东南大学网络空间安全学院
密码学与安全协议

第八讲 数字签名和认证协议

黄　杰
信息安全研究中心

本讲内容

- 数字签名简介
- 认证协议**N-S Protocol**
- 数字签名标准**DSS**
- 其他签名方式
- **Kerberos**协议

知识点：

1、数字签名的内涵？

2、直接数字签名和仲裁数字签名？

3、认证协议

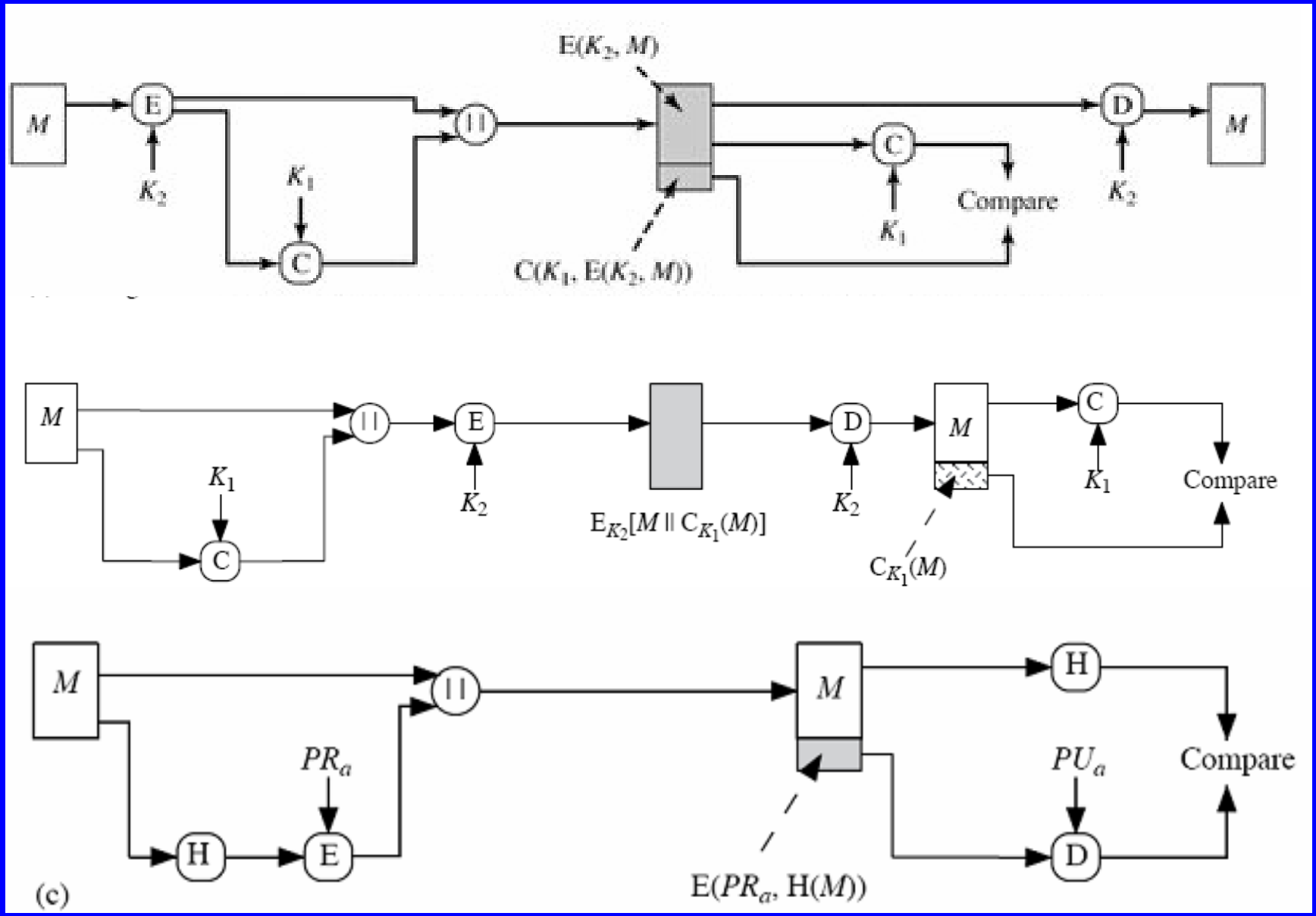
4、DSA算法的原理

5、Kerberos协议原理

数字签名的定义

- 数字签名（*Digital Signature*）是公钥密码体系加密技术发展的一个重要的成果。
- 数字签名就是附加在数据单元上的一些数据，或是对数据单元所作的密码变换。这种数据或变换允许数据单元的接收者用以确认数据单元的来源和数据单元的完整性并保护数据，防止被人(例如接收者)进行伪造。
- 数字签名是对现实生活中笔迹签名的模拟。

消息认证码和数字签名的区别



数字签名的要求

- 传统签名的基本特点:
  - 必须能验证签名者、签名时间
  - 必须能够认证被签名的消息内容
  - 签名能够由第三方仲裁，以解决争执

- 数字签名是传统签名的数字化，基本要求:
  - 签名必须是与消息相关的二进制位串
  - 签名必须使用发送方特有的信息，防伪造或否认
  - 签名的产生和识别比较容易
  - 伪造数字签名在计算上是不可行的
  - 保存数字签名的拷贝是可行的

数字签名体制

- 签名算法(Signature Algorithm)
  - Sig(K，M)=S
  - 签名密钥K是秘密的，只有发方掌握
- 验证算法(Verification Algorithm)
  - Ver(K，S)={1，0}={真，伪}
  - 验证算法公开，便于他人进行验证
- 签名体制的安全性在于，从M和其签名S难以推出签名密钥K或伪造一个M’使M’和S可被证实为真。

数字签名的分类

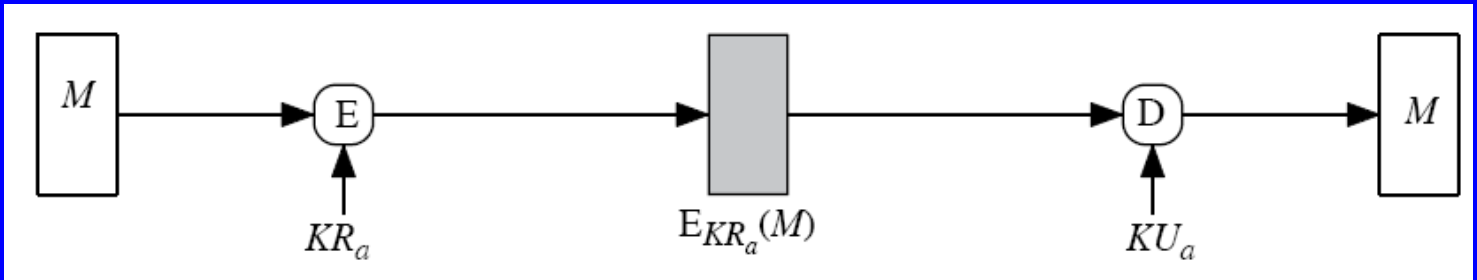
- 直接数字签名(direct digital signature)
  - 只涉及通信双方
- 仲裁数字签名(arbitrated digital signature)
  - 可信第三方介入

直接数字签名方法1

- 用发送方的私钥对整条消息进行签名.
- (1) A→B: $E_{KRa}[M]$

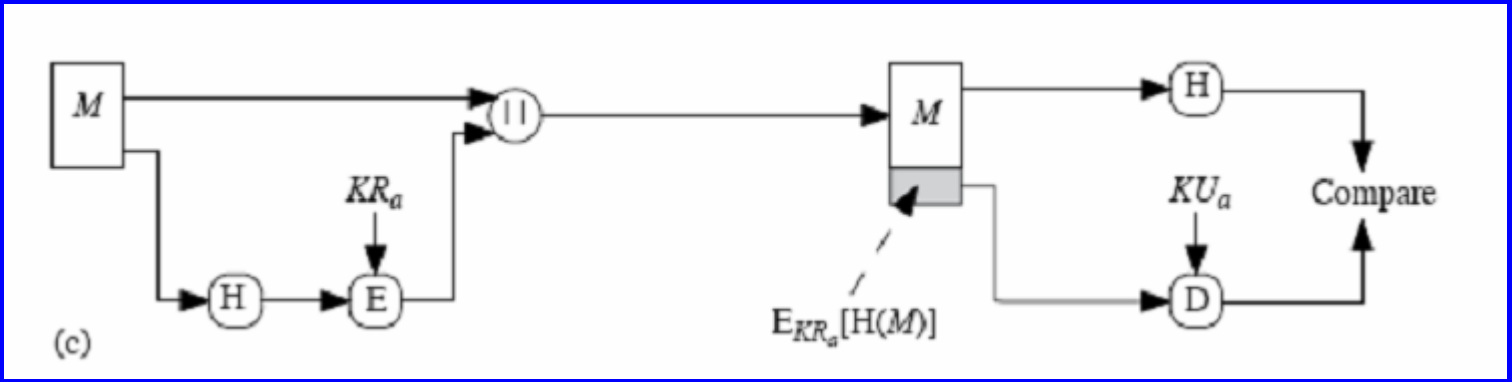
  提供了鉴别与签名
  - 只有A具有K_{Ra}进行加密;
  - 传输中没有被篡改;
  - 需要某些格式信息/冗余度;
  - 任何第三方可以用KU_a 验证签名
- (1') A→B: E_{KUb} [E_{KRa}(M)]

  提供了保密(K_{Ub})、鉴别与签名(K_{Ra})



直接数字签名方法2

- 用发送方的私钥对消息的hash码进行签名
- (2) A→B: $M||E_{KRa}[H(M)]$
  提供数字签名
  - H(M) 受到密码算法的保护,例如MD5或SHA-1;
  - 只有A 能够生成$E_{KRa}[H(M)]$
- (2') A→B: $E_K[M||E_{KRa}[H(M)]]$
  提供保密性、数字签名。
- 加密和签名的先后顺序?



直接数字签名的缺点

- 依赖于发送方的保密密钥
  - 发送方要抵赖发送某一消息时，可能会声称其私有密钥丢失或被窃，从而他人伪造了他的签名。
  - 通常需要采用与私有密钥安全性相关的行政管理控制手段来制止或至少是削弱这种情况，但威胁在某种程度上依然存在。
  - 改进的方式例如可以要求被签名的信息包含一个时间戳（日期与时间），并要求将已暴露的密钥报告给一个授权中心。
- X的私钥确实在时间T被窃取，敌方可以伪造X的签名及早于或等于时间T的时间戳。

仲裁数字签名

- 工作原理：
  - 通常的做法是所有从发送方X到接收方Y的签名消息首先送到仲裁者A，A将消息及其签名进行一系列测试，以检查其来源和内容，然后将消息加上日期并与已被仲裁者验证通过的标识一起发给Y。
- 仲裁者在这一类签名模式中扮演敏感和关键的角色。
  - 所有的参与者必须极大地相信这一仲裁机制工作正常。（trusted system）

仲裁数字签名模式1

- X与A之间共享密钥Kxa，Y与A之间共享密钥Kay；
  （1）X→A：$M \| E_{Kxa}[IDx \| H(M)]$
  （2）A→Y：$E_{Kay}[IDx \| M \| E_{Kxa}[IDx \| H(M)] \| T]$
- X：准备消息M，计算其散列码H(M)，用密钥Kxa加密X的标识符IDx 和散列值构成签名，并将签名之后的消息与明文组合后发送给A；
- A：解密签名，用H(M)验证消息M，然后将ID_x，M，签名和时间戳一起经Kay加密后发送给Y；
- Y：解密A发来的信息，并可将M和签名保存起来。Y不能直接读取签名，但可以将消息和签名发给A进行验证。
- 特点：传统加密方式，仲裁者可以看见消息明文。

仲裁数字签名模式2

- 在这种情况下，X与Y之间共享密钥Kxy
  （1）X→A： IDx || $E_{Kxy}[M]$||$E_{Kxa}[IDx|| H(E_{Kxy}[M])]$
  （2）A→Y：$E_{Kay}[IDx||E_{Kxy}[M] || E_{Kxa}[IDx|| H(E_{Kxy}[M])] || T]$
- X：将标识符IDx ,密文$E_{Kxy}[M]$，以及对IDx和密文消息的散列码用Kxa加密后形成签名发送给A。
- A：解密签名，用散列码验证消息，这时A只能验证消息的密文而不能读取其内容。然后A将来自X的所有信息加上时间戳并用Kay加密后发送给Y。
- 特点：传统加密方式，仲裁者不可以看见消息明文。
- （1）和（2）共同存在一个共性问题：
  - 发送方可以否认签名的信息；
  - A和接收方联手可以伪造发送方的签名

仲裁数字签名模式3

- 特点：公钥加密方式，仲裁者不可以看见消息

  （1）X→A： $ID_x \| E_{KR_x}[ID_x \| E_{KU_y}(E_{KR_x}[M])]$

  （2）A→Y： $E_{KR_a}[ID_x \| E_{KU_v}[E_{KR_x}[M]] \| T]$

- X：对消息M双重加密：首先用X的私有密钥KRx，然后用Y的公开密钥KUy。形成一个签名的、保密的消息。然后将该信息以及X的标识符一起用KRx签名后与IDx 一起发送给A。这种内部、双重加密的消息对A以及对除Y以外的其它人都是安全的。
- A：检查X和Y的公开/私有密钥对是否仍然有效，若是则确认消息。并将包含IDx、双重加密的消息和时间戳构成的消息用KRa签名后发送给Y。

模式3的优点

- 1、在通信之前各方之间无须共享任何信息，从而避免了联手作弊；
- 2、即使KRx 暴露，只要KRa 未暴露，不会有错误标定日期的消息被发送；
- 3、X发送给Y的消息的内容对A和任何其他人是保密的。

身份认证技术及协议

- **身份认证：**对用户身份的证实，用以识别合法或非法的用户，阻止非授权用户访问资源。

  数字签名和鉴别技术的一个最主要的应用领域就是身份认证。

  在网络应用环境中，网络资源的安全性保障通常采用基于用户身份的资源访问控制策略

- **身份认证的作用**是对用户身份进行鉴别，能够保护网络中的数据和服务不被未授权的用户所访问。

认证协议

- 双方认证**(mutual authentication)**

  最常用的协议，该协议使得通信各方互相认证和鉴别各自的身份，然后交换会话密钥。
- 单向认证**(one-way authentication)**

  收发双方不需要同时在线联系，例如电子邮件。

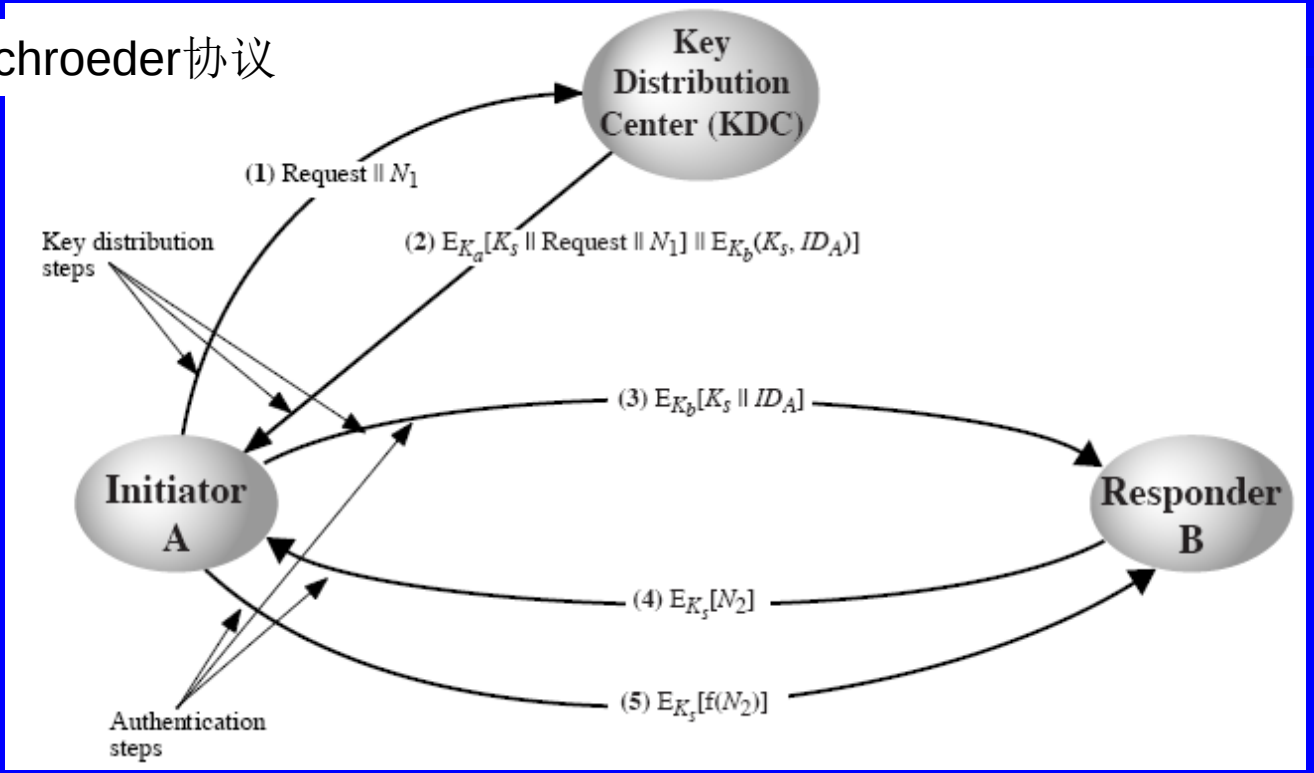
双方认证

- 解决重放攻击的方法？
  - 序列号
  - 时间戳
  - 挑战/应答

- 对称加密方法
- 公钥加密方法

对称加密方法

Needham-Schroeder协议



1、A->KDC： $ID_A||ID_B||N_1$

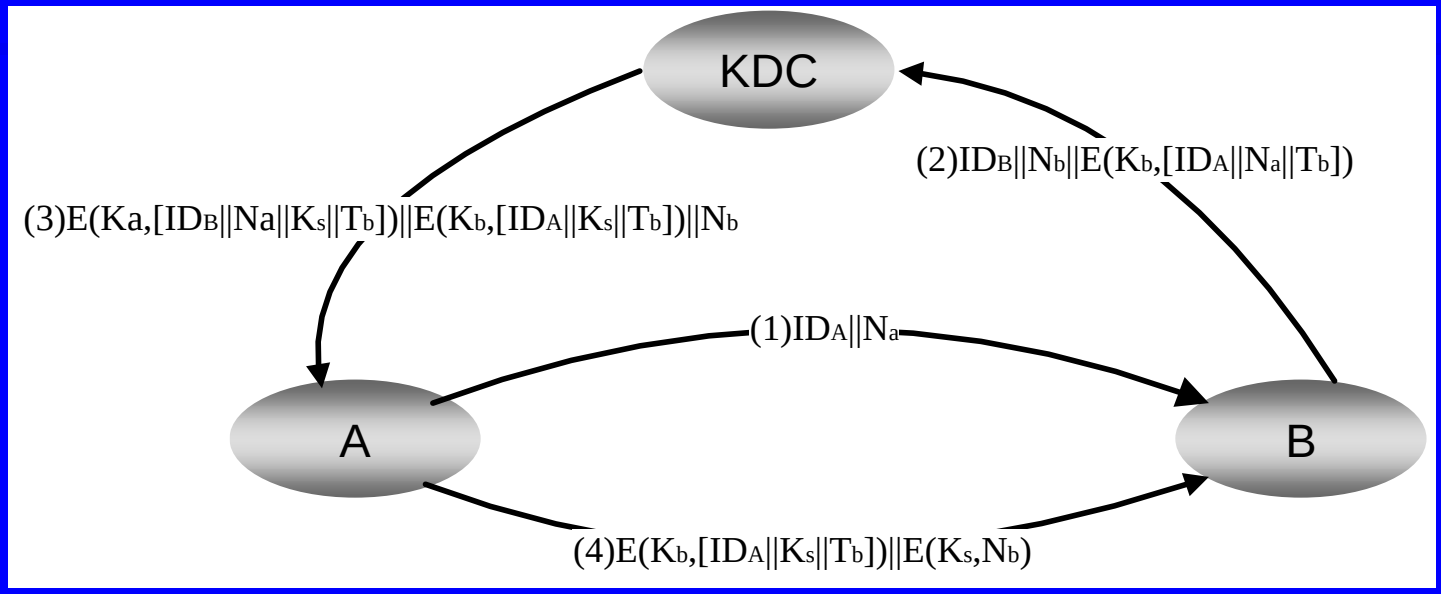
2、KDC->A： $E(K_a, [K_s||ID_B||N_1])||E(K_b, [K_s||ID_A]|T])$

3、A->B　： $E(K_b, [K_s||ID_A]|T])$

4、B->A　： $E(K_s, N_2)$

5、A->B　： $E(K_s, f(N_2))$

改进的方法



1、A->B: $ID_A||N_a$

2、B->KDC: $ID_B||N_b||E(K_b, [ID_A||N_a||T_b])])$

3、KDC->A: $E(K_a, [ID_B||N_a||K_s||T_b])||E(K_b, [ID_A||K_s||T_b])||N_b$

4、A->B: $E(K_b, [ID_A||K_s||T_b])||E(K_s, N_b)$

5、A->B: $E(K_b, [ID_A||K_s||T_b])$, N_a'

6、B->A: N_b', $E(K_s, N_a')$

7、A->B: $E(K_s, N_b')$

公钥加密方法

- 1、A → AS：$ID_A||ID_B$
- 2、AS → A：$E(PR_{as},[ID_B||PU_b||T])||E(PR_{as},[ID_A||PU_a||T])$
- 3、A → B：$E(PU_b,[ID_A||N_A])$
- 4、B → AS：$E(PR_{as},[ID_A||ID_B||E(PU_{as},N_A)])||E(PR_{as},[ID_B||PU_a||T])$
- 5、AS → B：$E(PR_{as},[ID_A||PU_a||T])||E(PR_{as},[N_A||Ks||ID_B])$
- 6、B →A：$E(PU_a,E(PR_{as},[N_A||Ks||ID_B||N_B]))$
- 7、A →B：E（Ks,N_B）

单 向 认 证

单向认证

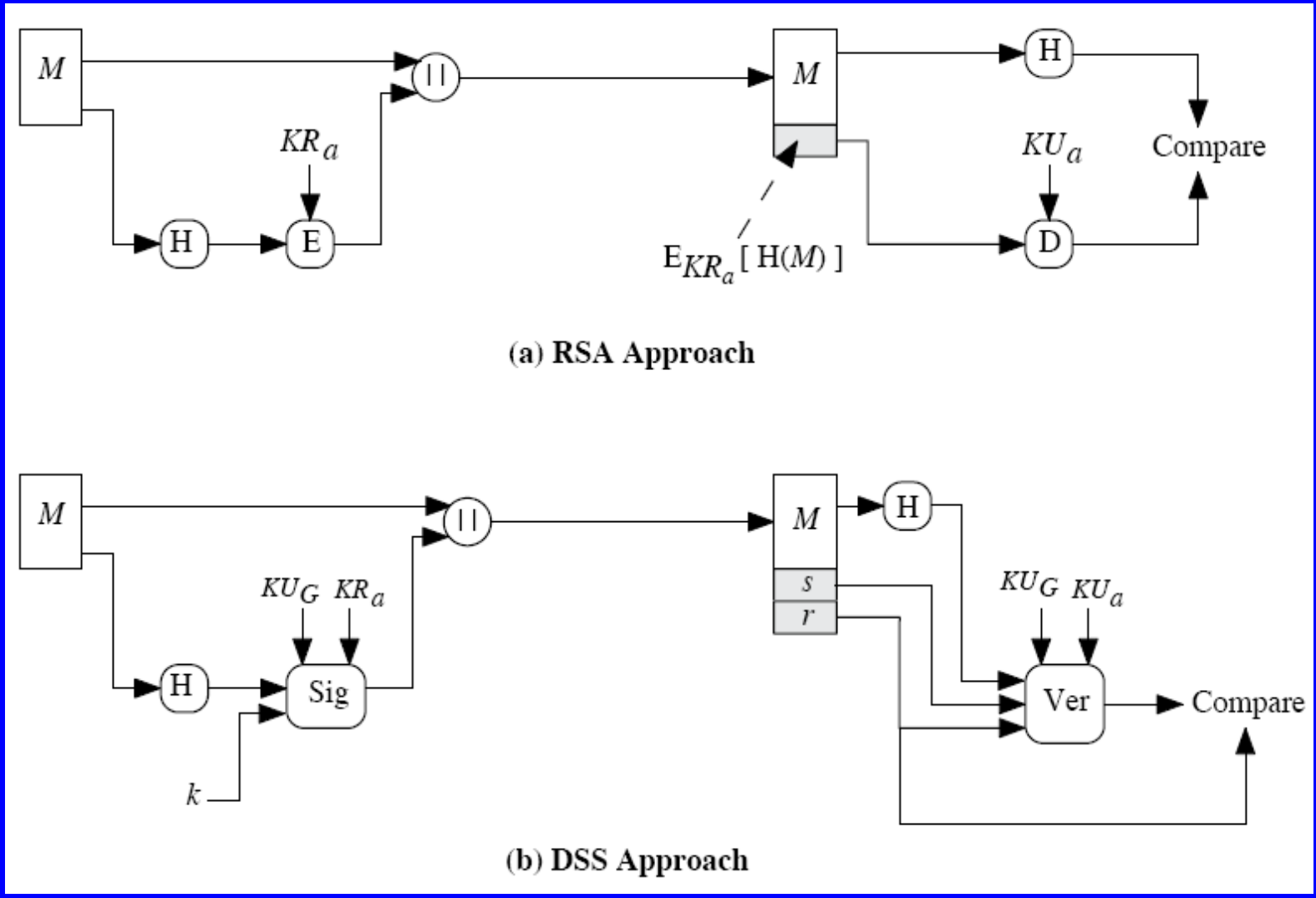
- 对称加密方法
  1、A->KDC： $ID_A||ID_B||N_1$
  2、KDC->A：$E(K_a,[K_s||ID_B||N_1])||E(K_b,[K_s||ID_A])$
  3、A->B： $E(K_b,[K_s||ID_A])||E(K_s,M)$

- 公钥加密方法
  - A →B： $E(PU_b,K_s)||E(K_s,M)$ （保密）
  - A →B： $M||E(PR_a,H(M))$ （真实性）
  - A →B： $E(PU_b,[M||E(PR_a,H(M))])$ 保密和真实

数字签名标准DSS

- 1991年，美国国家技术与标准研究所发布的联邦信息处理标准FIPS 186，称为DSS。使用新的数字签名算法，即DSA(数字签名算法)。
- 2000年发布了该标准的扩充版，即FIPS 186-2。该最新版还包括基于RSA和椭圆曲线密码的数字签名。
- DSS仅提供数字签名功能，不能用于加密或密钥分配。

RSA 与DSS的比较


M
||
H
E
KR_a
M
H
D
KU_a
Compare
E_{KR_a}[H(M)]
(a) RSA Approach
M
||
H
Sig
KU_G KR_a
k
M
s
r
H
Ver
KU_G KU_a
Compare
(b) DSS Approach

DSS签名函数

- 输入：
  - 明文的hash函数
  - 为签名而产生的随机数k
- 控制参数：
  - 发送方的私钥KR_a
  - 一组公共参数KU_G，称为全局公钥
- 输出：
  - 包含两部分s和r

DSA的全局公钥

Global Public Key Components

p prime number where $2^{L-1} < p < 2^L$
for $512 \leq L \leq 1024$ and L a multiple of 64
i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p-1)$, where $2^{159} < q < 2^{160}$
i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$
where h is any integer with $1 < h < (p-1)$
such that $h^{(p-1)/q} \bmod p > 1$

发送方的密钥生成

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

y $= g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

签名、验证算法

Signing

$$r = (g^k \bmod p) \bmod q$$

$$s = \left[k^{-1}(\mathrm{H}(M) + xr)\right] \bmod q$$

$$\text{Signature} = (r, s)$$

Verifying

$$w = (s')^{-1} \bmod q$$

$$u_1 = \left[\mathrm{H}(M')w\right] \bmod q$$

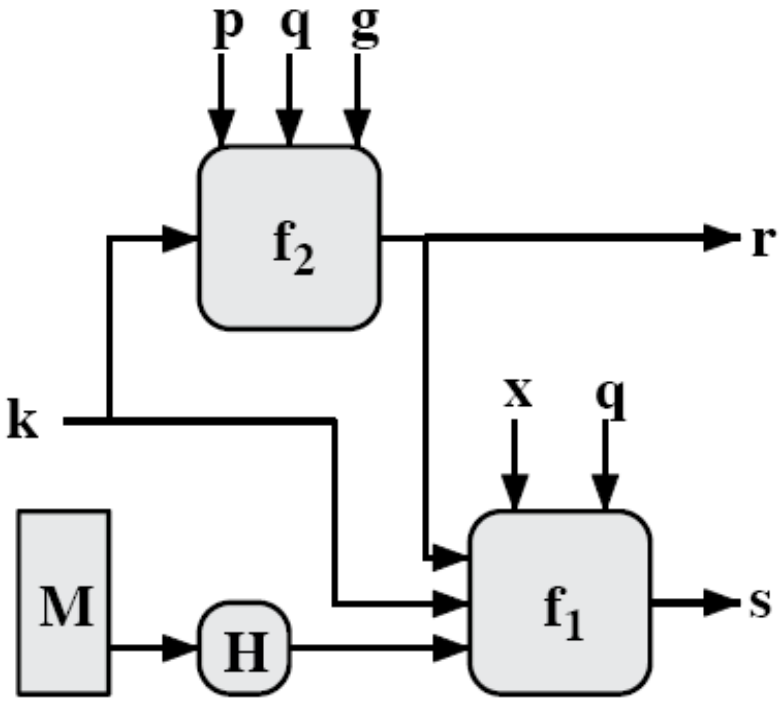
$$u_2 = (r')w \bmod q$$

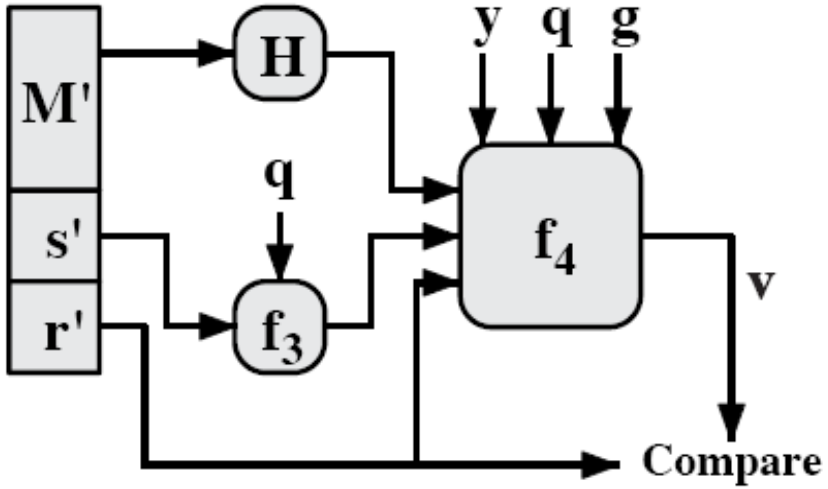
$$v = \left[\left(g^{u1} y^{u2}\right) \bmod p\right] \bmod q$$

$$\text{TEST: } v = r'$$

M：要签名的消息
H(M)：使用SHA-1求得的M的hash码
M’,r’,s’：接收的M,r,s

DSS签名和验证





$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

DSS算法的讨论

- 算法的安全性依赖于离散对数求解的困难性
- 算法正确性的证明

其他数字签名方式

- 群签名:一个群体中的任意一个成员可以以匿名的方式代表整个群体对消息进行签名。

例如：在一些情况下，一些重要文件需要多个人的签名，例如一份项目合同，可能需要参与的多方来共同签署，于是就要采用群体数字签名。

- 限制签名次数的签名方案:在很多实际应用中，需要控制签名的随意性，可在RSA基础上，设计控制签名次数的方案，其缺点是要使用“可信中心”和“信用中心”。

其他数字签名方式

- 授权签名或称代理签名:是一种特殊形式的签名，它在现代企业活动中具有重要意义。

设A为某公司经理，他希望其下属B作为全权代表与C进行谈判，并在合同上签名。一方面，C要求B的签名具有与A同等的法律效果；另一方面，A又不能将自己的密钥直接交给B。这时，A可以事先约定签名的时间，并在系统内公布该时间，然后利用密钥转换函数，将自己的密钥转换为带有时间标志的授权密钥，B利用此授权密钥进行签名。而C可以利用A原来的公钥和时间参数验证授权签名的有效性。

其他数字签名方式

- 盲签名:与其他签名的不同处在于，签名者并不知道（或不需要知道）所签发消息的具体内容。

例如：在电子投票系统中，投票人的意愿是消息M，他把M转换成盲消息M’（这一过程称为盲化），并把M’发给管理中心签名；如果是合法投票人的合法选票，管理中心对M’签名为sig(M’)；投票人将sig(M’)送计票中心；计票中心验证选票的签名，处理选票人的意愿M。在整个过程中，管理中心不知道M是什么，其签名称为盲签名。根据投票人的匿名性，盲签名又分为强盲签名（不能恢复出投票人）和弱盲签名（可以恢复出投票人）。在电子货币中，盲签名也有实际用途。

身份认证协议 —— Kerberos

- 是美国麻省理工学院（MIT）开发的一种身份鉴别服务。
- 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

Kerberos 的设计目标

- 安全性:能够有效防止攻击者假扮成另一个合法的授权用户。
- 可靠性:分布式服务器体系结构，提供相互备份。
- 对用户透明性
- 可伸缩: 能够支持大数量的客户和服务器。

总的来说，Kerberos协议设计目标是提供一种可信的第三方的身份认证服务。

Kerberos设计的基本思路

- 使用一个(或一组)独立的认证服务器(AS-Authentication Server)为网络中的客户提供身份认证服务；
- 认证服务器(AS),用户口令由AS保存在数据库中；
- AS与每个服务器共享一个唯一保密密钥（已被安全分发）。

存在的主要问题

(1)希望用户输入口令次数最少,尽量减少用户不便。若允许票据的多次使用将导致安全性下降;用户访问不同服务器时仍需要多次申请票据的过程。

(2)口令P_C以明文传送会被窃听。

(1)C→AS:$ID_C||P_C||ID_V$

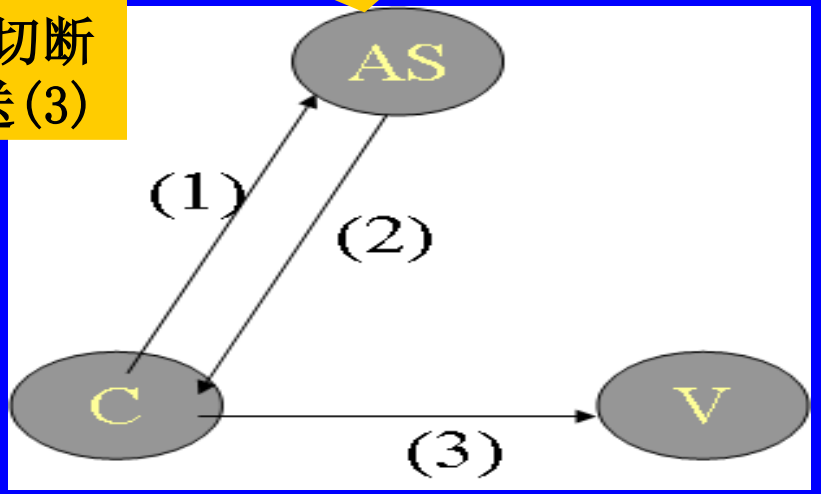
(2)AS→C:Ticket

(3)C→V:$ID_C||Ticket$

$Ticket=E_{K_V}[ID_C||AD_C||ID_V]$

保证对手无法切断(2),冒充C发送(3)

服务器能证实票据解密的正确性



C:客户

AS:认证服务器

V:应用服务器

ID:用户标识符

解决办法:(1)票据重用

(2)引入票据许可服务器(TGS)

a、用于向用户分发服务器的访问票据;

b、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中的票据

- 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$Ticket_{tgs}$表示访问TGS服务器的票据；
  - $Ticket_{tgs}$在用户登录时向AS申请一次，可多次重复使用；
  - $Ticket_{tgs}$定义为$E_{Ktgs}[ID_C \| AD_C \| ID_{tgs} \| TS_1 \| LT_1]$。
- 服务许可票据（Service granting ticket）
  - 是客户访问服务器时需要提供的票据；
  - 用$Ticket_V$表示访问应用服务器V的票据。
  - $Ticket_V$定义为$EK_v[ID_C \| AD_C \| ID_V \| TS_2 \| LT_2]$。

票据启用时间

票据生存时间

更安全的认证对话

口令→密钥K_C→解密→获取$Ticket_{tgs}$

客户C与认证服务器AS认证服务阶段

(1) $C \rightarrow AS: ID_C || ID_{tgs}$

(2) $AS \rightarrow C: E_{K_C}[Ticket_{tgs}]$

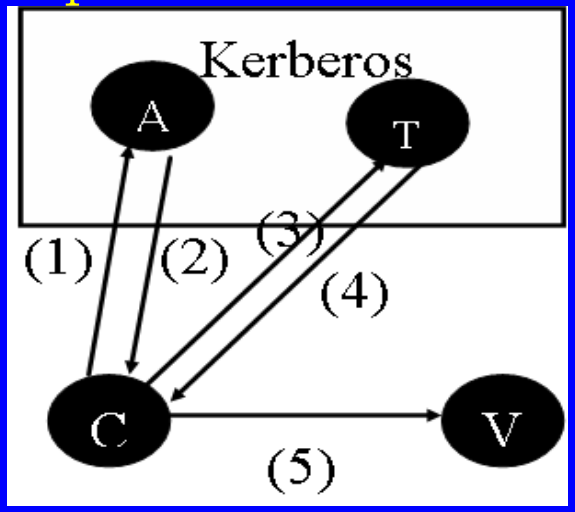
避免口令P_c明文传输

防止对手在获取$Ticket_{tgs}$后，等待C退出登录再冒充C登录

$Ticket_{tgs} = E_{K_{tgs}}[ID_C || AD_C || ID_{tgs} || TS_1 || Lifetime_1]$

票据重用，避免用户口令重复输入

仅AS和TGS知道K_{tgs}



利用票据许可票据申请服务许可票据

(3) $C \rightarrow TGS: ID_C || ID_v || Ticket_{tgs}$

(4) $TGS \rightarrow C: Ticket_V$

$TicketV = EK_v[ID_C || AD_C || ID_V || TS_2 || Lifetime_2]$

仅V和TGS知道K_V

特定服务许可票据访问相应的

(5) $C \rightarrow V: ID_C || Ticket_V$

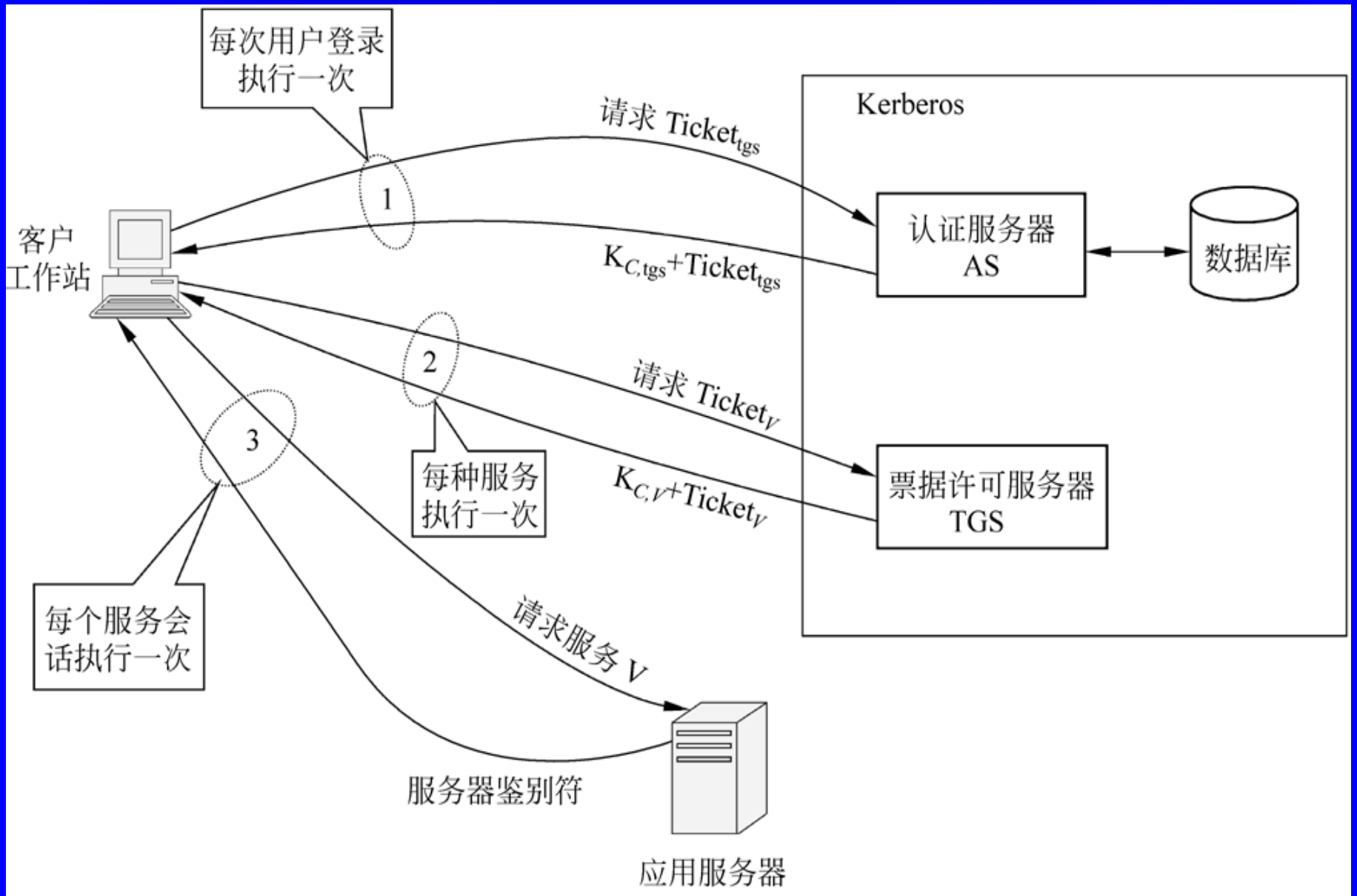
存在问题

1)票据许可票据有效期的确定。

2)用户对服务器身份的认证。

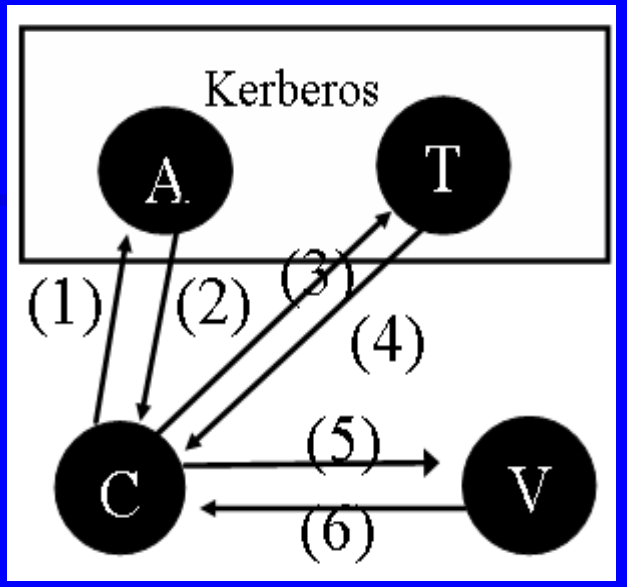
Kerberos V4进一步研究。

Kerberos V4认证过程示意图



Kerberos V4认证过程

采用会话密钥保证使用票据的人就是申请票据的人,时间戳避免票据重放攻击。即:由AS产生C和TGS之间的会话密钥,并安全地传送给C和TGS;由TGS产生C和V之间的会话密钥,并安全地传送给C和V。供客户C加密传输鉴别符$Authenticator_c$



第1阶段:认证服务器的交互,用于获取票据许可票据

(1) $C \rightarrow AS: ID_C \| ID_{tgs} \| TS_1$

使AS验证报文的及时性,验证客户时钟是否与AS同步

(2) $AS \rightarrow C: E_{Kc}[K_{C,tgs} \| ID_{tgs} \| TS_2 \| LT_2 \| Ticket_{tgs}]$

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

口令→密钥Kc→解密→获取$Ticket_{tgs}$

仅AS和TGS知道Ktgs

AS产生，C和TGS之间拥有了会话密钥

TGS通过$Ticket_{tgs}$获取$K_{c,tgs}$

Kerberos V4认证过程

第2阶段,票据许可服务器的交互,用于获取服务许可票据

(3) $C \rightarrow TGS: ID_V \| Ticket_{tgs} \| AU_C$

鉴别符

生存期很短且仅使用一次,证明客户的身份

(4) $TGS \rightarrow C: E_{Kc,tgs}[K_{C,V} \| ID_V \| TS_4 \| Ticket_V]$

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

仅V和TGS知道K_V

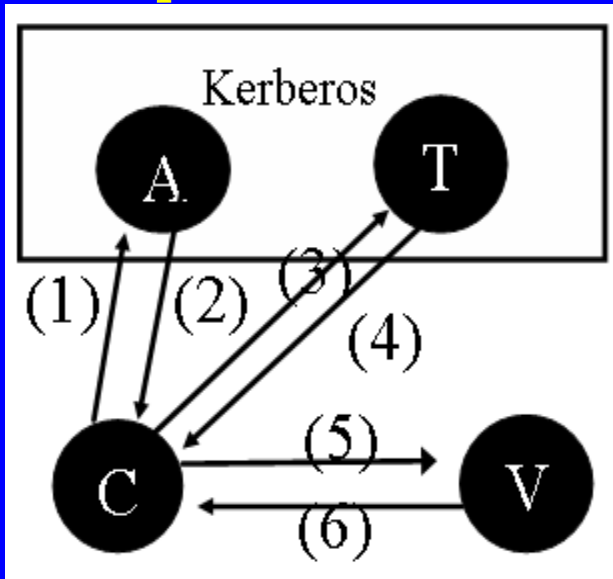
$Ticket_V = E_{Kv}[K_{C,V} \| ID_C \| AD_C \| ID_V \| TS_4 \| LT_4]$

$AU_C = E_{Kc,tgs}[ID_C \| AD_C \| TS_3]$

TGS产生，C和V共同拥有

V通过$Ticket_V$获取$K_{c,v}$

使得ADC是不可重用的,每个鉴别符只有很短的生存期



Kerberos V4认证过程(3)

第3阶段,客户与应用服务器的交互,用于获得服务

可重用票据

生存期很短且仅使用一次
证明$Ticket_v$拥有者的身份

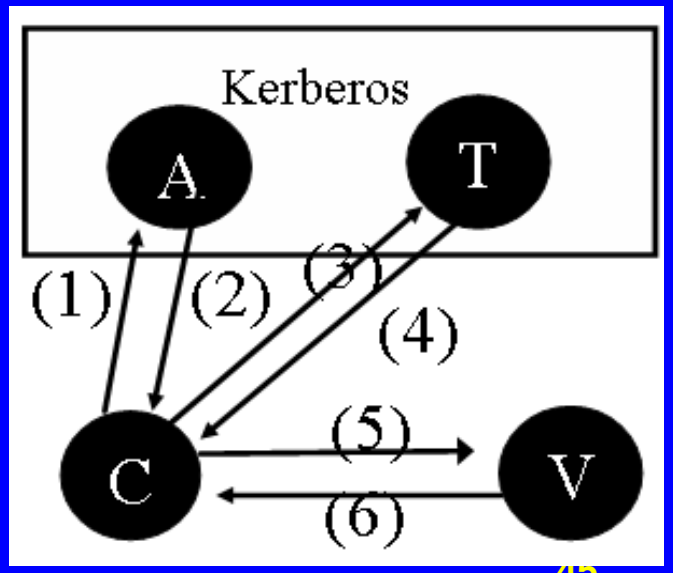
向用户鉴别他们自己

5) $C \rightarrow V: Ticket_V \| AU_C$

(6) $V \rightarrow C: E_{Kc,v}[TS_5 + 1]$

$Ticket_V = E_{Kv}[K_{C,V} \| ID_C \| AD_C \| ID_V \| TS_4 \| LT_4]$

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



Kerberos 领域(realm)

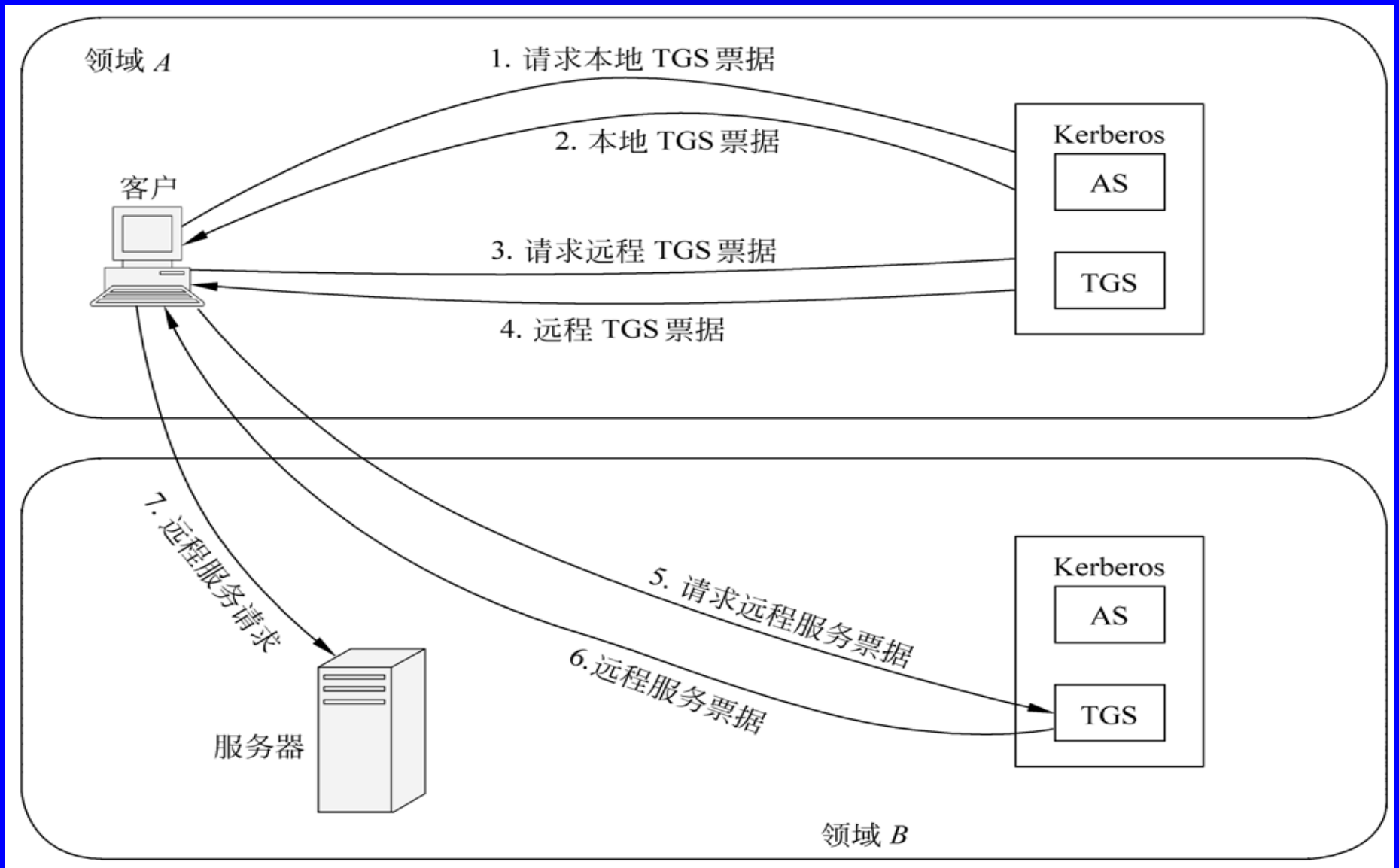
- 构成:一个完整的 Kerberos 环境包括一个Kerberos服务器,一组工作站和一组应用服务器。

- 具有如下特征
  - Kerberos服务器数据库中拥有所有的用户ID和口令散列码;所有用户需要向Kerberos服务器注册。
  - 每个应用服务器都与Kerberos服务器共享一个密钥
  - 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

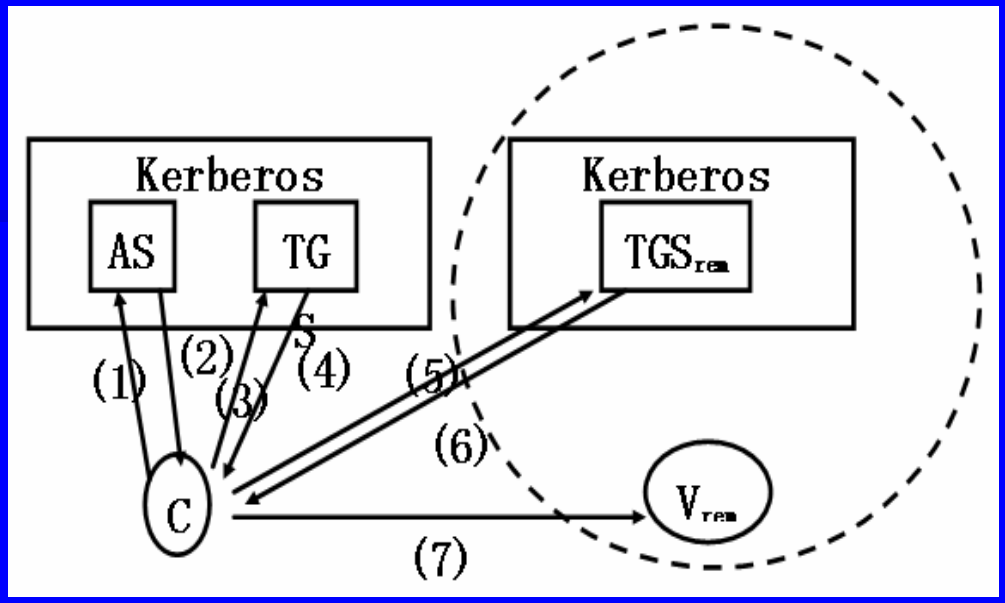
- 跨领域的服务访问
  - 一个用户可能需要访问另一个Kerberos 领域中应用服务器；
  - 一个应用服务器也可以向其他领域中的客户提供网络服务。
- 领域间互通的前提
  - 支持不同领域之间进行用户身份鉴别的机制；
  - 互通领域中的 Kerberos 服务器之间必须共享一个密钥；
  - 两个 Kerberos 服务器也必须进行相互注册。

远程服务访问的认证过程



Kerberos跨领域认证交互

向远程服务器Vrem出示票据的用户已经在其本领域内经过了身份认证



(1) $C \rightarrow AS: ID_C \| ID_{tgs} \| TS_1$

(2) $AS \rightarrow C: E_{Kc}[K_{C,tgs} \| ID_{tgs} \| TS_2 \| LT_2 \| Ticket_{tgs}]$

(3) $C \rightarrow TGS: ID_{tgsrem} \| Ticket_{tgs} \| AU_C$

(4) $TGS \rightarrow C: E_{Kc,tgs}[K_{C,tgsrem} \| ID_{tesrem} \| TS_4 \| Ticket_{tgsrem}]$

(5) $C \rightarrow TGS_{rem}: ID_{Vrem} \| Ticket_{tgsrem} \| AU_C$

(6) $TGS_{rem} \rightarrow C: E_{Kc,tgsrem}[K_{C,Vrem} \| ID_{Vrem} \| TS_6 \| Ticket_{Vrem}]$

(7) $C \rightarrow V_{rem}: Ticket_{Vrem} \| AU_C$

V_{rem}远程应用服务器，TGS_{rem}远程TGS服务器。

服务器可以根据自身安全策略决定是否允许该用户的远程服务请求

Kerberos v4的缺陷

- 依赖性:加密系统对IP协议的依赖性和对时间依赖性
- 字节顺序:用户自定义
- 票据有效期:8位字段来表示,有效期最小为5分钟,最大约为21小时
- 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
- 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
- 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
- 提供了鉴别转发能力
- 更有效的方法来解决领域间的认证问题
- 抗口令攻击:提供了一种预鉴别机制,使口令攻击更加困难,但不能完全避免口令攻击

问题：

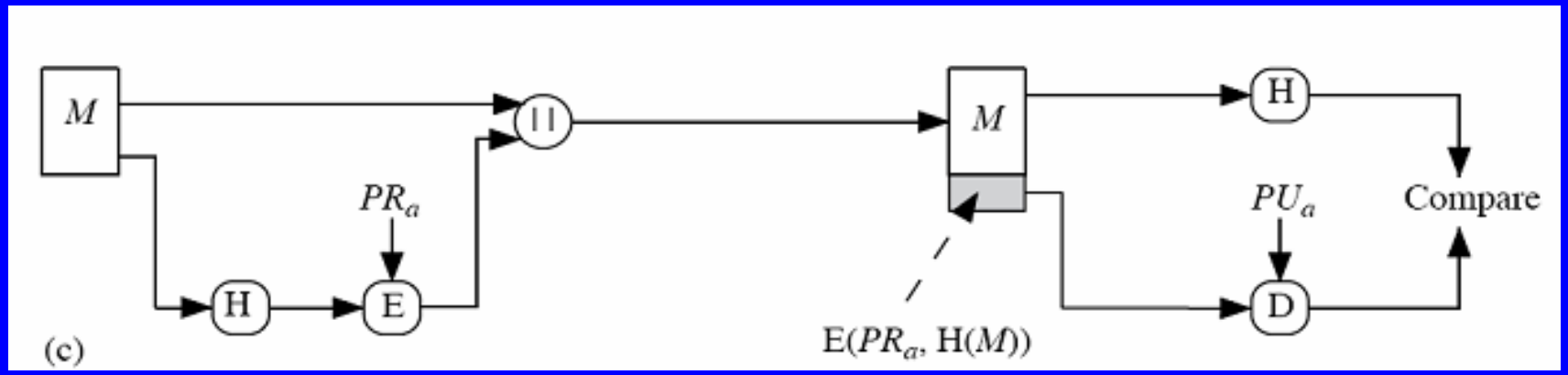
1、数字签名是否需要保证消息的机密性？能否保证消息的完整性？

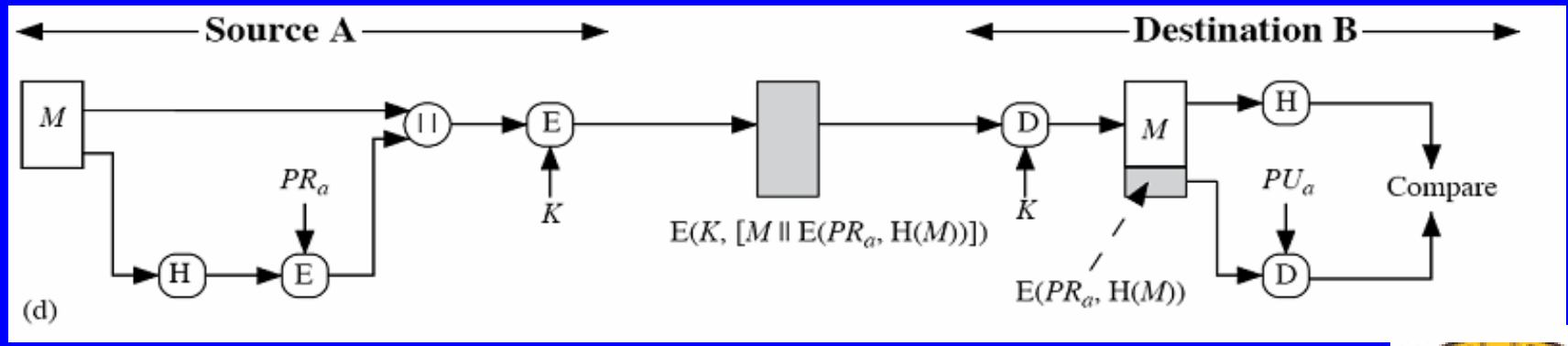
2、为什么对称加密算法不能用于数字签名？但仲裁数字签名中用到了对称加密算法？

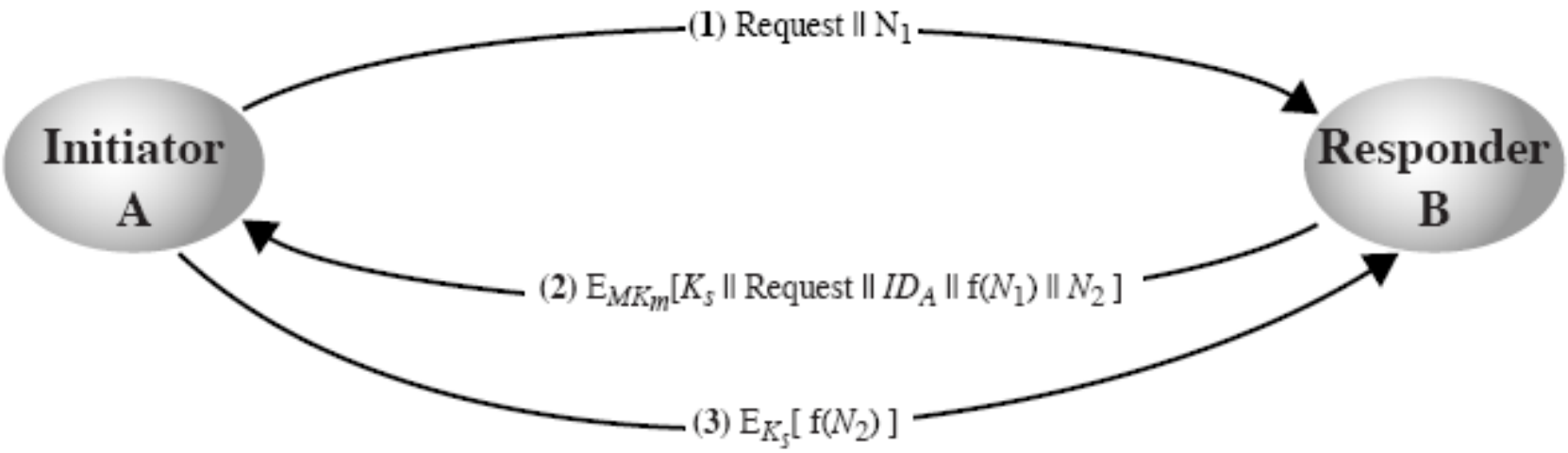
3、阻止重放攻击的方法有哪些？区别是什么？应用场景是什么？

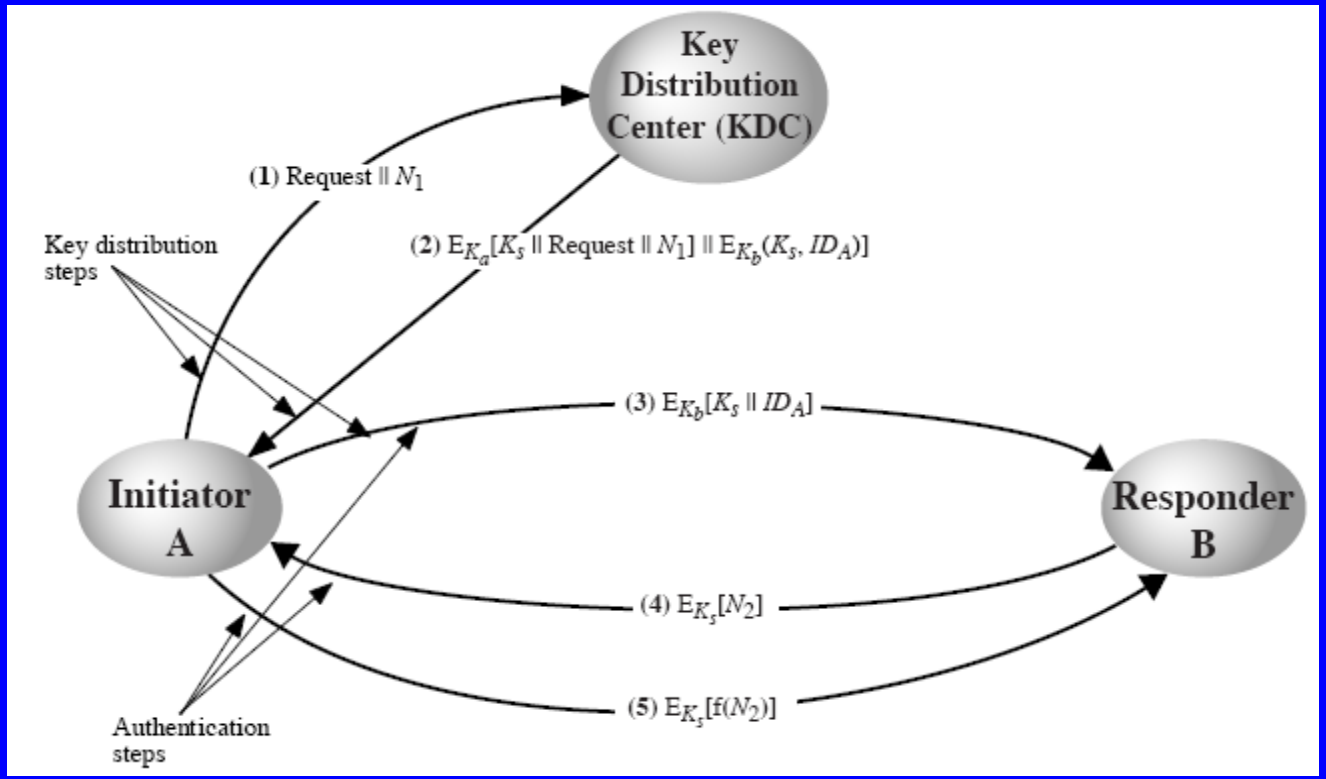
4、DSA算法中，为什么每次签名时都必须产生新的k值？

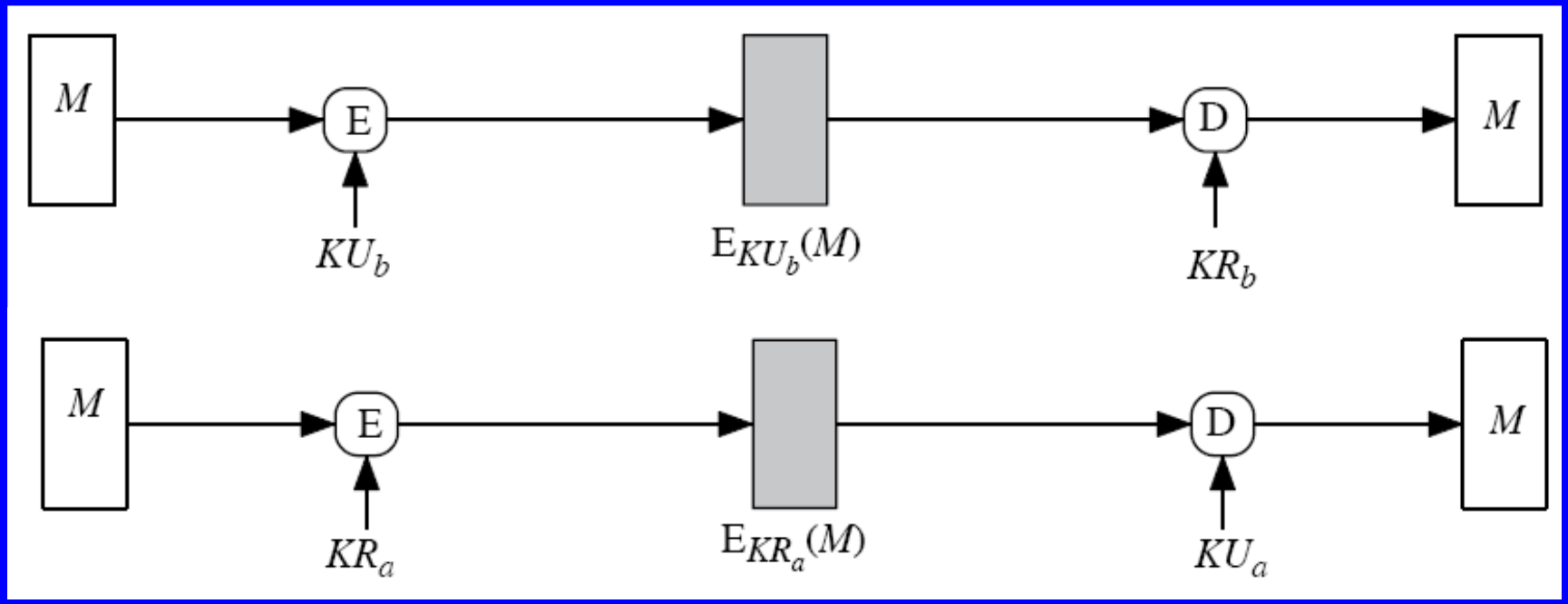
Thank You !

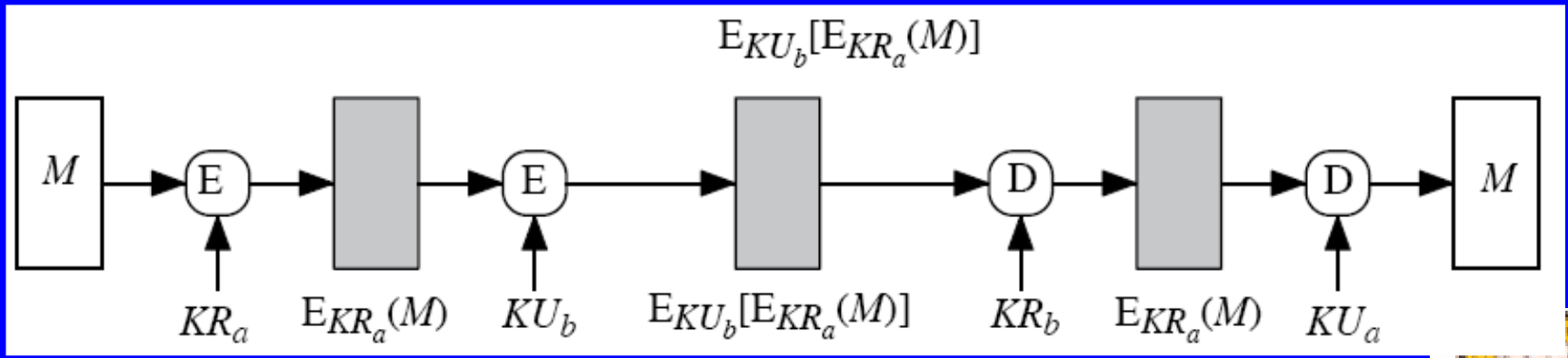

M
||
H
E
PR_a
M
H
PU_a
D
Compare
$E(PR_a, H(M))$
(c)


Source A
Destination B
M
||
E
K
H
E
PR_a
$E(K, [M \| E(PR_a, H(M))])$
D
K
M
H
PU_a
D
Compare
$E(PR_a, H(M))$
(d)


(1) Request || N1
Initiator A
Responder B
(2) EMKm[Ks || Request || IDA || f(N1) || N2]
(3) EKs[f(N2)]


Key Distribution Center (KDC)
(1) Request || N1
(2) EKa[Ks || Request || N1] || EKb(Ks, IDA)]
Key distribution steps
(3) EKb[Ks || IDA]
Initiator A
Responder B
(4) EKs[N2]
(5) EKs[f(N2)]
Authentication steps


M
E
KU_b
$E_{KU_b}(M)$
D
KR_b
M
M
E
KR_a
$E_{KR_a}(M)$
D
KU_a
M


$E_{KU_b}[E_{KR_a}(M)]$
M
E
KR_a
$E_{KR_a}(M)$
E
KU_b
$E_{KU_b}[E_{KR_a}(M)]$
D
KR_b
$E_{KR_a}(M)$
D
KU_a
M