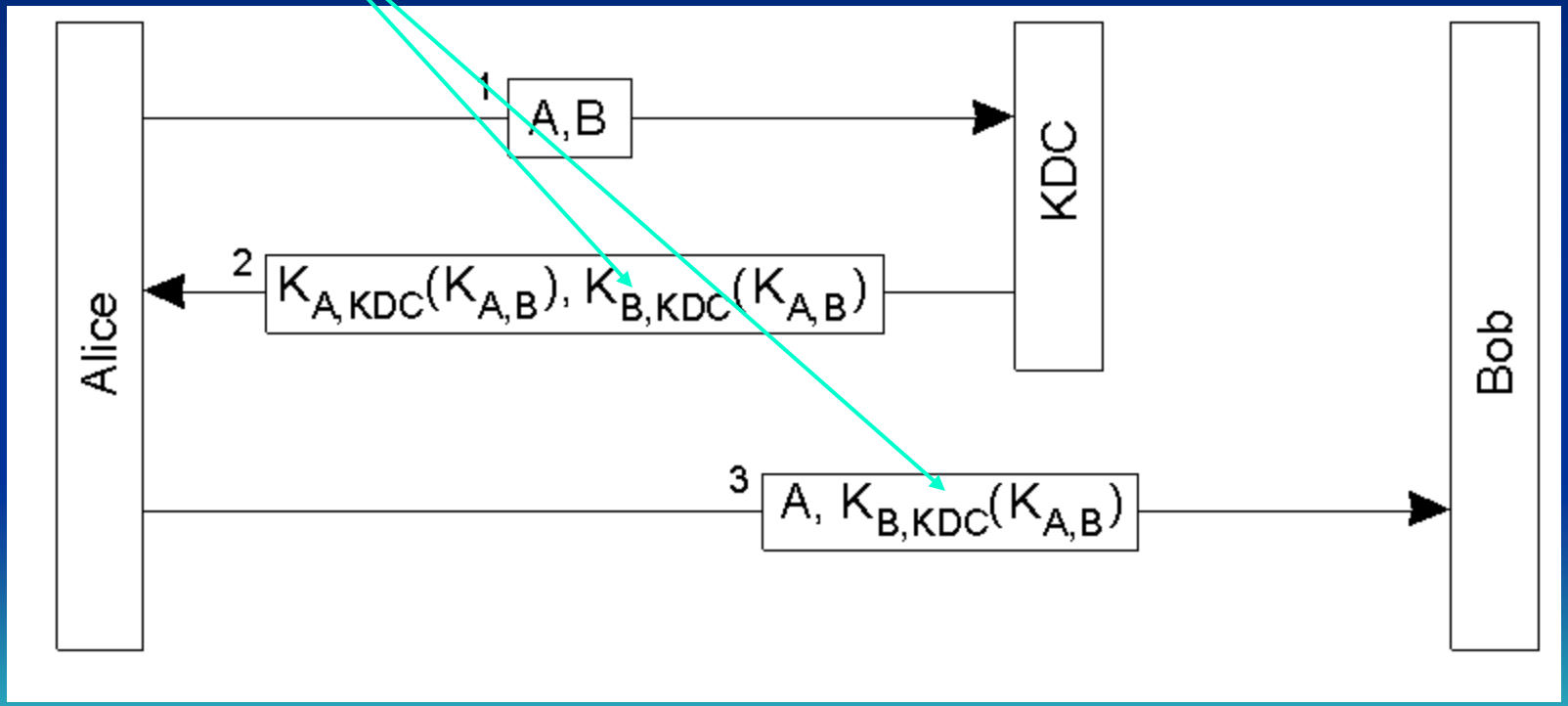
- 基于共享密钥的身份验证存在可扩展性问题：N台主机，需要N*(N-1)/2个密钥
- 使用 KDC（key distribution center）只需要管理 N 个密钥
- KDC与每台主机共享一个密钥；向通信的两主机分发一个密钥通信



使用 KDC（key distribution center）

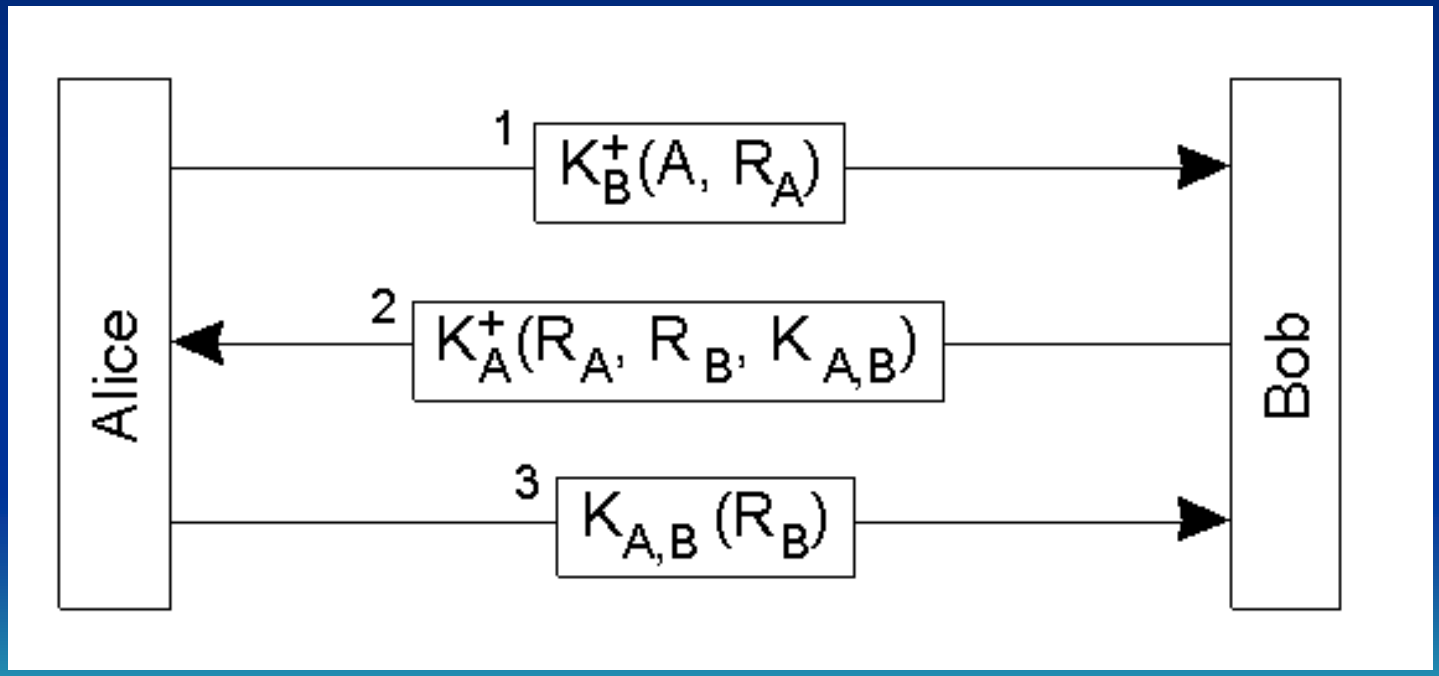
使用密钥发布中心的身份验证 (2)

- 使用票据（ticket）让 Alice 建立与 Bob 的通道



使用公钥加密的身份验证

- 使用公钥加密的身份验证

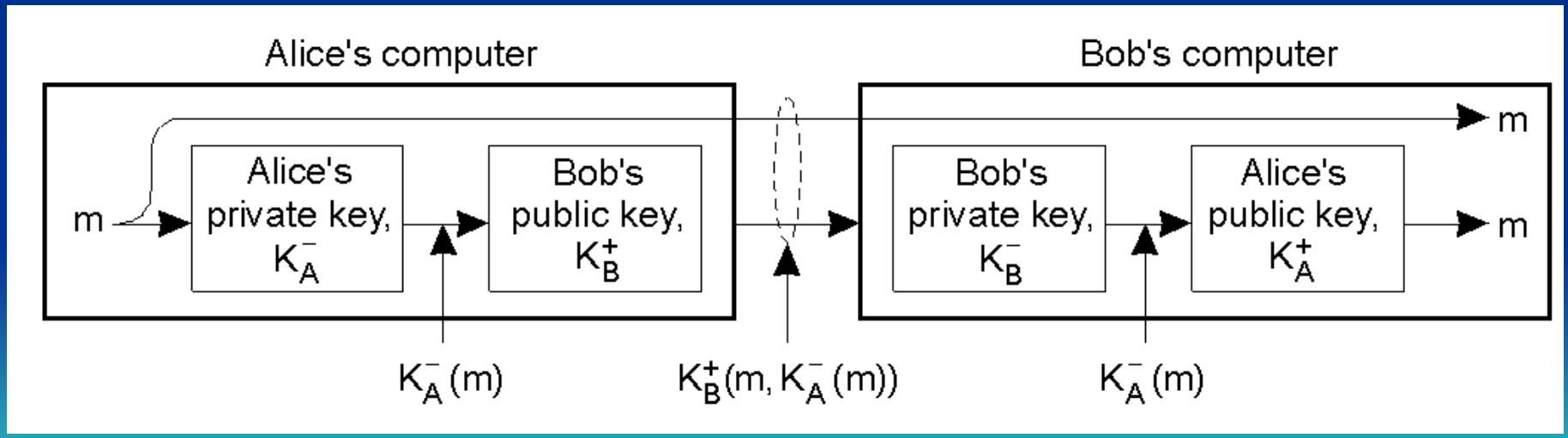


消息的完整性和机密性

- 消息的完整性和机密性：消息不受到窃听、修改和伪造的攻击。
  - 机密性确保窃听者不能截获和读取消息，通过加密实现。
  - 完整性确保消息免受修改。
- 数字签名
  - 如果消息签名检验为真，发送者不能否认消息签名这一事实
  - 消息与其签名的唯一关联防止了对消息进行修改而未发现的可能
  - 使用公钥加密对消息进行数字签名
- 会话密钥

数字签名 Digital Signatures (1)

- **Bob通过比较m与其解密版确定m是否来自Alice**
- **Bob 保留m的签名版：**
  - **防止Alice的否认**
  - **防止Bob对消息的恶意修改：他必须证明修改版本也是由Alice签名的**



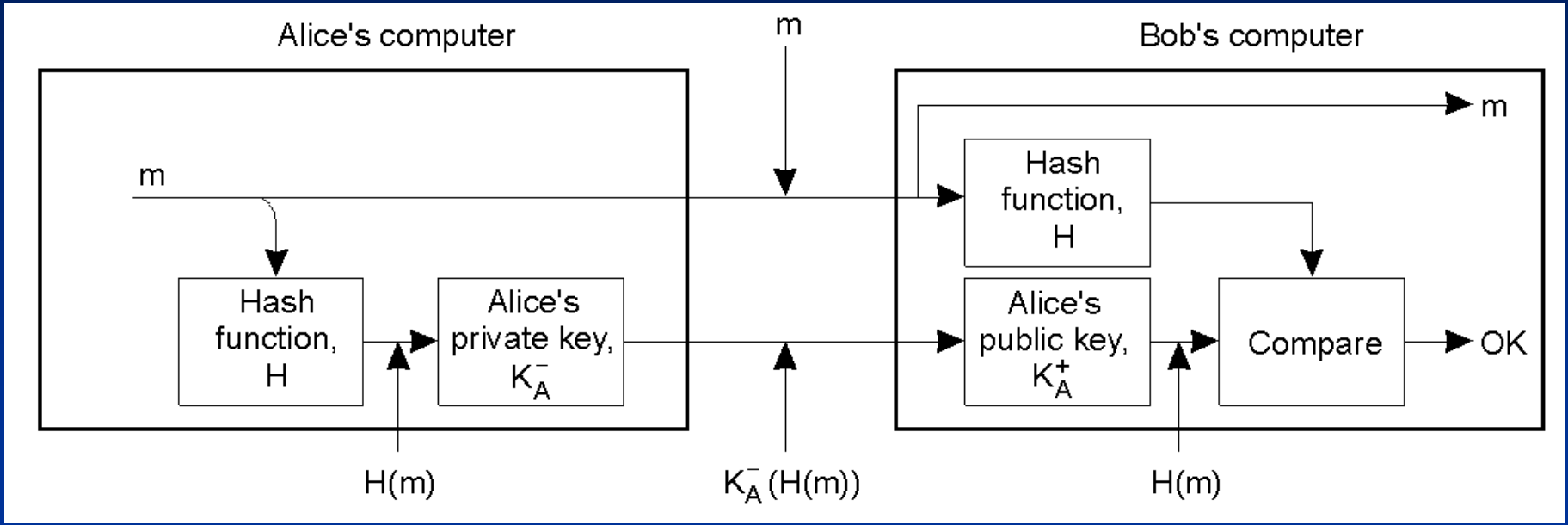
使用公钥加密对消息进行数字签名

数字签名 (2)

存在的问题：

- Alice可以声称她的私钥在消息发送前被盗了
- Alice可能会改变她的私钥
- 使用私钥加密整个消息开销可能很大
  - 使用消息摘要解决，消息摘要是固定长度的位串 h= H(m), m是任意长度的消息，H是加密散列函数

数字签名 (3)



使用消息摘要对消息进行数字签名

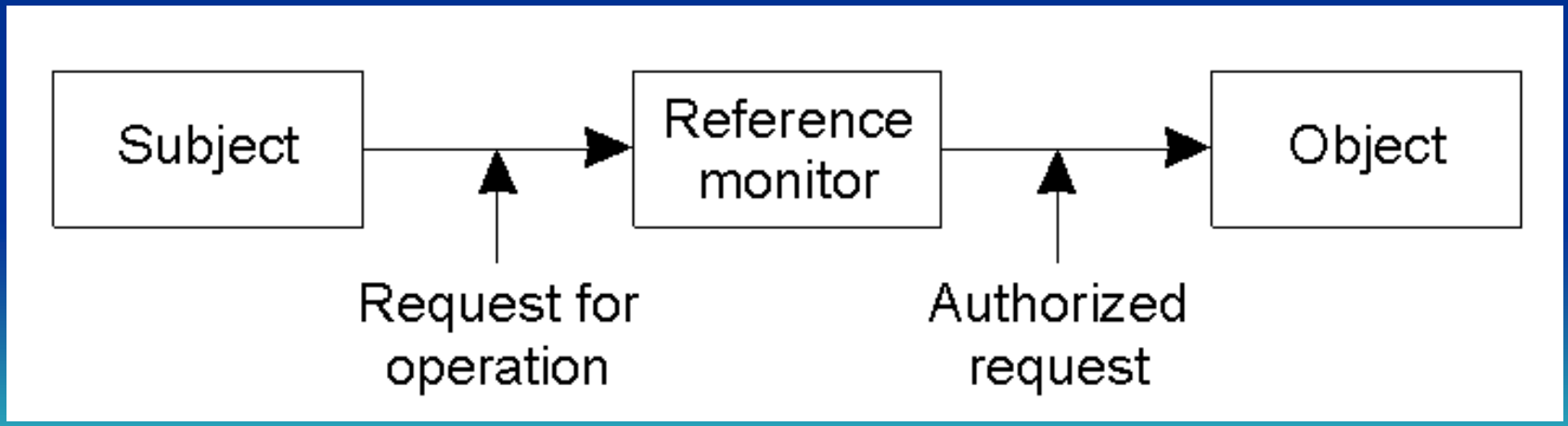
会话密钥

- 在身份验证完成后，通信双方一般使用唯一的共享会话密钥以实现机密性，通信完毕后丢弃
- 也可以使用身份验证密钥
- 但使用会话密钥具有以下优点：
  - 避免经常使用一个密钥
  - 确保通信双方免受重发攻击
  - 会话密钥的成本低：廉价且临时性强

访问控制

建立安全通道后，客户就可以向服务器发送执行的请求，该请求可能会涉及到访问控制

- 访问控制矩阵
- 保护域
- 防火墙
- 保护移动代码



对象访问控制的一般模型

访问控制矩阵

访问控制矩阵

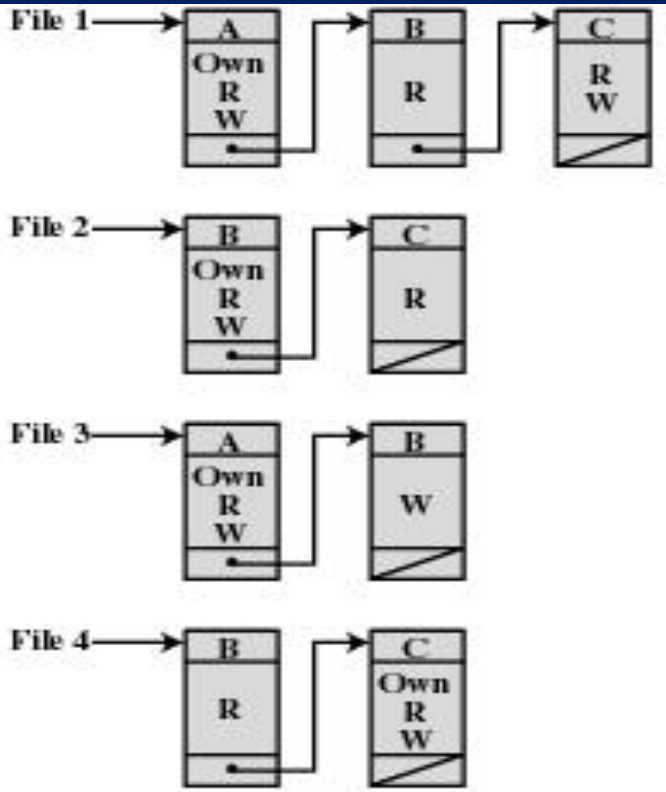
- 按列分解： 访问控制列表（Access control list，ACL)
- 按行分解： 权能表（Capability list）

	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry Credit	
User B	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
User C	R W	R		Own R W		Inquiry Debit

(a) Access matrix

Figure 15.4 Example of Access Control Structures

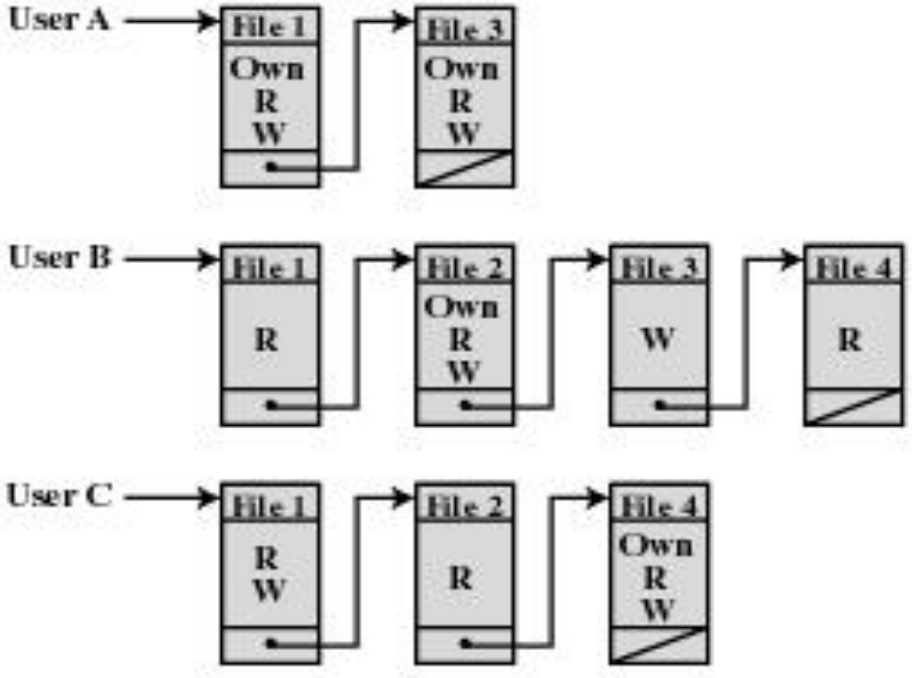
访问控制列表 ACL



(b) Access control lists for files of part (a)

Figure 15.4 Example of Access Control Structures

权能表（Capability list）

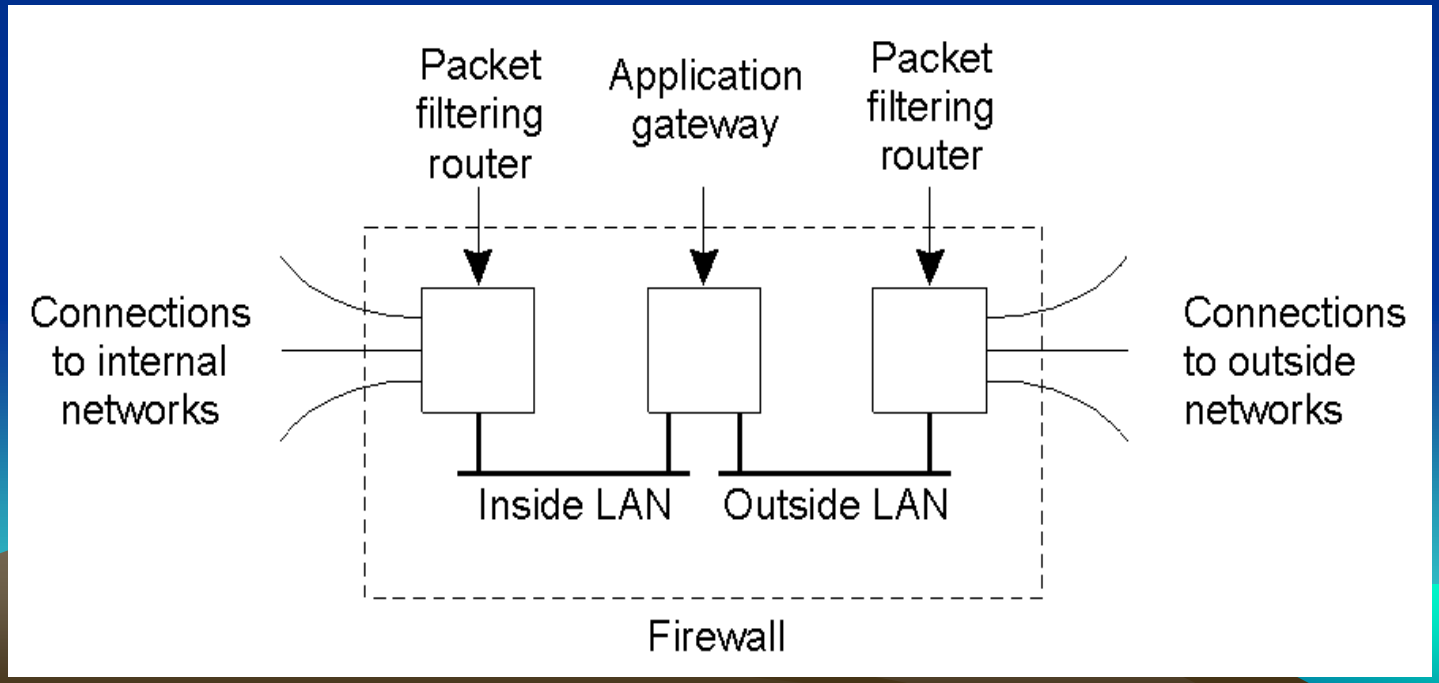


(c) Capability lists for files of part (a)

Figure 15.4 Example of Access Control Structures

防火墙

- 数据包过滤网关：基于数据包报头包含的原地址和目的地址制定是否传送该包的决定
- 应用层的网关：检查入站或出站的消息内容：邮件网关、数据图书馆
- 两者经常结合使用



保护移动代码

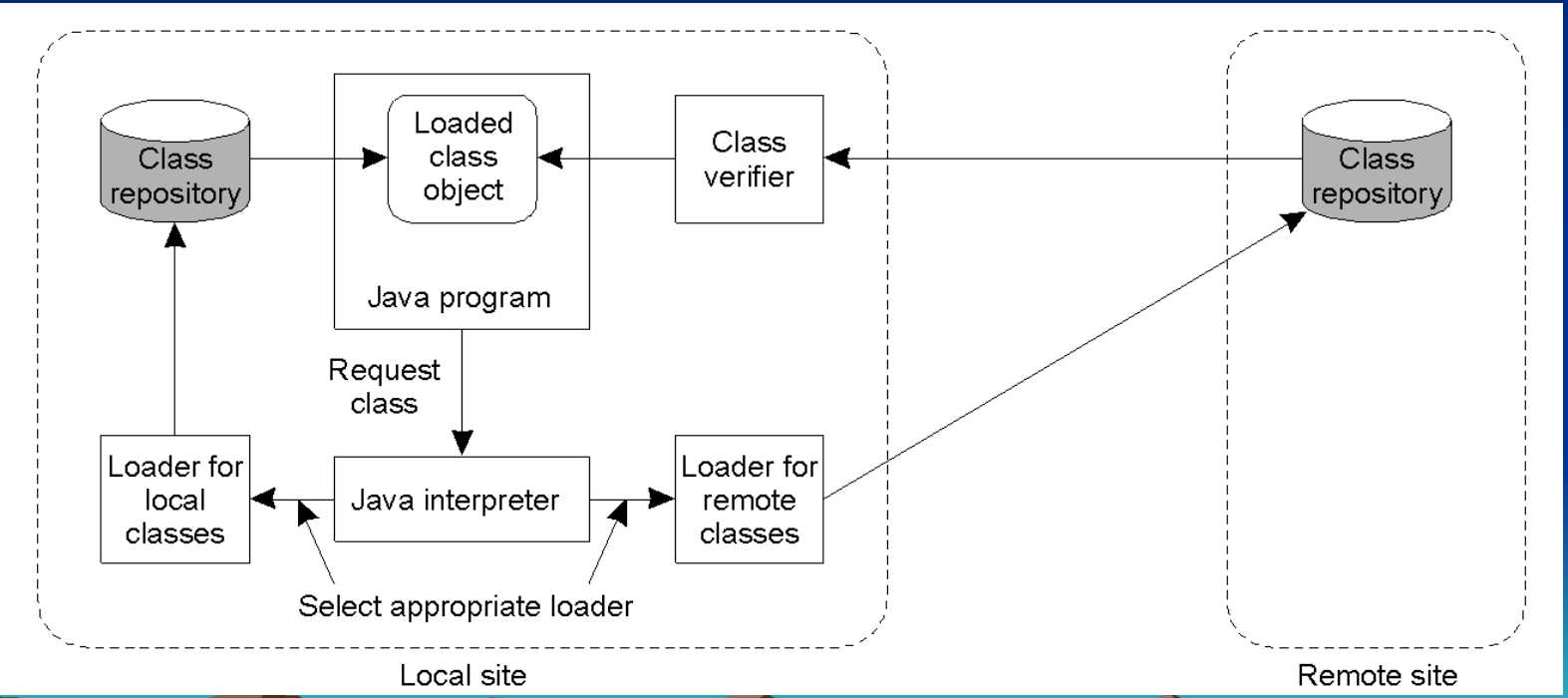
- 保护代理：防止恶意主机盗窃或修改代理程序所带的信息
  - 攻击方式：
    - 盗窃或修改代理程序所带的信息：订票代理携带电子信用卡
    - 恶意破坏代理程序
    - 篡改代理程序以便在其返回时进行攻击或盗窃信息
  - 至少可以检测出代理程序被修改
    - 只读状态
    - 只追加状态
    - 有选择地揭示状态：数据只允许特定的服务器访问
- 保护目标：保护主机防止恶意代理程序的破坏
  - 沙箱
  - 运动场
  - 通过身份验证

沙箱

- 沙箱(sandbox)是一种技术，通过该技术下载的程序的每条指令都能够被完全控制
- 访问一些指令、某些寄存器或内存空间将被禁止
- 通过沙箱模型, 用户可以有效地阻止那些具有潜在危险性的活动, 如对本地硬盘读写、 创建新进程、连接动态链接库等。

沙箱

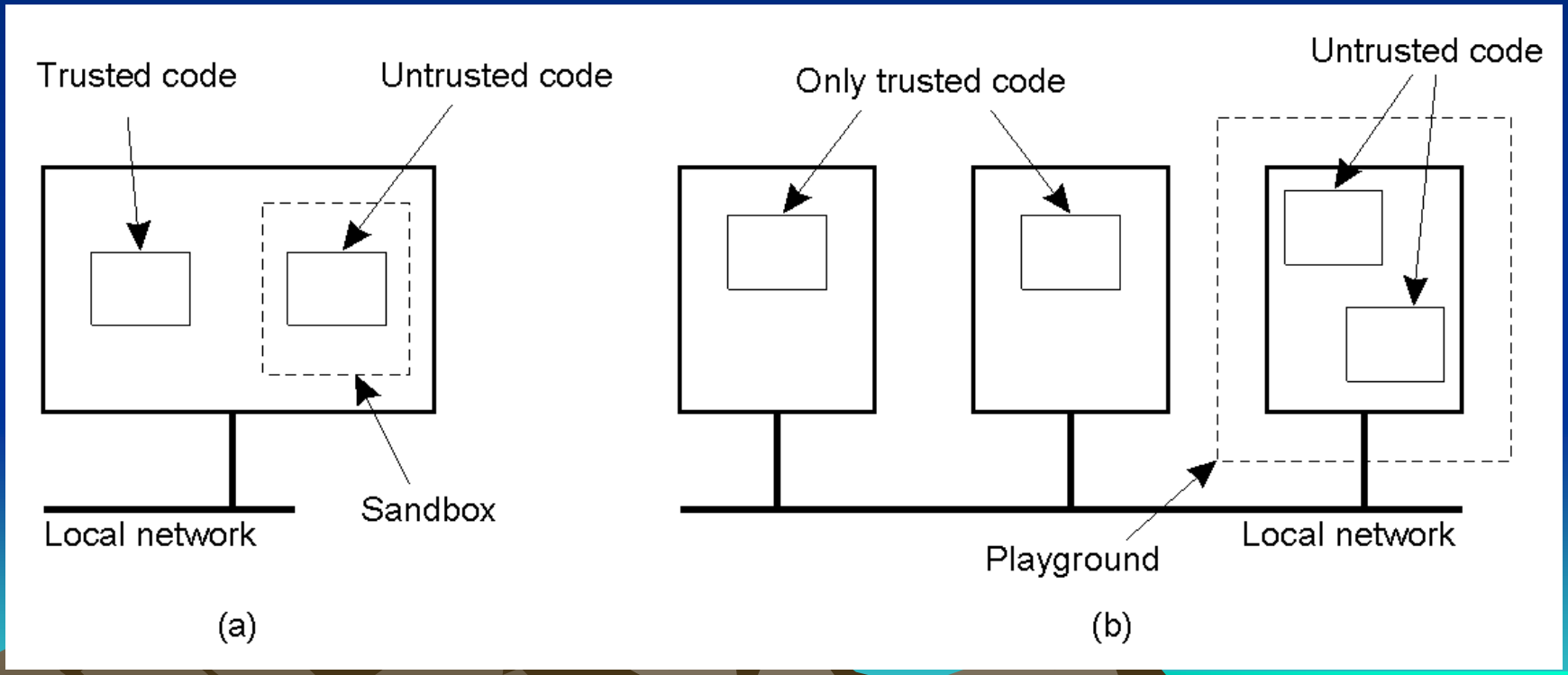
- 类加载器：负责从服务器获取指定的类
- 字节代码验证器：验证下载的类是否符合沙箱的安全规则
- 安全管理程序：在运行时执行各种检查，通常拒绝访问本地文件



一个 Java 沙箱（sandbox）的组织

运动场

a) 沙箱
b) 运动场：符合沙箱但更灵活，为运行移动代码专门保留指定的机器，可使用这些机器的本地资源；但不能访问其他机器的资源



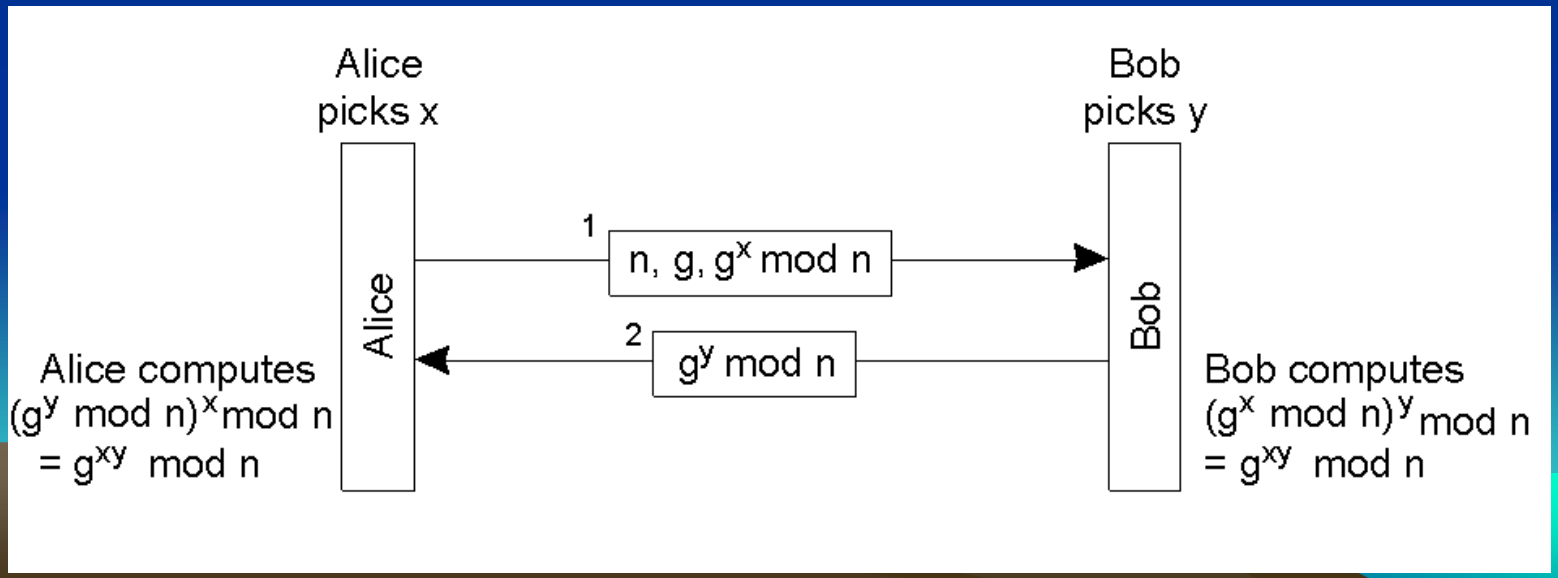
安全管理

- 密钥管理
  - 密钥建立
  - 密钥分发
  - 证书的生存期
- 授权管理
  - 权能和属性证书
  - 委派

密钥管理--密钥建立

Diffie-Hellman 建立共享密钥的原理：

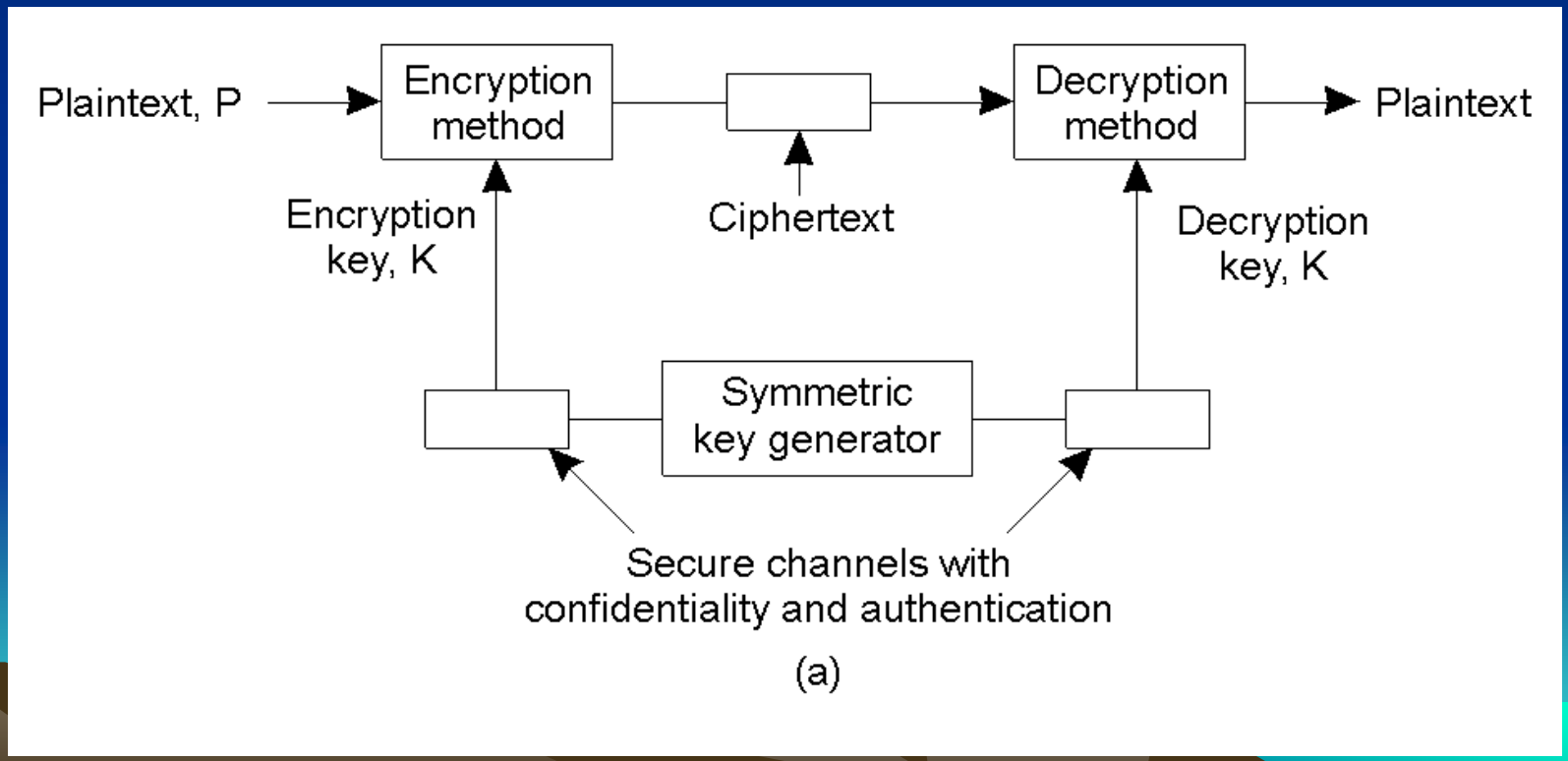
- 首先，Alice和Bob双方约定2个大整数n和g,其中1<g<n，这两个整数无需保密，然后，执行下面的过程：
- Alice随机选择一个大整数x (保密)，并计算$X=g^x \bmod n$
- Bob随机选择一个大整数y (保密)，并计算$Y=g^y \bmod n$
- Alice把X发送给B,B把Y发送给ALICE
- Alice计算$K=Y^x \bmod n = (g^y \bmod n)^x \bmod n = g^{xy} \bmod n$
- Bob计算$K=X^y \bmod n = (g^x \bmod n)^y \bmod n = g^{xy} \bmod n$
- K即是共享的密钥。
- 监听者在网络上只能监听到X和Y，但无法通过X，Y计算出x和y，因此，无法计算出K



密钥分发 (1)

共享密钥分发

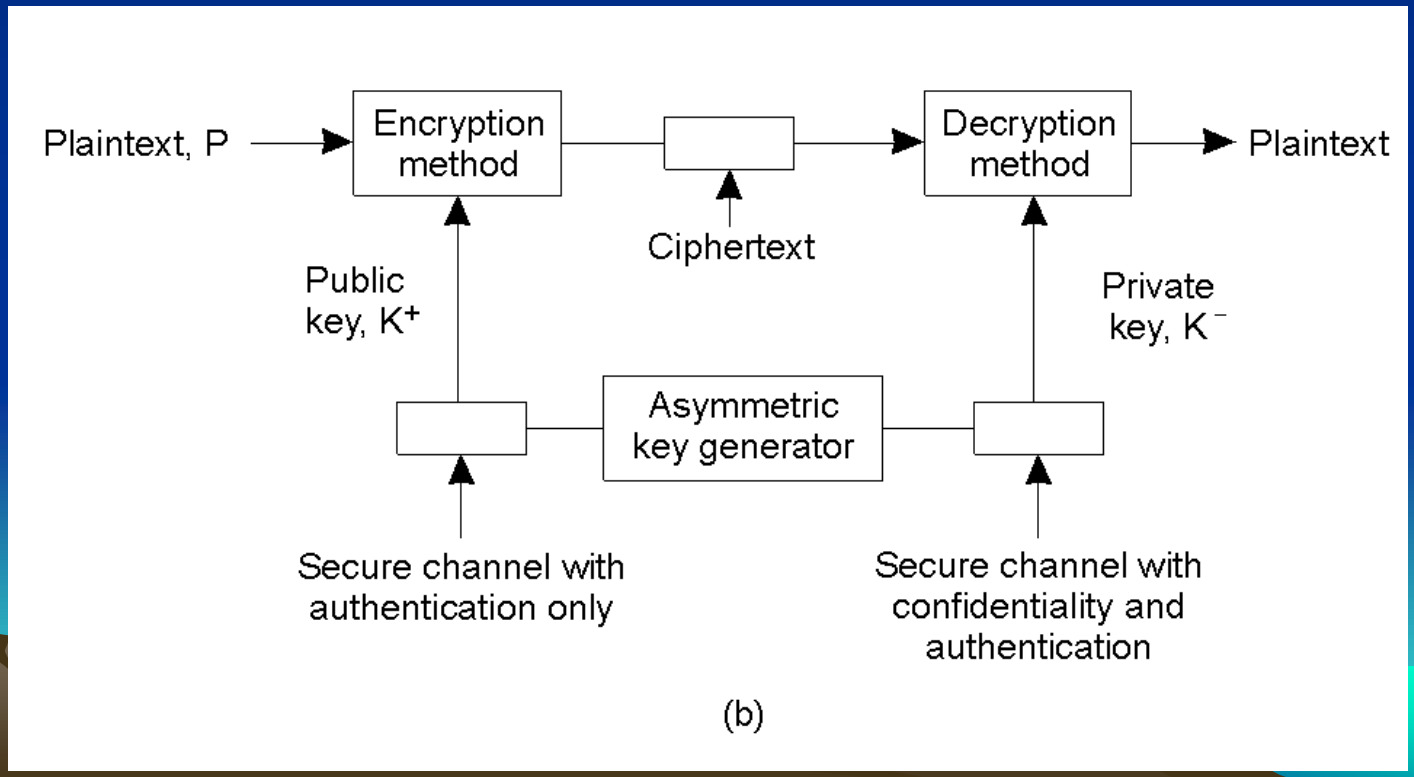
- 必须有提供身份验证和机密性的安全通道
- 带外分发：电话、邮递



密钥分发 (2)

公钥加密系统分发：

- 私钥发送使用提供身份验证和机密性的安全通道
- 公钥发送使用提供身份验证的安全通道：接收者能够确信该密钥肯定可以与声明的一个私钥配对
- 公钥证书：（公钥，公钥关联的实体（用户等）），由认证机构签发，使用该机构的私钥进行数字签名



证书的生存期

- 终生证书
- 吊销证书
  - CLR（certificate revocation list）证书吊销表
  - 限制证书的生存期
  - 缩短证书的生存期为零，客户需要一直联系证书颁发机构以检验公钥的有效性

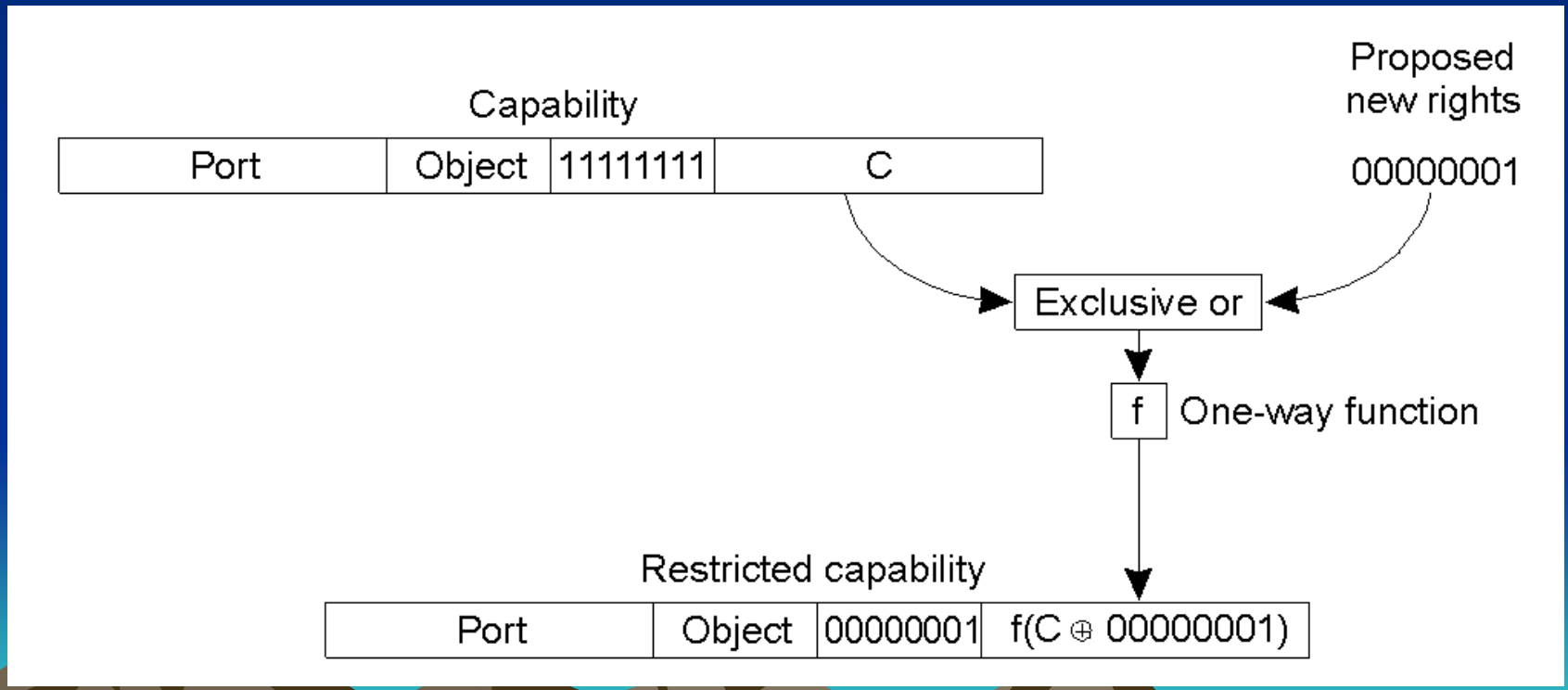
授权管理--权能和属性证书 (1)

- 为了调用一个对象的操作，客户必须将权能传给服务器检查

Amoeba 中的权能

权能和属性证书 (2)

- 服务器创建对象时，客户得到是所有者权能（全1），check字段是随机选择的，同时存储在权能和服务器的一个表中
- 客户可以从一个所有者权能生成一个受限权能，并发送给另一进程



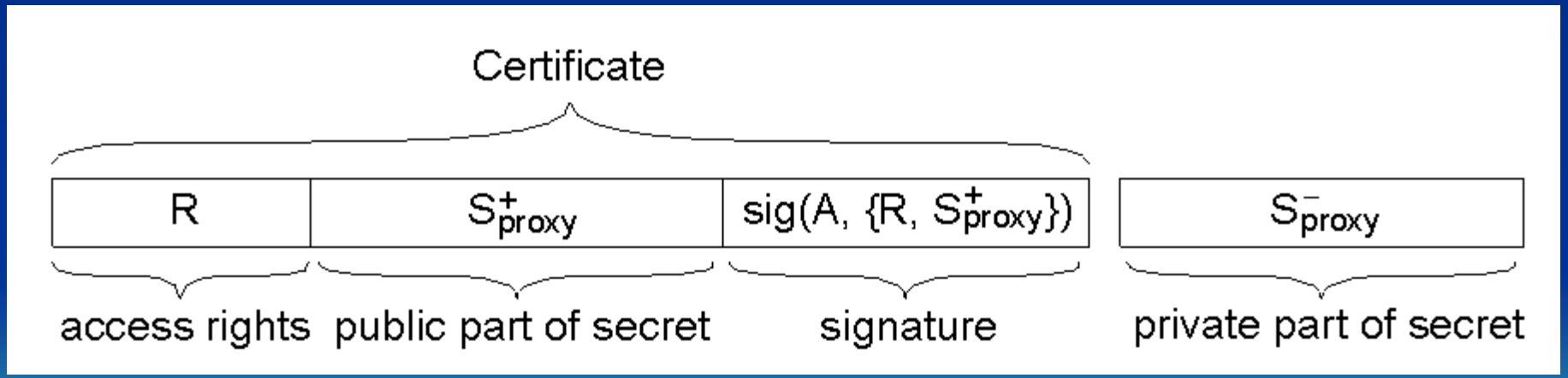
从一个所有者权能生成一个受限权能

委派 (1)

委派：将某些访问权限从一个进程传递给另一个进程

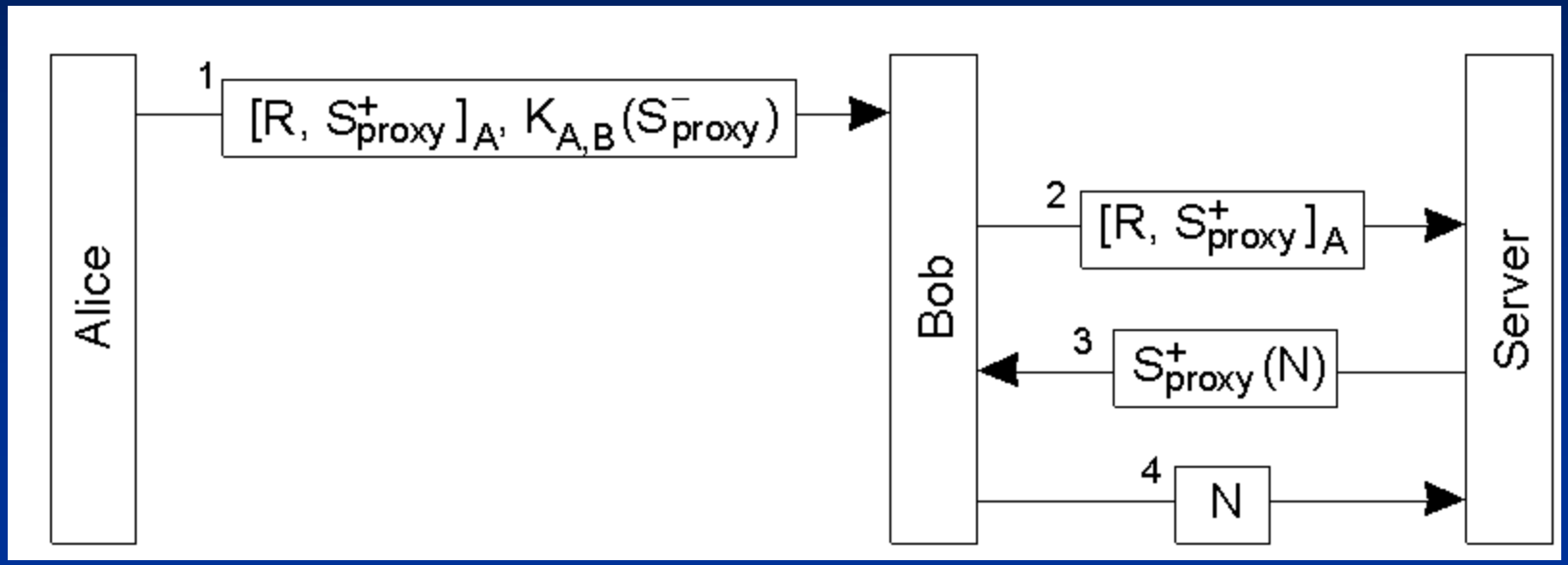
Alice可以构造证书：

- Bob具有权限R
- 此证书的持有者具有权限R



用于委派的一般代理结构

委派 (2)



使用代理来委派和证实访问权限的所有权

实例: Kerberos

- Kerberos是MIT大学在20世纪80年代开发的为MIT校园网和其他企业内部的网提供的一系列认证和安全措施。
- Windows 2000包含Kerberos的实现
- 名称来源于希腊神话，Kerberos是地狱入口的守护者，通常有三个头。设计者的设计初衷是要用Kerberos的三个头来守卫网络之门。
- Kerberos的核心概念是：
  - Ticket, Authenticator, Session key

Kerberos 要解决的问题

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提供的服务，存在很多问题：

- 工作站上的用户可以冒充另一用户操作。
- 用户可以改变工作站地址冒充另一台工作站。
- 用户可以窃听并回放他人的信息交换，获得对于某种服务的访问权或中断服务的运行。
- 使用假冒服务器从而骗得用户的机密信息。

Kerberos 的解决方案

- 在一个分布式的Client/Server体系机构中，引入一个可信任的第三方（Kerberos身份验证服务器），让其提供认证服务。
- 采用共享密钥加密技术。

Kerberos 基本性质

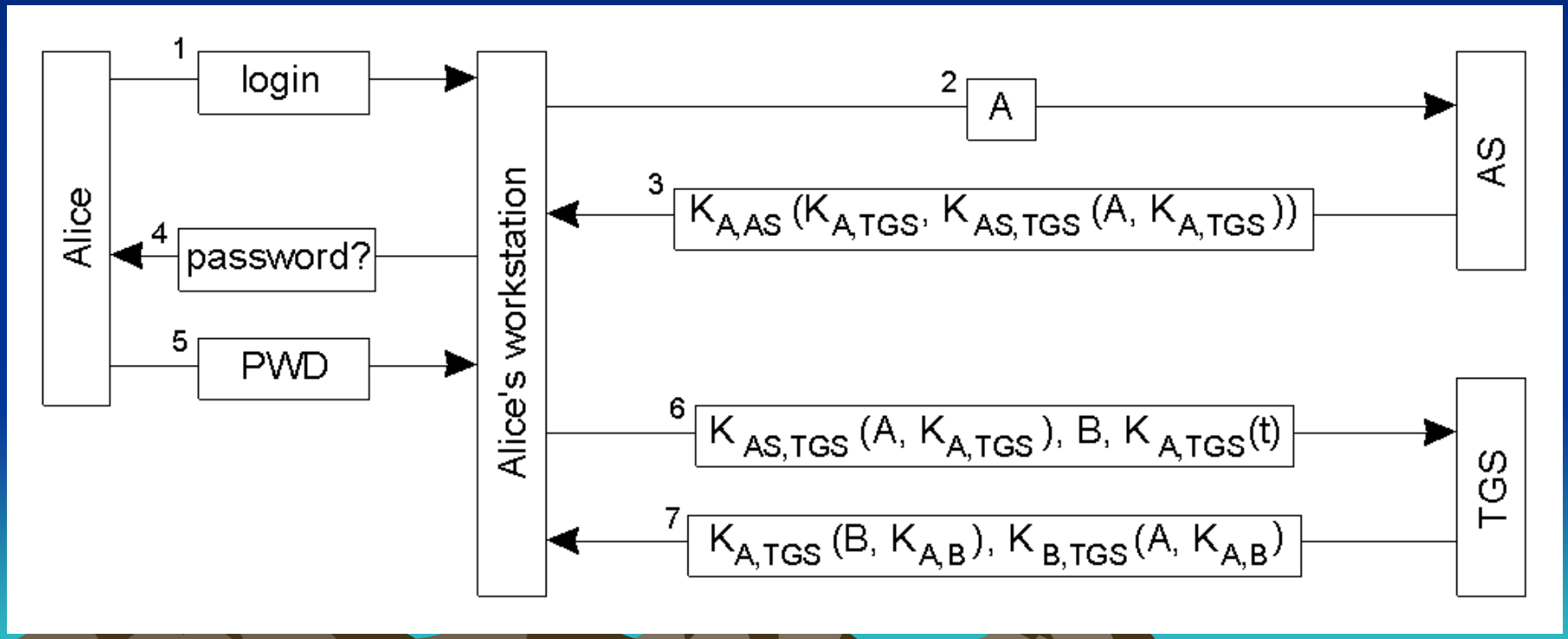
- 安全：使网络窃听者不能获得必要的信息来伪装成另一客户。
- 可靠：对所有以Kerberos进行访问控制的服务来说，客户无法通过Kerberos身份验证服务器的认证就意味着无法获得所需要的服务。
- 透明：用户除了需要输入一个口令外，不必知道鉴别过程的存在以及细节。
- 可伸缩：可支持大量用户和服务器。（采用模块化、Client/Server结构）

Kerberos 应用

- **Kerberos 第5版是Windows 2000 中最基本的安全协议，是默认的用户和服务器验证提供者。**
- **Linux 、Unix 类系统也都支持Kerberos协议。**
- 作为网络环境的双向鉴别协议。

实例: Kerberos

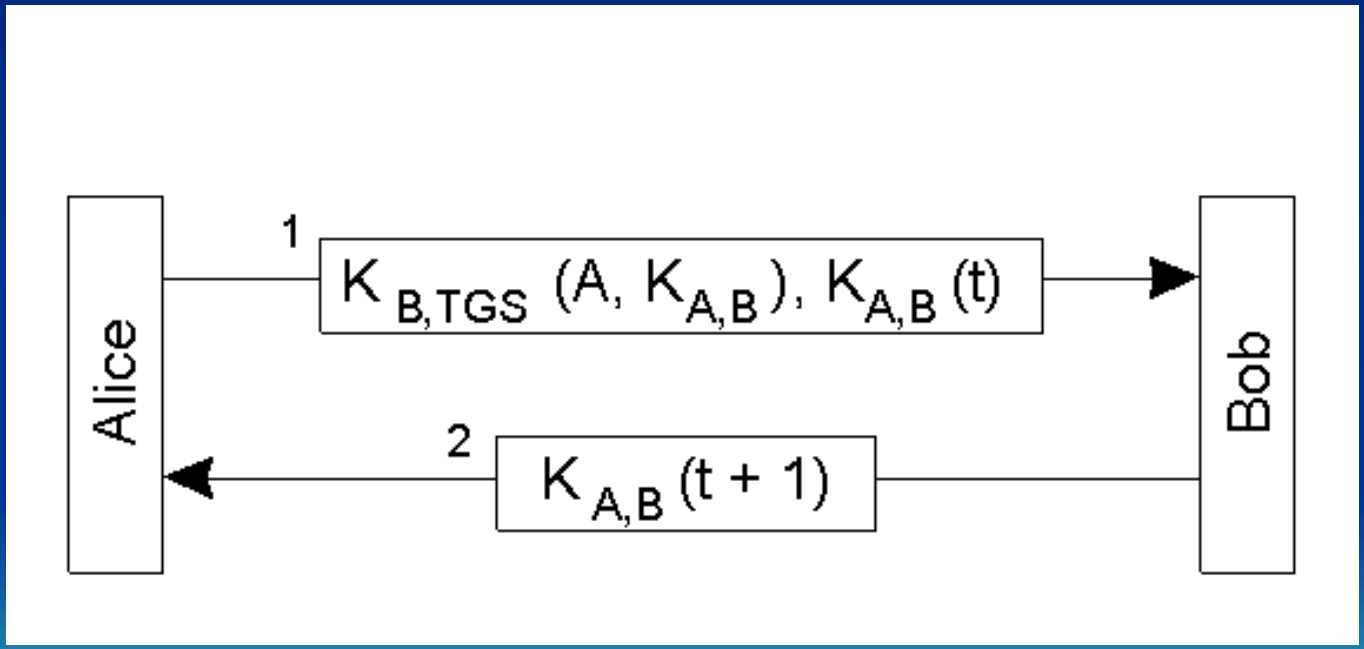
- 身份验证服务器AS：处理客户的登录请求；AS对用户身份验证,并提供一个建立安全通道的密钥
- 票据授予服务TGS：建立安全通道；分发称为票据的特殊信息，用于使服务器确信该客户正是其所声称的那个客户



Kerberos的身份验证

实例: Kerberos

- 在 **Kerberos**中构建安全通道



局限性分析

- **Kerberos服务器易受攻击**
  - 它的安全性决定了整个系统得安全性，若此关键环节发生问题，危害是灾难性的。
- 口令攻击
  - 对手截获基于口令的密钥加密的内容，采用暴力破解成功后，得到口令也就到该用户的全部资源

资 源

- 掌握Kerberos概念的简单方法请参考：
  http://web.mit.edu/kerberos/www/dialogue.html
- MIT Kerberos 站点
  http://web.mit.edu/kerberos/
- USC/ISI Kerberos 站点
  http://www.isi.edu/gost/info/kerberos/

小结

- 分布式安全简介
  - 安全模型
  - 加密技术
- 安全通道
  - 身份验证：通信双方需要验证身份
  - 消息的完整性和机密性：消息未受到窃听、修改和伪造的攻击
- 访问控制
  - 访问控制矩阵
  - 保护域
  - 防火墙
  - 保护移动代码
- 安全管理
  - 密钥管理
  - 授权管理
- 实例: **Kerberos**

习 题

- Assume Alice wants to send a message *m* to Bob. Instead of encrypting *m* with Bob's public key K_B^+, she generates a session key $K_{A,B}$ and then sends $[K_{A,B}(m), K_B^+(K_{A,B})]$.Why is this scheme generally better?

  (*Hint:* consider performance issues.)

- The Diffie-Hellman key-exchange protocol can also be used to establish a shared secret key between three parties. Explain how.
- There is no authentication in the Diffie-Hellman key-exchange protocol. By exploiting this property, a malicious third party, Chuck, can easily break into the key exchange taking place between Alice and Bob, and subsequently ruin the security. Explain how this would work.

Diffie-Hellman算法受到中间人攻击