

身份鉴别和访问控制



实体鉴别的目的和过程

- **实体鉴别（身份鉴别）**：某一实体确信与之打交道的实体正是所需要的实体。只是简单地鉴别实体本身的身份，不会和实体想要进行何种活动相联系。
- **在实体鉴别中**，身份由参与某次通信连接或会话的远程参与者提交。这种服务在连接建立或在数据传送阶段的某些时刻提供，使用这种服务可以确信(仅仅在使用时间内)：一个实体此时没有试图冒充别的实体，或没有试图将先前的连接作非授权地重演。



实体鉴别分类

- 实体鉴别可以是单向的也可以是双向的。
 - 单向鉴别是指通信双方中只有一方向另一方进行鉴别。
 - 双向鉴别是指通信双方相互进行鉴别。



实现身份鉴别的途径

- 三种途径之一或他们的组合
 - (1) 所知 (**Knowledge**): 密码、口令
 - (2) 所有 (**Possesses**): 身份证、护照、信用卡、钥匙
 - (3) 个人特征: 指纹、笔迹、声纹、手型、血型、视网膜、虹膜、**DNA**以及个人动作方面的一些特征

设计依据:

安全水平、系统通过率、用户可接受性、成本等



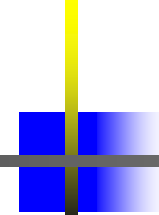
非密码的鉴别机制

- A.** 口令机制
- B.** 一次性口令机制
- C.** 询问—应答机制
- D.** 基于地址的机制
- E.** 基于个人特征的机制
- F.** 基于设备的鉴别



A. 口令机制

- 常规的口令方案涉及不随时间变化的口令，提供所谓的弱鉴别(**weak authentication**)。
- 口令或通行字机制是最广泛研究和使用的身份鉴别法。通常为长度为**5~8**的字符串。选择原则：易记、难猜、抗分析能力强。

- 
- 禁止不必要的用户登录服务器。如邮件服务器不应该给用户使用**SHELL**的权利;
 - 尽可能强制新添加的用户在第一次登录时修改密码;
 - 保护好系统中密码配置文件不被窃取;
 - 禁止**Root**远程登录;少用**Telnet**或安装**SSL**加密**Telnet**信息;
 - 保护用户名不要泄露。因为登录一台机器需要两个部分—用户名和口令。
 - 不要将多台计算机的用户密码设成一样的,防止黑客攻破一台机器后就可攻击所有机器。



对付口令猜测的措施

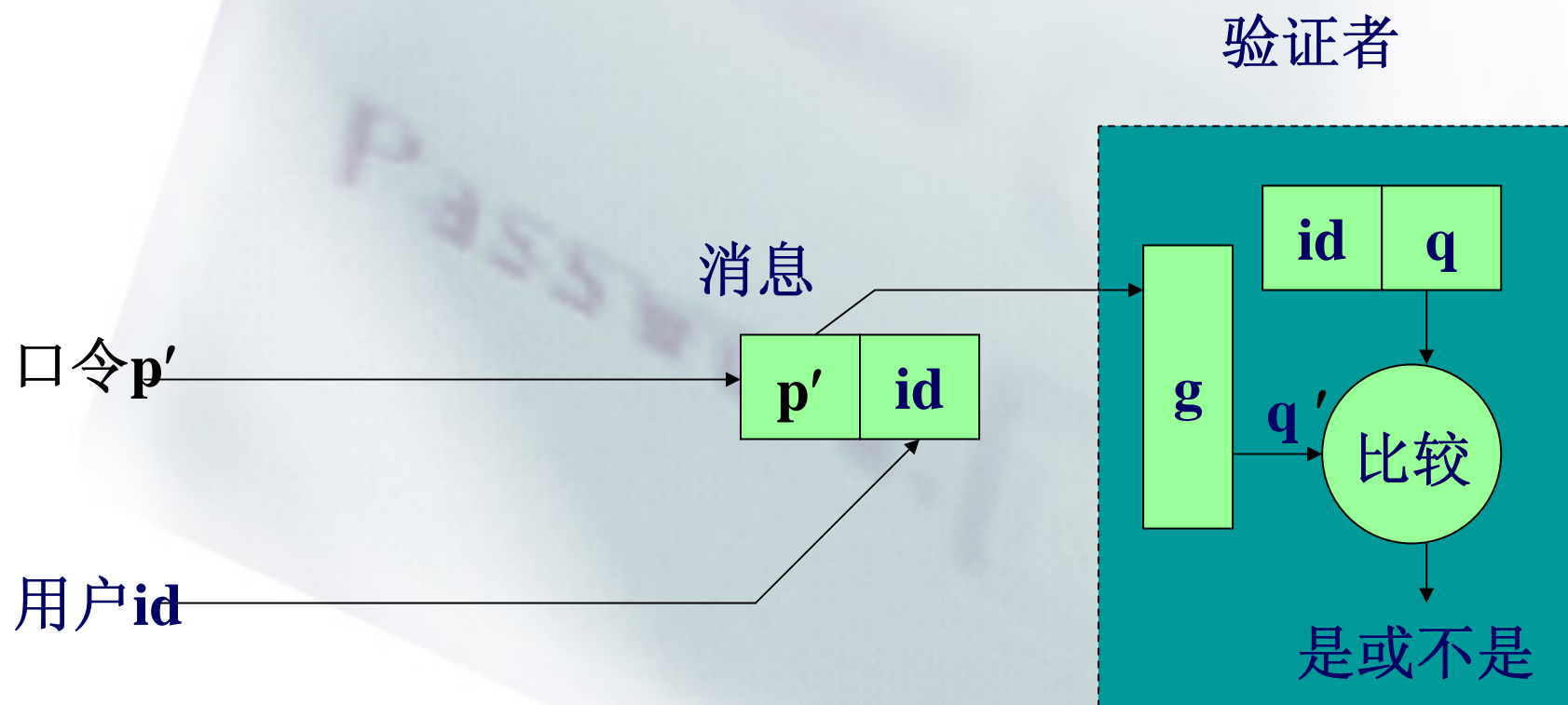
- 严格限制非法登录的次数；
- 口令验证中插入实时延迟；
- 限制最小长度，至少**6~8**字节以上
- 防止用户特征相关口令，
- 口令定期改变；
- 及时更改预设口令；
- 使用机器产生的口令。



口令带来的问题

- 窃听
- 口令数据文件被获取
- 口令猜测
- 重放
- 其他问题

保护口令：单向函数

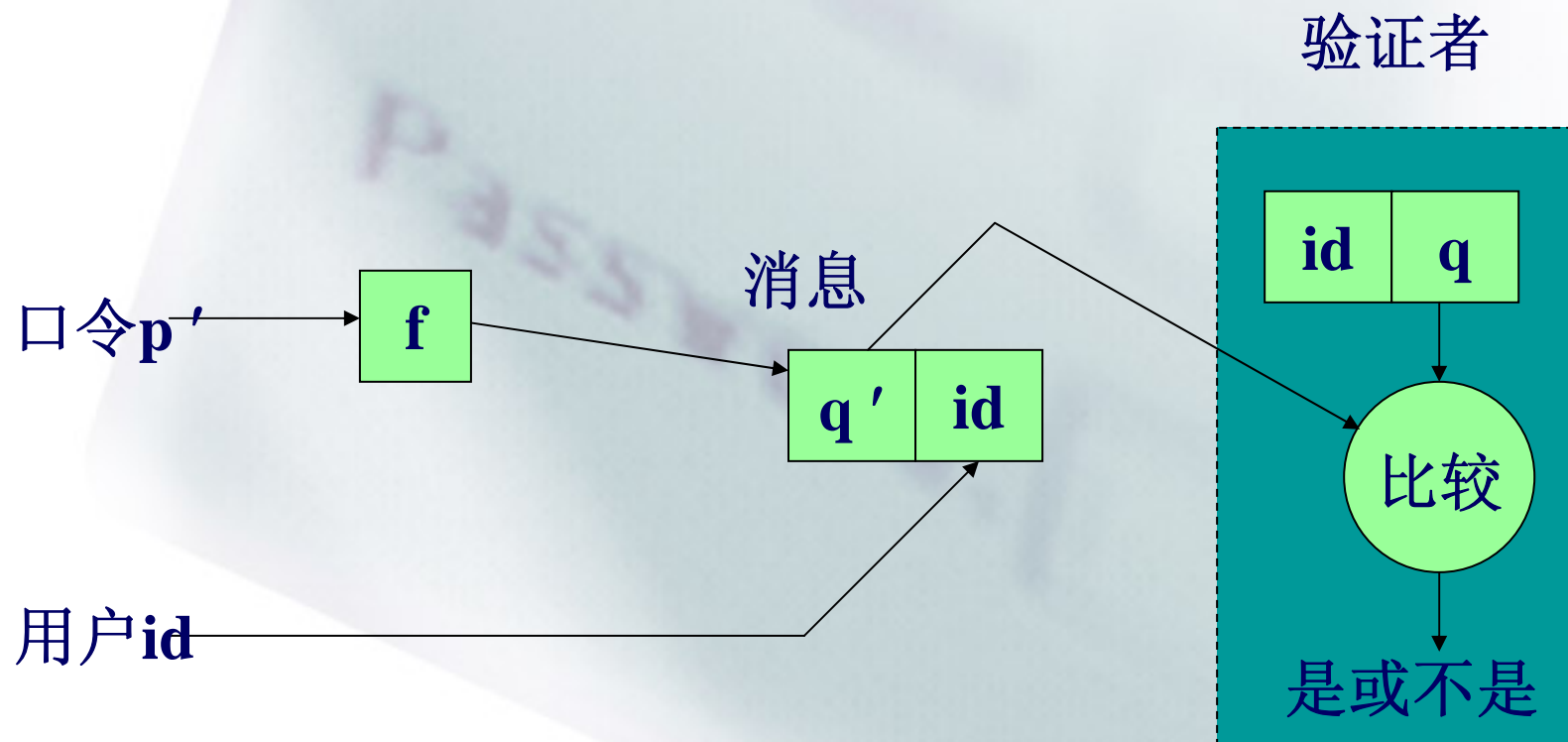




SAM (Security Accounts Manager)

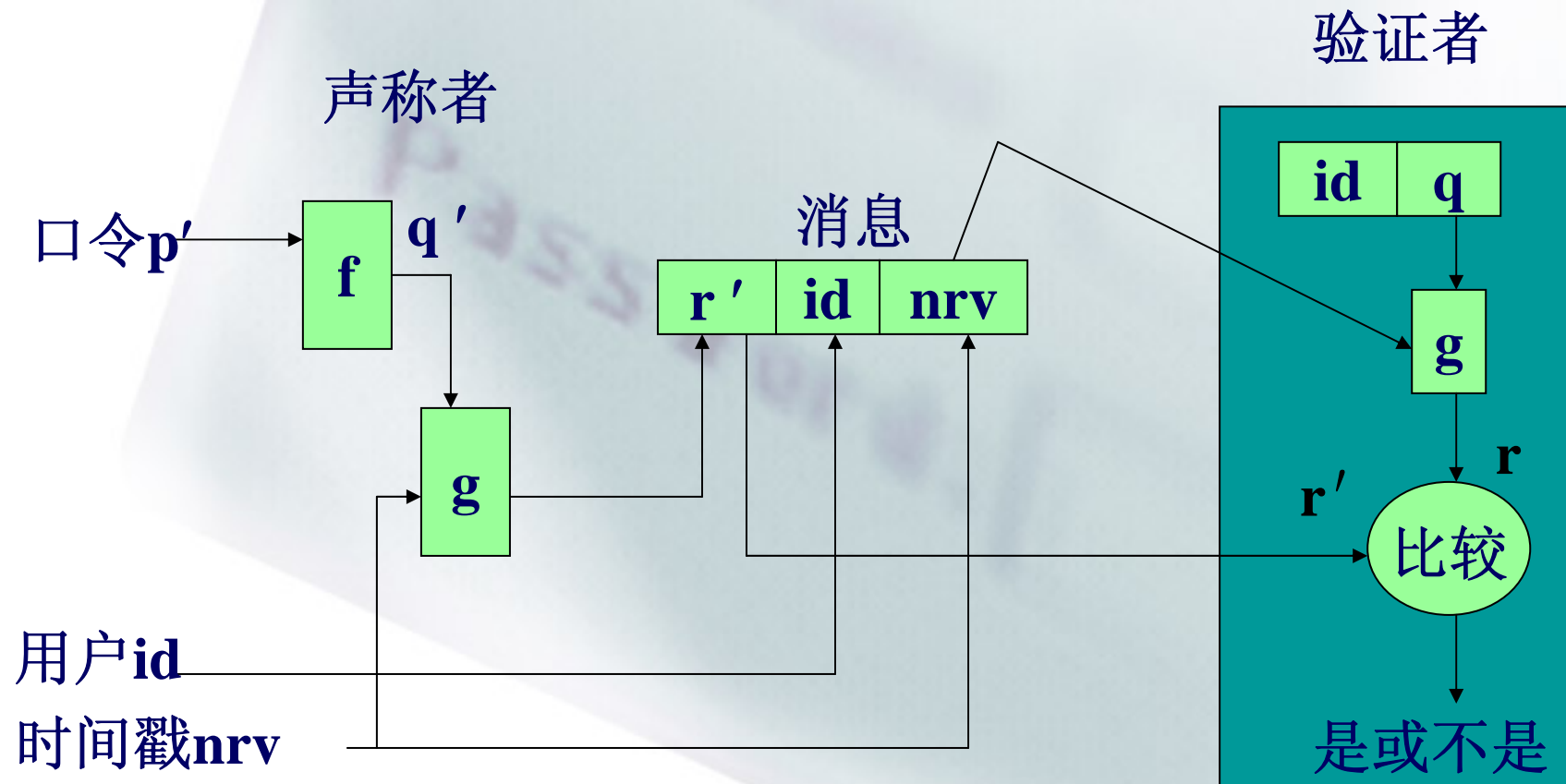
- 在独立的**Windows 2000**计算机上，安全账户管理器负责保存用户账户名和口令的信息。
- 口令通过**散列并被加密**，现有的技术不能将打乱的口令恢复(尽管如此，散列的口令是可以被暴力猜解)。
- **SAM** 组 成 了 注 册 表 的 5 个 配 置 单 元 之 一 ， 它 在 文 件 **%systemroot%\system32\config\sam**中实现。
- 在**Windows 2000**域控制器上，用户账户和散列的数据保存在活动目录中(默认为**%systemroot%\ntds\ntds.dit**)。散列是以相同的格式保存的，但是要访问它们必须通过不同的方法。

保护口令：防止窃听



对付重放攻击的措施

- 抵抗对通信线路的主动攻击——重放攻击。





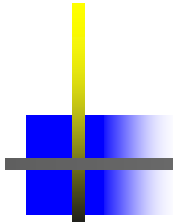
例：窃取口令

- 例如某个UNIX公用系统,普通用户账号“PUBLIC”,口令也是“PUBLIC”。该公用
- 远程登录,所以系统管理员也常使用。“PUBLIC”进行登录后,使用SU命令切换到ROOT。
- 以利用的一个管理漏洞。



具体做法如下:

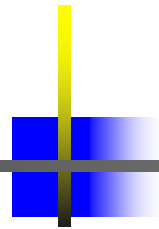
- 1. 改变PUBL IC账号的PATH环境变量值。
- 2. 在/HOME /PUBL IC中放入一个小程序,名字就叫SU。由它模拟真正的SU程序密码,并在此截获超级用户的口令。
- 3. 发送超级用户的密码到某人的个人信箱。
- 4. 显示“SU: INCORRECT PASSWORD”。
- 5. 删除本程序,并恢复PUBL IC账号的PATH环境变量。

- 
- 这个例子提醒我们,系统管理员应当尽量分配给每一个用户以独立的账号,如果一定需要公共账号,系统管理员应当避免使用该公共账号登录,并且在执行命令时也最好加上绝对路径。



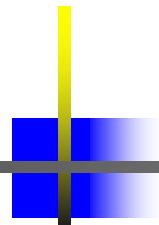
例：绕过口令机制

- 用户名和密码登录的页面中，其源文件可能类似下面的内容：
- `<tr> <td width="63">用户名:</td><td width="231"> <input type="text" name="name"></td></tr>`
- `<tr> <td width="63">密码:</td><td width="231"> <input type="password" name="pass"></td></tr>`



- 可以看出，用户名提交到**name**变量中，密码提交到**pass**变量中。
ASP程序设计代码中，经典的验证代码为：

```
dim rs,sql
set rs=server.createobject("adodb.recordset")
sql="select * from USERS where name = ' "&name&" ' and pass='
    "&pass&" ' "
name=request("name")
pass=request("pass")
rs.open sql,conn,1,3
if rs.eof then
...
```



- 如果用户名是ABC,密码是123, 那么这条SQL语句查找:
name='ABC' and pass='123'
如果把密码输入123' or '1'='1
那么SQL变为查找:
name='ABC' and pass='123'or '1'='1'
- 验证将会被通过, 成功绕过密码检验机制!
- 防范: 密码中不能带'号

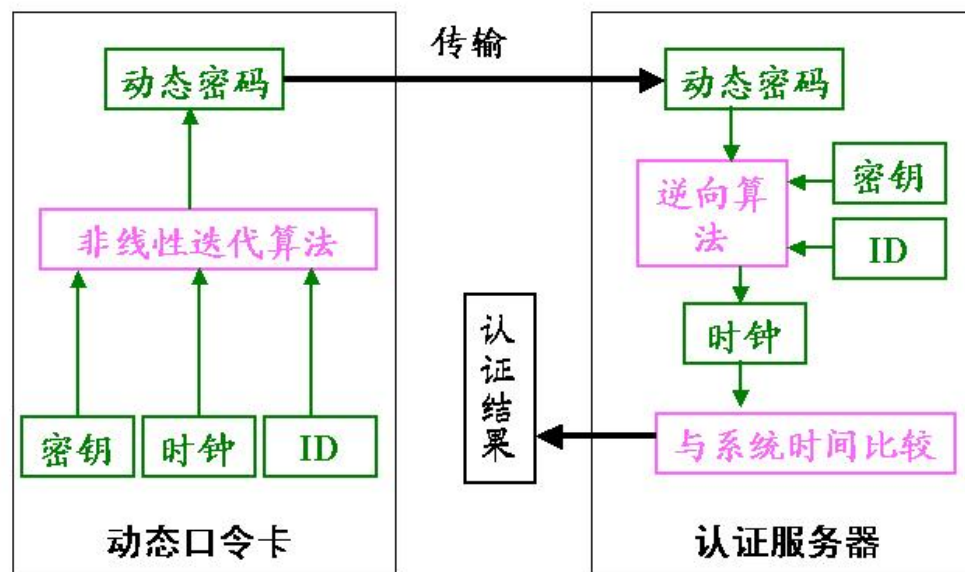


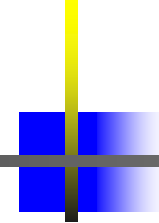
双因素动态口令卡

- 基于密钥/时间双因素的身份鉴别机制;
- 用户登录口令随时间变化, 口令一次性使用, 无法预测, 可以有效抵御密码窃取和重放攻击行为
- **RSA**等多家安全公司

动态口令卡的原理

- 动态口令卡是发给每个用户的动态口令发生器，通过同步信任认证算法，以时间为参数，每隔**16~64**秒钟产生一个一次性使用的“动态口令”





询问—应答机制

- 询问—应答原理可以扩张基于口令的方案，能大大地提高抵抗重放攻击的能力，但通常通信代价很高。
- 前面所示的对付重放攻击的机制存在两个重要的问题。一个是为了两端都知道`nr`值需要维持同步。另一个是验证者要知道`nr`值是否被重复使用过是比较困难的。
- 询问—应答方法克服了这些问题。

安全协议

- 安全协议：建立在密码体制基础上的一种协议，运行在计算机通信网或分布式系统中，为安全需求的各方提供一系列步骤，借助于密码算法来达到密钥分配、身份认证、信息保密以及电子交易等目的。
 1. 密钥交换协议
 2. 认证协议
 3. 电子商务协议



攻击者的能力

- 1) 转发消息
- 2) 延迟消息
- 3) 篡改消息（包括伪造）
- 4) 重放消息
- 5) 中断消息

可看作是一个恶意的网络环境

NSSK协议

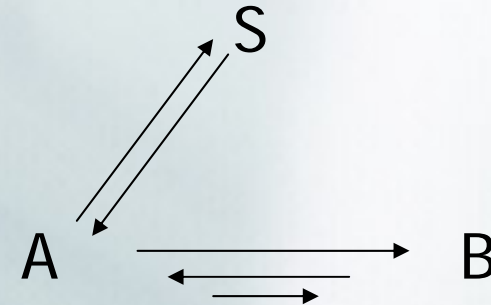
Msg 1 $A \rightarrow S : A, B, Na$

Msg 2 $S \rightarrow A : \{ Na, B, Kab, \{Kab, A\} Kbs \} Kas$

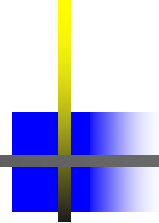
Msg 3 $A \rightarrow B : \{ Kab, A \} Kbs$

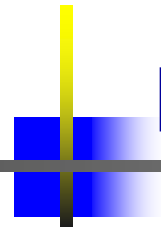
Msg 4 $B \rightarrow A : \{ Nb \} Kab$

Msg 5 $A \rightarrow B : \{ Nb-1 \} Kab$



- ◆ Needham Schroeder Protocol 1978
- ◆ Denning and Sacco 1981 发现一个攻击

- 
- 问题1： 为什么需要随机数**Na**？
 - 问题2： 为什么要把**B**放在**Kas**加密的部分中？
 - 问题3： 为什么要在消息2中出现**A**无法解读的**{Kab,A}Kbs**？
 - 问题4： 为什么不把消息2写成**{Na,B,Kab}Kas, {Kab,A}Kbs**的样子？
 - 问题5： 这两条消息是做什么用的？
 - 问题6： 为什么在消息5中使用**{Nb-1}Kab**而不是**{Nb}Kab**或者**{Nb-2}Kab**？



Needham Schroeder所作的改进 1987

Msg 1 $A \rightarrow B : A$

Msg 2 $B \rightarrow A : \{ A , Nb \} Kbs$

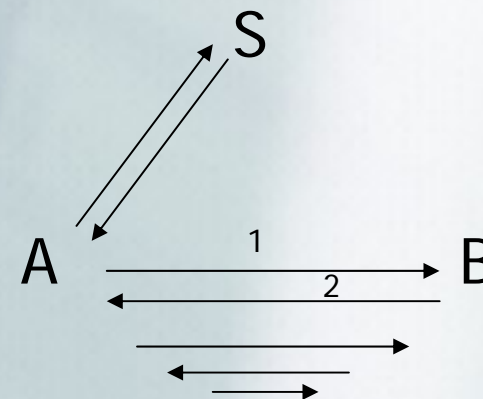
Msg 3 $A \rightarrow S : A , B , Na , \{ A , Nb \} Kbs$

Msg 4 $S \rightarrow A : \{ Na , B , Kab , \{ Kab , Nb , A \} Kbs \} Kas$

Msg 5 $A \rightarrow B : \{ Kab , Nb , A \} Kbs$

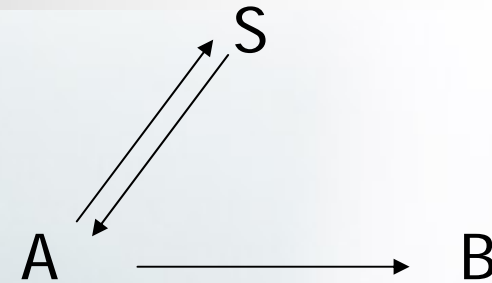
Msg 6 $B \rightarrow A : \{ Nb \} Kab$

Msg 7 $A \rightarrow B : \{ Nb-1 \} Kab$



Denning and Sacco

- *Msg1* $A \rightarrow S: A, B$
- *Msg2* $S \rightarrow A: CA, CB$
- *Msg3* $A \rightarrow B: CA, CB, \{ \{Kab, Ta\}Ka^{-1} \}Kb$



at time T_a , A says that K_{ab} is a good key for session between A and B

◆ *Msg3* $B(A) \rightarrow C: CA, CC, \{ \{Kab, Ta\}Ka^{-1} \}Kc$



基于地址的机制

- 在大多数的数据网络中，呼叫地址的辨别都是可行的。在不能可靠地辨别地址时，可以用一个呼叫—回应设备来获得呼叫的源地址。
- 一个验证者对每一个主体都保持一份合法呼叫地址的文件。
- 这种机制最大的困难是维持一个主机和网络地址的联系。地址的转换频繁、呼叫—转发或重定向引起了一些主要问题。
- 基于地址的机制自身不能被作为鉴别机制，但可作为其它机制的**有用补充**。



基于个人特征的机制

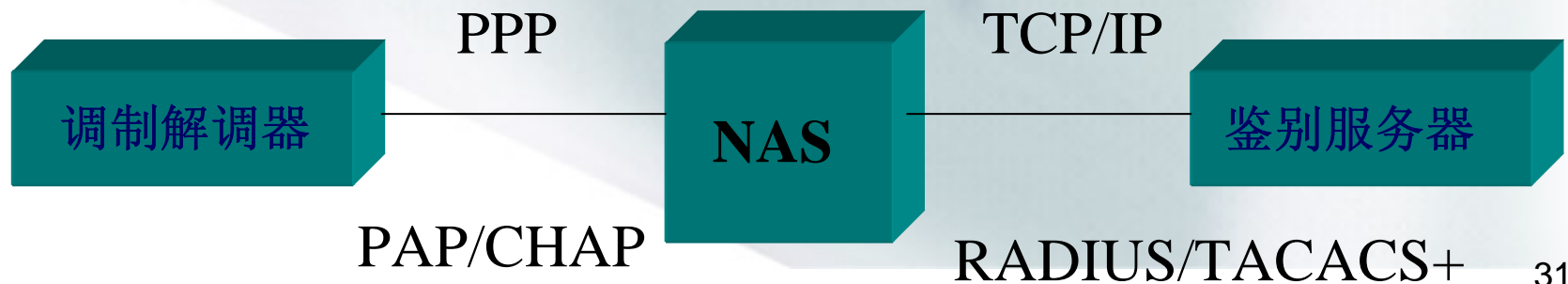
- 生物特征识别技术主要有：

- 1) 指纹识别；
- 2) 声音识别；
- 3) 手迹识别；
- 4) 视网膜扫描；
- 5) 手形。

拨号鉴别协议

拨号鉴别协议：在拨号环境中，要进行两部分的鉴别：

- ① 首先是用户的调制解调器和网络访问服务器（**NAS**）之间使用**PPP**鉴别协议鉴别，这类鉴别协议包括**PAP**、**CHAP**等；
- ② 然后**NAS**和鉴别服务器（**AS**）进行鉴别，这类协议包括**RADIUS**、**TACACS+**等。



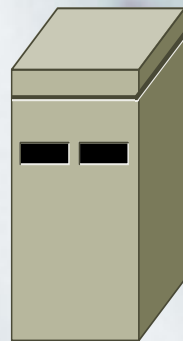


CHAP

- 拨号用户鉴别协议 (**challenge-Handshake Authentication Protocol**), 是一种通过**PPP**链接传送数据, 完成对**PPP**链接的身份鉴别协议。
- **CHAP**主要适用于**NAS(network access Server)**对来自于**PSTN**或**ISDN**的电路交换连接、拨入连接或专有连接的身份鉴别
- **RFC1994**
 - 采用提问/应答方式进行鉴别, 通过在**PPP**链接的双方进行一次三次握手,完成对对方身份的鉴别

CHAP 示意

Server



Client



Challenge

Respond

Succeed/failed

Challenge = SessionID, Challenge String

Respond = MD5 Hash(SessionID, Challenge String, User Password), User Name



CHAP协议的优点与缺点

- 优点

- ☐ 协议简单、易于实现
- ☐ 虽为单向鉴别协议，但可以通过在另一方的配置请求，实现对通信实体的双向鉴别
- ☐ 通过不断地改变鉴别标识符和提问消息的值来防止重放(**playback**)攻击。
- ☐ 利用周期性的提问防止通信双方在长期会话过程中被攻击。

- 缺点

- ☐ **CHAP**鉴别基于共享密钥，给密钥管理带来巨大不便，不适合于大规模用户鉴别
- ☐ **CHAP**要求提供明文的密钥形式。



Kerberos历史

- 80年代中期
- 是MIT的Athena工程的产物
- 版本
 - 前三个版本仅用于内部
 - 第四版得到了广泛的应用
 - 第五版于1989年开始设计
 - RFC 1510, 1993年确定
 - 标准Kerberos



Kerberos要解决的问题

- 防止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鉴别协议，**Kerberos** 提供一个中心鉴别服务器，提供用户到服务器和服务器到用户的鉴别服务。
- **Kerberos**采用传统加密算法。
- **Kerberos Version4和Version5 (RFC1510)**。

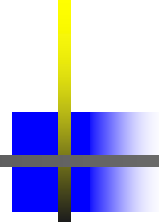


基本概念

- **Kerberos**的实现由客户(**C**)、服务器(**V**)、认证服务器(**AS**)以及票据提供服务器 (**TGS**) 组成。
- 客户最终从服务器获得所需的服务，**AS**和**TGS**是**Kerberos**服务器的主要内容，**AS**提供对客户的认证，**TGS**颁发客户所需服务的一个服务许可证，也即票据，名为服务票据，记作**TicketV**。
- 值得注意的是，**AS**也颁发票据，名为**TGS**票据，记作**Tickettgs**，客户可以拿着这个**Tickettgs**向**TGS**索取**TicketV**。

过程

- 客户**C**登录工作站所位于的域（**realm**，每个组织或机构所拥有的网络范围）后，首先向**AS**认证自己，**AS**会返回一个**Tickettgs**给客户，票据被**AS**和**TGS**共享的密钥加密，这个认证过程在整个登录期间（从登入到登出）只发生一次。
- 然后，客户需要服务时，向**TGS**发送这个**TGS**票据，以从**TGS**获得一个服务票据**TicketV**，这个过程在登录期间，根据所需服务种类的需要，每种服务会发生一次，如客户需要邮件服务，会向**TGS**要一张邮件服务票据，需要**FTP**服务时，会向**TGS**要一张**FTP**服务票据。

- 
- 最后，客户向服务器V发送票据**TicketV**，服务器查看票据后，确认无误后，向客户提供服务，这个过程在登录期间，会发生多次，每次服务提供前都要发生一次，如客户一天中，多次检查收取邮件时，都会提供一次票据**TicketV**。整个过程中，除了登录时需要用户输入口令，剩下的认证过程都不需要用户了解和参与。

认证符

- 上述过程中省略了一个重要的内容，即认证符。它的作用是在一个事务中验证用户的身份，客户不仅要传输票据，还需要发送额外的信息来证明自己确实是票据的合法拥有者。这个信息就是认证符(**authenticator**)，它使用会话密钥加密，并包含了用户名和时间戳。
- **AS**发给**C**票据的同时，还发送客户**C**和**TGS**之间的会话密钥 $K_{c,tgs}$ ，这个会话密钥本身由用户口令衍生而得的密钥加密；**C**发给**TGS**的认证符由 $K_{c,tgs}$ 加密的，**TGS**发给**C**票据的同时，还发送**C**和**V**之间的会话密钥 K_{cv} ，这个会话密钥也由 $K_{c,tgs}$ 加密，**C**发给**V**的认证符由 K_{cv} 加密。

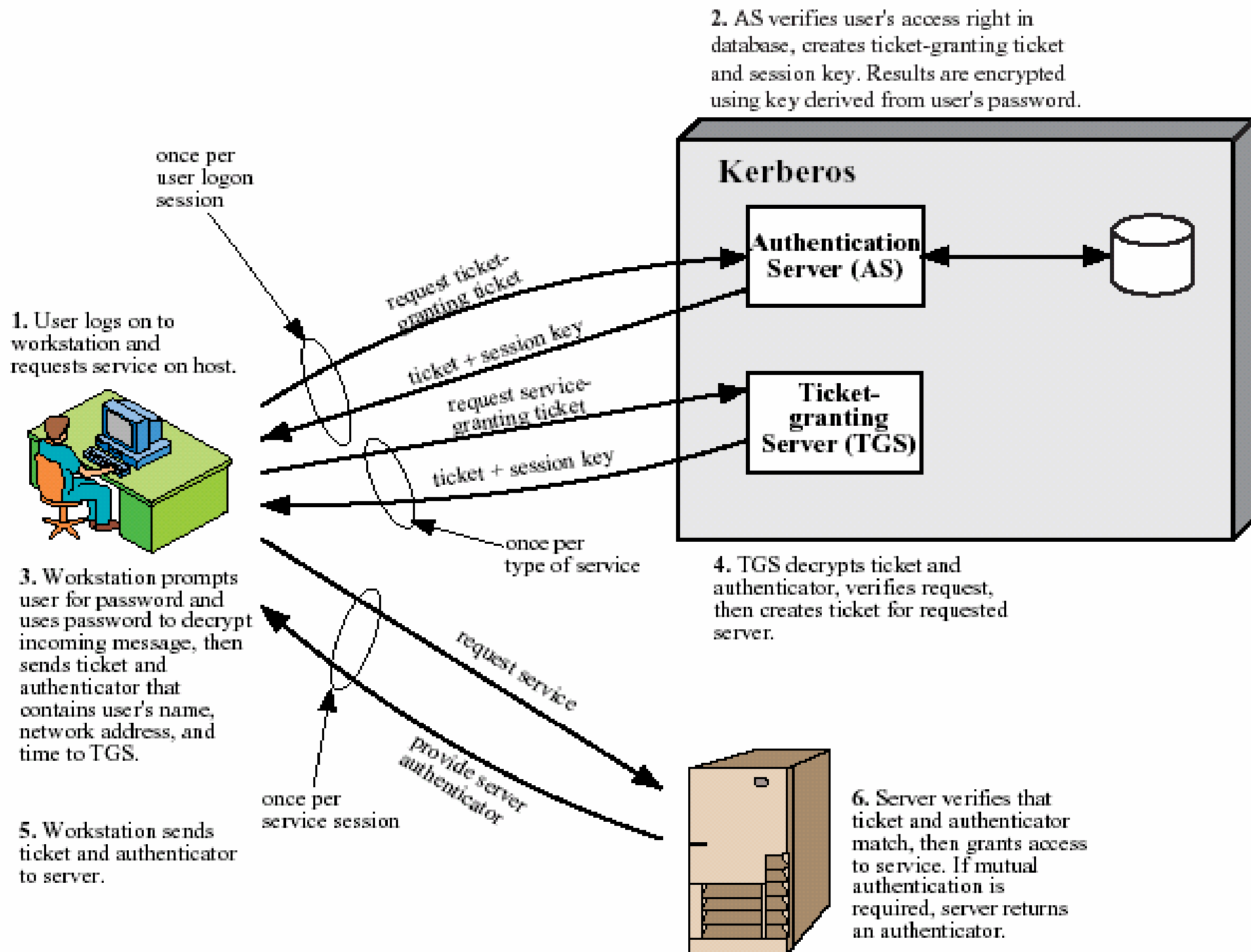
跨域操作

- **Kerberos**协议被设计为可以用来跨组织边界进行操作。一个组织中（如一个部门或一个科室）的客户可以被授权获得另一个组织中服务器的服务。从网络角度讲，每一个运行**Kerberos**服务器的子网都是一个域。
- 通过建立“域间密钥”，一个客户不仅可以从他本地域的**TGS**获得**Ticket_V**，还可获得一个远程域的**Ticket_{tgssrem}**。然后客户在需要使用远程域的服务时，将**Ticket_{tgssrem}**发送给远程域的**TGS**，远程的**TGS**使用域间密钥来解密**Ticket_{tgssrem}**，验证它是否是由客户所在域的**TGS**颁发的。如果验证通过，远程**TGS**颁发远程服务票据**Ticket_{Vrem}**。

kerberos报文

Msg1 $C \rightarrow AS$: *Options*, C , $Realm_C$, TGS , $Times$, $Nonce_1$
Msg2 $AS \rightarrow C$: $Realm_C$, C , $Ticket_{TGS}$, $\{K_{C,TGS}, Times, Nonce_1, Realm_{TGS}, TGS\}K_C$
Msg3 $C \rightarrow TGS$: *Options*, V , $Times$, $Nonce_2$, $Ticket_{TGS}$, $Authenticator_{C,TGS}$
Msg4 $TGS \rightarrow C$: $Realm_C$, C , $Ticket_V$, $\{K_{C,V}, Times, Nonce_2, Realm_V, V\}K_{C,TGS}$
Msg5 $C \rightarrow V$: *Options*, $Ticket_V$, $Authenticator_C$
Msg6 $V \rightarrow C$: $\{TS_2, Subkey, Seq\# \}K_{C,V}$

$Ticket_{TGS} = \{Flags, K_{C,TGS}, Realm_C, C, AD_C, Times\}K_{AS,TGS}$
 $Ticket_{TGS} = \{Flags, K_{C,V}, Realm_C, C, AD_C, Times\}K_{TGS,V}$
 $Authenticator_{C,TGS} = \{C, Realm_C, TS_1\}K_{C,TGS}$
 $Authenticator_{C,V} = \{C, Realm_C, TS_2, Subkey, Seq\#\}K_{C,V}$





Kerberos鉴别协议小结

- 特点
 - 基于口令的鉴别协议
 - 利用对称密码技术建立起来的鉴别协议
 - 可伸缩性——可适用于分布式网络环境
 - 环境特点: **User-to-server authentication**

NSPK协议

Needham-Schroeder Public Key Protocol (1978)

Msg 1 $A \rightarrow S : A, B$

Msg 2 $S \rightarrow A : \{ Kb, B \}_{Ks^{-1}}$

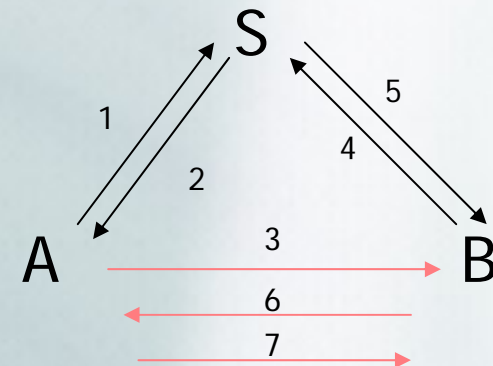
Msg 3 $A \rightarrow B : \{ Na, A \}_{Kb}$

Msg 4 $B \rightarrow S : B, A$

Msg 5 $S \rightarrow B : \{ Ka, A \}_{Ks^{-1}}$

Msg 6 $B \rightarrow A : \{ Na, Nb \}_{Ka}$

Msg 7 $A \rightarrow B : \{ Nb \}_{Kb}$



攻击

Lowe在**95**年使用**CSP**模型和**FDR**工具发现一个攻击

Msg 3 $A \rightarrow Z : \{ Na, A \} Kz$

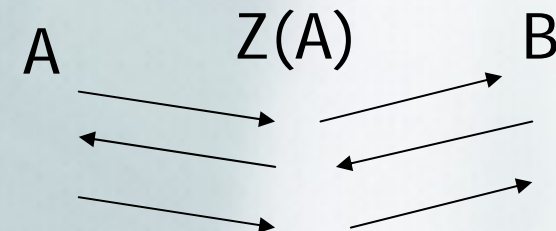
Msg 3' $Z(A) \rightarrow B : \{ Na, A \} Kb$

Msg 6' $B \rightarrow Z(A) : \{ Na, Nb \} Ka$

Msg 6 $Z \rightarrow A : \{ Na, Nb \} Ka$

Msg 7 $A \rightarrow Z : \{ Nb \} Kz$

Msg 7' $Z(A) \rightarrow B : \{ Nb \} Kb$





Lowe对NS公钥协议的改进

Msg 1 $A \rightarrow S : A, B$

Msg 2 $S \rightarrow A : \{Kb, B\} Ks^{-1}$

Msg 3 $A \rightarrow B : \{Na, A\} Kb$

Msg 4 $B \rightarrow S : B, A$

Msg 5 $S \rightarrow B : \{Ka, A\} Ks^{-1}$

Msg 6 $B \rightarrow A : \{Na, Nb, B\} Ka$

Msg 7 $A \rightarrow B : \{Nb\} Kb$

仍然有缺陷 --type flaw攻击

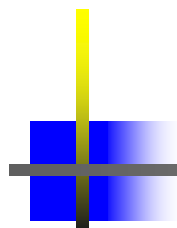
Msg 3 $Z(A) \rightarrow B : \{ Na, A \} Kb$

Msg 6 $B \rightarrow Z(A) : \{ Na, Nb, B \} Ka$

Msg 3' $Z \rightarrow A : \{ Na, \underline{Nb}, \underline{B} \} Ka$

Msg 4' $A \rightarrow S : A, \underline{Nb}, \underline{B}$

Msg 7 $Z(A) \rightarrow B : \{ Nb \} Kb$



Q&A

谢谢！