第 5 章　密钥管理

第 5 章 密钥管理

- 5.1 密钥管理简介
- 5.2 对称密钥分发
- 5.3 公钥分发
- 5.4 X.509协议
- 5.5 公钥基础设施PKI

5.1 密钥管理简介

- 密钥----秘密的精华
- 密钥是一个巨大的数字
  - 数字本身的大小不重要，重要的是密钥空间的大小
  - 密钥空间的大小是由密钥长度决定的
  - 信息的机密性不依赖于加密算法本身，而依赖于密钥的安全性
- 复习：请列举以下对称密码算法的密钥长度

  DES、3DES、AES、RC4

5.1 密钥管理简介

- 由于密码算法是公开的，网络的安全性就完全基于密钥的安全保护上。因此在密码学中出现了一个重要的分支——密钥管理
- 密钥管理就是在授权各方之间实现密钥关系的建立与维护的一整套技术和程序。
- 密钥管理包括
  - 确保产生的密钥具有必要的特性
  - 通信双方事先约定密钥的方法
  - 密钥的保护

密钥管理的原则

- 全程安全原则
- 最小权利原则
- 责任分离原则
- 密钥分级原则
- 密钥更换原则
- 密钥应当有足够的长度
- 密钥体制不同，密钥管理不同

密钥的生命存期

- 所有密钥都有生命存期，原因如下：
  - 大量的密文有助于密码分析，一个密钥使用得太多了，会给攻击者增大收集密文的机会
  - 假定一个密钥受到危及或用一个特定密钥的加密/解密过程被分析，则限定密钥的使用期限就相当于限制危险的发生
- **密钥的生命周期**：指授权使用该密钥的周期，在一定的安全策略指导下完成密钥从产生到最终销毁的整个过程，包括密钥的**生成、建立（分配和协商）、存储、托管、使用、备份/恢复、更新、撤销和销毁等。**

密钥管理简介

- **密钥生成**：是密钥生命周期的基础阶段
  - 用户可以自己生成密钥，也可以从可信中心或密钥管理中心申请密钥。
  - 不同密钥体制，密钥的生成方法不同
  - 密钥长度要适中
- **密钥建立**：使密钥安全到达密钥使用的各实体对象，通常分为密钥分配和密钥协商。
- **密钥存储**：对静态密钥的保护

密钥管理简介

- **密钥使用：**利用密钥进行正常的密码操作，如加密、解密、签名等，通常情况下，密钥在有效期之内都可以使用。
- **密钥备份：**指密钥处于使用状态时的短期存储，为密钥的恢复提供密钥源，要求安全存储，防止泄密。
- **密钥恢复：**从备份或者存档中获取密钥的过程。若密钥丧失但未被泄露，可以用安全方式从密钥备份中恢复。

密钥管理简介

- **密钥存档：**存档指过了有效期的密钥进行长期离线保存，以便在需要时（如解决争议）能够对其进行检索。
- **密钥托管：**为政府机构提供了实施法律授权下的监听功能。

密钥管理简介

- **密钥更新：**在密钥有效期快结束时，为了继续使用相应密码体制，保证密钥的安全性，该密钥需要由一个新的密钥代替，这就是密钥更新。
- **密钥撤销：**由于某种原因（如丢失）需要在密钥过期之前将其从正常使用的集合中删除。
- **密钥销毁：**对于不再需要使用的密钥，将其所有复本销毁，不能再出现。

密钥的状态

- **使用前状态：**密钥不能用于正常的密码操作。
- **使用状态：**密钥可用于正常的密码操作，并处于正常使用中。
- **使用后状态：**密钥不再正常使用，但为了某种目的对其进行离线访问是可行的。
- **过期状态：**密钥不再使用，所有的密钥记录已被删除。

密钥的种类

- 会话密钥（Session Key）
  - 一次通信或数据交换中，用户之间所使用的密钥。
  - 用户之间协商得到的，也称为数据加密密钥(Data Encrypting Key)，网内分配
- 密钥加密密钥(Key Encrypting Key)
  - 二级密钥，对传输的会话密钥进行加密的密钥
- 主密钥（Master Key）
  - 层次化密钥结构中的最高层次，对密钥加密密钥进行加密的密钥
  - 需要严格保护，一般会长期使用，网外分配。

密钥的保护

- 密钥从产生到销毁的整个生存期都需要加强保护
- 密钥的完整性和机密性都需要保护
- 密钥安全存储：放在物理上安全的地方
- 密钥安全传输
  - 由一个可信方来分配
  - 将一个密钥分成两部分，委托给两个不同的人或机构
  - 通过机密性和/或完整性服务来保护

密钥分配

- 密钥分配是密钥管理中最大的问题，密钥系统的强度取决于密钥分发技术
  - 必须通过最安全的通路进行分配
  - 需要频繁改变密钥来减少某个攻击者可能知道密钥带来的数据泄密

密钥分配

- **密钥分配**指传递密钥给希望交换数据的双方，且不允许其他人看见密钥的方法。
- **网外分配方式：**派非常可靠的信使携带密钥分配给互相通信的各用户。
- **网内分配方式**：密钥自动分配。

但随着用户的增多和网络流量的增大，密钥更换频繁（密钥必须定期更换才能做到可靠），派信使的办法已不再适用，而应采用网内分配方式。

5.2 对称密钥分配

- 5.2.1 基于对称加密的对称密钥分配
- 5.2.2 基于非对称加密的对称密钥分配

5.2.1 基于对称加密的对称密钥分发

- 用于加密大部分数据的密钥需要频繁更改（例如：每次会话更改一次）
- 密钥主要分为两类：会话密钥(数据密钥)和密钥加密密钥（主密钥）
  - 会话密钥：用于保护大部分数据
  - 密钥加密密钥：用于保护会话密钥

基于对称加密的对称密钥分发

- 多种可能的方法实现，举例：A、B双方

  (1)A能够选定密钥并通过物理方法传递给B

  (2)第三方可以选定密钥并通过物理方法传递给A和B

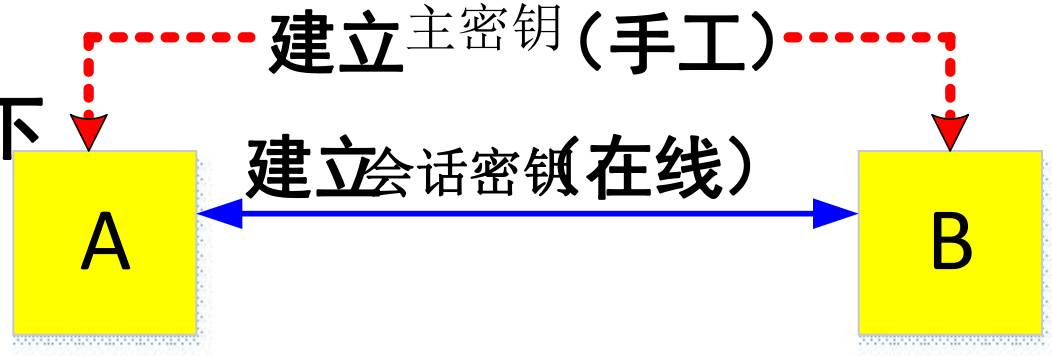
  (3)如果A和B不久之前使用过一个密钥，一方能够把使用旧密钥加密的新密钥传递给另一方

  (4)如果A和B各自有一个到达第三方C（密钥管理中心）的加密链路，C能够在加密链路上传递密钥给A和B

- 密钥分发两类基本结构：点到点结构、密钥中心结构

点到点结构

- 通信双方共享主密钥，没有涉及到其他成员
  - 一方产生新的会话密钥
  - 会话密钥在主密钥保护下发送给另一方

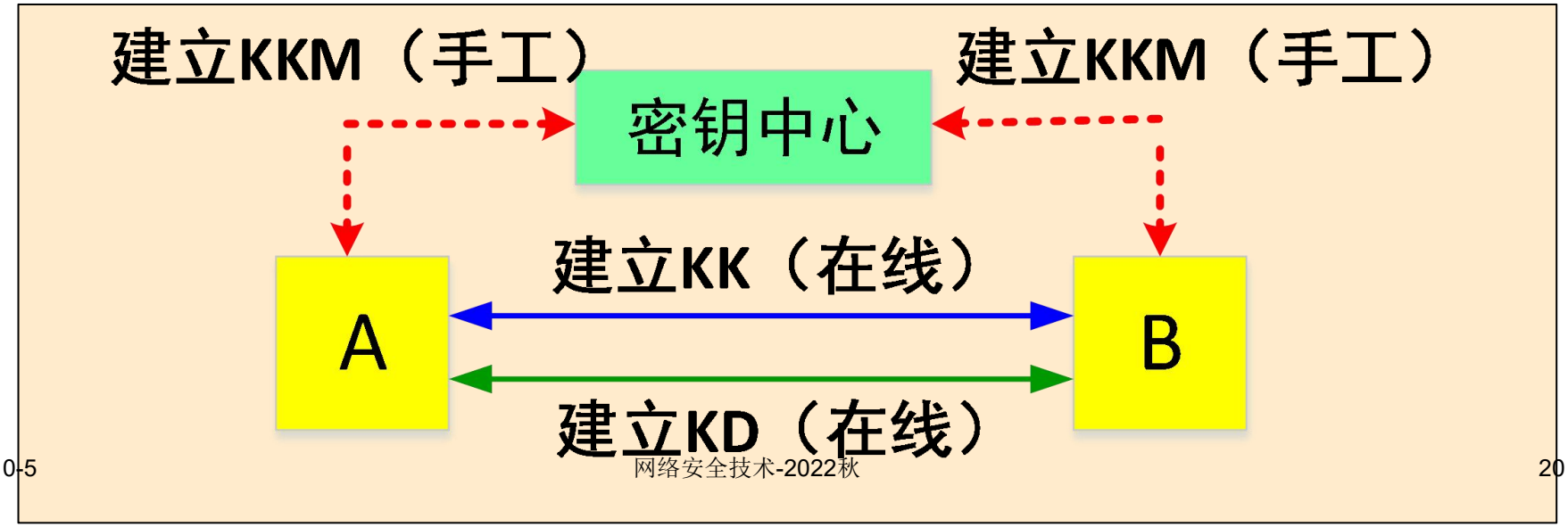


- 存在的问题：成员数量多时，密钥分配难处理

  例如：n个成员互相通信，需要n(n-1)/2个主密钥
- 不适用于大型网络使用
- 解决方案：在主密钥分配结构中引入密钥中心

密钥中心KDC

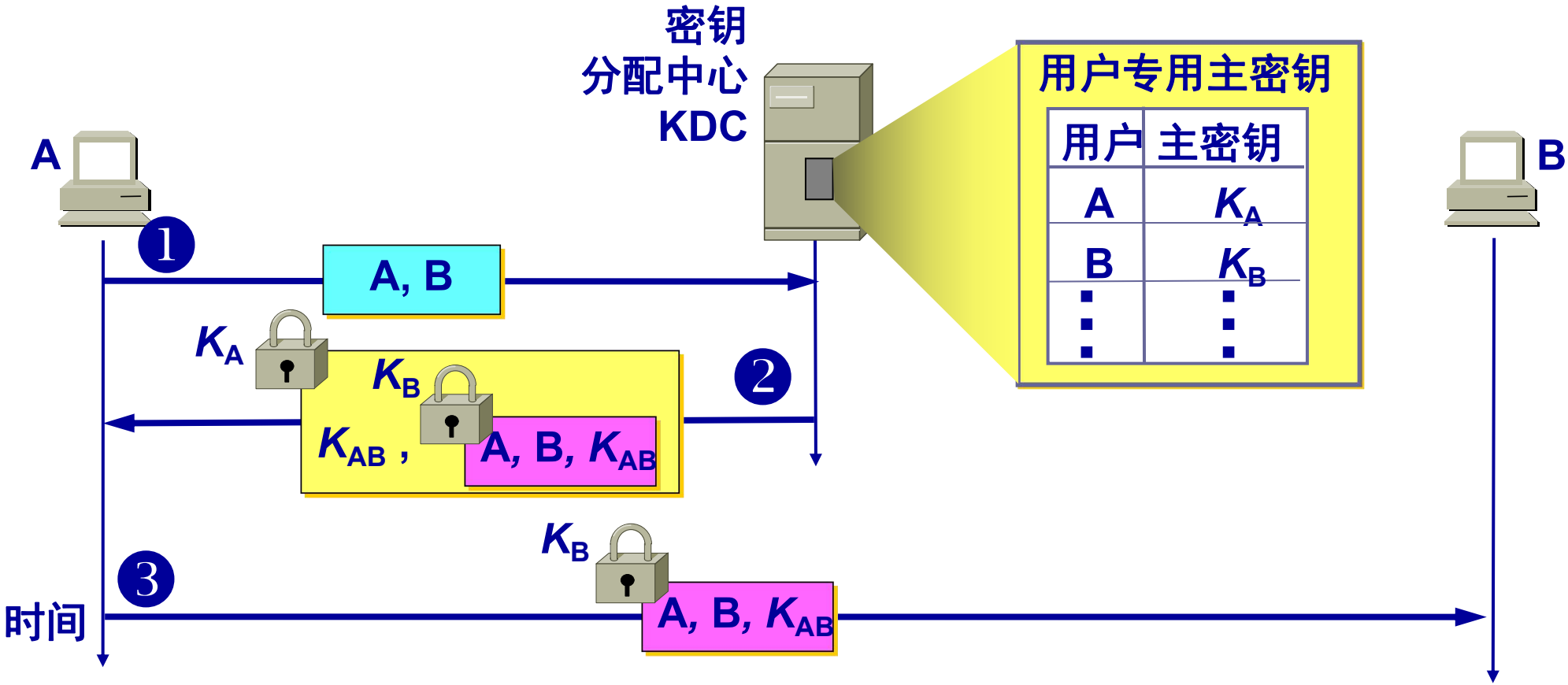
- 密钥分发中心KDC是基于密钥层次体系的，至少需要两个密钥层
- 每个通信方和密钥中心共享一个主密钥，通信方之间无共享的主密钥，因此n个成员手工分配的主密钥数量为n



密钥分配中心KDC

- 目前常用的密钥分配方式是设立密钥分配中心 KDC (Key Distribution Center)。
- KDC 是大家都信任的机构，其任务就是给需要进行秘密通信的用户临时分配一个会话密钥（仅使用一次）。
- 假设用户 A 和 B 都是 KDC 的登记用户，并已经在 KDC 的服务器上安装了各自和 KDC 进行通信的主密钥（master key）KA 和 KB。“主密钥”可简称为“密钥”。

密钥分发方案1


密钥
分配中心
KDC
用户专用主密钥
用户
主密钥
A
K_A
B
K_B
A
B
1
A, B
K_A
K_B
K_{AB} ,
A, B, K_{AB}
2
K_B
3
A, B, K_{AB}
时间

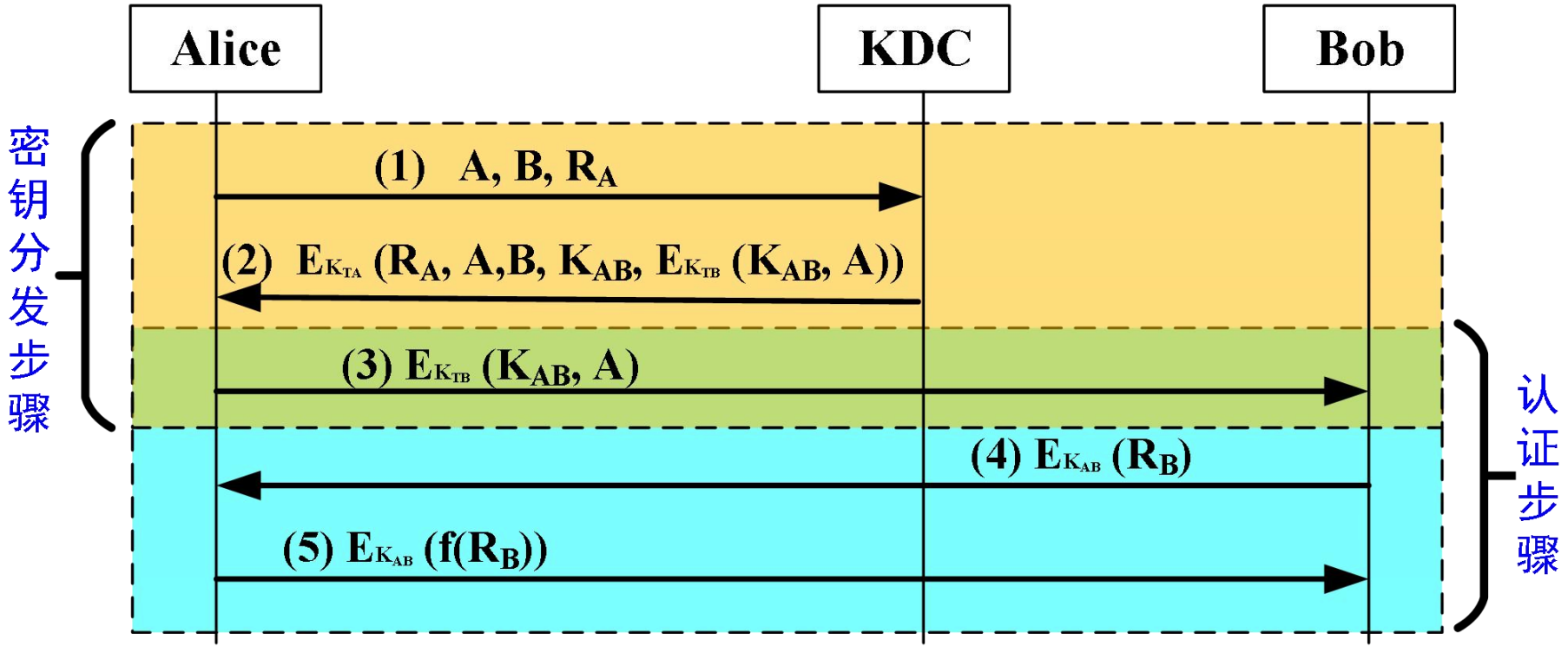
对称密钥的分配说明

- ❶ 用户 A 向密钥分配中心 KDC 发送时用明文，说明想和用户B通信。在明文中给出 A 和 B 在 KDC 登记的身份。
- ❷ KDC 用随机数产生“一次一密”的会话密钥 K_{AB} 供 A 和 B 的这次会话使用，然后向 A 发送回答报文。这个回答报文用 A 的密钥 K_A 加密。这个报文中包含有这次会话使用的密钥 K_{AB} 和请 A 转给 B 的一个**票据(ticket)**，它包含 A 和 B 在 KDC 登记的身份，以及这次会话将要使用的密钥 K_{AB}。这个票据用B的密钥 K_B 加密，因此 A 无法知道此票据的内容，因为 A 没有 B 的密钥 K_B。当然 A 也不需要知道此票据的内容。

对称密钥的分配说明

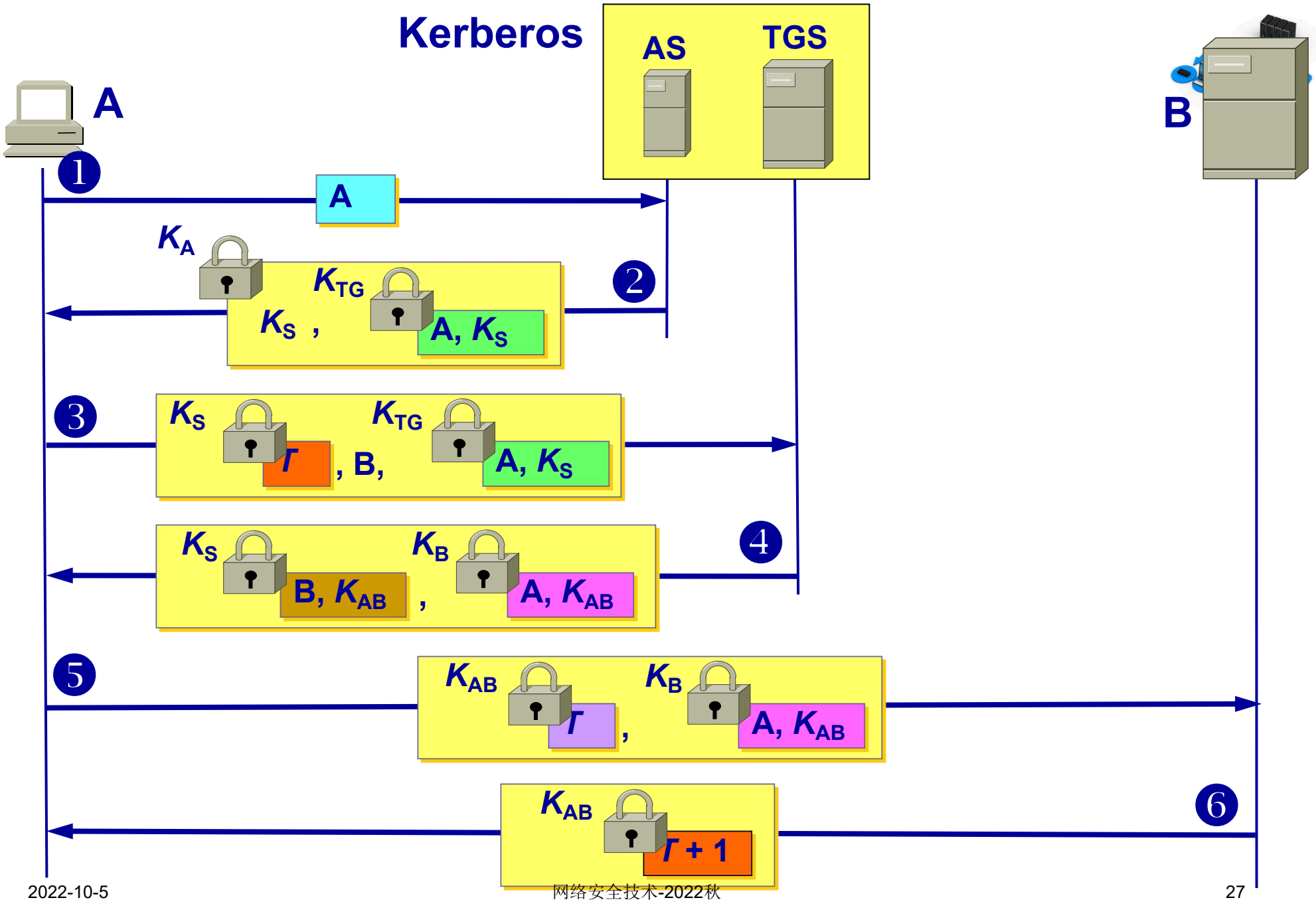
- ❸ 当 B 收到 A 转来的票据并使用自己的密钥 K_B 解密后，就知道 A 要和他通信，同时也知道 KDC 为这次和 A 通信所分配的会话密钥 K_{AB} 。
- 此后，A 和 B 就可使用会话密钥 K_{AB} 进行这次通信了。

密钥分发方案2


Alice
KDC
Bob
密钥分发步骤
(1) A, B, R_A
(2) $E_{K_{TA}}$ (R_A, A,B, K_{AB}, $E_{K_{TB}}$ (K_{AB}, A))
(3) $E_{K_{TB}}$ (K_{AB}, A)
(4) $E_{K_{AB}}$ (R_B)
(5) $E_{K_{AB}}$ (f(R_B))
认证步骤

举例：Kerberos

- 目前最出名的密钥分配协议是 Kerberos。
- Kerberos 既是鉴别协议，同时也是 KDC，它已经变得很普及，现在是互联网建议标准。
- Kerberos 使用比 DES 更加安全的高级加密标准 AES 进行加密。
- Kerberos 使用两个服务器：鉴别服务器AS (Authentication Server)、票据授予服务器TGS (Ticket-Granting Server)。
- Kerberos 只用于客户与服务器之间的鉴别，而不用于人对人的鉴别。


Kerberos
AS
TGS
A
B
1
A
K_A
K_{TG}
K_S ,
A, K_S
2
3
K_S
T
, B,
K_{TG}
A, K_S
4
K_S
B, K_{AB}
,
K_B
A, K_{AB}
5
K_{AB}
T
,
K_B
A, K_{AB}
6
K_{AB}
$T + 1$

Kerberos密钥分配说明

- ❶ A 用明文（包括登记的身份）向鉴别服务器 AS 表明自己的身份。
- ❷ AS 向 A 发送用 A 的对称密钥 K_A 加密的报文，这个报文包含A和TGS通信的会话密钥 K_S，以及 AS 要发送给 TGS 的票据（这个票据是用 TGS 的对称密钥 K_{TG} 加密的）。

Kerberos密钥分配说明

- ❸ A 向 TGS 发送三个项目：
  - 转发鉴别服务器 AS 发来的票据。
  - 服务器 B 的名字。这表明 A 请求 B 的服务。请注意，现在 A 向 TGS 证明自己的身份并非通过键入口令（因为入侵者能够从网上截获明文口令），而是通过转发 AS 发出的票据（只有 A 才能提取出）。票据是加密的，入侵者伪造不了。
  - 用 K_S 加密的时间戳 T。它用来防止入侵者的重放攻击。

Kerberos密钥分配说明

- ❹ TGS 发送两个票据，每一个都包含 A 和 B 通信的会话密钥 K_{AB}。给 A 的票据用 K_S 加密；给 B 的票据用 B 的密钥 K_B 加密。请注意，现在入侵者不能提取 K_{AB}，因为不知道 K_A 和 K_B。入侵者也不能重放步骤❸，因为入侵者不能把时间戳更换为一个新的（因为不知道 K_S）。
- ❺ A 向 B 转发 TGS 发来的票据，同时发送用 K_{AB} 加密的时间戳 T。
- ❻ B 把时间戳 T 加 1 来证实收到了票据。B 向 A 发送的报文用密钥 K_{AB} 加密。
- 以后，A 和 B 就使用 TGS 给出的会话密钥 K_{AB} 进行通信。

5.2.2 基于非对称加密的对称密钥分发

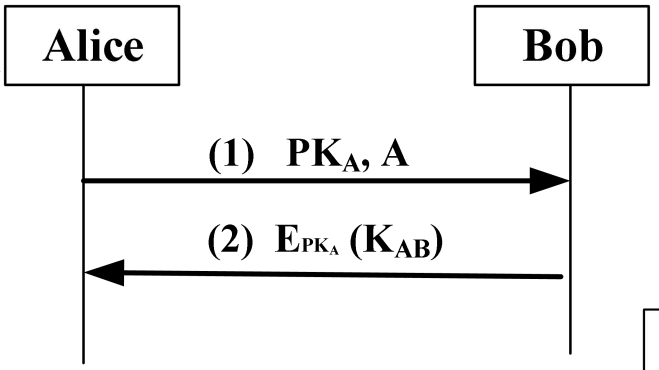
- 对称密码体制缺点
  - 使用对称密码体制，必须维护许多对密钥关系并配置可信在线密钥中心
  - 使用公钥密码体制只需要维护较少的密钥关系，公钥分配时无需机密性保护
- 公钥密码体制便于密钥管理，适合大型网络使用
- 公钥密码体制不足：处理速度慢
- 因此：采用公钥密码体制分发密钥，采用对称密码体制进行数据加密

基于公钥密码体制分发密钥的举例

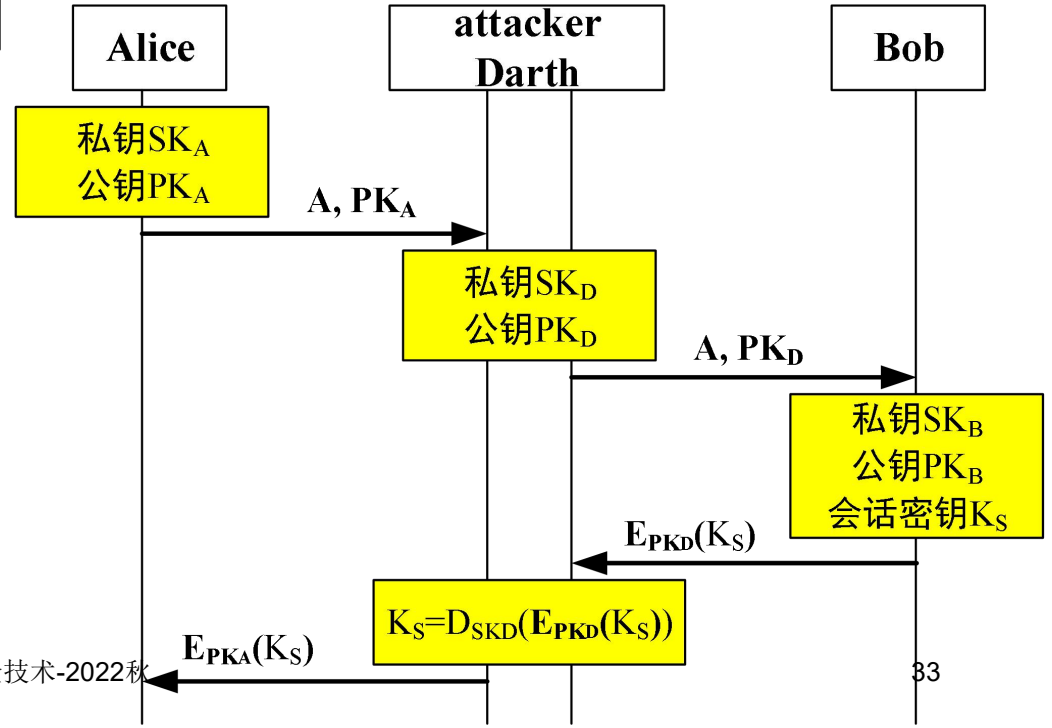
- 场景：成员A与B使用RSA建立对称初始密码
- 步骤
  - 成员A获得成员B的公钥（如何知道该公钥的确属于B?）
  - 成员A随机地产生一个对称密钥K,将K用B的公钥加密后发送给B
  - 成员B用自己的私钥解密后获得K
- 不需在线服务器和成员之间的协商，适合于诸如电子邮件等应用
- 缺点：如果密码系统被破坏，则所有被保护的初始密钥以及被这些初始密钥保护的信息都将受到危及

简单密钥分发方案

- 目的：使得两个用户能够安全地交换密钥
- 基本方法



- 存在的问题：中间人攻击



5.3 公钥分发

- 公钥密码体制的密钥分配与对称密码体制的密钥分配有着本质的差别
  - 在公钥密码体制中，如果每个用户都具有其他用户的公钥，就可实现安全通信。
  - 在公钥密码体制中，秘密密钥只有通信一方知道，其余任何一方都不知道，公钥是公开的。
- 分配公钥：不需要机密性，但需要完整性保护。

伪造攻击举例

- 举例1：设想用户 C 要欺骗用户 B。C可以向 B 发送一份伪造是 A 发送的报文。C用自己的秘密密钥进行数字签名，并附上 C自己的公钥，谎称这公钥是 A 的。这样攻击者成功地冒充了成员A
- 问题：B 如何知道这个公钥不是 A 的呢？
- 因此：公钥的分配不像公布电话号码这样简单，需要以某种特定的方式来分配。

公钥证书

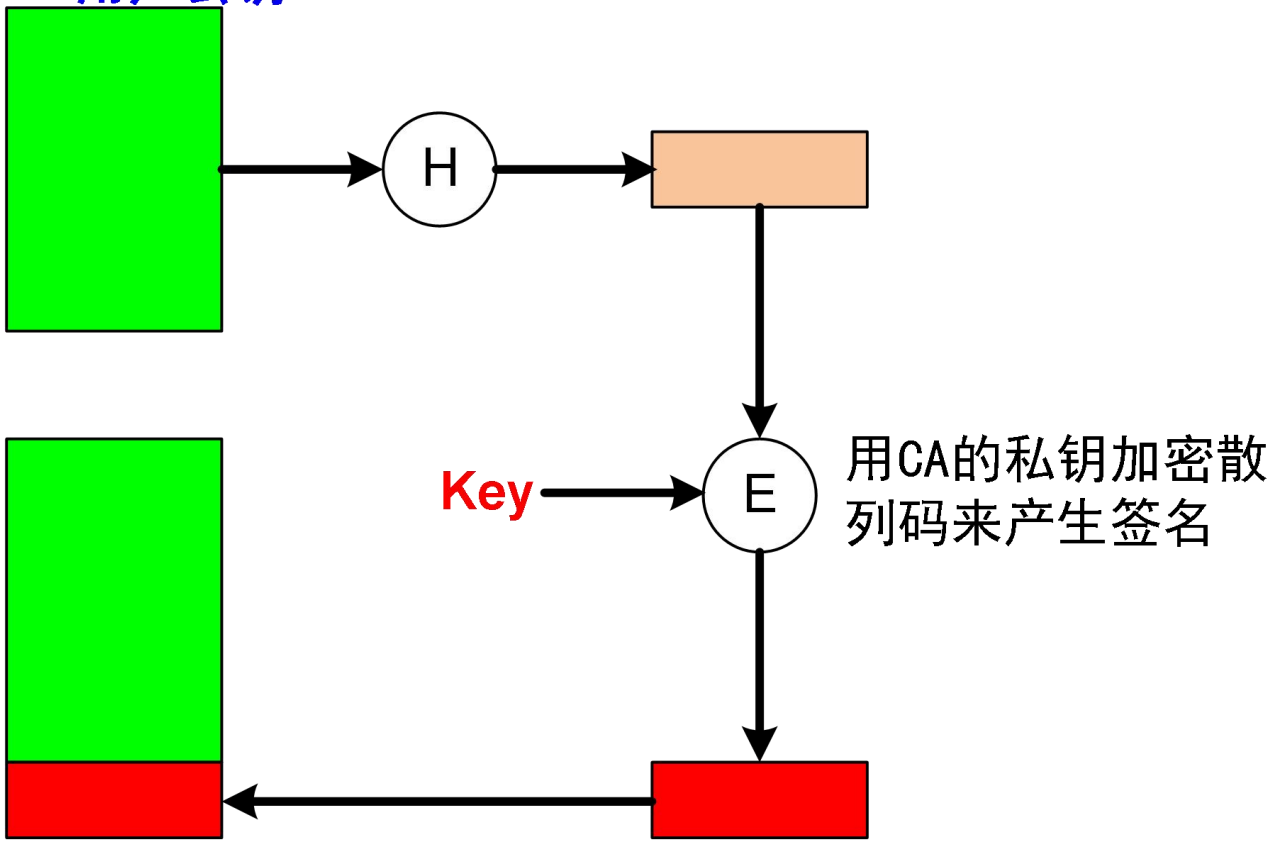
- **证书**：是一个数据结构，由证书用户可信的某一成员进行数字签名
- **公钥证书**：一个数据结构，用来绑定**实体姓名**（以及有关该实体的其他属性）和**相应的公钥**

公钥证书=用户ID+用户公钥+可信第三方的签名

证书

没有经过签名的证书：

用户ID+用户公钥



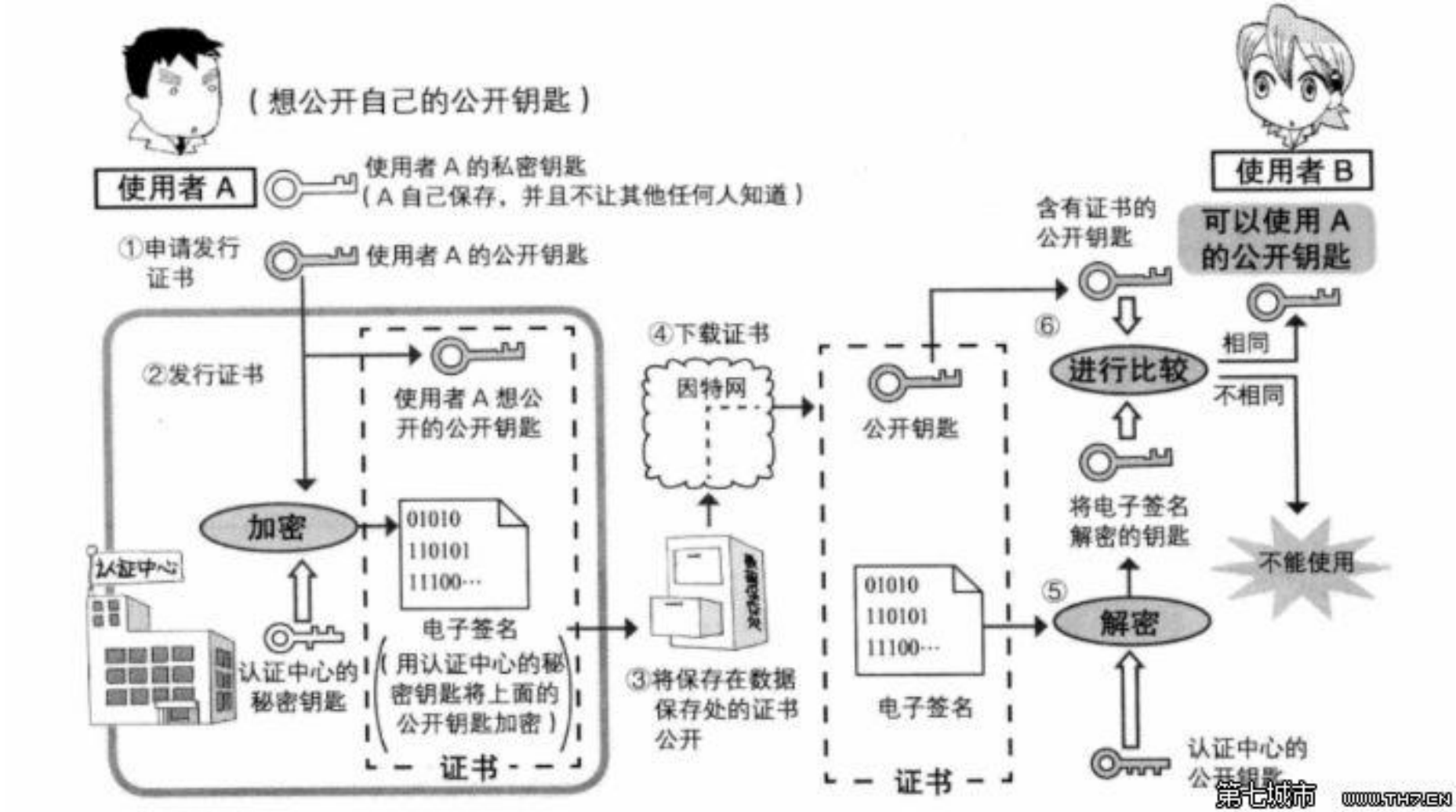
经过签名的证书：

用户ID+用户公钥+CA的签名

证书

- 证书格式类型
  - X.509公钥证书:广泛应用于IP安全、TLS、SET和S/MIME等
  - PGP（pretty good privacy）证书
  - 属性证书

公钥分发示意图


（想公开自己的公开钥匙）
使用者 A
使用者 A 的私密钥匙
（A 自己保存，并且不让其他任何人知道）
①申请发行证书
使用者 A 的公开钥匙
②发行证书
使用者 A 想公开的公开钥匙
加密
01010
110101
11100…
电子签名
（用认证中心的秘密钥匙将上面的公开钥匙加密）
认证中心
认证中心的秘密钥匙
证书
④下载证书
因特网
③将保存在数据保存处的证书公开
公开钥匙
01010
110101
11100…
电子签名
证书
使用者 B
含有证书的公开钥匙
可以使用 A 的公开钥匙
⑥
进行比较
相同
不相同
将电子签名解密的钥匙
不能使用
⑤
解密
认证中心的公开钥匙
第七城市
WWW.THZ.CN

5.4 X.509证书

- 证书具有统一的格式，ITU-T 制定了 X.509 协议标准，用来描述证书的结构，使用 ASN.1。
- IETF 接受了 X.509（仅少量改动），并在 RFC 5280（现在是建议标准）中给出了互联网 X.509 公钥基础结构 PKI (Public Key Infrastructure)。
- PKI：支持公开密钥管理并能支持认证、加密、完整性和可追究性服务的基础设施。
- X.509是一个重要的协议，其定义的证书结构和认证协议在互联网的很多环境下使用。

X.509证书

- X.509方案的核心是与每个用户相关联的公钥证书
  - 用户的公钥证书由可信任的认证中心（Certification authority, CA）创建
  - 在目录服务器的目录中存放
- CA生成的证书的特征
  - 任何可以访问CA公钥的用户都可以验证此经过签名的用户公钥
  - 只有认证中心才可以修改用户证书而不会被发现

认证机构和证书机构（CA）

- 需要有一个值得信赖的机构——即认证机构或认证中心 CA (Certification Authority)，来将公钥与其对应的实体（人或机器）进行绑定(binding)。
- 认证中心一般由政府出资建立。每个实体都有 CA 发来的证书(certificate)，里面有公钥及其拥有者的标识信息。此证书被 CA 私钥进行了数字签名。任何用户都可从可信的地方获得认证中心 CA 的公钥，此公钥用来验证某个公钥是否为某个实体所拥有。
- 有的大公司也提供认证中心服务。

X.509证书结构和语义

- X.509基于公钥加密体制和数字签名
  - 加密没有规定具体的算法，推荐使用RSA
  - 数字签名方案假定需要使用散列函数,但未强制使用某种特定的散列函数
- X.509版本3的证书结构（P95，图4.4）

版本号	序列号	签名	颁发者	有效期	主体	主体公钥信息	颁发者唯一标识符	主体唯一标识符	扩展

  - 签名：签名算法和参数（在证书末尾的签名域被重复）
  - 主体公钥信息：主体的公钥（算法、参数、密钥）

X.509证书结构和语义

$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, UCA, A, UA, A_p, T^A\}$

Y<<X>>=由认证中心Y发放的用户X的证书

Y{I}=Y对I的签名，包括I并在其后增加一个加密过的散列码

V=证书的版本

SN=证书的序列号

AI=签名算法标识符

CA=认证中心名称

UCA=可选的CA唯一标识符

A=用户A名称

UA=可选的用户A唯一标识符

A_p=用户A的公钥

T^A=证书的有效期

CA用其私钥对证书进行签名，用户利用CA的公钥验证CA签名过的证书是否合法

证书验证

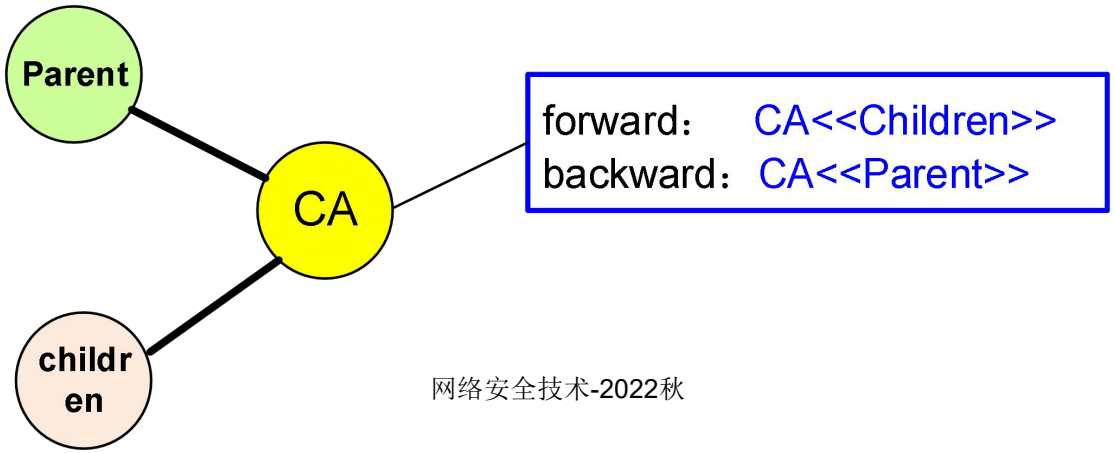
- 检查一份给定的证书是否可用的过程成为证书验证
  - 一个可信CA已经在证书上签了名，即CA的数字签名被验证是正确的
  - 证书有良好的完整性
  - 证书处在有效期内
  - 证书没有被撤销
  - 证书的使用方式与任何声明的策略和使用限制相一致

证书认证过程

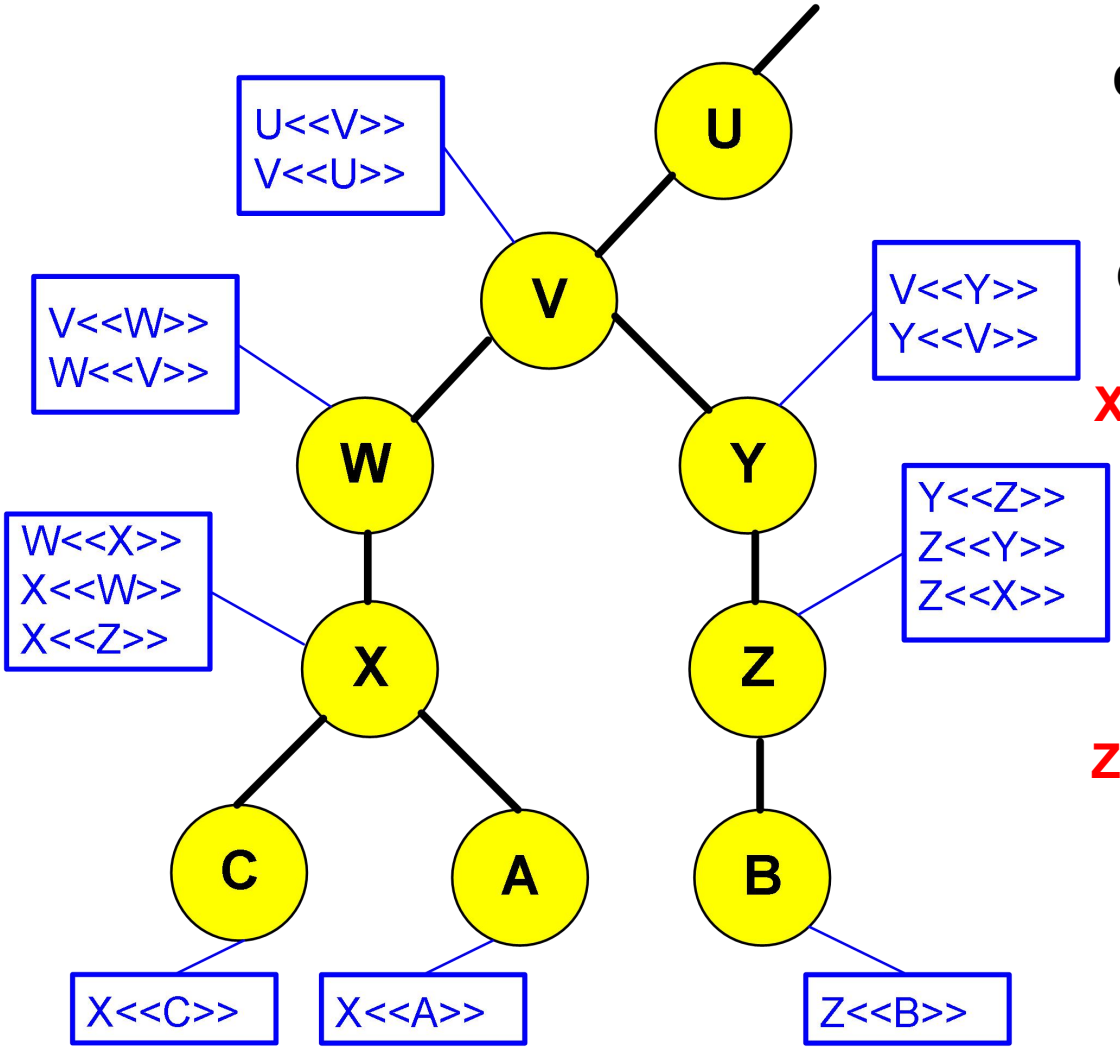
- 不同CA之间证书的验证过程
  - 场景：Alice与Bob使用不同CA颁发的证书：CAa<<Alice>>和CAb<<Bob>>，Alice只有CAa的公钥，Bob只有CAb的公钥
  - X.509验证过程
    - Alice给Bob发送自己的公钥证书CAa<<Alice>>，但是Bob没有CAa的公钥（Bob不信任CAa）
    - Bob向CAb请求CAa的公钥（向CAb询问CAa是否可信）
    - CAb将CAa的公钥证书CAb<<CAa>>发给Bob
    - Bob得到CAa后，就可以验证Alice的证书并获得Alice的公钥
    - Bob的证书链：CAb<<CAa>>CAa<<Alice>>
  - 典型应用
    - Web，email等信息传输
    - 如果不能确认证书的合法性，则会提示风险网站

CA的层次结构

- 用户知道自己CA的公钥，如果两个用户属于同一个CA，则认为这两个用户均知道该CA的公钥
- 如果两个用户不属于同一个CA,X.509推荐在CA之间使用层次结构
- 利用层次结构之间的链接证书验证其他CA的合法性
  - 前向(forward)证书和后向(backward)证书
  - 用户能够验证层次结构中的任意CA颁发的证书的可信性



CA层次结构举例



Case 1: User A get user C's certificate

Case 2: User A get user B's certificate

X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

Case 3: User B get user A's certificate

Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

证书撤销

- 证书是用来绑定身份和相应公钥的，通常这种绑定在证书的整个生命周期中都是有效的，但是有时证书没有到期，但是已颁发的证书不再有效，这时就要进行**证书撤销**。
- 证书具有有效期，证书过期前撤销的情况：
  - 用户的私钥被认为已经泄露
  - 用户不再被CA信任
  - CA的证书被认为已经泄露
- 证书撤销需要一种有效、可信的方法。

证书撤销

- CA必须存储一个包含由其发放的所有被撤销而未到期证书的列表（包含给用户或其他CA发放的证书）--证书撤销列表CRL
  - 由证书发放者签名
  - 包括：发放者的名称、列表创建日期、下一个CRL计划发放日期、每一个被撤销证书的入口
  - 撤销证书入口包括：证书序列号、证书的撤销日期

证书撤销

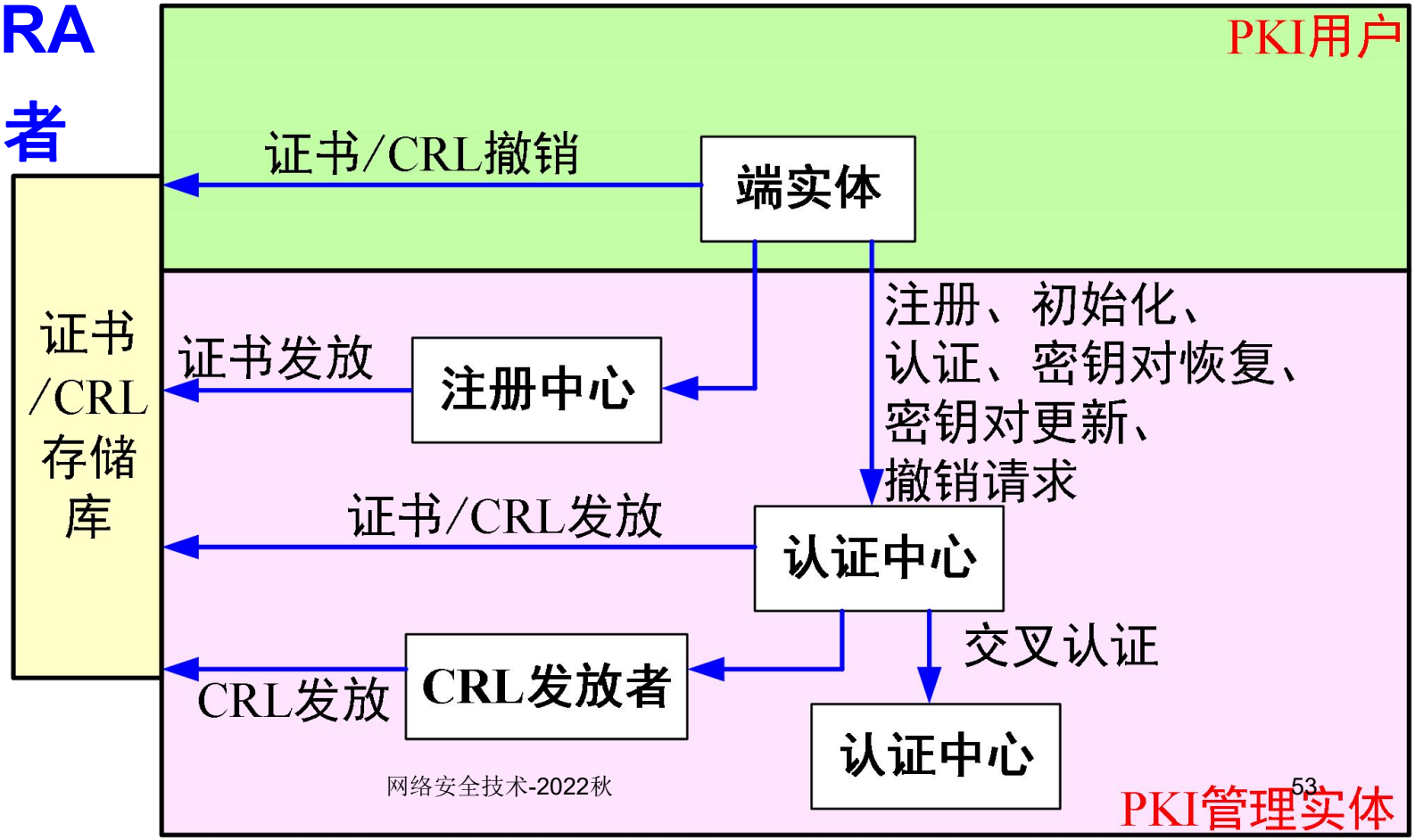
- 用户从消息中得到证书时，必须要确定证书是否被撤销
  - 检查存放在CA目录中的CRL
  - 维护一个记录证书和被撤销证书列表的本地缓存

5.5 公钥基础设施PKI

- 公钥基础设施PKI(RFC4949)是基于非对称密码体制的用来生成、管理、存储、分配和撤销数字证书的一套硬件、软件、人员、策略和过程。
- 开发PKI的主要目标：使安全、方便和高效获取公钥成为可能。
- PKIX:是一个基于X.509、适用于在互联网上部署基于证书架构的模型。

PKIX架构模型

- 端实体
- 认证中心CA：证书和CRL的发放者
- 注册中心RA
- CRL发放者
- 存储库



注册机构(RA)

- 专门实现注册的机构，目的是分担CA的一定功能以增强可扩展性，RA不允许颁发证书或证书注销列表CRL。
- 当某个PKI区域里的终端实体用户的数量增加或者终端实体在地理上分布很广泛，则需要多个RA（也叫局部注册机构或LRA）实现注册
- RA功能
  - 建立并确定个体的身份，在线初始化进程中分发共享秘密以便下一步的确认
  - 代表终端用户启动和CA的认证过程
  - 生成代表用户的密钥资料
  - 执行一定的密钥/证书生命周期管理功能

PKIX管理功能

- **注册**：让CA知道用户的存在，该过程中CA将一个或多个共享密钥（后续认证使用）发放给端实体。
- **初始化**：客户端安全工作前的准备过程。
- **认证**：CA为用户的公钥发放一个证书，并将该证书返回给用户的客户端和/或将此证书存放在一个存储库中
- **密钥对恢复**：允许端实体从CA处恢复加解密密钥对
- **密钥对更新**：更新密钥对并颁发新证书
- **撤销申请**：经过授权的用户请求CA撤销证书
- **交叉认证**：两个CA互相交换用于建立交叉证书的信息