数字货币和区块链 - 密码学（2）

山东大学网络空间安全学院

钱宸 2023年11月1日

密码学温故知新

- 哈希函数
- 零知识证明
- 签名
- 盲签名
- 加密算法

签名算法

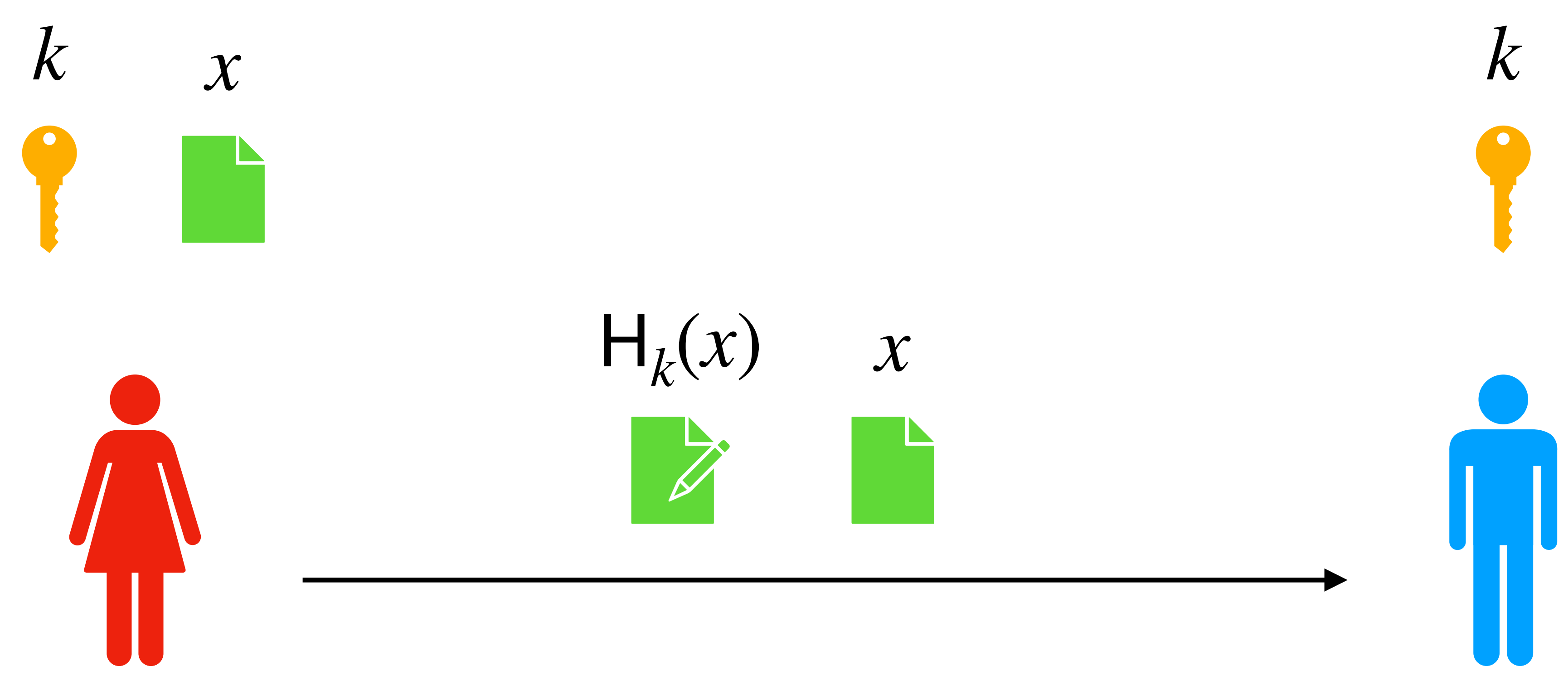
- 什么是签名?
- 一次性签名
- 多次签名
- 签名的多种构造方式与证明

签名算法

- 什么是签名?
  - 文件内容完整性
  - 文件归属权
  - 不可抵赖，伪造

签名算法

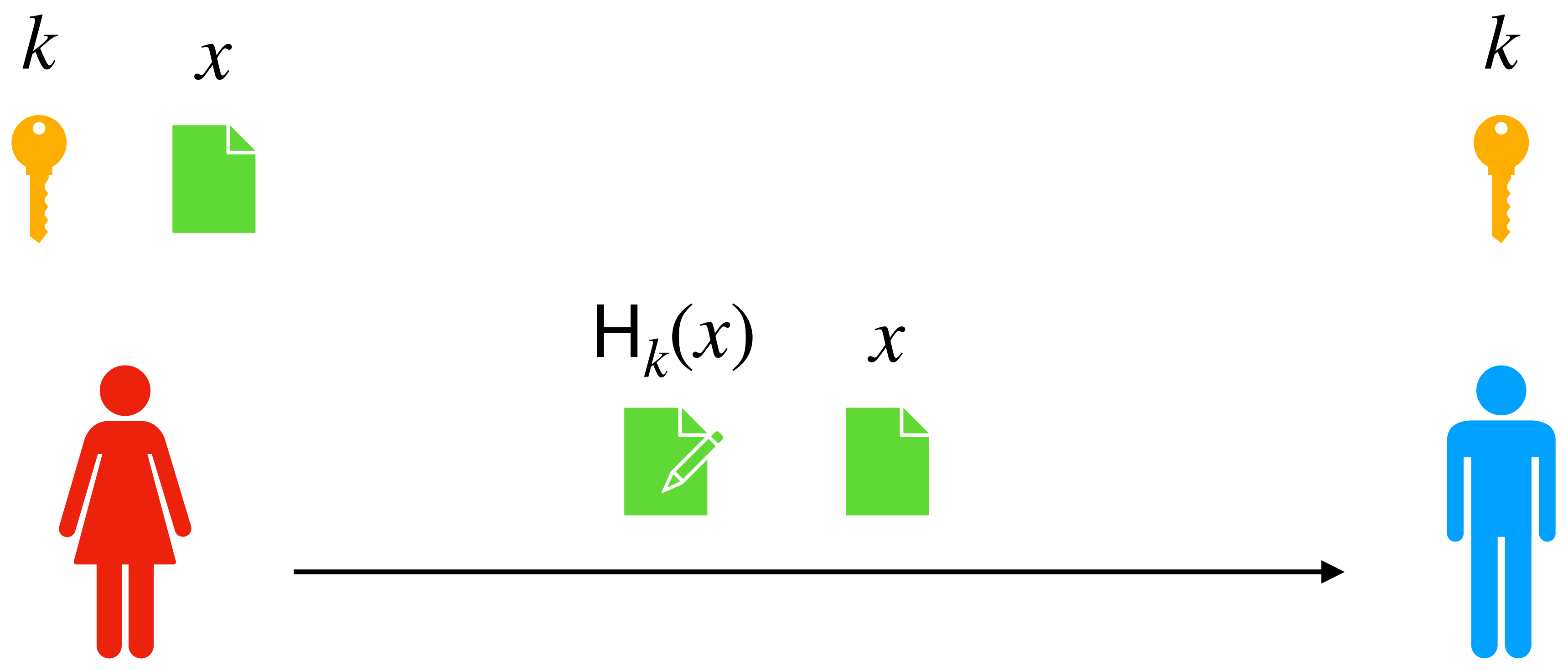
- 最简单的想法：



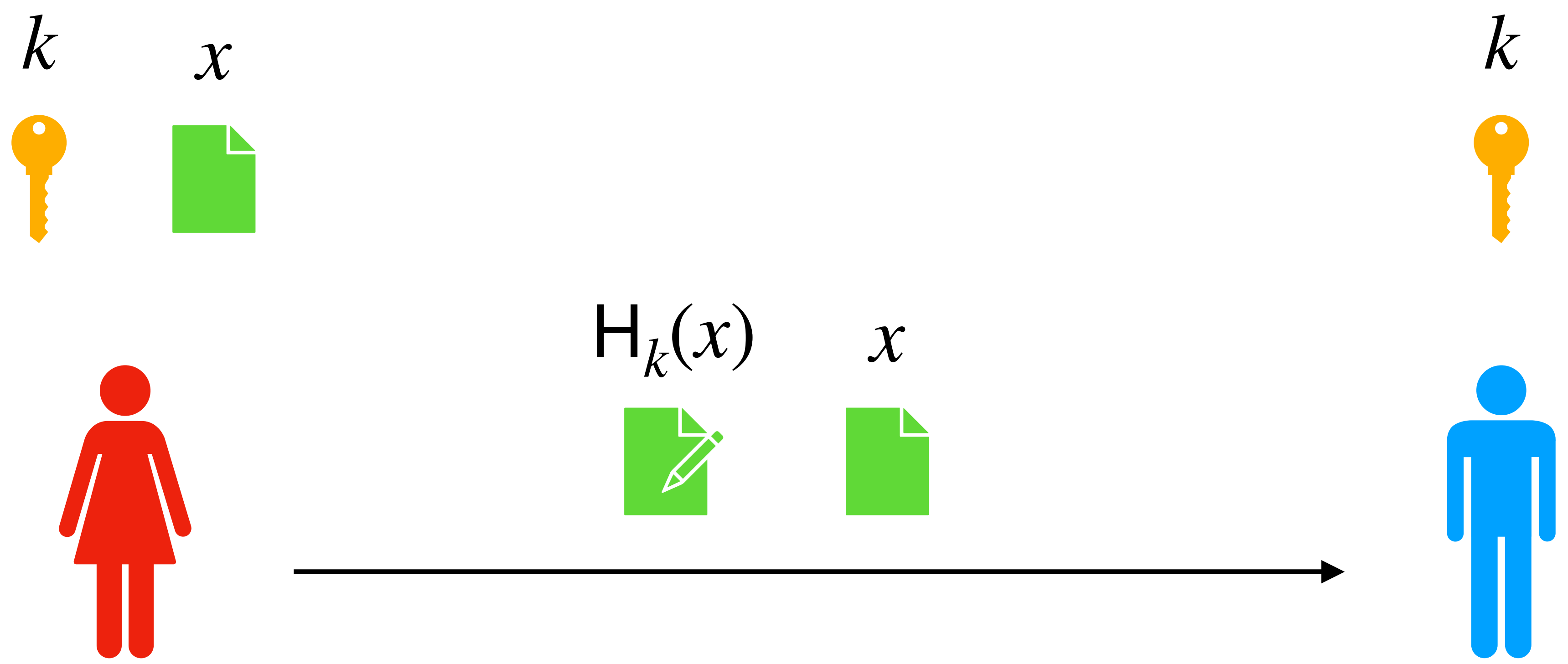
- 利用哈希函数保证文件的完整性。

签名算法

- 简单运用哈希函数具有以下的优点和缺点：
  - 优点：
    - 验证成功→则文件的完整性没有问题
  - 缺点：
    - 可以被轻易被Bob轻易复制



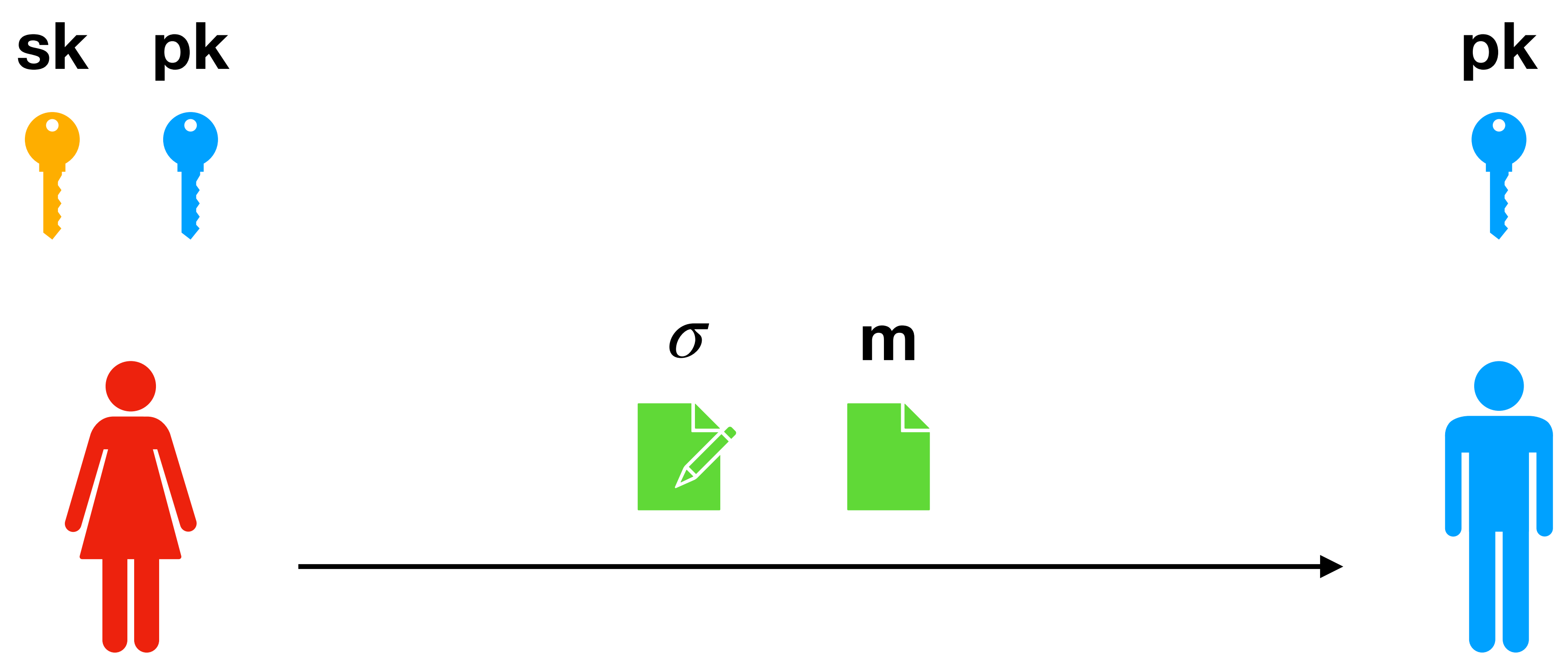
签名算法



- 对上述过程进行一些哲学思考：
  - 首先：数字世界中我是谁?
    - 每个人的秘密定义了个人的身份：私钥k
  - 签名：向他人证明文件的真实性和归属性。
    - 签名者唯一，验证签名者可以是任何人：需要签名者私钥参与，不需要验证者私钥。

签名算法

- 签名算法包含以下算法：
  - $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$：生成公私钥
  - $\mathsf{Sig}(\mathsf{pk}, \mathsf{sk}, \mathsf{m}) \to \sigma$：生成签名
  - $\mathsf{Verif}(\mathsf{pk}, \mathsf{m}, \sigma) \to \{0, 1\}$：验证签名



签名算法 - 历史

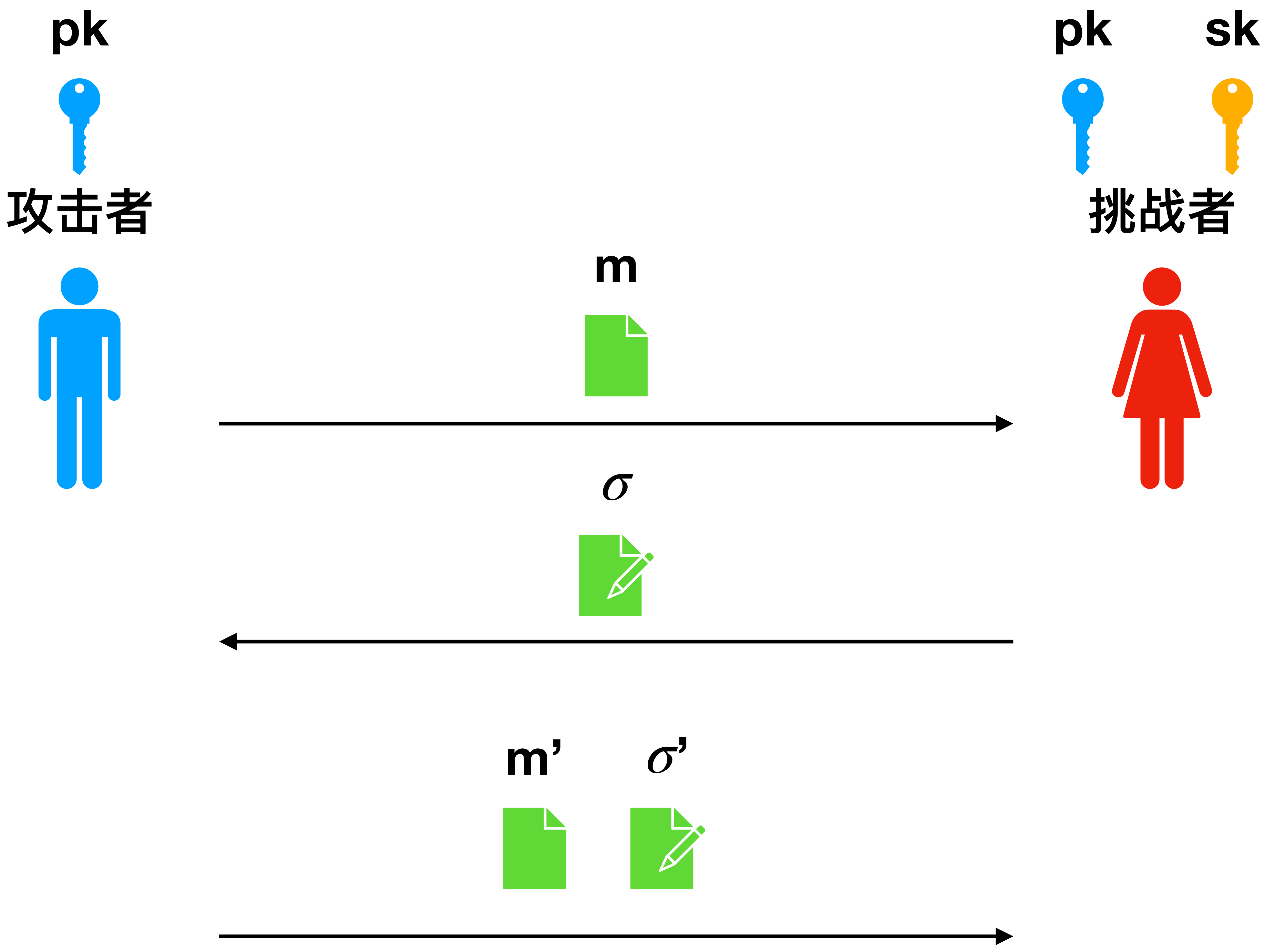
- 1978年，Whitfield Diffie和Martin Hellman在公钥密码体系的重要奠基论文：
  - “New direction in Cryptography”（2015年图灵奖）中首次提出基于单向函数，数字签名应该存在。
- 关于单向函数的简单说明：
  - 单向函数：$f(x) \overset{?}{\rightarrow} x$ 计算性角度上很困难。（思考：参考上节课中的内容如何严格定义？）
  - 此外：“P为可以在多项式时间内计算的问题”、“NP为可以在多项式时间内验证的问题”
  - 即如果单项函数存在则P $\neq$ NP
  - 具体例子：离散对数$g^x \overset{?}{\rightarrow} x$，RSA函数$m^e \overset{?}{\rightarrow} m$，格基SIS假设$\mathsf{A}^\top \mathsf{s} \overset{?}{\rightarrow} \mathsf{s}$，哈希函数等。

Lamport一次性签名

- 后续我们用哈希函数来替代单向函数
- Lamport一次性签名：
  - 签名一个比特：
    - Setup(1^λ)：
      - 随机产生x_0, x_1和k，计算$y_0 = H_k(x_0), y_1 = H_k(x_1)$
      - 公钥：$pk = (y_0, y_1, k)$，私钥：$sk = (x_0, x_1)$
    - Sig($pk, sk, m \in \{0, 1\}$)：
      - 输出$\sigma = x_m$

Lamport一次性签名

- “一次”性签名（One-Time Signature）的安全性
- 获得一次签名以后，无法伪造签名



Lamport一次性签名（一个比特）

- $\text{Setup}(1^\lambda)$：
  - 随机产生x_0, x_1和k，计算$y_0 = H_k(x_0), y_1 = H_k(x_1)$
  - 公钥：$pk = (y_0, y_1, k)$，私钥：$sk = (x_0, x_1)$
- $\text{Sig}(pk, sk, m \in \{0, 1\})$：
  - 输出$\sigma = x_m$
- 是否满足一次性签名安全性? ✔

Lamport一次性签名（多个比特）

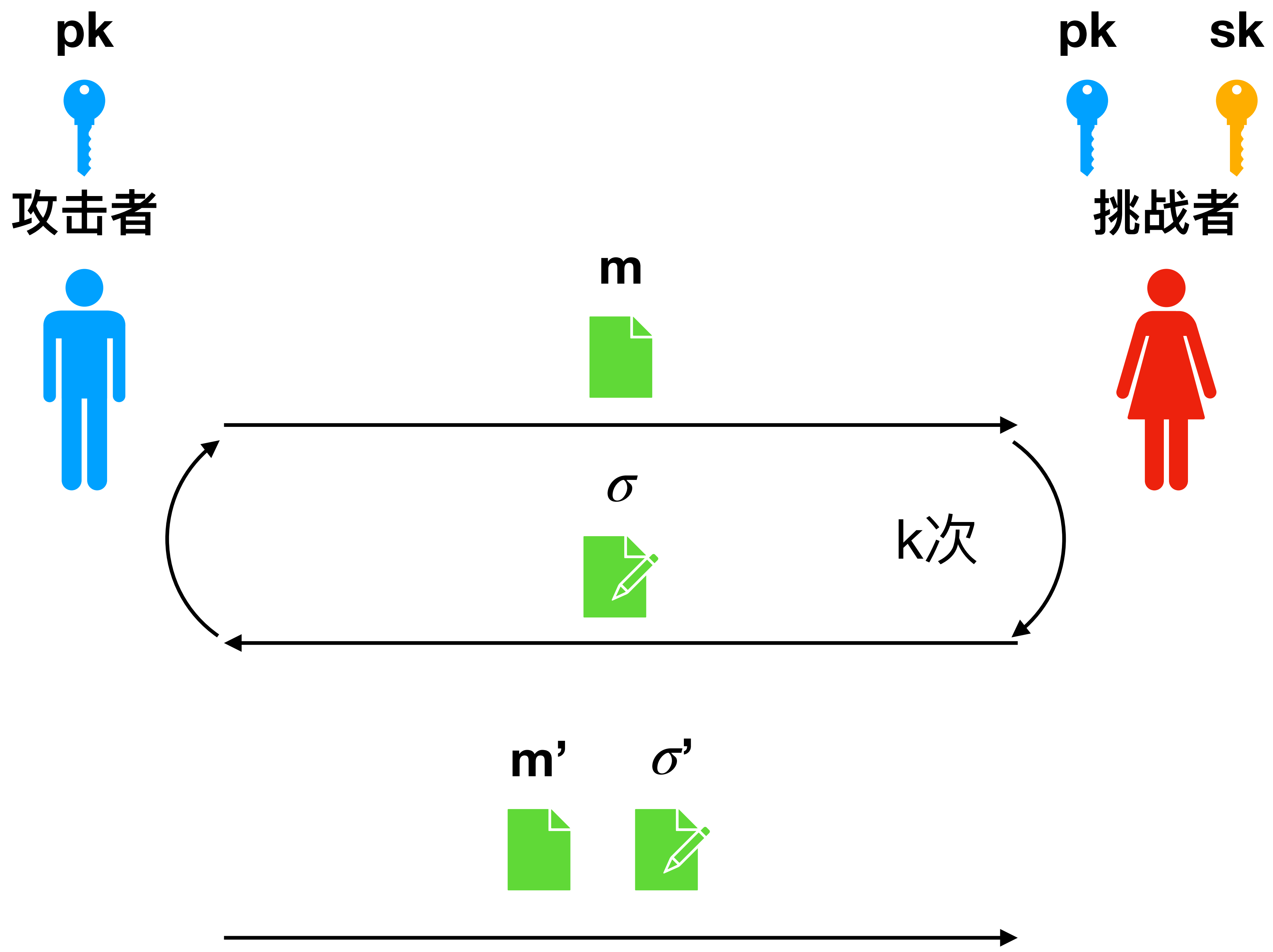
- Setup(1^λ)：
  - 随机产生$\{x_{l,0}, x_{l,1}\}_{l\in\{0,..,n-1\}}$和k，计算$y_{l,0} = H_k(x_{l,0}), y_{l,1} = H_k(x_{l,1})$
  - 公钥：$pk = (\{y_{l,0}, y_{l,1}\}, k)$，私钥：$sk = (\{x_{l,0}, x_{l,1}\})$
- Sig($pk, sk, m \in \{0,1\}^n$)：
  - 输出$\sigma = \{x_{l,m_l}\}_{l\in\{0,..,n-1\}}$
- 是否满足多次签名安全性? ✗

Lamport签名总结

- 优点：
  - 一次性签名安全
  - 基于单向函数（哈希，离散对数，RSA，格基，编码…）
- 缺点：
  - 多次签名不安全
  - 签名长度和信息比特长度线性相关

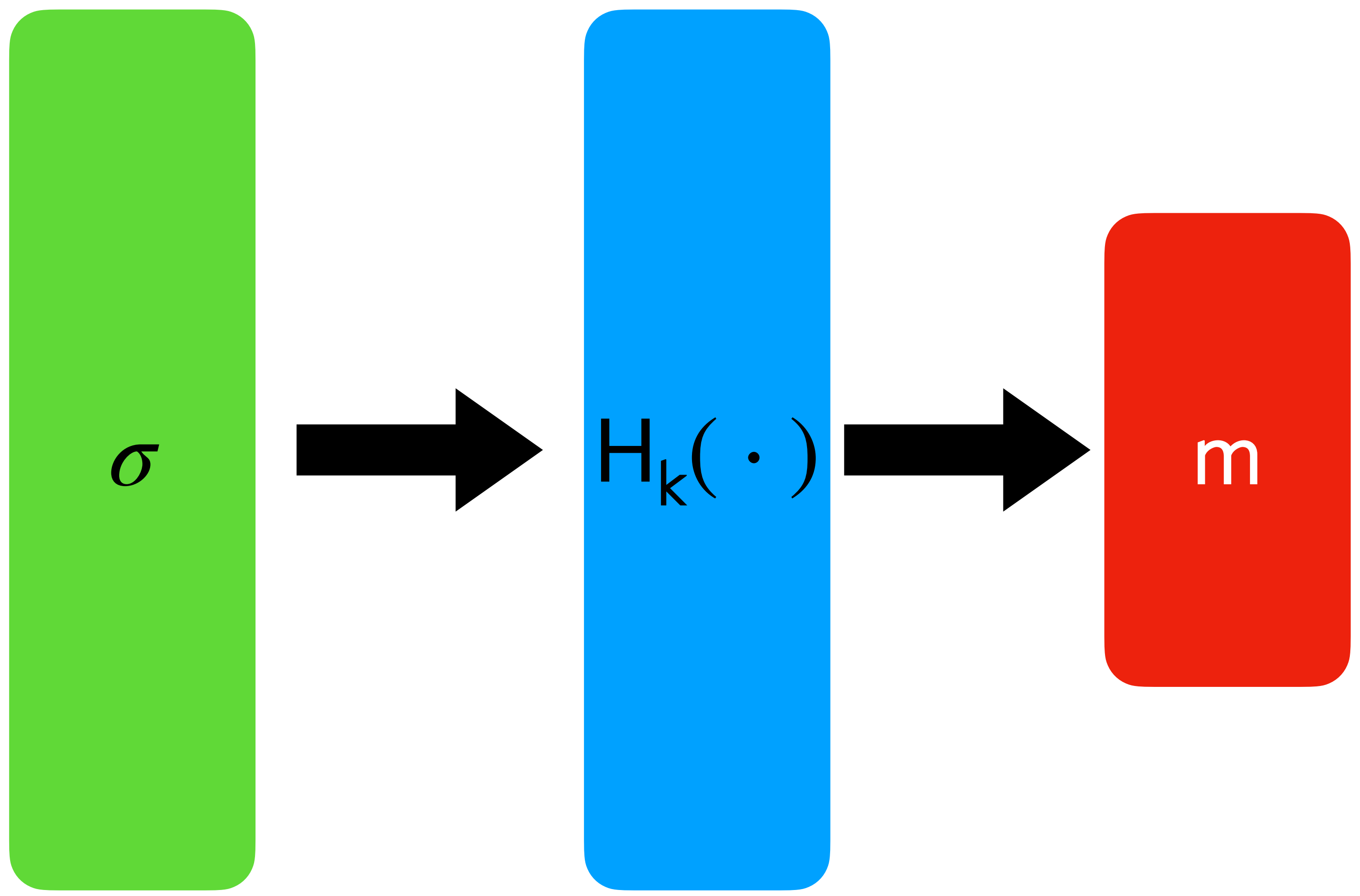
多次签名安全

- 一次签名安全在实际使用过程中不安全
- 多次签名安全?

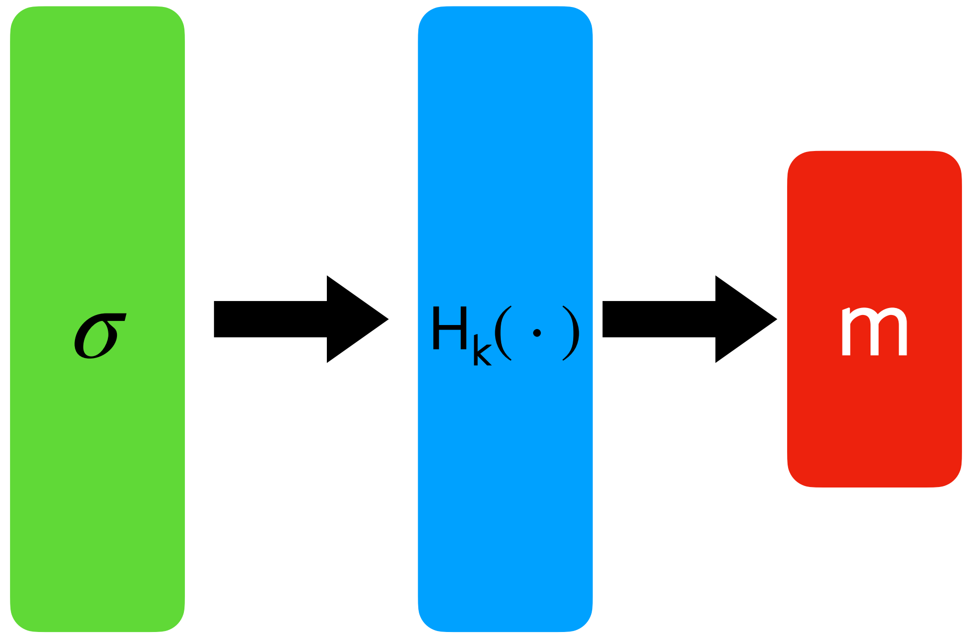


多次签名安全构造

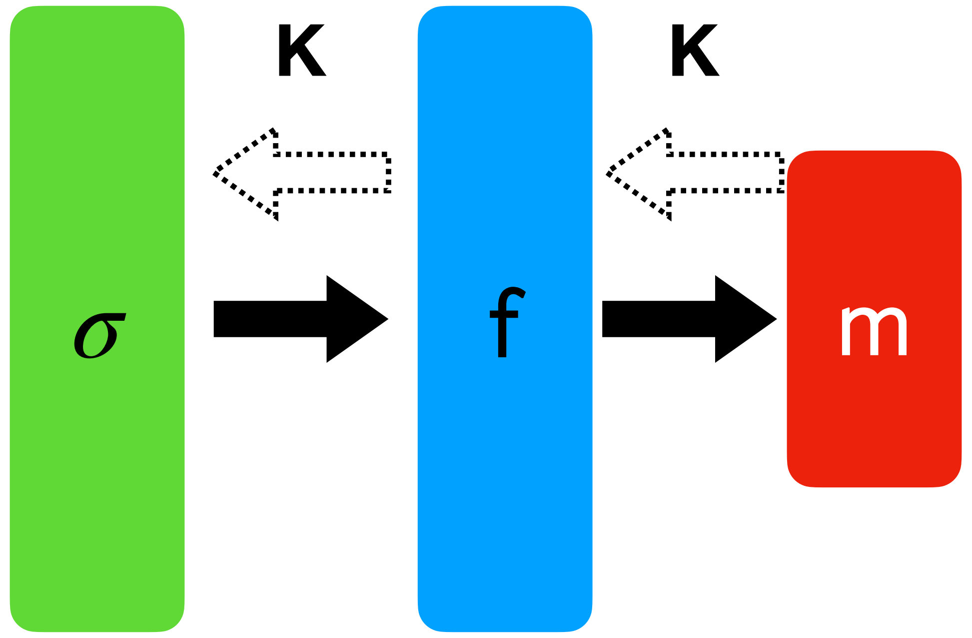
- 我们回顾一下Lamport签名的构造：
  - 单比特Lamport签名可以看成：



多次签名安全构造



- 单比特签名的瓶颈：单向函数导致必须先生成签名再生成信息
- 所以为了生成n比特信息的签名，需要提前准备2^n次方个签名
- 如何压缩呢?
- 陷门单项函数!



多次签名安全构造

- 陷门单向函数：
  - RSA假设：
    - 令n = pq，两个大素数的乘积，且$e \cdot d \equiv 1 \mod \phi(n)$。
    - $f(x) = x^e \mod n$ 是一个陷门单向函数，陷门为d。
- 那么运用RSA陷门函数，我们可以签名单个比特：
  - $\mathrm{PK}: y_0, y_1, e, \mathrm{SK}: d$
  - 签名：$\sigma = f^{-1}(y_m) \equiv y_m^d \mod n$ 验证：$\sigma^e \equiv y_m \mod n$
- 将私钥进行了压缩！

多次签名安全构造

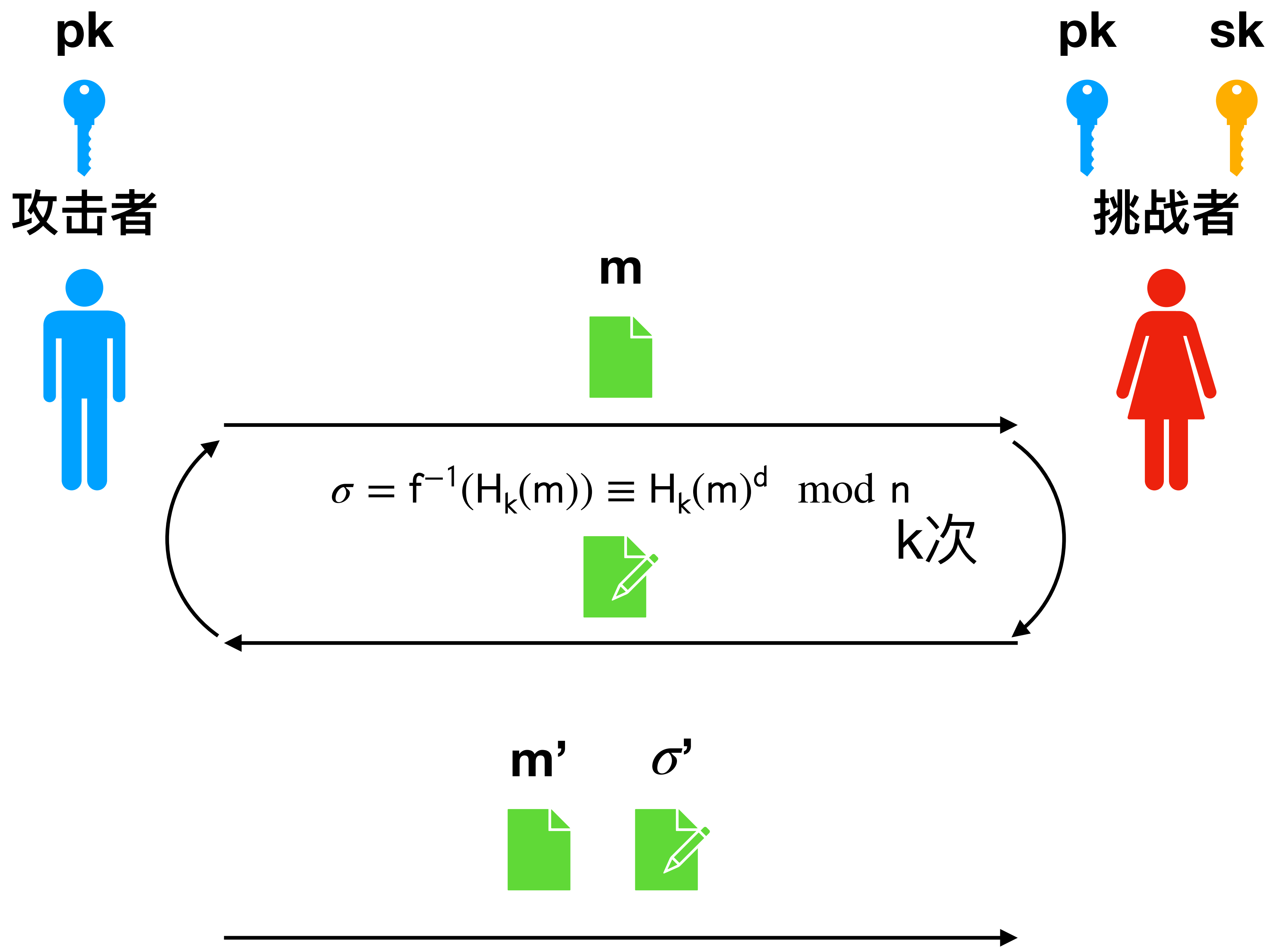
- 多次签名安全的问题实际来自于从单个比特到多个比特签名
- 换种思路?
  - 原来：签名单个比特 → 签名n个比特
  - 现在：签名m_0, m_1→签名$m_0, m_1, \ldots, m_{2^n}$
- 笨办法
  - $\mathrm{PK}: \mathrm{y}_0, \mathrm{y}_1, \ldots, \mathrm{y}_{2^n}, \mathrm{e}, \mathrm{SK}: \mathrm{d}$
  - 签名：$\sigma = \mathrm{f}^{-1}(\mathrm{y}_\mathrm{m}) \equiv \mathrm{y}_\mathrm{m}^\mathrm{d} \mod \mathrm{n}$ 验证：$\sigma^\mathrm{e} \equiv \mathrm{y}_\mathrm{m} \mod \mathrm{n}$
- 满足多次签名安全需要!

多次签名安全构造

- 如何压缩公钥大小呢?
  - $\mathrm{PK}: y_0, y_1, \ldots, y_{2^n}, e, \mathrm{SK}: d$
  - 签名：$\sigma = f^{-1}(y_m) \equiv y_m^d \mod n$ 验证：$\sigma^e \equiv y_m \mod n$
- 我们需要的性质是$y_0, \ldots, y_{2^n}$都是独立随机挑选的
  - 哈希函数！哈希函数是输入的特征值
  - $\mathrm{PK}: H_k, e, \mathrm{SK}: d$
  - 签名：$\sigma = f^{-1}(H_k(m)) \equiv H_k(m)^d \mod n$ 验证：$\sigma^e \equiv H_k(m) \mod n$

多次签名安全性证明

- 随机寓言机模型（ROM）



多次签名安全性证明

- 利用攻击者获得$x = y^d \mod n$
- 第一步：将哈希函数替换成为随机寓言机
- 第二步：提前随机生成$\sigma_1, \sigma_2, \ldots, \sigma_\mathsf{k}$，并令$\mathcal{O}'(m_i) = \sigma_i^e$
- 第三步：攻击者每次询问随机寓言机m都回复$r^e \cdot y$

最后思考题

- 构造签名算法还有另一种思路：
- 什么是签名？要向别人证明文件已经被我同意了！
- 结合上节课提到的如何用零知识证明$f = g^x$
  - 提示：只需要将信息m放入哈希函数中