比特币源码研读系列3 创建tx(1)

今天来说说tx是如何创建的。前面我们也已经反复强调tx的重要性，因此创建tx的过程是一个非常重要的过程，在看代码之前我们来讨论下一个创建tx需要做哪些事：

1. **Find my spendable money**
2. **创建tx。**
3. **广播tx**

下面我们通过代码来看看这个过程，由于代码非常多，下面我只针对一些重要或者难以理解的方法进行重点介绍。

**一 Find my spendable money**

为了更好地理解，我们先来介绍几个重要的方法。

1. IsMine方法。该方法用于检查某个coin是否是我的。其中入参keystore指的是与signature creator签名生成器有关的，scriptPubKey是用我的公钥加密生成的加密串，只有我能解锁出来才证明该coin是属于我的，sigversion是加密版本，该方法会返回该coin的状态如未消费的，已消费的，不能解锁等状态。

*isminetype IsMine(const CKeyStore &keystore, const CScript& scriptPubKey, bool& isInvalid, SigVersion sigversion)*

*{*

*isInvalid = false;*

*std::vector<valtype> vSolutions;*

*txnouttype whichType;*

*if (!Solver(scriptPubKey, whichType, vSolutions)) {****//Solver方法就是用我的私钥来解锁sciptPubKey串，如果解锁成功说明收款人身份验证通过，说明该coin就是支付给我的。至于脚本具体是如何解锁的，感兴趣的读者可参考这篇文章***[***https://en.bitcoin.it/wiki/Script***](https://en.bitcoin.it/wiki/Script)***。***

*if (keystore.HaveWatchOnly(scriptPubKey))*

*return ISMINE\_WATCH\_UNSOLVABLE;*

*return ISMINE\_NO;*

*}*

*CKeyID keyID;*

*switch (whichType)****//根据script类型进行相应处理***

*{*

*…..****//此处源码非常多，主要是如何解锁script的，这里节约篇幅我就不贴出来了***

*}*

*return ProduceSignature(DummySignatureCreator(&keystore), scriptPubKey, sigs) ? ISMINE\_WATCH\_SOLVABLE : ISMINE\_WATCH\_UNSOLVABLE;****//*** ***ProduceSignature方法作用是用我的私钥对该coin进行签名，签名数据保存在变量sigs中，接下来可能会需要，如果我要消费该coin，就用这个签名来授权使用。***

*}*

*return ISMINE\_NO;*

*}*

1. GetDebit方法。该方法是把该tx的所有vin所指向的prev tx的vout中收款人是我的那部分资产之和，它代表的是我在当前tx中所使用的资产，注意不是花费的资产，因为还存在有找零的情况。

*CAmount CWallet::GetDebit(const CTransaction& tx, const isminefilter& filter) const*

*{*

*CAmount nDebit = 0;*

*for (const CTxIn& txin : tx.vin)*

*{*

*nDebit += GetDebit(txin, filter);* ***//对每个引用的txin计算我使用的金额。因此nDebit就是我在当前tx所使用的全部金额***

*if (!MoneyRange(nDebit))*

*throw std::runtime\_error(std::string(\_\_func\_\_) + ": value out of range");*

*}*

*return nDebit;*

*}*

*CAmount CWallet::GetDebit(const CTxIn &txin, const isminefilter& filter) const*

*{*

*{*

*LOCK(cs\_wallet);*

*std::map<uint256, CWalletTx>::const\_iterator mi = mapWallet.find(txin.prevout.hash);****//mapWallet记录着与我有关(包含进和出)的全部tx。***

*if (mi != mapWallet.end())****//在mapwallet找到txin说明与我有关，接下来在它的vout中找到属于我的那部分金额。***

*{*

*const CWalletTx& prev = (\*mi).second;*

*if (txin.prevout.n < prev.tx->vout.size())*

*if (IsMine(prev.tx->vout[txin.prevout.n]) & filter)*

*return prev.tx->vout[txin.prevout.n].nValue;****//返回消费的金额***

*}*

*}*

*return 0;*

*}*

1. checkFinalTx方法。检查tx是否还处于锁定状态，tx有个nLockTime是BIP113的一个新功能，通过参数配置该更新是否启用。如果启用了，则当tx处于锁定期返回false，表示该tx暂时还不可用。

*bool CheckFinalTx(const CTransaction &tx, int flags)*

*{*

*AssertLockHeld(cs\_main);*

*flags = std::max(flags, 0);****//flags控制nLockTime是否启用***

*const int nBlockHeight = chainActive.Height() + 1;*

*const int64\_t nBlockTime = (flags & LOCKTIME\_MEDIAN\_TIME\_PAST)*

*? chainActive.Tip()->GetMedianTimePast()*

*: GetAdjustedTime();****//GetMedianTimePast是当前活跃链的前11个Block按时间从小到大排序取第6块的时间，GetAdjustedTime是当前unix time，如果nLockTime未启用，则用nLockTime与当前unix time做比较。***

*return IsFinalTx(tx, nBlockHeight, nBlockTime);*

*}*

*bool IsFinalTx(const CTransaction &tx, int nBlockHeight, int64\_t nBlockTime)*

*{*

*if (tx.nLockTime == 0)****//如果nLockTime未设置说明该功能未启用，则返回true表示该tx可用。***

*return true;*

*if ((int64\_t)tx.nLockTime < ((int64\_t)tx.nLockTime < LOCKTIME\_THRESHOLD ? (int64\_t)nBlockHeight : nBlockTime))****//*** ***LOCKTIME\_THRESHOLD = 500000000;如果nLockTime小于这个值则nLockTime表示该tx在活跃链的深度，如果大于则表示unix time。((int64\_t)tx.nLockTime < LOCKTIME\_THRESHOLD ? (int64\_t)nBlockHeight : nBlockTime)可以理解成currentTime，因此如果nLockTime<currentTime，说明当前时间已经超过了tx的锁定时间，该tx可用，返回true。***

*return true;*

*for (const auto& txin : tx.vin) {****//如果tx还处于锁定状态，检查下tx的nSequence字段，这里看上去也能通过设置nSequence强制启用tx。***

*if (!(txin.nSequence == CTxIn::SEQUENCE\_FINAL))*

*return false;*

*}*

*return true;*

*}*

1. IsTrust方法。该方法表示某个tx是否是spendable可以花费的状态，true表示是。

*bool CWalletTx::IsTrusted() const*

*{*

*// Quick answer in most cases*

*if (!CheckFinalTx(\*this))* ***//检查tx的锁定时间。如果false表示tx还不能使用***

*return false;*

*int nDepth = GetDepthInMainChain();****//nDepth为tx所在Block在主链中的位置。nDepth>=1则意味着已经在Block中。nDepth<0表示tx有冲突，该tx不能用。***

*if (nDepth >= 1)*

*return true;*

*if (nDepth < 0)*

*return false;*

*if (!bSpendZeroConfChange || !IsFromMe(ISMINE\_ALL))* ***//IsFromMe检查是不是我spent，如果不是我spent，则not trusted***

*return false;*

*// Don't trust unconfirmed transactions from us unless they are in the mempool.*

*if (!InMempool())****//既不在Block也不在mempool中则tx不可信***

*return false;*

*// Trusted if all inputs are from us and are in the mempool:*

*for (const CTxIn& txin : tx->vin)****//对tx中的每个 txin进行校验，是否我签名授权使用的，这里为什么还要检查tx引用的coin是否可信呢，我们在上一篇文章也提到了，主要是为了防止别人在没有成功解锁签名信息时就当成是自己的coin来spend。***

*{*

*// Transactions not sent by us: not trusted*

*const CWalletTx\* parent = pwallet->GetWalletTx(txin.prevout.hash);*

*if (parent == nullptr)*

*return false;*

*const CTxOut& parentOut = parent->tx->vout[txin.prevout.n];****//parentOut是txin***

*if (pwallet->IsMine(parentOut) != ISMINE\_SPENDABLE)****//IsMine方法前面我们已经介绍过，它会检查coin是不是属于我的，且该coin是什么状态（大概有可消费的，已消费的等多状态）****。****如果不是可以spend状态则coin不可信。***

*return false;*

*}*

*return true;*

*}*

未完待续…