区块链复习笔记

一、二 货币

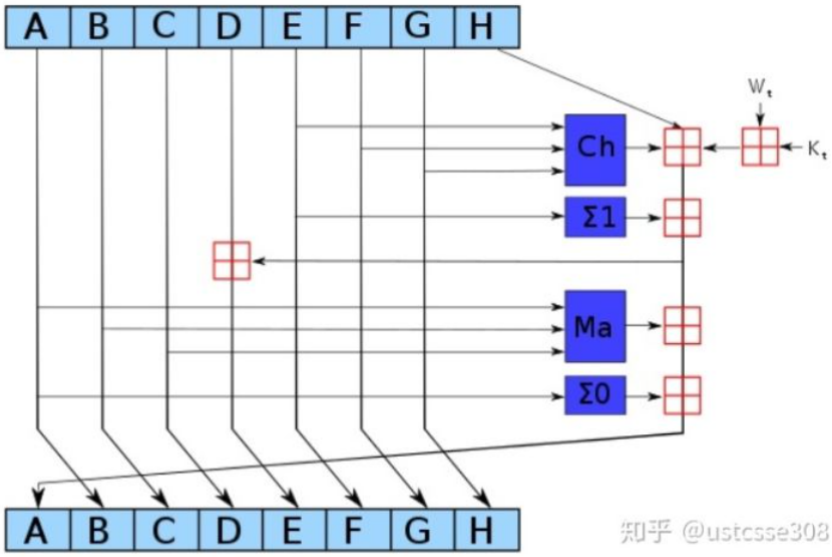
1. 区块链 = Cryptography + Economics + Distributed systems
2. 金融常识

三 电子货币

1. eCash（David Chaum）
    - 首次提出盲签名：一种特殊类型的数字签名，签名者不知道所签署的消息内容
2. HashCash（Adam Back）
    - 一种用于防止垃圾电子邮件和拒绝服务攻击的工作量证明系统
3. B-money（Wei Dai ）
    - 分布式账簿 + PoW + 无法追踪的交易
4. BitCoin（中本聪）

四 对称加密算法 哈希算法

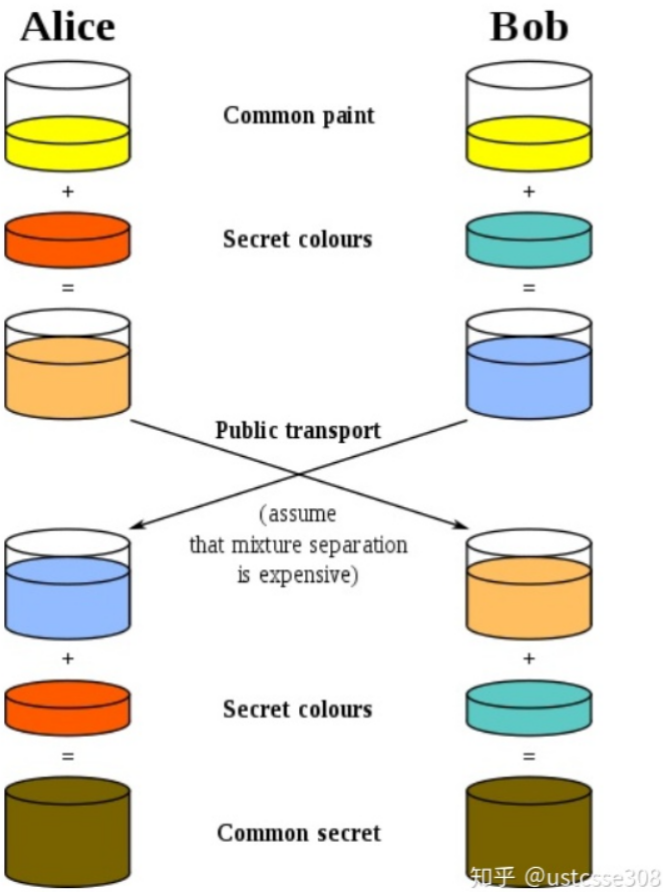
1. hash
    - 哈希表用的哈希函数：ELFHash，HASHPJW
    - **加密哈希:**
        - 特性
            - 确定性
            - 快速计算
            - Hiding（加盐，世界杯预测）
            - 雪崩效应
            - 抗冲突
        - 常见的Hash加密算法
            - MD5
            - SHA-1
            - RipeMD-160
            - SHA-256
    - ***SHA-256工作过程***



来自wiki，SHA2的一轮计算

- 应用：puzzle-friendliness的工作量证明 && 6个区块 (防止双花)
- ***长度拓展攻击***

五 非对称加密算法

1. 出现：例如电子商务中，密钥的协商；密钥的管理；身份验证（防止中间人攻击）
2. DH密钥交换算法（Diffie-Hellman）
   - 

   - 原理：DH算法的有效性依赖于***计算离散对数的难度***。也即，当已知大素数 p 和它的一个原根（primitive root） g，对于给定的 b，要计算指数 i，是非常困难的（暴力破解），而给定 i，计算 b 却很容易。
   - 举例

     > 1. Alice和Bob通过交流，决定选择素数 $p = 23$ 以及原根 $g = 5$。

> 2. Alice选择了一个秘密整数 $a=4$，Bob选择了秘密整数 $b=3$。
> 3. Alice和bob分别使用 p，g，a 和 b 计算出A和B，其中
>
> $$A=g^a mod(p)=5^4 mod(23)=4$$
> $$B=g^b mod(p)=5^3 mod(23)=10$$
>
> 4. Alice和Bob分别将这两个数字A=4和B=10通过网络发送给对方。
> 5. Alice和Bob收到B和A之后，分别计算：
>
> $$s=B^a mod(p)=10^4 mod(23)=18$$
> $$s=A^b mod(p)=4^3 mod(23)=18$$

补充数学知识

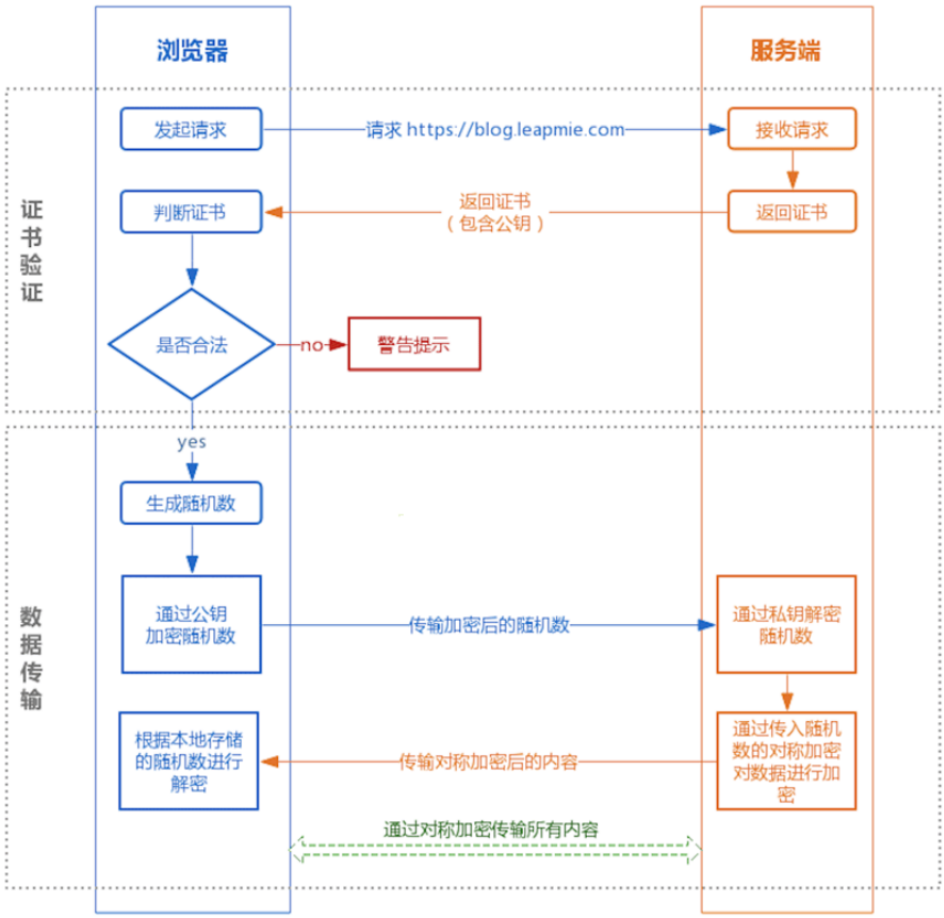
- 原根
  - 概念：如果*a*是素数*p*的一个原根，那么数值：*a* mod*p*，*a^2* mod*p*，...，*a^(p-1)* mod*p*
  - 为什么要是用原根

    > 如果使用的 g 不是 p 的原根，那么 g 的所有指数只能生成 小于p 的整数的一个子集。那么对于攻击者而言，此时即使是暴力破解，需要计算的也只是小于 p 的整数的子集，而不是小于 p 的整数全部。
- 欧拉函数
  - 概念：***φ(n)***任意给定正整数n，在小于等于n的正整数之中，有多少个与n构成互质关系
  - 作用：如果正整数m有原根，那么原根的个数为 $\varphi(\varphi(m))$

    φ(10) = $10\times（1-\frac{1}{2}）\times（1-\frac{1}{5}）$ = 4

    其中应该将20分解为两个没有公共因子的两个数
- 欧拉定理
  - 概念：如果整数 a 和 m 互质，那么 $a^{\varphi(m)}\equiv 1(mod m)$
  - 费马小定理：欧拉定理当 m 时素数时的特例
- HD的缺点：不能防止中间人攻击

3. ***RSA算法***（Rivest、Shamir 和 Adleman）
   - 原理：***对极大整数做因数分解的难度决定了RSA算法的可靠性***
   - 正确性：使用欧拉函数和欧拉定理证明
   - 公钥加密，私钥解密（隐私性） && 私钥加密，公钥解密（不可抵赖）
   - 应用
     - 数字签名（即https上网）



上述流程只是一种最常见的单向验证。如果服务器还要验证客户端，那么客户端也需要把自己的证书发送给服务器验证，这种场景常见于网银等。

- 数字签名的三个特点
  - 加密
  - 不可篡改
  - 不可伪造
- ECC椭圆加密算法

六 去中心化 （比特币中的分布式共识 PoW）

1. 拜占庭问题
2. 共识算法
   - FLP
     - *FLP不可能原理*概念：在网络可靠，但允许节点失效（即便只有一个）的最小化异步模型系统中，不存在一个可以解决一致性
   - 问题的确定性共识算法CAP
     - 概念：一个分布式系统最多只能同时满足一致性（Consistency）、可用性（Availability）和分区容忍性（Partition tolerance）这三项中的两项
   - Paxos
     - 概念：基于消息传递且具有高效容错特性的一致性算法，目前公认的解决分布式一致性问题最有效的算法之一
   - ***Raft***
     - 概念：简化后的Paxos
     - 工作过程：
3. **比特币的共识算法**

- 引入了激励机制
- 比特币的随机性（6个确认块，防止双花）
- 简化过程

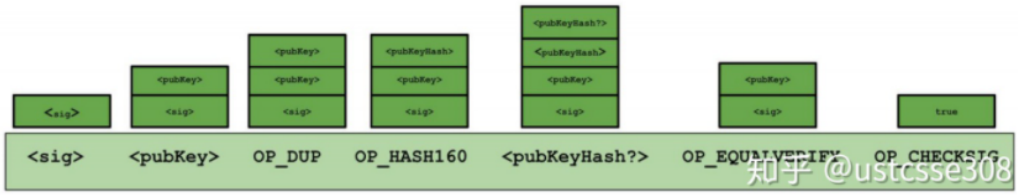
    1. 向所有的节点广播新的交易。
    2. 每个节点将新交易打包进区块。
    3. 每一轮中一个随机的节点广播该区块。
    4. 如果区块中所有的交易都是有效的（比特币没有重花，签名正确等），则其他节点接受这个区块；
    5. 节点表示接受该区块的方式是在之后新创建的区块中包括这个块的哈希。
- ***怎么防止双重支付?***
    - 采用非零确认交易，一般交易后有6个交易确认，则认为安全

4. PoW
    - 区块奖励
        - 区块奖励也是唯一的创建新的比特币的方法
        - 不管节点创造的是好的区块还是包括攻击交易的区块，他都能获得奖励。但是，这个奖励什么时候才有用呢？只有在他的区块被包括在长链中才行。因此，如果一个区块中包括无效的交易，这个区块相当于会被丢弃。所以，所有的节点都能尽力表现得诚实，这样，其他节点才会沿着他的区块继续添加新的区块
    - 交易费用

七 比特币的机制一

1. 交易
    - 为什么不使用account-based账本?
        - 必须对每个账户的余额进行查询，才能确定一笔交易是否有效，如果没有全局的数据结构维护用户的余额，那么可能得一路追踪回去到起始交易，看看Alice到底剩多少钱。如果想要快一点，那就得额外地维护数据结构，譬如有一个全局的数据结构，在每次交易后更新账户余额　（ETH的MPT树就是干这个的！）
    - 比特币的交易
        - 采用这种方式的好处?
            - 方便矿工快速地验证交易的正确性
                - 验证步骤：验证一个交易时，我们首先找到输入所指向的交易的输出，同时为了确保它并没有被花掉，所以我们需要**扫描所指向的交易区块和 *最新区* 块之间所有的区块**，而不需要找到创世区块。然后在验证交易是否合法
        - 看懂比特币交易脚本，一下答案是2，为什么？？？（考点）

            ```
            7    OP_ADD    3    OP_SUB    1    OP_ADD    7    OP_EQUAL
            ```
    - 比特币的脚本
        - 图灵不完备
            - 原因：一种简单的基于栈的编程语言。基于栈意味着每个指令以线性的方式仅仅执行一遍。特别地，比特币脚本中没有循环。因此，脚本的指令的数量就暗示了执行脚本的时间和所用的内存的上限。该语言不是图灵完备的，也即不能执行任意复杂的操作。这也是合理的，因为矿工需要验证交易，也即矿工需要执行这些脚本，如果脚本中出现了死循环，矿工就被坑了
    - P2PKH工作过程(***总是1开头***)



- P2PKH的输入：sigScript = sig + Pubkey
- P2PKh的输出：scriptPubKey = PubkeyHash
- 输入的签名数据是什么?
  - 要求：对矿工明文可见 + 每次不唯一
  - 内容：交易X2
- Question1：在验证过程中，最后一步OP_CHECKSIG的输入实际上就是在新交易中Bob提供的scriptSig的两个部分，最后一步的验证也就是在验证Bob的私钥。那中间还有那么多步骤能不能省略呢?
  - 省略了就是P2PH，也即为什么不用P2PH？（量子加密技术）
- Question2：为啥没有人提前知道公钥会更安全?
  - 量子加密技术，量子计算机可以破解椭圆曲线数字签名算法

- P2PH工作过程

八 比特币的机制二

1. **OP_CHECKMULSIG**指令
   - 出现目的：用于安全钱包、托管交易、以及其他需要多于一个签名的使用情况
   - 应用场景
     - 使用WPS（wallet protection service）保护的钱包
     - 组织资金使用
     - 第三方托管服务（类似支付宝付款）
     - 中心化：去中心化，不是不要中心，而是由节点来自由选择中心、自由决定中心
     - OP_CHECKMULTISIG执行之前，栈中的数据是这样的

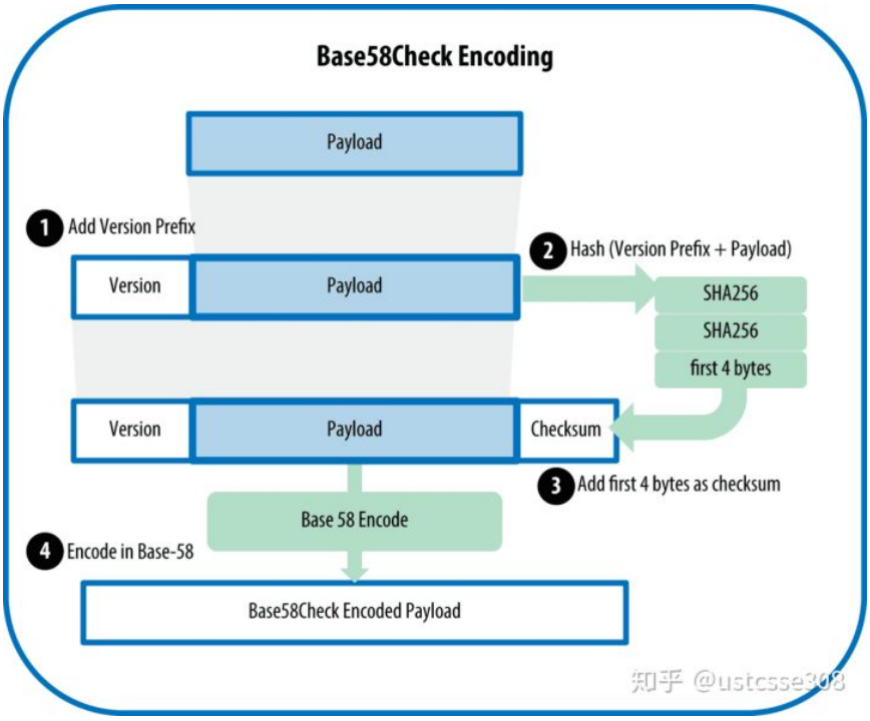
       ```
       3
       (pubKey3)
       (pubKey2)
       (pubKey1)
       2
       (sig2)
       (sig1)
       0
       ```

2. P2SH(使用了MULTiSIG实现，***总是3开头***)
   - 出现目的：要使用MULTISIG，那么对付钱的用户，也就是买家，就不够友好
   - 过程：

     > 验证时是将Bob的签名脚本+Alice的输出脚本，所以基本上可以猜出，在OP_HASH160之前，栈里应该是Bob提供给Alice的脚本的原文，然后Alice的输出脚本中的SH也入栈，最后是运行OP_EQUAL，判断是否相等

   - 输出：***P2SHAddress***
   - ***P2SHAddress和redeemScriptHash的区别***:

    - redeemScriptHash是P2SHAddress经过base58编码
    - P2SHAddress应该是原始赎回脚本经过hash160
    - redeemScriptHash和P2SHAddress可以相互转换
  - base58的好处：
    - 更简洁方便地表示长串的数字
    - 不包括`0， O， l， 1`,防止出错



  - P2SH交易的过程
    - 首先是将scriptPubKey脚本和scriptSig脚本合并，然后得到

      <OP_0> <OP_HASH160> <OP_EQUAL>

1. *OP_0*和 $sigA$ 以及 $sigC$ 入栈。
2. redeemScript 入栈。
3. *OP_HASH160* 对redeemScript执行哈希，栈顶是redeemScript的哈希值。
4. redeemScriptHash入栈。
5. *OP_EQUAL*将会比较 OP_HASH160(redeemScript)的结果和后入栈的 redeemScriptHash，这一步证明了是否提供了正确的redeemscript，也即是否是币的合法的所有者。
6. 然后开始执行redeemScript:
   <OP_2> <A pubkey> <B pubkey> <C pubkey> <OP_3> <OP_CHECKMULTISIG>
7. *OP_CHECKMULTISIG* 将对 3个公钥和栈中的2 个签名进行验证。

  - TimeLock
    - 举例（可以实现简单的智能合约）
    - 指示具体的 **Unix 时间戳**，这笔交易在 Unix 时间戳`nLockTime`之后，才可以被写入账本

九 比特币的机制三

1. Merkle tree
  - ***哈希指针***
    - 哈希指针是一种数据结构，是一个指向数据存储位置及其位置数据的哈希值的指针。这就使得正常的指针可用于取回信息，哈希指针用于验证信息是否发生改变

- 用途:
  - 分布式系统中用于数据一致性验证,快速定位出错位置
  - 比特币中，merkle树主要适用于组织正常的交易，使得**交易易于验证**并且使用较少的资源。可以大大降低SPV节点的存储和计算负担

2. 核心客户端

3. SPV客户端

   - 可以验证交易确实存在，不是双重支付，就是用少量的资源去验证用户的这笔交易已经在区块链中被承认了，还可以判断某个钱包的余额（即正常支付使用、收款、余额、即日常的消费需求，不参与挖矿）

     > 完整的区块链节点是通过检查整个链中在它之下的数千个区块来保证这个UTXO没有被支付，从而验证交易。而SPV节点是通过检查在包含该交易的区块所收到的确认数目来验证交易?

   - **SPV易受到的攻击**

     不能验证某个交易（譬如同一个UTXO的双重支付）在整个链中不存在

4. Bloom Filter

   - 出现原因：SPV节点一般只需要的是和自己的地址相关的交易，那么SPV节点怎么样从网络中接收到与自己相关的交易，确定交易所在的区块呢
   - 作用：用来判断某个元素是否在集合内，存在FP，不存在FN，它具有运行速度快（时间效率），占用内存小的优点（空间效率），但是有一定的误识别率和删除困难的问题

     广泛的用途：Bloom filter被广泛应用于各种领域，比如拼写检查、字符串匹配算法、网络包分析工具、Web Cache、文件系统、存储系统等
   - 优势：空间效率和查找时间复杂性；不需要存储元素本身，在保护隐私方面具有优势。（也是可以追踪的，比如攻击节点可以通过运行全节点来大致知道某些信息，例如：带宽和硬件条件宽裕，SPV节点可以选择具有高FP（false positive）率的Bloom Filter，此时如果有第三方对SPV进行跟踪，看到的将是大量的数据中混杂着与节点相关的数据，从而隐私性得到了保护）

5. 布隆过滤器工作过程:

6. 几个问题:

   - Q1：***peer什么时候要更新filter，为什么要更新***?

     - 第一个区块是由服务节点从磁盘上读取的。它包含TX 1，TX1向客户的密钥（公钥、地址）发送资金。它与过滤器相匹配，因此被发送到客户端。
     - 第二个块是由服务节点从磁盘上读取的。它包含TX 2，它花费了TX 1。然而TX 2不包含任何客户的密钥（公钥、地址），因此没有被发送。客户端不知道他们收到的钱已经被花掉了。

   - Q2：为什么有Bloom_update_P2PUBKEY_ONLY选项?

     在P2PKH类型的交易中，如果要花费，则在scriptsig中一定会有用户的公钥提供，所以用户可以通过在filter中添加自己的公钥来查询；从而防止filter性能快速下降

   - Q3：能不能通过删除项来更新bloom filter?

     CBF

7. merkle路径的构造（深度优先遍历）

   - *Partial Merkle branch format*

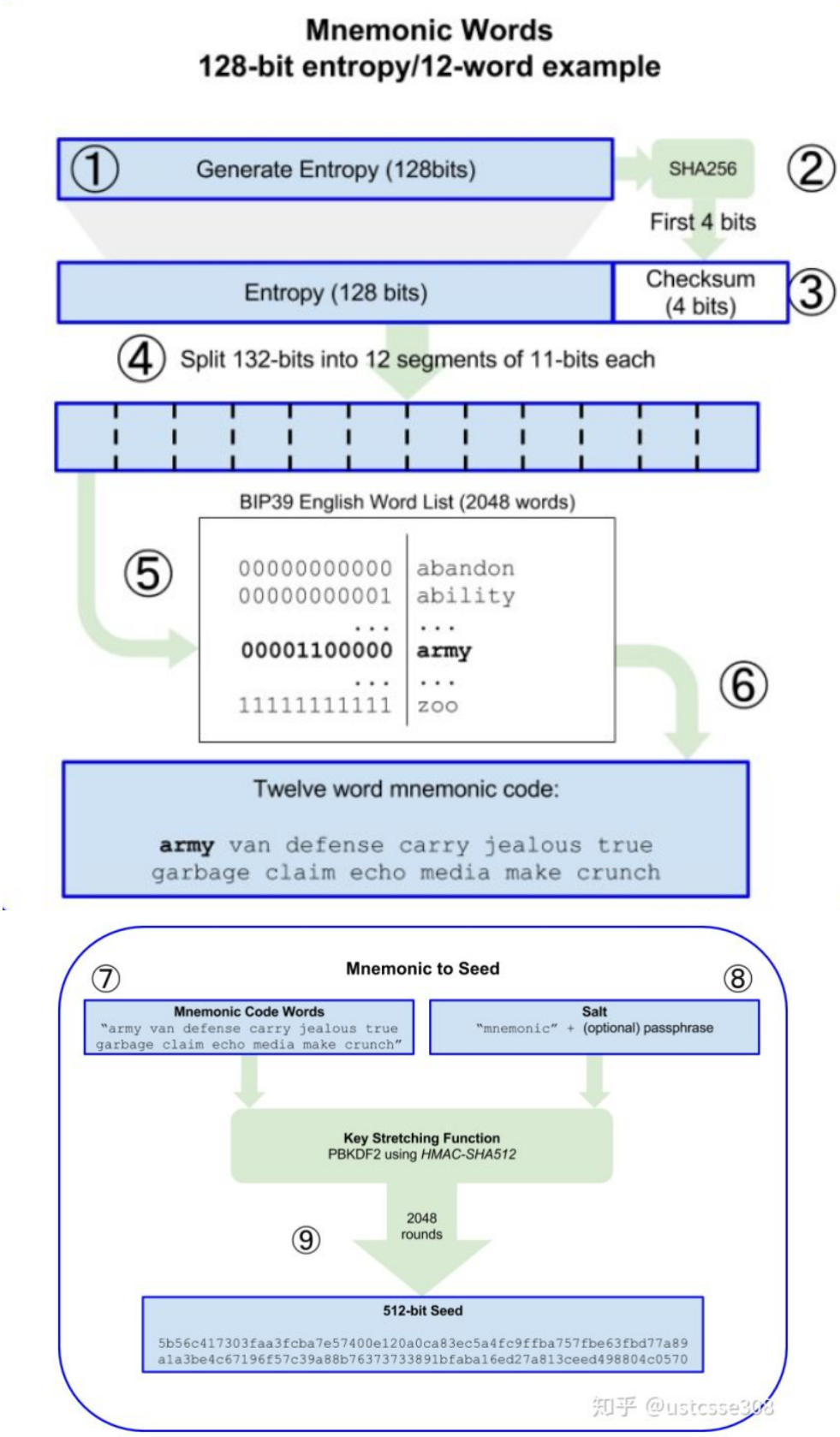
```
if (当前节点是祖先或者自身):
    访问并标记为1
else:
    访问并标记为0
if (当前结点是祖先)
    访问此节点，并递归访问左孩子
else:
    访问此节点，不访问孩子
```

- *Constructing a partial merkle tree object*

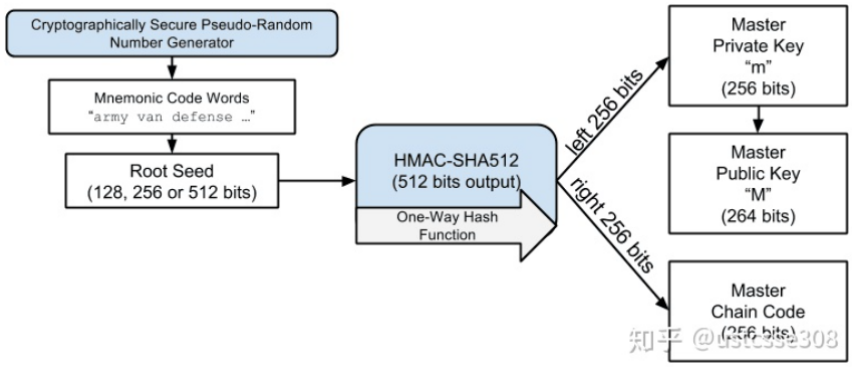
```
o对应hash，1对应null
```

十 如何存储和使用比特币

1. 如何管理密钥
    - 可用性
    - 安全性
    - 方便性
2. 钱包的分类
    - 非确定性钱包（JBOK钱包）
    - 确定性钱包（例如HD钱包）
    - BIP32(定义了HD钱包)
        - 所有的密钥都是从一个主密钥派生出来，这个主密钥即为***种子（seed）***，种子被编码为英文单词，也称为助记词
        - 优势：
            1. 树状结构可以被用来表达额外的组织含义
            2. 用户可以建立一个公钥的序列而不需要访问相对应的私钥
        - seed生成密钥的步骤



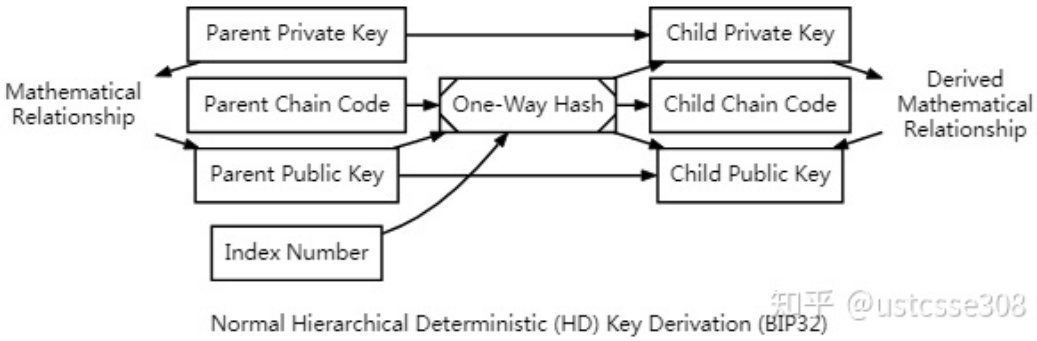
- 有了根种子seed，开始生成一系列的私钥：



> 输出的高 256 位，是主私钥
>
> 输出的低 256 位，是主链码（Master Chain Code）

- **HD 钱包的确定性来源于种子，当种子确定后，钱包中的所有私钥就都是确定的，都可以从种子计算出来**
- 种子的确定由**助记词**和**用户密语**决定
- HD 钱包中的私钥是树状的层级结构
  - 树根位置的私钥，称为主私钥（Master Private Key），从种子直接计算得到
  - 树中的某个私钥，从其父私钥计算得到。
  - 新生成的子公钥可以继续生成后代公钥

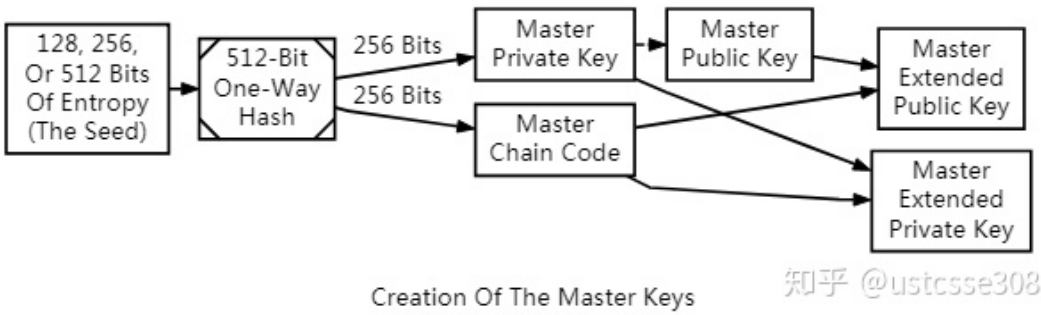
- 链码用来干什么的?



Normal Hierarchical Deterministic (HD) Key Derivation (BIP32)

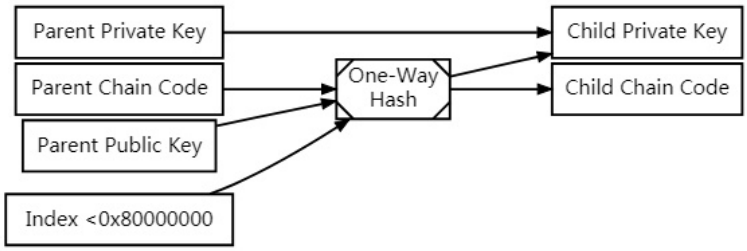
- 可以从私钥和链码，衍生出其**所有的**子私钥及对应的子公钥（及之后每层所有的子私钥及对应的子公钥）
- 可以从公钥和链码，衍生出其**常规衍生**的子公钥（及之后每层常规衍生的子公钥）
- 无法从某个密钥（公钥和私钥）计算出其父密钥，或同层的其他兄弟密钥

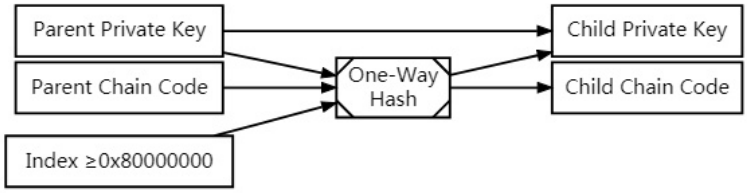
- 拓展密钥：衍生子密钥时需要将*密钥*、*链码和子密钥*序号作为 CKD 的输入，三者缺一不可。为了方便转录，可以将**密钥**和**链码**编码在一起，得到**扩展密钥**（Extended Key）。扩展私钥和相对应的扩展公钥具有相同的链码。扩展密钥使用 Base58Check 编码，并添加特定的版本前缀。如图：



Creation Of The Master Keys

- 强化子密钥：***扩展密钥使用方便，但要注意存在安全隐患。***强化的子密钥生成需要祖先链码、祖先私钥和索引值生成子链码和子私钥





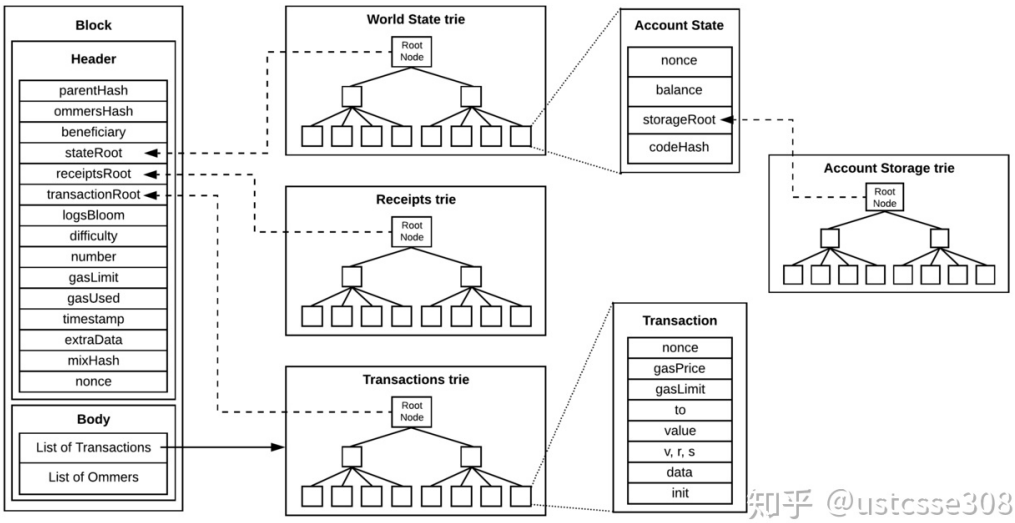
Normal (Top) And Hardened (Bottom) Child Private Key Derivation

        - 强化的子密钥生成需要祖先链码、祖先私钥和索引值生成子链码和子私钥。这样的话，仅仅知道祖先扩展公钥不能用来生成强化的子公钥。
    - BIP-44
        - BIP-44提议了多账户结构作为“purpose”。所有遵循BIP-44的HD钱包依据只使用树的第一个分支的要求而被定义：m/44'/。 BIP-44指定了包含5个预定义树状层级的结构：

          `m / purpose' / coin_type' / account' / change / address_index`
    - HD钱包总结
        - 只需要备份**助记词**和**密语**，就等于备份了整个钱包内的所有私钥
        - 除此之外，你还要记下使用的**衍生路径**，这样才能知道使用了哪些私钥
        - 从扩展公钥可以常规衍生子公钥及对应地址而不用访问扩展私钥或私钥本身，这是 HD 钱包一个很重要的安全特性

十一 以太坊简介

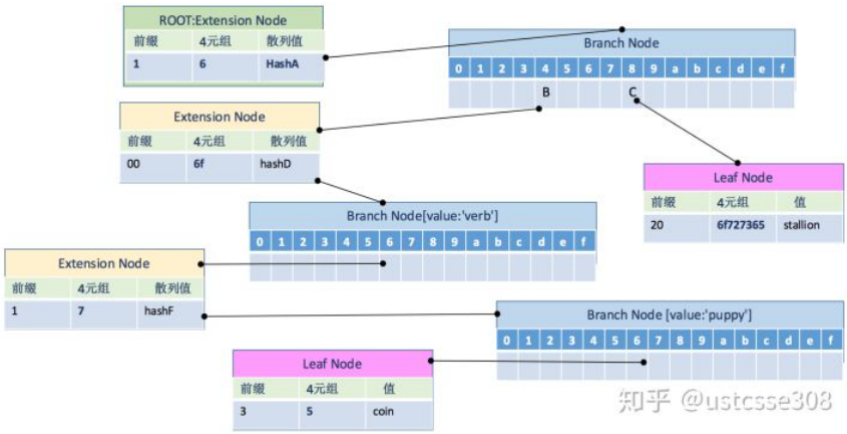
1. 比特币脚本语言的限制
    - 缺少图灵完备性
    - 价值盲
    - 缺少状态
    - 区块链盲
2. 账户类型
    - 外部拥有的账户
    - 合约账户：合约账户不可以自己发起一个交易。相反，合约账户只有在接收到一个交易之后(从一个外部拥有账户或另一个合约账户接)，为了响应此交易而触发一个交易
3. 账户的状态
    - nonce
    - balance
    - storageRoot
    - codeHash



账户存储树是保存与账户相关联数据的结构。该项只有合约账户才有，而在 EOA 中，storageRoot 留空、 codeHash 则是一串空字符串的哈希值

4. 以太币的优点：
    - 图灵完备：引入了gas，每个操作都需要不等的gas
    - Value-blindness
        - UTXO机制的优点：
            1. 较高程度的隐私保护
            2. 潜在的可拓展性
        - 账户的优点：
            1. 节约空间
            2. 可替代性高
            3. 简单
            4. 轻客户端
    - 区块头的三棵树：
        - 世界状态树（world state trie）
        - 交易树（transaction trie）：交易树用来证明某笔交易在某个区块当中
        - receipt树：以太坊拥有智能合约，而智能合约的执行过程比较复杂，通过增加收据树，有利于系统快速查询执行的**结果**
    - GHOST协议
        - 摒弃了单一的最长链原则, 取而代之的是最大子数原则；孤块奖励问题。
    - MPT
        - ***画法***
        - ```
          <64 6f> : 'verb'
          <64 6f 67> : 'puppy'
          <64 6f 67 65> : 'coin'
          <68 6f 72 73 65> : 'stallion'
          ```

          ```



十四 智能合约和solidity

1. 攻击＃1：重入攻击
    - 过程
        1. 攻击者将以太币捐赠给目标合约
        2. 因为受到捐赠，所以目标合约更新了攻击者的余额
        3. 攻击者要求退还资金
        4. 资金被退回
        5. 攻击者的fallback函数被触发，并要求再次取款
        6. 智能合约更新攻击者余额的代码尚未执行，因此成功再次调用了取款
        7. 资金发送给攻击者
        8. 重复步骤5-7
        9. 攻击结束后，攻击者会将合同中的资金发送到其个人地址
    - 解决方案

      重入攻击利用了两个特定的智能合约漏洞。第一种是在发送资金后而不是在发送资金前更新合同状态。由于没有在汇款前更新合同状态，因此该函数可能会在计算过程中中断，并且会诱使合约以为资金尚未实际汇出。第二个漏洞是合同错误地使用address.call.value()发送资金，而不是address.transfer()或address.send()。两者的花费上限为2300 gas，只足以记录一个事件，而不是多个外部呼叫。

        - 汇款前更新合约余额
        - 汇款时使用address.transfer（）或address.send（）
2. Verge
    - 区块链系统是，区块时间戳允许乱序（生成大量的有误时间戳，达到降低难度）
    - Verge 使用了五种算法是 Scrypt，X17，Lyra2rev2，myrgroestl 以及 blake2s作为工作量证明的算法

Uniswap介绍

1. ERC20
    - 概念：ERC20 是以太坊定义的一个代币标准。定义了在实现代币的时候必须要遵守的协议，如指定代币名称、总量、实现代币交易函数等，只有支持了协议才能被以太坊钱包支持。
    - Question1：为什么会出现uniswap这样的协议来支持ETH和各种代币进行自动兑换呢?
        - 挣差价
    - 订单簿式 DEX：IDEX、DDEX、Radar Relay、EtherDelta、Paradex 和 Ethfinex
        - 缺点：不适合缺乏流动性的市场
        - 优点：适合流动性市场

2. Uniswap

- 概念：利用储备金流动性来实现协议上的数字资产交易兑换。Uniswap本质是一个自动化做市商（AMM），它舍弃了传统订单薄的撮合方式，采用流动池加恒定乘积公式算法($x * y = k$)为不同加密资产提供即时报价和兑换服务。
- 优点：Uniswap是一个基于以太坊的自动代币交换协议，它的设计目标是：易用性、gas高利用率、抗审查性和零抽租
- **流动池加恒定乘法公式**
  - 根据以下公式来计算ERC20代币的交易汇率：

    $$x * y = k$$

    k 表示一个不变的常数；x 和 y 表示特定交易对中ETH和ERC20代币的可用数量。
  - **Uniswap 上的交易类型：ETH ⇄ ERC20 交易**
    - ```
      ETH 池 * token 池 = 恒定乘积值
      ```
    - 滑点：滑点是指用户最终实现兑换的汇率偏离了真实的汇率
    - ETH ⇄ DAI（一种ERC20代币）来举例说明：**当兑换金额相对于兑换池资金量过大，或者说对于一定的兑换金额，兑换池资金太少时，就会出现较大滑点。但当兑换池中的代币数量充足时，滑点的数额就会减少，而且资金越多，滑点越小。滑点的大小和常说的交易深度【交易深度】是指市场在承受大额交易时币价不出现大幅波动的能力】大小相反，滑点越小，交易深度越大，用户越能够按照稳定的价格完成交易。**
    - 鼓励用于向Uniswap的流动池中提供更多的流动性，Uniswap会从每笔交易总额中抽取0.3%当成交易手续费，并将手续费全额交给那些将注资金到Uniswap资金池提供流动性的流动性提供者。第一个流动性提供者把自己认为等价值的ETH数量和ERC20代币数量充值到此交易合约，就可以实现设置汇率。而如果第一个流动性提供者设置的这个汇率和外面更大盘的市场不一致，那么套利交易者就会通过交易来把这些价差抹平，和大盘保持一致的汇率。此后所有流动性提供者将以其充值时的汇率作为计算等价的依据。
      ```