数字签名
(Digital Signature)

哈尔滨工业大学

张宇

2024春

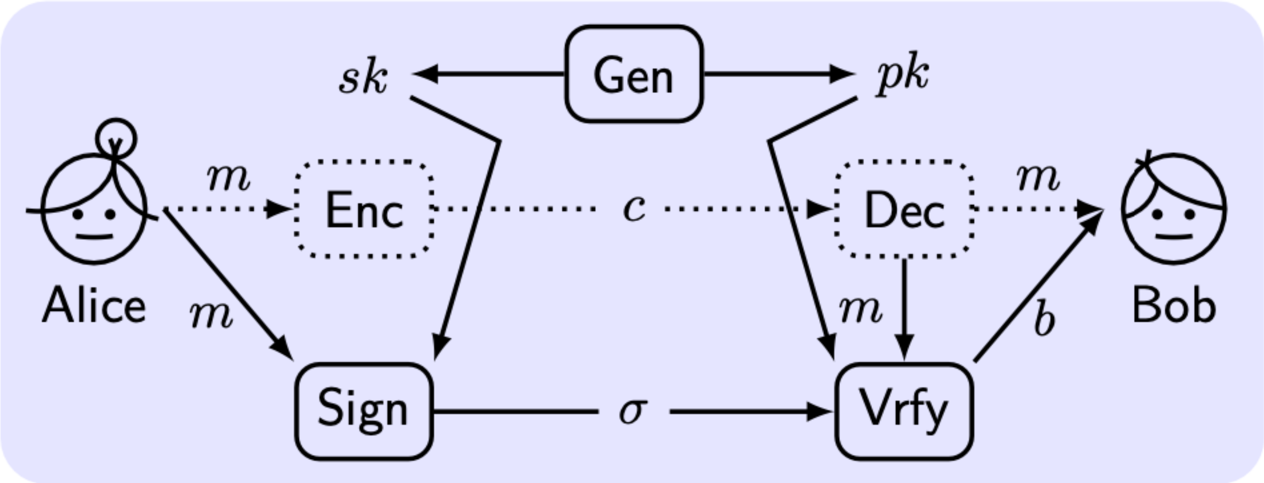
概览

1. 数字签名定义
2. RSA签名
3. 来自离散对数问题的数字签名
4. 一次签名方案/理论构造安全签名方案
5. 证书与公钥基础设施

数字签名概念

- 数字签名（Digital signature）是一个用来证明一个数字消息的真实性/完整性的数学方案。
- 数字签名允许一个签名者（Signer）S 用其自己的私钥来“签名”（sign）一个消息，并且任何知道 S 的公钥的人可以验证（verify）其真实性/完整性。
- 与MAC相比，数字签名是：
  - 公开可验证的（publicily verifiable）；
  - 可转移的（transferable）；
  - 不可抵赖（non-repudiation）；
  - 但速度慢。
- 问题：数字签名和手写签名的区别是什么？
- 数字签名**不是**公钥加密的逆。

数字签名词法



- signature σ, a bit b means valid if $b = 1$; invalid if $b = 0$.
- **Key-generation** algorithm $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$, $|pk|, |sk| \geq n$.
- **Signing** algorithm $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.
- **Verification** algorithm $b := \mathsf{Vrfy}_{pk}(m, \sigma)$.
- **Basic correctness requirement**: $\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m)) = 1$.

定义签名安全

- 安全数字签名与安全MAC类似，敌手难以伪造一个“新消息”的签名。

The signature experiment $\mathsf{Sigforge}_{\mathcal{A},\Pi}(n)$:

1. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given input 1^n and oracle access to $\mathsf{Sign}_{sk}(\cdot)$, and outputs (m, σ). $\mathcal{Q}$ is the set of queries to its oracle.
3. $\mathsf{Sigforge}_{\mathcal{A},\Pi}(n) = 1 \iff \mathsf{Vrfy}_{pk}(m, \sigma) = 1 \land m \notin \mathcal{Q}$.

Definition 1

A signature scheme Π is **existentially unforgeable under an adaptive CMA** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that:

$$\Pr[\mathsf{Sigforge}_{\mathcal{A},\Pi}(n) = 1] \le \mathsf{negl}(n).$$

- 问题：在MAC和数字签名中敌手能力的差别是什么？如果敌手不限制算力为PPT会如何？

“书本上RSA”的不安全性

- 构造：
  - Gen: on input 1^n run GenRSA(1^n) to obtain N, e, d. $pk = \langle N, e \rangle$ and $sk = \langle N, d \rangle$.
  - Sign: on input sk and $m \in \mathbb{Z}_N^*$, $\sigma := [m^d \bmod N]$.
  - Vrfy: on input pk and $m \in \mathbb{Z}_N^*$, $m \stackrel{?}{=} [\sigma^e \bmod N]$.
- 无消息攻击（no-message attack）：
  - 选择一个任意 $\sigma \in \mathbb{Z}_N^*$ 并且计算 $m := [\sigma^e \bmod N]$。输出伪造签名 (m, σ)。
  - 例子：$pk = \langle 15, 3 \rangle$, $\sigma = 2$, $m =?$ $m^d =?$
- 任意消息攻击（Forging a signature on an arbitrary message）：为了伪造 m 的签名，选择一个随机的 m_1，令 $m_2 := [m/m_1 \bmod N]$，查询预言机获得消息 m_1, m_2 的签名 σ_1, σ_2 。
  - 问题：$\sigma := [\_\_\_\_ \bmod N]$ 是 m 的一个有效签名。

哈希RSA签名

- 思路：用哈希函数来打破消息和签名之间的的强代数关系
- RSA-FDH 签名方案：随机预言机作为一个全域哈希（Full Domain Hash，FDH)），其定义域大小为 RSA 的模数 N-1（PKCS \#1 v2.1）
- 目前实际使用哈希RSA数字签名方案：
  - Gen: a hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$ is part of public key.
  - Sign: $\sigma := [H(m)^d \bmod N]$.
  - Vrfy: $\sigma^e \stackrel{?}{=} H(m) \bmod N$.
- 如果 H 无法有效求逆，那么无消息攻击和伪造任意消息签名都是难的。
  - 无消息攻击：敌手无法求逆
  - 任意消息攻击：σ_2 与σ没有关系
- 不安全性：没有已知函数 H被证明使得哈希RSA签名是安全的。

DSS/DSA

- NIST从1994年到2013年颁布的数字签名标准（Digital Signature Standard，DSS) 使用数字签名算法（Digital Signature Algorithm，DSA），该算法是一个ElGamal签名方案的变体。DSS中还包括椭圆曲线数字签名算法（Elliptic Curve Digital Signature Algorithm，ECDSA）和 RSA签名算法。
- 这两种算法基于相同的算法抽象：基于身份认证方案的签名方案。
- 构造：
  - Gen: $(\mathbb{G}, q, g) \leftarrow \mathcal{G}$. 两个哈希函数 $H, F : \{0,1\}^* \to \mathbb{Z}_q$.
    - $x \leftarrow \mathbb{Z}_q$ 和 $y := g^x$.
    - $pk = \langle \mathbb{G}, q, g, y, H, F \rangle$. $sk = \langle \mathbb{G}, q, g, x, H, F \rangle$.
  - Sign: $k \leftarrow \mathbb{Z}_q^*$ 并且 $r := F(g^k)$, $s := (H(m) + xr) \cdot k^{-1}$. 输出 (r, s).
  - Vrfy: 输出 1 $\iff r \stackrel{?}{=} F(g^{H(m)\cdot s^{-1}} y^{r \cdot s^{-1}})$.
- DSA中验证的正确性?

DSS/DSA安全性

- 不安全性：DSS的安全性依赖于离散对数问题的难解性，尚未有基于离散对数假设的DSS安全性证明。
- k 的熵、保密和唯一性是安全性的关键。
- 情况1：如果k是可预测的，那么x将泄漏，因为$s := (H(m) + xr) \cdot k^{-1}$中只有$x$是未知的。
- 情况2：如果同一个k被用于同一私钥下的两个不同签名，那么k和x都将泄漏。问题：如何做?
  - 该攻击曾在2010年用于对Sony PlayStation (PS3) 提取私钥。

一次签名

- 下面学习不基于数论假设，而是基于哈希函数来构造数字签名方案。
- 一次签名（One-Time Signature，OTS）：在一种较弱的攻击场景下，一个秘密只用于一个消息签名。
- 模拟一次签名场景，敌手最多只允许查询一次签名预言机，之后需要给出新消息和签名。

One-Time Signature (OTS): Under a weaker attack scenario, sign only one message with one secret.

The OTS experiment $\mathsf{Sigforge}^{\text{1-time}}_{\mathcal{A},\Pi}(n)$:

1. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given input 1^n and a single query m' to $\mathsf{Sign}_{sk}(\cdot)$, and outputs (m, σ), $m \neq m'$.
3. $\mathsf{Sigforge}^{\text{1-time}}_{\mathcal{A},\Pi}(n) = 1 \iff \mathsf{Vrfy}_{pk}(m, \sigma) = 1$.

Definition 5

A signature scheme Π is **existentially unforgeable under a single-message attack** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that:

$$\Pr[\mathsf{Sigforge}^{\text{1-time}}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

Lamport的OTS (1979)

- 思路：从单向函数构造OTS；每个比特为一个映射

Construction 6

f is a one-way function.

- Gen: *on input* 1^n, *for* $i \in \{1, \ldots, \ell\}$:
  1. *choose random* $x_{i,0}, x_{i,1} \leftarrow \{0,1\}^n$.
  2. *compute* $y_{i,0} := f(x_{i,0})$ *and* $y_{i,1} := f(x_{i,1})$.

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix} \quad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}.$$

- Sign: $m = m_1 \cdots m_\ell$, *output* $\sigma = (x_{1,m_1}, \ldots, x_{\ell,m_\ell})$.
- Vrfy: $\sigma = (x_1, \ldots, x_\ell)$, *output* $1 \iff f(x_i) = y_{i,m_i}$, *for all* i.

Theorem 7

If f is OWF, Π is OTS for messages of length polynomial ℓ.

Lamport的OST例子

Signing $m = 011$

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix} \Longrightarrow \sigma = \underline{\qquad}$$

$\sigma = (x_1, x_2, x_3)$:

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{pmatrix} \Longrightarrow \begin{array}{l} f(x_1) \stackrel{?}{=} \underline{\qquad} \\ f(x_2) \stackrel{?}{=} \underline{\qquad} \\ f(x_3) \stackrel{?}{=} \underline{\qquad} \end{array}$$

Lamport的OST安全证明

- 思路：如果 $m \neq m'$，那么 $\exists i^*, m_{i*} = b^* \neq m'_{i*}$。因此，为了伪造一个消息至少要对一个 y_{i^*,b^*}求逆。
- 证明：将对y求逆的 $\mathcal{I}$ 算法规约到攻击 Π的 $\mathcal{A}$ 算法：
  - $\mathcal{I}$算法构造 pk：选择 $i^* \leftarrow \{1,\ldots,\ell\}$ 并且 $b^* \leftarrow \{0,1\}$，令 $y_{i^*,b^*} := y$。对于 $i \neq i^*$ $y_{i,b} := f(x_{i,b})$；
    - 在公钥中随机选择一个位置(i^*, b^*)，将待求逆的y放在该位置；对于其它位置，正常构造公私钥对。
  - $\mathcal{A}$算法查询 m'：如果 $m'_{i_*} = b^*$，则停止。否则，返回 $\sigma = (x_{1,m'_1}, \ldots, x_{\ell,m'_\ell})$；
    - 如果$\mathcal{A}$ 的查询正好落在位置(i^*, b^*)，而该位置的x_{i^*,b^*}本应该是y对应的x，是未知的，终止实验。否则，正常返回签名。
  - 当 $\mathcal{A}$ 输出 (m, σ)，$\sigma = (x_1, \ldots, x_\ell)$，如果 $\mathcal{A}$ 在 (i^*, b^*)输出了一个伪造的值，并且有 $\mathsf{Vrfy}_{pk}(m,\sigma) = 1$ 且 $m_{i^*} = b^* \neq m'_{i^*}$，那么输出 x_{i^*,b^*}；
    - 通过验证并且在y对应位置上输出签名，说明 $\mathcal{A}$ 输出的签名满足 $f(x_{i,m_i}) = y_{i,m_i}$ 。
  - $\Pr[\mathcal{I} \text{ succeeds}] \geq \frac{1}{2\ell} \Pr[\mathcal{A} \text{ succeeds}]$
    - 这是因为位置正好在特定位置满足条件的概率是 $\frac{1}{2\ell}$.

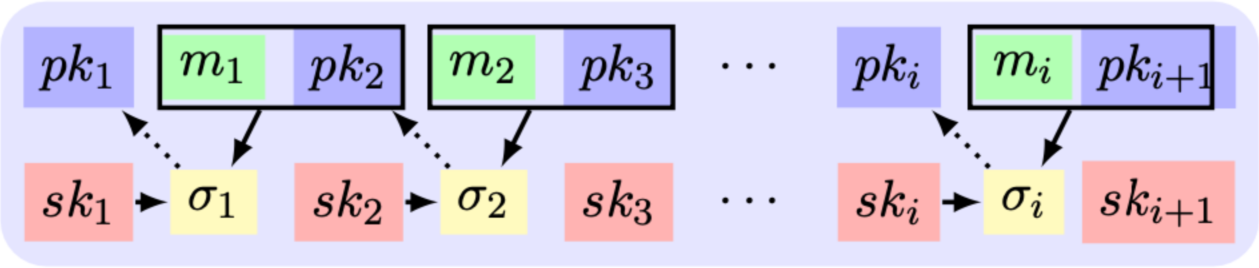
有状态签名

- 思路：为了对任意数量的消息签名，可以从旧状态中获得新的密钥并以此来实现和OTS一样的效果
- 定义：有状态签名方案（Stateful signature scheme）
  - 密钥生成算法：$(pk, sk, s_0) \leftarrow \mathsf{Gen}(1^n)$。$s_0$ 是初始状态。
  - 签名算法：$(\sigma, s_i) \leftarrow \mathsf{Sign}_{sk, s_{i-1}}(m)$。
  - 验证算法：$b := \mathsf{Vrfy}_{pk}(m, \sigma)$.
- 一个简单的有状态OTS签名方案：独立产生 (pk_i, sk_i)，令 $pk := (pk_1, \ldots, pk_\ell)$ 并且 $sk := (sk_1, \ldots, sk_\ell)$。从状态 1 开始，用 sk_s 签第 s 个消息，用 pk_s 来验证，并且更新状态到 $s+1$。
  - 安全性：每个密钥只签了一个消息。
  - 弱点：消息数量上届 ℓ 必须事先确定。

链式签名

- 思路：按需随时产生密钥并且对密钥链签名，解决消息数量有上限的问题。

Idea: generate keys "on-the-fly" and sign the key chain.



Use a single public key pk_1, sign each m_i and pk_{i+1} with sk_i:

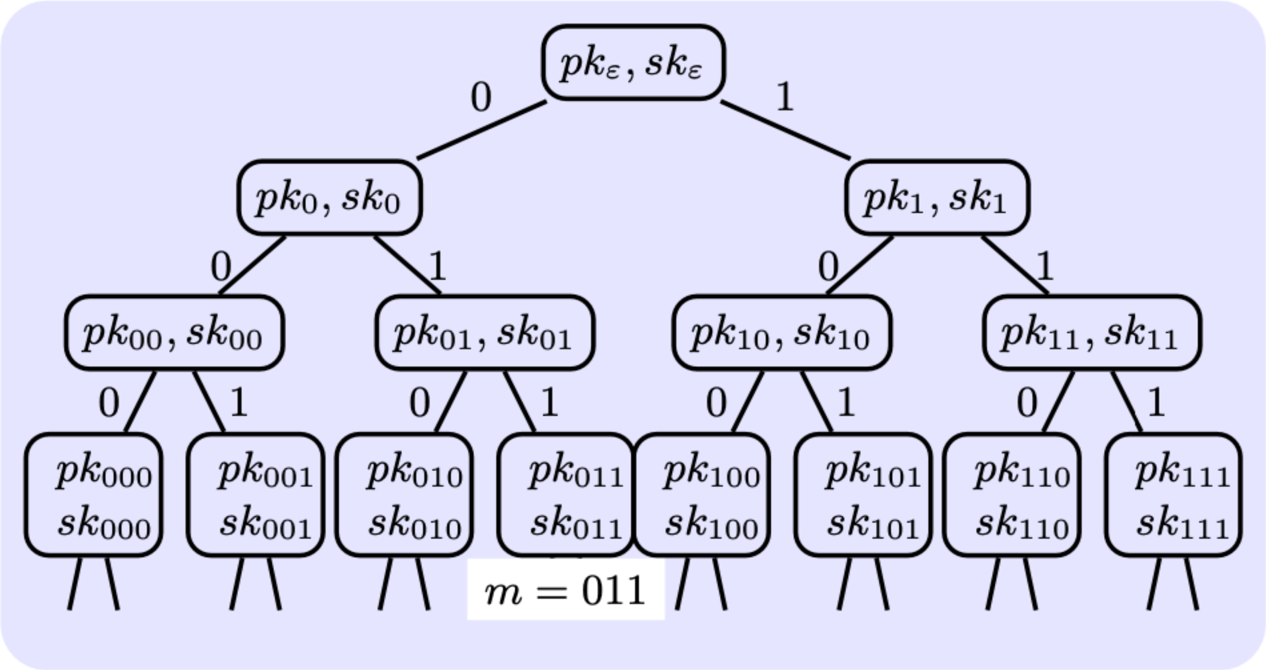
$$\sigma_i \leftarrow \mathsf{Sign}_{sk_i}(m_i \| pk_{i+1}),$$

output $\langle pk_{i+1}, \sigma_i \rangle$, and verify σ_i with pk_i.
The signature is $(pk_{i+1}, \sigma_i, \{m_j, pk_{j+1}, \sigma_j\}_{j=1}^{i-1})$.
Weakness: stateful, not efficient, revealing all previous messages.

树式签名

- 思路：减少所需维护状态，构造一个密钥树，树上的一个分支是为每个消息生成一个密钥链并且对这个链签名。

Idea: generate a chain of keys for each message and sign the chain.



- root is ε (empty string), leaf is a message m, and internal nodes (pk_w, sk_w), where w is the prefix of m.
- each node pk_w "certifies" its children $pk_{w0} \| pk_{w1}$ or w.

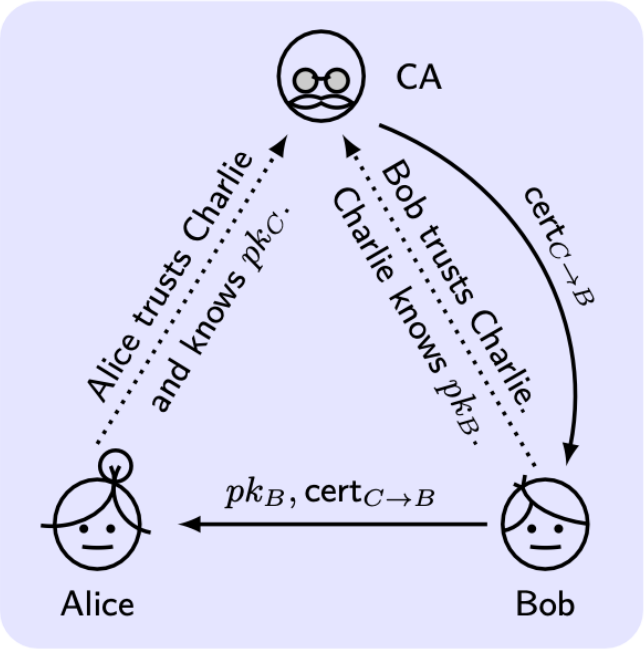
无状态签名

- 思路：用确定的随机性来模拟树的状态

- 使用PRF F 和两个密钥 k, k' 来产生 pk_w, sk_w：
  1. 计算 $r_w := F_k(w)$。
  2. 计算 $(pk_w, sk_w) := \mathsf{Gen}(1^n; r_w)$, 用 r_w 作为随机硬币。
  3. k' 用于产生 r'_w，后者在产生签名 σ_w 时使用。
- 如果 OWF 存在，那么 ∃ OTS (对于任意长度消息)。
- 定理：如果 OWF 存在，那么 ∃ (无状态) 安全签名方案。

证书（certifcate）

- 对一个公钥的数字签名被称为，数字证书（Certificate）；签发数字证书的机构被称为，证书权威机构（Certificate Authority，CA），CA是一个可信第三方，其公钥（pk_C）被所有相信CA的主体所持有。
- CA 用其私钥（sk_C）给一个主体 Bob 签发的数字证书，其中消息内容包括：主体的身份（Bob）和该主体所持的公钥（pk_B）。其本质是**绑定一个身份和一个公钥**



Certificates $\text{cert}_{C\to B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{'Bob's key is } pk_B\text{'})$.

证书（续）

- CA 的公钥是如何分发的？通常随应用程序一起分发，例如浏览器中内置了由“CA与浏览器论坛”（CAB）组织确定的约170个左右的CA的公钥。在DNSSEC中，递归服务器软件中内置了DNS根的公钥。

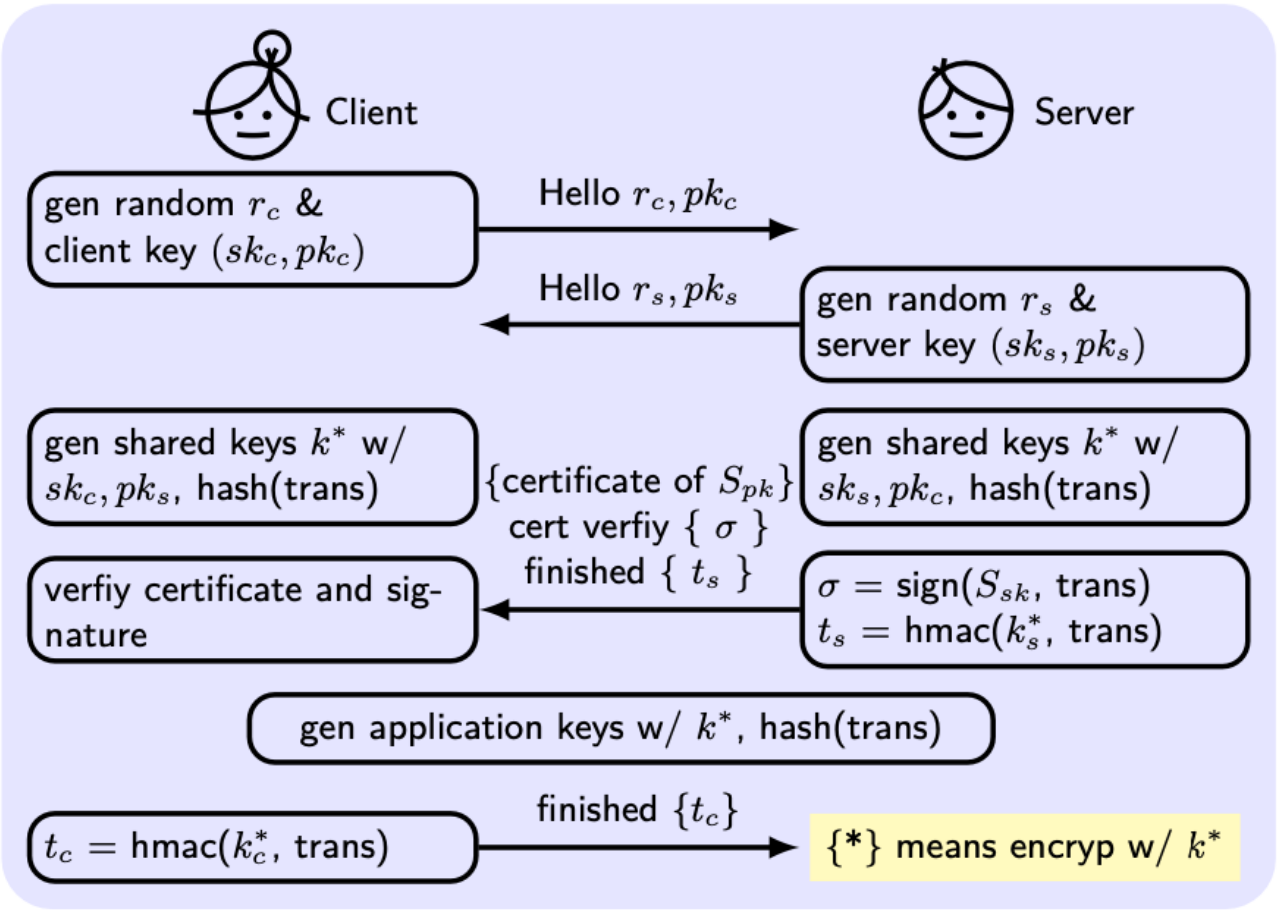
- CA如何知道收到的公钥是否是Bob的？需要采用其它渠道，例如一个CA “Let's Encrypt” 通过证明域名的所有权来识别证书申请者的身份。

PKI（公钥基础设施）

- PKI（Public-Key Infrastructure），一个分发公钥的分布式系统
- 单一 CA: 被所有人信任。
  - 优点：简单
  - 缺点：单点失效
- 多重CA：被所有人信任。
  - 优点：鲁棒
  - 缺点：水桶定律
- 授权与证书链（Delegation，certificate chains）：信任可以被传递。
  - 优点：减轻根 CA 的负担
  - 缺点：难以管理，水桶定律
- 信任网（Web of trust）：没有信任的中心，例如，PGP。
  - 优点：可靠，草根级
  - 缺点：难以管理，难以对信任作出保证

证书应用：TLS1.3握手

- 目的：客户端与认证的服务器之间协商对称密钥用于保密通信
- 要求：客户端具有可信第三方CA的公钥，服务器具有由可信第三方CA发布的服务器公钥证书



无效化证书

- 当私钥泄漏发生时，需要更换公私钥对，并将之前旧的公钥证书无效化。
- 过期法（Expiration）：在证书中包含一个过期时间，待过期后自动作废。

$$\mathsf{cert}_{C\to B} \stackrel{\text{def}}{=} \mathsf{Sign}_{sk_C}(\text{‘bob’s key is } pk_B\text{’},\ \text{date}).$$

- 撤销/召回法（Revocation）：显式地撤销证书。

$$\mathsf{cert}_{C\to B} \stackrel{\text{def}}{=} \mathsf{Sign}_{sk_C}(\text{‘bob’s key is } pk_B\text{’},\ \#\#\#).$$

其中的 ### 表示证书的序列号。

累积撤销：CA 产生证书撤销列表（Certificate revocation list，CRL）包含所有被撤销证书的序列号，并且带着当前日期一起签名。

独占所有权（Exclusive Ownership）

- 独占所有权：给定任意公钥的签名，没有敌手能够使得该签名可以被另一个不同的公钥验证。
- 重复签名公钥选择攻击：
  - 一个签名由Bob的公钥验证有效，是否意味着Bob用其私钥产生了该签名?
  - 不能。例如，Bob的用密钥对$(e=1, d=1)$和$N=\sigma-m$。可以通过任意消息的“书本上RSA签名”的验证，$\sigma^e \mod N = \sigma \mod (\sigma-m) = m$。
  - 该攻击曾被用来在域名的所有权上欺骗Let's Encrypt系统，以骗取对一个域名所有权的认证。
  - 防御：在验证之前检查公钥。

签名加密（Signcryption）

- 一群人相互直接通信，每个人生成两对密钥：(ek, dk)表示加密公钥和解密私钥；(vk, sk)表示验证公钥和签名私钥。大家知道彼此的两个公钥。当一个发送者S向接收者R发送一个消息m时，如何在CCA攻击下同时保证通信的机密性（其他人不能知道消息m）和完整性（接受者R确信消息来自发送者S）？
- 提示：下面的问题的关键在于"完整性"，即是否能能够伪装为其他人发送消息。
- "先加密后认证"：消息 $\langle S, c \leftarrow \mathsf{Enc}_{ek_R}(m), \mathsf{Sign}_{sk_S}(c)\rangle$ 是否安全?
  - 注：消息中包含发送者身份是必要的，因为接收者需要用发送者的验证公钥来验证签名；消息中不包含接收者身份，因为接收者默认收到的消息都是发给自己的（直接通信，不存在中转）。
  - 完整性有问题，发送者被伪造。因为敌手A可以将原签名替换成自己的身份和签名，$\langle A, c \leftarrow \mathsf{Enc}_{ek_R}(m), \mathsf{Sign}_{sk_A}(c)\rangle$，欺骗接收者将$A$当作消息的发送者。

签名加密（Signcryption）（续）

- “先认证再加密”：先签名，$\sigma \leftarrow \mathsf{Sign}_{sk_S}(m)$，然后发送消息 $\langle S, \mathsf{Enc}_{ek_R}(m\|\sigma)\rangle$ 是否安全?
  - 完整性有问题，发送者被伪造。因为敌手A可以将原来发送给自己的消息解密后重新用另一个人R'的加密公钥加密（不改变签名部分）后发送给R'，使得R'误认为是S给他发的消息。
- 正确的方法是将身份应作为消息的一部分：签名中包含接收者身份 $\sigma \leftarrow \mathsf{Sign}_{sk_S}(m\|R)$；加密消息中包含发送者身份 $\langle S, \mathsf{Enc}_{ek_R}(S\|m\|\sigma)\rangle$。接收者解密后，提取发送者身份和接受者身份并验证。
  - 这里的关键之一是签名将消息，发送者，接收者绑定在一起
- 当将身份和消息一起加密时，先加密后认证的方法也可以保证安全。

本节小结

- 数字签名提供了公开可验证的真实性和完整性
- 签名安全与只有某人知道的某事有关，这件事是可以公开验证的
- 签名用来将对一个公钥的信任转化为对其签名数据的信任