如何存储和使用比特币

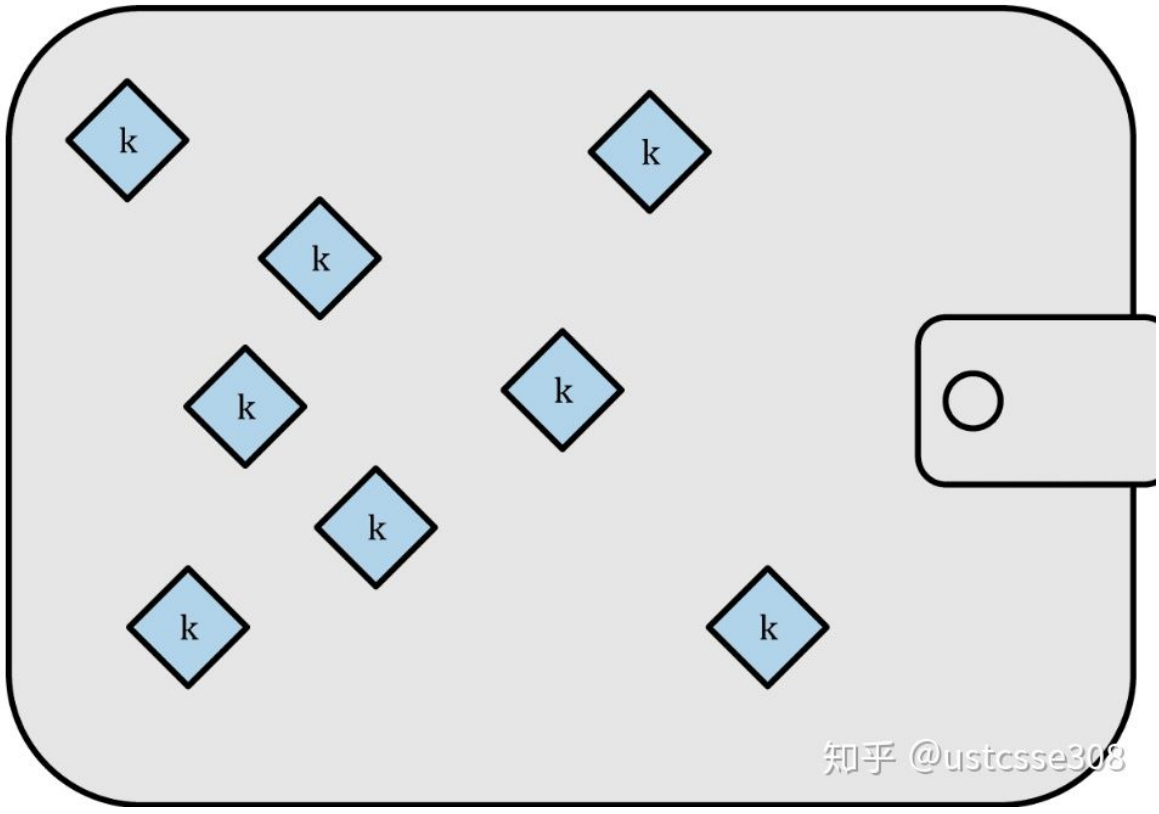
问题：比特币的钱包里面存储的是什么?

在某种意义上，比特币钱包是密钥链。每个用户有一个包含多个密钥的钱包。 钱包只包含私钥/公钥对的密钥链

- 秘钥保管方法的三个目标
  1. 可用性：当需要使用比特币的时候，可以用得上；
  2. 安全性：保证其他人不能使用你的私钥；
  3. 方便性：用的时候比较简单
- nondeterministic wallet

  之前介绍非对称密码的时候，了解过公钥和私钥，知道给定公钥或私钥中的一个，我们可以生成相对应的密钥。这样一次性生成一个密钥，而且密钥根据随机数生成，多次生成的密钥之间也没有相关性。并且每一笔交易之后，必须进行备份，防止私钥丢失。早期的做法是一次性生成100个左右的公私钥对，从最开始就生成足够多的私钥并且每个密钥只使用一次。管理这种密钥的钱包称作，非确定性钱包（nondeterministic wallet），这种钱包也被称为"Just a Bunch Of Keys（一堆密钥）"，简称JBOK钱包。这种钱包的缺点很明显，因为很难管理。如果生成多个密钥，必须保存所有的副本。而且一旦钱包不能访问，所有的资金就不能使用了。在建议每个比特币地址只使用一次的情况时，这种钱包不是好的选择，虽然比特币核心客户端包含了这种钱包，但是不鼓励大家使用。这种钱包现在正在被确定性钱包替换。

  下图为非确定性钱包示意图



- 确定性钱包

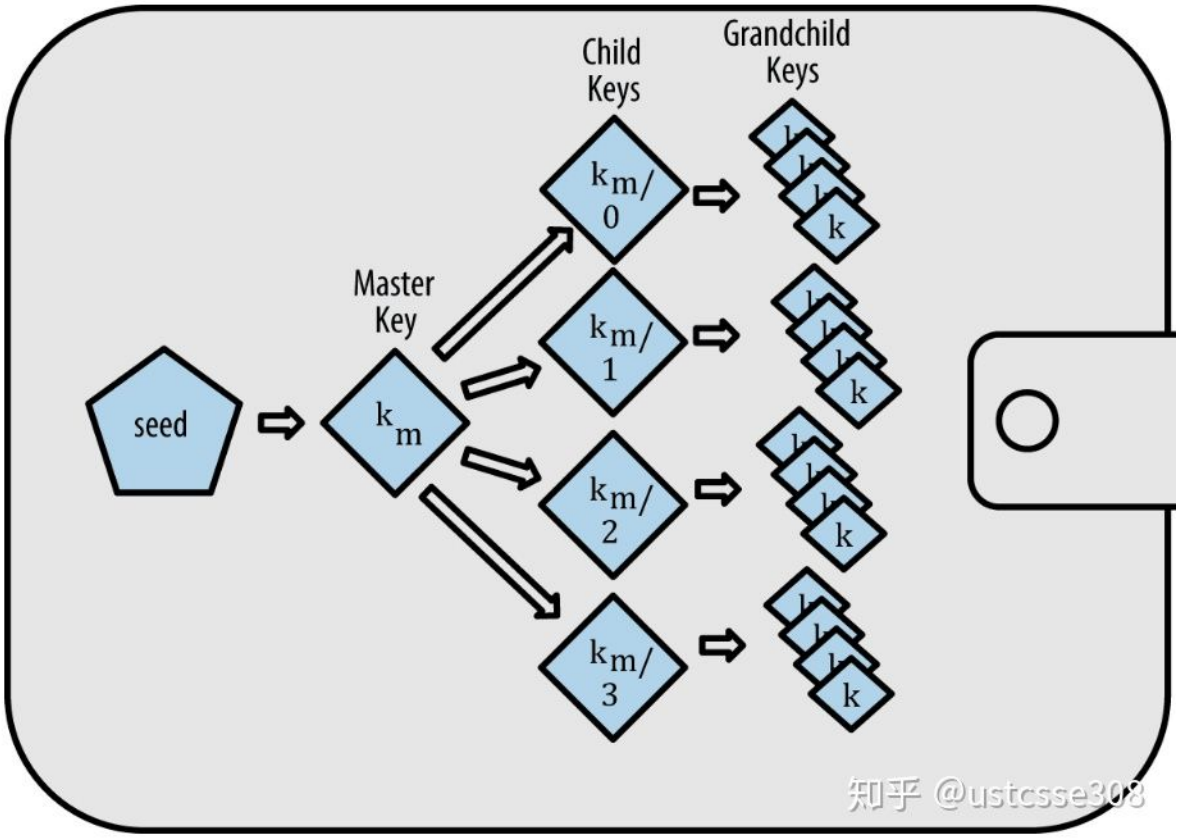
第二种类型是确定性钱包（deterministic wallet），其中所有的密钥都是从一个主密钥派生出来，这个主密钥即为种子（seed）。该类型钱包中所有密钥都相互关联，如果有原始种子，则可以再次生成全部密钥。

确定性钱包使用了许多不同的秘钥推到方法，最常用的推导方法是使用树状结构，称为分级确定性钱包或HD钱包。确定性钱包由种子衍生创造。为了便于使用，种子被编码为英文单词，也称为助记词。

下列 BIP 共同定义了一种确定性钱包的实现，这种钱包被称为分层确定性（HD，Hierarchical Deterministic）钱包。

- BIP-32

  BIP-32标准定义了HD钱包。HD钱包包含以树状结构衍生的密钥，使得父密钥可以衍生一系列子密钥，每个子密钥也可以衍生出一系列孙密钥，以此类推。



  HD钱包的两种主要优势：

  1. 树状结构可以被用来表达额外的组织含义。比如当一个特定分支的子密钥被用来接收交易收入并且有另一个分支的子密钥用来负责支付花费。不同分支的密钥都可以被用在企业环境中，这就可以支配不同的分支部门、子公司、具体功能以及会计类别
  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。所以HD钱包在不安全的服务器中使用或者在每笔交易中发行不同的公钥。公钥不需要被预先加载或者提前衍生，而在服务器中不需要可用来支付的私钥。

- 从种子生成秘钥数

  下面来看一下，怎么样从种子生成密钥树。因为16进制表示的种子：0C1E24E5917779D297E14D45F14E1A1A 难以记忆，所以BIP-39标准中定义了助记词。上面的种子相对应的助记词表示：

  army van defense carry jealous true garbage claim echo media make crunch

思考：128位是如何生成12个助记词的?

助记词是由钱包使用BIP-39中定义的标准化过程自动生成的。 钱包从熵源开始，增加校验和，然后将熵映射到单词列表：

1、创建一个128到256位的随机序列（熵）。

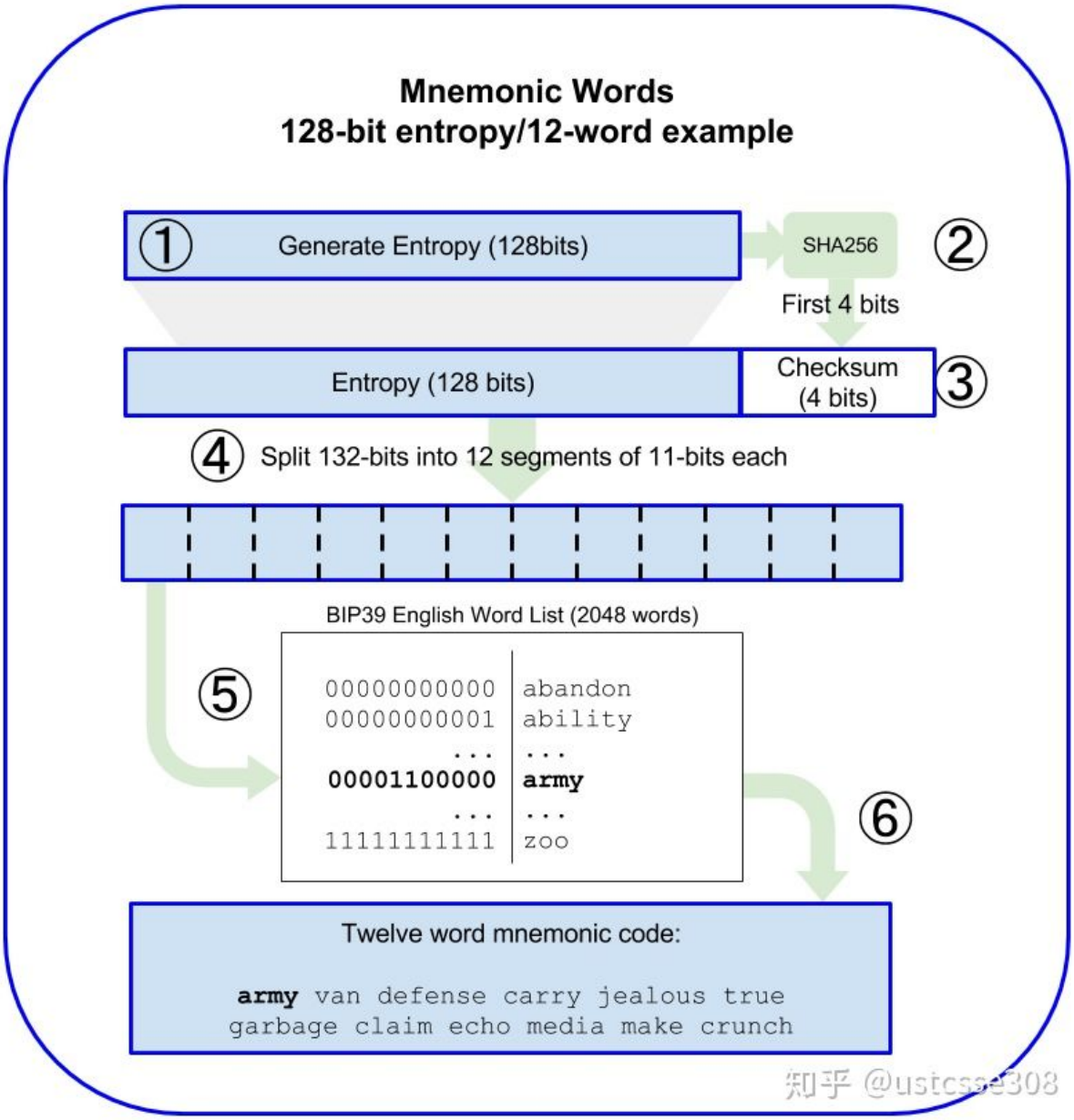
2、提出SHA256哈希前几位（熵长/ 32），就可以创造一个随机序列的校验和。

3、将校验和添加到随机序列的末尾。

4、将序列划分为**包含**11位的不同部分。

5、将每个包含11位部分的值与一个已经预先定义2048个单词的字典做对应。

6、生成的有顺序的单词组就是助记码。



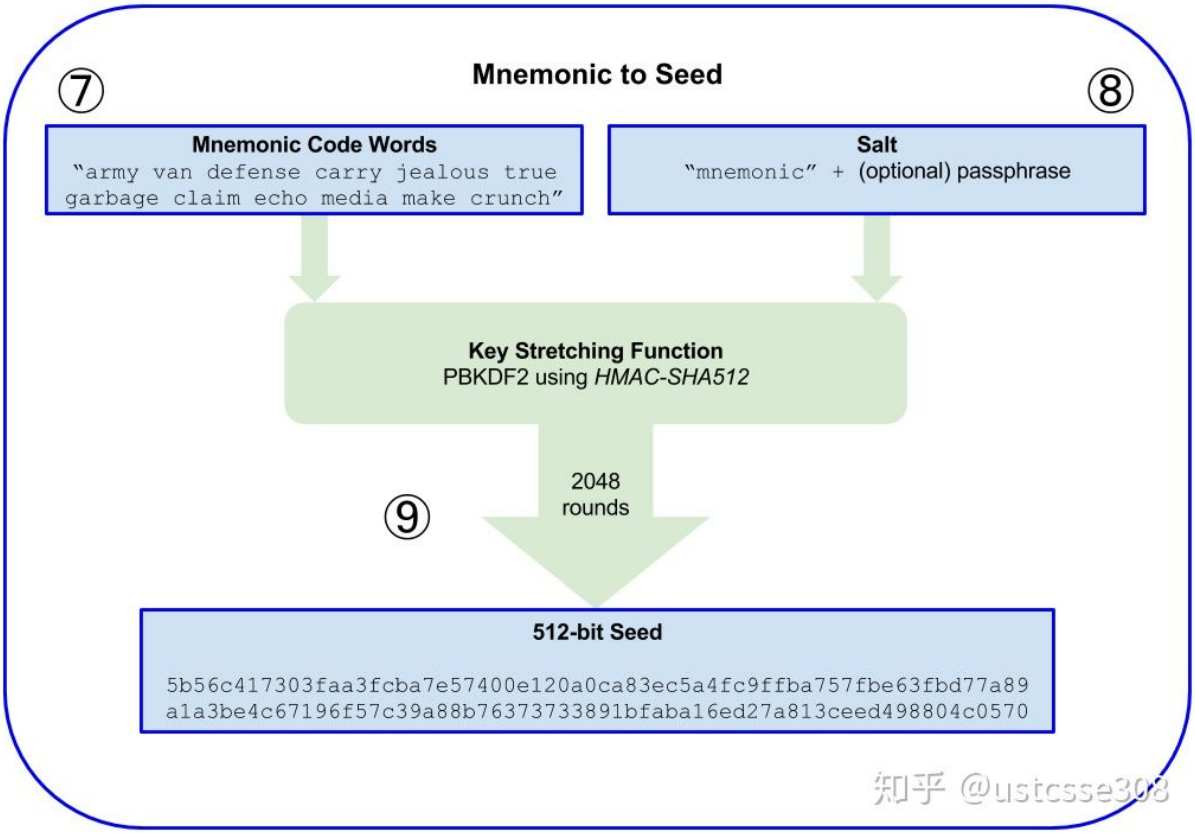
总结一下：HD钱包具有管理多个密钥和地址的强大机制。由一系列英文单词生成种子是个标准化的方法，这样易于在钱包中转移、导出和导入， 这些英文单词被称为助记词，标准由BIP-39定义。大多数比特币钱包（以及其他加密货币的钱包）使用此标准，并可以使用可互操作的助记词导入和导出种子进行备份和恢复。

创建助记词之后的7-9步是：

7、PBKDF2密钥延伸函数的第一个参数是从步骤6生成的助记符。

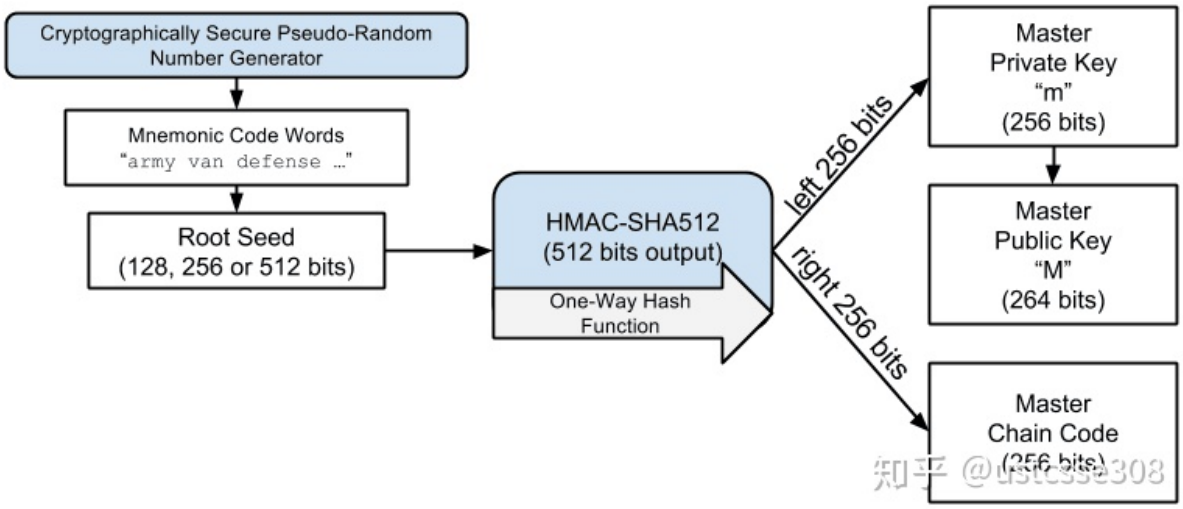
8、PBKDF2密钥延伸函数的第二个参数是盐。 由字符串常数“助记词”与可选的用户提供的密码字符串连接组成。

9、PBKDF2使用HMAC-SHA512算法，使用2048次哈希来延伸助记符和盐参数，产生一个512位的值作为其最终输出。 这个512位的值就是种子。



有个这个512比特的种子，可以开始公私钥对的生成。

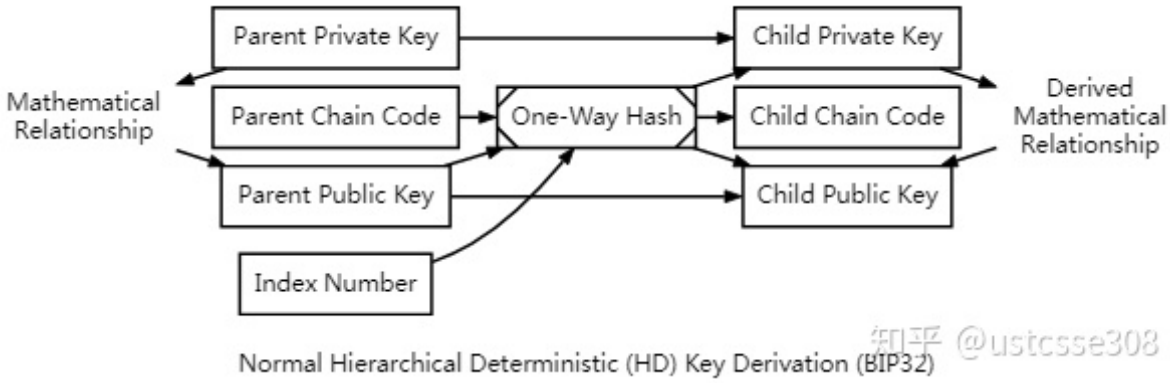
总结：HD钱包从单个根种子（root seed）中创建。最常见的是，这个种子是从助记符产生的，HD钱包的所有的确定性都衍生自这个根种子。任何兼容HD 钱包的根种子也可重新创造整个HD钱包。所以简单的转移HD钱包的根种子就让HD钱包中所包含的成千上百万的密钥被复制，储存导出以及导入。



从上图可以看出，从root seed生成公私钥对的过程，就是对root seed进行一次HMAC-SHA512的加密哈希，在生成512的结果之后，将512位结果划分为两个256部分，分别是Master private key（主私钥）和Master Chain Code（主链码）。

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。

- 扩展秘钥



Normal Hierarchical Deterministic (HD) Key Derivation (BIP32)

HD协议使用256位的Chain Code（也称作熵）来生成子密钥对，而且每个子密钥对都有自己的chain code。这样，即使有一个子密钥分支被攻击，其他的分支可以不受影响。

如上图所示，HD密钥的生成接收了四个输入：

- 祖先私钥 和 祖先公钥
- 256位的祖先链码
- 32位的索引值

上图的计算中，链码、公钥和索引值作为HMAC-SHA512的参数输入，产生512位确定的但是足够随机的输出。这512位输出的右一半的256位作为子密钥的链码。左一半的256位作为生成子密钥的输入。

```
child_private_key == (parent_private_key + lefthand_hash_output) % G
child_public_key == point( (parent_private_key + lefthand_hash_output) % G
)
child_public_key == point(child_private_key) == parent_public_key +
point(lefthand_hash_output)
```

一个问题：链码可以被公开吗?

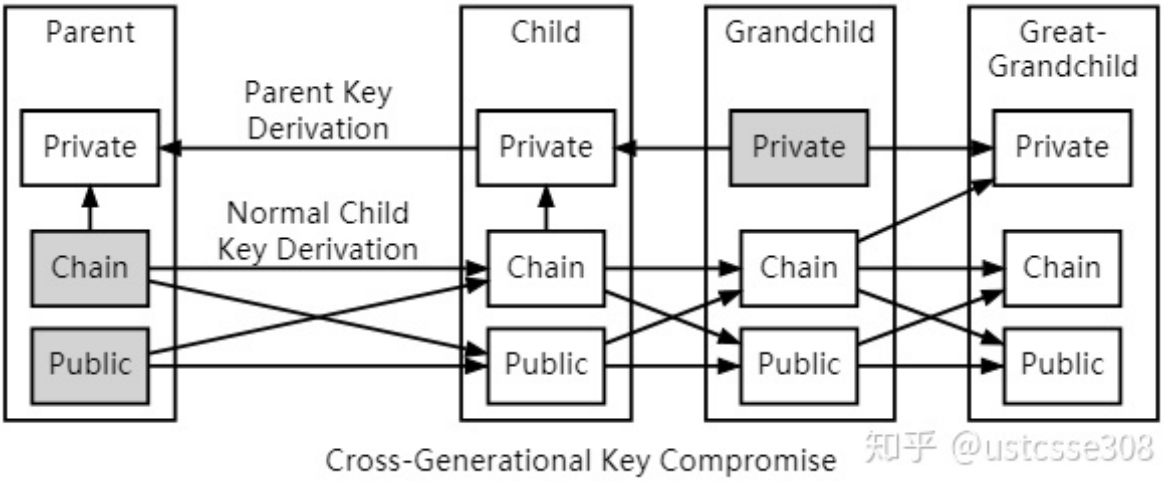
不可以，因为根据上图，有了父链码之后，可以根据父public key和index number推算出子public key，**引入链码的目的就是消除兄弟节点之间的关系**，公开链码之后和不引入链码区别不大了，因为公开链码之后依旧可以推算出其子节点兄弟节点之间的关系

- 拓展秘钥的注意事项

扩展密钥使用方便，但要注意：

- 虽然泄露某个扩展公钥不会丢币，但会导致以此为根节点衍生出的扩展公钥全部泄露，破坏了隐私性
- 泄露扩展公钥和该公钥衍生出的之后任一代公钥对应的私钥，有被推导出该扩展公钥所有后代私钥的可能

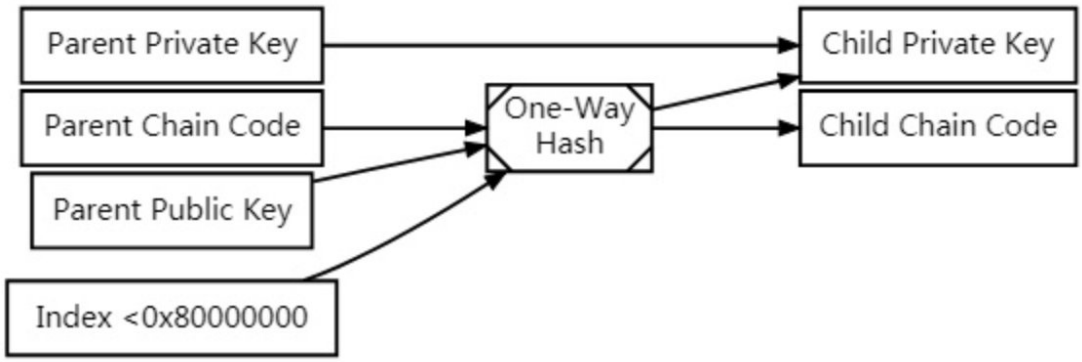
也即，如果攻击者获得了父链码和父公钥，那么就可以获得所有的子链码。有了子链码，如果又获得了底层的某一个私钥（孙子密钥），那么可以根据这个链码生成所有的扩展私钥。



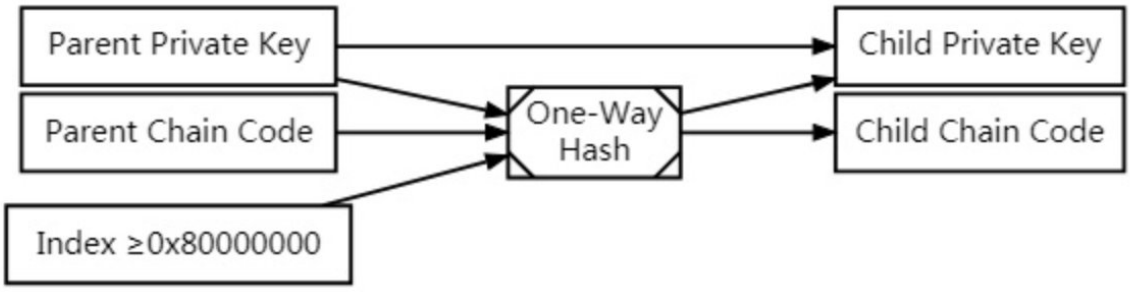
更惨的是，攻击者可能通过子密钥逆推回祖先私钥。也即，如果攻击者获得了一个扩展公钥，以及任何它的子孙私钥，那么就能够推出这个公钥对应的私钥，以及所有的后代密钥。

因此，扩展公钥的链码部分需要好好保存。也因此提出了**强化子密钥的概念**

- 强化子秘钥



上图是之前的子秘钥生成机制，下图是强化子秘钥生成机制



即使用父私钥(之前使用的是父公钥)来参与链码的计算

> 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。（无法生成Left-hand-output，所以知道父公钥的情况下，也不能直接计算出子公钥；）

因此，强化的子密钥的应用场景没有正常生成的子密钥多。但是可以防御上面提到的攻击

频率较高的考试题：

练习（天书般的bip32）：

The function N((k, c)) → (K, c) computes the extended public key corresponding to an extended private key.

Child key derivation (CKD)： The function CKDpriv((kpar, cpar), i) → (ki, ci) computes a child extended private key from the parent extended private key，function CKDpub((Kpar, cpar), i) → (Ki, ci) computes a child extended public key from the parent extended public key。

To shorten notation, we will write CKDpriv(CKDpriv(CKDpriv(m,3H),2),5) as m/3H/2/5. Equivalently for public keys, we write CKDpub(CKDpub(CKDpub(M,3),2),5) as M/3/2/5. This results in the following identities:

- N(m/a/b/c) = N(m/a/b)/c = N(m/a)/b/c = N(m)/a/b/c = M/a/b/c.
- N(m/aH/b/c) = N(m/aH/b)/c = N(m/aH)/b/c.

However, N(m/aH) cannot be rewritten as N(m)/aH, as the latter is not possible.

注意/不是除号