数字签名、鉴别协议

郑燕飞

数字签名（Conf.）

- 数字签名是认证的重要工具
- 为什么需要数字签名：
  - 报文认证用以保护双方之间的数据交换不被第三方侵犯；但它并不保证双方自身的相互欺骗。假定A发送一个认证的信息给B，双方之间的争议可能有多种形式：
    - B伪造一个不同的消息，但声称是从A收到的。
    - A可以否认发过该消息，B无法证明A确实发了该消息

数字签名应满足的要求

- 收方能确认或证实发方的签字，但不能伪造；
- 发方发出签名后的消息，就不能否认所签消息；
- 收方对已收到的消息不能否认；
- 第三者可以确认收发双方之间的消息传送，但不能伪造这一过程

数字签名应具有的性质

- 必须能够验证作者及其签名的日期时间；
- 必须能够认证签名时刻的内容；
- 签名必须能够由第三方验证，以解决争议；

数字签名的设计要求

- 签名必须是依赖于被签名信息的比特模式；
- 签名必须使用某些对发送者是唯一的信息，以防止双方的伪造与否认；
- 必须相对容易生成该数字签名；
- 必须相对容易识别和验证该数字签名；
- 伪造该数字签名在计算复杂性意义上具有不可行性，既包括对一个已有的数字签名构造新的消息，也包括对一个给定消息伪造一个数字签名；
- 在存储器中保存一个数字签名备份是现实可行的。

两类数字签名

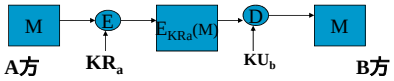
- 直接数字签名
  - 仅涉及通信双方
  - 有效性依赖发方密钥的安全性
- 仲裁数字签名
  - 使用第三方认证

直接数字签名

(1)A→B: $E_{KRa}[M]$

提供了认证与签名：

- 只有A具有KRa进行加密;
- 传输中无法被篡改；
- 需要某些格式信息/冗余度；
- 任何第三方可以用KUa 验证签名



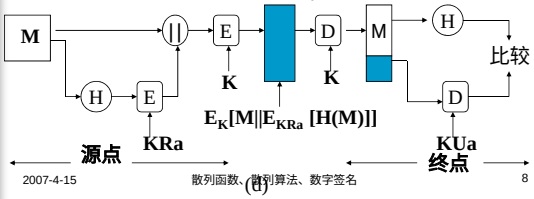
直接数字签名

(2) A→B: $E_K(M \| E_{KRa}(H(M)))$

提供认证及数字签名，保密

-- H(M) 受到密码算法的保护；

-- 只有 A 能够生成 $E_{KRa}[H(M)$



直接数字签名的缺点

- 验证模式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。
- 如X的私有密钥确实在时间T被窃取，敌方可以伪造X的签名并附上早于或等于时间T的时间戳。

仲裁数字签名

- 引入仲裁者。
  - 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指示一起发给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）

仲裁数字签名技术1

- 单密钥加密方式，仲裁者可以看见消息

(1) X→A：$M||E_{Kxa}[ID_x|| H(M)]$

(2) A→Y：$E_{Kay}[ID_x|| M || E_{Kxa}[ID_x|| H(M)] || T]$

X与A之间共享密钥K_{xa}，Y与A之间共享密钥K_{ay}

X：准备消息M，计算其散列码H(M)，用X的标识符ID_x 和散列值构成签名，并将消息及签名经K_{xa}加密后发送给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K_{ay}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。

解决纠纷：

Y：向A发送 $E_{Kay}[ID_x|| M || E_{Kxa}[ID_x|| H(M)]]$

A：用K_{ay}恢复ID_x，M，和签名（$E_{Kxa}[ID_x|| H(M)]$），然后用K_{xa}解密签名并验证散列码.

仲裁数字签名技术2

- 单密钥加密方式，仲裁者不可以看见消息

(1) X→A： ID_x || $E_{K_{xy}}[M]$||$E_{K_{xa}}[ID_x$|| $H(E_{K_{xy}}[M])]$

(2) A→Y：$E_{K_{ay}}[ID_x||E_{K_{xy}}[M]$ || $E_{K_{xa}}[ID_x||$ $H(E_{K_{xy}}[M])]$ || T]

在这种情况下，X与Y之间共享密钥K_{xy}，

X：将标识符ID_x ,密文 $E_{K_{xy}}[M]$，以及对ID_x和密文消息的散列码用K_{xa}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K_{ay}加密后发送给Y。

- (1)和(2) 存在一个共性问题：
  - A和发送方联手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；

仲裁数字签名技术3

- 双密钥加密方式，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： $E_{KR_a}[ID_x||$ $E_{KU_y}[E_{KR_x}[M]]$ || 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的公开/私有密钥对是否仍然有效，如果是，则认证消息。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

- 本模式比上述两个模式具有以下好处：

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、从X发送给Y的消息的内容对A和任何其他人是保密的。

数字签名算法

- 分类
  - 确定性 eg:RSA
  - 随机化的（概率式） eg: ElGamal, DSA
- 体制
  - 签名算法
  - 验证算法
- 安全性
  - 从M和其签名S难以推出K
  - （或）伪造一个M'使M'和S可被证实为真

RSA签名方案

- 1. 密钥的生成（同加密系统）
  公钥Pk={e,n};私钥Sk={d,n}。
- 2. 签名过程 (用d,n)
  明文：M<n　密文：$S=M^d \pmod n$.
- 3. 验证过程 (用e,n)
  给定M，S，Ver（M，S）为真，当且仅当，$M=S^e \pmod n$

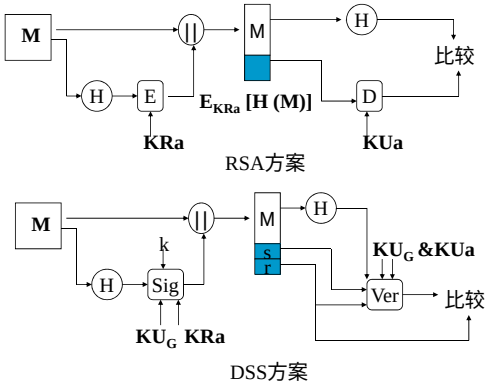
实用的RSA签名方案

- **1.** 密钥的生成（同加密系统）

公钥Pk={e,n};私钥Sk={d,n}。H为一安全散列函数。

2. 签名过程 (用d,n)
   文件：Message<n
   计算M=H(Message)
   签名：$S=M^d \pmod n$.
- 3. 验证过程 (用e,n)

给定Message，S，Ver（M，S）为真，当且仅当，
H(Message)=$S^e \pmod n$



RSA方案

DSS方案

数字签名标准

- DSS （Digital Signature Standard)签名标准是1991年8月由美国NIST公布，1994年5月19日的正式公布，并于1994年12月1日采纳为美国联邦信息处理标准。
- DSS为EIGamal和Schnorr签名方案的改进，其使用的算法记为DSA（Digital Signature Algorithm)。此算法由D. W. Kravitz设计。DSS使用了SHA，安全性是基于求离散对数的困难性。
- DSS＝SHA+DSA
- RSA签名标准=MD5(SHA) +RSA

DSA算法描述

- 全局公钥（p，q，g）－ KU_G
  – p为512～1024bit的大素数，
  – q是（q－1）的素因子，为160比特的素数，
  – g:=$h^{(p-1)/q} \bmod p$, 且1<h<(p-1),使得 $h^{(p-1)/q} \bmod p > 1$
- 用户私钥x：x为0<x<q内的随机数
- 用户公钥y：$y=g^x \bmod p$

- 用户每个消息用的秘密随机数k：0<k<q;
- 签名过程：对报文M，签名为（r，s）
  – $r = (g^k \bmod p) \bmod q$
  – $s = (k^{-1}(H(M) + xr)) \bmod q$
- 验证：
  – $w = s^{-1} \bmod q$
  – $a = (H(M)w) \bmod q$　$b = rw \bmod q$
  – $v = ((g^a y^b) \bmod p) \bmod q$
- Ver（M，r，s）＝真 iff v = r

对数字签名的攻击

- 分类
  - 不知明文密文对攻击
  - 已知明文密文对攻击
    - The plain known-msg attack
    - The generic chosen-msg attack
    - The oriented chosen-msg attack
    - The adaptively chosen-msg attack

其他签名技术简介

- 不可否认签名
  - Chaum和Van Antwerprn 1989年提出
  - 该签名的特征是：验证签名者必须与签名者合作。验证签名是通过询问------应答协议来完成。
  - 一个不可否认的签名方案有三个部分组成：

    签名算法、验证协议、否认协议
  - 不可否认的签名的本质是无签名者合作不能验证签名，从而防止复制和散布其签名文件的可能，适应于电子出版系统知识产权的保护。

- 盲签名
  - Chaum在1983年提出。
  - 需要某人对文件签名，但又不想签名者知道文件内容，称为盲签名。
  - 适应于电子选举、数字货币协议中。
  - 零知识证明
- 群签名
  - 群签名是群体密码学中的课题，1991由Chaum和van Heyst提出。
  - 特点：1 只有群体成员才能代表群体签名；2 可用公钥验证签名，但不知是谁签的名；3 争议发生时可由群体成员或可信第三方确认签名者。

认证协议

- 双方认证 (mutual authentication)
- 单向认证 (one-way authentication)

双边认证协议

- 最常用的协议，该协议使得通信各方互相认证鉴别各自的身份，然后交换会话密钥。
- 基于认证的密钥交换核心问题有两个：
  - 保密性
  - 时效性

- 消息重放：最坏情况下可能导致暴露会话密钥，或成功地冒充其他人；至少也可以干扰系统的正常运行，处理不好将导致系统瘫痪。
- 对付重放攻击的一种方法是在认证交换中使用一个序数来给每一个消息报文编号。仅当收到的消息序数顺序合法时才接受之。但这种方法的困难是要求双方必须保持上次消息的序号。

两种更为一般的方法是：
1、**时间戳**：A接受一个新消息仅当该消息包含一个时间戳，该时间戳在A看来，是足够接近A所知道的当前时间；这种方法要求不同参与者之间的时钟需要同步。

时间戳方法似乎不能用于面向连接的应用，因为该技术固有的困难
(1) 某些协议需要在各种处理器时钟中维持同步。该协议必须既要容错以对付网络出错，又要安全以对付重放攻击。
(2) 由于某一方的时钟机制故障可能导致临时失去同步，这将增大攻击成功的机会。
(3) 由于变化的和不可预见的网络延迟的本性，不能期望分布式时钟保持精确的同步。

2、盘问/应答方式。（Challenge/Response）A期望从B获得一个新消息　，首先发给B一个临时值(challenge)，并要求后续从B收到的消息（response）包含正确的这个临时值。

盘问/应答方法不适应非连接性的应用，因为它要求在传输开始之前先有握手的额外开销，这就抵消了无连接通信的主要特点。

传统加密方法　　Needham/Schroeder Protocol [1978]

1、$A \rightarrow KDC：ID_A||ID_B||N_1$
2、$KDC \rightarrow A：E_{K_a}[K_s||ID_B||N_1||E_{K_b}[K_s||ID_A]]$
3、$A \rightarrow B：E_{K_b}[K_s||ID_A]$
4、$B \rightarrow A：E_{K_s}[N_2]$
5、$A \rightarrow B：E_{K_s}[f(N_2)]$

保密密钥K_a和K_b分别是A和KDC、B和KDC之间共享的密钥。
本协议的目的就是要安全地分发一个会话密钥K_s给A和B。

A在第2步安全地得到了一个新的会话密钥，第3步只能由B解密、并理解。第4步表明B已知道K_s了。第5步表明B相信A知道K_s并且消息不是伪造的。
第4，5步目的是为了防止某种类型的重放攻击。特别是，如果敌方能够在第3步捕获该消息，并重放之，这将在某种程度上干扰破坏B方的运行操作。

上述方法尽管有第4,5步的握手，但仍然有漏洞。

假定攻击方C已经掌握A和B之间通信的一个老的会话密钥。C可以在第3步冒充A利用老的会话密钥欺骗B。除非B记住所有以前使用的与A通信的会话密钥，否则B无法判断这是一个重放攻击。如果C可以中途阻止第4步的握手信息，则可以冒充A在第5步响应。从这一点起，C就可以向B发送伪造的消息而对B来说认为是用认证的会话密钥与A进行的正常通信。

Denning Protocol [1982] 改进：

1、$A \rightarrow KDC：ID_A||ID_B$
2、$KDC \rightarrow A：E_{K_a}[K_s||ID_B||T||E_{K_b}[K_s||ID_A||T]]$
3、$A \rightarrow B：E_{K_b}[K_s||ID_A||T]$
4、$B \rightarrow A：E_{K_s}[N_1]$
5、$A \rightarrow B：E_{K_s}[f(N_1)]$

如何验证T的时效性？
Clock - T | < $\Delta t_1 + \Delta t_2$
其中：Δt_1 是KDC时钟与本地时钟（A或B）之间差异的估计值；
Δt_2 是预期的网络延迟时间。

Denning Protocol 比 Needham/Schroeder Protocol在安全性方面增强了一步。然而，又提出新的问题：即必须依靠各时钟均可通过网络同步。

如果发送者的时钟比接收者的时钟要快，攻击者就可以从发送者窃听消息，并在以后当时间戳对接收者来说成为当前时重放给接收者。这种重放将会得到意想不到的后果。（称为抑制重放攻击）。

一种克服抑制重放攻击的方法是强制各方定期检查自己的时钟是否与KDC的时钟同步。

避免时钟同步需求的协议 （KEHN92）

1、A → B： $ID_A||N_a$
2、B → KDC： $ID_B||N_b$ || $E_{K_b}[ID_A || N_a || T_b]$ – 信任状
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： $E_{K_b}[ID_A || K_s || T_b]$ || $E_{K_s}[N_b]$

公钥加密方法：

一个使用时间戳的方法是：

1、A → AS：$ID_A||ID_B$
2、AS → A：$E_{KR_{as}}[ID_A ||KU_a || T]$ || $E_{KR_{as}}[ID_B ||KU_b || T]$
3、A → B： $E_{KR_{as}}[ID_A||KU_a|| T]$ || $E_{KR_{as}}[ID_B || KU_b || T]$ ||
$E_{KU_b}[E_{KR_a}[K_s||T]]$

一个基于现时握手协议：WOO92a

1、A → KDC：$ID_A||ID_B$
2、KDC → A：$E_{KR_{auth}}[ID_B ||KU_b]$
3、A → B： $E_{KU_b}[N_a ||ID_A]$
4、B → KDC： $ID_B||ID_A$ || $E_{KU_{auth}}[N_a]$
5、KDC → B： $E_{KR_{auth}}[ID_A ||KU_a]||$ $E_{KU_b}[E_{KR_{auth}}[N_a||K_s||ID_B]]$
6、B → A： $E_{KU_a}[E_{KR_{auth}}[N_a ||K_s || ID_B]||N_b]$
7、A → B： $E_{K_s}[N_b]$

一个基于现时握手协议的改进：WOO92b

1、A → KDC：$ID_A||ID_B$
2、KDC → A：$E_{KR_{auth}}[ID_B ||KU_b]$
3、A → B： $E_{KU_b}[N_a ||ID_A]$
4、B → KDC： $ID_B||ID_A$ || $E_{KU_{auth}}[N_a]$
5、KDC → B： $E_{KR_{auth}}[ID_A ||KU_a]||$ $E_{KU_b}[E_{KR_{auth}}$
$[N_a||K_s||ID_A||ID_B]]$
6、B → A： $E_{KU_a}[E_{KR_{auth}}[N_a ||K_s ||ID_A || ID_B]||N_b]$
7、A → B： $E_{K_s}[N_b]$

One-Way Authentication

- E-mail
  - 收发方无需同时在线
  - Email确实是宣称的发方发送的

传统加密方法：

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：

1、A → B： $E_{KU_b}[K_s]|| E_{K_s}[M]$

1、A → B： $M || E_{KR_a}[H(M)]$

1、A → B： $E_{KU_b}[M || E_{KR_a}[H(M)]]$

1、A → B： $M || E_{KR_a}[H(M)] || E_{KR_{as}}[T || ID_A ||KU_a]$


END
2007-4-15
散列函数、散列算法、数字签名
43