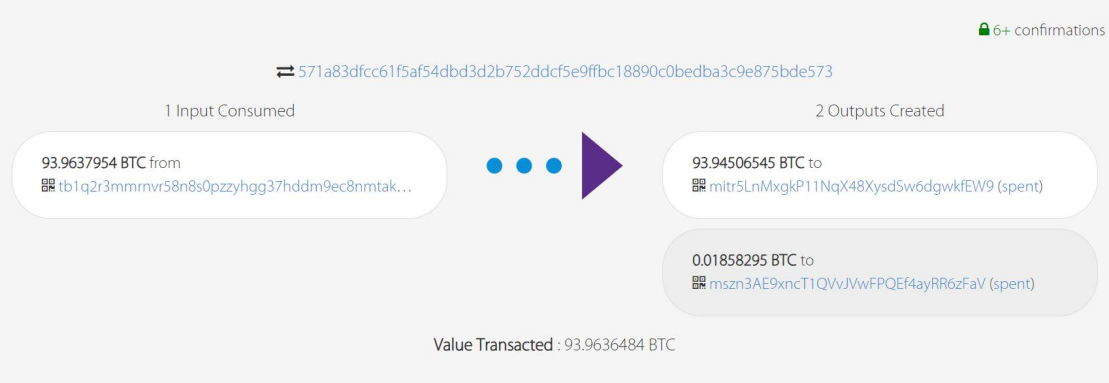
第一次实验报告

2111460 张洋

一、获取比特币

1. 使用 keygen.py 生成一个 testnet 私钥和地址，在 faucet（https://coinfaucet.eu/en/btctestnet/）粘贴得到的地址，得到一些 testnet BTC：



Private key:

cQmwgAmqUwuEZaQ3AVSAuKjvB3gURCDD8cTmU4DDz 44ya1ETKfWe

Address:

mszn3AE9xncT1QVvJVwFPQEf4ayRR6zFaV

交易哈希值:

571a83dfcc61f5af54dbd3d2b752ddcf5e9ffbc18890c0bedba3c9e875bde573

2. 将上一步得到的交易哈希值在 https://live.blockcypher.com/网址中查询交易，可以得到结果如下

Block Hash	0000000000006e617c5e407ebe5296099673aa90b7ff5e8a56be34b12c83d04d
Block Height	2,477,600
Transaction Index	18 (permalink)
Size	228 bytes
Virtual Size	147 vbytes
Lock Time	2477599
Version	2
Relayed By:	185.232.70.226:18333

二、分币

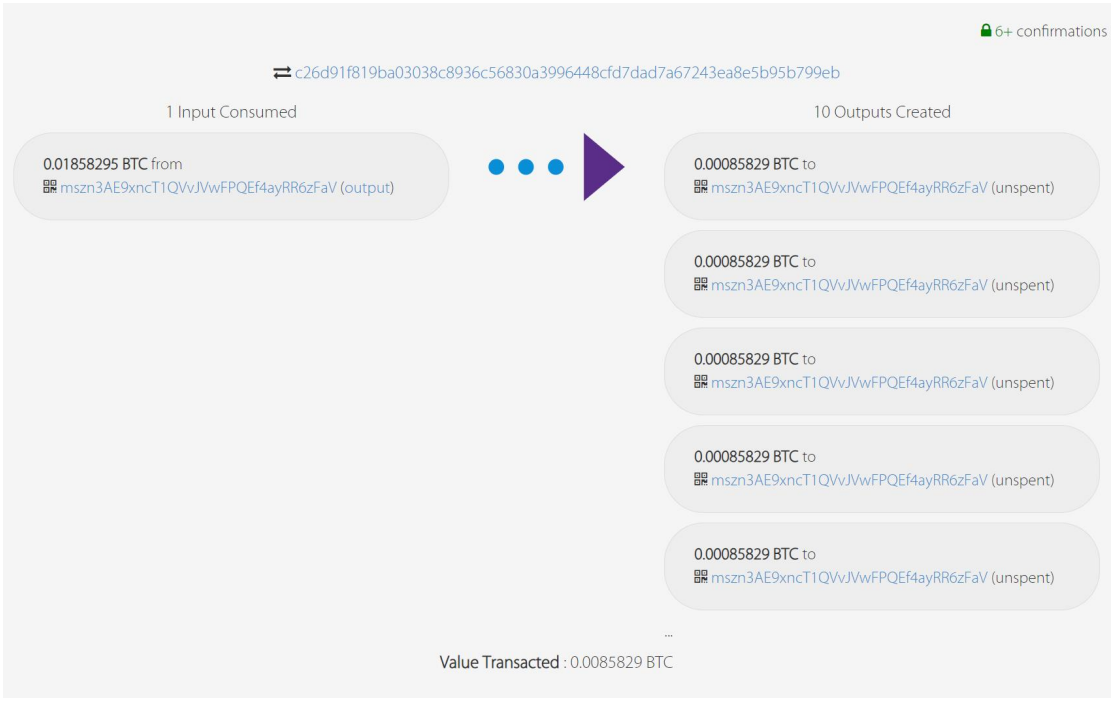
1. 修改 config.py 文件，将 my_private_key 修改为第一步中得到的私钥。

```python
from bitcoin import SelectParams
from bitcoin.base58 import decode
from bitcoin.wallet import CBitcoinAddress, CBitcoinSecret, P2PKHBitcoinAddress


SelectParams('testnet')

# TODO: Fill this in with your private key.
my_private_key = CBitcoinSecret(
    'cQmwgAmqUwuEZaQ3AVSAuKjvB3gURCDD8cTmU4DDz44ya1ETKfWe')
my_public_key = my_private_key.pub
my_address = P2PKHBitcoinAddress.from_pubkey(my_public_key)

faucet_address = CBitcoinAddress('mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB')
```

2. 修改 split_test_coins.py 文件，txid 为第二步中得到的哈希值；将第一步中得到的比特币数额减去 0.01 后平均分为 10 份，使用第二个输出（utxo_index=1）。

```python
from bitcoin.core.script import *

from utils import *
from config import (my_private_key, my_public_key, my_address,
                    faucet_address)


def split_coins(amount_to_send, txid_to_spend, utxo_index, n):
    txin_scriptPubKey = my_address.to_scriptPubKey()
    txin = create_txin(txid_to_spend, utxo_index)
    txout_scriptPubKey = my_address.to_scriptPubKey()
    txout = create_txout(amount_to_send / n, txout_scriptPubKey)
    tx = CMutableTransaction([txin], [txout]*n)
    sighash = SignatureHash(txin_scriptPubKey, tx,
                            0, SIGHASH_ALL)
    txin.scriptSig = CScript([my_private_key.sign(sighash) + bytes([SIGHASH_ALL]),
                              my_public_key])
    VerifyScript(txin.scriptSig, txin_scriptPubKey,
                 tx, 0, (SCRIPT_VERIFY_P2SH,))
    response = broadcast_transaction(tx)
    print(response.status_code, response.reason)
    print(response.text)

if __name__ == '__main__':
    ######################################################################
    # TODO: set these parameters correctly
    amount_to_send = 0.01858295-0.01 # amount of BTC in the output you're splitting minus fee
    txid_to_spend = (
        '571a83dfcc61f5af54dbd3d2b752ddcf5e9ffbc18890c0bedba3c9e875bde573')
    utxo_index = 1
    n=10 # number of outputs to split the input into
    ######################################################################

    split_coins(amount_to_send, txid_to_spend, utxo_index, n)
```

3. faucet 截图



Block Hash	0000000000000012d42b361e832aa4a924619948421c01805abe208a7eabcb47
Block Height	2,477,601
Transaction Index	1 (permalink)
Size	497 bytes
Virtual Size	497 vbytes
Lock Time	
Version	1
Relayed By:	60.29.153.61

4. 交易输出存在了“分币交易信息.txt”文件中。

三、发币

1. 修改 ex1.py 文件，具体修改如下：

（1）P2PKH_scriptPubKey 函数主要用于生成一个标准的 PayToPublicKeyHash 交易的输出脚本 scriptPubKey。该部分主要由以下几部分构成：

- OP_DUP：将栈顶元素复制一份，压入栈中，便于后续的验证操作；
- OP_HASH160：计算栈顶元素的 RIPEMD160 哈希值，并将结果压入栈中；
- pubkey_hash：接收方的比特币地址的公钥哈希；
- OP_EQUALVERIFY：比较栈中的两个元素是否相等，如果相等，继续执行，否则交易失败；
- OP_CHECKSIG：验证发送方的数字签名是否与公钥匹配，如果匹配，交易有效，否则交易失败

```
def P2PKH_scriptPubKey(address):
    ###############################################################################
    # TODO: Complete the standard scriptPubKey implementation for a
    # PayToPublicKeyHash transaction
    return [OP_DUP, OP_HASH160, address, OP_EQUALVERIFY, OP_CHECKSIG]
    ###############################################################################
```

（2）P2PKH_scriptSig(txin, txout, txin_scriptPubKey)函数主要用于生成一个有效脚本用来解锁输出并发送回 faucet，其参数含义如下：

- txin：表示输入的交易数据；
- txout：表示输出的交易数据；
- txin_scriptPubKey：表示输入交易的脚本公钥。

P2PKH 交易是比特币中最常见的交易类型之一，它使用公钥哈希作为地址，并且需要提供与之对应的私钥进行签名验证。该函数需要完成以下几个任务：

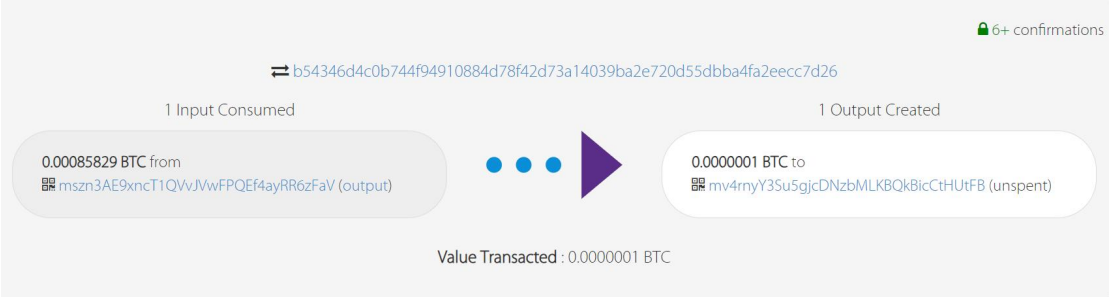
- 验证 txin 和 txout 的有效性，确保输入和输出的交易数据是有效的；
- 解析 txin_scriptPubKey，提取出公钥哈希；
- 使用私钥对 txin 进行签名，生成一个脚本签名；
- 将脚本签名和公钥作为输入的脚本签名（scriptSig）返回。

```
def P2PKH_scriptSig(txin, txout, txin_scriptPubKey):
    signature = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey,
                                             my_private_key)
    ################################################################################
    # TODO: Complete this script to unlock the BTC that was sent to you
    # in the PayToPublicKeyHash transaction. You may need to use variables
    # that are globally defined.
    return [signature, my_public_key]
    ################################################################################
```

（3）main 函数，设置 utxo_index=0

```
if __name__ == '__main__':
    ################################################################################
    # TODO: set these parameters correctly
    amount_to_send = 0.0000001
    txid_to_spend = (
        'c26d91f819ba03038c8936c56830a3996448cfd7dad7a67243ea8e5b95b799eb')
    utxo_index = 0
    ################################################################################

    txout_scriptPubKey = P2PKH_scriptPubKey(faucet_address)
    response = send_from_P2PKH_transaction(
        amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey)
    print(response.status_code, response.reason)
    print(response.text)
```

2. faucet 截图



Block Hash	0000000000002f257c11a30c3e2dc82f0fda0a48b12d17059f6437e6f8d13a90
Block Height	2,477,611
Transaction Index	1 (permalink)
Size	192 bytes
Virtual Size	192 vbytes
Lock Time	
Version	1
Relayed By:	117.131.219.7

3. 交易输出存在了“发币交易信息.txt”文件中。