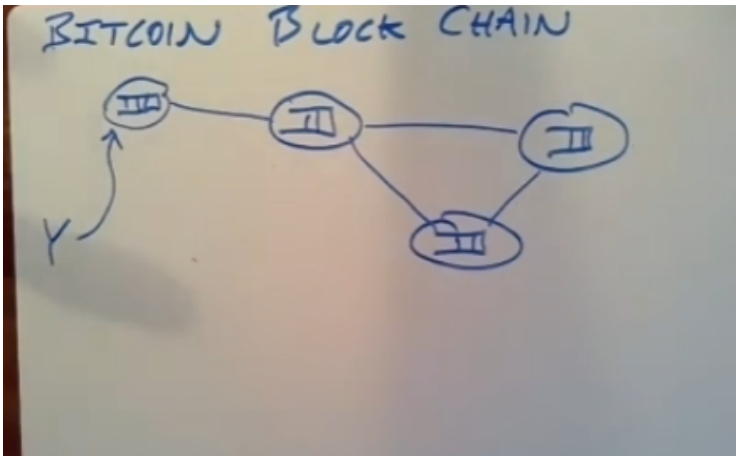
LECTURE 19 BITCOIN

1. 背景

- **Public ledger：** 使得各个结点能够对已发生的交易以及交易发生的顺序达成共识
  - 中心化的方式在现实世界不可取，不存在每个结点都信任的中心化权威节点。
  - 各个结点之间相互不信任
- 货币交易信息传播方式：
  - 洪泛法发送给所有对等实体
  - bitcoin 策略：发送给若干节点，若干节点继续转发
- 并发交易之间达成一致：
  - 对于交易的某一个槽位，可以使用投票的方式，多数票获胜。但事实上非常难以统计像比特币这样的开放系统中参与者的数目，并且很难保证一个参与者只能投出一票；即使一个参与者只能投出一票，攻击者也可以使用肉鸡的方式操纵选举，来实现double spend的目的。（攻击者同时对A和B发出交易，对于交易槽位x，当A询问x槽位是哪个交易时，攻击者操纵肉鸡投票给A交易；当B询问时，攻击者操纵肉鸡投票给B交易，从而达到double spend的目的。）

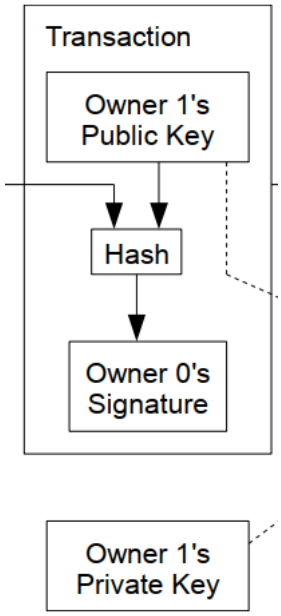
2. 基本架构

- 各个站点之间通过TCP协议相互连接，如图所示：



- 区块（block）：各个站点积累一定的交易，组合成为一个区块。而后区块转发给系统中的各个结点。

- 结构：
  - 前一个区块的哈希值
  - 一组交易
  - nounce字段
  - 时间戳
- 交易的结构：



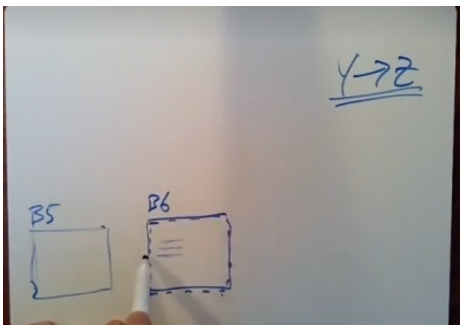
  - 此货币上一次交易
  - 货币新拥有者的公钥
  - 用原拥有者的私钥对上面两项内容的签名

- 当发生交易时，payee（接受者）需要去区块链中寻找对应的交易。**但不能因为最后一个区块中含有对应的交易就认为该交易实际发生，需要考虑到fork的情况。如果只看最后一个区块则会导致double spend现象，通常需要等待5~6个后续区块出现。** 当交易额度很大时，尤其需要注意。

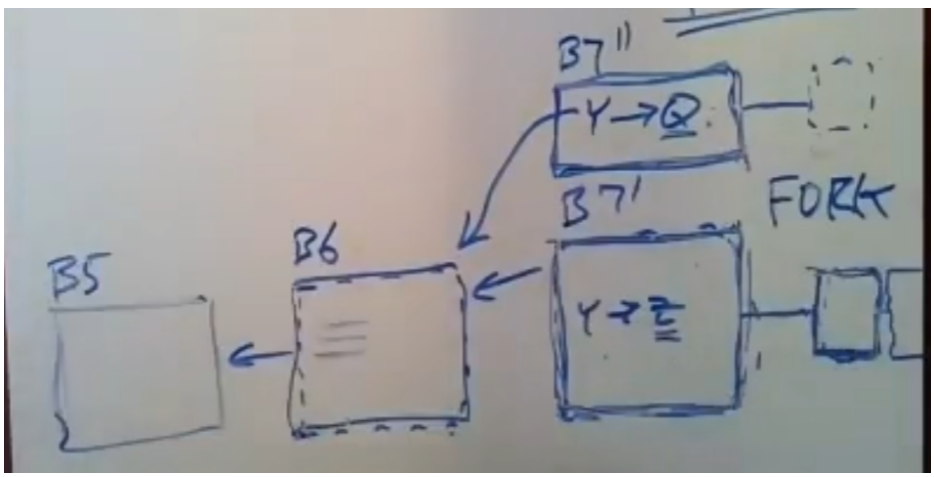
3. 挖矿/工作量证明（proof of work）

- 仅当一个区块的哈希值 含有 规定数目的前导0时，才能认为是有效的
  - 可以更改区块之中的nounce字段来获取不同的哈希值。通常nounce字段是随机选取的

    - 区块链要求一个区块大约10min左右被挖出来，当系统中的结点过多时，则会增加前导零位数要求来保证这一点
    - 单台机器大约需要一个月时间进行挖矿
- 工作量证明并非是一种历史交易的表决方案，是在各个结点中对区块的随机选择过程
- 只要非恶意结点的CPU资源大于恶意节点的CPU资源，就能保证整个比特币系统的安全
- 当有交易到达时，该交易无法影响当前正在计算的区块，会被暂存到缓冲区中，放在下一个区块中，如图：



- 当两个结点在同一时间挖出了同一个区块（假设为B7）时：
    - 两个结点可能分别向一些结点发送了自己的区块B7a和B7b，这些区块分别在这两个结点的基础上再进行下一个区块B8的挖掘
    - 区块挖掘时间的方差非常大，尽管这两个结点碰巧同时挖出了区块B7，下一个区块B8挖掘的时间大概率不相等。假设采用B7b的结点先挖掘出了B8，则所有采用B7a的结点将会切换到B7b这一分支，而后继续挖掘，如图：



    - 关于double-spend的问题：当区块链产生fork的时候，会短暂的发生double-spend现象，但是随着fork的一个分支被抛弃，double-spend现象会消失。

        - 假设y->z和y->Q是并发的两个交易，则后面发生的交易会被抛弃（可能是指在同个区块的环境之中，未明确）

- 中间人并不能随意修改中间任何一个区块而不修改其之后的区块，因为每一个区块都包含了前面一个区块的哈希值。（这本质上也是比特币安全性的保障，即只要绝大多数结点非恶意，绝大多数的结点的CPU资源比恶意结点的CPU资源更多，则恶意结点永远不可能修改区块链）

- 关于货币有效性的问题：Moris猜测在每个结点中可能为未花费的货币建立了一个数据库，当一个交易到达，且交易的货币不在该数据库中（意味着已使用），则丢弃该交易。（个人思考：每个交易之中都包含了对应的货币上一次交易的信息，因此一个货币可能也是形成一个链式的结构）

- 关于前导零动态变化的问题：

    - 存在一个确定性函数，在最后一个区块上，通过该确定性函数来得到前导零的个数。

- 新节点连接：比特币软件中附加了若干个结点，可供新节点与之建立TCP连接

4. 比特币的缺陷

- 交易时间过长，必须要等待5~6个区块的出现才能确认交易达成

- 比特币区块链过大（几百个G）

- 一个区块只有几兆大小。限制了交易的数量

- 对于特定对象的交易记录的追踪很困难，公钥/私钥每一次可以重新生成
