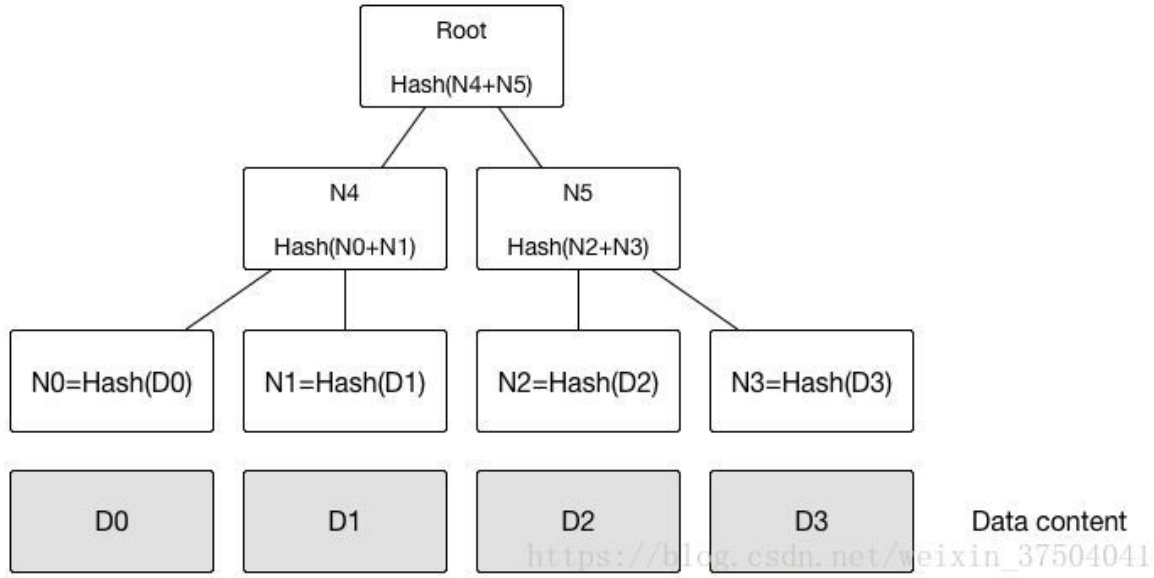
默尔克树

默克尔树是一种二叉树，由一组叶节点、一组中间节点和一个根节点构成，看下图：



我们从最底部开始看，D0、D1、D2和D3是叶子节点包含的数据，也就是叶子节点的value，继续往上看，N0、N1、N2和N3是就是叶子节点，它是将数据（也就是D0、D1、D2和D3）进行hash运算后得到的hash值；继续往上看，N4和N5是中间节点，它们各是N0和N1经过hash运算得到的哈希值以及N2和N3经过hash运算得到的哈希值，注意，它们是把相邻的两个叶子结点合并成一个字符串，然后运算这个字符串的哈希；接着往上，Root节点是N4和N5经过hash运算后得到的哈希值，这就是这颗默克尔树的根哈希。

分析到这里我们大概可以知道在默克尔树中最下面的大量的叶节点包含基础数据；每个中间节点是它的两个叶子节点的哈希，根节点也是由它的两个子节点的哈希，代表了默克尔树的顶部。

还有从默克尔树的结构可以看出，任意一个叶子节点的交易被修改，叶子节点hash值就会变更，最终根节点的hash值就会改变。所以确定的根节点的hash值可以准确的作为一组交易的唯一摘要。

Merkle Tree和Hash List的主要区别是，可以直接下载并立即验证Merkle Tree的一个分支。因为可以将文件切分成小的数据块，这样如果有一块数据损坏，仅仅重新下载这个数据块就行了。如果文件非常大，那么Merkle tree和Hash list都很到，但是Merkle tree可以一次下载一个分支，然后立即验证这个分支，如果分支验证通过，就可以下载数据了。而Hash list只有下载整个hash list才能验证。

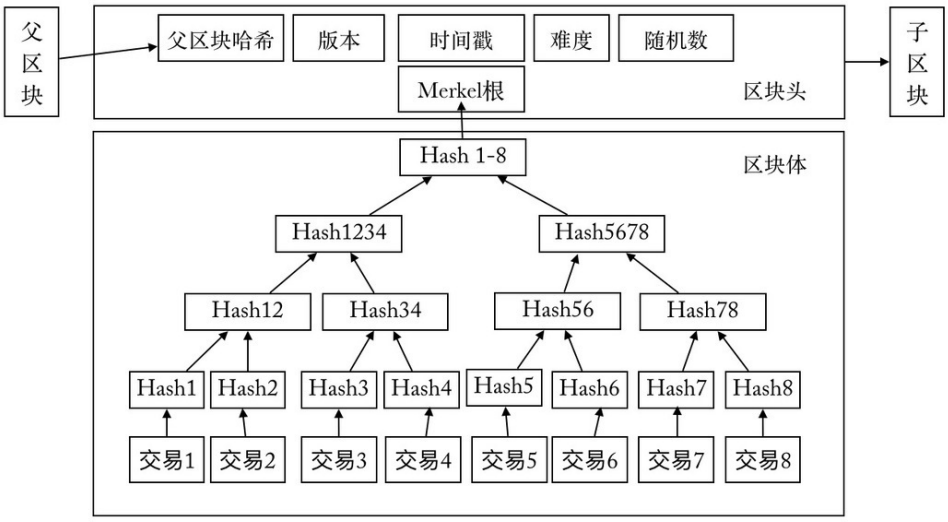
现在可以总结一下默克尔树的特点：

1.首先是它的树的结构，默克尔树常见的结构是二叉树，但它也可以是多叉树，它具有树结构的全部特点。

2.默克尔树的基础数据不是固定的，想存什么数据由你说了算，因为它只要数据经过哈希运算得到的hash值。

3.默克尔树是从下往上逐层计算的，就是说每个中间节点是根据相邻的两个叶子节点组合计算得出的，而根节点是根据两个中间节点组合计算得出的，所以叶子节点是基础。

比特币中的默克尔树



可以看到区块头包含了根节点的hash值，而中间节点、叶子节点还有基础数据在放在了区块体中。

这里有一点需要提的就是在比特网络中的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。

应用

数字签名

最初Merkle Tree目的是高效的处理Lamport one-time signatures。 每一个Lamport key只能被用来签名一个消息，但是与Merkle tree结合可以来签名多条Merkle。这种方法成为了一种高效的数字签名框架，即Merkle Signature Scheme。

P2P网络

在P2P网络中，Merkle Tree用来确保从其他节点接受的数据块没有损坏且没有被替换，甚至检查其他节点不会欺骗或者发布虚假的块。大家所熟悉的BT下载就是采用了P2P技术来让客户端之间进行数据传输，一来可以加快数据下载速度，二来减轻下载服务器的负担。BT即BitTorrent，是一种中心索引式的P2P文件分分析通信协议。

要进下载必须从中心索引服务器获取一个扩展名为torrent的索引文件（即大家所说的种子），torrent文件包含了要共享文件的信息，包括文件名，大小，文件的Hash信息和一个指向Tracker的URL。Torrent文件中的Hash信息是每一块要下载的文件内容的加密摘要，这些摘要也可运行在下载的时候进行验证。大的torrent文件是Web服务器的瓶颈，而且也不能直接被包含在RSS或gossiped around(用流言传播协议进行传播)。一个相关的问题是大数据块的使用，因为为了保持torrent文件的非常小，那么数据块Hash的数量也得很小，这就意味着每个数据块相对较大。大数据块影响节点之间进行交易的效率，因为只有当大数据块全部下载下来并校验通过后，才能与其他节点进行交易。

就解决上面两个问题是用一个简单的Merkle Tree代替Hash List。设计一个层数足够多的满二叉树，叶节点是数据块的Hash，不足的叶节点用0来代替。上层的节点是其对应孩子节点串联的hash。Hash算法和普通torrent一样采用SHA1。

可信计算

可信计算是可信计算组为分布式计算环境中参与节点的计算平台提供端点可信性而提出的。可信计算技术在计算平台的硬件层引入可信平台模块(Trusted Platform，TPM)，实际上为计算平台提供了基于硬件的可信根(Root of trust，RoT)。从可信根出发，使用信任链传递机制，可信计算技术可对本地平台的硬件及软件实施逐层的完整性度量，并将度量结果可靠地保存再TPM的平台配置寄存器(Platform configuration register，PCR)中，此后远程计算平台可通过远程验证机制(Remote Attestation)比对本

地PCR中度量结果，从而验证本地计算平台的可信性。可信计算技术让分布式应用的参与节点摆脱了对中心服务器的依赖，而直接通过用户机器上的TPM芯片来建立信任，使得创建扩展性更好、可靠性更高、可用性更强的安全分布式应用成为可能[10]。可信计算技术的核心机制是远程验证(remote attestation),分布式应用的参与结点正是通过远程验证机制来建立互信,从而保障应用的安全。

文献提出了一种基于Merkle Tree的远程验证机制，其核心是完整性度量值哈希树。

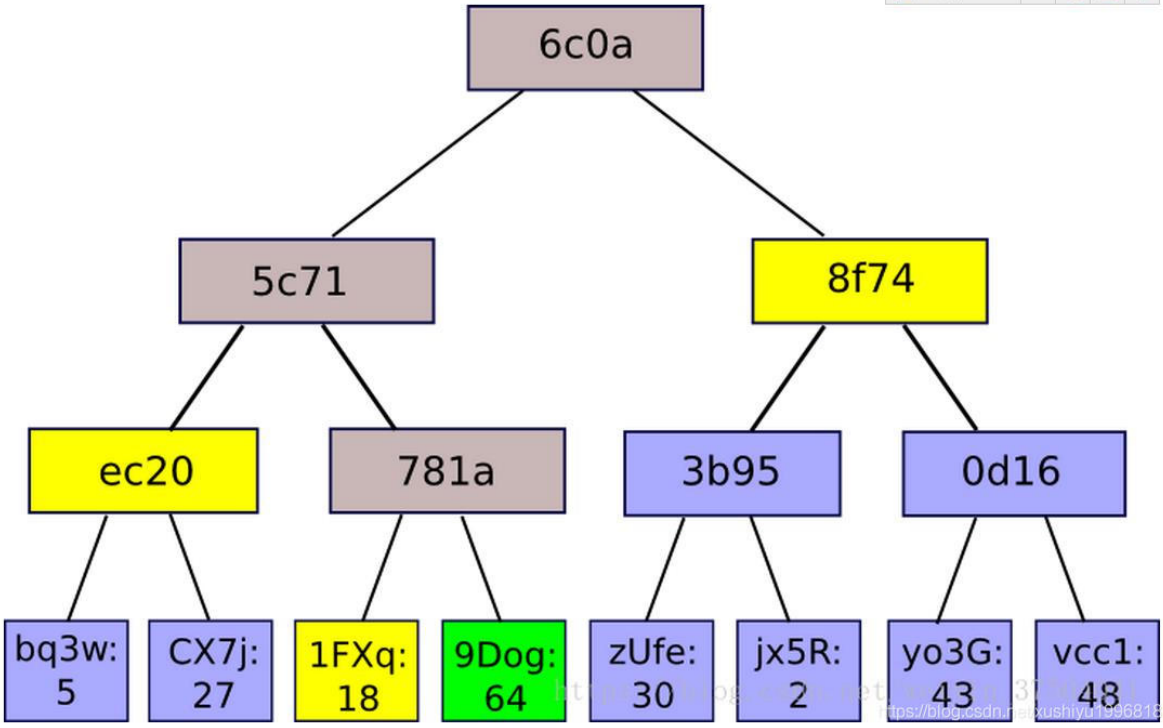
首先,RAMT 在内核中维护的不再是一张完整性度量值列表(ML),而是一棵完整性度量值哈希树(integrity measurement hash tree,简称IMHT).其中,IMHT的叶子结点存储的数据对象是待验证计算平台上被度量的各种程序的完整性哈希值,而其内部结点则依据Merkle 哈希树的构建规则由子结点的连接的哈希值动态生成。

其次,为了维护IMHT 叶子结点的完整性,RAMT 需要使用TPM 中的一段存储器来保存IMHT 可信根哈希的值。

再次,RAMT 的完整性验证过程基于认证路径(authentication path)实施.认证路径是指IMHT 上从待验证叶子结点到根哈希的路径。

区块链-简单验证支付

中本聪在他的创世论文中一个概念，就是SPV，中文意思是简单支付验证，从这里我们可以看出SPV指的是“支付验证”而不是“交易验证”，那这两者有什么区别吗？简单的说就是支付验证只需验证该笔交易是否被确认过了，而交易验证是需要验证该笔交易是否满足一些条件如“余额”是否足够，还有该笔交易有没有存在双花等等一些问题，只有一切都没什么问题后该笔交易才算验证通过，可以看出交易验证要比支付验证更加复杂，所以它一般是由挖矿节点来完成的，而支付验证只要普通的轻钱包就可以完成。那现在有一个问题了，SPV是如何实现的？答案就是默克尔树



假设我们要验证区块中存在Hash值为9Dog:64（绿色框）的交易，我们仅需要知道1FXq:18、ec20、8f74（黄色框）即可计算出781a、5c71与Root节点（藕粉色框）的哈希，如果最终计算得到的Root节点哈希与区块头中记录的哈希（6c0a）一致，即代表该交易在区块中存在。这是因为我上文提到的两个点，一个是默克尔树是从下往上逐层计算的，所以只要知道相邻的另一个节点的hash值就可以一直往上计算直到根节点，另一个是根节点的hash值可以准确的作为一组交易的唯一摘要，依据这两点就可以来验证一笔交易是否存在。

优点

Merkle树明显的一个好处是可以单独拿出一个分支（作为一个小树）来对部分数据进行校验，这个特性在很多使用场合可以带来哈希列表所不能比拟的方便和高效。

Merkle树在区块链中的应用

在区块链中，区块中的交易是按照Merkle的形式存储在区块上面的。每笔交易都有一个哈希值，然后不同的哈希值向上继续做哈希运算，最终形成了唯一的Merkle裉。这个Merkle根将会被存放到区块的区块头中。利用Merkle树的特性可以确保每一笔交易都不可伪造。