区块链基本密码算法

范磊
上海交通大学

主要的密码算法

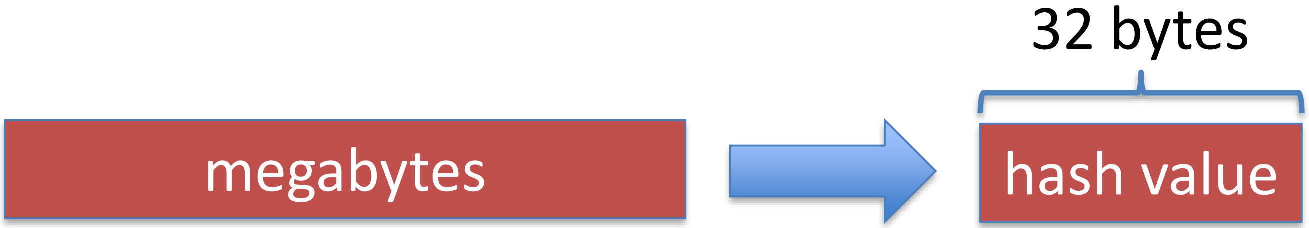
- 单向散列函数
  - 数字摘要，完整性检验
- 公钥密码算法
  - 数字签名，公钥加密
- 对称密码算法
  - 信息加密
- 高级密码算法
  - 环签名、群签名、零知识证明

单向散列函数

密码学单向散列函数描述：

一个可以高效计算的多对一映射函数：H: $M \rightarrow T$

其中集合 $|M| \gg |T|$

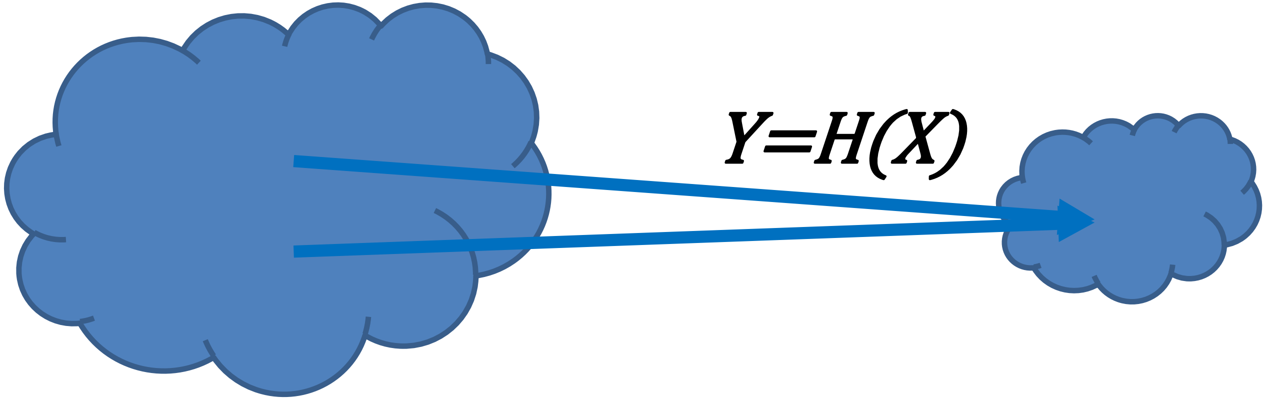


$T = \{0,1\}256$

单向散列函数也称为哈希函数

单向散列函数的碰撞

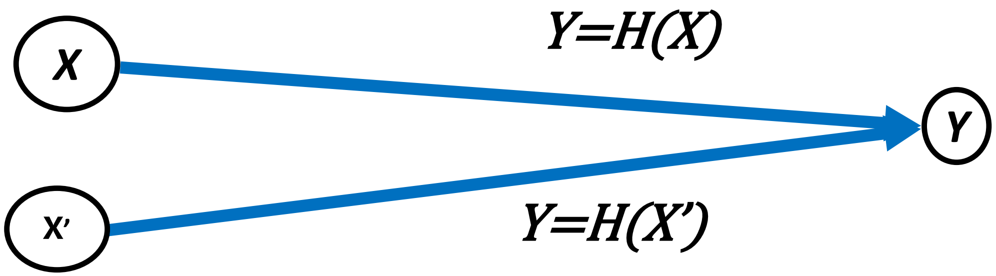
单向散列函数可将任意长度的输入映射为固定长度的输出



哈希函数将任意长度的输入映射为较短的输出，必然会产生**碰撞**，理想的哈希函数会保证碰撞出现的**概率很小**。

哈希函数的两个特性

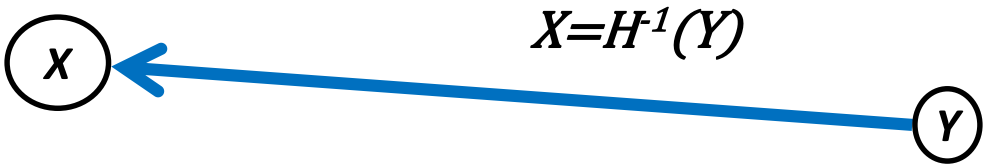
抗碰撞性：



找到X,X'，使得H(X)=H(X')

非常困难

不可逆性：



给定Y，通过逆运算找到$X=H^{-1}(Y)$

非常困难

哈希函数应用1：数据承诺

A拥有数据文件 m.

A公布m的散列值 $h = H(m)$ (32 bytes)， B收到 h.

后续B如果得到 m' , $H(m') = h$，则

B相信 $m' = m$ (否则，m 和m' 是H的碰撞)

$h = H(m)$ 称为m的承诺

序列数据承诺

A有一个数据序列 $S = (m_1, m_2, \dots, m_n)$

目标:

- A公布一个 S的短承诺 $h = \text{commit}(S)$
- B收到h. 给定$(m_i,\ \text{proof}\ \pi_i)$ 检验 $S[i] = m_i$

  B运行 $\text{verify}(h, i, m_i, \pi_i) \rightarrow$ accept/reject

安全性: 敌手不能生成 (S, i, m, π) s.t. $m \neq S[i]$ 并且

$\text{verify}(h, i, m, \pi) = \text{accept}$ 其中 $h = \text{commit}(S)$

序列数据承诺

- 方法1

commit(S) $= h = H(H(m_1), \ldots, H(m_n))$

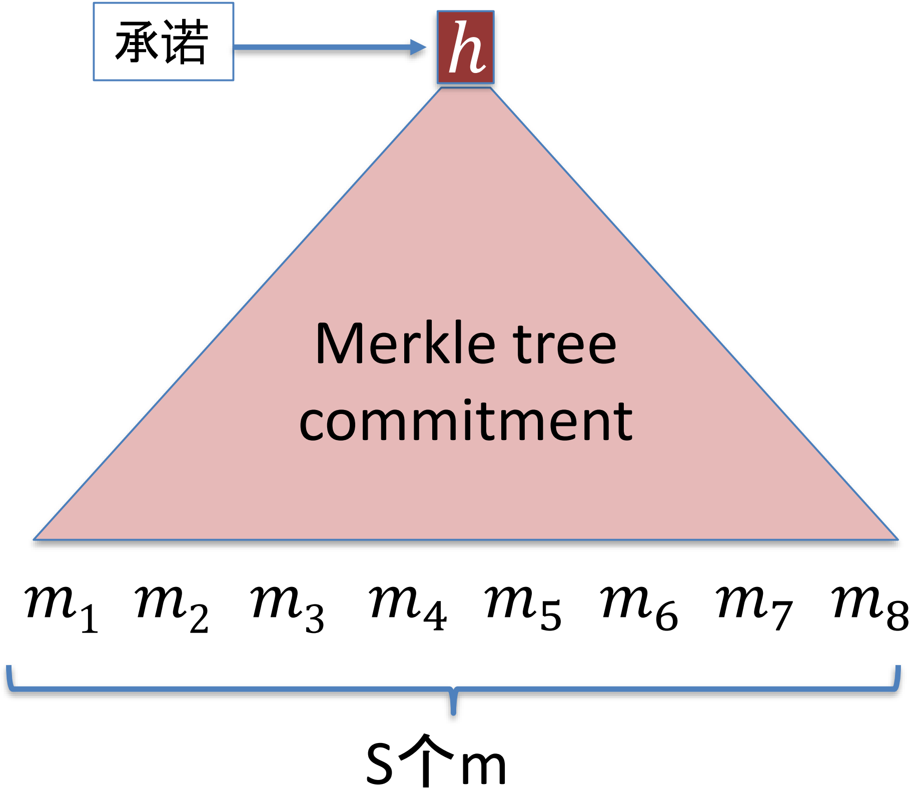
给定h, m_1 以及$\underbrace{H(m_2), \ldots, H(m_n)}_{\text{proof } \pi_1}$ B可以检验 $S[1] = m_1$

问题：证明太长了， $(n-1)$ 哈希值

改进方法： **Merkle tree（默克树）**.

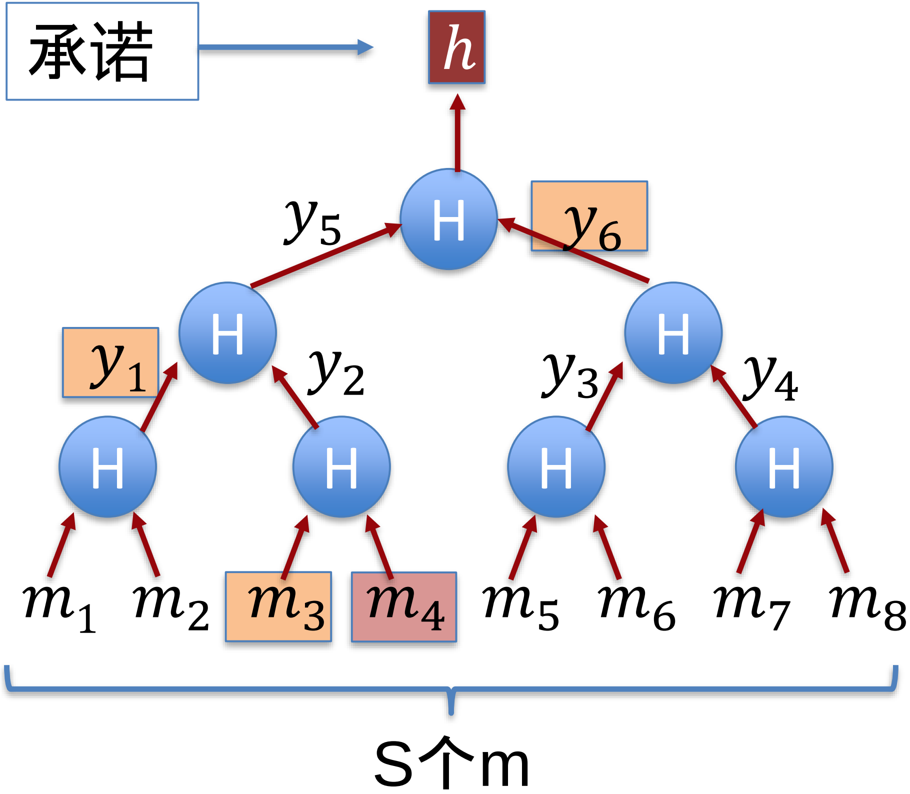
Merkle tree (Merkle 1989)

- 方法2



目标:
- 生成S的承诺
- 可以验证 $S[i] = m_i$

Merkle tree (Merkle 1989)



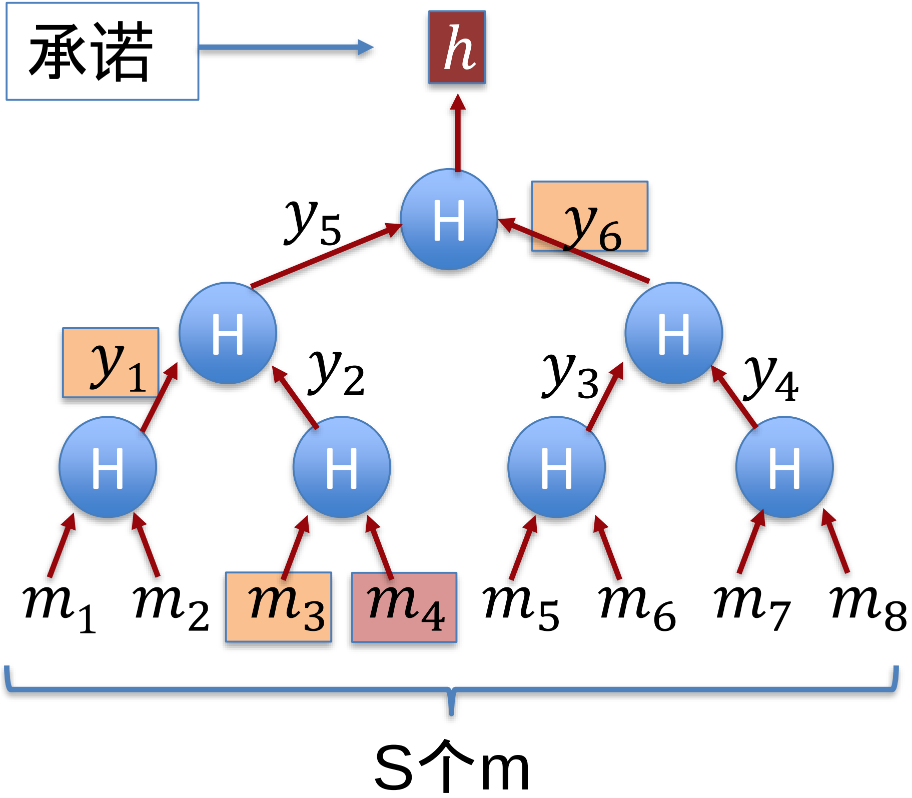
目标:
- 生成S的承诺
- 可以验证 $S[i] = m_i$

验证 $S[4] = m_4$,

proof π = (m_3, y_1, y_6)

π的长度: $\log_2 |S|$

Merkle tree (Merkle 1989)



验证 $S[4] = m_4$ ，

proof $\pi = (m_3, y_1, y_6)$

B验证:

$y_2 \leftarrow H(m_3, m_4)$

$y_5 \leftarrow H(y_1, y_2)$

$h' \leftarrow H(y_5, y_6)$

accept if $h = h'$

哈希函数应用2：工作量证明

目标: 满足如下两个条件的计算任务
- 解决问题的时间复杂度 $\Omega(D)$
- 验证问题的解的时间复杂度O(1)

(D 是困难系数 **difficulty**)

给定哈希函数$H: X \times Y \rightarrow \{0,1,2,\ldots,2^n-1\}$ $n=256$

- 解题: 给定输入 $x \in X$，找到输出 $y \in Y$ 使得 $H(x,y) < 2^n/D$
- 检验(x,y): 如果 $\boxed{H(x,y) < 2^n/D}$ 接受

Bitcoin 使用$H(x) = \text{SHA256}(\text{SHA256}(x))$

公钥密码算法

定义:公钥加密算法由下面三个子算法构成

- **Gen**(): 生成一对密钥对(pk, sk)，其中pk称为公钥，可以公开给所有人，sk为私钥，由密钥所有者私密保存。
- **Enc**(pk, m): 利用公钥对任意消息加密，输出密文s
- **Dec**(sk, s)：利用私钥解密密文，输出明文m

公钥密码算法

定义:数字签名算法由下面三个子算法构成

- **Gen**(): 生成一对密钥对(pk, sk)，其中pk称为公钥，可以公开给所有人，sk为私钥，由密钥所有者私密保存。
- **Sign**(sk, m)： 输出签名信息σ
- **Verify**(pk, m, σ) 输出‘accept’ or ‘reject’

常见的公钥算法

基于大数分解算法：RSA 算法：

- 最早的公钥密码算法，曾经被广泛使用
- 签名以及公私钥长度较长 (≥256 bytes)

基于离散对数算法: Schnorr 和 ECDSA

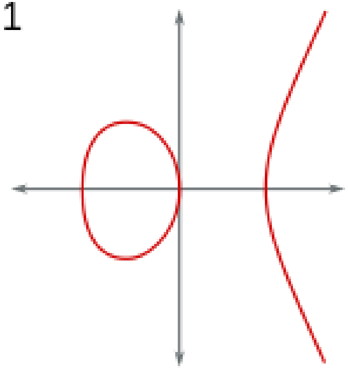
- 较短的签名 (48 or 64 bytes) 和公钥(32 bytes)
- Bitcoin, Ethereum使用

BLS 签名:

- 较短的签名（48 bytes），确定性签名，支持聚合，用于构造VRF函数
- Ethereum 2.0, Chia, Dfinity，iChing等新共识协议使用

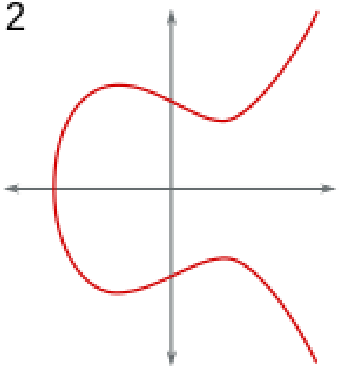
椭圆曲线密码算法

1

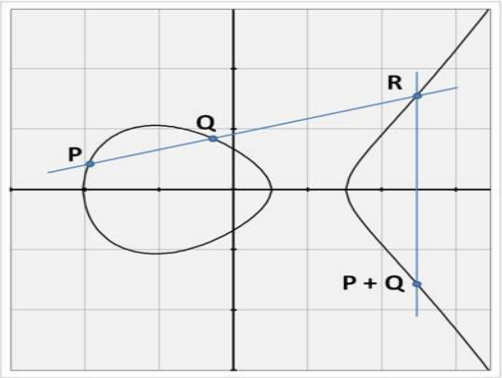


$y^2 = x^3 - x$

2



$y^2 = x^3 - x + 1$



- $y^2 = x^3 + ax + b$
- $4a^3 + 27b^2 \neq 0$
- $a, b, x, y \in K$
- O 点为无穷远

$$m = \frac{y_Q - y_P}{x_Q - x_P}$$

$$x_{P+Q} = m^2 - x_P - x_Q$$

$$y_{P+Q} = m\left(x_P - x_{P+Q}\right) - y_P$$

ECDSA签名

- 参数：
  - 椭圆曲线C，其上的点构成有限域，阶为n的生成元G，其中n为素数
  - 单项散列函数H
  - 随机选择私钥d，生成其对应公钥$Q = dG$
- 签名 针对消息M
  - 选择随机数k，计算点$(x, y) = k \times G, r = x \bmod n$
  - 生成消息摘要$e = H(M)$，z是e的最左边L_n位，L_n是n的长度
  - 计算$s = k^{-1}(z + rd) \bmod n$
  - 签名对为：(r, s)

如果$r, s = 0$，重新计算

ECDSA签名(续)

- 验证
  - 生成消息摘要$e = H(M)$，z是e的最左边L_n位，L_n是n的长度
  - 计算$u_1 = zs^{-1} mod\ n$，$u_2 = rs^{-1} mod\ n$
  - 计算椭圆曲线上的点$(x, y) = u_1 \times G + u_2 \times Q$
  - 检验$r \equiv x\ mod\ n$
  - 其中检查r, s, Q的有效性，检查$(x, y) \neq O$
- 公钥提取
  - 可从签名中提取公钥，减少公钥传输长度

Schnorr签名 （Claus Schnorr)

- 参数：
  - 阶为q的群G，及其中的生成元g
  - 单项散列函数$H: \{0,1\}^* \to \mathbb{Z}_q$
  - 随机选择私钥x，生成其对应公钥$y = g^x$
- 签名 针对消息M
  - 选择随机数k，计算$r = g^k$，$e = H(r||M)$，$s = k - xe$
  - 签名(s, e)
- 验证
  - 计算$r_v = g^s y^e$,$e_v = H(r_v||M)$
  - 检验$e_v = e$

可证明安全
多个签名可以聚合

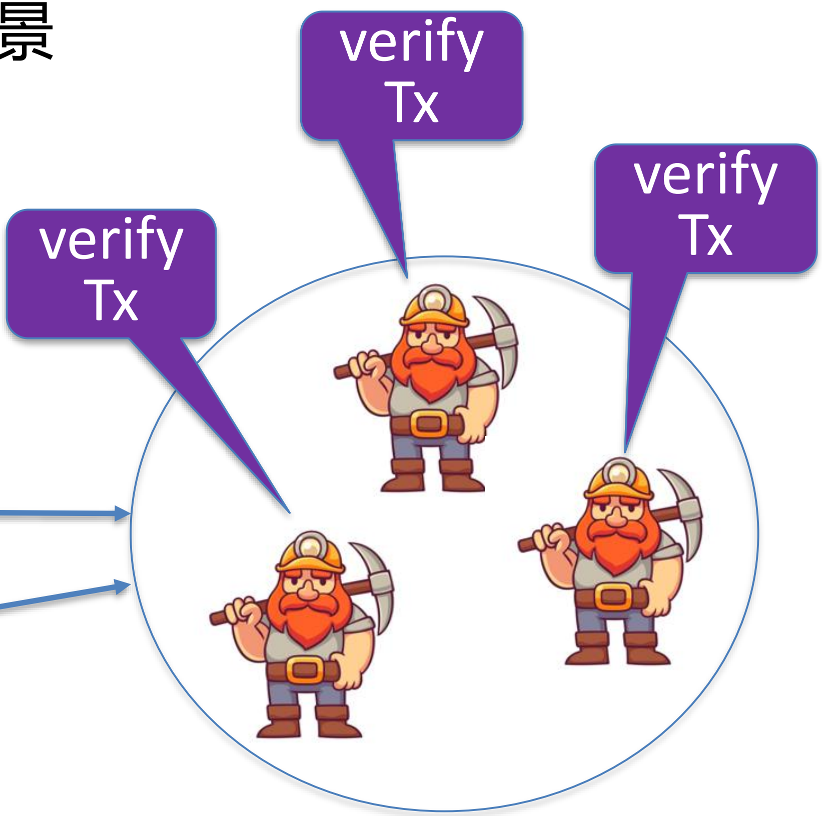
BLS签名

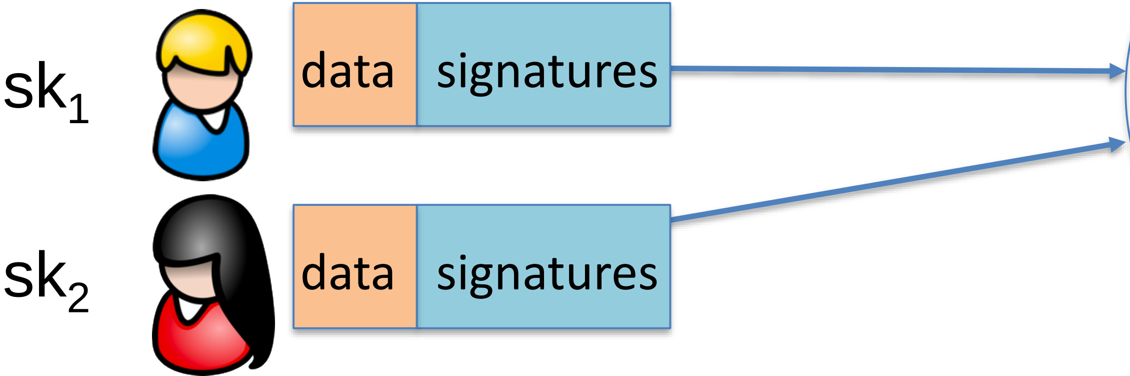
- 双线性映射
  - 三个阶为q的群，一个映射函数：$e\colon \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$，$g_0$与$g_1$分别是$\mathbb{G}_0$与$\mathbb{G}_1$的生成元
  - 单项散列函数$H_0\colon M \to \mathbb{G}_0$
  - 随机选择私钥x，生成其对应公钥$y = g_1^x \in \mathbb{G}_1$
- 签名 针对消息M
  - 生成签名$\sigma \leftarrow H_0(M)^x \in \mathbb{G}_0$
- 签名验证
  - $e(g_1, \sigma) = e(y, H_0(M))$

确定性签名
任意聚合能力

数字签名在区块链中的应用

- 区块链中任何需要证明来源的场景
  - 交易信息确认
  - 治理投票
  - 共识协议





谢谢