BTC 起源

https://img.learnblockchain.cn/pdf/2024/Intro-to-cypherpunk-and-bitcoin.pdf

Cypherpunk

- 星系可以持久保存，但面临暴露更多信息
- Ref
    - https://cdn.nakamotoinstitute.org/docs/cypherpunk-manifesto.txt
    - Web3 101 密码朋克的博客 去看

密码学

- 对称加密的困难
    - 密钥递送难题，
    - 两端同一个私钥
- 非对称加密（公钥密码学）的概念
    - Diffie, Hellman, 1976
    - 两端不用的私钥
- 第一种公开的非对称加密算法
    - RSA，1977
- 公钥数字签名的安全性概念，1988
- 公开的算法，但是要递送密钥
- Diffie Hellman Key Exchange
    - Large prime numbers
    - Reminder
- RSA
- 公钥数字签名的安全性概念

- 通信隐私权
    - https://chaum.com/wp-content/uploads/2021/12/chaum-mix.pdf
- 交易隐私权
    - https://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/2009/Chaum.BlindSigForPayment.1982.PDF
- Ref
    - https://www.btcstudy.org/2021/09/10/trusted-third-parties/
    - https://www.btcstudy.org/2020/08/23/the-origins-of-money-by-nick-szabo/
    - https://www.btcstudy.org/2021/10/27/money-blockchains-and-social-scalability-echo-edition/
- Bit-Gold
- B-money
    - 质押货币成为服务器（PoS 概念）
    - 以拍卖发行货币
        - 服务器给出增发数量
        - 人们承诺用一定的计算量竞拍一定的数额
- RPOW

- GPG
- https://www.btcstudy.org/2021/11/23/bitcoin-and-me-by-Hal-Finney/

总结

1. 稀缺性
    a. PoW
    b.
2. 抗重复花费
    a. PBFT 注册表
        i. PBFT 是一种算法，用于在一个分布式系统中让多个计算机达成一致，即使有一些计算机出现故障或恶意行为。它能确保系统中的所有正常计算机都同意一个操作，即使有一些计算机不可靠或恶意。
        ii. 假设有 ABCD 4 个节点，然后 A 和 B 同时算出了区块 hash，然后都会将数据同步给其它 3 个节点，然后 3 个节点会验证收到的数据是否合法。如果合法，那就暂且将 2 个区块都当作正确的暂存，等到下一个区块计算出来，也就是假设 A 和 B 分别产出了区块 89999 和 89999'，然后后续的区块分别在 89999 和 89999' 上新增 90000 和 90000' 看谁先出来就用谁的。
    b. HSM-based Authority
3. 抗通胀
    a. None
4. 匿名性
    a. 公钥密码学

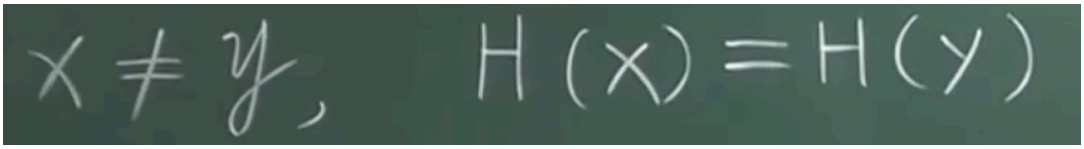
白皮书与比特币

- https://www.btcstudy.org/2022/11/23/bitcoin-paper-errata-and-details/
- https://www.btcstudy.org/2021/12/16/bitcoins-academic-pedigree-modified-edition/
- https://www.btcstudy.org/2021/09/29/the-incomplete-history-of-bitcoin-development/
- https://www.btcstudy.org/2021/09/29/soft-fork-activation-by-optech/

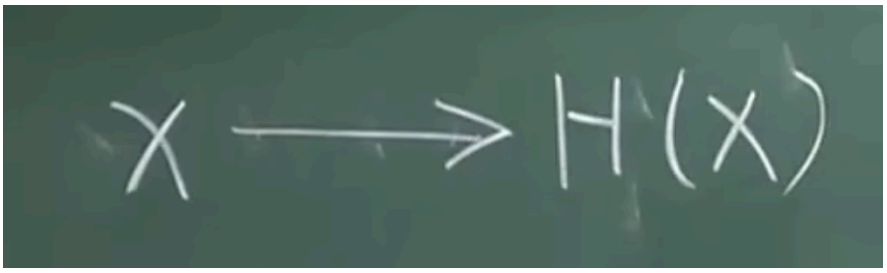
区块链技术与应用 - note

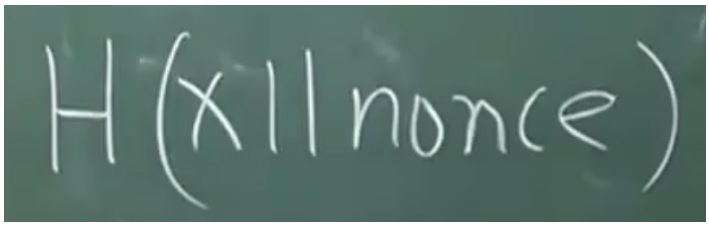
- 北京大学计算机系肖臻
- https://www.bilibili.com/video/BV1Vt411X7JF/?spm_id_from=333.337.search-card.all.click&vd_source=57674ecc8bddfd5c3a912056417dbb5c

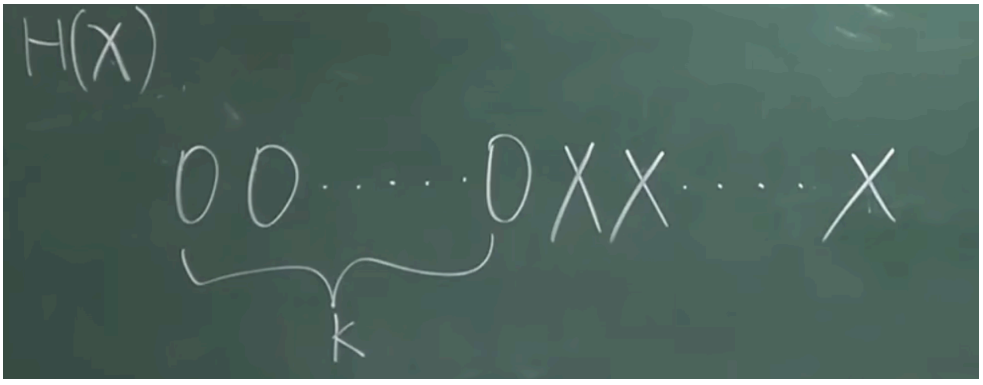
Crypto-Currency 密码学原理

- Cryptographic hash function
  - Collision resistance（抗碰撞性）
    - Hash 碰撞
    - 

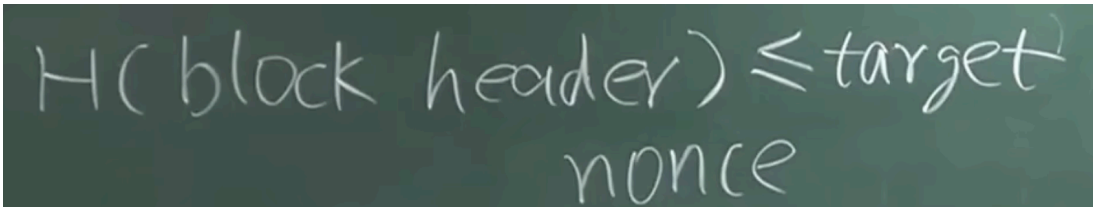
    - 2^256
    - 又称Collision free (碰撞会存在的，并不是完全不存在，但是目前没有很好的方法，只能用暴力拆解法)
    - 需要遍历 x or y 然后得出那个 hash 相等
    - Brute -force
    - 工作量太大了
    - H(m)
      - m = message
      - 如果有人修改这个 m 的内容，他的 hash 值就会发生变化
      - Collision resistance 的作用是你无法找到另外的 m 的信息 hash 出来的结果与之前的 hash 相等。
      -
    - Message Digest
      - 通常指的是一个哈希函数生成的固定长度的输出，也被称为“消息摘要”或“哈希值”。消息摘要是对输入数据进行哈希运算后得到的结果，具有以下几个重要特性：
        - 固定长度：无论输入数据的长度是多少，生成的消息摘要都是固定长度的。例如，SHA-256 总是生成 256 位（32 字节）的摘要。
        - 唯一性：不同的输入数据几乎总会生成不同的消息摘要。如果两个不同的输入数据生成相同的消息摘要，这种情况称为哈希碰撞。好的哈希函数应该尽量避免碰撞。
        - 不可逆性：从消息摘要无法反推出原始输入数据。这意味着哈希函数是单向函数，确保数据的安全性。
        - 敏感性：任何微小的输入数据变化都会导致消息摘要发生巨大变化。这种特性确保了即使是非常小的输入修改，也会显著改变输出哈希值。

    - 没有那个 hash 函数无法从理论证明他是 Collision resistance，只能靠时间的经验
    - 例如 MD5 这个就可以认为的制造 hash 碰撞了
- Hiding
    - Hash 函数计算过程是单项不可逆转的
    - 

    - 无法反向获取，除非用蛮力的方法遍历 x 碰撞 hash 与 H(x) 进行对比，如果一致就找到了 x 的值
    - Digital commitment
        - 又叫做 digital equivalent of a <u>sealed envelope</u>
            - 预测结果不能够提前公开
                - 怕提前公开的结果干扰到
            - 把预测结果密封起来，等结果公布，再把预测结果公布进行对比
            - 但是再 digital 的世界里，把 预测结果 x 先算出一个 hash value，然后公布 hash value，等实际结果得出后，把之前公布的预测结果 x 再公布出去，验证 hash value 是否一致。
            - 如果结果范围不够大，很容易找到规则无法完成随机的效果所以使用一个随机数 nonce 与 x 预测结果进行拼接，这样不公布出 nonce 无法从规律中找到 x 从而产生随机。
            - 

- Puzzle friendly（这是密码学以外，btc 所以要的）
    - 

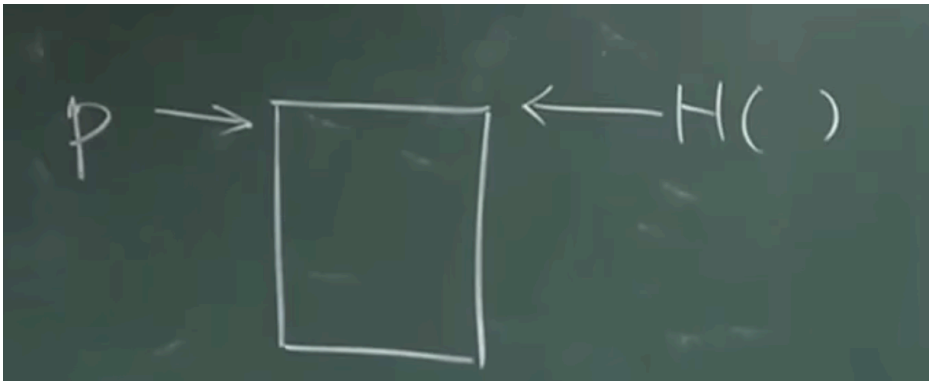
    - 使其满足特定的特性和要求，以确保系统的安全性和效率。

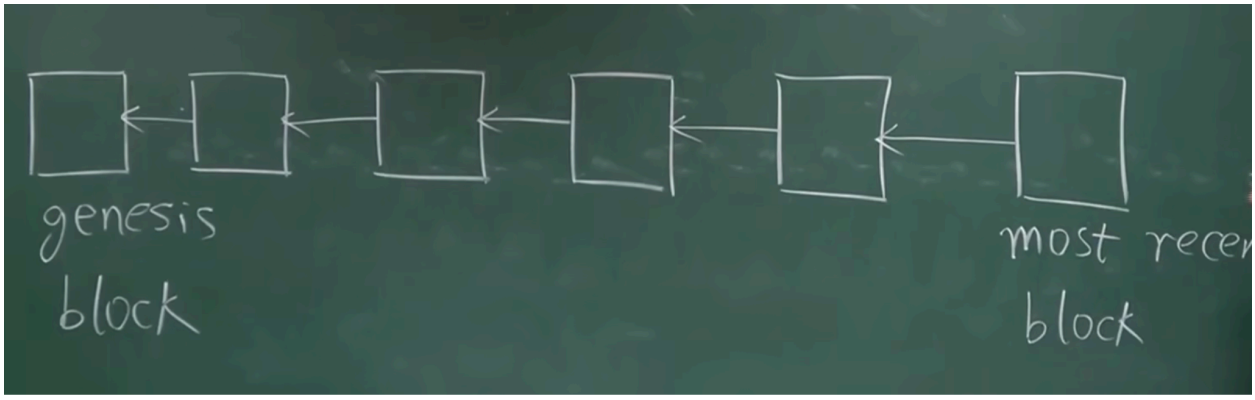
        - 比特币矿工需要找到一个称为 nonce 的值，使得将区块头（包含区块数据和 nonce）经过 SHA-256 哈希运算后得到的哈希值小于当前网络目标值。这个目标值根据网络难度调整，每大约 10 分钟调整一次，以确保区块生成时间稳定。
        - 没有捷径，必须要一个一个去试
            - 保持系统稳定：通过调整难度，保持区块生成时间恒定，避免网络过载或处理速度过慢。
            - 保障安全性：确保谜题的解答难度足够高，以防止恶意攻击者轻易篡改区块链数据。
            - 鼓励参与：公平的计算竞争，激励更多矿工参与，提升系统的去中心化程度和整体安全性
            - Difficult to solve，but easy to verify
        - 其实 btc 挖矿就是找这个 nonce 的值
        - 

        - 没有捷径，需要一点儿一点儿的试
        - 这个叫 Proof of work
        - 如何验证计算出得 hash 符不符合条件只需要和 target 进行一次比较就行了
        - 所以挖矿很难验证很容易
        - Difficult to solve，but easy to verify
        - SHA-256 btc
        - Secure hash algorithm
        - 与 collision resistance 有点儿像，其实不完全一样
- (Public key, private key) 这一对就生成了 btc 的帐户
    - 这个加密体系叫做 asymmetric encryption algorithm
        - 加密用的是 public key
            - 不用保密
        - 解密用的是 private key
            - 需要保密
        - 如果我想给你传递一个信息，我用你的 public key 加密传给你，你用你的私钥解密得到原来的信息
        - 解决了对称加密体系当中密钥分发不方便的问题
    - 加密解密用同样的 Encryption key 对称加密体系
        - 弱点不方便的传递 encryption key
    - Sign message（btc）
        - 发布交易，先用我自己的 private key 进行签名
        - 然后链上用 public key 进行验证这个信息的正确性
    - A good source of randomness
        - 需要好的随机源
        - 防止随机源被反查从而破解

BTC 数据结构

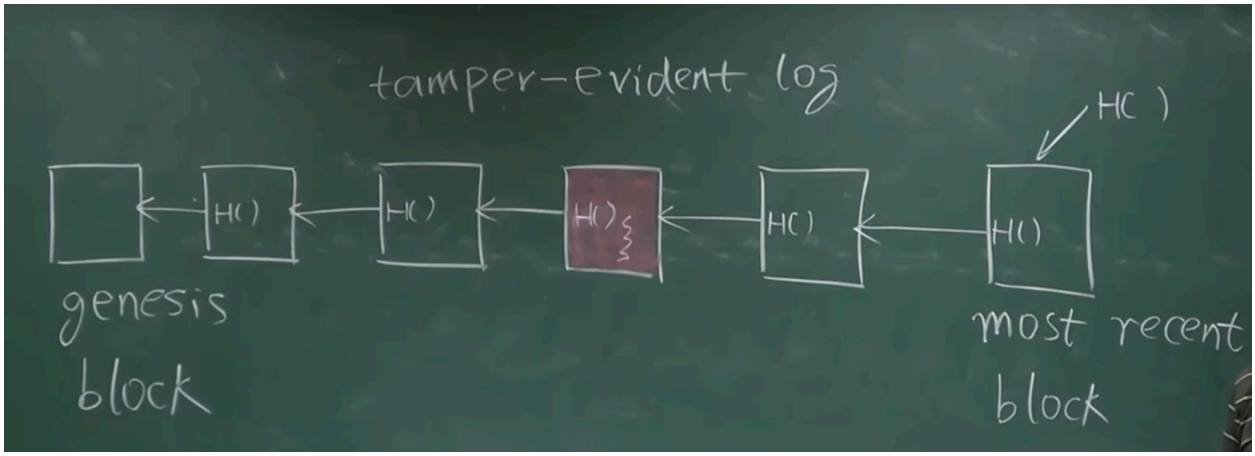
Hash pointers

- 数据传输的时候没有指针只有 Hash
- 可以把 Hash 当作成指针
- 全节点会把这些区块存在 key value 数据库里 levelDB
- 通过 block hash 找到对应的 value
- 

    - H() 代表 hash pointer
    - 通过一个 hash pointer 找到对应的区块函数
- 区块链是 linked list 结构用 Hash pointer 代替普通指针
- Block chain is a linked list using hash pointers.
- 

    - Genesis block - 创世记块
    - Most recent block

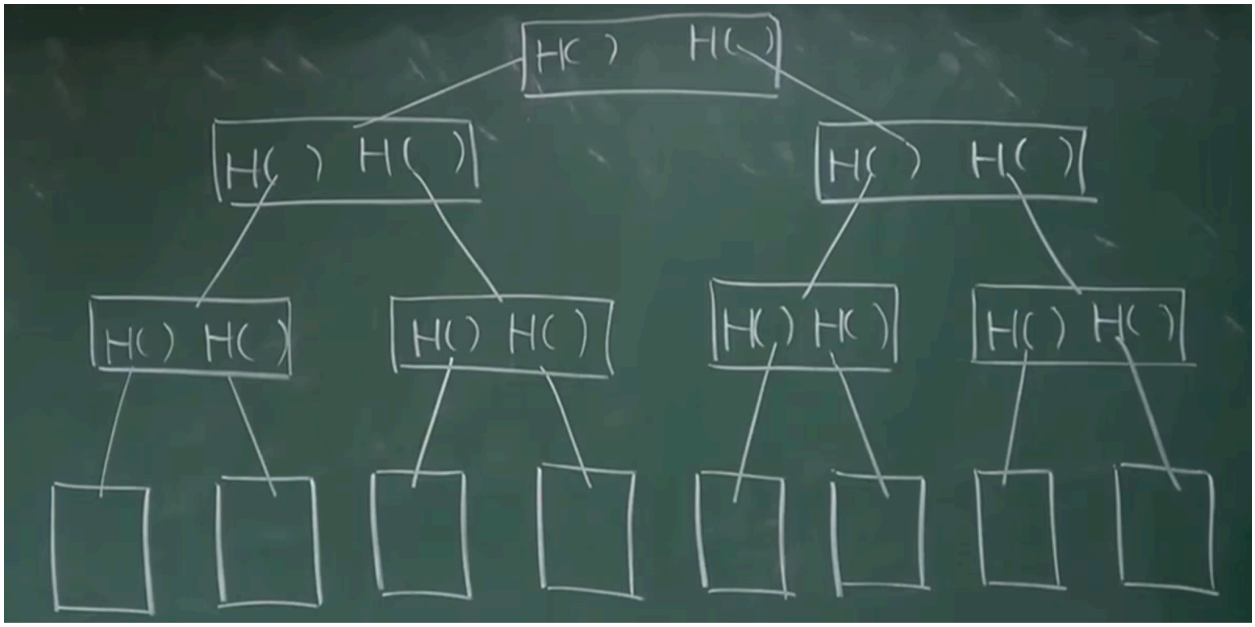
-



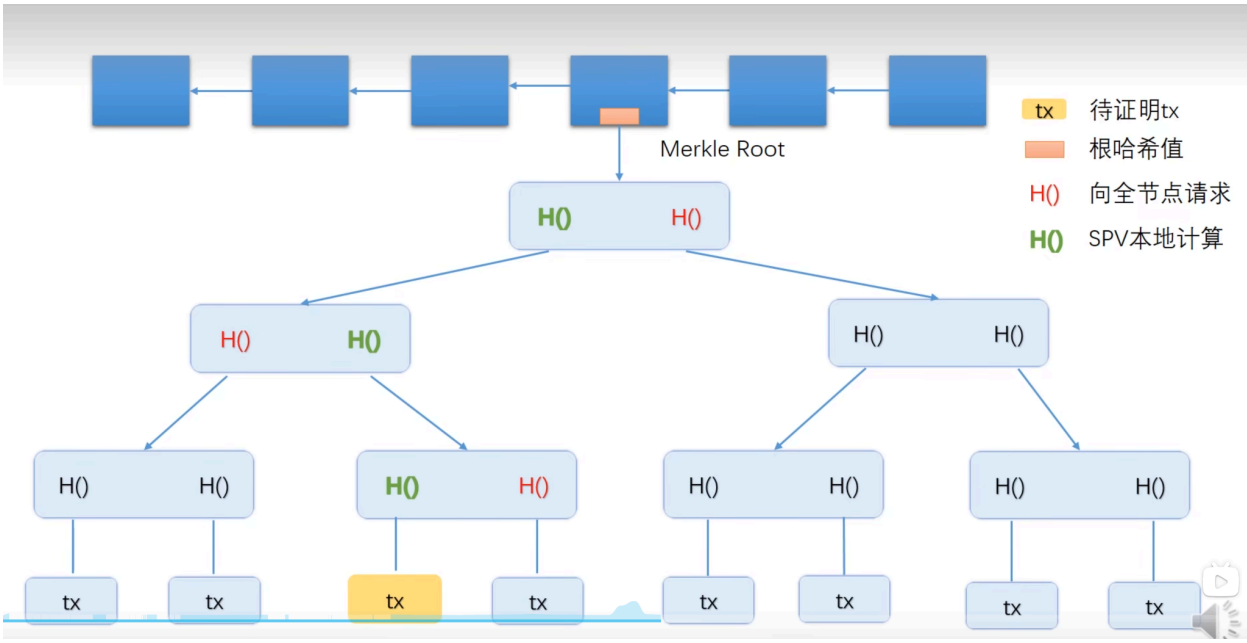
  - Tamper-evident log
  - 这个作用的好处我只用对最后的 hash 进行校验
  - 无法对任何部位进行修改，如果我对那个部位进行改动我都需要对所有后面的 hash 进行变动，因为每个新的 hash pointer 都是我前一个区块的 hash pointer 加上区块中的所有信息
  - 普通链表可以随意改变其中的元素但是不会对其他产生影响，但是区块链会
  - 可以很好的防止作恶区块

Merkle tree

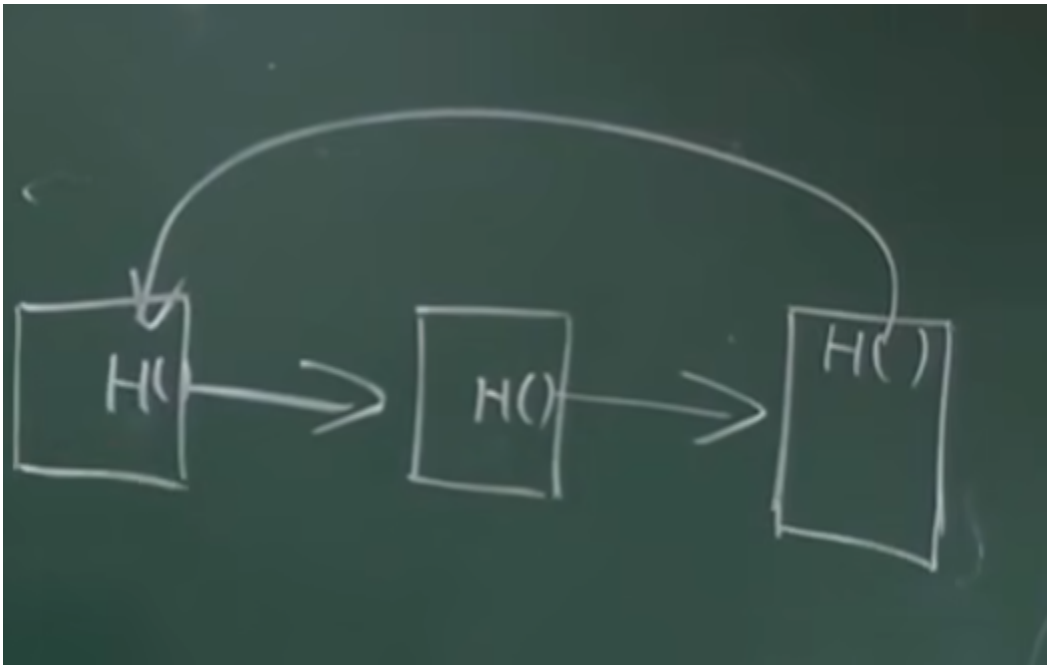
- Binary tree 一样只是用 hash pointers 代替了普通指针
-



  - 最下面一层是 Data blocks
  - 上面的都是 hash pointers
  - 最上面的一层是 root hash
  - 只要我检测 root hash 我就能检测出任何链条的修改
  - 每个数据块都是 tx

- Block header
  - Root hash
  - 没有交易只有 root hash
- Block body
  - 有交易列表的
- Merkle tree 的作用
  - 时间复杂度 logn
  - 提供 Merkle proof
  - 验证某个特定交易是否在区块中，必须生成该交易的梅克尔证明。
  - 使用给定的交易哈希值和梅克尔路径中的哈希值，逐步计算父节点的哈希值，直到得到梅克尔根。
  - 

  - 一个区块的 Merkle tree
  - 从全节点请求，只需要从全节点请求红色区域的
  - 绿色的 hash 值可以本地计算得出
  - Spv simple payment verification
    - 下载区块头：SPV 客户端只下载区块链的区块头，而不下载完整的区块数据。区块头包含区块的基本信息，例如上一个区块的哈希值、当前区块的哈希值、时间戳、难度目标和梅克尔根（Merkle Root）。
    - 请求交易验证：当用户需要验证某笔交易时，SPV 客户端会向全节点请求包含该交易的区块头和相关的梅克尔树路径（Merkle Path）。梅克尔树路径是一系列哈希值，用于从特定交易生成梅克尔根。
    - 验证梅克尔树路径：SPV 客户端使用交易数据和收到的梅克尔树路径，逐步计算梅克尔树的各层哈希值，直到生成梅克尔根。然后将计算得到的梅克尔根与区块头中的梅克尔根进行对比。如果两者匹配，则表明该交易确实包含在区块中。

        - 验证工作量证明：SPV 客户端还需要验证区块头中的工作量证明（Proof of Work）。这确保了区块链的完整性和安全性。具体步骤是通过计算区块头的哈希值，并检查其是否满足网络设定的难度目标。
        - 双重支付检查：为了防止双重支付，SPV 客户端通常会连接多个全节点，并且对比从不同节点收到的交易信息和区块头，以确保交易的唯一性和正确性。
    - 从 待验证的 tx 从下网上计算最后于 merkle root hash 是否一致
        - 最后 全节点的 block header hash 进行对比
    - 因为是 Collision resistance 的性质 Merkle proof 很难进行篡改
- 全节点 full node fully validating node
    - Block header
    - Block body
- 轻节点 light node
    - Block header
    - 如何轻节点如何去去 验证交易
- Proof of membership
- Proof of inclusion
- Proof of non-membership
    - big n
    - 没有更高效的方法
- Sorted Merkle tree
    - Btc 不需要这种做法
    - 如果我们对每个叶节点的 hash 从小到大的排序，就可以用 binnary search 的方法进行查找
    - 

    - 不能和 b+ tree 一样使用循环指针的原因是 使用了 Hash pointer，因为会递归依赖

梅克尔前缀树（Merkle Patricia Tree，简称 MPT）是一种结合了梅克尔树和 Patricia 树优点的数据结构。它可以高效地进行成员和非成员查询。MPT 在区块链（如以太坊）中被广泛使用，以存储和验证键值对数据。

Patricia 树：

- 一种压缩前缀树，用于存储关联数组。
- 每个节点代表一个字符串的公共前缀，从根节点到某个叶节点的路径表示一个键。

梅克尔树：

- 一种哈希树结构，其中每个节点的哈希值由其子节点的哈希值计算得到。
- 梅克尔根（Merkle Root）是树的根节点哈希值，代表整个数据集的哈希值。

梅克尔 **Patricia** 树：

- 结合了 Patricia 树和梅克尔树的特性，用于高效地存储和验证键值对。
- 每个节点存储键的部分前缀，并计算子节点的哈希值，用于验证数据完整性。

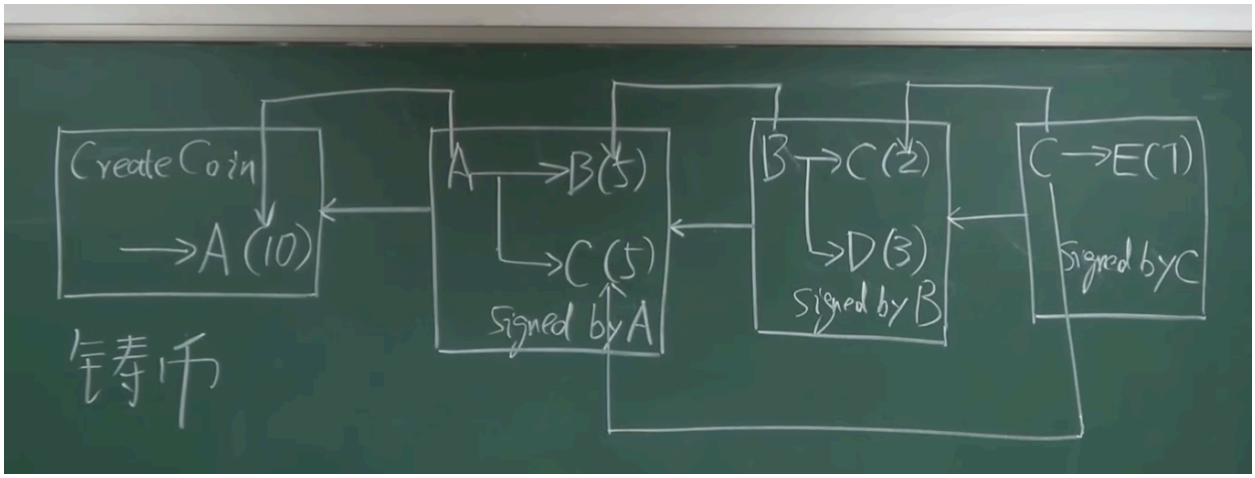
梅克尔前缀树（Merkle Patricia Tree，简称 MPT）是一种结合了梅克尔树和 Patricia 树优点的数据结构。它可以高效地进行成员和非成员查询。MPT 在区块链（如以太坊）中被广泛使用，以存储和验证键值对数据。

https://yangchenglong11.github.io/2018/03/16/merkle-patricia-tree/

- ETH 使用的方法

BTC 协议

- 数字货币主要防范的问题就是
    - Double spending attack
    - 区块链就是解决这个问题

-


    - 两种 hash pointers
        - 一种区块链接
        - 指向币的来源
            - 为了防范 double spending
    - 节点只需要回溯一下这个钱的来源
        - 是否已经花费出去
    - 一笔交易需要
        - Input 的签名
            - 签名是用私钥签的
        - Output public key
        - Input 需要知道 out public key
        - Input 的 public key 用于验证
    - 铸币交易
        - Coinbase tx
- Block
    - Block header
        - Version
        - Hash of previous block header
        - Merkle root hash
            - 确保 transaction list 无法被篡改
        - Target 挖矿的难度目标 target
            - nBits
        - Nonce
    - Block body
        - Transaction list
- Distributed consensus 分布式共识
    - 为什么 BTC 系统可以绕过分布式共识中的不可能理论？
        - 严格意义上来说 btc 并没有解决，例如分叉攻击
    - Distributed hash table
    - FLP impossibility result
        - 在一个异步的系统里（asynchronous）
        - 即使一个成员有问题的也不会存在共识 faulty

- CAP Theorem
  - 以下三个只能满足两个
  - Consistency 一致性
  - Availability 可用性
  - Partition tolerance 分区容错
- Paxos 关系不大与 btc 的实际运用
  - 能保持 consistency 但是客观存在无法达成共识
  - Paxos 算法通过提议者（Proposer）提出提案，接受者（Acceptor）进行投票，并最终让学习者（Learner）确认结果，从而在分布式系统中多个节点达成一致，即使在存在部分节点故障或网络分区的情况下，确保系统的一致性和可靠性

  假设我们有三个节点 A、B 和 C，它们需要在一个值上达成一致。

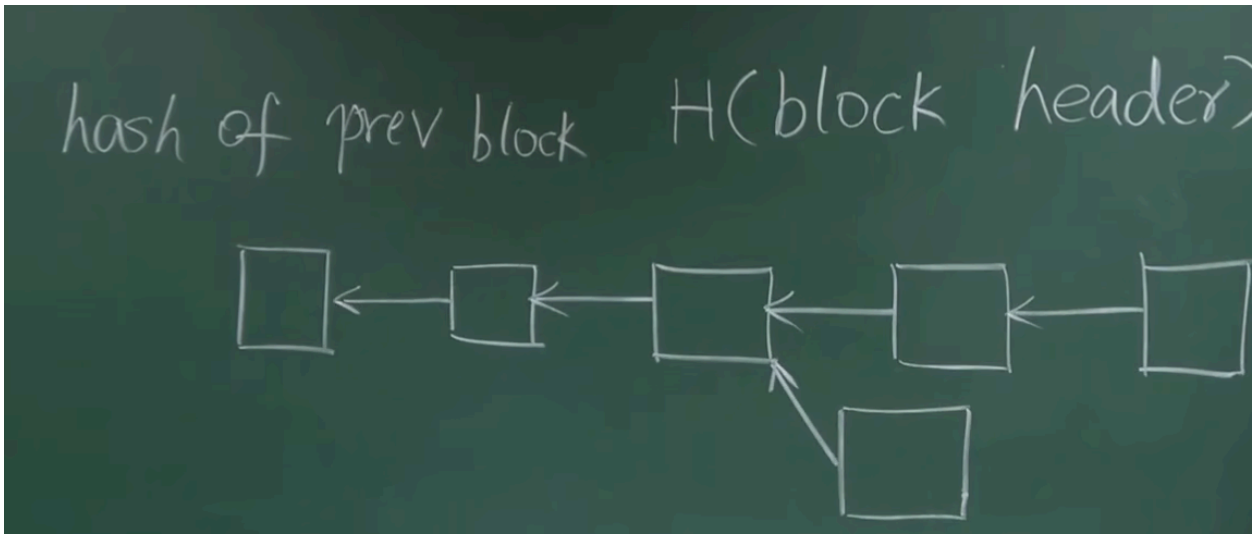
  1. 准备阶段：
     - 提议者 P1 选择一个提案编号（例如 1），向 A、B、C 发送准备请求。
     - 接受者 A、B、C 收到请求后，如果提案编号大于它们之前响应的编号，则承诺不再接受编号小于 1 的提案，并回复确认。
  2. 接受阶段：
     - 提议者 P1 收到大多数（例如 2/3）接受者的确认后，选择一个值（例如 v1）作为提案，向 A、B、C 发送接受请求。
     - 接受者 A、B、C 收到请求后，如果提案编号不小于它们的承诺编号，则接受该提案并记录下来。
  3. 达成一致：
     - 接受者将已接受的提案通知学习者（Learner），学习者记录最终决定的值 v1。

  通过这两个阶段，Paxos 算法确保了在分布式系统中，即使部分节点故障或网络不可靠，仍能一致地选择一个值。

- 补充
  - 发送信息，需要 output public key 进行加密，output 收到钱后需要用自己的私钥解密

Consensu in BitCoin

- POS
  - Membership
  - Hyperledger
  - Fabric
  - Sybil attack
    - 女巫攻击
- POW
  - 计算力来投票

- H(block header) LE target
    - Nonce 随机数
        - 4 bytes 4 * 8 = 32 bit
        - nBits
            - 检查是否
- 

    - 最长合法链上 longest valid chain
    - 选最长的合法链
    - Forking attack
- 当两个矿工同时算出一个 block 都是最长合法链我们该接受那个
    - 接受区块是按照最先受到的
    - 知道谁先往下扩展，谁就成为合法链
    - 没有成为的叫 Orphan block
- 为什么大家要竞争记账权
    - Block reward 出块奖励
    - Coinbase transaction
        - 是 BTC 发行新的 btc 的唯一方法
        - 开始是 50 btc 每 21w blocks 减半
- Puzzle friendly
    - 靠算力来投票
    - 这个性质保证了 H(block header) LE target 没有捷径
- Hash rate
- 如何避免 sybil attack
    - 投票的权重没有改变
    - Pow 算力

BTC 实现

Transaction-based ledger

- UXTO: Unspent Transaction Output
    - 为了检测 double spending
- Total inputs = total outputs
- Transaction Gas fee
    - 为了让矿工打包，和打包的优先级
    - 避免自私区块
- 21w block half life
- 缺点需要说明币的来源
    - 每一次转账都需要检测这个 utxo 的合法性
- 2016 blocks 调整难度

Account-based ledger

- ETH
- 不需要证明来源

Block Example

Block #529709

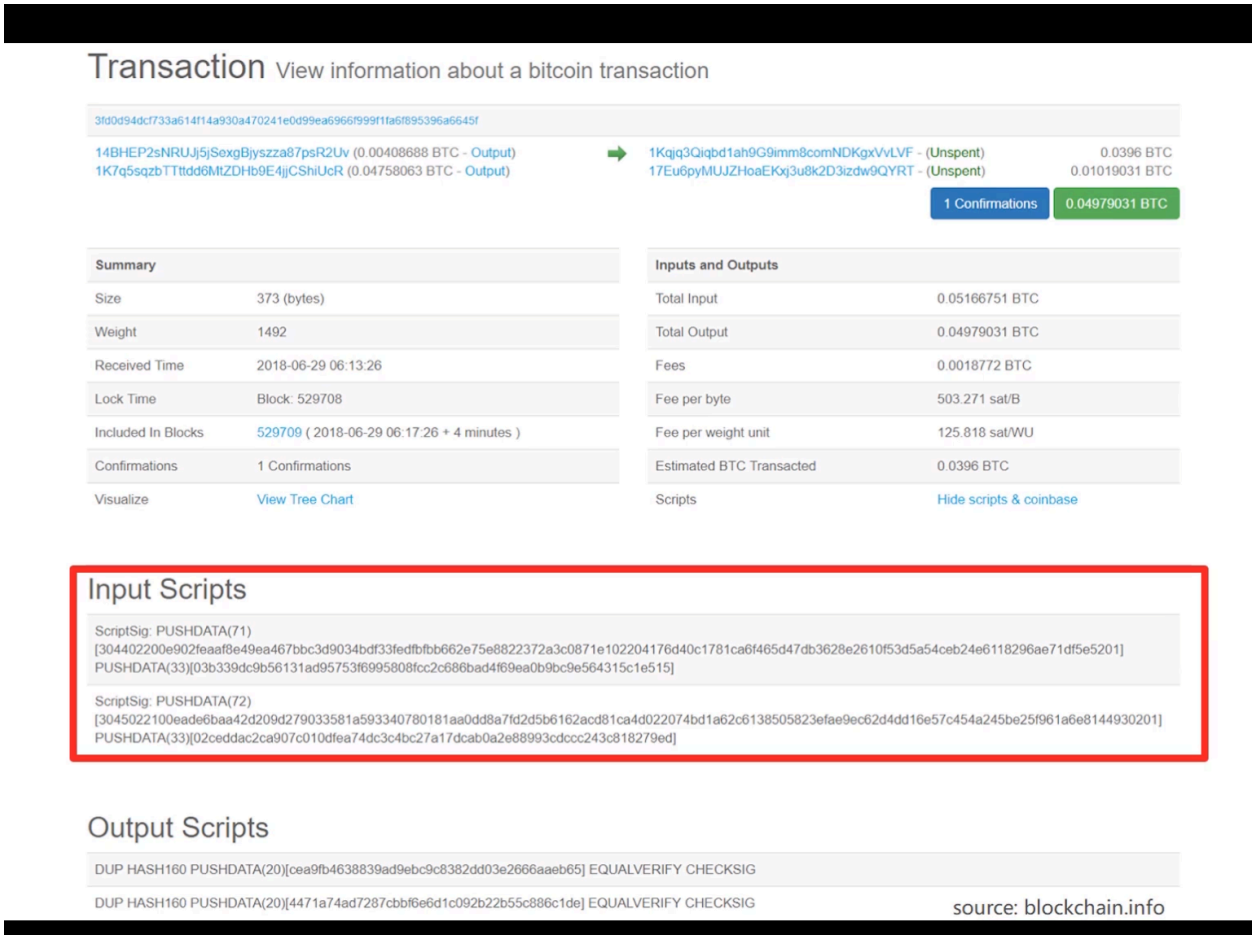
Summary	
Number Of Transactions	686
Output Total	4,220.46616378 BTC
Estimated Transaction Volume	651.93844862 BTC
Transaction Fees	0.12458867 BTC
Height	529709 (Main Chain)
Timestamp	2018-06-29 06:17:26
Received Time	2018-06-29 06:17:26
Relayed By	BTC.com
Difficulty	5,077,499,034,879.02
Bits	389508950
Size	333.53 kB
Weight	1160.618 kWU
Version	0x20000000
Nonce	3897564446
Block Reward	12.5 BTC

Hashes	
Hash	0000000000000000030f1531dbfa069037188c5048b23f0cb979ce8728ed8c5
Previous Block	0000000000000000000841c59a4679d6e707152f24f5b195b66b47397d65175e
Next Block(s)	
Merkle Root	6f73d36264f05e0c55c4703f85e85e628a29231e9673bcef3c02bfe76dc2b378

source: blockchain.info

- Block header 数据结构
    - int32 nVersion
        - 当前 btc 的版本号
    - uint256 hashPrevBlock
        - 前一个区块的块头 hash
    - uint256 hashMerkleRoot
        - 可以改 Merkle Root
        - 再 CoinBase 交易的时候 没有 input
        - 因为 Nonce 有最大的范围

    - 所以通过改CoinBase 里写入任何内容修改 Merkle Root hash 从而找到目标 Hash
- uint32_t nTime
- uint32_t nBits;
- uint32_t nNonce;



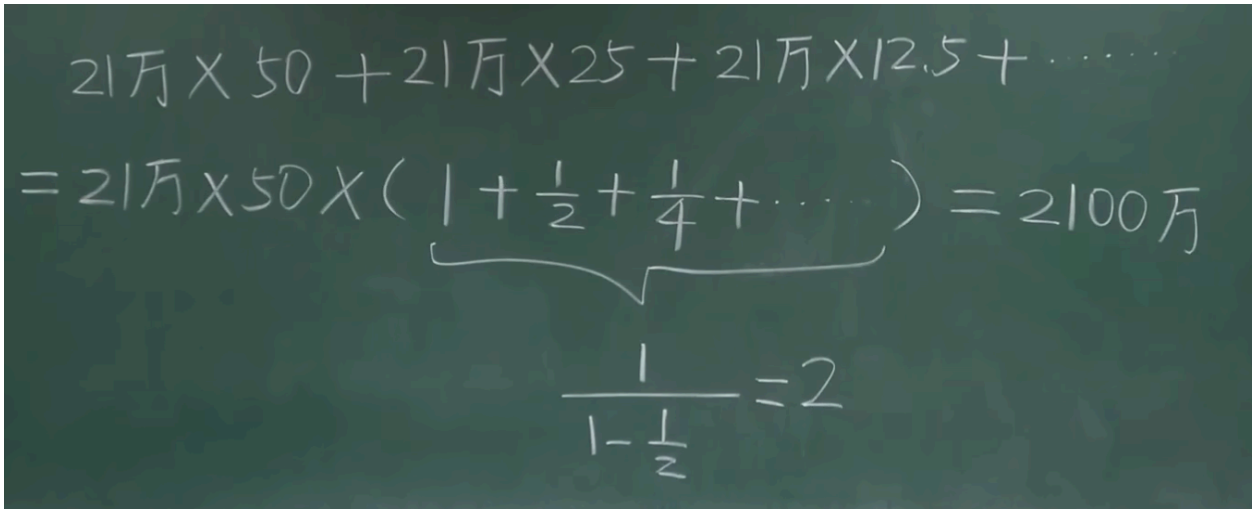
- 每个交易都有 Input Scripts 和 Output Scripts
- 每个 Input Scripts 是与 上一个 Output Scripts 配对
    - 如果两个脚本能配对这个交易就是合法的
- 区块求解只需要 Block header 的内容

POW 的过程

- 每次尝试 Nonce 称作
    - Bernoulli trial：a random experiment with binary outcome
    - 很多个 Bernoulli trial 构成 Bernoulli process: a sequence of independent Bernoulli trials
        - Memory less | progress free
        - 每个时间都是 Independent
- Poisson process
    - 实验的次数很多但是成功的机会很小

- 出块时间符合 Exponential distribution
    - 平均都要 10 分钟
    -
    - 服从指数分布
        - 例如现在很多矿机再运作，那难度就会变大，总算力少，难度就会减少

Geometric series

- 

- 系统中所有 btc 的总量

攻击者难以攻击链 或者 double spending

- 因为最长合法链的原因
- Six confirmation 确认前面无法篡改的
- Zero confirmation 就会过滤 double spending 的问题

Selfish mining

- 挖到区块先藏着不发布
- 分叉攻击的手段
- 普通的矿工这么做的意义不大
- 占有全球百分之 51 的算力有可能做到
- 可以占有接下来几个区块的奖励
- 可以让其他 Miner 无法沿着你算出的 hash
- 减少了竞争

BTC 网络

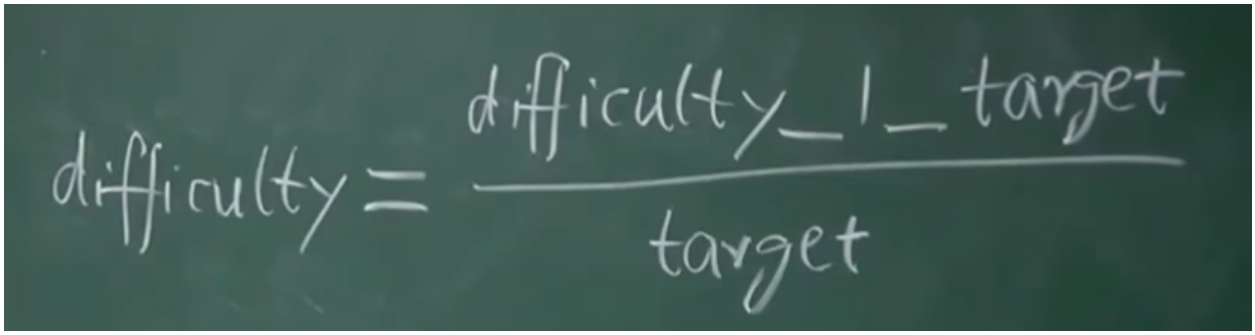
The BitCoin Network

- Application layer: Bitcoin Block chain
- Network layer: P2P overlay network

- TCP 通信为了穿透防火墙
- 设计原则
    - Simple robust but not efficient
    - Flooding
- BTC 没有 super node 或者 Master node
- Mempool 的会检测冲突交易
    - A -> B
    - A -> C
    - 如果没有出块前发现了，会发前一个删掉，（现在是会把 gas fee 低的优先删掉）
    - 如果出快了后交易，出块的是哪个，就会把另一个删掉，因为出块前打包价会确保一个 utxo 只能花费一次
- 1M 限制
- Best effort
- 无法回滚交易

BTC 挖矿难度

H(block header) LE target

- Sha-256 | 2^256
- 挖矿难度与目标阈值成反比的
- 

- Target 越大越容易
- 平衡出块时间为了抗衡摩尔定律
- 难度过于简单会增加分叉
    - 分叉过多会难以达成统一
- 51% attack

2016 区块调整一次挖矿难度

- Target = target * actual time / expected time
- Next_difficulty = previous_difficulty * expected time ./ actual time
- Target 和 difficult 成反比的，Target 越小 0 越多难度越大
- Actual time
    - Time spent mining the last 2016 blocks
- Expected time

    - 2016 * 10 / (60 * 24) = 14 days = 2 weeks
- 如果有 恶意的节点不调难度
    - 如果不调的话 hash 出来的数据与大部分数据会有区别
    - 检查区块的合法性就无法通过
- nBits
    - 压缩 target 256
    - 这个数值为了保证无法恶意调整难度
- Hash Rate/s TH/s 每秒万亿次 hash

BTC 挖矿

Full node

- 一直在线
- 本地硬盘上维护完整的区块链消息
- 内存里维护 UTXO 集合，以便快速校验交易的正确性
- 坚听 BTC 网络上的交易信息，验证每个交易的合法性
- 决定那个交易会被被打包到区块里
- 监听别的矿工挖出来的区块，验证合法性
    - 检测区块的每个交易都需要合法性以及 coinbase tx 的合法性
    - 发布的区块是否符合难度要求的
    - 对 Block header 进行检查
    - 检查区块衍生最长合法链
- 挖矿
    - 决定沿着那条链路继续挖
    - 当出现等长分叉的时候选择哪一个分叉

Light node

- 不用一致在线
- 不需要保存完整的区块链，只要保存区块头就行
- 不用保存全部交易，只保存与自己相关的交易
- 无法验证大多数交易，只能验证与自己相关的交易合法性
    - SPV 丨 Simplified payment verification
- 无法检测网上发布的区块正确性
- 无法检测 double spending
- 可以验证挖矿难度
- 只能检测那个是最长链，但不知道那个是最长合法链
    - 因为无法检测交易

ASIC：Application-specific integrated circuit

Pool manager

Pool manager 会把 coinbase transaction 固定后再 分享给 矿工

当一个矿池占到百分之 51 的算力可以发起哪些攻击了

- 分叉攻击
- Boycott
    - 可以封锁具体的帐户让他无法交易
- 盗笔是不可能出现的因为会出现分叉，诚实的矿工只会跟着正确的挖

On demand Mining

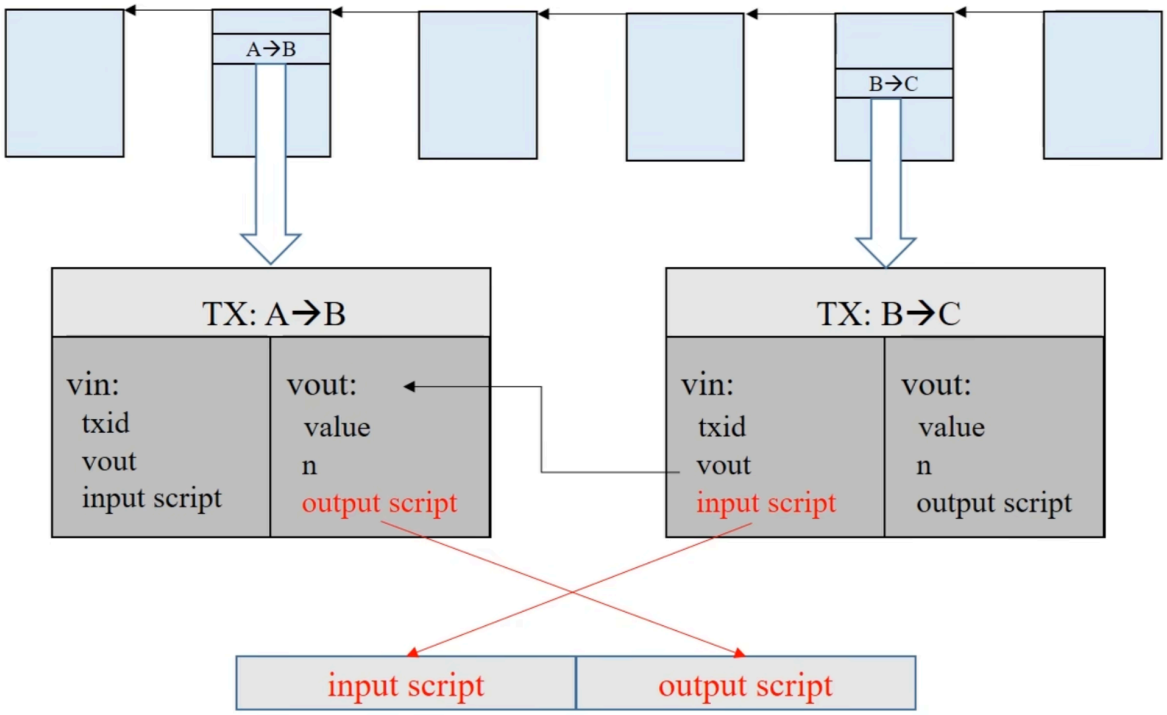
- 通常指的是一种比特币挖矿模式，其中矿工可以根据他们的计算能力和资源选择在特定时间内参与挖矿。这种模式相比传统的持续挖矿，允许矿工在需要时启动和停止挖矿活动，从而更灵活地管理资源和成本。

BTC Scripting

- Stack base language
- Locktime 是对 tx 的时间限制

scriptSig ｜ input script
Ams 是输出脚本内容 ｜ 现在是 witness
Reqsigs - 是指多少个签名需要兑现

scriptPubKey ｜ out script



- 首先只是 input script 然后只是 output script 只要这个过程中不出现 0 都是正确的，一旦出现 0 停止，之后结果也是要是非 0 值 来认证这个交易是否合法

一上面的图 A TO B then B TO C 为交易顺序

P2PK (Pay to Public Key)

- Input script
  - PUSHDATA（Sig）B私钥对交易数据进行签名生成的
- Output script
  - PUSHDATA（PubKey）B的公钥
  - Checksig

拼接后自上往下先压入栈 并进行每次 op
PushData（Sig）
PushData（PubKey）
CheckSig

P2PKH（Pay to Public Key Hash）

- Input script 解锁脚本
  - PUSHDATA (Sig)
  - PUSHDATA (PubKey) 收款人的公钥
- Output Script 锁定脚本
  - Dup
  - HASH160
  - PUSHDATA (PubkeyHash) 收款人公钥的 Hash
  - EQUALVERIFY
  - CHECKSIG

拼接后脚本执行

PUSHDATA (Sig)
PUSHDATA (PubKey)
Dup - 将堆栈顶部的元素复制一份并将副本压入堆栈
HASH160
PUSHDATA (PubkeyHash)
EQUALVERIFY
CHECKSIG

最后会留下 ture 就是个合法的交易

P2SH (Pay to Script Hash)

采用 BIP16 的方案
应用场景

- 多重签名，目前以不推荐使用

不是提供收款人的 public hash 是体用一个 叫 redeem script hash

- Input script
  - …
  - PUSHDATA (Sig)
  - …
  - PUSHDATA (serialized redeem script)
- Output script
  - HASH160
  - PUSHDATA (redeemScriptHash)
  - EQUAL

Step

- Input script 要给出一个签名（数目不定）及一段序列化的 redeem script 验证分为如下两步：
  - 验证序列化的 redeem script 是否与 output script 中的 hash value 匹配
  - 反序列化并执行 redeem script，验证 input script 中给出的签名是否正确？
- Redeem script 的形式
  - P2PK
  - P2PKH
  - 多重签名形式

Example 用 P2SH 实现 P2PK

- Redeem script 赎回脚本
  - PUSHDATA (PubKey)
  - CHECKSIG
- Input Script
  - PUSHDATA (Sig)
  - PUSHDATA (serialized redeem script) 序列化脚本
- Output script
  - HASH160
  - PUSHDATA(Redeem Script Hash)
  - EQUAL

执行过程
第一步认证
PUSHDATA (Sig)
PUSHDATA (serialized redeem script)
HASH160 - 这个过程把 **serialized redeem script** 变成 **Redeem Script Hash**
PUSHDATA(Redeem Script Hash)
EQUAL

第二步认证 反序列化
PUSHDATA (PubKey)
CHECKSIG

最原始的多重签名

多重签名

最早的多重签名，目前已经不推荐使用

input script:

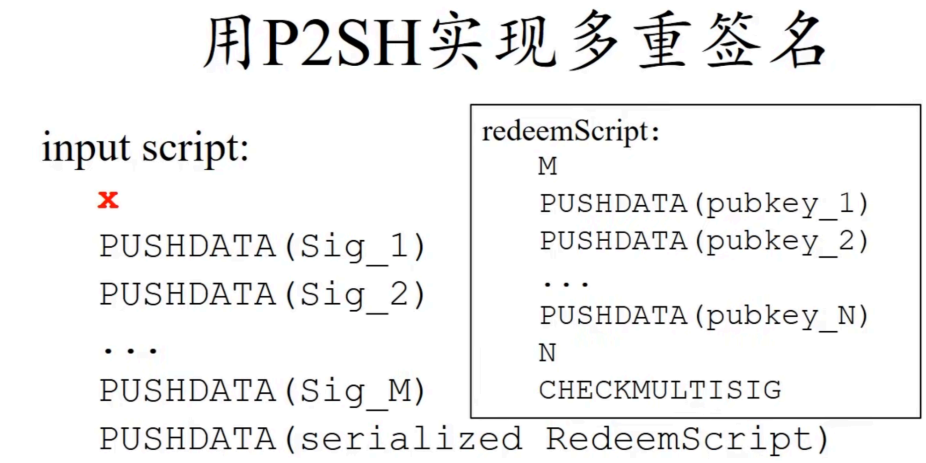
```
x
PUSHDATA(Sig_1)
PUSHDATA(Sig_2)
...
PUSHDATA(Sig_M)
```

outputScript:

```
M
PUSHDATA(pubkey_1)
PUSHDATA(pubkey_2)
...
PUSHDATA(pubkey_N)
N
CHECKMULTISIG
```

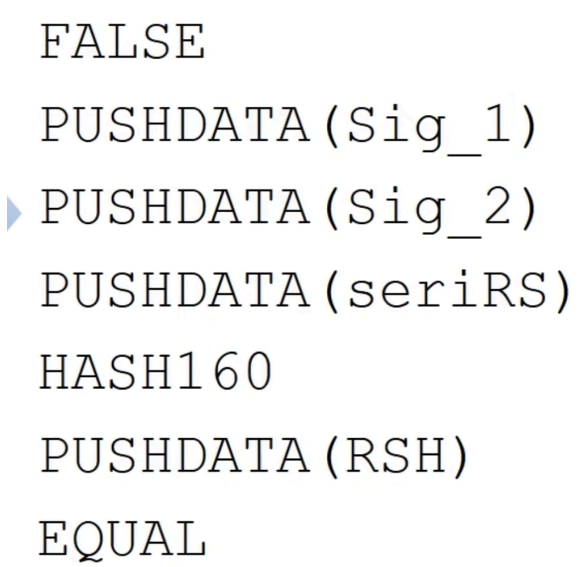
- M 是最少需要签名数量
- N 多签钱包的 PUBKEY 总数
- 发钱的人知道了多签规则

用 P2SH 实现多重签名



- 本质把复杂度从 output 转移到 input
- 现在可以自己知道规则即可

脚本执行



- False 占位符

```
2
PUSHDATA(pubkey_1)
PUSHDATA(pubkey_2)
PUSHDATA(pubkey_3)
3
CHECKMULTISIG
```

- Checkmultisig 进行验证

Proof of Burn

- Output script
  - RETURN
    - Zero or more ops or text
  - 这种形式的 output 被称为
    - Probably Unspendable/Prunable outputs
- 脚本说明
  - 假如有一个加一的 Input 指向这个 Output。不轮 input 里的 input script 如何设计，执行到 RETURN 命令之后都会立刻返回 false，不再执行 RETURN 后面的指令，所以这个 output 无法被花出去，其对应的 UTXO 无法保存
- burn btc
- 保存内容
- Digital commitment
-

使用 OP_RETURN 销毁比特币 - 其实是 btc 给矿工了

- 交易输出：包含一个 OP_RETURN 输出，该输出不可花费。
- 比特币销毁：比特币被永久移出流通，因为它们被锁定在一个不可花费的输出中。
- 交易费用：需要支付交易费用以激励矿工将交易包含在区块中。

{ "vin": [{ "txid": "your-input-transaction-id", "vout": 0, "scriptSig": { "asm": "...", "hex": "..." }, "sequence": 4294967295 }],
"vout": [{ "value": 0.00000000, // 0 BTC "n": 0, "scriptPubKey": { "asm": "OP_RETURN 48656c6c6f2c20576f726c6421", "hex": "6a1048656c6c6f2c20576f726c6421", "type": "nulldata" } }] }

使用 OP_RETURN 记录消息而不销毁比特币

{ "vin": [{ "txid": "your-input-transaction-id", "vout": 0, "scriptSig": { "asm": "...", "hex": "..." }, "sequence": 4294967295 }],
"vout": [
{ "value": 0.00000000, // 0 BTC "n": 0, "scriptPubKey": { "asm": "OP_RETURN 48656c6c6f2c20576f726c6421", "hex": "6a1048656c6c6f2c20576f726c6421", "type": "nulldata" } },
{ "value": 0.00049000, // 找零金额 "n": 1, "scriptPubKey": { "asm": "OP_DUP OP_HASH160 abcdef1234567890abcdef1234567890abcdef12 OP_EQUALVERIFY OP_CHECKSIG", "hex": "76a914abcdef1234567890abcdef1234567890abcdef1288ac", "reqSigs": 1, "type": "pubkeyhash", "addresses": ["1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa"] } }] }

BTC 分叉 fork

- State fork
  - 临时性的分叉
  - 例如突然两个矿工同时提交计算好的区块
  - Forking attack
  - Deliberate fork
- Protocol fork
  - Hard fork
  - Soft fork

Hard fork

- 必须要所有的节点都需要更新软件
- Block size limit
  - 1m, 1000000 byte
  - 250 byte per tx
  - Around 4000 txs in a block
- 出现 2 条合法链
- 通过 Chain ID 区分分叉后的链

Sort fork
- 临时分叉
- 例如有些 block 支持 RBF
- P2SH

CoinBase
- Extra nonce
  - 前 8 个字节作为 nonce
- 后面的字节

BTC 匿名性

Bitcoin and anonymity
- Pseudonymity
- 只是没有名字

Network layer
- Muti route
- Tor

Application layer
- Coin mixing
- 交易所

零知识证明

- 一方（验证者）向另一方（验证者）证明一个陈述是真确的，而无需透露除该陈述是正确的外的任何信息
- 私钥签名这个就是零知识证明 -> 有争议

同态隐藏

- 如何 x, y 不同，那么他们的加密函数 E(x) 和 E(y) 也不相同
    - 反命题 - 如果 E(x) 和 E(y) 也相同，x, y 相等
- 给定 E(x) 的值，很难反推出 x 的值
    - 加密函数不可逆的
    - Hiding property
- 给定 E(x) 和 E(y) 的值，我们可以很容易地计算出某些关于 x,y 的加密函数值。
    - 同态加法：通过 E(x) 和 E(y) 计算出 E(x + y) 的值
    - 同态乘法：通过 E(x) 和 E(y) 计算出 E(x y) 的值
    - 扩展到多项式
- 例子
    - Alice 想要向 Bob 证明她知道一组数 x 和 y 使得 x + y = 7，同时不让 Bob 知道 x 和 y 的具体数值？
        - Alice 把 E(x) 和 E(y) 的数值发给 Bob
        - Bob 同个收到 E(x) 和 E(y) 计算出 E(x + y) 的值 (利用了性质 3)
        - Bob 通过计算得出 E(x + y) = E(7), 如果不是验证失败

盲签方法
- 是一种密码学技术，允许一方对另一方的信息进行签名，而不泄露该信息的内容。盲签名的主要用途是在需要保护隐私的场景中，如电子投票、数字现金等。
- 用户 A 提供一个 Serial num，银行在不知道 Serial num 的情况下返回签名 Token，减少 A 的存款
- 用户 A 把 Serial num 和 Token 交给 B 完成交易
- 用户 B 拿 Serial num 和 Token 给银行验证，银行验证通过，增加 B 的存款

- 银行无法把 A 和 B 联系起来
- 中心化

零币和零钞

- 零币和零钞在协议层就融合了匿名化处理，其名属性来自密码学保证
- Zerocoin 系统中存在基础的币和零币，通过基本币和零币带来回转换，消除旧地址和新地址的关联性，其原理类似于混币服务。
- Zerocash 系统使用 zk-SNARKs 协议，不依赖一种基础比，区块链中记录交易的存在性和矿工用来验证系统正常运行所需要关键属性的证明。区块链上不显示交易也不现实交易金额，所有交易通过零知识验证的方式进行。

- 从数学保证存在Zerocoin 和 Zerocash 存在，但是破坏掉关联性
- 与实体发生交互的时候，任然要暴露身份

Note

肖臻老师说的比较受用的话：

- 知识改变命运，如果学的一知半解有可能使你的命运更差。
- 不要被学术限制头脑，不要被程序员思维限制了想象力

Runestone

https://www.cnblogs.com/studyzy/p/18166591

- Rune
  - 代币
- Etch
  - 创造方式
- Rune Name
  - A - Z
- Spacer
  - •
- Rune Number
  - 按照实创建的先后顺序
- Rune ID
  - Block number: tx number in block
  - 84000:1
- Rune symbol
  - 符號
- Divisibility
  - 可分性
- Oremine
- Mint
  - 铸造
- Mint Terms
  - Mint 规则
- Runestone
  - OP_RETURN 操作符 记录内容
- Edict
  - 符石内的一条信息，它允许你自定义符文在转账交易中将要转向的输出，以及符文将转向某个输出的数量（这允许你定制特殊交易，例如在同一笔交易中将符文从一个地址发送到另外 10 个地址）
- Cenotaph
  - 由于某种原因导致符石格式错误时，它被称为纪念碑，在出现纪念碑的事件中，交易中的所有符文都会被销毁。
-

Fractal

Brc20

- 6 -12 chars
- 可能是为了避免之前的资产过度
- 所以 6 -12 很有可能成为后面的 Fractal 主流资产货币
- 4 - 5 都会从 btc 迁移

InfinityAi

- 给予 Web3 Ai 工具的去中心化 ALGC 人工智能生产内容 基础设置，构建在比特币 Layer2 网络上，它允许用户成为比特币生态系统中的内容创建者，并以社区驱动的方式使用人工智能工具对创作进行代币化

UniWorlds.io

- UniWorlds是一个3D主权个人内容创作平台，充分赋能个人通过3D创作引擎在比特币网络上构建自己的虚拟空间。利用分形比特币 Layer 2 网络的高性能和安全性，以及开放和去中心化的社区治理，它激励更多的个人参与 3D 内容创作。这使得更多的“超级个体"能够在虚拟空间中建立不可替代的链上身份，防范人工智能时代人类身份的混乱。

SatsPumpfun

- SatsPump.Fun是比特币上第一个利用 Fractal Bitcoin 的先进技术的平台，通过社区驱动的流动性池提供安全、简单、快速、可靠和可扩展的代币发布。由于通过 Fractal Bitcoir的快速 30 秒区块确认和递归缩放技术，SatsPump.Fun 的目标是成为比特币代币推出的去中心化去中心化
- Meme 币快速发布和融资平台

OP_NET

- OPNET 是一种协议，通过在比特币上引入智能合约功能，实现了比特币区块链的可编程性，而无需修改其核心协议。它利用 Tapscript在比特币现有框架内创建了一个智能合约环境，增强了比特币的功能。两者结合可以在保证比特币安全性的前提下，提高交易速度和效率，降低成本。目前团队成功集成到 fractal bitcoin 测试网

Motoswap

- DEX

Cat20

在密码学和区块链中，**preimage（前像）是指某个值通过一个哈希函数**（例如 SHA-256）计算后生成的输出（即哈希值）的输入值。简单来说，preimage 是生成哈希值的原始数据。

在哈希函数中，有一个重要的特性叫做“preimage resistance”，即根据已知的哈希值，很难逆向计算出对应的 preimage。这一性质确保了哈希函数的安全性。

举个简单的例子：

- 假设你有一个哈希值 `h = SHA256("hello")`。
- 其中 `"hello"` 就是 preimage，而 `h` 是通过 SHA-256 计算出来的哈希值。

在区块链系统中，preimage 常见于**哈希锁定合约**或**支付通道**的应用中。例如，某个交易可能要求提供特定哈希值的 preimage 才能完成交易，从而保证安全性和不可逆性。