跨链原子交换

作业介绍

alice和bob拥有btc testnet密钥和bcy testnet密钥。用keygen生成btc密钥，用

`curl -X POST https://api.blockcypher.com/v1/bcy/test/addrs?token=7c4041024be5449b84b4c353121fc9b3` 生成他们的bcy密钥。在coinfaucet上为alice生成btc的测试币，在blockcypher上为bob生成bcy的测试币。本次作业的任务就是在bcy和btc的两个区块链上实现虚拟币的交易。

代码解释

两种交易的情况分别是

赎回交易和交易退还。

1. 对于赎回交易，我们需要用到recipient和对应的密钥：

   解锁脚本应该如下：

   ```
   def coinExchangeScriptSig1(sig_recipient, secret):
       return [sig_recipient,secret]
   ```

2. 对于交易退还，需要sender和recipient分别的签名

   解锁脚本如下：

   ```
   def coinExchangeScriptSig2(sig_sender, sig_recipient):
       return [
       OP_0,sig_sender,sig_recipient
       ]
   ```

加锁脚本针对两种不同情况来分别设计，使用了比特币脚本的OP_IF和OP_ELSE来进行判断，首先需要判断解锁脚本的长度，分别是2和3 。设计比特币的if的条件是OP_DEPTH，如果长度为2就跳入第一种加锁逻辑，如果长度为3就跳入第二种加锁逻辑。

交易脚本如下：

```
def coinExchangeScript(public_key_sender, public_key_recipient, hash_of_secret):
    return [
    OP_DEPTH, 2, OP_EQUAL,
    OP_IF, OP_HASH160, hash_of_secret, OP_EQUALVERIFY,
    public_key_recipient,OP_CHECKSIG,
    OP_ELSE,2, public_key_sender,
    public_key_recipient, 2, OP_CHECKMULTISIG,OP_ENDIF]
```

针对上面两种情况，比特币脚本的入栈情况大概如下：

1. 情况一：进入第一个if判断

```
<sig_recipient><seret> //解锁脚本入栈
<sig_recipient><seret><2> //利用OP_DEPTH得到栈大小为2
<sig_recipient><seret><2><2> //加锁脚本中的2入栈
<sig_recipient><seret><True> //加锁脚本比较栈顶两个2发现相等，压栈true
<sig_recipient><seret> //OP_IF检测到True
<sig_recipient><hash_of_secret>//使用函数中的hashofsecret
<sig_recipient><hash_of_secret><hash_of_secret>//hash_of_secret压栈
<sig_recipient> |OP_EQUALVERIFY//检测栈顶两个值相等
<sig_recipient><public_key_recipient>//public_key_recipient入栈
<True> |OP_CHECKSIG //检查签名
<True> |OP_ENDIF
```

2. 情况二：进入else判断

```
<OP_0><sig_recipient><sig_recipient> //解锁脚本入栈
<OP_0><sig_recipient><sig_recipient><3>     //OP_DEPTH得到栈大小为，将3压栈
<OP_0><sig_recipient><sig_recipient><3><2> //2入栈
<OP_0><sig_recipient><sig_recipient><False>//将False压栈
<OP_0><sig_recipient><sig_recipient> //有OP_ELSE，跳转到else分支
<OP_0><sig_recipient><sig_recipient><2>  // 需要2个人的签名
<OP_0><sig_recipient><sig_recipient><2><public_key_sender>
<public_key_recipient>
<True> //OP_CHECKMULTISIG进行验证，成功返回Ture
<Ture> //OP_ENDIF
```

问题

以 Alice 用 coinExchangeScript 向 Bob 发送硬币为例：如果 Bob 不把钱赎回来，Alice 为什么总能拿回她的钱?

答：当 Alice 用 原子交换向 Bob 发送硬币时，如果 Bob 不履行承诺，Alice 将不会在交易上签名，Bob 就不能拿到 Alice 的硬币。Alice就可以赎回虚拟币。

为什么不能用简单的 1/2 MultiSig 来解决这个问题?

答：因为如果使用的 1/2 multsig，两方都可以不履行约定，只取虚拟币。

使用以上机制可以保证没有任何一方可以单独毁约，只靠自己赎回硬币。因此双方的利益都不会受损。

解释Alice（Bob）创建的一些交易内容和先后次序，以及背后的设计原理

1. Alice 创建第一笔交易，指定解锁方式为：双方共同签名，或 Bob 的签名加上 Alice 创建的secret
2. Alice 创建第二笔交易，指定解锁方式为：在到达指定的解锁时间 locktime 之后，可以赎回自己的币.
3. Bob 创建第一笔交易，指定解锁方式为：双方共同签名，或 Alice 的签名加上 Alice 创建的秘密 x.（使用Hash(x))
4. Bob 创建第二笔交易，指定解锁方式为：在到达指定的解锁时间之后，可以赎回自己的币.

原理：

1. 在 Alice 创建交易后，Bob 需要知道 Alice 的secret才可
2. 在 Bob 创建交易后，Alice 可以索要 Bob 的硬币，使用其签名和secret来解锁 Bob 的第一笔交易。此时secret将会被传到区块链上。这时 Bob在得知区块链上的secrete之后也就可以解锁第一

笔交易，完成交易了。

本次作业中，一次成功的跨链原子交换中，资金是如何流转的

- 对于用secret x及接收者签名赎回的脚本，我们只需要先验证secret x的哈希值是否正确，然后再用P2PK脚本验证接收者签名
- 对于用两个签名赎回的脚本，我们只需要用P2MultiSig就可以实现
- 对于知道秘密X的解锁脚本，我们只需将签名与秘密依次压栈
- 对于两个签名的解锁脚本，把两个签名压栈即可

1. Alice 生成secret x，将自己用于交换的 BTC 锁定在输出脚本
2. Bob 使用哈希函数，将自己的 BCY 锁定在输出脚本中
3. Alice 使用secret x解锁 Bob 的输出脚本，获取 Bob 的 BCY 到自己的钱包中，并且广播
4. Bob 使用 x 解锁 Alice 的输出脚本，获取 Alice 的 BTC 到自己的钱包中，交易结束

交易进行与结果

```
liujc@LAPTOP-5L4A5MGE:~/Desktop/blockchain/Exercise4$ python3 swap.py
Alice swap tx (BTC) created successfully!
Bob swap tx (BCY) created successfully!
Alice redeem from swap tx (BCY) created successfully!
Bob redeem from swap tx (BTC) created successfully!
```