Описание работы **BITCOIN**.

engi@sumus.team

18 сентября 2019 г.

1 Транзакции

1.1 Структура tx

Структура tx является носителем транзакций. tx объединяет приходящие и исходящие переводы монет. При формировании tx исходящие переводы будут все $UTXO$ — *непотраченные*. Структура tx содержит два массива структур: tx_in и tx_out.

1.2 Структура tx_in

В структуре есть записи: $prevHash$, $prevIndex$, $scriptSig$.

Запись $prevHash$ идентифицирует структуру tx где находится tx_out.

Запись $prevIndex$ указывает на индекс массива $tx\_out[]$ для определения структуры tx_out.

Запись $scriptSig$ является первой частью подтверждающего скрипта.

1.3 Структура tx_out

В структуре есть записи: $value$ и $scriptPubKey$.

Запись $value$ декларирует количество монет принадлежащих данной записи.

Запись $scriptPubKey$ является второй частью подтверждающего скрипта.

1.4 fee

Значение fee вычисляется как сумма монет, полученных по tx_in Σ_{in} минус сумма монет, полученных по tx_out Σ_{out}. Очевидно, что Σ_{in} не может быть меньше Σ_{out}. Суммарное значение всех fee по блоку является неявным источником монет для структуры tx_0.

1.5 Структура tx_0 (*coinbase*)

Структура tx с индексом 0 в блоке имеет особые свойства. tx_0 получает монеты за закрытие блока ($reward$) и как плату за проведение транзакций (Σfee). *Не запрещено получать монеты и от tx_out источников в качестве дополнительного источника.*

$reward$ обозначен явно с особыми значениями $prevHash = 0$, $prevIndex = -1$. Скрипт *$scriptSig$ может быть заполнен произвольно* и может служить дополнением $nonce$. Ожидаемое количество монет за закрытие блока вычисляется по формуле в зависимости от порядкового номера блока.

Σfee представляет собой сумму fee от остальных tx структур блока. Отдельный fee может быть нулевым, могут быть нулевыми все fee, в блоке может быть только одна tx структура. В этом случае Σfee не добавляет монет. *Конечно, для* tx_0 fee *не вычисляется.*

1.6 Формирование структуры tx

Для перевода монет от одного или нескольких источников tx_out требуется подтвердить указываемые связи. Один tx_out $UTXO$ может быть связан только с одним tx_in. После установления связи такой tx_out считается *потраченным* и перестаёт быть $UTXO$.

Подтверждением связи будет формирование скрипта $scriptSig$ в tx_in, который при конкатенации со скриптом $scriptPubKey$ из tx_out образует скрипт, дающий единственный результат $TRUE$.

Возможно написание $scriptPubKey$ такого, что к нему невозможно написать преамбулу–$scriptSig$ которая даст результат $TRUE$. Так формируется tx_out, который невозможно "потратить".

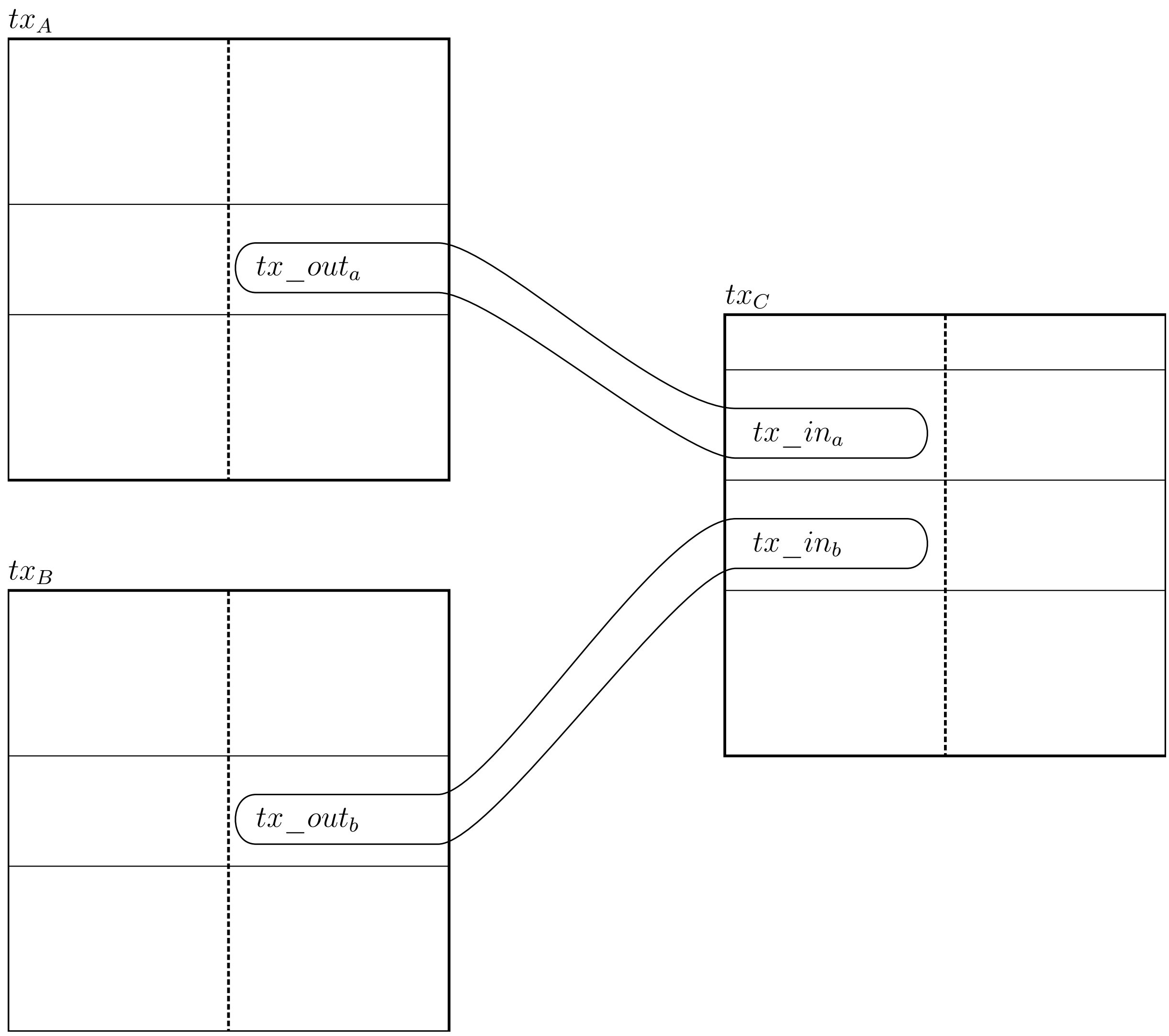

Рис. 1: Составление транзакции.

1.7 Скрипты BITCOIN

BITCOIN для подписывания переводов монет использует гибкий подход. Так называемый “скрипт” исполняется стековым процессором, напоминающим процессор языка $FORTH$ без ветвлений. Команды процессора содержат (кроме обычных) вычисление криптографических величин. Это позволяет использовать криптографию для подписывания переводов.

2 Блоки *block*

Структуры tx упакованы в блоки. Последовательность блоков называется блокчейн. Формирование блока является ответственным процессом. Структуры tx должны быть проверены упаковщиком блока (майнером). Майнер связывает данный блок с предыдущим блоком путём указания хеша на предыдущий блок, формирует дерево Меркла хешей структур tx блока и подгоняет $nonce$ и $tx_0.tx_in.scriptSig$ так, чтобы хеш всего блока удовлетворял условию сложности.

2.1 Структура *block*

Блок состоит из заголовка и массива структур tx:

- $block\_header$ заголовок блока 80 байт
- $tx[]$ массив структур tx

Структура заголовка:

- $version$ Версия типа блока
- $prev\_block$ Хеш предыдущего блока. *Note: блок не содержит собственного хеша.*
- $merkle\_root$ Хеш дерева Меркла структур $tx[]$.
- $timestamp$ *(несущественная величина)*
- $bits$ сложность майнинга
- $nonce$ подгонка хеша
- $tx[]$ массив структур tx

2.2 Дерево Меркла (*ДМ*) и *SPV* (*Simplified Payment Verification*)

К существенным свойствам *ДМ* отнесём:

- неизменяемость значений элементов *ДМ*
- неизменяемость расположения элементов *ДМ*

Если *НОД* использует *SPV* то ему достаточно синхронизировать только цепочку $block\_header$. В принципе, *НОД* может первоначально синхронизировать блоки, но, убедившись в целостности содержимого блока, локально сохранять только $block\_header$. Для количества блоков $6 * 10^5$, объём хранимой информации составит $6 * 10^5 * 81 = 5 * 10^7$ или 50 МБайт. Такой объём вполне помещается на самых скромных мобильных устройствах.

Для того, чтобы продемонстрировать существование транзакции (выполнение платежа) достаточно получить от системы или контрагента цепочку хешей по одному на каждый уровень *ДМ*. Затем, посчитать хеш от проверяемой транзакции и кумулятивные хеши до получения хеша *ДМ* блока. Полученный хеш должен совпасть с хешем одного из $block\_header.merkle\_root$.

Очевидна недопустимость наличия одинаковых хешей *ДМ* блока в блокчейне и одинаковых хешей tx.

Первое условие выполняется при выполнении второго условия. Действительно, при разных хешах tx, хеши *ДМ* тоже будут разные.

Второе условие выполняется так: tx_in содержит записи $prevHash$ и $prevIndex$, которые не могут быть одновременно одинаковыми. Тогда бы они ссылались на одну структуру tx_out, а это запрещено (двойная трата).

3 Передача монет

3.1 "Адрес" кошелька

В регистре BITCOIN нет явно определяемого универсального адреса отправителя/получателя монет. Гибкость формирования сущности "транзакция" позволяет задавать несколько разных источников монет и несколько разных получателей монет.

Разные источники монет позволяют добавлять/объединять монеты при их получении. При этом, владелец одного источника будет фактическим получателем, а владельцы других источников будут отправителями. Можно предположить, что такая транзакция "не знает" заранее куда она сама будет переводить монеты. Поэтому эта транзакция будет иметь один tx_out.

Разные получатели монет позволяют разделить монеты при их передаче. Такая необходимость возникает при трате "со сдачей". При этом один получатель — это владелец монет и получает он "сдачу". Остальные получатели (возможно только один получатель) — это фактические получатели монет.

Кроме того, возможны более сложные комбинации количества, принадлежности и аутентификации участников транзакции. Также, возможно создать tx_out такой, который невозможно "потратить".

Эти условия усложняют идентификацию (получение "адреса" кошелька) участников.

Ничто не мешает лицу-владельцу кошелька использовать разные $scriptPubKey$ (с разными фактическими публичными ключами). Такая стратегия потребует лишь увеличения персонального хранилища приватных ключей и более сложного процесса выполнения транзакций. Можно хранить только ключи, связанные с непотраченными tx_out.

> При совершении очередной операции с BTC чаще всего создается новый адрес ...[1]
>
> Постоянный "адрес" кошелька может потребоваться, например, для автоматизированного интернет-магазина, когда получатель монет не согласовывается при каждом индивидуальном переводе.
>
> Также, постоянный "адрес" кошелька удобен при использовании приложения для индивидуального применения.

[1] https://ru.bitcoinwiki.org/wiki/Адрес [Для чего нужен биткоин адрес]

3.2 Идентификация участников транзакции

tx_in помогает определить источники монет. Эта запись указывает на tx_out предшествующей транзакции.

tx_out содержит $scriptPubKey$, *при анализе которого можно получить* (не всегда) публичный ключ $pubKey$, который идентифицирует владельца монет.

Итак, в момент существования транзакции:

- $pubKey$, указанный в записи tx_out идентифицирует владельца монет
- $pubKey$, указанный в записи tx_out на которую ссылается tx_in идентифицирует отправителя монет