Bitcoin基础：密码学基础

吴嘉婧
副教授

中山大学
计算机学院

What is Cryptography?

From wiki:

- Cryptography is the practice and study of techniques for secure communication in the presence of third parties (called adversaries).

第三方存在下的安全通信技术的研究与实践。

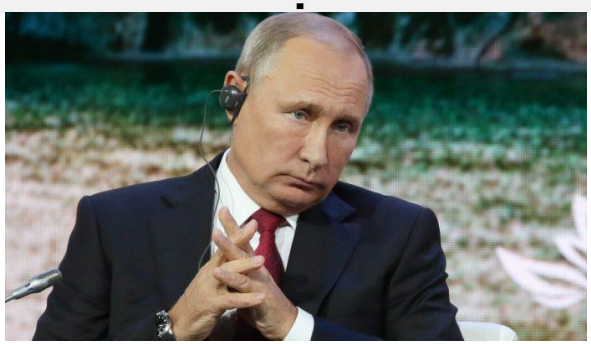
What is Cryptography?

著名的密码学者Ron Rivest解释道：

“密码学是研究如何在敌人存在的环境中通讯”

甲

乙

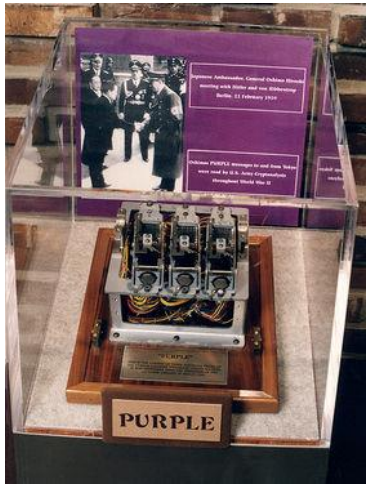


丙

密码学悠久历史

古典密码学（1976年以前）

Hieroglyphic encipherments of proper names and titles, with cipher hieroglyphs at left, plain equivalents at right



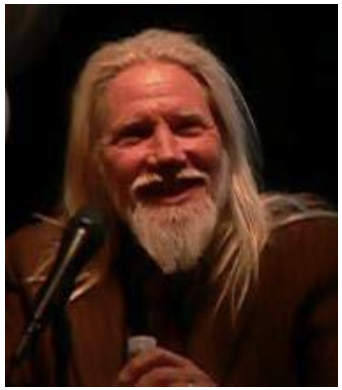
密码学悠久历史

现代密码学（1976/1977年）

代表性事件

- 公钥密码学的提出和美国数据加密标准DES的颁布

左为Hellman，右为Diffie



Whitfield Diffie，

Martin Hellman

意义：

- 密码学成为了一门科学，研究从军事和外交走向了公开

密码学主要研究内容

公钥加密

数字签名

私钥加密：分组密码，流密码

Hash函数

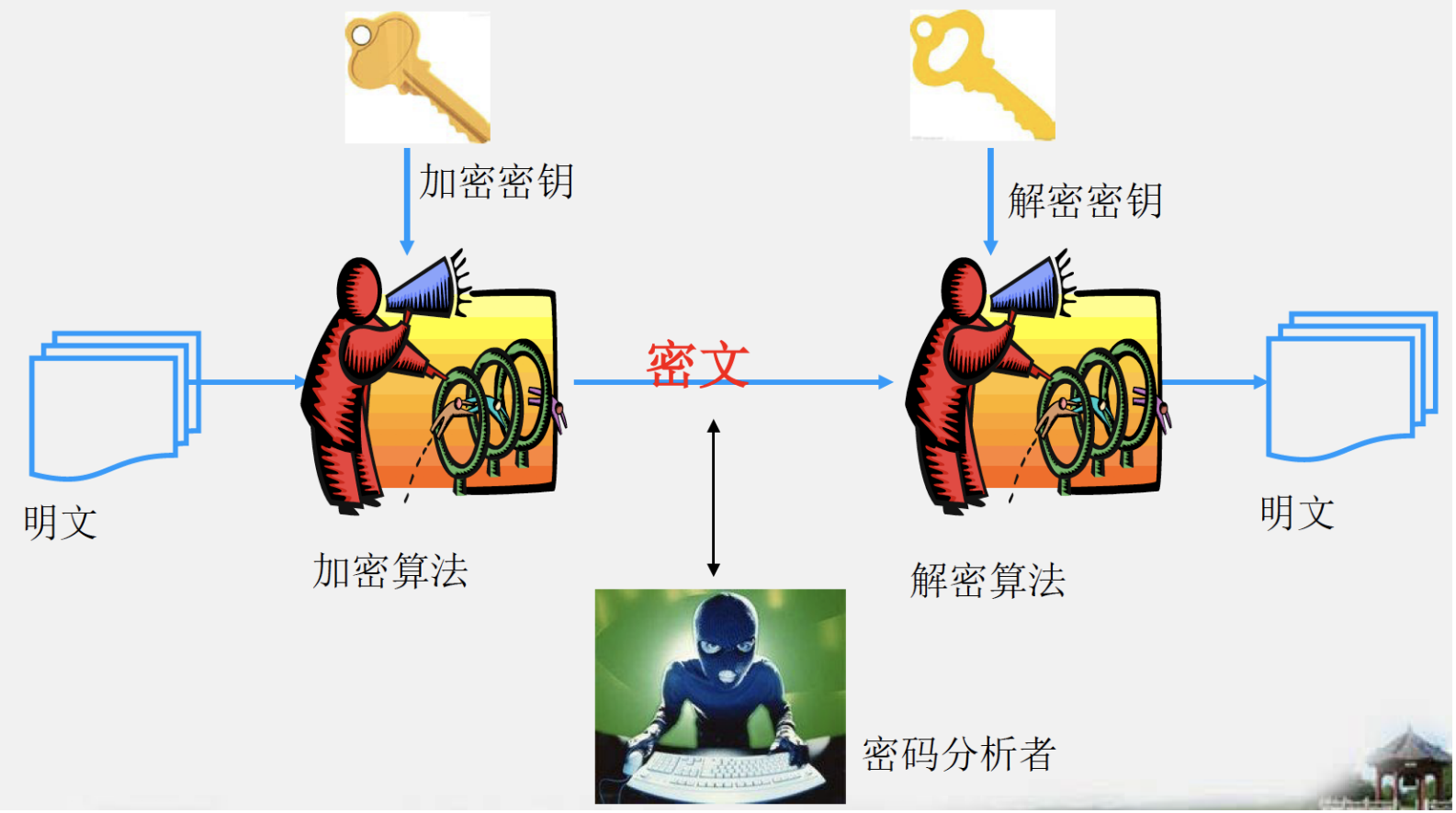
伪随机数

安全协议：承诺，

零知识证明，

多方计算等

引言 —— 基本概念

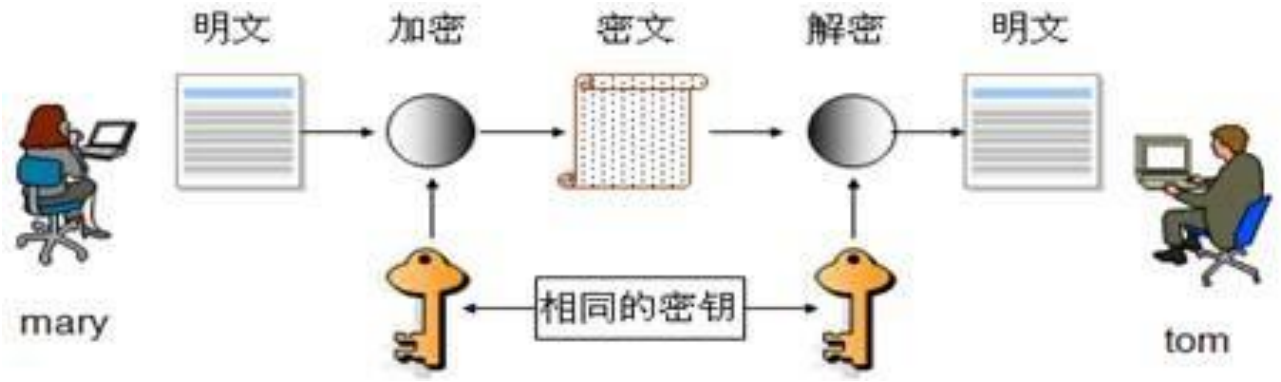


引言 —— 密码学基础

对称加密

使用相同的密钥

加密大量数据



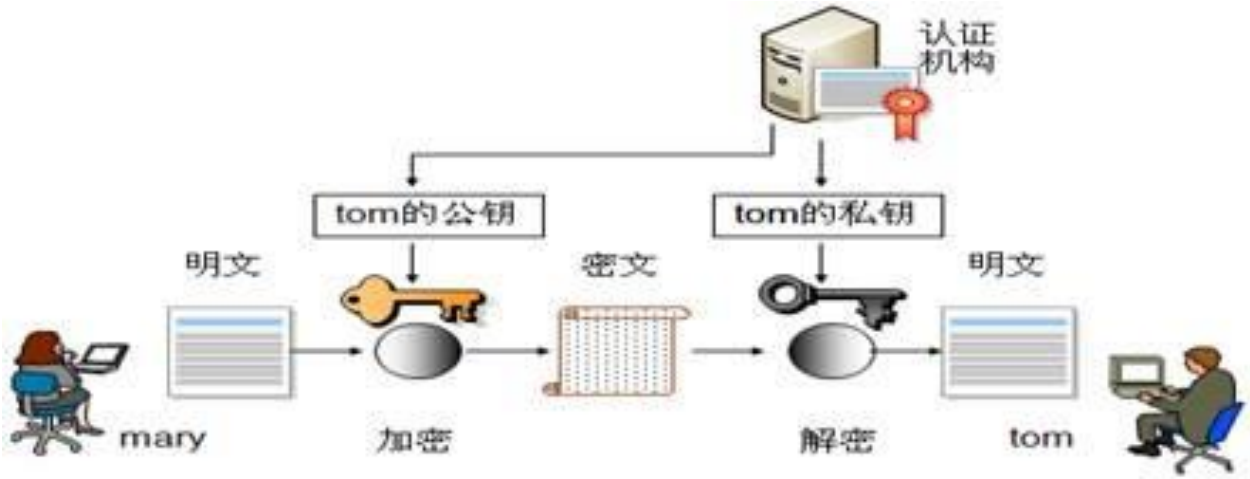
非对称加密

采用不同的密钥

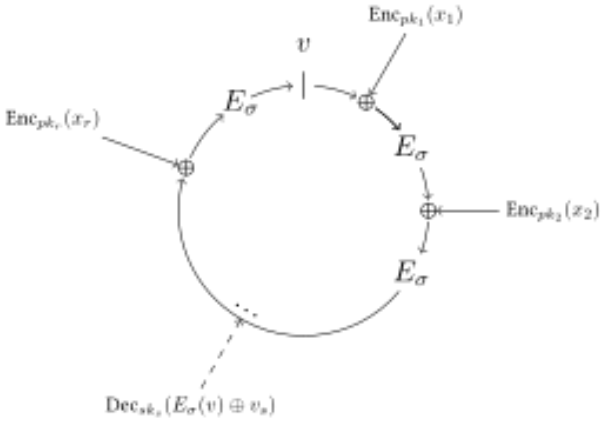
加密少量数据

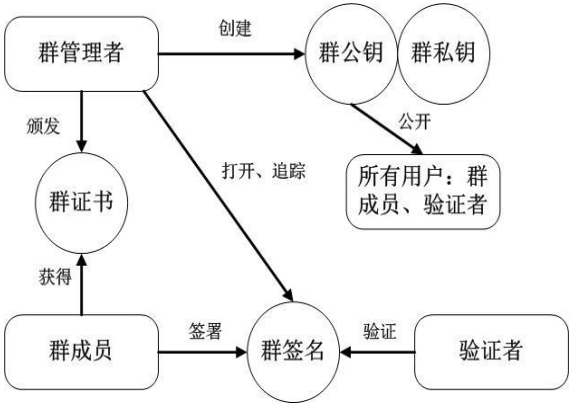
用于交换对称密钥

用于签名验签



引言 —— 博大精深的密码学算法



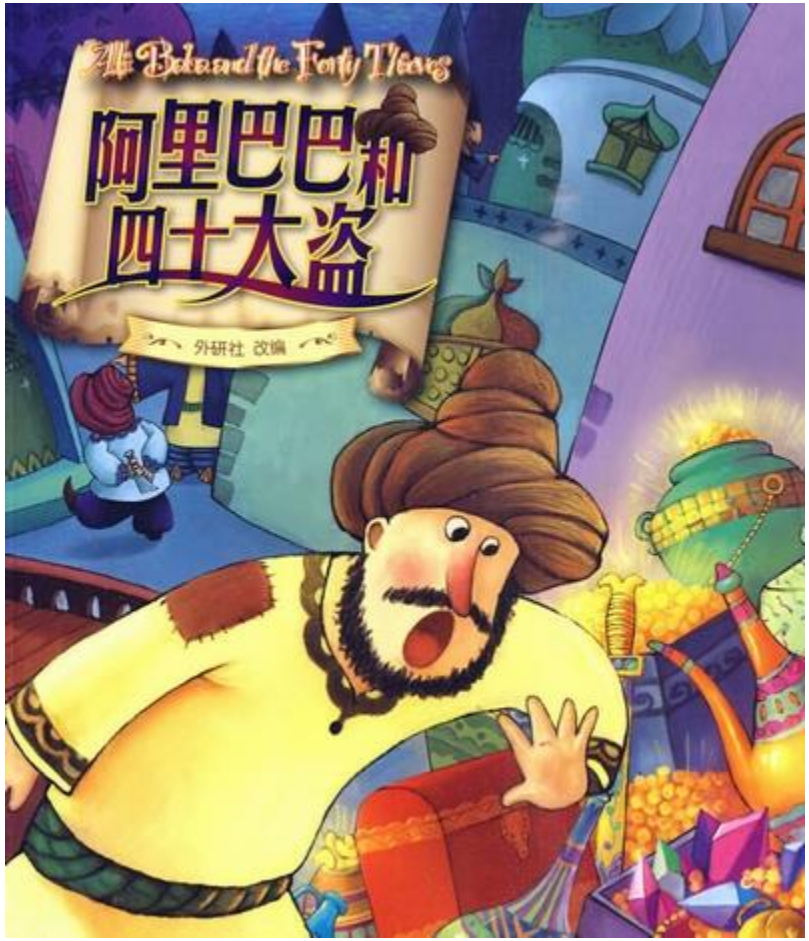


- 零知识证明
- 同态加密
- 属性加密，格密码学
- 群签名，环签名，盲签名、代理签名、门限签名…
- 量子密码

引言 —— 博大精深的密码学算法

- 零知识证明Zero Knowledge
  - S.Goldwasser、S.Micali及C.Rackoff在20世纪80年代初提出
  - 证明者能够在不向验证者提供任何有用的信息的情况下，使验证者相信某个论断是正确的。
  - Zcash零币：比其他加密货币更强的隐私性。

零知识证明：阿里巴巴和四十大盗



- 阿里巴巴的零知识证明
  - 阿里巴巴想要向强盗证明自己拥有密码，又不想把密码告诉强盗。
  - 让强盗离开一箭之地，距离足够远听不到口令，足够近无法在龚剑下逃生。
  - 通过看强盗的手势展示开关门
  - 零知识（不提供石门口令）证明（阿里巴巴知道石门打开的方法）

引言 —— Keywords of this Class

- **哈希(HASH):**表示大量数据的唯一摘要值。原数据的少量更改会在哈希值中产生不可预知的大量更改,可以作为数据的验证凭据
- **数字签名**：信息的发送者（掌握私钥）能产生的别人无法伪造的一段数字串，且可以通过其公布出去的公钥验证是由他发送。

HASH:

完整性,正确性

签名 → 验签

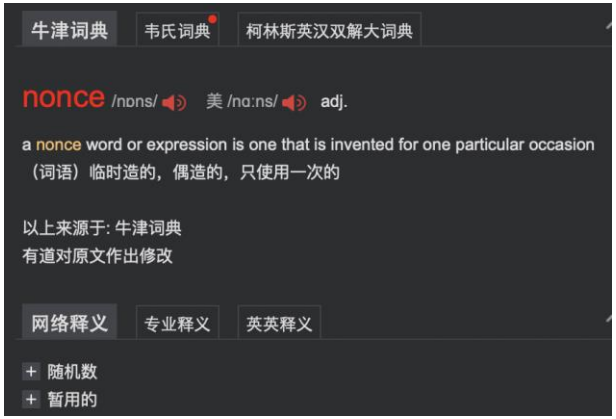
数字签名:

所有者确权

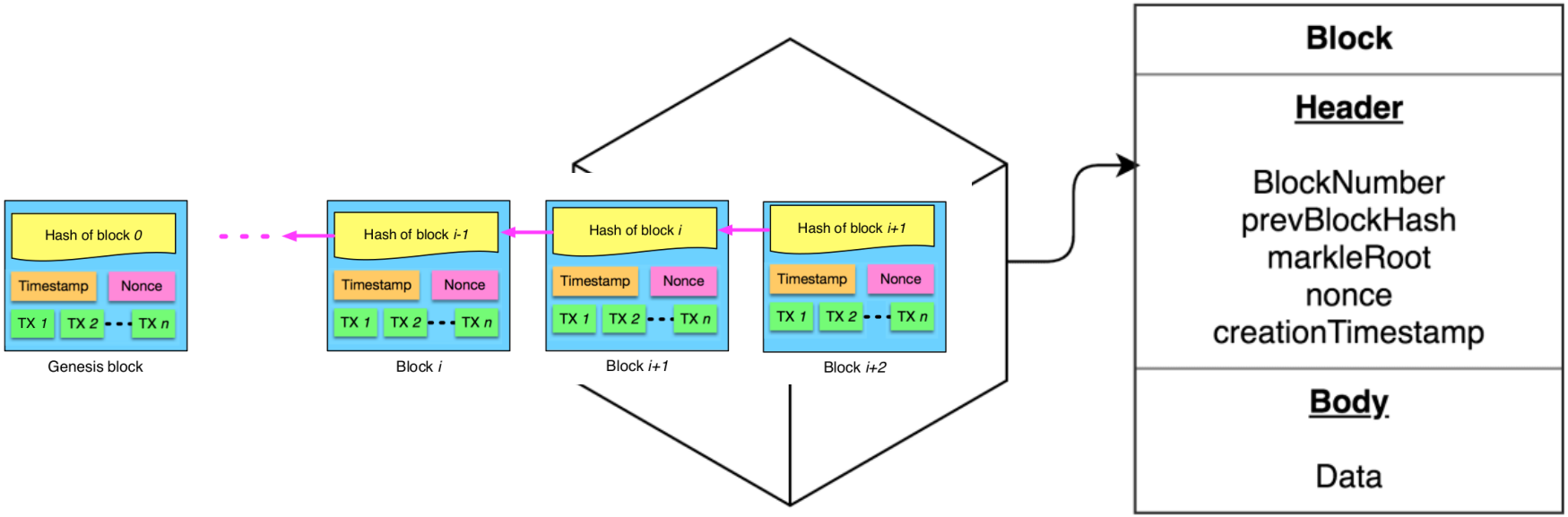
引言 —— Keywords

比特币的密码学相关的关键名词：

- 哈希函数 & 数字签名



- 背后的原理是什么?
- 为什么要使用它们?



比特币密码学基础

PART 1: HASH FUNCTION

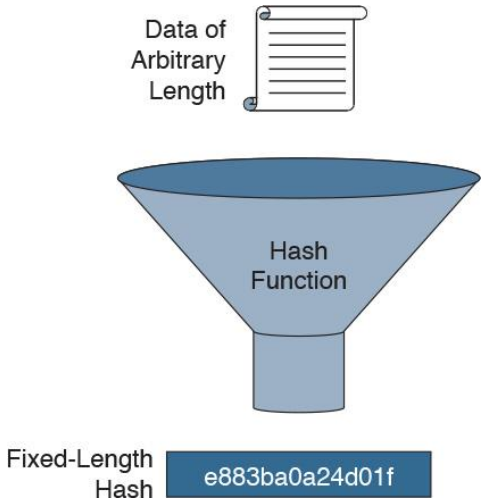
Hash Function

定义：Hash函数是将任意长度的消息映射成一个较短的定长输出消息的函数。

- 如下形式: h = H(M), M是变长的消息, h是定长的Hash值.

目的：

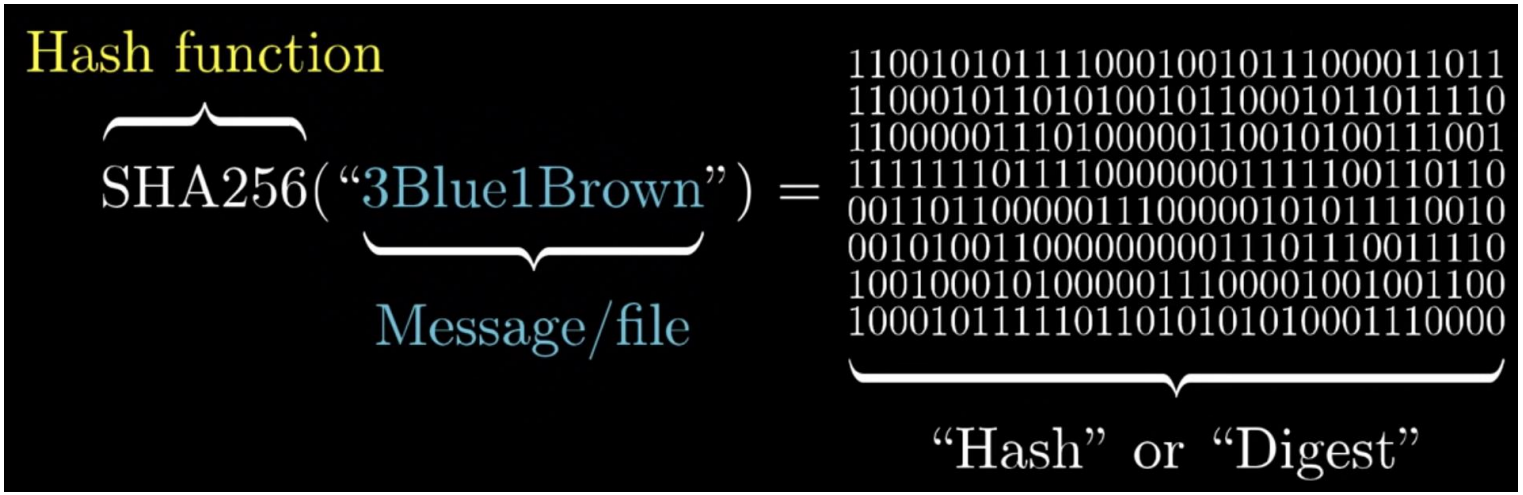
- 为文件、 消息或其它的分组数据产生 “数字指纹”



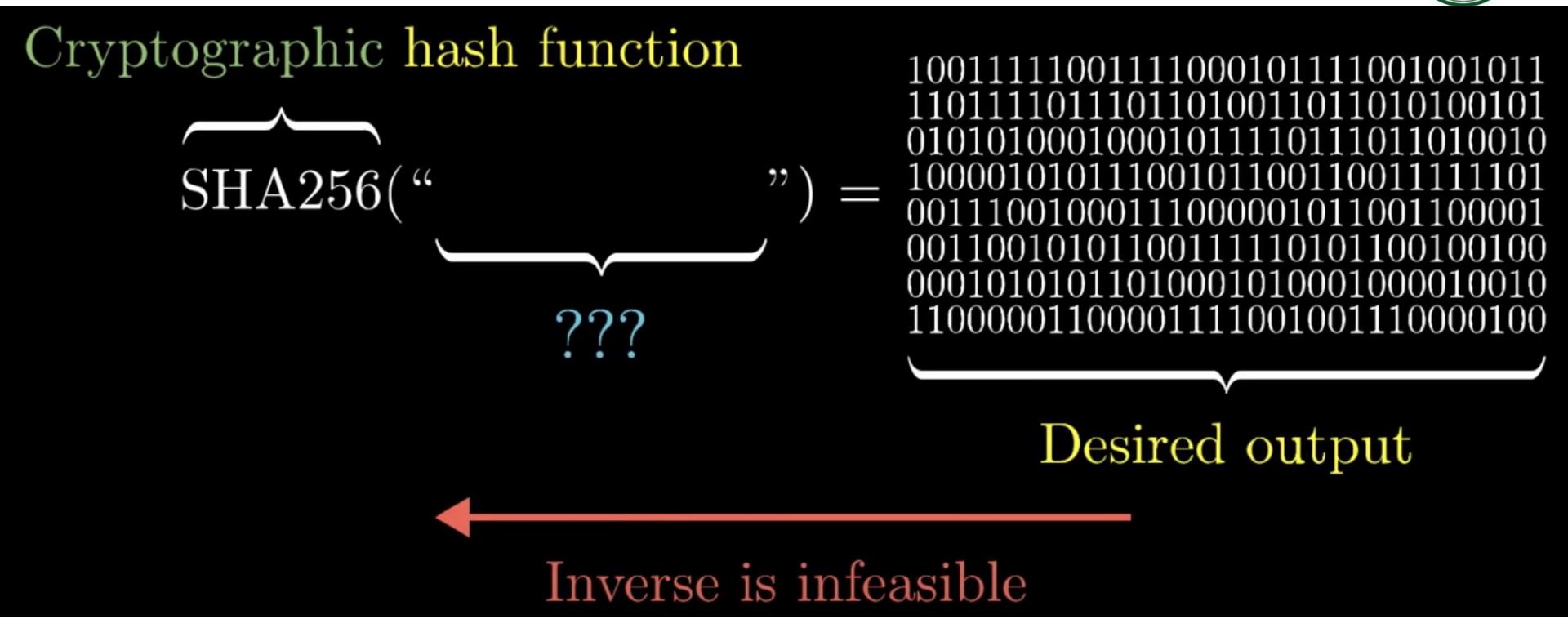
Cryptography of Bitcoin – Hash Function

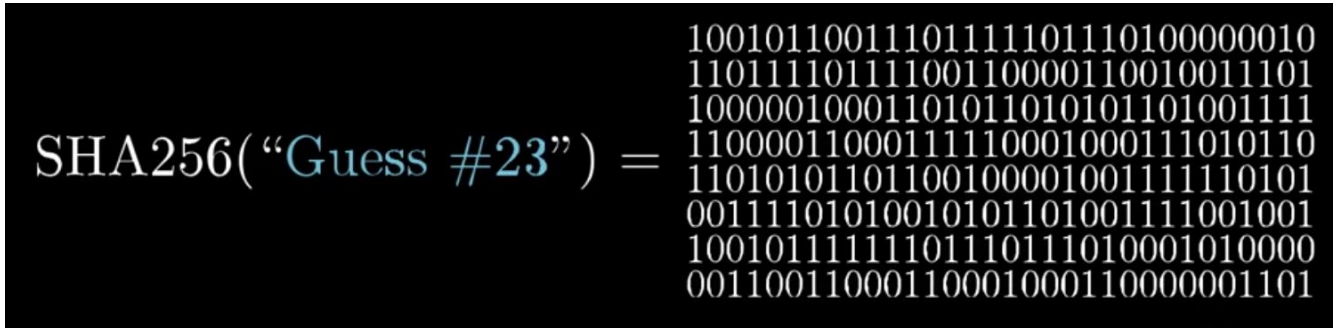
密码学**哈希函数（Hash Function）**，特性

- 输入可以为 任意大小的 string
- 输出固定大小(长度)，e.g., 256 bit-long
- 有效计算：特定的输入字符串，合理时间内输出 —— *O*(n) 复杂度

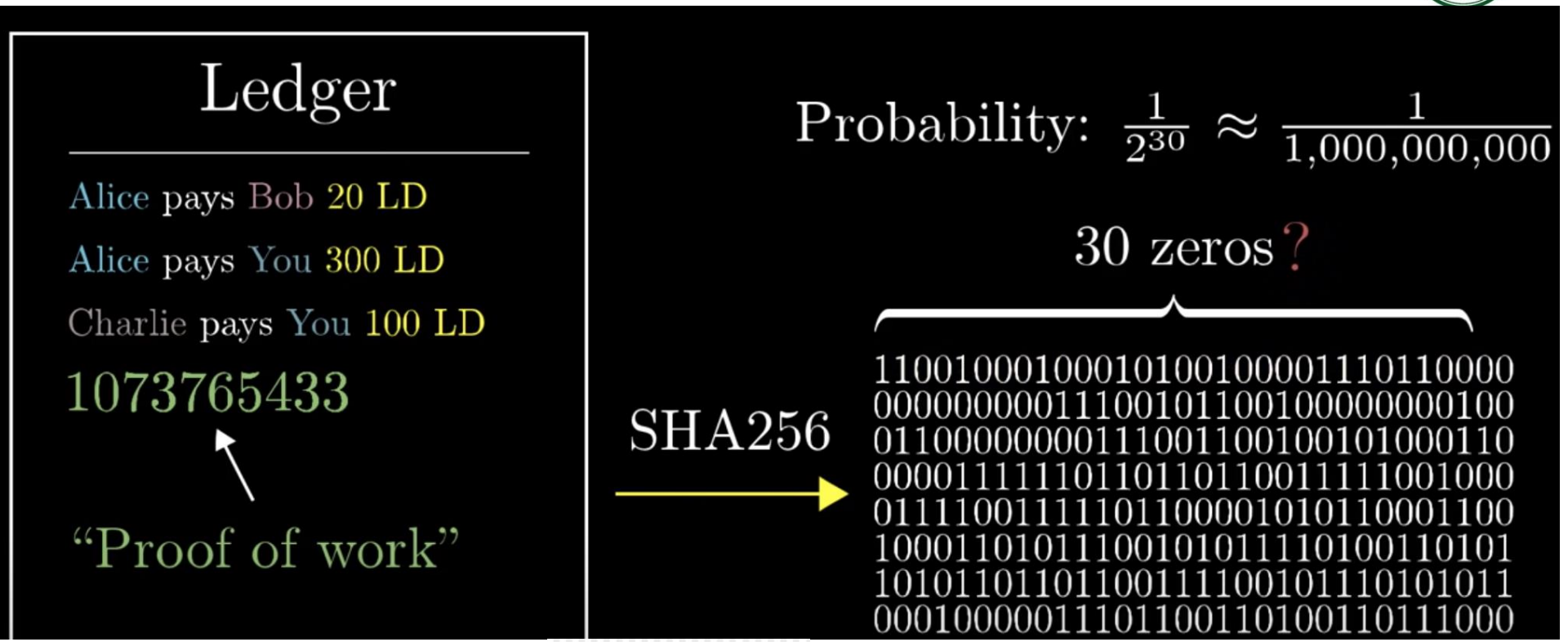


Property of Cryptographic Hash Function





剧透：Proof of Work, & Difficulty



Cryptography of Bitcoin

如果达到***密码学安全***，还需如下附加特性

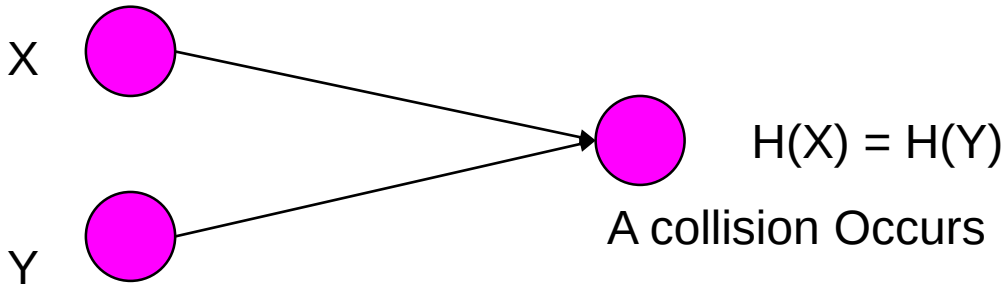
- 碰撞阻力 (collision-resistance)
- 隐秘性 (hiding)
- 谜题友好 (puzzle-friendliness)

密码学特性1-碰撞阻力 (Collision-Resistance)

Definition

- A hash function H is said to be collision resistant if it is infeasible to find two values, x and y , such that x != y, yet H(x)= H(y).

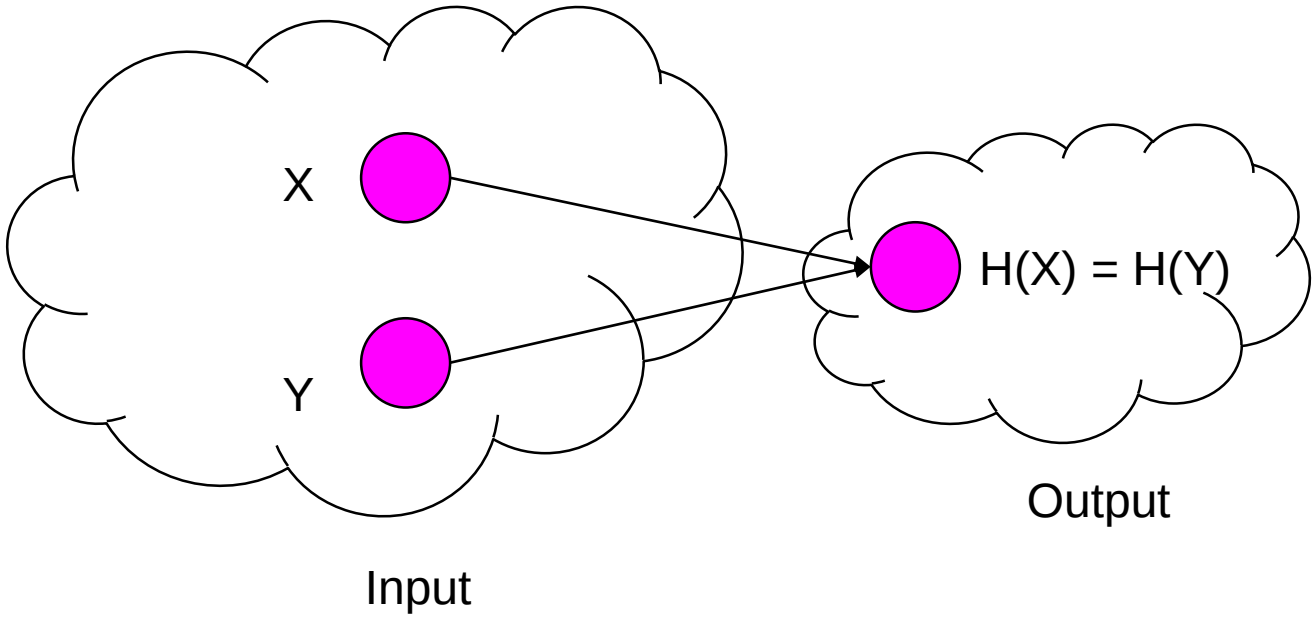
- 如果无法找到两个值，x 与 y, x!=y, 而 H(x) = H(y)，那么，称哈希函数 H 具有碰撞阻力。



密码学特性1-碰撞阻力 (Collision-Resistance)

特性1—— 碰撞阻力 (Collision-Resistance)

- 找不到碰撞，不代表碰撞不存在



密码学特性1-碰撞阻力 (Collision-Resistance)

特性1—— 碰撞阻力 (Collision-Resistance)

- 找不到碰撞，不代表碰撞不存在

Theorem: 假如有n+1个元素放到n个集合中去，其中必定有一个集合里至少有两个元素。

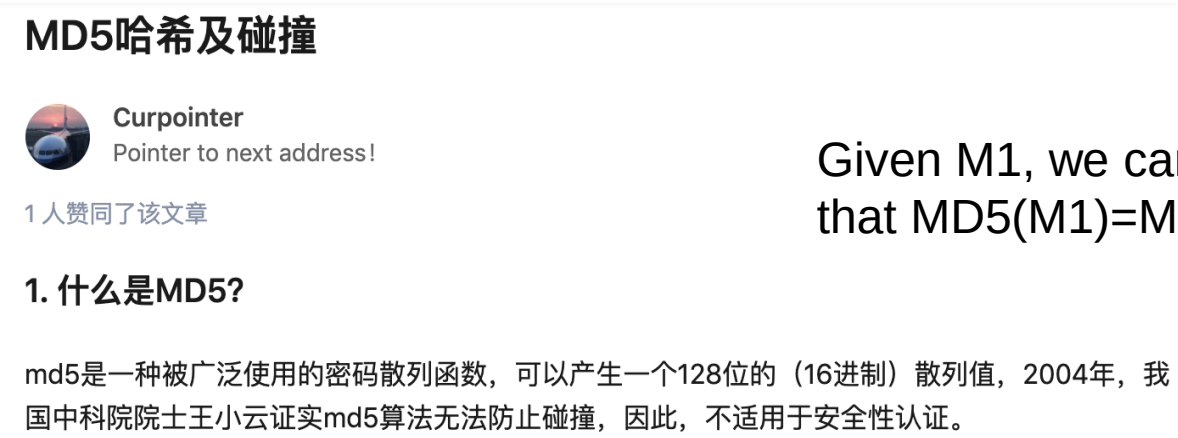
10只鸽子放进9个鸽笼，那么一定有一个鸽笼放进了至少两只鸽子。

随机源 matters

密码学特性1-碰撞阻力 (Collision-Resistance)

随机碰撞检测

- E.g.,SHA256, the worst case: conduct 2^256 +1 times of hash detection
- Average # of detections: 2^128
- 10^27 年 = A PC (10000 hashes/sec) X 2^128

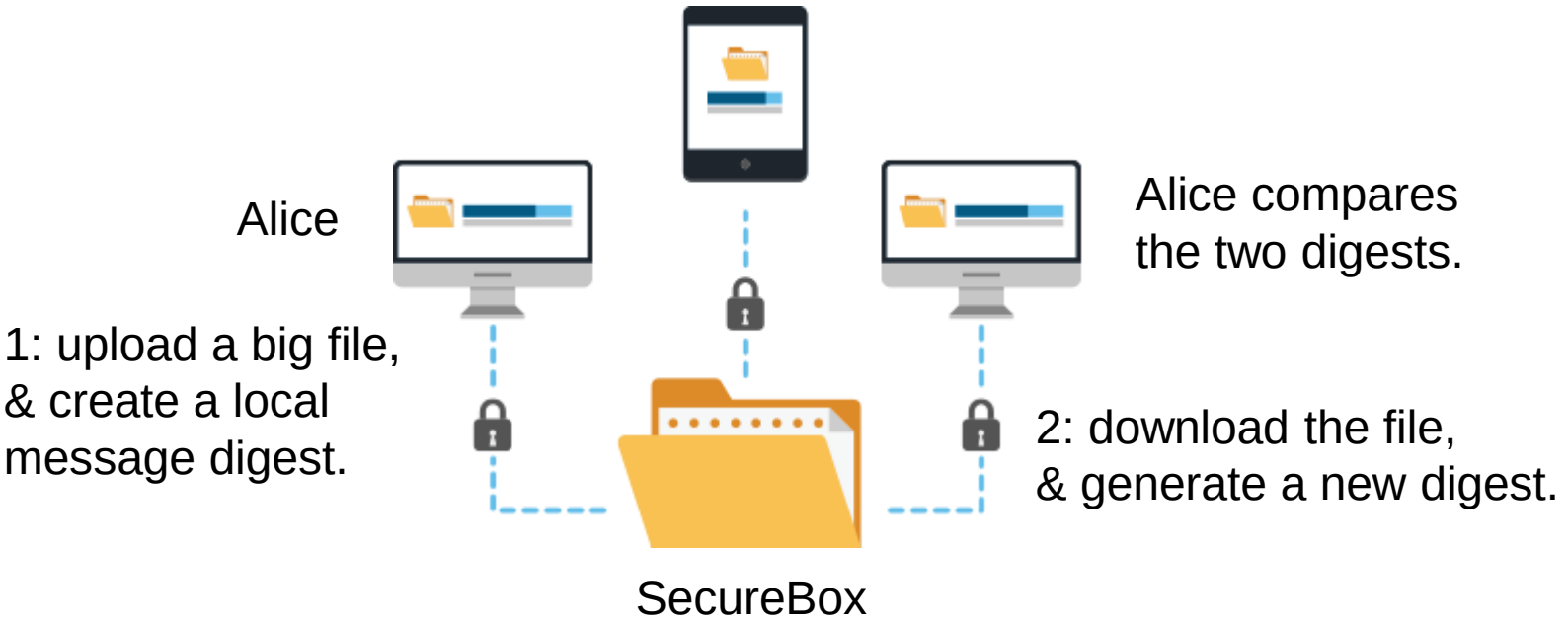


Given M1, we can find M2, such that MD5(M1)=MD5(M2).

密码学特性1-碰撞阻力 (Collision-Resistance)

有什么用?

- 信息摘要 (Message Digest)
  - X!=Y, 则 H(X) != H(Y)
  - 例子：文件存储系统，生成信息摘要 SHA256(file)



密码学特性2 – 隐秘性 (hiding)

Definition

- A hash function H is hiding if: when a secret value *r* is chosen from a probability distribution that has high entropy, then given H(r ‖ x) it is infeasible to find *x*.
- ‖ means concatenation of two strings.

隐秘性： 哈希函数H具有隐秘性，如果：当其输入r选自一个高阶最小熵（high min-entroy）的概率分布，在给定 H（r‖x）条件下来确定x是不可行的。

密码学特性2 – 隐秘性 (hiding)

作用

- 保证：如果仅仅知道哈希函数的输出 y = H(x)，则没有可行的办法算出输入值 x.

- 要求：
  - x 需要取值自一个很广泛的集合
  - 仅仅通过尝试几个特定的 x，找不到特定的输出值

- 如果：x 的取值并非来自分散的集合，怎么办？
  - e.g.,抛硬币实验：H (正面朝上) = “正面”, H (正面朝下) = “反面”。

密码学特性2 – 隐秘性 (hiding)

隐秘处理：

- 如果：x 的取值并非来自分散的集合，怎么办?
- 通过与另一个较为分散的输入进行结合

隐秘性： 哈希函数H具有隐秘性，如果：当其输入 *r* 选自一个高阶最小熵（high min-entroy）的概率分布，在给定 H（*r* ‖ x）条件下来确定x是不可行的。

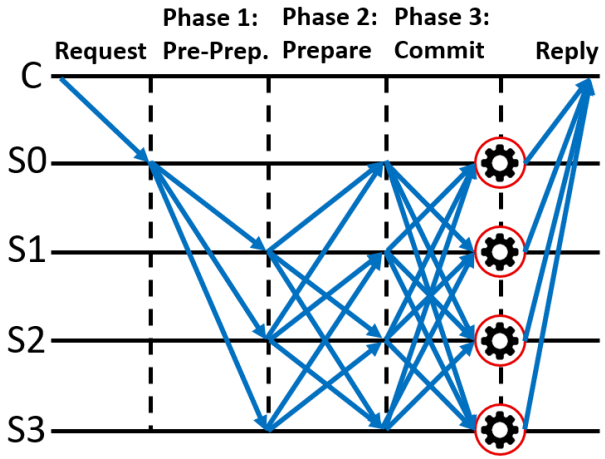
- 最小熵：用于测试 结果可预测性 的手段
- 高阶最小熵：一个(随机)变量分布的分散程度
  - 从这样的分布中取样时，我们无法判定取样的倾向
  - e.g., if *r* is selected randomly from a set of 256-bit strings, thus, a specific string is selected in ½^256

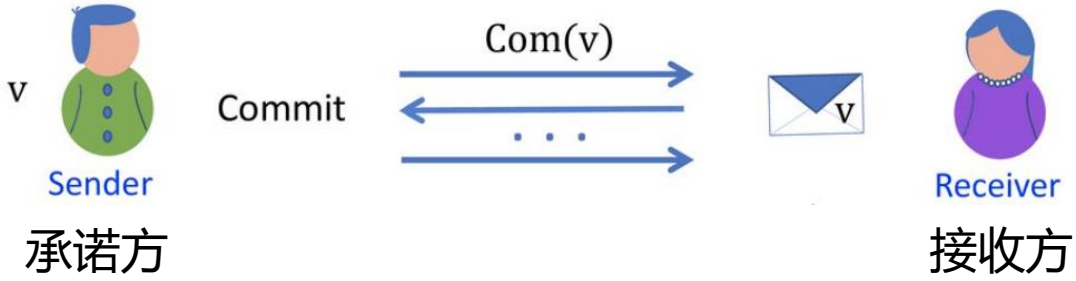
无法判定 *r* 取样的倾向

密码学特性2 – 隐秘性 (hiding)

应用

- **承诺 (commitment)**. 一个数字化的过程，类比为：
- 1）选取一个数字; 2) 将其封入信封; 3) 将信封展示给所有人.
- 这样意味着对一个数字进行了承诺，生成了一个信封。





承诺方　　接收方

密码学特性2 – 隐秘性 (hiding)

应用 —— **承诺** (commitment)

- 一个承诺协议方案应该包括两个算法
  - com := commit(msg, nonce)，承诺函数将信息（msg）和一个临时随机数（nonce）作为输入，输出就是一个“承诺 (commitment, shorten as com here)”。
  - verify(com, msg, nonce)，验证函数将某个承诺输出（com）、临时随机数（nonce）及信息（msg）作为输入，如果com == commit(msg, nonce)，则返回“真”（true）；反之则返回“假”（false）。

Nonce: 密码学术语，该值只能使用一次

密码学特性2 – 隐秘性 (hiding)

使用 **承诺** 协议，要求满足以下两个安全特性

- 隐秘性：已知com，没有可行的方法找到 msg。
- 约束性：没有可行的办法找到两组(msg, nonce)和(msg',nonce')，msg≠msg'，而 commit(msg, nonce) == commit(msg',nonce')。

为了使用承诺的方案，需要进行以下步骤

- 承诺方生成一个随机数: nonce;
- 将其与信息msg一同放入承诺函数commit() ---> com (承诺) = commit(msg, nonce);
- 承诺方向所有人公开明文和随机数< nonce, msg>;
- 验证方可以根据< nonce, msg> 再次生成承诺从而验证承诺方给的 com 的真假。

密码学特性2 – 隐秘性 (hiding)

使用承诺协议的时候，如何保证隐秘性和约束性?

首先如何 执行 承诺方案?

- 使用 hash function: commit (msg, nonce) := H (nonce || msg),
- where nonce could be a 256-bit long random number

通过代替来实现以下两个:

- 隐秘性：已知 H (nonce || msg)，没有可行的方法找到 msg。
- 约束性：没有可行的办法找到两组(msg, nonce)和(msg',nonce')，msg≠msg'，而 H (nonce || msg) == H (nonce' || msg').

此隐秘性，正是我们期待 hash function 要具备的隐秘性!
约束性：隐含在 哈希函数 的 collision-resistance 特性中。
如果一个 Hash Function 具有 碰撞阻力 与 隐秘性，我们称之为一个有效的承诺！

密码学特性3 – 谜题友好（puzzle friendliness）

Definition:

– Puzzle friendliness. A hash function H is said to be puzzle-friendly if for every possible n-bit output value y, if k is chosen from a distribution with *high entropy*, then it is infeasible to find *x* such that H(k ‖ x) = y in time significantly less than 2^n.

谜题友好. 如果对于任意 n 位输出值 y，假定 k 选自高阶最小熵分布，如果无法找到一个可行的方法，在比 2^n 小很多的时间内找到 x，保证 H(k ‖ x) = y 成立，那么我们称 哈希函数H 为 *谜题友好*。

帮助理解：如果有一个人想找到y值所对应的输入，
假定在输入集合中，有一部分是非常随机的，
那么他将非常难以求得 y值 对应的输入。

密码学特性3

特性3 --谜题友好 -- 的应用

- 什么是 谜题搜索?

搜索谜题 搜索谜题构成：

- 一个哈希函数H。
- 从高阶最小熵分布选出的一个取值，id（我们称其为谜题ID）。
- 目标集合Y。

该谜题的解决方法为一个解，x，应该满足以下公式：

$$H(id \| x) \in Y$$

要求找到一个位于集合Y内的输出值

- Y 集合的大小决定了谜题难度
  - If Y集合的排列组合数==n，（n为谜题输出字符串的种类数），难度为0。
  - If Y集合的排列组合数==1，难度最大。

密码学特性3

特性3 --谜题友好 -- 的作用

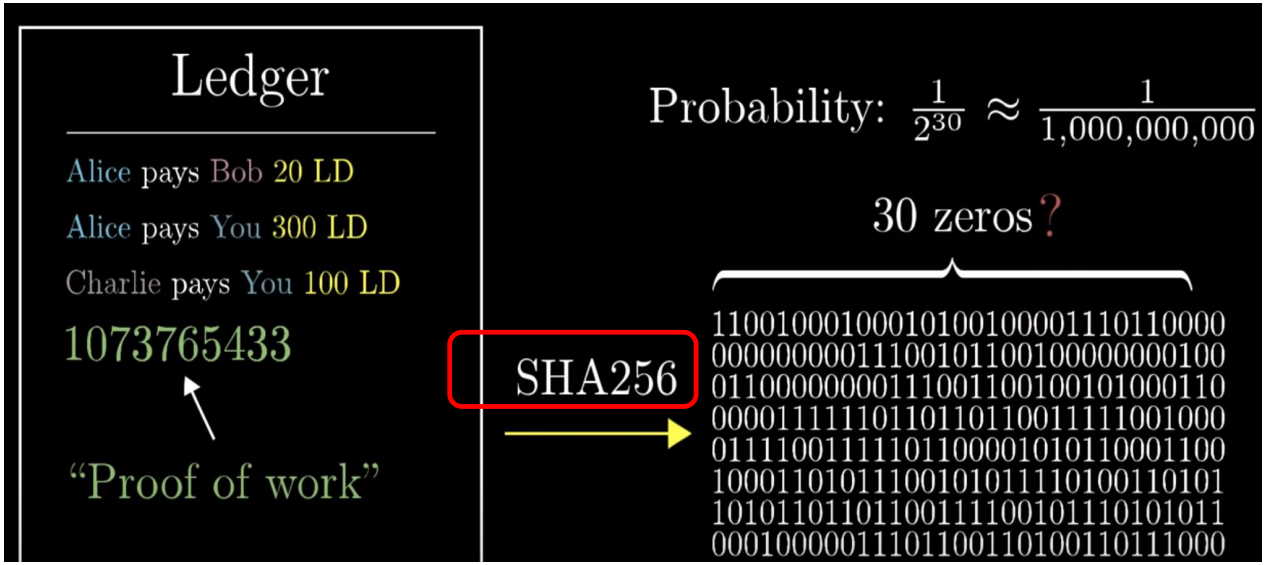
- 谜题搜索 —— Bitcoin's mining：挖矿就是"解数学难题"?
- 如果一个 H(id || x) 具备此特性，则 对于这个谜题没有一个解决策略，比只是随机地尝试 x 会更好。
- 那么，我们可以为 Bitcoin 设计一个谜题搜索，来保证所有参与者的公平性

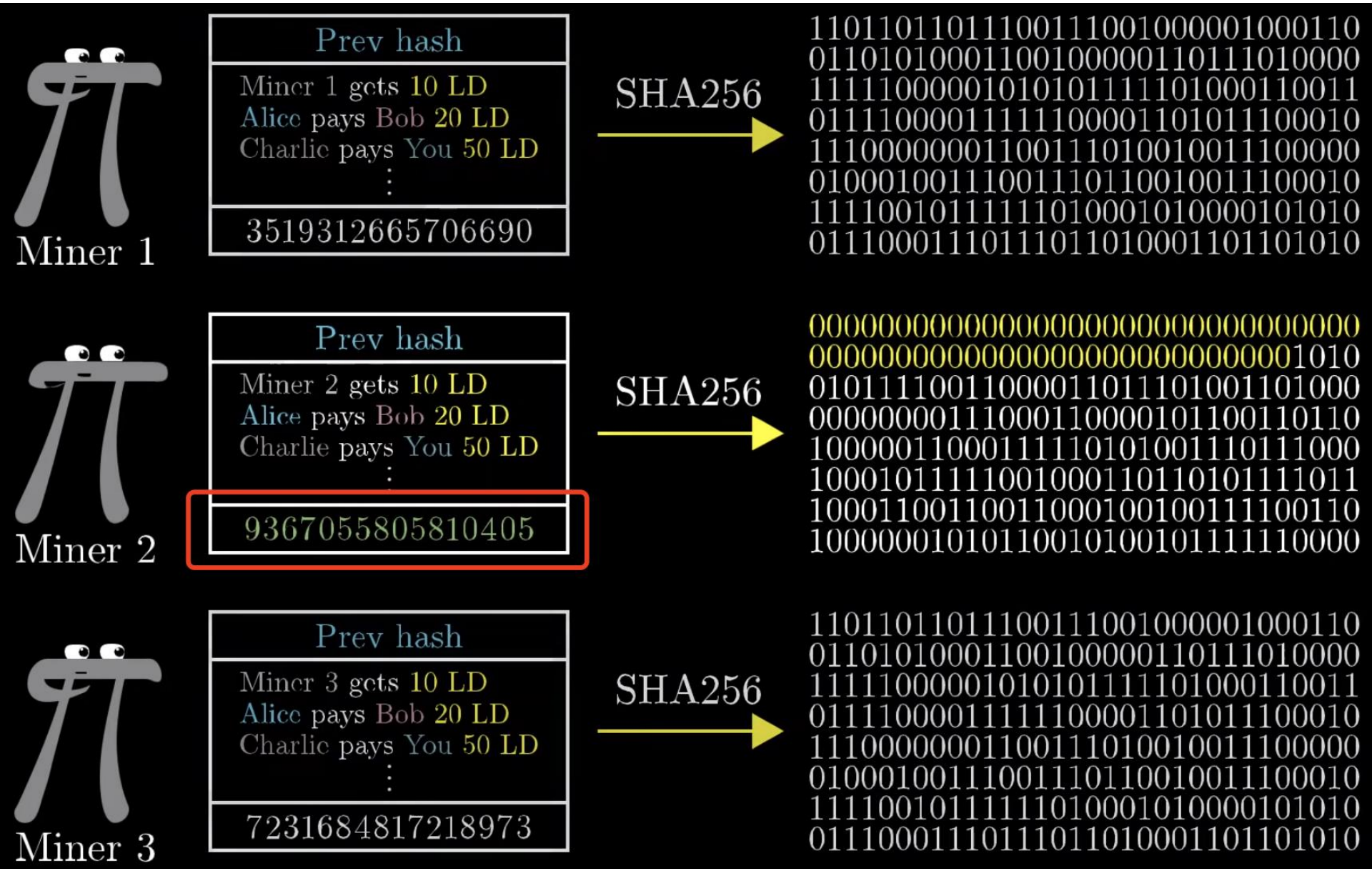
密码学特性3

特性3 --谜题友好 -- 的作用

- Mining 对每一个 miner 都是公平的：大家都不停地寻找一个合格的解 x —— nonce！

- H (header || TXs || nonce) < target




Miner 1
Prev hash
Miner 1 gets 10 LD
Alice pays Bob 20 LD
Charlie pays You 50 LD
⋮
3519312665706690
SHA256
11011011011100111001000001000110
01101010001100100000110111010000
11111000001010101111101000110011
01111000011111100001101011100010
11100000001100111010010011100000
01000100111001110110010011100010
11110010111111010001010000101010
01110001110111011010001101101010
Miner 2
Prev hash
Miner 2 gets 10 LD
Alice pays Bob 20 LD
Charlie pays You 50 LD
⋮
936705580581040 5
SHA256
00000000000000000000000000000000
00000000000000000000000000001010
01011110011000011011101001101000
00000000111000110000101100110110
10000011000111110101001110111000
10001011111001000110110101111011
10001100110011000100100111100110
10000001010110010100101111110000
Miner 3
Prev hash
Miner 3 gets 10 LD
Alice pays Bob 20 LD
Charlie pays You 50 LD
⋮
7231684817218973
SHA256
11011011011100111001000001000110
01101010001100100000110111010000
11111000001010101111101000110011
01111000011111100001101011100010
11100000001100111010010011100000
01000100111001110110010011100010
11110010111111010001010000101010
01110001110111011010001101101010

密码学特性3

Mining 的比喻

- H (header || TXs || nonce) < target
- 策略：狂轰滥炸，鸟枪法

nonce

Hash Function

解空间 Y

比特币密码学基础

PART 2: DIGITAL SIGNATURES

引言 —— 比特币用户有账户吗?

银行账户

- 中心化账户管理<用户ID，用户密码>

比特币没有账户

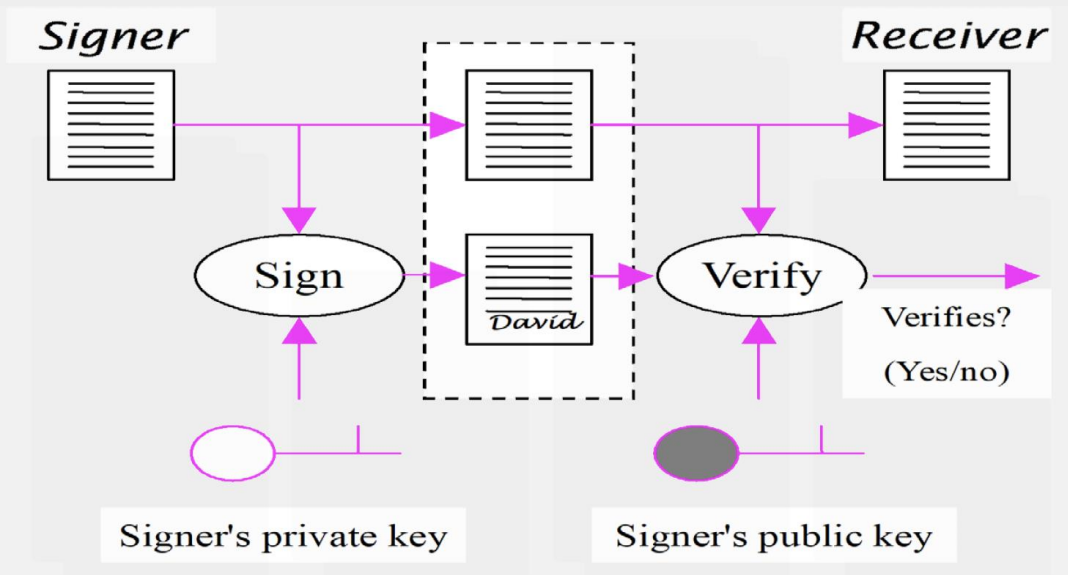
- 用户自己开账户 —— <public key, secret key>
- <pk, sk> 来源于 非对称加密
- pk：用户名称（地址）
- sk：用户密码（密码）
- pk是根据sk生成的

加密货币中的公钥私钥

加密货币不加密

那么，公钥私钥有什么用?

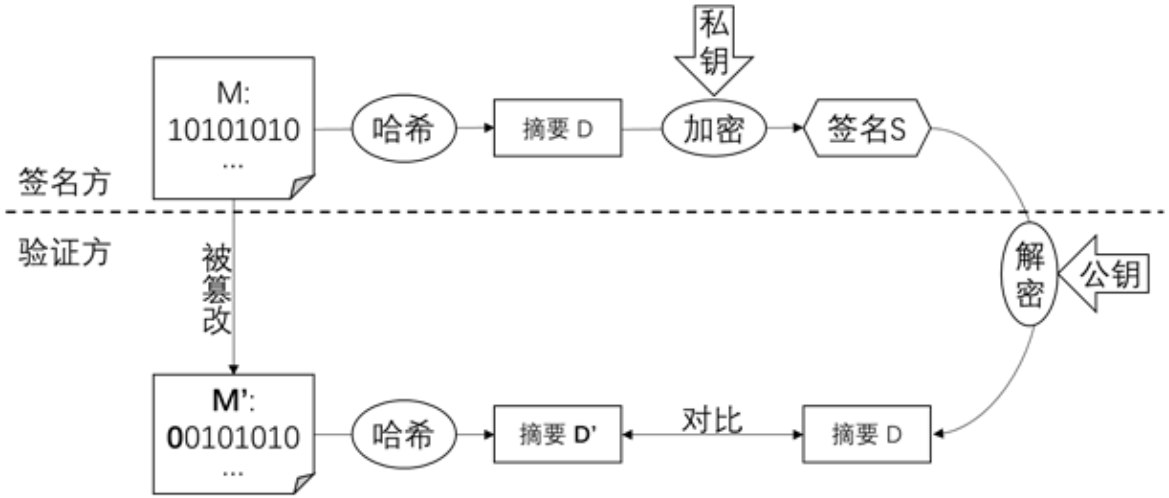
- 签名 Sign (message, sk) = Signature
- 验证Verify (message, Signature, pk) = message', 对比message和message', 判定 True / False



加密货币中的公钥私钥

如果直接对原有数据进行加密，这个过程的开销是十分巨大的。对此，通常采用了对原本信息的哈希值进行加密的方式来进行签名。

对于检验方而言，通过检查数字签名S解密得到的摘要D和接收到的消息M'的摘要D'是否相同，便可以确定消息M的完整性和正确性。



简单的数字签名和检验流程

数字签名方案

由三个算法构成

- (sk, pk) := generateKeys (keysize)
  - 把 keysize 作为输入，来产生一对公钥和私钥
  - 私钥 sk 被安全保存，并用来签名一段消息；
  - 公钥 pk 是人人都可以找到的，拿到它用来验证你的签名。

- sig := sign(sk, msg)， 签名过程
  - 把一段消息message 和私钥 sk 作为输入，输出是 签名sig

- isValid := verify(pk, message, sig)， 验证过程
  - 通过把一段消息和签名消息与公钥作为输入，
  - 如果返回是真，证明签名属实；否则，证明签名的消息为假。

数字签名方案

要求两个性质有效：

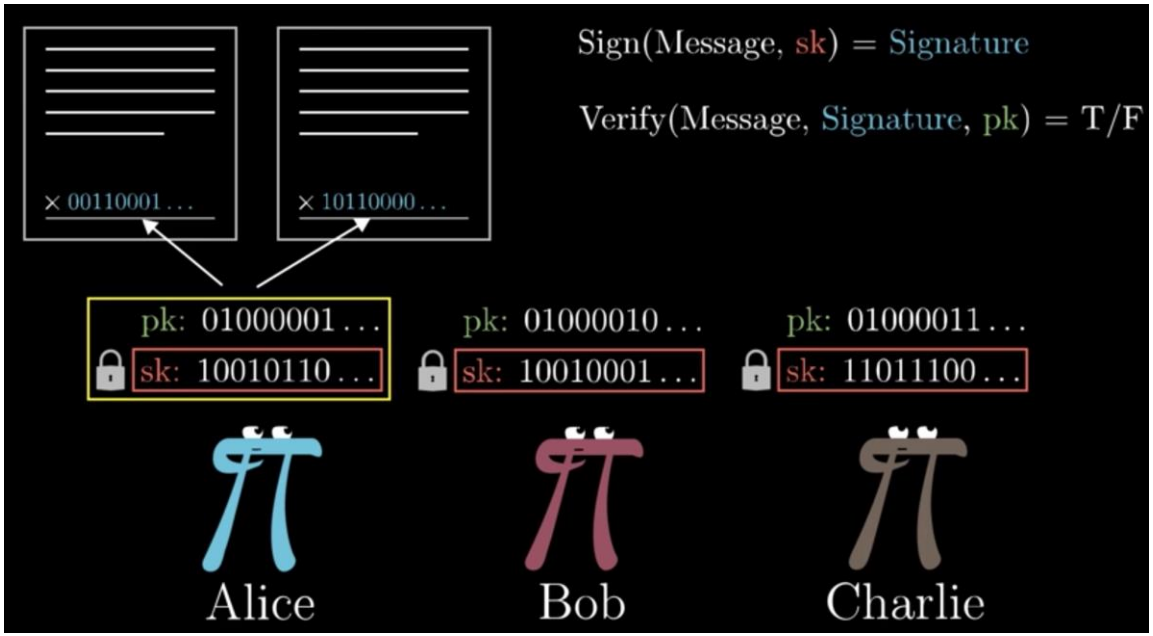
- 有效的签名可以通过验证：
  - verify(pk, message, sign(sk, message)) == true
- 签名不可伪造

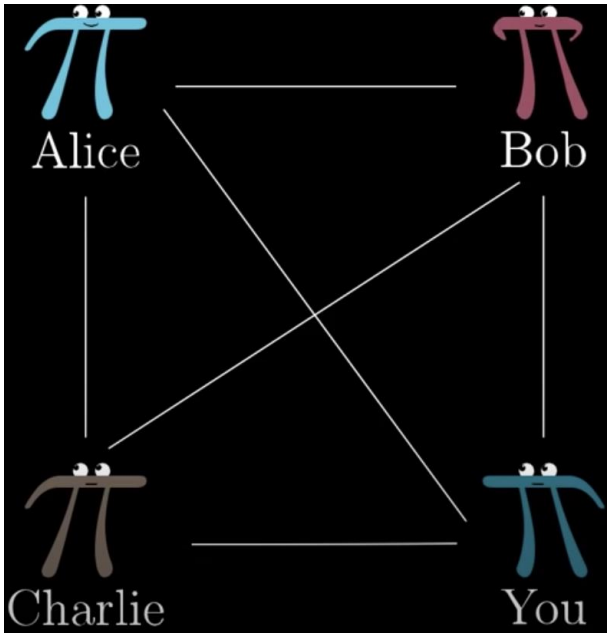
实践中

- 比特币使用的签名算法是随机的，需要良好随机源
- 对要签名的信息 message 大小有限：使用 Hash Func

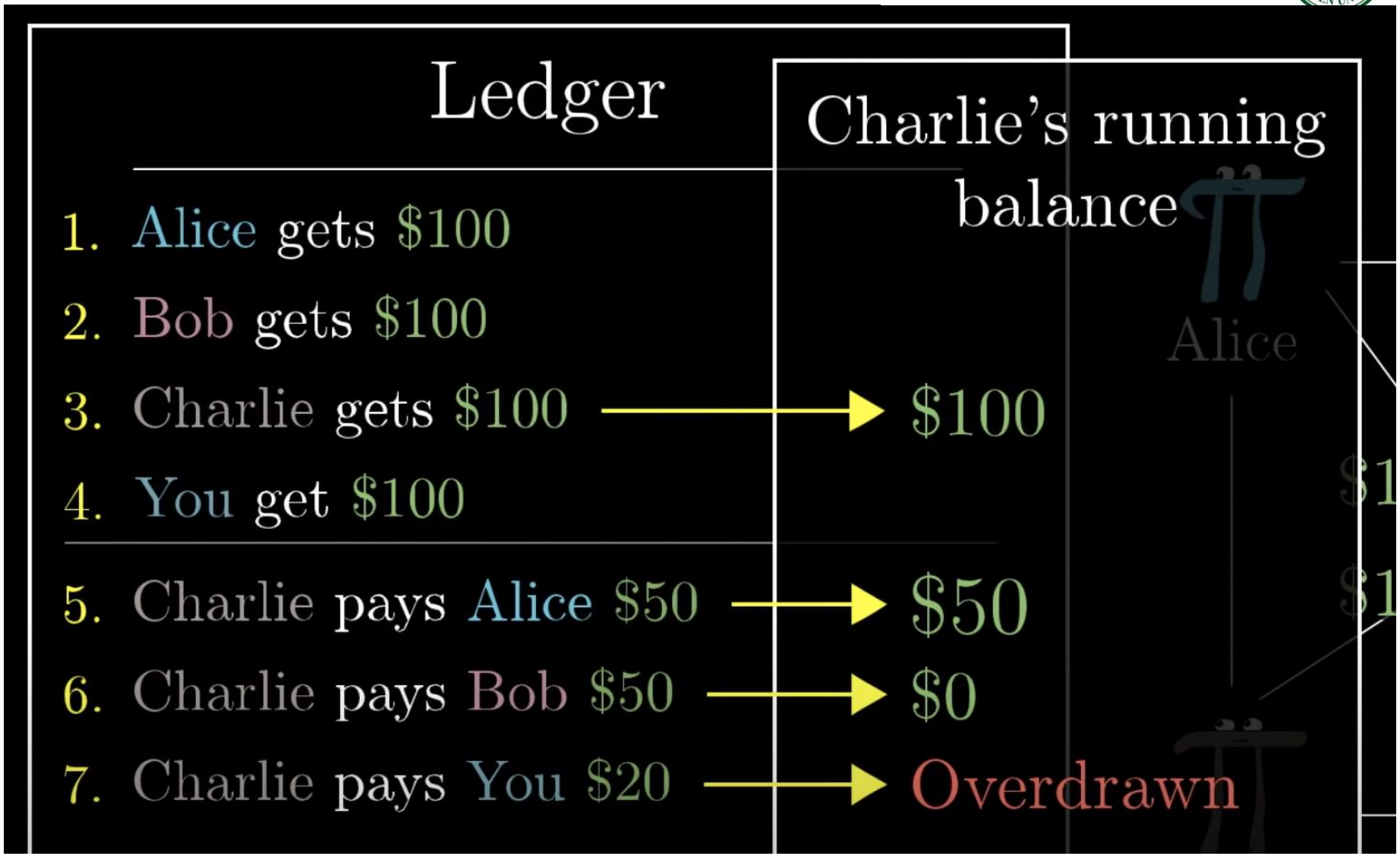
比特币中签名

- 每笔 Tx 的发起方用他的 sk 签名
- 其他人通过 Tx 发起方的 pk 验证 Tx 的合法性





比特币中通过签名验证，防止Double-Spending



比特币中：公钥即身份

为什么?

- Bitcoin 用户自己开账户 —— <pk, sk>
- 其他用户看到一个签名，并被一个 pk 验证了： pk 就可以代表一个人的身份
- 比特币中用户的身份： 地址 := Hash (pk || x)

去中心化身份管理

- 随时定制新的随机身份
  - new <pk, sk> = generateKeys (keysize)
- 匿名，一个人可以有多个 <pk, sk>
  - Not strictly anonymous: pattern analysis

Everyone Creates its own <sk, pk>

Any problems?

- Is it possible: two pairs of <sk, pk> are same?
- If yes, someone can steal other's coins, by keeping to try <sk, pk> pairs, until to find one

Theoretically, it sounds a good idea !

Don't worry, it is impossible, in practice.

- Probability of success is too low: ½^{256}

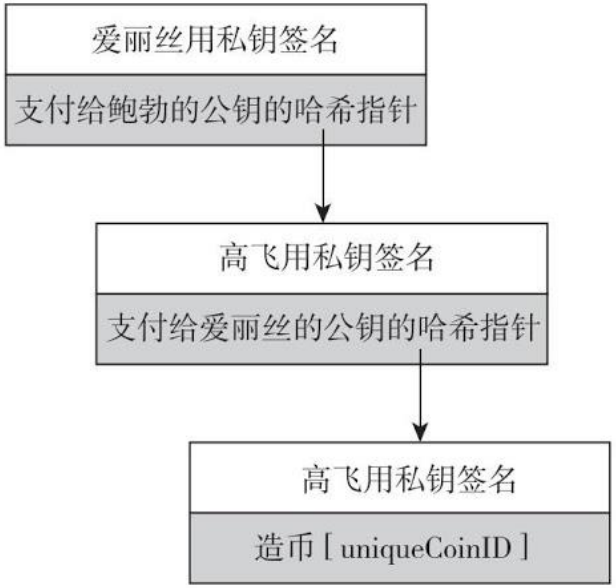
Randomness matters!

- A good *random source* determines *security of $*.

高飞币 (GoofyCoin): 最简单的加密货币

- 简单的规则1：用户可随时创建新币 CreateCoin[uniqueCoinID]
- 简单的规则2：用户可将币转给其他人，必须通过密码程序完成
  - 签署声明： “transfer a coin to X” (X is pk)

- 安全隐患?
  - 双重支付Double Spending: 一个人可以同时签署多份 transfer
  - 用户无限制地铸币
  - 尽管 coin 的转移机制与 Bitcoin 很接近，但是不安全，不能被使用



高飞币交易

From Cryptography to Cryptocurrency

财奴币 (ScroogeCoin)

- 基于高飞币，为了解决双重支付的问题，采用了更加复杂的数据结构。
- 主要概念：仅增账目（append-only ledger）
  - 保证写入账目的数据都会永久保留下来
  - 账目数据只能增加，不能删除和修改
- 建立如下的区块链

财奴币 第一种交易：CreateCoins

- 扩展：一次可以造多个币
- 只有财奴才可以铸币

交易ID：73		类型：造币
被创造的货币		
序号	数量	造币记录
0	3.2	0x…
1	1.4	0x…
2	7.1	0x…

虚拟货币ID 73（0）
虚拟货币ID 73（1）
虚拟货币ID 73（2）

财奴币 第二种交易：PayCoins

- 被消耗的币必须是之前的交易中创建的
- 被消耗的币没有在之前的某个交易中被消耗掉：non double spending
- 本次交易产生的币值量==消耗的币值量：只有财奴才可以创建新币
- 本次Tx 被消耗的所有币都有其所有者的有效签署

交易 ID：73		类型：付币
消耗的虚拟货币 ID： 68（1），42（0），72（3）		
被创造的货币		
序号	数量	造币记录
0	3.2	0x…
1	1.4	0x…
2	7.1	0x…
签名		

From Cryptography to Cryptocurrency

财奴币 缺陷

- 权利太大
- 可以停止支持其他用户
- 财奴自己想要多少就创建多少，从而失去更新系统的动力

去财奴化 —— 去中心化 的加密货币 的要求

- 一个公开的 Blockchain
- 所有用户必须一致同意哪些 Tx 有效
- 去中心化的方式分配交易与货币的 ID
- 新币的铸造也通过 去中心化的方式