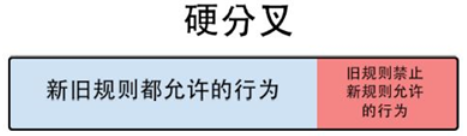
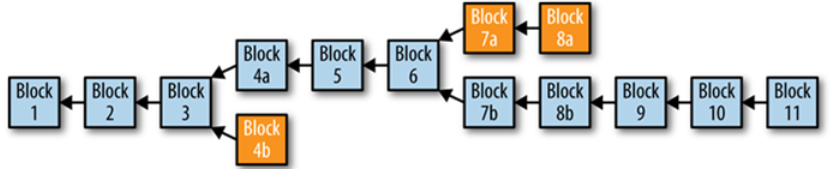
一、软分叉与硬分叉

1.硬分叉

如果区块链软件的共识规则被改变，并且这种规则改变无法向前兼容，旧节点无法认可新节点产生的区块，即为硬分叉。



这时候旧节点会拒绝新规则的区块，于是新节点和旧节点会开始在不同的区块链上运行（挖矿、交易、验证等），由于新旧节点可能长期存在，这种分叉也可能会长期持续下去。



上图中Block 4a与Block 4b是自然分叉的结果，最终在Block 5整个区块链达成了最终一致性

假设当区块链扩展到Block 6时发布了改变共识规则的新客户端版本，于是运行不同客户端的节点出现分歧：

运行新版本的矿工将接受新规则下产生的区块Block 7b;

运行旧版本的矿工由于不能认可新规则，将拒绝Block 7b，并挖出基于旧规则的Block 7a;

此时区块链即发生了硬分叉，新旧矿工将在各自的区块链上进行挖矿，他们的去留将决定两条链的生死和价值。

硬分叉的过程

1..开发者发布新的客户端，新的客户端改变了区块链的共识规则且不被旧客户端兼容，于是客户端软件出现了分叉（software fork）。

2..区块链网络中部分节点开始运行新的客户端，在新规则下产生的交易与区块将被旧节点拒绝，旧节点开始短暂的断开与这些发送被自己拒绝的交易与区块新节点的连接，于是整个区块链网络出现了分叉（network fork)。

3..运行新客户端的矿工开始基于新规则挖矿，并会接受新规则下的交易与区块，于是运行不同客户端版本的矿工算力出现了分叉（mining fork）。

4..运行新客户端的矿工开始挖出基于新规则的区块，而旧客户端的矿工仍然会挖掘基于旧规则的区块，于是整个区块链出现了分叉（chain fork）。

2. 软分叉

如果区块链的共识规则改变后，这种改变是向前兼容的，旧节点可以兼容新节点产生的区块，即为软分叉。



实际上，软分叉通常刚开始并不会产生两条区块链，因为新规则下产生的块会被旧节点接受，旧节点只是无法识别新规则的真实意义。所以新旧节点仍然处于同一条区块链上，对整个系统的影响也就较小。

软分叉在比特币历史上发生过多次，这里以BIP-34举例：

我们在《深入比特币原理（十）》中说过Coinbase交易的输入中有一个字段coinbase data内容可以任意定制。BIP-34就是要求在coinbase data中必须包含块高度（Block Height）信息且将块版本（Block version）从“1”修改为“2”。

显然在旧规则下，节点并不关心coinbase data是什么内容，完全可以兼容包含块高度信息的区块，所以不会就此产生分叉。

---------------------

作者：chabuduoxiansheng1

来源：CSDN

原文：https://blog.csdn.net/chabuduoxiansheng1/article/details/79740018

版权声明：本文为博主原创文章，转载请附上博文链接！

二、交易延展性攻击

交易延展性，或者叫做“交易可锻性”，指的是，比特币支付交易发出后、确认前可被修改（准确说是被伪造复制）。

为什么交易发出后，可能被篡改呢，不是有签名吗？ 其中1个原因就是多数挖矿程序是用openssl 库校验用户签名，而openssl兼容多种编码格式，还有，就是椭圆曲线数字签名算法(ECDSA)本身,签名(r,s) 和 签名（r,-s(mod n))都是有效的。所以，对签名字符串本身的表现方式做些调整，依旧是有效签名。

我们知道，每个Transaction有个Transaction ID，这个Transaction ID也就是对整个交易做的一个Hash，也是该Transaction的唯一标识。现在你对签名做了微调，签名还是有效签名，但是Transaction ID却因此改变了！！！

而黑客就是利用了这个特性，对交易所实施了攻击，下面就来看一下这个攻击过程是怎样的：

Step1: 黑客自己有1个账号，再交易所开了1个账号，把自己的bitcoin转进去。

Step2: 申请提现（withdraw）,交易所发起1笔Transaction。

Step3: 这笔交易被广播到网络上，还未打包进区块链之前。黑客收到这笔Transaction，稍微更改了scriptSig的格式，生成1笔新的交易广播出去，此时Transaction id已经变了。

Step4: 黑客的这笔新交易被区块链接收了。然后向交易所投诉，说它没收到钱。交易所根据自己生成的Transaction Id查询该笔交易，发现在网络上查询不到， 会再次转账给黑客，也就是double withdraw! 同1笔钱，被黑客提现了2次，甚至多次！ 导致交易所蒙受巨大损失。

总结：

交易延展性攻击之所以会发生，是因为Transaction ID会变（而这是1个Transaction的唯一标识），而Transaction ID会变，是因为里面的scriptSig可以被调整。

如果有办法保证Transaction ID在整个交易过程中，都不可能被改变，那也就解决了这个问题，而这就是后续要讲的”隔离见证“。

三、隔离见证

每一个比特币交易，都可以分为两部分。第一部分是转账记录，第二部分是用来证明这个交易合法性（主要是签名）的。第一部分可称为“交易状态”，第二部分就是所谓的见证（witness）。如果你只关心每个账户的余额，那么转账记录就已经足够了。只有部分人（主要是矿工）才有必要取得交易见证。

(交易信息就是谁给谁在什么时间转了多少钱。

见证信息就是哪个节点在什么时间验证交易信息的可靠性。)

中本聪在设计比特币的时候直接把这两个信息直接放在了区块内，所以一个区块就承载不了更多的交易信息，如果隔离了“见证信息”，那么区块链只记录交易信息，那么一个区块可承载的交易更多交易。

中本聪设计比特币时，并没有把两部分资料分开处理，因此导致交易ID的计算混合了交易和见证。因为见证本身包括签名，而签名不可能对其自身进行签名，因此见证可以由任何人在未得到交易双方同意的情况下进行改变，造成所谓的交易可塑性（malleability）。在交易发出后、确认前，交易ID可以被任意更改，因此基于未确认交易的交易是绝对不安全的。在2014年就曾有人利用这个漏洞大规模攻击比特币网络。

比特币核心开发员Pieter Wuille在2015年12月于香港提出的隔离见证（Segregated Witness，以下简称SW）软分叉解决了这个问题。SW用户在交易时，会把比特币传送到有别于传统的地址。当要使用这些比特币的时候，其签名（即见证）并不会记录为交易ID的一部分，而是进行另外处理。也就是说，交易ID完全是由交易状态来决定的，不会受见证部分的影响。

部分开发者认为比特币的设计有缺陷，在数据结构方面，它把必要的交易信息（输入和输出）和“没那么必要”的交易信息（见证）放在一起打包，这造成了一系列问题，比如“交易可塑性”、交易签名过程复杂、还有存储空间“浪费”。

隔离见证是对整个设计缺陷的一个修改方案，原理说起来挺简单，它允许交易把没那么必要的“见证”部分“隔离”在区块外面，这就是隔离见证的意思了。

目前人们主要关注的是存储空间浪费，因为现在比特币交易量太大，区块空间不够用，大量交易堆积。隔离见证的设计减少了每个交易脚本在区块内的体积，这相当于区块空间扩容，同时降低了数据传输和验证造成的内存、带宽、CPU成本。

---------------------

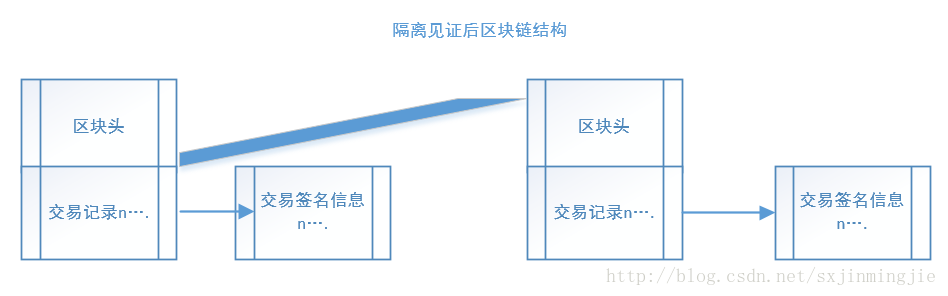
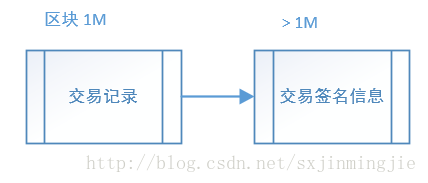
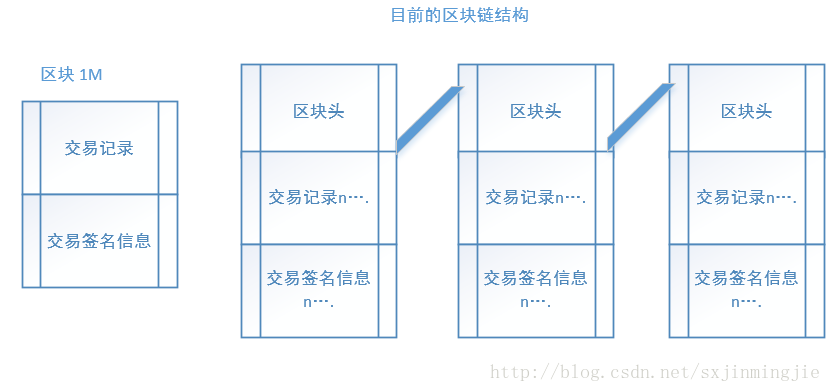
作者：紫魔戒

来源：CSDN

原文：https://blog.csdn.net/qq\_26499321/article/details/76021275

版权声明：本文为博主原创文章，转载请附上博文链接！

所有在比特币系统中出现一种将签名信息隔离开来的提议，那在UTXO中放一个指向签名信息的指针即可。



四、UTXO

UTXO 是 Unspent Transaction Output 的缩写,Transaction act发音简写为X。也就是未被使用的交易输出。

本质上，就是只记录交易本身，而不记录交易的结果。

业内流行一句话：”其实并没有什么比特币，有的只是UTXO“---孟岩；

一般会认为 UTXO 是比特币区块链设计当中的一部分，但事实上 UTXO 和区块链没有必然的联系，

比特币使用前后链接的区块链记录所有交易记录，

每笔交易都有若干交易输入，也就是资金来源，也都有若干笔交易输出，也就是资金去向。

一般来说，每一笔交易都要花费（spend）一笔输入，产生一笔输出，而其所产生的输出，就是“未花费过的交易输出”，也就是 UTXO。

当之前的 UTXO 出现在后续交易的输入时，就表示这个 UTXO 已经花费掉了，不再是 UTXO 了。

如果从第一个区块开始逐步计算所有比特币地址中的余额，就可以计算出不同时间的各个比特币账户的余额了。

比特币交易遵守几个规则。

1. 所有交易始于coinbase（即始于挖矿奖励所得）;

2. 除了 coinbase 交易之外，所有的资金来源都必须来自前面某一个或者几个交易的 UTXO。

3. 任何一笔交易的交易输入总量必须等于交易输出总量，等式两边必须配平。



花费与未花费

花费是什么概念，每一个Transaction Output都犹如现实中的一张纸币，他只有两种状态，属于你或者不属于你。

未花费就是该张纸币属于你，已花费就是该张纸币不属于你。

每一个 UTXO 和纸币一样，只可能有两种状态，要么是没有被花费的，要么就是已经被花费，所有权变成了其他人或者地址，成为其他地址的 UTXO。

找零

每个UTXO 都是一次性的，当用户A给用户B转账时，一定会碰到想转的数额和自己的UTXO数值不匹配的问题。

此时采用的方法是,用多个输出，有一些输出还给自己，例如：

输入(from x gas:10）

输出[0](to y gas:2)

输出[1](to x gas:8)

这个机制称为找零，其实并不只是找零，如果用兜里一把零碎utxo去转账，反而是找回一个整的。

输入输出都是由客户端自行决定的。 可以在一笔交易中产生大量的输出找给自己。

---------------------

作者：微风云重轻

来源：CSDN

原文：https://blog.csdn.net/weixin\_41926234/article/details/81027263

版权声明：本文为博主原创文章，转载请附上博文链接！



上图第一个交易 #1001 号交易是 coinbase 交易。比特币是矿工挖出来的。当一个矿机费尽九牛二虎之力找到一个合格的区块之后，它就获得一个特权，能够创造一个 coinbase 交易，在其中放入一笔新钱，并且在交易输出的收款人地址一栏，堂堂正正的写上自己的地址。在我写文章的这一天（2016年8月9日），这笔比特币的数额规定为 12.5 枚，市价 48,576元人民币。这个 coinbase 交易随着张三挖出来的区块被各个节点接受，经过六个确认以后永远的烙印在历史中。

过了几天，张三打算付 2.5 个比特币给李四，张三就发起一#2001号交易，这个交易的资金来源项写着“#1001(1)”，也就是 #1001 号交易——张三挖出矿的那个 coinbase 交易——的第一项 UTXO。然后在本交易的交易输出 UTXO 项中，把2.5个比特币的收款人地址设为李四的地址。

请注意，这一笔交易必须将前面产生那一项 12.5 个比特币的输出项全部消耗，而由于张三只打算付给李四 2.5 个比特币，为了要消耗剩下的10比特币，他只好把剩余的那 10 个比特币支付给自己，这样才能符合输入与输出配平的规则。

再过几天，张三和李四打算AA制合起来给王五付 5 枚比特币。那么张三或李四发起 #3001 号交易，在交易输入部分，有两个资金来源，分别是#2001(1) 和 #2001(2)，代表第 #2001 号交易的第 (1) 和第 (2) 项 UTXO。然后在这个交易的输出部分里如法炮制，给王五5比特币，把张三剩下的 7.5 比特币发还给自己。以后王五若要再花他这5比特币，就必须在他的交易里注明资金的来源是 #3001(1)。

所以，其实并没有什么比特币，只有 UTXO。当我们说张三拥有 10 枚比特币的时候，我实际上是说，当前区块链账本中，有若干笔交易的 UTXO 项收款人写的是张三的地址，而这些 UTXO 项的数额总和是 10。因为在比特币系统里，一个人可以拥有的地址资源，可谓取之不尽用之不竭。要知道自己的一大堆地址里一共收了多少 UTXO，人是算不过来的，需要由比特币钱包代为跟踪计算。

---------------------

作者：liduanwh

来源：CSDN

原文：https://blog.csdn.net/liduanwh/article/details/81032231

版权声明：本文为博主原创文章，转载请附上博文链接！