网络安全技术

刘振

上海交通大学 计算机科学与工程系

电信群楼3-509

liuzhen@sjtu.edu.cn

比特币简介

- 1 比特币的账本
- 2 比特币的交易
- 3 比特币的脚本语言
- 4 比特币的区块
- 5 比特币挖矿
- 6 比特币网络
- 7 比特币：从交易到上链
- 8 比特币的矿池
- 9 比特币的交易处理能力
- 10 比特币的协议升级
- 11 比特币中的一些攻击

4. 比特币的区块

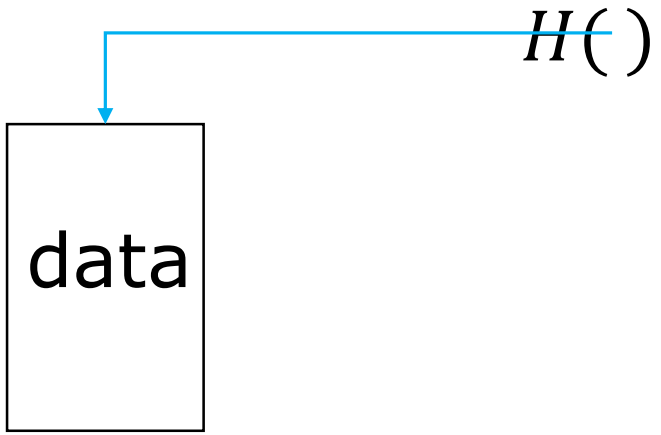
- 交易“体小量大”，自然而然的想法就是把交易打包成“块”、对块来操作，从而达到提高效率的目的。

- 在把交易打包到块的过程中如何组织这些交易？
  - 可以有各种方式
  - 比特币使用的是一种称为Merkle Tree的数据结构

- Merkle Tree
  - 用**Hash指针**建立的二叉树
  - 能够快速地（对数级复杂度）判定一个数据是否在一个集合中
  - 能够快速地判定集合中是否有数据被篡改

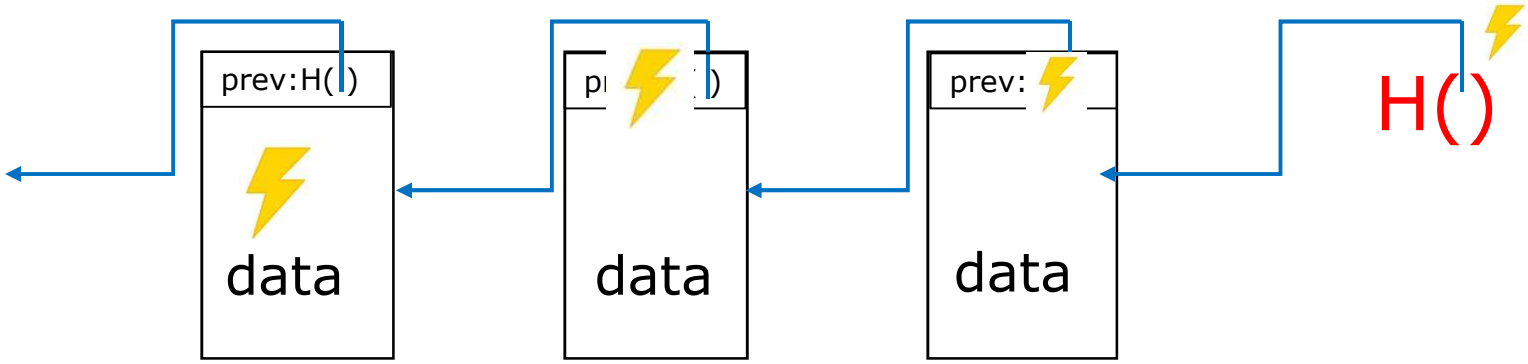
4. 比特币的区块：Hash指针

- Hash指针
  - 数据data的hash值H(data)蕴含了data的存储位置信息，即可以通过H(data) 的值读取到数据。从而，h(data)不仅指明了数据的存储位置，也提供了对数据是否被篡改过的验证证据。从一个hash值y处读取数据data，只有当y=H(data)成立，才说明数据没有被篡改过。例
  - 例如，比特币中每个交易的hash是交易的唯一标识。



4. 比特币的区块：Hash指针

- Hash链

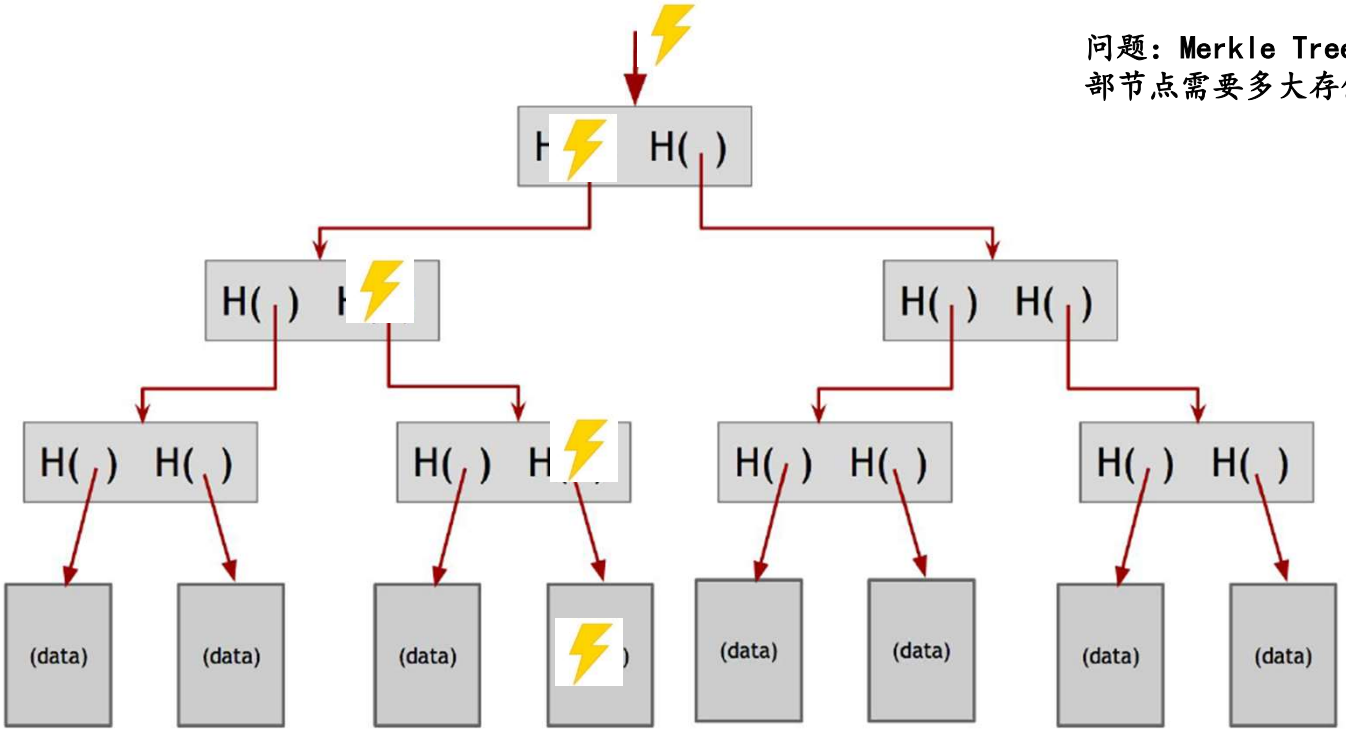


只需要记住最后的这个hash值，就能保证整条链的防篡改性。

基于的什么密码学原理？

4. 比特币的区块：Merkle Tree

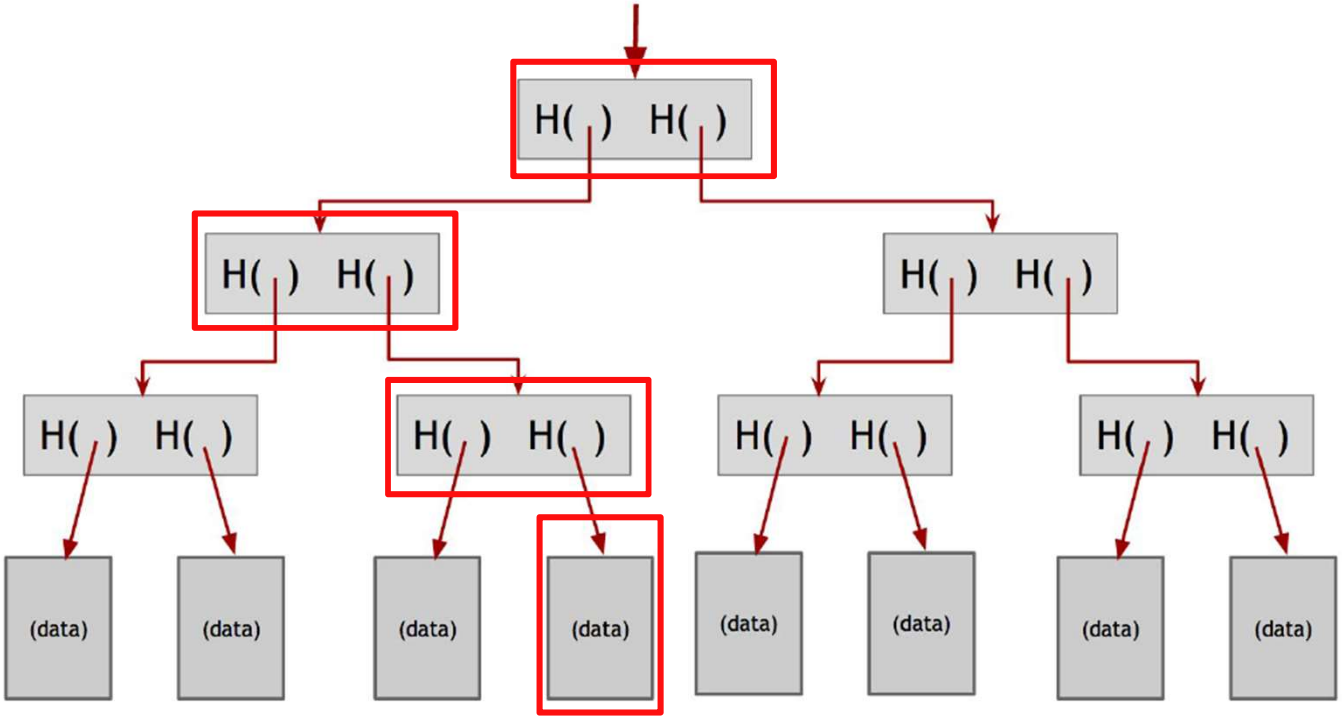
- Merkle Tree
  - 数据放在叶子节点，除了叶子节点之外，每个节点包含两个指向其子节点的Hash指针。（每个节点的Hash值存在其父节点中）
  - 任何数据被篡改的话，会影响该节点到根节点这个路径上所有节点的Hash值，当然也包含根节点的Hash值。所以，存储根节点Hash值即可保证数据的不可篡改性。



问题：Merkle Tree的每个内部节点需要多大存储空间？

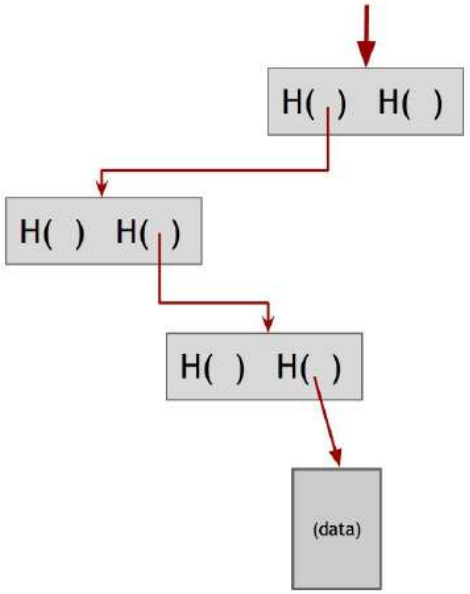
4. 比特币的区块：Merkle Tree

- 隶属关系证明
  - 在Merkle Tree中可以高效地证明一个隶属关系：一个数据在一个Merkle Tree的数据集合中
  - 输入：a) Merkle Tree 根节点的Hash值 b) 一个数据块，及从该数据块到根节点的路径上所有节点
  - 对于有n个叶子（数据成员）的Merkle Tree中，需要的空间和时间复杂都是log(n)。



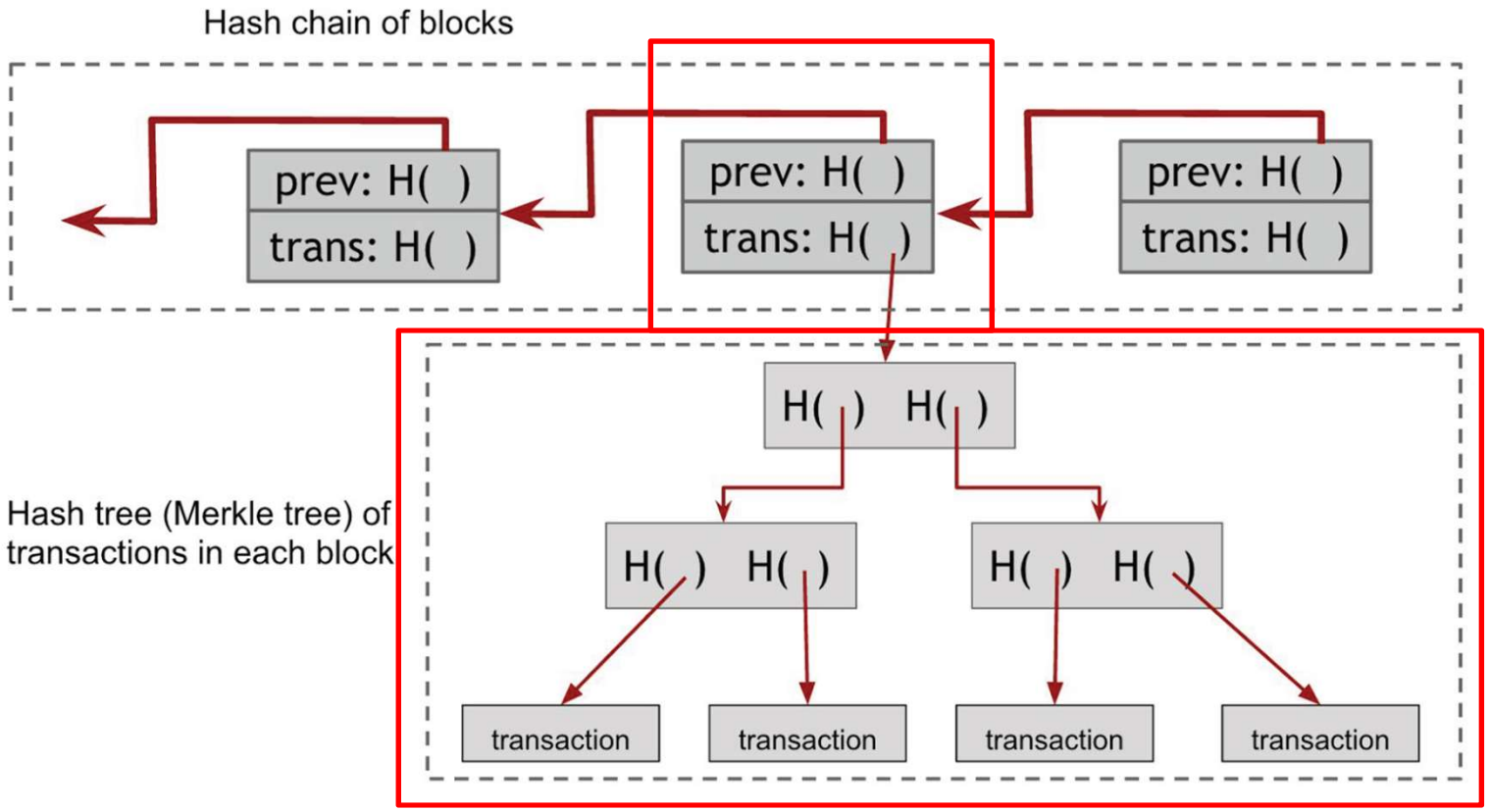
4. 比特币的区块：Merkle Tree

- 隶属关系证明
  - 在Merkle Tree中可以高效地证明一个隶属关系：一个数据在一个Merkle Tree的数据集合中
  - 输入：a) Merkle Tree 根节点的Hash值 b) 一个数据块，及从该数据块到根节点的路径上所有节点
  - 对于有n个叶子（数据成员）的Merkle Tree中，需要的空间和时间复杂都是log(n)。



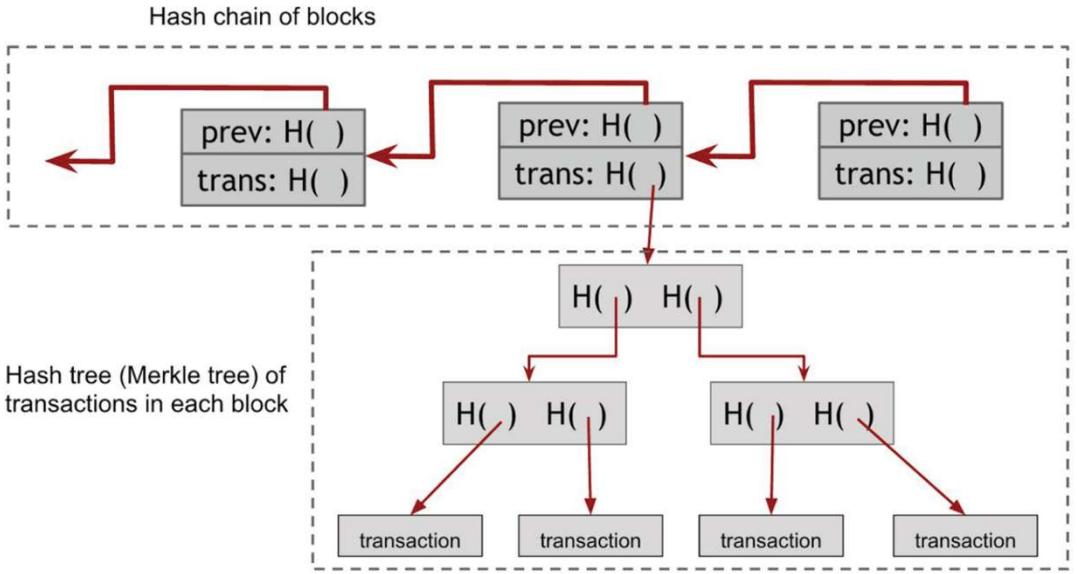
4. 比特币的区块

- 比特币把交易用Merkle Tree打包成区块(Block)，区块之间用Hash指针链起来，形成“区块链”（Blockchain）



4. 比特币的区块

- 区块链中“Hash”和“链”的只是区块头(Block head)
- 交易的Merkle Tree的根节点的Hash值在区块头中，保证了块中的交易数据被“包含在链中”
  - ---修改交易数据，会影响到Merkle Tree根节点的Hash值，进而影响到整个块的Hash值
- 区块：
  - 区块头
  - 区块主体（交易数据）



4. 比特币的区块

- 区块头：

字段	大小	描述
previousblockhash	32字节	前一区块Hash值
merkleroot	32字节	交易Merkle Tree根节点的Hash值
difficulty	4字节	挖矿的难度系数
nonce	4字节	挖矿挖到的临时随机数

- 区块（头）标识：
  - 区块Hash值
    - 区块本身的Hash值是区块的唯一性标识。
    - 区块Hash值并不包含在区块的数据结构里，不管是该区块在网络上传输时，抑或是它作为区块链的一部分被存储在某节点的永久性存储设备上时。
    - 任何人都可以根据区块数据计算其Hash值，实际上区块Hash值是当该区块从网络被接收时由每个节点计算出来的。
    - 区块的哈希值可能会作为区块元数据的一部分被存储在一个独立的数据库表中，以便于索引和更快地从磁盘检索区块。
  - 区块高度
    - 创世区块的高度是0，其后每个区块的高度比前一个区块高度加1。
    - 区块高度并不是唯一的标识符，有可能两个相同高度的区块同时存在、竞争在区块链中的位置。
    - 区块高度也不是区块数据结构的一部分，它并不被存储在区块里。当节点接收来自比特币网络的区块时，会动态地识别该区块在区块链里的位置（区块高度）。
    - 区块高度也可作为元数据存储在一个索引数据库表中以便快速检索。

4. 比特币的区块

- 每个区块的交易（树）中都有且只有一个特殊的交易：coinbase交易。
  - 只有一个输入，只有一个输出：且输入并不指向任何前驱交易的输出
  - 并不消费之前的交易的输出（的比特币），因此，输入并不指向“上一交易”
  - 输出币值=当前区块奖励值+交易费
    - 区块奖励值最初是50个比特币，每生产210000个区块奖励值减半。
    - 交易费： 每个交易的交易费是“输出额与输入额的差值”，区块内的交易费是所有交易的交易费求和 。
  - coinbase参数：矿工可以放任何数据进去

```
"in":[
      {
        "prev_out":{
          "hash":"000000.....0000000",
          "n":4294967295
        },
        "coinbase":"..."
      },
    ]
"out":[
      {
        "value":"25.03371419",
        "scriptPubKey":"OPDUP OPHASH160 ... ”
      }
     ]
```

4. 比特币的区块

- Coinbase交易中没有签名去保证输出不被更改，有问题吗？
  - 节点A发布一个“合法”的区块，其中的coinbase交易的输出是A的一个地址，节点B是否能够修改coinbase交易的地址为B的地址，从而获得相应的比特币？

```
"in":[
       {
         "prev_out":{
           "hash":"000000.....0000000",
           "n":4294967295
         },
         "coinbase":"..."
       },
     ]
"out":[
       {
         "value":"25.03371419",
         "scriptPubKey":"OPDUP OPHASH160 ... ”
       }
      ]
```

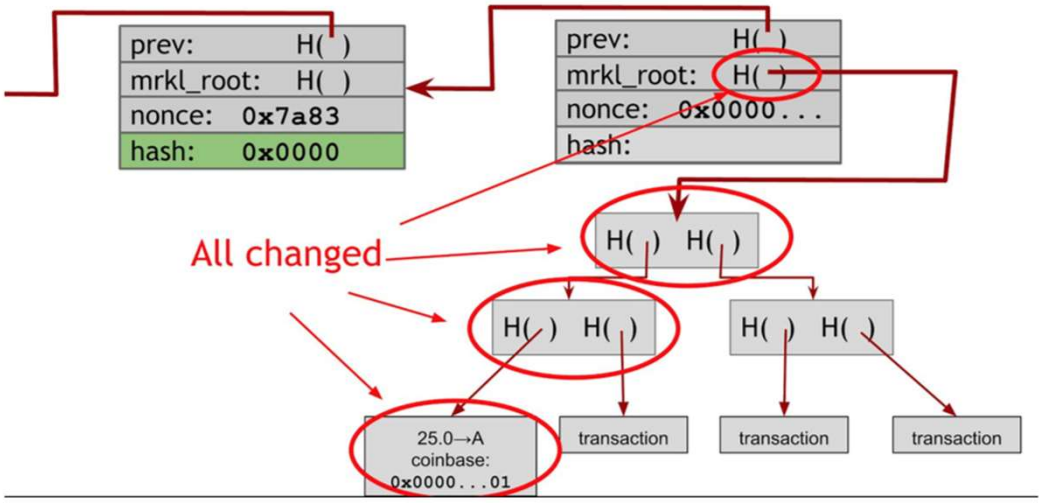
5. 比特币挖矿（mining）

- 交易是由用户发起的，区块是由谁产生的？
- 在比特币系统中，没有中心机构负责产生区块及“造币”，而是有所有的参与节点进行竞争，竞争的胜利者获得“产生区块”（或称为记账）的权利，把一批交易记入区块链（即账本），每个新建区块中的coinbase交易相当于“造币”，凭空生成出比特币。由于coinbase中的输出（区块奖励）由产生该区块的节点指定，这就鼓励了节点积极参与竞争、争取记账权利。
- 这个竞争通过“挖矿”实现：
  - 有一个系统级的参数：挖矿难度系数。该参数指明的是Hash值的开头有多少个零，或者把Hash值看作数字的话，该参数指明一个目标值target，挖矿的目标是找到Hash值小于该目标值的区块。
  - 如何找？：对区块头中的nonce值进行修改并计算相应的区块头的Hash值，如果Hash值小于target，则说明挖到了一个有效（可以成为区块链上）的区块，可以用这个区块区参与竞争记账

字段	大小	描述
previousblockhash	32字节	前一区块Hash值
merkleroot	32字节	交易Merkle Tree根节点的Hash值
difficulty	4字节	挖矿的难度系数
nonce	4字节	挖矿挖到的临时随机数

5. 比特币挖矿（mining）

- 挖矿：
  - 有一个系统级的参数：挖矿难度系数。该参数指明的是Hash值的开头有多少个零，或者把Hash值看作数字的话，该参数指明一个目标值target，挖矿的目标是找到Hash值小于该目标值的区块。
  - 如何找？：对区块头中的nonce值进行修改并计算相应的区块头的Hash值，如果Hash值小于target，则说明挖到了一个有效（可以成为区块链上）的区块，可以用这个区块区参与竞争记账。
  - Nonce的2^{32}个值都试过来，也没有找到合格的区块怎么办？：修改coinbase交易中的随机数，影响merkle tree 根节点的Hash值，然后重新对nonce的值搜索



5. 比特币挖矿（mining）

- 挖矿就是进行*Hash运算*、寻找合适的nonce值使区块的Hash值小于目标值。

```
TARGET = (65535 << 208) / DIFFICULTY;
coinbase_nonce = 0;
while (1) {
      header = makeBlockHeader(transactions, coinbase_nonce);
      for (header_nonce = 0; header_nonce < (1 << 32); header_nonce++){
            if (SHA256(SHA256(makeBlock(header, header_nonce))) <
      TARGET)
                  break; //block found!
      }
      coinbase_nonce++;
}
```

CPU挖矿的伪代码

5. 比特币挖矿（mining）

- 挖矿就是进行*Hash运算*、寻找合适的nonce值使区块的Hash值小于目标值。
  - 挖矿：消耗电和硬件资源，生产比特币（区块奖励）

- 挖矿“胜利者”的产生：投入计算资源多、运气好的节点
  - 投入资源越多，越可能称为胜利者（可以进行更快更多次的Hash运算尝试）
  - “运气”（随机性）保证了不全依赖于资源的投入多少来决定，保证了去中心化的特征
    - 投入200计算能力的节点一定比投入100计算能力的节点先找到合格的区块吗？

5. 比特币挖矿（mining）

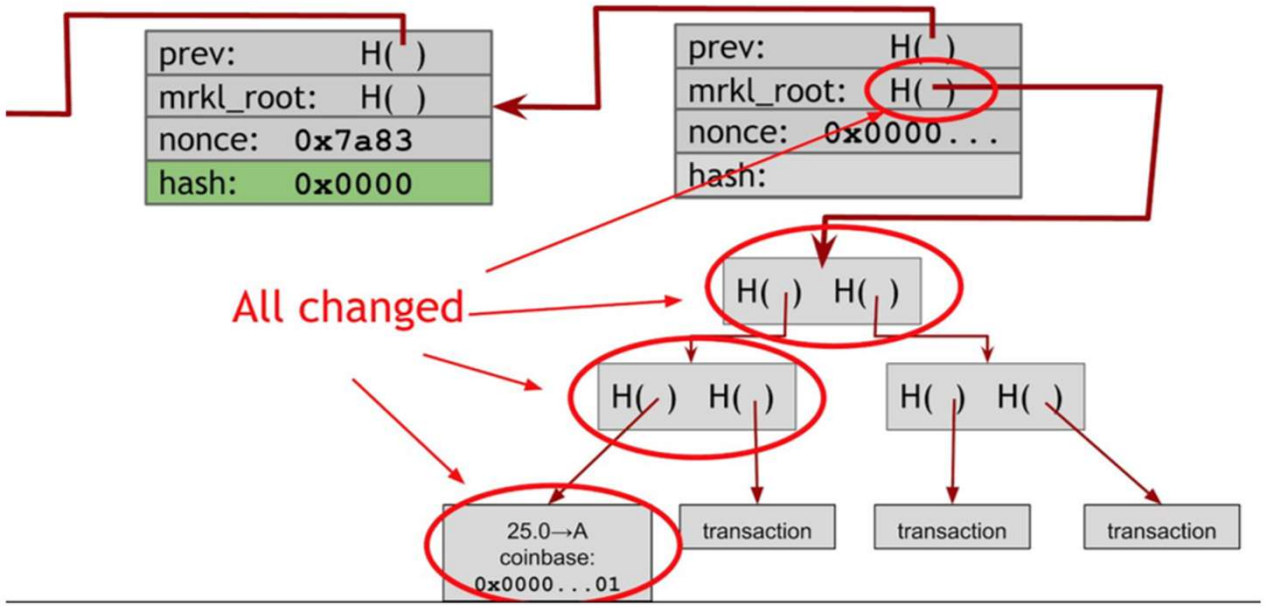
- 挖矿难度控制了区块产生的速度。
  - 为什么要控制区块产生的速度？
  - 可以从挖矿难度值计算出有效区块的hash值得target范围
- 每挖出2016个区块，挖矿难度会改变一次，这个周期大约是两个星期。难度的改变是根据上2016个区块的挖矿效率来决定的。用下列公式表达：

$$\text{下一个难度} = \frac{\text{上一个难度} \times 2016 \times \text{10分钟}}{\text{产生上2016个区块所花费的时间}}$$

- 如果这个周期太短，难度会随着每一个周期找到的区块的数目的不同而波动（概率问题）
- 如果这个周期太长，整个网络的哈希算力会与难度大大地失去平衡（难度的调整滞后于计算能力的变化）
  - 直观地，当计算资源在上一个周期内大幅增加时，会导致找到区块的时间变短，计算出的难度系数会变大，使得找到区块的时间重新回到期望的每十分钟一个块
- Block #0, 2009.01.03 difficulty=1， hash值 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
- Block #32255(=0+16*2016-1), difficulty=1, hash值 00000000984f962134a7291e3693075ae03e521f0ee33378ec30a334d860034b
- Block #32256, difficulty=1.18, hash值 000000004f2886a170adb7204cb0c7a824217dd24d11a74423d564c4e0904967
- Block #34271(=32256+2016-1), difficulty=1.18, hash值 000000005e36047e39452a7beaaa6721048ac408a3e75bb60a8b0008713653ce

5. 比特币挖矿（mining）

- coinbase交易中没有签名去保证输出不被更改，有问题吗？
  - 节点A发布一个“合法”的区块，其中的coinbase交易的输出是A的一个地址，节点B是否能够修改Coinbase交易的地址为B的地址，从而获得相应的比特币？



6. 比特币网络

- 交易和区块都是通过“比特币网络”在参与节点之间传播的。
- 比特币网络：
  - 点对点网络
  - 所有的节点是平等的，没有等级，也没有特殊的主节点
  - 运行在TCP网络上，可以是任意的拓扑结构，每个节点和其他的随机节点相连
  - 新节点可以随时加入，旧节点可以随时随意的离开
    - 加入：一个新节点启动时，先向一个其知道的节点发送一个简单的消息，这个节点称为是新节点的种子节点；然后问种子节点是不是还知道其他节点，获得新节点后，再多次重复这个过程，最后可以选择和哪些节点相连。
    - 离开：三个小时没有音讯，就会被别的节点忘记

6. 比特币网络

- 交易和区块都是通过“比特币网络”在参与节点之间传播的

- 比特币网络上的“泛洪算法”：
  - 节点收到数据后检查其有效性，
    - 如果有效，则检查是不是已经在本地有存储（已经收到过）
      - 如果收到过，则不再转发
      - 如果没有收到过，则在本地存储，并转发给相邻的节点
    - 如果无效，则不存储、不转发

7. 比特币：从交易到上链

1. 当一个用户/节点提出一笔交易时，该节点把交易广播到其相邻节
2. 当一个节点接收到一个交易时，执行泛洪算法，步骤如下：
   - ✓ 交易验证：
     - ✓ 检查是否有双重支付（输入指向的交易的输出已经被花销过）
     - ✓ 检查输出额不能超过输入额；
     - ✓ 验证交易在当前的区块链（节点本地的区块链）中是否是有效的，通过对交易的每个输入运行核验脚本，确保脚本的返回值都是TRUE；
   - ✓ 检查这笔交易是否被本节点接收过
   - ✓ 对于有效的交易，节点把交易放入其交易池

7. 比特币：从交易到上链

3. 挖矿：矿工节点从其交易池中选取准备包含进区块的交易，并构建相应的coinbase交易，进一步建立交易的merkle tree，使用当前（本地）的最新的区块的hash值作为previousblockhash，开始挖矿（使用不同的nonce值计算区块头的Hash值，必要的时候修改coinbase交易的随机数）

4. 某个节点挖矿成功后（找到了符合要求的区块），把整个区块（交易的merkle tree 、区块头）广播出去

7. 比特币：从交易到上链

5. 当一个节点接收到一个备选新区块时，执行相应的泛洪算法：
   - ✓ 确认区块头部的Hash值是在可以接收的范围内（每个节点本地都有系统的难度参数）
   - ✓ 确认区块里的每个交易（注意coinbase交易的校验是通过其他交易和系统的区块奖励值进行的）
   - ✓ 确认交易的Merkle Tree 的根节点的Hash值确实是等于区块头里的相应值
   - ✓ 确认这个新区块是接在当前最长链上的最新区块上的
     - ◆ 当前最长链是根据节点自己本地的认知的
   - ✓ 确认接收的备选新区块满足上述验证后，更新本地的区块链，并在更新后的区块链的基础上继续挖矿

7. 比特币：从交易到上链

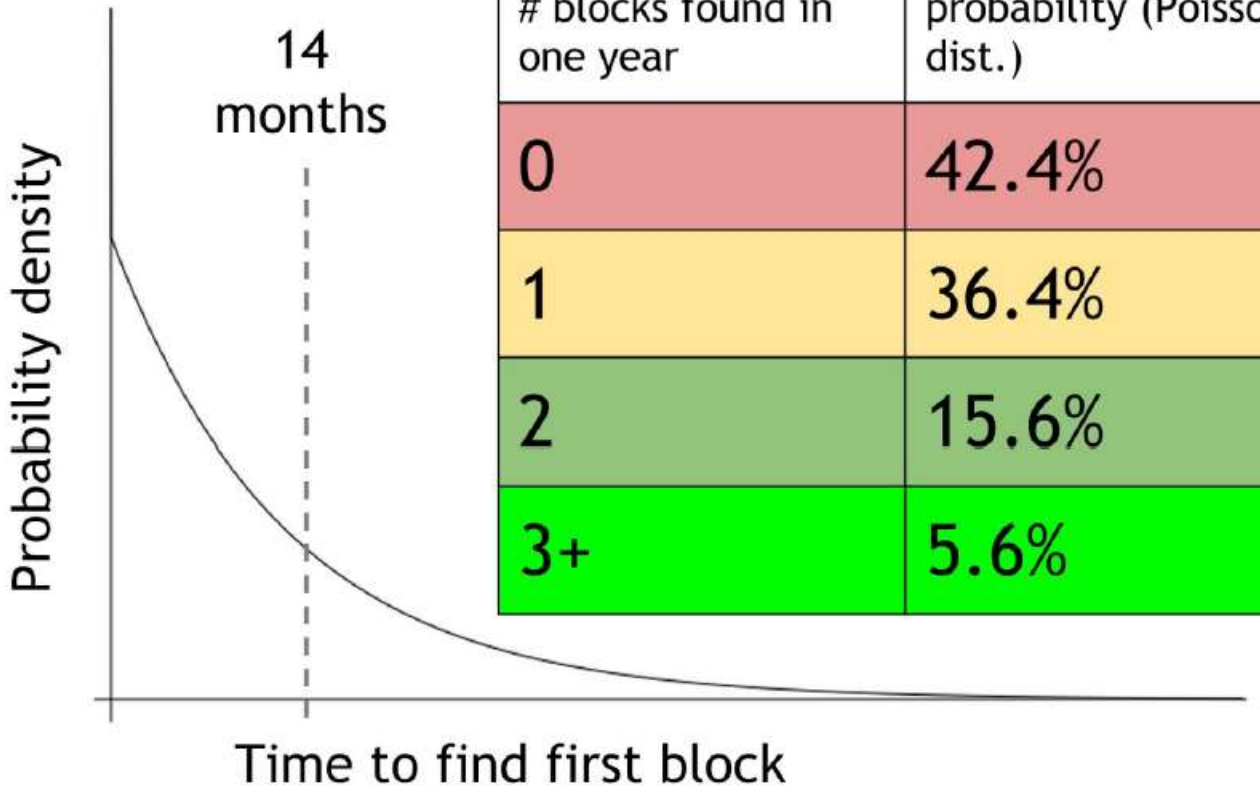
- 上述只是默认的规则，但比特币网络节点是完全自主的，可能执行不同的策略，或恶意或无意或单纯地为了追求利益的。
- 当有双花攻击的时候，例如Alice几乎同时发起两笔交易（输出分别是Bob和Charlie）、使用同一个输入。一些节点先收到Alice→Bob的交易，另一些节点先收到Alice→Charlie的交易。每个节点在收到另外一笔交易时，会怀疑双花而把后来的这笔交易拒绝（丢掉、不再处理）。结果就是众多节点对“将哪个交易放入区块链”产生分歧，这称为“竞争状态”。
  - 这个问题在矿工挖矿、区块上链时得到解决：挖矿成功的节点决定了哪个交易放入区块链。
- 区块也有竞争状态的限制：当有两个有效的区块同时被挖到时，只有其中一个可以进入区块链，哪个区块被最终纳入区块链，取决于其他节点选择在哪个区块上扩展区块链。

7. 比特币：从交易到上链

- 每个节点本地的默认策略：
  - 收到两个处于竞争状态（双花）的交易时，采用“先到原则”，把后收到的当做“无效”处理
  - 当本地链因为收到新的区块而产生分叉（竞争状态）时，采用“先到原则”和“最长链原则”：如果两个分叉链的长度一样，采用先到原则；两个分叉链长度不一样时，采用最长链原则。

8. 比特币中的矿池

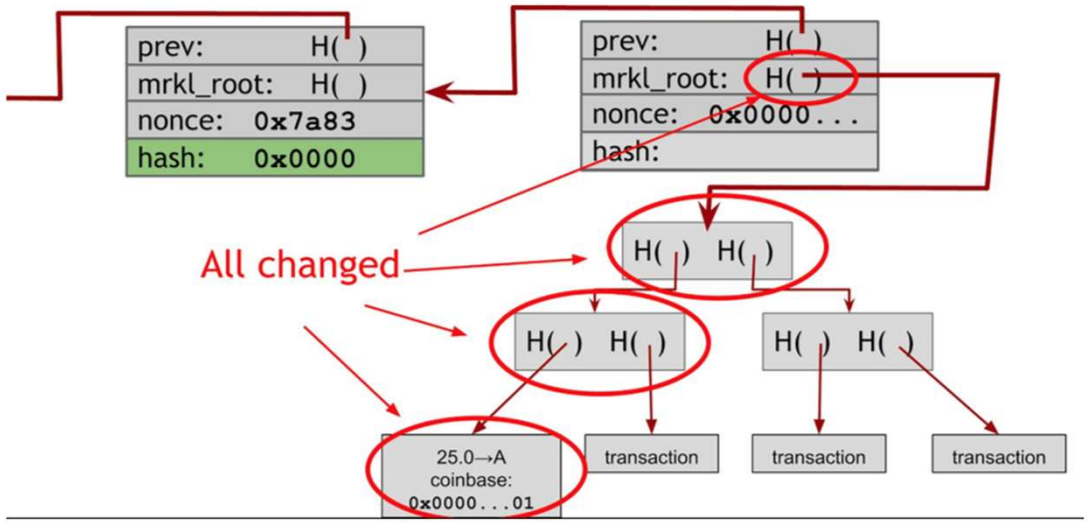
- 小矿工的风险很高



# blocks found in one year	probability (Poisson dist.)
0	42.4%
1	36.4%
2	15.6%
3+	5.6%

8. 比特币中的矿池

- 矿池：多个节点合作协同挖矿
- 矿池管理员把备选块的PreviousBlockHash决定好、coinbase交易的区块奖励接收地址决定好，向参与矿池的节点分发，共同挖矿。
  - 交易的验证、打包（建Merkle Tree）都是是由矿池管理员负责的
  - 矿池参与者只负责计算Hash、寻找nonce值

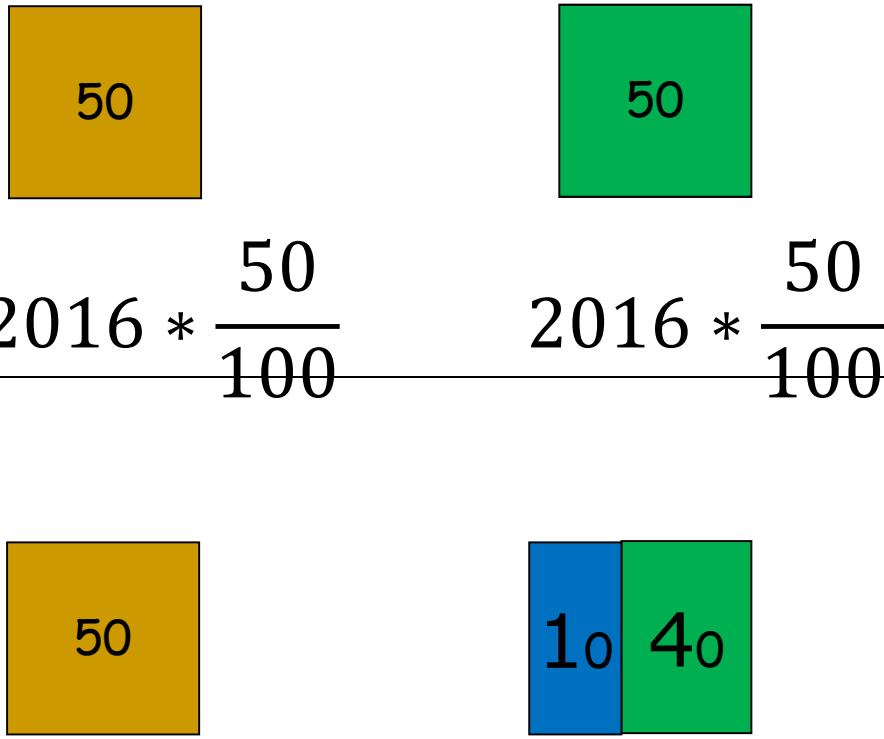


8. 比特币中的矿池

- 矿池的分红方式：
  - 工分分红
    - 管理员会对每一个超过特定区块难度的“区块”（工分）发放固定的奖励分红
    - 矿工发送工分之后，管理员马上对其支付奖励，不需要等到矿池发现一个有效区块
      - 对矿工有利，矿池管理员承担了一定的风险，管理员可以收取更高的管理费
  - 按实际比例分红
    - 每次找到一个有效的区块后，管理员把区块奖励按照每个矿工的实际工作量（工分）来按比例分配

8. 比特币中的矿池

- 矿池的丢弃区块攻击：



8. 比特币中的矿池

- 矿池的丢弃区块攻击：

$$2016 * \frac{50}{100} \qquad 2016 * \frac{50}{100}$$

发现有效区块时不提交

$$2016 * \frac{50}{90} * \frac{50}{60} \qquad 2016 * \frac{50}{90} * \frac{10}{60} + 2016 * \frac{40}{90}$$

8. 比特币中的矿池

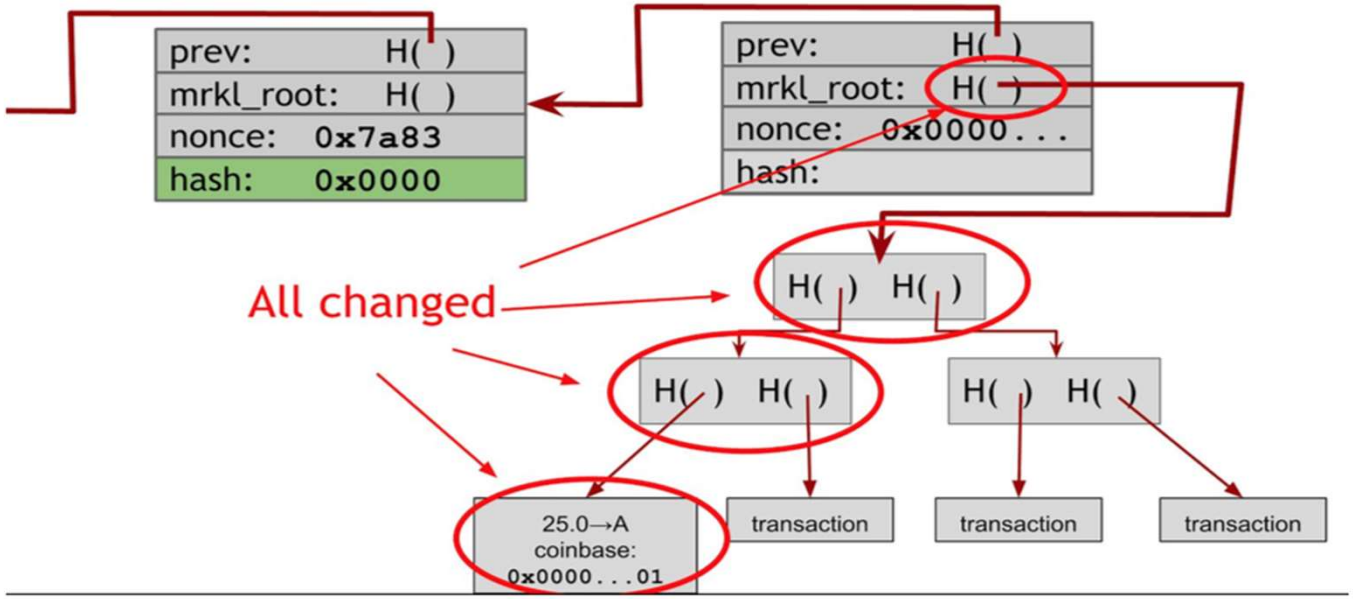
- “大”矿池的风险：
  - 当单个矿池的规模达到“大多数”时，会引起比特币社区、关注者对比特币系统的安全性的担心，进而导致比特币价格的下跌
  - 矿池会主动避免变的“过大”
  - 事实：实际上的算力集中在几个大的挖矿机构手上，“洗算力”避免引起关注和担心
    - 同时参与多个不同的矿池来掩盖

8. 比特币中的矿池

- 矿池的存在有益吗？
  - 好处：
    - 使得矿工挖矿收益变得更加容易预测，让小矿工更加容易参与
    - 每一个矿池都有一个中心化的矿池管理员在网络中组装区块，这使得网络更新变得更加容易
  - 坏处：
    - 中心化管理，矿池管理员可能掌握非常庞大的算力
      - 受贿而发动双花攻击等
    - 减少了比特币网络上校验全部交易节点的数量（全节点）

8. 比特币中的矿池

- 设计不能外包的挖矿机制：
  - 能形成矿池的原因：nonce的寻找与其他数据的设置是分离的，从而可以管理员预设其他数据，而矿工只负责需按照nonce
    - $H(prevhash, mr, nonce) \leq target$
  - 阻止矿池：使挖矿的人必须知道coinbase交易的输出地址对应的私钥
    - $H(prevhash, mr, nonce, \sigma) \leq target$
      $s.t. Verify(PK_{cb}, (prevhash, mr, nonce), \sigma) = 1$



9. 比特币交易处理能力

- 每个块大小为1MB
- 每个交易大约是250Byte，所以每个块最多4000个交易
- 每10分钟一个块，意味着每秒4000/600 约等于7笔交易
- Visa每秒2000笔，峰值每秒10000笔
- 提高比特币的交易处理速度是一个重要的工作方向
  - 有些**AltCoin**就是以提高交易吞吐率为卖点的
  - 简单的把块的大小调整大一些可以吗？
  - 提高块产生的速度？

10. 比特币的协议升级

- 功能改进、算法升级等等会对比特币有何影响？
  - 在比特币网络里（去中心化的分布式网络），不是简单的发布和更新版本的事情
- 硬分叉
  - 通过修订协议引入新的特性，运行新版本协议的节点认定有效的一部分区块，会被运行旧版本协议的节点认定为无效。
  - 很快最长区块链分支里包含的某些区块会被老节点认定为无效区块。老节点会认为其他的分支才是最长、有效的区块链分支，并一直扩展这个分支。此时区块链出现硬分叉。
  - 硬分叉使得原先的链分裂，网络上的节点会根据其所运行的协议版本去扩展两条不同的链，每个节点只能在这两个分叉中二选一，这两个分叉再也不会合并。
  - 可能触发硬分叉案例
    - 添加新的操作码
    - 区块大小改变（最近的比特币硬分叉案例）
    - 改变挖矿速率
    - 修正若干个bug

10. 比特币的协议升级

- 功能改进、算法升级等等会对比特币有何影响？
  - 在比特币网络里（去中心化的分布式网络），不是简单的发布和更新版本的事情

- 软分叉
  - 通过修订协议加入新的特性，让现有核验规则更为严格。这样老节点依然会接受所有的区块，而新节点有可能拒绝老节点挖出的部分区块。
  - 在这种情况下，老节点可能挖出一些无效的区块——这些区块中包含一些在新规则下无法核验通过的交易，从而被新节点拒绝在其上面延伸区块链。这种情形能够被老节点发现并更新其协议。
  - 而且，如果新节点用它们的区块扩展了老节点的分支，那么老节点也会转而支持这个分支，原因是新节点核验通过的区块，老节点也能核验通过。
  - 这样不会出现硬分叉，只是会有很多临时的小型分叉而已。
  - 软分叉的引入依赖于足够多的节点升级到新版本来实施新的核验规则。
  - 可能触发软分叉案例
    - 如P2SH，老节点只需验证hash，而新节点会额外执行脚本验证

11. 比特币中的攻击行为

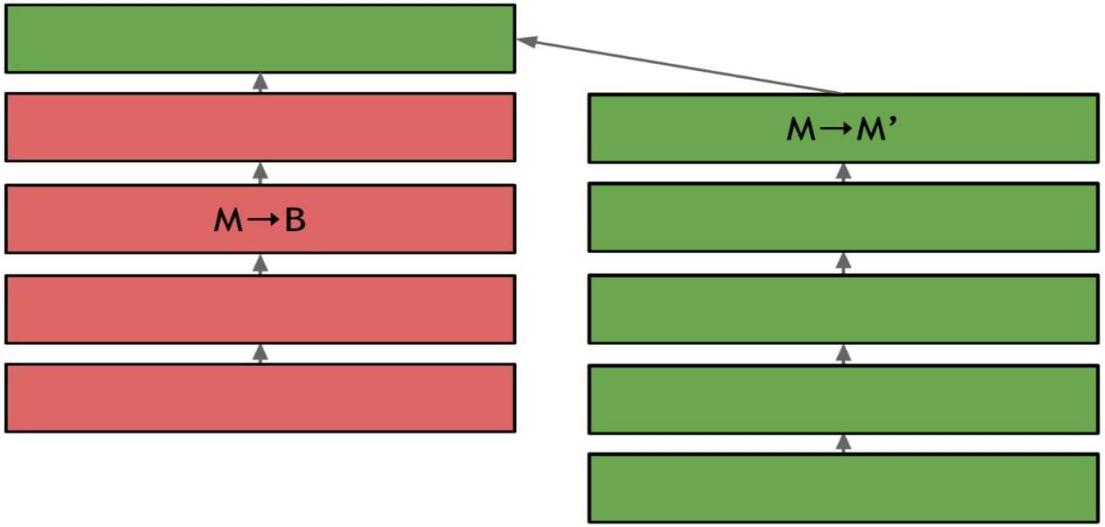
- 比特币中节点的默认原则：
  - 需要包含哪些交易？
    - 矿工可以选择将哪些交易放进他的区块里。默认是选择交易费高的交易。
  - 对哪一个区块进行挖矿运算？
    - 默认的做法是在最长的那条区块链上继续挖下去。
  - 在同一高度的多个区块中做选择如果两个不同的区块在同一时间被宣布发现，此时造成了一个区块的分叉，每个区块都是可以被延续下去的。
    - 默认的做法是选择最先被监听到的那一个区块。
  - 什么时候宣布新的区块？
    - 默认做法是立刻宣布，也可以选择等一下再宣布。

11. 比特币中的攻击行为

- 大多数的矿工是“诚实”的、按照默认策略运行；但是有些矿工会尝试使用一些非默认策略来使得自己的挖矿收益更高。
- 这些非默认策略可能损害了其他人的利益，我们将它们称之为攻击：
  - 分叉攻击
  - 通过贿赂进行分叉攻击
  - 自私挖矿（临时保留区块攻击）

11. 比特币中的攻击行为

- 分叉攻击：
  - 通过分叉攻击实现双重支付：
    - 恶意矿工给Bob发送了一些比特币来购买其服务和货品
    - 矿工进行一个分叉攻击，创建一个包含冲突交易的更长的分叉，在新的共识链中给Bob的支付就变成了无效的交易。



11. 比特币中的攻击行为

- 通过贿赂实现分叉攻击：
  - 通过购买矿机达到算力的“大多数”实现分叉攻击代价太大
  - 通过贿赂进行分叉攻击的核心思路：有别于直接获得大量算力，攻击者去贿赂那些已经拥有算力的人，让他们帮助自己分叉出另外一条最长的区块链
    - 只需让“帮忙一段时间”
  - 贿赂的方式：
    - 找到一些矿工直接用现金来贿赂
    - 创建一个新的矿池，提供更好的奖励吸引其他矿工，积累算力到一定程度后发起一次成功的分叉攻击

11. 比特币中的攻击行为

- 自私挖矿：
  - 又称为“临时保留区块攻击”
  - 攻击者找到一个有效区块的时候，先不立刻宣布，然后在这块上面继续挖矿，期望自己可以在其他矿工找到下一个区块之前连续找到两个有效区块，在整个过程中秘密地的保留你所发现的区块。
  - 运气好的话，确实在别人广播出一个块之前找到了两个块，当收到别人广播的块时，立即把自己的连续两个块广播出去。

小结

- 1 比特币的账本
  - Transaction-based
  - 签名
  - 公钥做身份
  - 隐私保护
  - Hash值做地址
- 2 比特币的交易
  - 比特币交易的输入、输出
  - 比特币交易的脚本
- 3 比特币的脚本语言
  - 比特币交易的脚本验证
  - 脚本语言的应用

小结

- 4 比特币的区块
  - Hash指针、Hash链
  - Merklee Tree
  - 区块头
  - Coinbase交易
- 5 比特币挖矿
  - 挖矿难度
  - 挖矿原理
- 6 比特币网络
  - P2P网络
  - 泛洪算法

小结

- 7 比特币：从交易到上链
  - 交易、块、挖矿、上链
- 8 比特币的矿池
  - 矿池原理
  - 矿池分红
  - 矿池优缺点
  - 组织矿池
- 9 比特币的交易处理能力
  - 受制于块的大小
- 10 比特币的协议升级
  - 硬分叉与软分叉
- 11 比特币中一些攻击行为