闪电网络与隔离见证

比特币区块扩容问题从 2015 年提出，到 2017 年结束。

矿工派支持硬分叉方式提升区块容量，

而核心开发者支持隔离见证+闪电网络方式。

2017/8/1，区块高度 478,559，比特币进行硬分叉产生 BCH。

2017/8/24，区块高度 481824，比特币隔离见证正式激活。

资料来源：网络

整理：xfli5

推特：@xfli5

1 微支付通道（MicroPayment Channel）

1.1 微支付通道（MicroPayment Channel)的提出

目前比特币网络每秒 4~5 笔交易，并且每笔交易还需要手续费。如果买卖双方有大量的小额或者微额交易，比如一个宽带提供商向外提供带宽服务，按小时付费。那大量的小额交易，不仅会让比特币网络负担沉重，而且手续费也不划算。微支付通道就是为了解决小额交易问题，通过在买卖双方建立一个支付通道，专门支持双方的小额支付，不需要经过比特币网络。这个通道除了建立、关闭的时候，要和比特币网络通信，其他时间都是双方点到点的通信。

比特币网络之所以是可靠的、值得信任的，这是因为每笔交易都是公开的，经过所有节点验证的，每一笔交易都是 on-chain，都会得到网络上每个节点的认可，所以交易的任何一方没办法抵赖。

微支付通道只在建立和关闭的时候才与比特币网络通信，其他的交易都是链下交易（off-chain Transaction），这也就是说其他交易都没有比特币区块链网络参与，只是交易的双方点对点交易，如何保证交易的一方不会出现抵赖，接下来我们看看，**微支付通道**是如何解决这个问题的。

1.2 微支付通道建立过程

微支付通道依赖于 **locktime 和 P2SH**。

考虑如下场景：A 是用户，B 是一个数据提供商，B 需要把 1 个 100G 的大数据文件发给 A，价值是 100 比特币。为了降低风险，A 不想 1 次性把 100 比特币给 B，而是每接收到 1G 的数据，给 B 支付 1 比特币。那就需要 100 次的交易。

现在看一下，微支付通道如何解决这个问题：

Step1:

用户 A 发起 1 笔交易，把 100 比特币打到 1 个**公共账号（P2SH）**上面（这个公共账号同时需要 A,B 的公钥，也就是前面所说的多重签名）。这笔钱，需要 A,B 这两个人同时出具私钥，才能把钱取出来。这笔交易叫做**保证金交易（Funding Transaction）**。

注：等 A 拿到 step2 中的 B 签名的退款交易（Refund Transaction）后，才会将保证金交易（Funding Transaction）广播的比特币网络上，此时微支付通道开启！

Step2:

与此同时，用户 A 发起 1 笔退款交易（Refund Transaction）。这笔退款交易的输入，就是 Step1 里面的交易，其目的是把 Step1 里面的 100 比特币，再返回给用户 A。**这笔交易的 nLockTime 为一个>0 的值**，也就是该笔交易是 Hold 在那的，不会立即生效。

具体操作：用户 A 先把这笔交易发给 B，B 认同这笔交易后，B 用自己的私钥签名（也就是写在 scriptSig 里面），再返回给 A，A 把这个 Refund Transaction 攥在手上，这笔交易其实是 A 的一个保底的措施，保证前面的 100 元不会永远拿不回来。

Step3:

我们知道在 Step2 的 Refund Transaction 里面，有 2 个输出：A，100 比特币；B，0 比特币。现在把 Step2 的 Refund Transaction 拷贝 1 份，调整一下输出：A, 99 比特币；B, 1 比特币。也就是付给 B：1 比特币。然后 A 把这个交易发

给 B，B 保留这个交易，不广播到网络上。

详细解释这一步，这一步中 B 发送 1G 的数据给 A，A 收到后，A 发起一个交易，就是把 Step2 的 Refund Transaction 拷贝 1 份，调整一下输出：A, 99 比特币；B, 1 比特币，A 对该交易签名后并把这个交易发给 B，B 将该交易攥在手中。

等 A 收到 B 的新的 1G 文件之后，重新调整输出，变成： A，98；B，2。A 签名后再把这个交易发给 B。如此，不断继续下去：

A: 100，B，0；

A: 99，B，1；

A: 98，B，2；

A：97，B，3；

……

A：1， 99。

这些交易，称为 updated Transaction(或者叫做 Commitment Transaction)，只会在 A,B 之间传递，不会广播到比特币网络上。

Step4:

等 A 收到最后的 1 个 G 的文件，发起 1 个 Settlement Transaction。这里交易里面：A：0 比特币， B：100 比特币。其中 LockTime = 0，B 收到这个交易，广播到比特币网络上, 交易立即生效, B 收到 100 比特币。此时微支付通道关闭！

1.3 微支付通道特点

（1）整个过程，我们会看到，只有 Step1 的 Funding Transaction 和 Step4

的 Settlement Transaction 这两个开始和结束的交易会广播到网络上。

（2）如何避免 B 跑路，A 的钱永远锁死在公共账号里面？

在第一步里面，A 把钱打到了 1 个公共账号上面。如果 B 跑了，A 的钱不是永远提不出来了？实际上 A 会等到 Step2 里面，拿到 Refunding Transaction 之后，A 才会把 Step1 里面的 Transaction 发给 B，同时广播到网络上面。假如 B 跑路了，A 就可以将 Refunding Transaction 拿出来广播到网络上，就可以将钱从公共账号里拿回来。

（3）如何避免 A 跑路，B 拿不到自己的钱？

在 Step3 里面，每个 update Transaction，都有 A,B 共同的签名。如果 A 跑路了，B 就把最新的 update Transaction 广播到网络上，该交易被执行，B 就会拿到最新的钱。

update transaction 有个特点，每一次 update transaction 的 LockTime，都是逐级减小的，所以 B 把最新的 update transaction 广播到网络上之后，肯定会被最先打包，最先执行。先前其他的 update transaction 就不会被执行了。

（4）如何避免 B 篡改交易内容，比如调大给自己的分成比例？

每笔交易里面都有 A,B 的双重签名，B 改了交易内容，和 A 的签名就对不上了，反过来，A 改了交易内容，就和 B 的签名对不上了。所以 A,B 都不可能更改篡改交易内容。

（5）如何防止 A 双花这笔钱？

在 Step2 里面，A 拿到了 Refund Transaction，A 把这个交易广播到网络上，拿回这 100 比特币，再进行第二次花费？这是做不到的，因为 Refund Transaction 有 LockTime，处于锁定状态。并且这个 nLockTime > 后面的任何 1 笔 updated

Transaction 中的值。

（6）这个过程是单向的，只能用来 A 给 B 转账。如果反过来，需要另外再建立 1 个 B 到 A 的通道。

（7）LockTime 的限制。假设 B 跑路了，A 也要等到 Refund Transaction 的 LockTime 到期了，才能拿回自己的钱；同样，假设 A 跑路了，B 也要等到 updated Transaction 的 LockTime 到期了，拿到属于自己的钱。

（8）纵观整个过程，最大程度的降低了风险，但其中的一个风险点就存在 B 损失的情形，不过最大损失也就是 1G 的数据。比如当第五十次时，B 将 1G 数据发送给 A，此时 A 跑路了，则 B 只能广播第 49 次的 update transaction，最大程度的减少损失。

2 闪电网络

2.1 闪电网络之 RSMC

（1）RSMC 简介

RSMC，全称 Revocable Sequence Maturity Contract。Revocable，就是可撤销的意思；Sequence 就是指第 12 课 nLockTime(CLTV)与 Sequence number(CSV) 讲述的 Sequence Number。Sequence Maturity，通俗点讲，就是等到 Sequence Number 满足条件了，进行履约。所以翻译成中文就是：可撤销的、基于 Sequence 成熟度的合约。RSMC 解决如下问题：

1）双向支付，而不是单通道。

2）一方中途退出，另一方可以立即拿回钱，而不是等到 LockTime 到期才能

拿回钱。同时，应该对主动退出方实行惩罚。

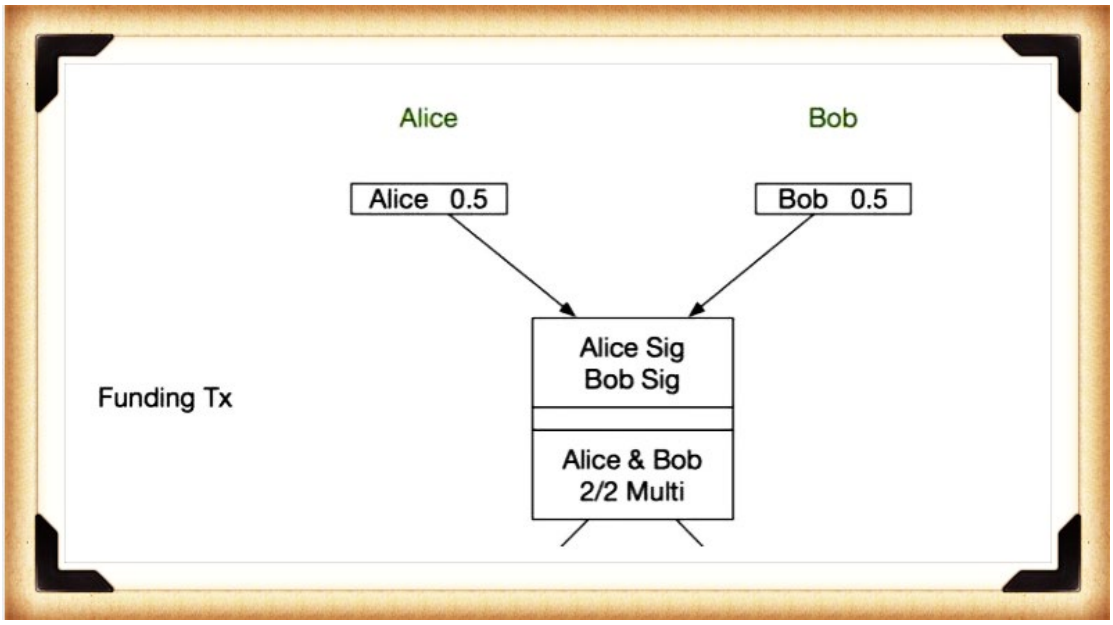
3）保证交易双方，任何一方都不能抵赖、反悔。

（2）RSMC 交易过程详解

下面就看一下，RSMC 如何达成上面的目标。

考虑如下场景，假设 Alice 与 Bob 之间经常有资金往来，看他们如何通过 RSMC 技术实现互相转账：

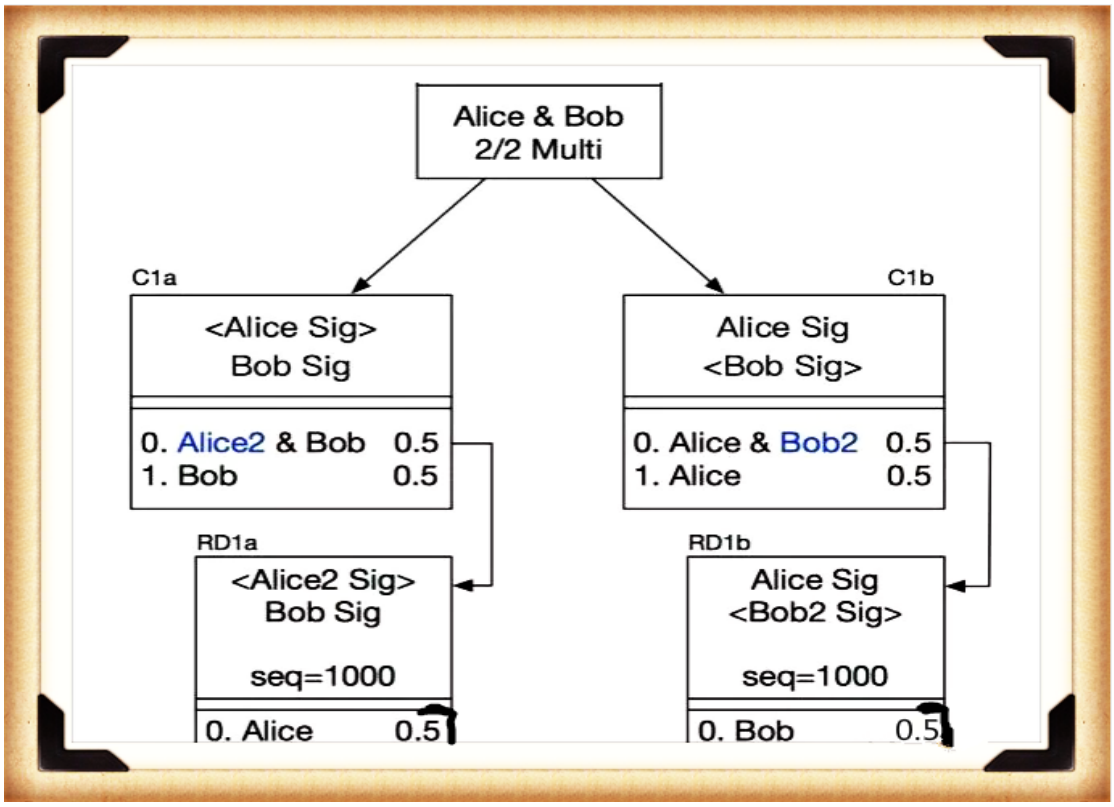
Step1: 同微支付通道一样，生成 1 个保证金交易(Funding Transaction)。不过和微支付通道的区别是，这里是双向支付。所以双方各拿出来相应数量的比特币打入这个公共账户（由 Alice 与 Bob 公钥生成的 P2SH)。如图：



Step2: 同微支付通道类似，为这笔钱生成退款交易（Refund Transaction)。双方可以各自拿回自己的 0.5 比特币。

备注：和微支付通道一样，实际过程是，双方完成了 Step2 之后，才会把

Step1 的交易广播出去。以防钱被死锁在公共账号里面！



重点来了：Alice 生成的退款交易是 C1a + RD1a，Bob 生成的退款交易是 C1b+RD1b，两者是对称的。假如 Alice 要拿回钱，它就广播 C1a + RD1a；假如 Bob 要拿回钱，它就广播 C1b + RD1b。

为什么这么处理呢？

我们看一下：假设 Alice 想主动中断交易，也就是它把 C1a + RD1a 广播到了区块链网络上，那结果是什么呢？？

我们会看到 C1a 里面，会把 Bob 的 0.5 比特币立即返还给 Bob，而 Alice 的 0.5 比特币被打到了 1 个新的公共账号： Alice2 & Bob 里面！Alice 要拿回自己的 0.5 比特币，要等到 RD1a 被兑现。而 RD1a 有个 seq = 1000 属性，也就是要

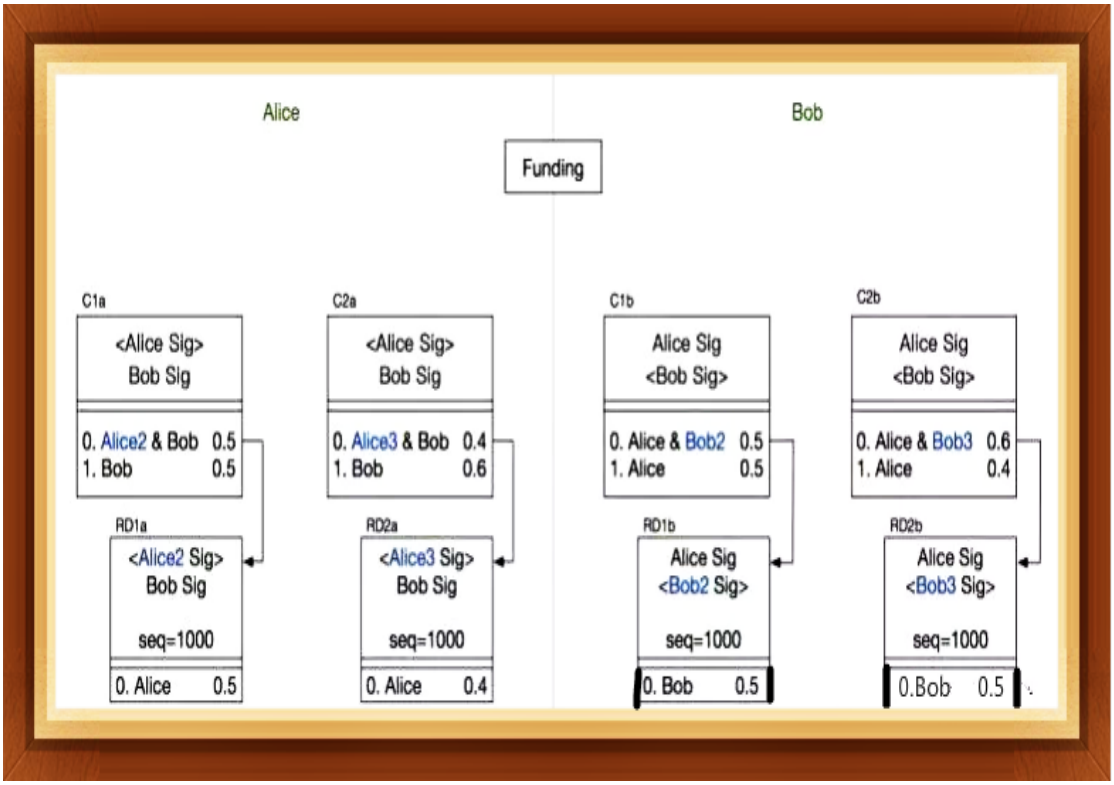
等到 C1a 所在的块，后面被追加了 1000 个块之后，RD1a 这个交易才会被进入区块链里面，Alice 才能拿到自己的钱！

如果 Alice 主动中断交易（把 C1a + RD1a 广播到了区块链上），Bob 立马拿回自己的钱，Alice 则要等到 Sequence Maturity 之后，才能拿回钱（Alice 被轻微惩罚了）。反之亦然！!

Step3: Alice 与 Bob 开始交易

假设 Alice 要付给 Bob 0.1 比特币，那么公共账号里面的资金分配，就从 0.5/0.5，变成了 0.4/0.6。过程如下：

Alice 生成了 C2a 与 RD2a，C1a 与 RD1a 废除；同样，Bob 生成了 C2b 与 RD2b， C1b 和 RD1b 废除。



重点：

在双方达成了 C2a/RD2a, C2b/RD2b 之后，如何让 C1a, RD1a, C1b, RD1b 废除呢？ 换句话说，如何保证 Alice 不反悔（不让 Alice 把 C1a 与 Rd1a 广播到区块链上去？）同样，如何保证 Bob 不反悔（不让 Bob 把 C1b 与 Rd1b 广播到区块链上去？）？**这需要引入惩罚机制**！！

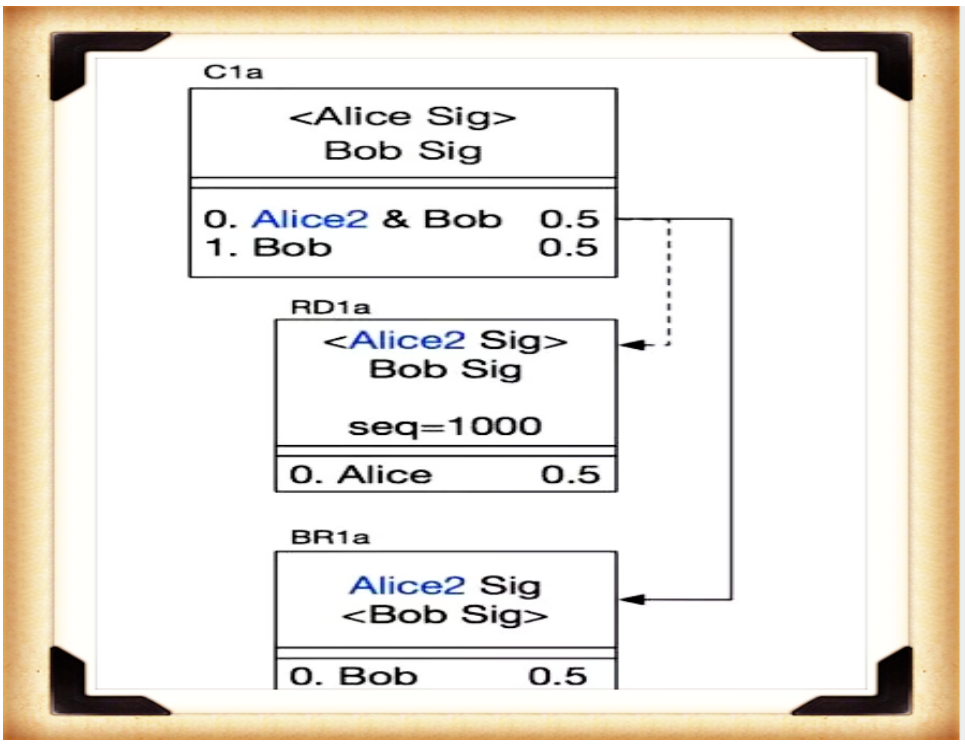
在 Alice 生成 C2a/RD2a 之前，他要把自己在 C1a 里面的私钥 Alice2 发给 Bob；同样，Bob 把自己的 C1b 里面的私钥 Bob2 发给 Alice。这样，各自会生成一个惩罚交易：

如下图所示：Alice 把秘钥 Alice2 给了 Bob，Bob 会为 C1a 生成 1 个惩罚交易 BR1a，攥在自己手里，以防 Alice 反悔。

注：此处 Alice 也不需要一定将私钥 Alice2 交给 bob，只需要生成一个惩罚交易 BR1a（已经带有私钥 Alice2 签名）交给 Bob，Bob 就可以通过自己的私钥进行签名，当 Alice 违规发布旧交易，则 Bob 就可以广播 BR1a 进行惩罚。Bob 也是一样操作。

假设 Alice 反悔，也就是把 C1a + RD1a 广播出去了，Bob 就把 BR1a 广播出去！！ BR1a 由于没有 Sequence，肯定会先于 RD1a 执行，所以结果是 RD1a 不会被执行，BR1a 执行了。造成的结果是，Alice 拿不回钱，Bob 会把 Alice 的 0.5 比特币全转账到自己账户里面，这就是对 Alice 的惩罚。反之亦然，会为 C1b 生成 BR1b。

一句话：BR1a 是 Bob 攒在手里的 Alice 的把柄，BR1b 是 Alice 攥在手里的 Bob 的把柄，任何 1 方都不敢把旧的交易广播出去。也就是一旦达成了 C2a/RD2a + C2b/RD2b，C1a/RD1a, C1b/RD1b 就废除了。



Step4: 同微支付通道一样，双方最终完成了交易，把 Step3 里面，最后 1 次更新，广播到网络上，各自得到自己的钱。最后 1 次的，sequence = 0，双方都立即拿到自己的钱。

（3）总结

通过上面的过程分析，我们可以看出，RSMC 设计的很巧妙：

1）通过双方各自往同一个公共账号打钱，实现了双向支付。

2）Alice 拿回钱的时候，没有直接打回到她自己的账号里面。而是打到 1 个

新的公共账号 Alice2 & Bob，然后再用一个有 sequence number 的 RD1a 最终拿回钱。通过这点，实现了谁主动中断，谁延迟退钱。这点做的很巧妙！！！

3）双方协商新一轮的时候，都把自己上一轮的私钥给对方，相当于把自己的把柄给了对方，这样双方都不敢反悔。

这里，又 1 个很巧的地方 :虽然 Alice 把私钥给了对方，但 Alice 不广播 C1a，那对方的处罚交易 BR1a 也不会执行。

Alice 广播了 C1a，对方就基于广播的交易执行处罚交易 BR1a；

Alice 不广播 C1a，账号 Alice2 & Bob 也就没有任何钱，因此对方也就没机会执行处罚交易 BR1a。

反之亦然！

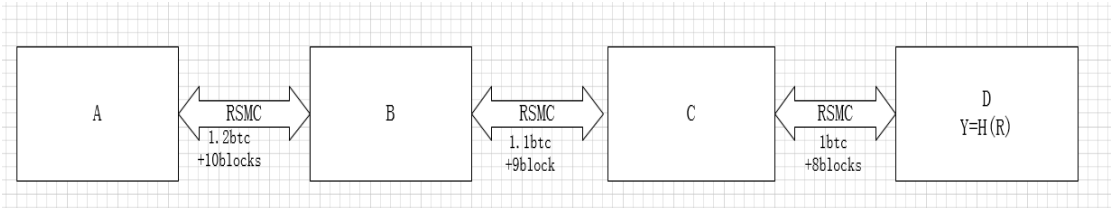
2.2 闪电网络之 HTLC

1.HTLC 引入

我们知道 RSMC 能够建立点对点链下双支付通道，这对于频繁交易来说十分有用，但如果针对很多一次性交易而言，却没有必要建立这样的支付通道，比如你外出旅游到咖啡店买一杯咖啡，这个时候如果建立 RSMC 支付通道意义就不是很大。同时任意两个人之间交易都要建立 RSMC 通道也确实很繁琐，有没有一种方式，不需要建立直接 RSMC 就可以完成点对点支付？HTLC（Hash Time Lock Contracts）就是这么一个解决方案，HTLC 解决方案基于已有的 RSMC 通道，采用哈希（hash）方式，在有限的时间内完成支付。HTLC 才让闪电网络成为真正的网络

2.HTLC 原理

我们基于如下图来解释一下 HTLC 的原理，想象一个场景，用户 A 需要给远在外地的用户 D 支付 1btc，A 与 D 之间并没有建立 RSMC 通道，但用户 A 很喜欢到自己家附件咖啡店 B 喝咖啡，所以 A 与 B 之间建立了 RSMC 通道；D 也喜欢去自家附近的咖啡店 C 喝咖啡，所以 C 与 D 之间建立了 RSMC 通道；咖啡店 B 和咖啡店 C 之间也有 RSMC 通道。



（1）用户 D 生成一个随机数 R，并通过哈希函数得到哈希值 Y=H(R)，R 只有用户 D 自己知道，并将哈希值 Y 给到用户 A。

（2）用户 A 与用户 B 形成一个智能合约：如果用户 B 在 3 天内能够提供哈希值 Y 的原像 R，则将支付给 B 1.2 个比特币，如果在期限内没有给出原像 R，则合约作废。

（3）用户 B 与用户 C 形成一个智能合约：如果用户 C 在 2 天内能够提供哈希值 Y 的原像 R，则将支付给 B 1.1 个比特币，如果在期限内没有给出原像 R，则合约作废。

（4）用户 C 与用户 D 形成一个智能合约：如果用户 D 在 1 天内能够提供哈希值 Y 的原像 R，则将支付给 B 1 个比特币，如果在期限内没有给出原像 R，则合约作废。

我们知道用户 D 是知道原像 R 的，所以用户 D 成功获得 1btc，同时用户 C 和 B 分别获得 0.1btc，就是所说的手续费，当然实际中，手续费并没有这么贵！

3.总结分析

（1）根据哈希函数的特性，尽管知道哈希值 Y，要直接找到原像 R 在计算上不可行的，这就保障在整个流程中不会出现作弊现象。

（2）在上述描述中，我们使用的是时间天数，这只是为了描述方便，在 HTLC 实际使用中，都是基于 block 数量而言。

（3）HTLC 技术才让闪电网络真的形成了网络，将独立 RSMC 通道联通，构筑成闪电网络。

（4）关于 HTLC 更深入的协议内容，感兴趣的读者可以参考白皮书。

3 隔离见证（Segarated Witness）

3.1 交易延展性（Malleability）攻击 -- 门头沟（前世界第 1 大比特币交易所）倒闭之罪魁祸首

3.1.1 延展性攻击解读 1

2014 年 2 月 25 日，日本时间上午 11 点，MT.GOX 交易所（俗语门头沟）停盘。众所周知，MT.GOX 曾经是比特币最大的交易所，一度交易量达到所有比特币交易的 80%，也是目前运营时间最长的交易所。门头沟的倒闭，源于黑客的攻击。整个丢失了 85 万数量的比特币，按当时的币价计算，这些损失的币价值近 4.54 亿美元。而这就是我们今天要讲的，大名鼎鼎的“交易延展性攻击”。

Transaction Malleability，翻译成“交易延展性”，也有人称为“交易可锻性”，这其实是个比喻。在现实生活中，一个金币在使用中，被人用锤子砸了几下，凹了几处，变得不是很圆。这个金币的本质含金量没变，只是外观看上去与标准的金币有些不同，这个金币依旧是一个被认可的金币。这就是“金币的可锻性”。

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：

(1) 黑客在交易所里面开了一个账号，并将一定数量的比特币转入交易所，然后申请提现，此时交易所发起一笔提现交易，该交易标识为 TXID1.

(2) 这笔交易 TXID1 被广播到网络上，还未打包进区块链之前。黑客收到这笔交易，更改了 scriptSig 的格式，换另一种有效签名，伪造一笔交易，交易标识为 TXID2，发送到网络。TXID2 交易的交易信息和 TXID1 交易一样，均是交易所向黑客转账。但交易所并不知道 TXID2 交易的存在。

(3) TXID1 交易与 TXID2 交易在网络中竞相处理，若 TXID2 交易先被处理，交易所的比特币会转入黑客账户中，TXID1 交易则失败。黑客如何保证伪造的

TXID2 交易会先被处理的，这里涉及到比特币网络的特性。比特币网络是去中心化的，节点与节点之间是点对点相连的，所有的交易请求都以网状广播的形式处理。黑客先查清楚交易所与哪些节点直连，用 DDOS 攻击瘫痪这些节点，使之不能对外广播交易信息。再伪造节点与交易所相连，用于广播 TXID2 交易。这样一来，TXID1 交易不会被广播到网络，网络中接收到的都是 TXID2 交易。

（4）黑客的这笔新交易 TXID2 被区块链接收了。然后向交易所投诉 TXID1 并没有收到，交易所确认 TXID1 交易失败后就会再次发送交易。如此一来，黑客就获得两笔比特币。

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“。

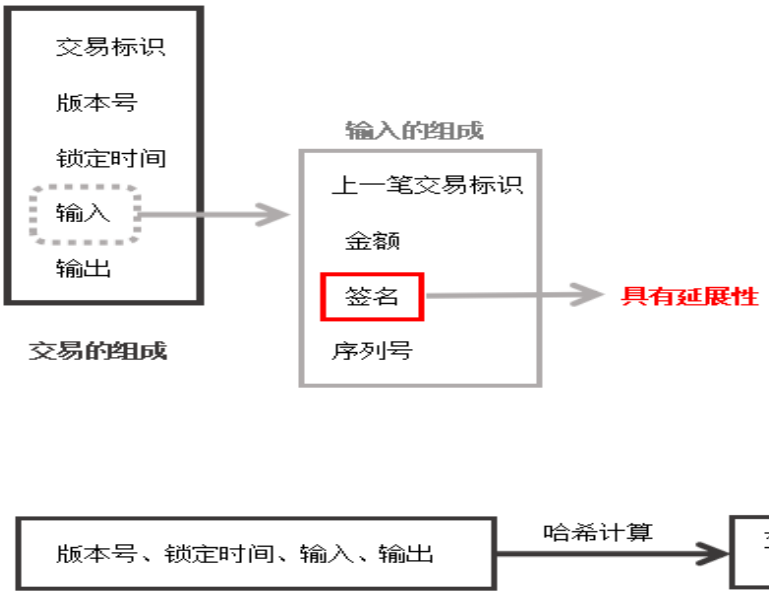
3.1.2 延展性攻击解读 2

要说延展性攻击，我们先来说说什么是延展性。延展性，又称为可锻性，是指一个东西的形状改变了，但其本质与质量没有发生改变，其本质属性仍然可以获得大众的认可。就如同一条含金量 7G 的项链，被高温熔铸后打造为一枚戒指，其黄金含量仍为 7G，其价值仍不变（不将生产成本纳入考虑）。

（1）比特币交易的可锻性

先简单说下比特币交易，一笔交易里由交易标识、版本号、锁定时间、输入、输出组成，后四者经过哈希计算会生成一个哈希值，这个值就是交易标识，即

TXID。每一个 TXID 就用来保证交易的唯一性。

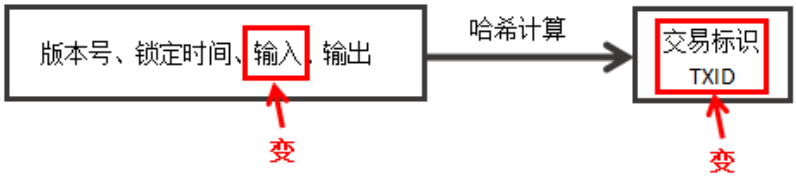


注：输入的组成中，金额指代 vout，上一笔交易的输出号，序列号指代 sequence。

每一笔交易的发起者都会用自己的私钥进行盖章，即常说的签名。签名也是一种算法计算，比特币用的是椭圆曲线数字签名算法(ECDSA)。矿工在验证一笔交易是否有效的时候经常就需要检查这个签名，检查的时候用的是 openssl 库校验用户签名。

问题就在于 openssl 可以兼容多种编码格式，ECDSA 的计算方式又不止一种，如用户用 signature(r,s)和 signature(r,-s(mod N)) 生成的签名数据是不一样的，但矿工都可以验证通过。这就是签名的延展性，即使签名数据改变了，但仍具有效性。

交易存在延展性，具体说来就是指一笔交易的签名具有延展性。签名改变了，签名作为输入的组成成分之一，交易里“输入”的组成数据也会有所不一样，那么哈希计算生成的交易标识 TXID 也会改变。



（2）延展性攻击怎么发生？

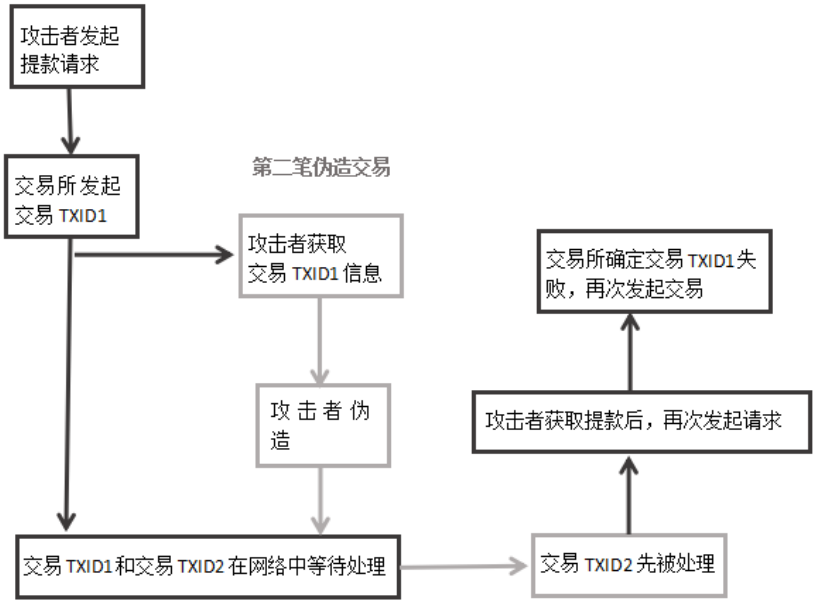
所谓的延展性攻击，就是通过对签名进行修改，但修改后的签名仍有效，攻击者可发起第二笔除了签名之外，其他数据信息均相同的交易。延展性攻击并不会阻止实际交易的进行，而是会引发二次转账交易。具体说来，延展性攻击的受害者通常是交易所。攻击的主要流程是这样的：

1) 黑客在交易所具有一个账户，其向交易所发出提款的请求，获得一个 TXID（先简称 TXID1）。

2) 黑客根据交易信息，换另一种有效签名，伪造一笔交易，交易标识为 TXID2，发送到网络。TXID2 交易的交易信息和 TXID1 交易一样，均是交易所向黑客转账。但交易所并不知道 TXID2 交易的存在。

3）TXID1 交易与 TXID2 交易在网络中竞相处理，若 TXID2 交易先被处理，交易所的比特币会转入黑客账户中，TXID1 交易则失败。

4) 黑客向平台表示 TXID1 交易失败，再次发起提款请求，交易所确认 TXID1 交易失败后就会再次发送交易。如此一来，黑客就获得两笔比特币。



关于第三点，黑客如何保证伪造的 TXID2 交易会先被处理的，这里涉及到比特币网络的特性。比特币网络是去中心化的，节点与节点之间是点对点相连的，所有的交易请求都以网状广播的形式处理。黑客先查清楚交易所与哪些节点直连，用 DDOS 攻击瘫痪这些节点，使之不能对外广播交易信息。再伪造节点与交易所相连，用于广播 TXID2 交易。这样一来，TXID1 交易不会被广播到网络，网络中接收到的都是 TXID2 交易。

为了更形象的去理解，举个例子：

> 1.肖恩让首饰店打造一条含金量为7G的金项链，首饰店向生产商发布消息A----打造一条含金量为7G的金项链。
> 2.肖恩通过某种手段截获了消息A，将消息A改为消息B----打造一枚含金量为7G的金戒指，再将消息B发送给生产商。
> 3.生产商接受消息B后，打造出金戒指，并发货给肖恩。
> 4.肖恩向首饰门店发起投诉，表示并没有收到金项链。首饰店向生产商对接，发现生产商的确没发金项链产品给肖恩，首饰店就会让生产商打造金项链，并发货给肖恩。

这样一来，肖恩就获得了两件含金量7G的产品。

3.2 隔离见证

上面我们详细介绍了比特币网络的一个漏洞："交易延展性攻击"，其原因就是 txid 在被确认之前，可以被修改。而隔离见证（Segarated Witness）方案的提出，就是为了修复这个漏洞。当然，隔离见证除了解决这个问题之外，还解决了"扩容"问题，以及其他一些问题。所以隔离见证，它是一个方案的集合，本篇就对这个方案的集合进行一个详细描述。

（1）隔离见证(Segarated Witness)一词的由来

隔离见证这个词，初听起来，不明觉厉。其实呢，没那么复杂。Witness，见证，其实就是我们在第 10 课 交易安全性如何保证？ --scriptSig/scriptPubKey/Script Engine 中提到的, scriptSig。回顾一下前面的内容，1 个 Transaction 有多个 input，多个 output；

每个 input 里面有 1 个 scriptSig（对应付款人的私钥签名）；

每个 output 里面有 1 个 scriptPubKey（对应收款人的公钥 Hash）。

那为什么把 scriptSig 叫做 Witness 呢？这个是密码学领域的一个对 scriptSig 的更 general 的一个称呼，此处就不深入研究了。

隔离，Segatated，就是指把这个 scriptSig 从每个 Transaction 的 input 里面拿出来，放到别的地方去，不要和 transaction 放在一起。拿出来放到什么地方去呢？？？放到整个 Block 的尾部。也就是说，一个 Block 的数据现在有 3 部分组成：

Block = Block Header + 所有的交易数据 + 所有交易的所有 input 对应的 witness

以上就是隔离见证这个词的由来。

（2）隔离见证实现的扩容效果

关于隔离见证，网上一个很大的误解就是 :认为 witness 被隔离走了，witness 数据不在 Block 里，所以一个 Block 能装更多的 Transaction。

其实不是，witness 数据仍然在 Block 里面。并且对于 1 个 Transaction 来说，如果把 witness 数据也算上的话，其 raw byte size 其实是变大了，而不是变小了！！！既然 Transaction 还变大了，那为什么 1 个 Block 可以装更多的 Transaction 呢？？

因为隔离见证是软分叉，不是硬分叉，下面就分别来分析一下，为什么对于老版本节点、新版本节点，1 个 Block 都可以装更多的 Transaction 呢？

1）对于老版本节点：

Block Limit Size = 1M，但由于你把 witness 数据移到了所有 Transaction 的外面，放在了整个 Block 的尾部。老版本在计算一个 Block 大小的时候，只计算了 Block Header + 所有 Transaction 的数据（witness 数据，老版本看不见！！！相当于老版本被欺骗了。）所以其实整个 Block 的物理大小(raw block size)已经超过了 1M，但老版本的节点不认识尾部的 witness 数据，所以认为总大小还是 < 1M。

2）对于新版本节点：

Block 的 size 的计算方式做了调整，引入了 Block weight 的概念。

block weight = base_size * 4 + witness_size

block weight <= 4M

其中，base_size 就是 block 的前 2 部分数据（header + 没有 witness 的所有交易数据）

通过上面的分析，我们会发现，数据还是那么多数据，没有减少，只是重新

排布了一下，却变相的把区块链扩容了！！！

（3）隔离见证解决的几大问题

1）交易延展性攻击

因为把 scriptSig 移到外面去了，scriptSig 变成了空值，那么计算出来的 txid 也就不可能改变了。txid 不可能改变，也就解决了交易延展性攻击。

2）扩容

在前面我们说过了，1 个 Block 最多 1M，也就装 2000 多笔交易，每 10 分钟产生 1 个新区块，意味着 1 秒钟就才处理 3 到 4 笔交易。这使得现在的比特币网络已经满负荷运行，很多交易要排队等待被打包确认，比特币网络的扩容迫在眉睫。而 scriptSig 呢，其实占了 1 个 Transaction 的很大一部分空间。这是为什么呢？回想第 11 课　账号被黑客盗取怎么办？ -- 多重签名(MultiSig)与 P2SH 所讲的 P2SH 交易，scriptSig 是个 redeem script，尤其在多重签名，或者前面讲的 RSMC 之类的交易中，占了很大一部分空间，所以把它移出去，的确可以节省很大的空间。

3）增加了 script version

类似 Block, Transaction，隔离见证为 Script 也引入了版本号，这使得 Script Language 也可以以一种后向兼容(backward-compatible)的方式来发展。

4）签名算法复杂度有了大的优化

这个涉及到签名算法，后面有机会再仔细讲这个问题。

5）网络和存储的扩容

因为 witness 只在交易的验证环节需要，其他环节未必用到。所以在其他环节，可以不存储 witness。这对于磁盘和网络传输，也有扩容作用。

（4）隔离见证的争议

1）起初隔离见证是作为一种“硬分叉”方案来搞的，也就是 1 刀切，不考虑向后兼容，但后来又演变成了软分叉。软分叉虽然实现了平滑切换，但技术实现很复杂，为了兼容老版本，设计上也有很多妥协，这就增加了出 Bug 的可能性。

2）Any One Can Pay

我们知道，隔离见证是作为"软分叉”来部署的，新老版本节点同时存在。那新版本产生的，带有 witness 的 Transaction，老版本的节点怎么处理呢？

答案是：为了兼容老版本，这种 Transaction 的验证永远为 True，也就是任何人都可以花这笔交易。但由于 95%以上的节点都是新版本的节点，所以即使老版本的节点全部无条件的接受这些 Transaction，也没关系。在新版本的节点上，会经过完全的验证。

但假设这样一种情况 :隔离见证激活之后, 假如发现了这个技术有重大 bug, 要回滚。那就意味着所有的节点，都会抢着去花那些 witness transaction，因为是 Any one can pay。这对整个比特币网络，将是一个灾难！!!