VICTORLAMP

区块链技术核心篇
之四：比特币区块链核心架构

e休，爱编程的葫芦娃：exiu@victorlamp.com

煤油灯科技公司 http://www.victorlamp.com
版权所有，仅供个人学习用，不许用于商业目的，不许上载到victorlamp之外的共享平台再次分发。

多媒体课程：《深入浅出区块链技术核心篇》

深入浅出区块链技术

课程介绍

《区块链技术基础篇之一：白话非对称加解密》
《区块链技术基础篇之二：白话哈希算法》
《区块链技术基础篇之三：白话P2P网络》
《区块链技术基础篇之四：白话拜占庭将军问题》
《区块链技术核心篇之一：比特币区块链起源及原理》
《区块链技术核心篇之二：比特币区块链密钥与地址》
《区块链技术核心篇之三：比特币区块链交易共识》
《区块链技术核心篇之四：比特币区块链核心架构》

讲师介绍

网名：一休

2015 年开始，从事区块链技术开发，先后成功的研发出：区块链技术的数字版权管理（DRM）系统、基于区块链IPFS的CDN产品开发。

是多项区块链技术专利发明人。

对于比特币区块链、IPFS项目源码非常熟悉。

本人负责的基于区块链的创新技术方案，获得了当年华为公司年度十大发明奖。

VICTORLAMP 煤油灯科技

区块链为什么能够发挥作用，本质的原因是周期性创建新的区块，新区块不断的加入到区块链里面，形成一种群体性参与的“永动机”。

以去中心化的动态性（类似于构建了一个移动靶）来应对各种恶意的攻击，保证区块链的安全可信。

本课程介绍比特币区块链里面的新区块是怎么构建的？新区块如何加入到区块链里面？对应到比特币，也就是大家熟悉的比特币的全过程，以及相关的技术架构。

VICTORLAMP
煤油灯科技

比特币区块链常用术语表

地址 :比特币地址(例如:1DSrfJdB2AnWaFNgSbv3MZC2m74996JafV)由一串字符和数字组成，以阿拉伯数字"1"开头。就像 别人向你的email地址发送电子邮件一样，他可以通过你的比特币地址向你发送比特币。
比特币 :"比特币"既可以指这种虚拟货币单位，也指比特币网络或者网络节点使用的比特币软件。
区块 :一个区块就是若干交易数据的集合，它会被标记上时间戳和之前一个区块的独特标记。区块头经过哈希运算后会生成一份工 作量证明，从而验证区块中的交易。有效的区块经过全网络的共识后会被追加到主区块链中。
区块链 ： 区块链是一串通过验证的区块，当中的每一个区块都与上一个相连，一直连到创世区块。
确认 ： 当一项交易被区块收录时，我们可以说它有一次确认。矿工们在此区块之后每再产生一个区块，此项交易的确认数就再加一。当确认数达到六及以上时，通常认为这笔交易比较安全并难以逆转。
难度 ： 整个网络会通过调整"难度"这个变量来控制生成工作量证明所需要的计算力。
难度目标 ： 使整个网络的计算力大致每10分钟产生一个区块所需要的难度数值即为难度目标。
难度调整 ： 整个网络每产生2,106个区块后会根据之前2,106个区块的算力进行难度调整。
矿工费 ： 交易的发起者通常会向网络缴纳一笔矿工费，用以处理这笔交易。大多数的交易需要0.5毫比特币的矿工费。
哈希 ： 二进制数据的一种数字指纹。
创世区块 ： 创世区块指区块链上的第一个区块，用来初始化相应的加密货币。
矿工 :矿工指通过不断重复哈希运算来产生工作量证明的各网络节点。
网络 :比特币网络是一个由若干节点组成的用以广播交易信息和数据区块的P2P网络。
工作量证明 :工作量证明指通过有效计算得到的一小块数据。具体到比特币，矿工必须要在满足全网目标难度的情况下求解SHA256算法。
奖励 :每一个新区块中都有一定量新创造的比特币用来奖励算出工作量证明的矿工。现阶段每一区块有6.25比特币的奖励。
私钥 :用来解锁对应(钱包)地址的一串字符，例如5J76sF8L5jTtzE96r66Sf8cka9y44wdpJjMwCxR3tzLh3ibVPxh。
交易 :简单地说，交易指把比特币从一个地址转到另一个地址。更准确地说，一笔"交易"指一个经过签名运算的，表达价值转移的数据结构。每一笔"交易"都经过比特币网络传输，由矿工节点收集并封包至区块中，永久保存在区块链某处。
钱包 :钱包指保存比特币地址和私钥的软件，可以用它来接受、发送、储存你的比特币。
BIP :比特币改进提议 (Bitcoin Improvement Proposals 的缩写)，指比特币社区成员所提交的一系列改进比特币的提议。例 如，BIP0021是一项改进比特币统一资源标识符(URI)计划的提议。

VICTORLAMP 煤油灯科技

什么是比特币区块链挖矿？

挖矿是增加比特币货币供应的一个过程。挖矿同时还保护着比特币系统的安全，防止欺诈交易，避免“双重支付”，“双重支付”是指多次花费同一笔比特币。矿工们通过为比特币网络提供算力来换取获得比特币奖励的机会。

矿工们验证每笔新的交易并把它们记录在总帐簿上。每10分钟就会有一个新的区块被“挖掘”出来，每个区块里包含着从上一 个区块产生到目前这段时间内发生的所有交易，这些交易被依次添加到区块链中。我们把包含在区块内且被添加到区块链上 的交易称为“确认”交易，交易经过“确认”之后，新的拥有者才能够花费他在交易中得到的比特币。

矿工们在挖矿过程中会得到两种类型的奖励:创建新区块的新币奖励，以及区块中所含交易的交易费。为了得到这些奖励， 矿工们争相完成一种基于加密哈希算法的数学难题，这些难题的答案包括在新区块中，作为矿工的计算工作量的证明，被称 为”“工作量证明”。该算法的竞争的机制以及获胜者有权在区块链上进行交易记录的机制，这二者是比特币安全的基石。
新比特币的生成过程被称为挖矿是因为它的奖励机制被设计为速度递减模式，类似于贵重金属的挖矿过程。比特币的货币是 通过挖矿发行的，类似于中央银行通过印刷银行纸币来发行货币。矿工通过创造一个新区块得到的比特币数量大约每四年 (或准确说是每210,000个块)减少一半。开始时为2009年1月每个区块奖励50个比特币，然后到2012年11月减半为每个区 块奖励25个比特币。之后将在2016年的某个时刻再次减半为每个新区块奖励12.5个比特币。基于这个公式，比特币挖矿奖励 以指数方式递减，直到2140年。届时所有的比特币(20,999,999.98)全部发行完毕。换句话说在2140年之后，不会再有新 的比特币产生。

矿工们同时也会获取交易费。每笔交易都可能包含一笔交易费，交易费是每笔交易记录的输入和输出的差额。在挖矿过程中 成功“挖出”新区块的矿工可以得到该区块中包含的所有交易“小费”。目前，这笔费用占矿工收入的0.5%或更少，大部分收益 仍来自挖矿所得的比特币奖励。然而随着挖矿奖励的递减，以及每个区块中包含的交易数量增加，交易费在矿工收益中所占 的比重将会逐渐增加。在2140年之后，所有的矿工收益都将由交易费构成。

尽管挖矿带来的奖励是一种激励，但它最主要的目的并不是奖励本身或者新币的产生。挖矿是一种将结算所去中心化的过程，每个结算所对处理的交易进行验证和结 算。挖矿保护了比特币系统的安全，并且实现了在没有中心机构的情况下，也能使整个比特币网络达成共识。

挖矿这个发明使比特币变得很特别，这种去中心化的安全机制是点对点的电子货币的基础。铸造新币的奖励和交易费是一种激励机制，它可以调节矿工行为和网络安全，同时又完成了比特币的货币发行。

VICTORLAMP 煤油灯科技

回顾区块链结构

区块链是由包含交易信息的区块从后向前有序链接起来的数据结构（区块）。它可以被存储为flat file(一种包含没有相对关系记录的 文件)，或是存储在一个简单数据库中。

比特币核心客戶端（Bitcoin Core）使用Google的LevelDB数据库存储区块链元数据。区块被从后向前有序地链接在这个链条里，每个区块都指向前一个区块。区块链经常被视为一个垂直的栈，第一个区块作为栈底的首区 块，随后每个区块都被放置在其他区块之上。用栈来形象化表示区块依次堆叠这一概念后，我们便可以使用一些术语，例 如:“高度”来表示区块与首区块之间的距离;以及“顶部”或“顶端”来表示最新添加的区块。

对每个区块头进行SHA256加密哈希，可生成一个哈希值。通过这个哈希值，可以识别出区块链中的对应区块。同时，每一 个区块都可以通过其区块头的“父区块哈希值”字段引用前一区块(父区块)。也就是说，每个区块头都包含它的父区块哈希值。这样把每个区块链接到各自父区块的哈希值序列就创建了一条一直可以追溯到第一个区块(创世区块)的链条。

由于区块头里面包含“父区块哈希值”字段，所以当前区块的哈希值因此也受到该字段的影响。如果父区块的身份标识发生变 化，子区块的身份标识也会跟着变化。当父区块有任何改动时，父区块的哈希值也发生变化。父区块的哈希值发生改变将迫 使子区块的“父区块哈希值”字段发生改变，从而又将导致子区块的哈希值发生改变。而子区块的哈希值发生改变又将迫使孙 区块的“父区块哈希值”字段发生改变，又因此改变了孙区块哈希值，等等以此类推。一旦一个区块有很多代以后，这种瀑布 效应将保证该区块不会被改变，除非强制重新计算该区块所有后续的区块。正是因为这样的重新计算需要耗费巨大的计算 量，所以一个长区块链的存在可以让区块链的历史不可改变，这也是比特币安全性的一个关键特征。

区块是一种被包含在公开账簿(区块链)里的聚合了交易信息的容器数据结构。它由一个包含元数据的区块头和紧跟其后的 构成区块主体的一长串交易组成。区块头是80字节，而平均每个交易至少是250字节，而且平均每个区块至少包含超过500 个交易。因此，一个包含所有交易的完整区块比区块头的1000倍还要大。

VICTORLAMP 煤油灯科技

区块结构和区块头结构

一个区块block

字段	大小	描述
区块大小	4字节	用字节表示的该字段之后的区块大小
区块头	80字节	组成区块头的几个字段
交易计数器	1-9 (可变整数)	交易的数量
交易	可变的	记录在区块里的交易信息

字段	大小	描述
版本	4字节	版本号，用于跟踪软件/协议的更新
父区块哈希值	32字节	引用区块链中父区块的哈希值
Merkle根	32字节	该区块中交易的merkle树根的哈希值
时间戳	4字节	该区块产生的近似时间(精确到秒的Unix时间戳)
难度目标	4字节	该区块工作量证明算法的难度目标
Nonce	4字节	用于工作量证明算法的计数器

新区块的创建也就包括两件事情：1）如何正确的填写区块的字段内容，包括区块头；2）填写好的区块如何加入到区块链里面去。

VICTORLAMP 煤油灯科技

什么是Merkle树

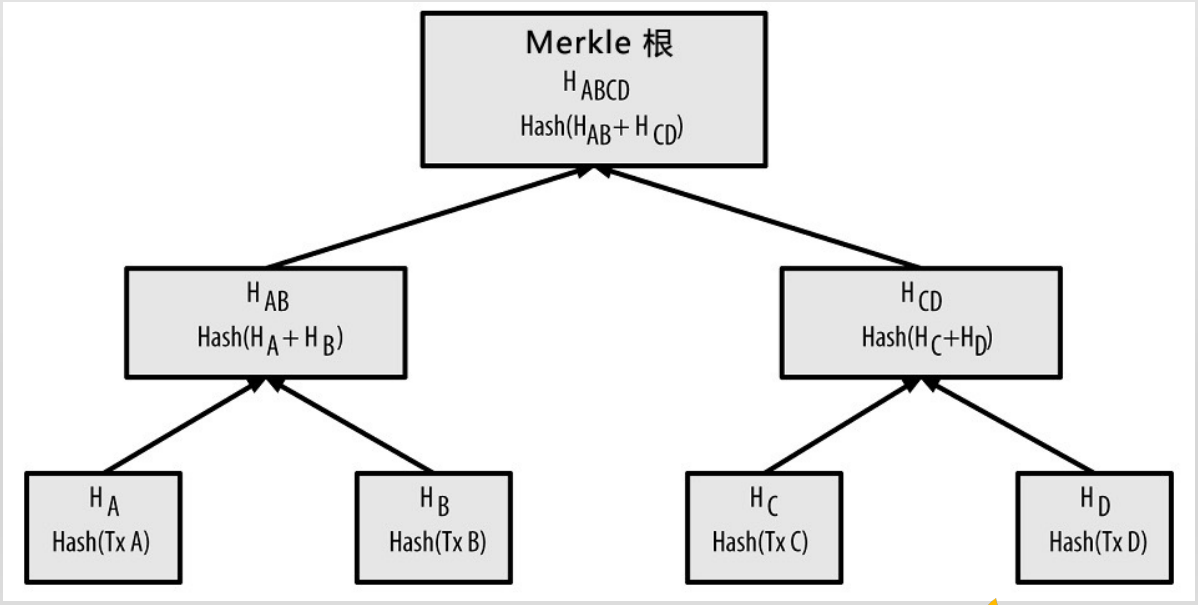
区块链中的每个区块都包含了产生于该区块的所有交易，且以Merkle树表示。
Merkle树是一种哈希二叉树，它是一种用作快速归纳和校验大规模数据完整性的数据结构。这种二叉树包含加密哈希值。

在比特币网络中，Merkle树被用来归纳一个区块中的所有交易，同时生成整个交易集合的数字哈希值，且提供了一种校验区块是否存在某交易的高效途径。生成一棵完整的Merkle树需要递归地对哈希节点对进行哈希，并将新生成的哈希节点插入到 Merkle树中，直到只剩一个哈希节点，该节点就是Merkle树的根。在比特币的Merkle树中两次使用到了SHA256算法，因此其加密哈希算法也被称为double-SHA256
。
当N个数据元素经过加密后插入Merkle树时，你至多计算2*log2(N)次就能检查出任意某数据元素是否在该树中，这使得该数 据结构非常高效。
Merkle树是自底向上构建的。在如下的例子中，我们从A、B、C、D四个构成Merkle树树叶的交易开始，如图7-2。起始时所 有的交易都还未存储在Merkle树中，而是先将数据哈希化，然后将哈希值存储至相应的叶子节点。
这些叶子节点分别是 H (A) 、H (B) 、H (C)和H (D) 。 如：H(A) = SHA256(SHA256(交易A))

通过串联相邻叶子节点的哈希值然后哈希之，这对叶子节点随后被归纳为父节点。 例如，为了创建父节点H (AB)，子节点A和 子节点B的两个32字节的哈希值将被串联成64字节的字符串。

随后将字符串进行两次哈希来产生父节点的哈希值:
H (AB) =SHA256(SHA256(H (A) + H (B)))

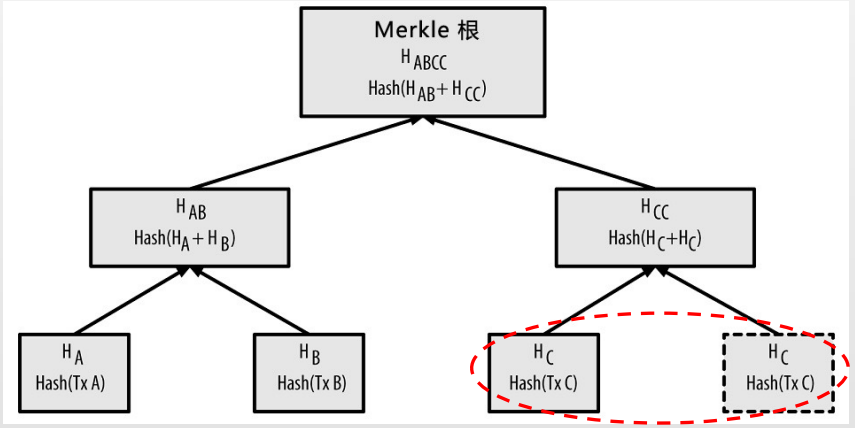
继续类似的操作直到只剩下顶部的一个节点，即Merkle根。产生的32字节哈希值存储在区块头，同时归纳了四个交易的所有 数据。



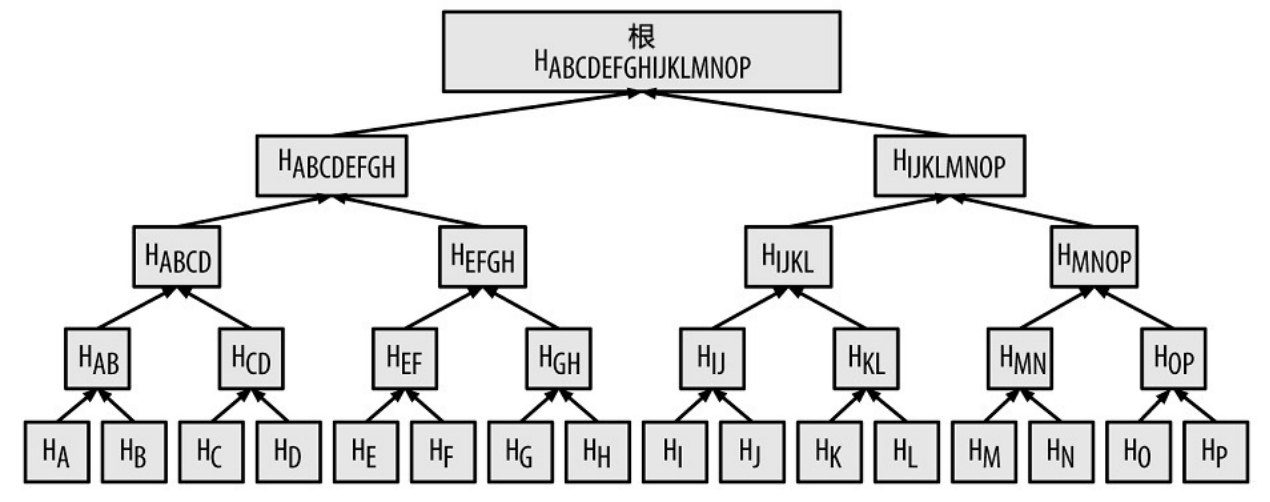
VICTORLAMP 煤油灯科技

区块头结构Merkle根

因为Merkle树是二叉树，所以它需要偶数个叶子节点。如果仅有奇数个交易需要归纳，那最后的交易就会被复制一份以构成 偶数个叶子节点，这种偶数个叶子节点的树也被称为平衡树。



在比特币中，在单个区块中有成百上千的交易是 非常普遍的，这些交易都会采用同样的方法归纳起来，产生一个仅仅32字节的数据作为Merkle根。



VICTORLAMP 煤油灯科技

区块头结构的Merkle树有什么用处

要解决的问题是：

1）区块链全网的历史总交易纪录数量巨大（近10亿条纪录，每时每刻都还在增长），很多时候需要快速的查询某条交易记录在那个区块中变得非常困难，如何快速的确定某个交易TX在哪个区块中？

2）比特币区块链全节点的容量已经非常大（接近400G），对于有些设备（如智能手机）来说要存储所有这些数据非常困难，成本也很大。

解决方案：

1）有些节点只要存储区块头就可以，存储的数量大大降低（现在区块高度60多万，每个区块头80字节，总计就是50多M），大大降低设备的存储容量要求；

2） Merkle根存储在区块头里面，通过Merkle树查找可以确定某个交易记录是否在一个区块的交易记录列表里，通过获取Merkle树的局部验证路径就可以验证这个交易是否在这个区块头所属的区块里面。

接下来分两种场景：

a)如果只是需要查询某个交易是否在已经记录在区块链的交易记录中，那只要进行Merkle根验证即可，求助全量节点告诉我在哪个区块，然后返回给我Merkle树验证路径即可，有这个验证路径我就可以进行本地再验证；

b)如果需要查询到具体交易内容，那就求助全量节点返回给我具体的这一个区块内容体，我在本地就可以进行查找具体内容。

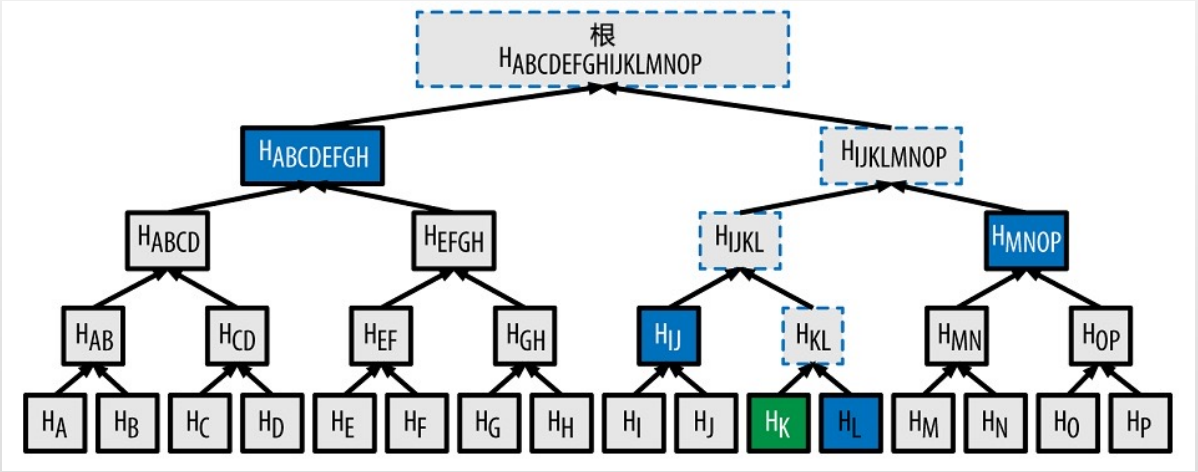
所以说，Merkle树在比特币区块链中的用处：1）轻量化存储区块数据；2）快速验证查找一个交易。

VICTORLAMP 煤油灯科技

区块头结构的Merkle树有什么用处

为了证明区块中存在某个特定的交易，一个节点只需要计算log2(N)个32字节的哈希值，形成一条从特定交易到树根的认证路 径或者Merkle路径即可。随着交易数量的急剧增加，这样的计算量就显得异常重要，因为相对于交易数量的增长，以基底为 2的交易数量的对数的增长会缓慢许多。这使得比特币节点能够高效地产生一条10或者12个哈希值(320-384字节)的路 径，来证明了在一个巨量字节大小的区块中上千交易中的某笔交易的存在。

图中，一个节点能够通过生成一条仅有4个32字节哈希值长度(总128字节)的Merkle路径，来证明区块中存在一笔交易K。该路径有4个哈希值(在图中由蓝色标注)H(L)、H(IJ)、H(MNOP)和H(ABCDEFGH)。由这4个哈希值产生的认证路径，再通过计算另外四对哈希值H(KL)、H(IJKL)、H(IJKLMNOP)和Merkle树根(在图中由虚线标注)，任何节点都能证明H(K)(在图 中由绿色标注)包含在Merkle根中，也就交易K在Merkle根所在的区块中。



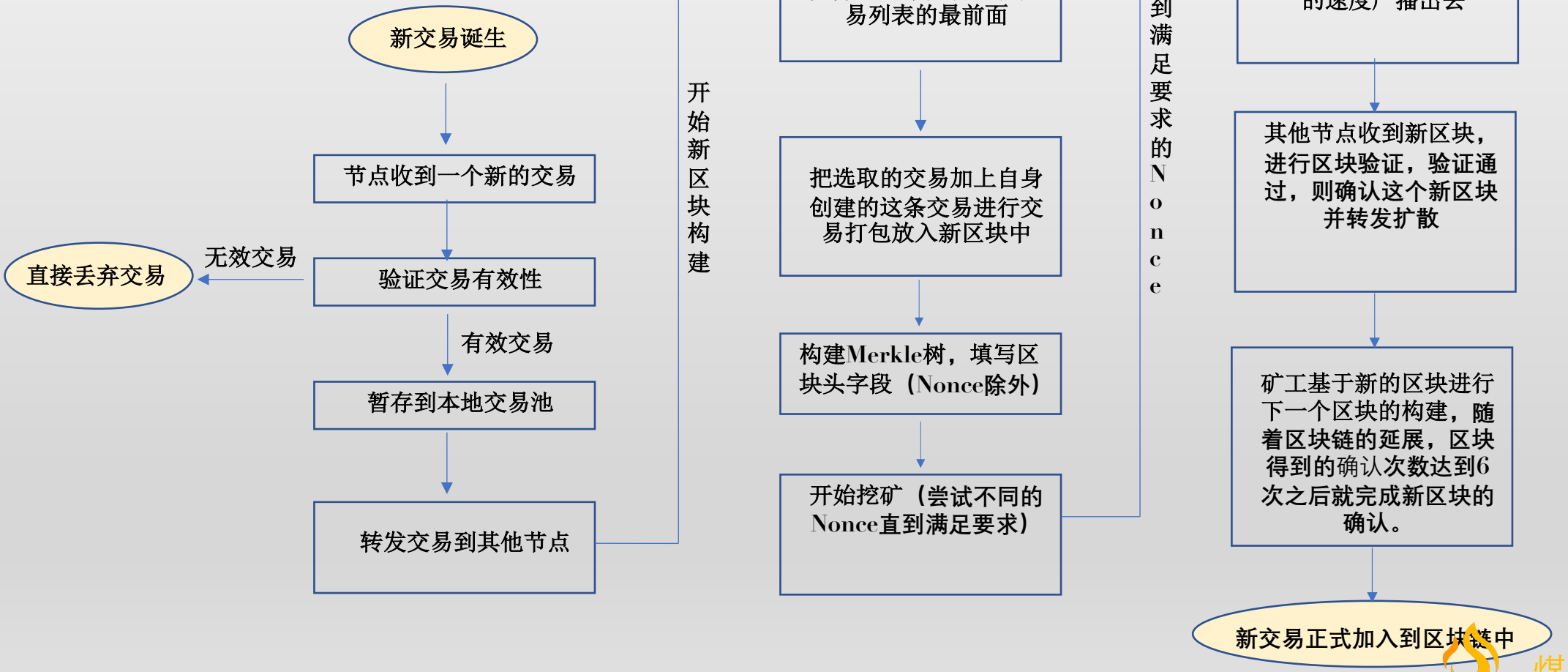
Merkle树的效率提升结果

交易数量	区块的近似大小	路径大小(哈希数量)	路径大小(字节)
16笔交易	4KB	4个哈希	128字节
512笔交易	128KB	9个哈希	288字节
2048笔交易	512KB	11个哈希	352字节
65,535笔交易	16MB	16个哈希	512字节

VICTORLAMP 煤油灯科技

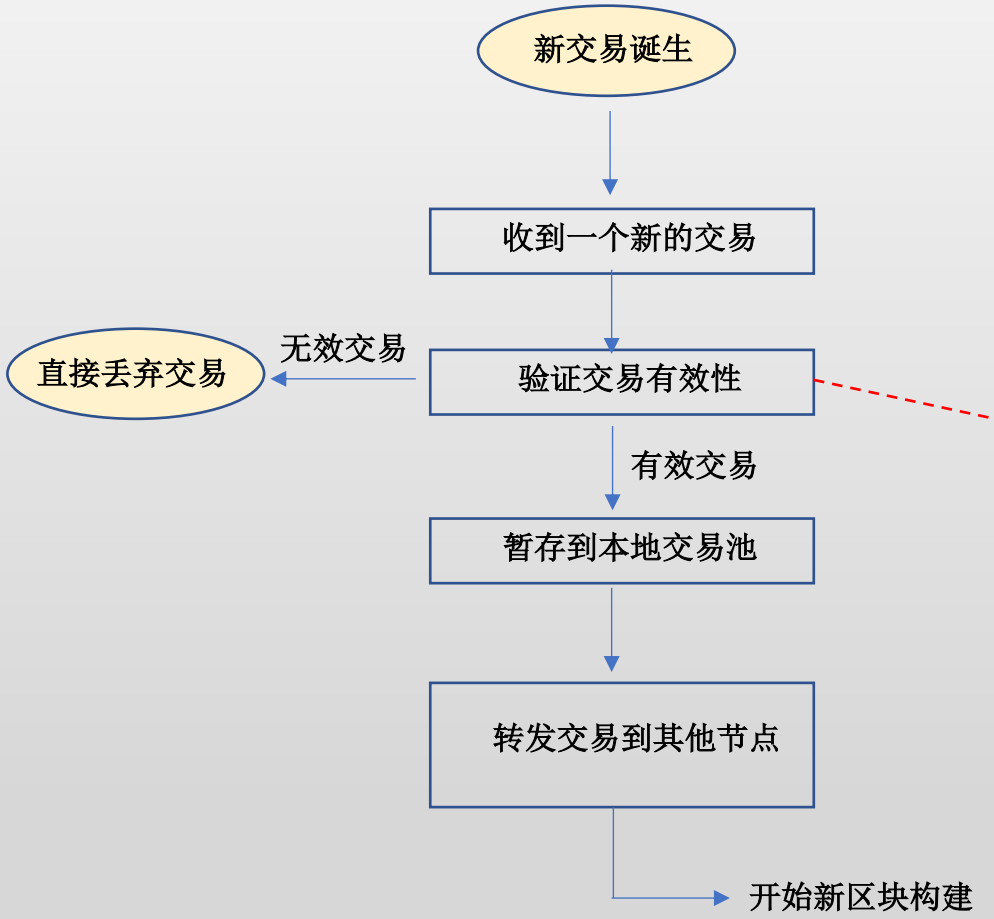
一个比特币交易TX加入到区块链的完整旅程

说明：
1）开始区块构建前的事情是每个全量节点都做的事情，新区块构建是挖矿节点才做的事情。
2）交易打包中，区块中的第一笔交易是笔特殊交易，称为创币交易或者coinbase交易。



煤油灯科技
VICTORLAMP

区块体交易验证

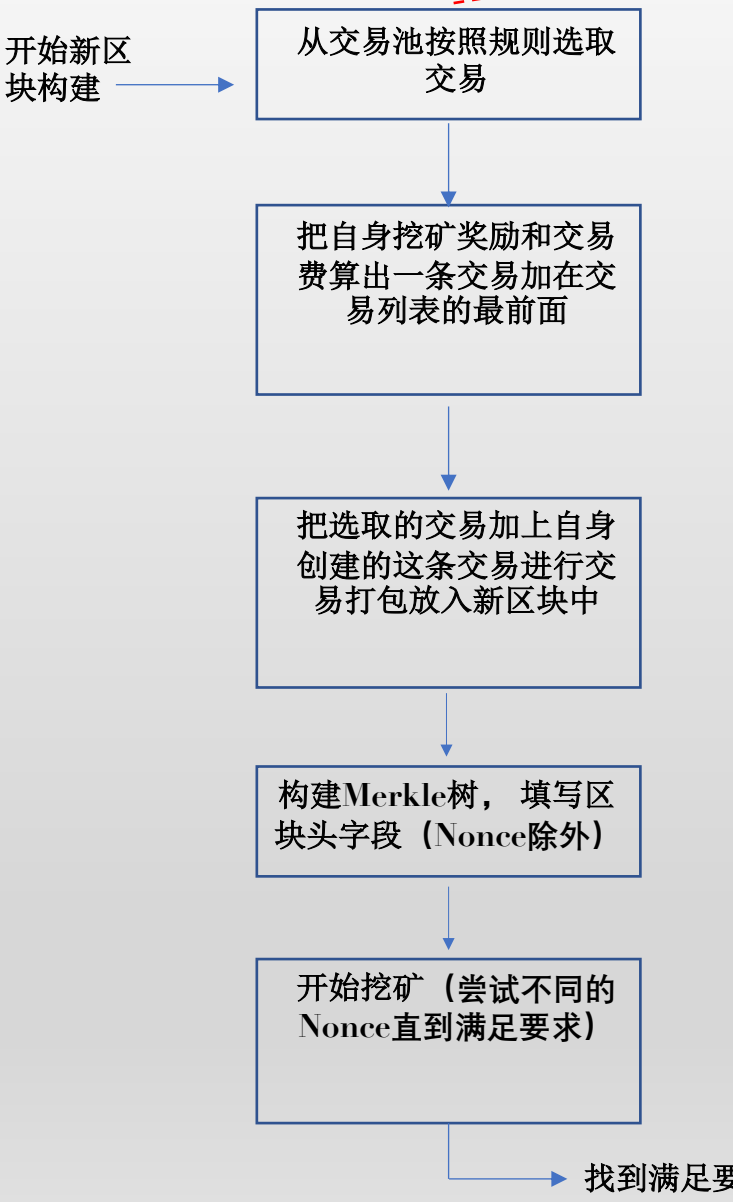


每一个节点在校验每一笔交易时，都需要对照长长的标准列表进行校验:

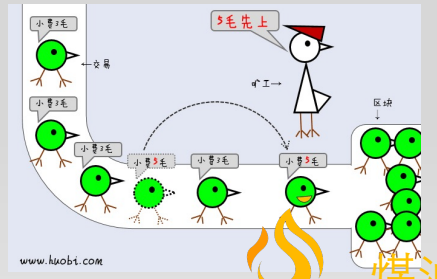
1. 交易的语法和数据结构必须正确。
2. 输入与输出列表都不能为空。
3. 交易的字节大小是小于 MAX_BLOCK_SIZE 的。
4. 每一个输出值，以及总量，必须在规定值的范围内 (小于2,100万个币，大于0)。
5. 没有哈希等于0，N等于-1的输入(coinbase交易不应当被中继)。
6. nLockTime是小于或等于 INT_MAX 的。
7. 交易的字节大小是大于或等于100的。
8. 交易中的签名数量应小于签名操作数量上限。
9. 解锁脚本(scriptSig)只能够将数字压入栈中，并且锁定脚本(scriptPubkey)必须要符合 isStandard 的格式 (该格式 将会拒绝非标准交易)。
10. 池中或位于主分支区块中的一个匹配交易必须是存在的。
11. 对于每一个输入，如果引用的输出存在于池中任何的交易，该交易将被拒绝。
12. 对于每一个输入，在主分支和交易池中寻找引用的输出交易。如果输出交易缺少任何一个输入，该交易将成为一个孤立的 交易。如果与其匹配的交易还没有出现在池中，那么将被加入到孤立交易池中。
13. 对于每一个输入，如果引用的输出交易是一个coinbase输出，该输入必须至少获得 COINBASE_MATURITY (100)个确认。
14. 对于每一个输入，引用的输出是必须存在的，并且没有被花费。
15. 使用引用的输出交易获得输入值，并检查每一个输入值和总值是否在规定值的范围内 (小于2100万个币，大于0)。
16. 如果输入值的总和小于输出值的总和，交易将被中止。
17. 如果交易费用太低以至于无法进入一个空的区块，交易将被拒绝。
18. 每一个输入的解锁脚本必须依据相应输出的锁定脚本来验证。

VICTORLAMP 煤油灯科技

区块体交易验证和打包



1. 比特币节点需要为内存池中的每笔交易分配一个优先级，并选择较高优先级的交易记录来构建候选区块。交易的优先 级是由交易输入所花费的UTXO的"块龄"决定，交易输入值高、"块龄"大的交易比那些新的、输入值小的交易拥有更高的优先 级。如果区块中有足够的空间，高优先级的交易行为将不需要矿工费。
2. 交易的优先级是通过输入值和输入的"块龄"乘积之和除以交易的总长度得到的: Priority = Sum (Value of input * Input Age) / Transaction Size
3. 在这个等式中，交易输入的值是由比特币单位"聪"(1亿分之1个比特币)来表示的。UTXO的"块龄"是自该UTXO被记录到区 块链为止所经历过的区块数，即这个UTXO在区块链中的深度。交易记录的大小由字节来表示。
4. 一个交易想要成为"较高优先级"，需满足的条件:优先值大于57,600,000，相当于一个比特币(即1亿聪)，年龄为一天 (144个区块)，交易的大小为250个字节: High Priority > 100,000,000 satoshis * 144 blocks / 250 bytes = 57,600,000
5. 区块中用来存储交易的前50K字节是保留给较高优先级交易的。在填充这50K字节的时候，会优先考虑这些最高优 先级的交易，不管它们是否包含了矿工费。这种机制使得高优先级交易即便是零矿工费，也可以优先被处理。
6. 然后，挖矿节点会选出那些包含矿工费的交易，并按照"每千字节矿工费"进行排序，优先选择矿工费高的交易来填 充剩下的区块，区块大小上限为 MAX_BLOCK_SIZE 。
7. 如区块中仍有剩余空间，挖矿节点可以选择那些不含矿工费的交易。有些矿工会竭尽全力将那些不含矿工费的交易整 合到区块中，而其他矿工也许会选择忽略这些交易。
8. 在区块被填满后，内存池中的剩余交易会成为下一个区块的候选交易。因为这些交易还留在内存池中，所以随着新的区块被加到链上，这些交易输入时所引用UTXO的深度(即交易"块龄")也会随着变大。由于交易的优先值取决于它交易输入的"块龄"，所以这个交易的优先值也就随之增长了。最后，一个零矿工费交易的优先值就有可能会满足高优先级的门槛，被免费地 打包进区块。
9. 比特币交易中没有过期、超时的概念，一笔交易现在有效，那么它就永远有效。然而，如果一笔交易只在全网广播了一次， 那么它只会保存在挖矿节点的内存中。因为内存池是以未持久化的方式保存在挖矿节点存储器中的，所以一旦这个节点重新启动，内存池中的数据就会被完全擦除。而且，即便一笔有效交易被传播到了全网，如果它长时间未处理，它将从挖矿 节点的内存池中消失。如果交易本应该在一段时间内被处理而实际没有，那么钱包软件应该重新发送交易或重新支付更高的矿工费。



煤油灯科技
VICTORLAMP

区块头难度目标和找随机数Nonce

用最简单的术语来说，构建新区块就是重复计算区块头的哈希值，不断修改Nonce参数，直到与哈希值匹配的一个过程。

哈希函数的结 果无法提前得知，也没有能得到一个特定哈希值的模式。哈希函数的这个特性意味着:得到哈希值的唯一方法是不断的尝 试，每次随机修改输入，直到出现适当的哈希值。

哈希函数的输入数据的长度是任意的，将产生一个长度固定且绝不雷同的值，可将其视为输入的数字指纹。

对于特定输入， 哈希的结果每次都一样，任何实现相同哈希函数的人都可以计算和验证。一个加密哈希函数的主要特征就是不同的输入几乎不可能出现相同的数字指纹。因此，相对于随机选择输入，有意地选择输入去生成一个想要的哈希值几乎是不可能的。无论输入的大小是多少，SHA256函数的输出的长度总是256bit。

目标值 = 最大目标值 / 难度值
其中，最大目标值为一个恒定值:
0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
目标值的大小与难度值成反比。比特币工作量证明的达成就是矿工计算出来的区块哈希值必须小于目标值。我们也可以简单理解成，比特币工作量证明的过程，就是通过不停地变换区块头（即尝试不同的随机值）作为输入进行SHA256哈希运算，找出一个特定格式哈希值的过程（即要求有一定数量的前导0）。而要求的前导0的个数越多，代表难度越大。

VICTORLAMP 煤油灯科技

区块头难度目标和找随机数Nonce（通过举例，容易理解）

计算hash值的python脚本

```
# example of iterating a nonce in a hashing algorithm's input import hashlib
text = "I am Satoshi Nakamoto"
# iterate nonce from 0 to 19
for nonce in xrange(20):
# add the nonce to the end of the text
input = text + str(nonce)
# calculate the SHA-256 hash of the input (text+nonce)
hash = hashlib.sha256(input).hexdigest() # show the input and hash result
print input, '=>', hash
```

通过迭代 nonce 来生成不同哈希值的脚本的输出

```
》python hash_example.py
I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...
I am Satoshi Nakamoto2 => ea758a8134b115298a1583ffb80ae629...
I am Satoshi Nakamoto3 => bfa9779618ff072c903d773de30c99bd...
I am Satoshi Nakamoto4 => bce8564de9a83c18c31944a66bde992f...
I am Satoshi Nakamoto5 => eb362c3cf3479be0a97a20163589038e...
I am Satoshi Nakamoto6 => 4a2fd48e3be420d0d28e202360cfbaba...
I am Satoshi Nakamoto7 => 790b5a1349a5f2b909bf74d0d166b17a...
I am Satoshi Nakamoto8 => 702c45e5b15aa54b625d68dd947f1597...
I am Satoshi Nakamoto9 => 7007cf7dd40f5e933cd89fff5b791ff0...
I am Satoshi Nakamoto10 => c2f38c81992f4614206a21537bd634a...
I am Satoshi Nakamoto11 => 7045da6ed8a914690f087690e1e8d66...
I am Satoshi Nakamoto12 => 60f01db30c1a0d4cbce2b4b22e88b9b...
I am Satoshi Nakamoto13 => 0ebc56d59a34f5082aaef3d66b37a66...
I am Satoshi Nakamoto14 => 27ead1ca85da66981fd9da01a8c6816...
I am Satoshi Nakamoto15 => 394809fb809c5f83ce97ab554a2812c...
I am Satoshi Nakamoto16 => 8fa4992219df33f50834465d3047429...
I am Satoshi Nakamoto17 => dca9b8b4f8d8e1521fa4eaa46f4f0cd...
I am Satoshi Nakamoto18 => 9989a401b2a3a318b01e9ca9a22b0f3...
I am Satoshi Nakamoto19 => cda56022ecb5b67b2bc93a2d764e75f...
```

1. 为了使这个哈希算法变得富有挑战，我们来设定一个具有任意性的目标:找到一个语句，使之哈希值的十六进制表示以0开 头。幸运的是，这很容易!在例8-10中语句 "I am Satoshi Nakamoto13" 的哈希值是
2. 0ebc56d59a34f5082aaef3d66b37a661696c2b618e62432727216ba9531041a5，刚好满足条件。我们得到它用了13次。用概率的角度来 看，如果哈希函数的输出是平均分布的，我们可以期望每16次得到一个以0开头的哈希值(十六进制每一位数字为0到F)。 从数字的角度来看，我们要找的是小于0x1000000000000000000000000000000000000000000000000000000000000000的哈希值。我们称这个为目标阀值， 我们的目的是找到一个小于这个目标的哈希值。如果我们减小这个目标值，那找到一个小于它的哈希值会越来越难。
3. 简单打个比方，想象人们不断扔一对色子以得到小于一个特定点数的游戏。第一局，目标是12。只要你不扔出两个6，你就 会赢。然后下一局目标为11。玩家只能扔10或更小的点数才能赢，不过也很简单。假如几局之后目标降低为了5。现在有一 半机率以上扔出来的色子加起来点数会超过5，因此无效。随着目标越来越小，要想赢的话，扔色子的次数会指数级的上升。 最终当目标为2时(最小可能点数)，只有一个人平均扔36次或2%扔的次数中，他才能赢。
4. 在例中，成功的nonce为13，且这个结果能被所有人独立确认。任何人将13加到语句 "I am Satoshi Nakamoto" 后面再 计算哈希值都能确认它比目标值要小。这个正确的结果同时也是工作量证明(Proof of Work)，因为它证明我们的确花时间找到了这个nonce。验证这个哈希值只需要一次计算，而我们找到它却花了13次。如果目标值更小(难度更大)，那我们需 要多得多的哈希计算才能找到合适的nonce，但其他人验证它时只需要一次哈希计算。此外，知道目标值后，任何人都可以 用统计学来估算其难度，因此就能知道找到这个nonce需要多少工作。
5. 比特币的工作量证明和例中的挑战非常类似。矿工用一些交易构建一个候选区块。接下来，这个矿工计算这个区块头信 息的哈希值，看其是否小于当前目标值。如果这个哈希值不小于目标值，矿工就会修改这个nonce(通常将之加1)然后再试一次。按当前比特币系统的难度，矿工得试10^15次(10的15次方)才能找到一个合适的nonce使区块头信息哈希值足够 小。

VICTORLAMP 煤油灯科技

区块头难度目标和难度值调整

比特币的区块平均每10分钟生成一个。这就是比特币的心跳，是货币发行速率和交易达成速度的基础。不仅是在短期内，而 是在几十年内它都必须要保持恒定。在此期间，计算机性能将飞速提升。此外，参与挖矿的人和计算机也会不断变化。为了 能让新区块的保持10分钟一个的产生速率，挖矿的难度必须根据这些变化进行调整。事实上，难度是一个动态的参数，会定 期调整以达到每10分钟一个新区块的目标。简单地说，难度被设定在，无论挖矿能力如何，新区块产生速率都保持在10分钟 一个。

挖矿就是不断尝试区块块头中的nonce的值，使得：H(block header)≤target

显然目标阈值target越小，则挖矿的难度就越大。所以调整挖矿难度就是在调整target，以调整目标空间在整个输出空间中所占的比例。比特币中使用的哈希函数是SHA-256，产生的哈希值是256位的，所以整个输出空间是2 256 2^{256}2256，调整目标空间所占的比例，在这个问题里直观的来看就是最后得到的哈希值前面有多少位0（这只是通俗直观来看的，也许第一个非0位是要小于4也说不定），这个0越多显然值就越小，也就是挖矿难度越大了。

挖矿难度和目标阈值的大小成反比：difficulty=difficulty_1_target/target

上式中常量difficulty_1_target是指挖矿难度difficulty定义为1时所对应的目标阈值target的值(对于比特币区块链， difficulty_1_target为一个恒定值：0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF)。挖矿难度最小就是1，所以这个常量也就是target允许的最大值。

怎么调整挖矿难度?
比特币协议中规定，每隔2016个区块（大约每2个星期）要重新调整一下目标阈值target，具体的迭代更新公式是：
target=target×(expected time/actual time)

这里expected time就是预期的两次调整的间隔时间，即2016乘以10分钟；而actual time是系统中产生最近的2016个区块实际花费的时间。
为了避免系统中出现某些意外情况，导致系统出现非常大的波动，每次对目标阈值target的调整最大不能超过4倍，最小不能小于1/4。也即上式中的actual time/expected time即便超过4了也按4使用，即便小于1/4也只按1/4使用。

VICTORLAMP 煤油灯科技

校验新区块

新区块广播出去之后，每个节点独立校验每个新区块。当新区块在网络中传播时，每一个节点在将它转发到其节点之前，会进行一系列的测试去验证它。这确保了只有有效的区块会在网络中传播。独立校验还确保了诚实的矿工生成的区块可以被纳入到区块链中，从而获得奖励。行为不诚实的矿工所产生的区块将被拒绝，这不但使他们失去了奖励，而 且也浪费了本来可以去寻找工作量证明解的机会，因而导致其电费亏损。 当一个节点接收到一个新的区块，它将对照一个长长的标准清单对该区块进行验证，若没有通过验证，这个区块将被拒绝。 这些标准可以在比特币核心客戶端的CheckBlock函数和CheckBlockHead函数中获得，它包括:

▷ 区块的数据结构语法上有效

▷ 区块头的哈希值小于目标难度(确认包含足够的工作量证明)

▷ 区块时间戳早于验证时刻未来两个小时(允许时间错误)

▷ 区块大小在长度限制之内

▷ 第一个交易(且只有第一个)是coinbase交易

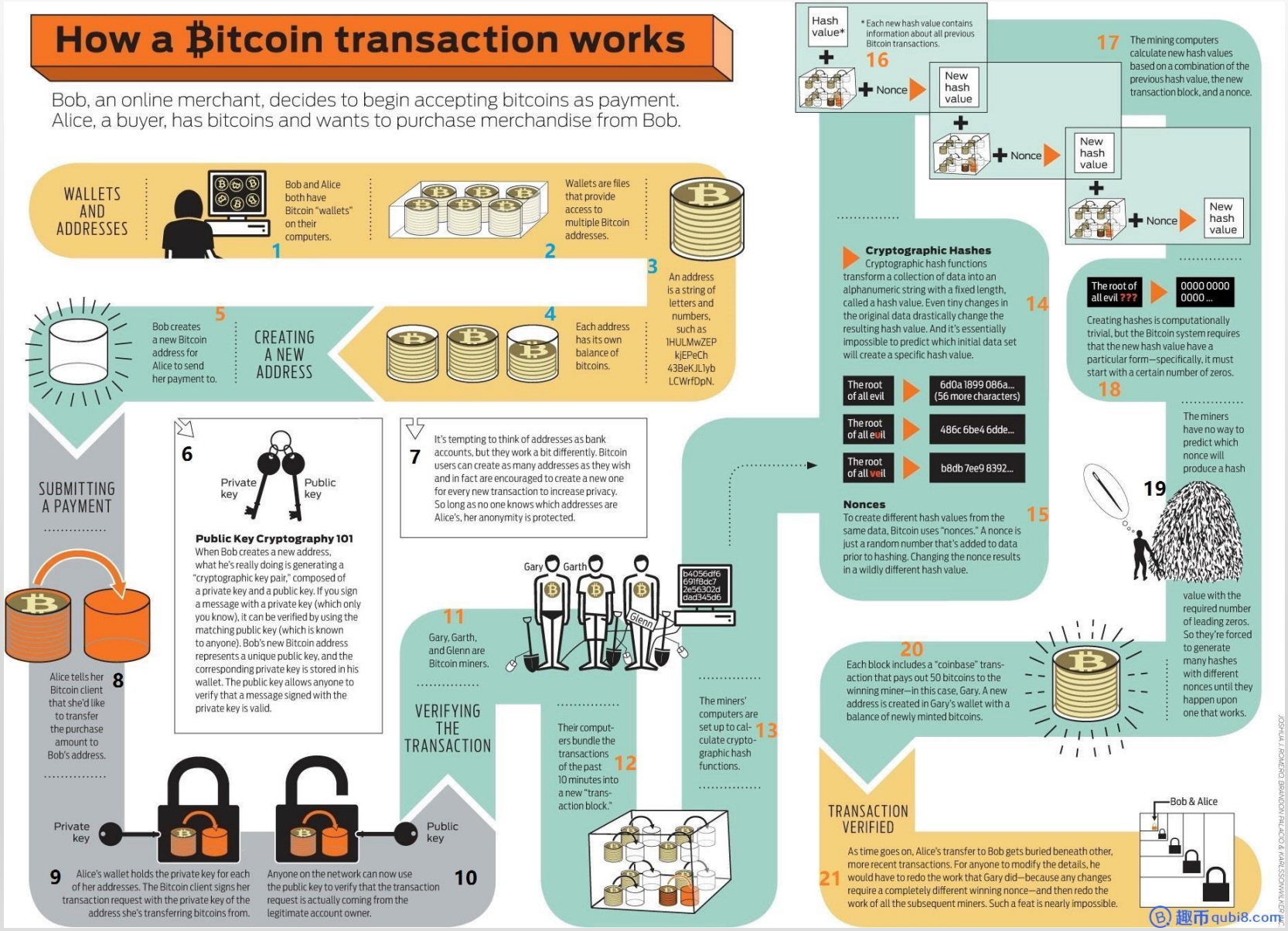
▷ 使用检查清单验证区块内的交易并确保它们的有效性，见前面"区块体交易验证和打包"页。

每一个节点对每一个新区块的独立校验，确保了矿工无法欺诈。

对于coinbase交易，我们看到了矿工们如何去记录一笔交易， 以获得在此区块中创造的新比特币和交易费。为什么矿工不为他们自己记录一笔交易去获得数以千计的比特币?这是因为每 一个节点根据相同的规则对区块进行校验。一个无效的coinbase交易将使整个区块无效，这将导致该区块被拒绝，因此，该 交易就不会成为总账的一部分。矿工们必须构建一个完美的区块，基于所有节点共享的规则，并且根据正确工作量证明的解 决方案进行挖矿，他们要花费大量的电力挖矿才能做到这一点。如果他们作弊，所有的电力和努力都会浪费。这就是为什么 独立校验是去中心化共识的重要组成部分。

VICTORLAMP 煤油灯科技

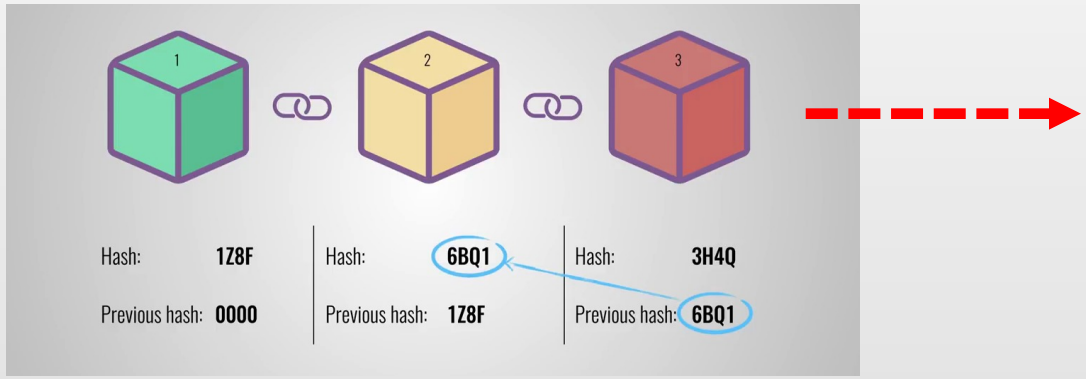
比特币区块链闭环回顾



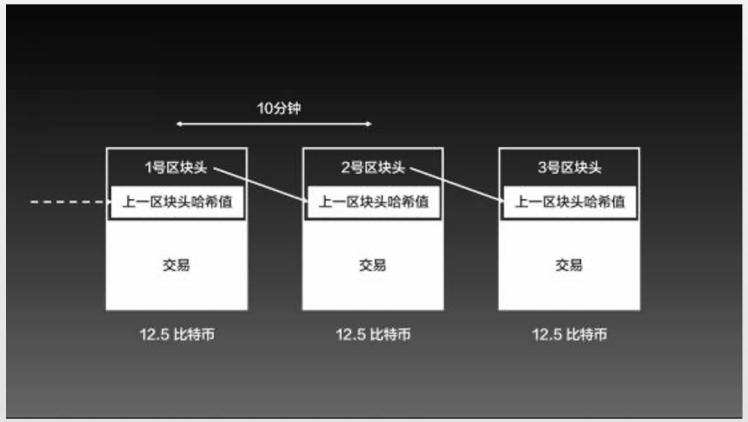
趣币 qubi8.com
VICTORLAMP 煤油灯科技

比特币区块链闭环回顾

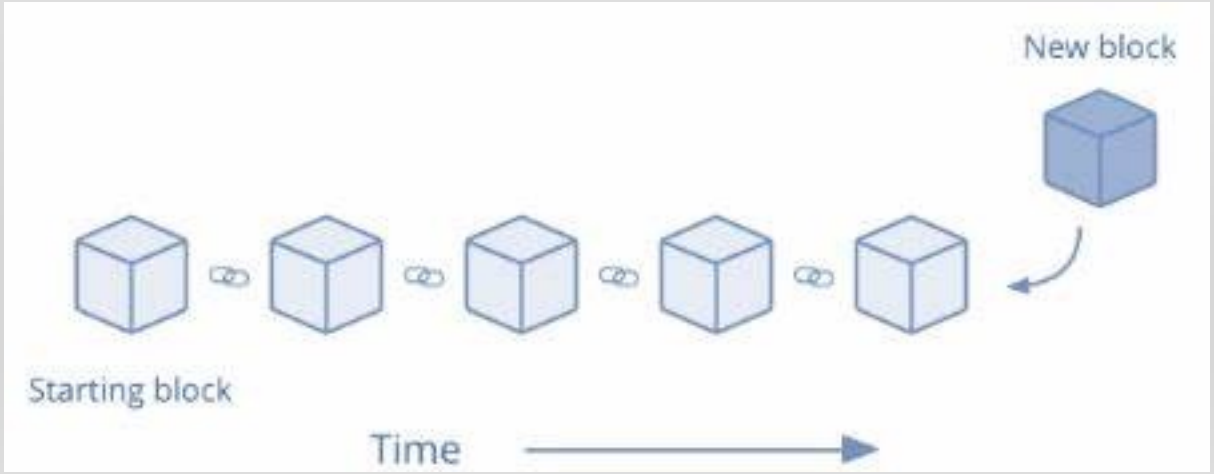
前后区块之间相互关联，一个接一个链接在一起，形成一条链



每一个区块里面装的就是交易记录



每隔10分钟左右，会新生成一个区块，加入到这个链条里面来，永无止尽。区块链网络就是一个创建区块的“永动机”。



煤油灯科技
VICTORLAMP

谢 谢!

VICTORLAMP

Shenzhen VictorLamp Technologies CO. Ltd.
http://www.victorlamp.com

多媒体课程：《深入浅出区块链技术核心篇》

深入浅出区块链技术