南开大学

Nankai University

南　开　大　学

网 络 空 间 安 全 学 院

区块链基础及应用实验报告

第一次实验作业

罗功成 1910487

年级：2019 级

专业：信息安全

指导教师：苏明

2021 年 10 月 17 日

摘要

创建多个交易，并将其发布到比特币测试网

关键字：比特币的获取、拆分，python **脚本编辑**

目录

一、 实验流程

(一) 实验内容和目的

要发布每个练习的事务，请编辑 python 文件参数，以指定事务输出以及要在事务中发送的金额。注意所创建事务的事务哈希并将其保存至 transactions.py.

如果你编写的脚本无效，将在发布之前引发异常。

完成每个练习后，在区块链浏览器中查找交易哈希，以验证交易是否被网络获取。在提交哈希之前，请确保所有事务都已成功确认。

要求：

1. 打开 ex1.py 并完成标有 TODOs 的脚本，以赎回您拥有的输出
2. 通过标准 PayToPublicKeyHash 事务将其发送回 faucet

环境配置：

1. python3 版本下运行 pip install-r requirements.txt 安装所需的依赖项
2. 用 keygen.py 生成一个 testnet 私钥和地址，并在 config.py 适当位置复制并粘贴私钥
3. 编辑 split_test_coins.py.py，其中 txid 是上一步中事务的事务哈希值，（如果你的输出是 faucet 事务中的第一个输出值，则 utxo id 为 0 如果是第二个输出值，则为 1）。split_test_coins.py 中的 n 是期望将测试硬币平均分割成的输出数。

(二) 实验操作和相关结果

1. 运行 kengen.py, 得到相应的私钥和地址

2. 在 faucet 的 testnetBTC 添加上述得到的地址，收到对应的比特币。

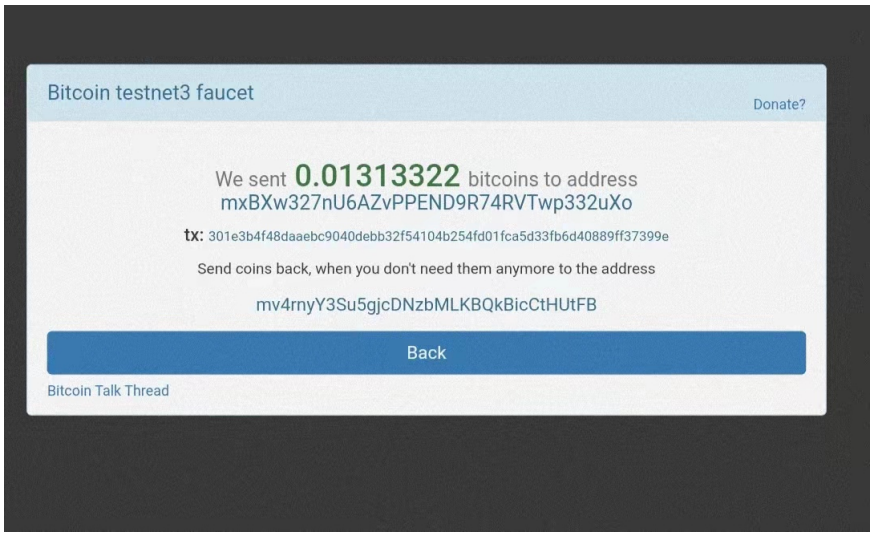

图 1: 收到测试用的比特币

3. 填写好 split_test_coins.py 中对应的 todo 部分，之后可以在网页中看到已经成功分成了 10 份。

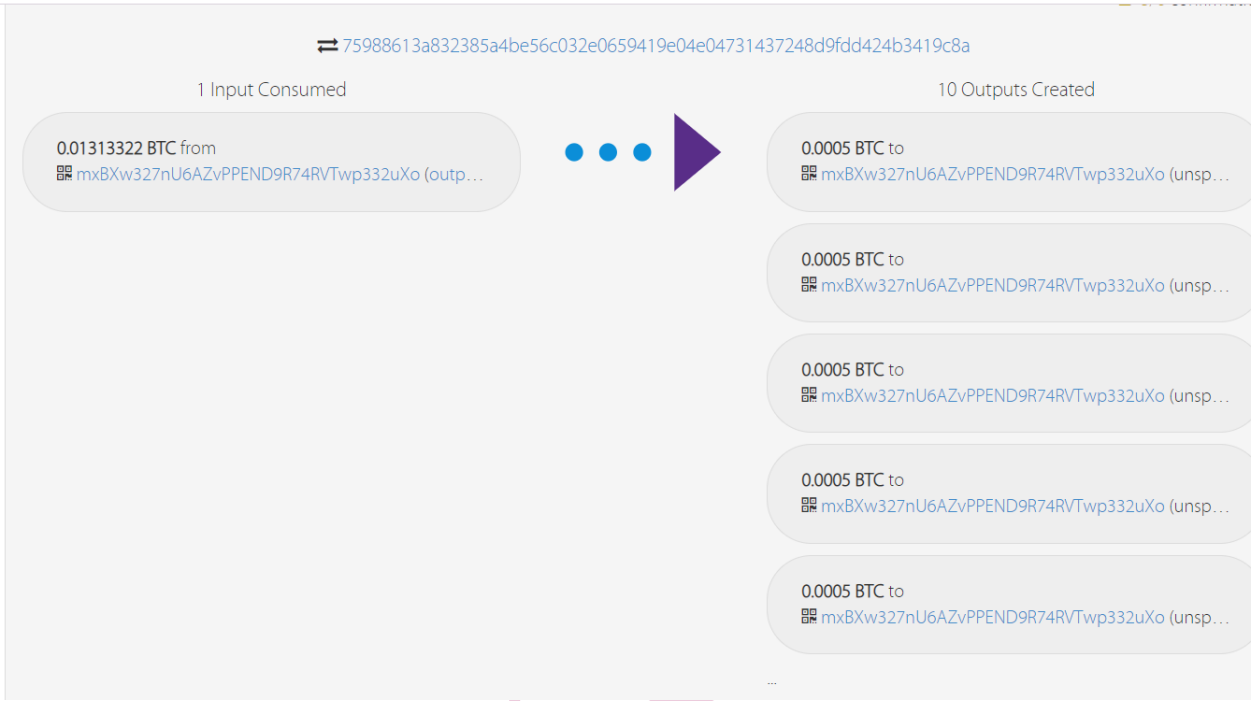

图 2: 分成 10 份成功的截图

图 3: 部分交易后的输出内容

4. 注意，其他脚本的 to do 部分也需要填写好对应的私钥，地址等信息。

在补全 ex1.py 的相关信息后，可以看到 python shell 中输出的相关文本信息。

(三) 实验核心代码

ex1.py 交易回收 bitcoin

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

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


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


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]
    ##############################################################################


def send_from_P2PKH_transaction(amount_to_send, txid_to_spend, utxo_index,
                                txout_scriptPubKey):
    txout = create_txout(amount_to_send, txout_scriptPubKey)

    txin_scriptPubKey = P2PKH_scriptPubKey(my_address)
    txin = create_txin(txid_to_spend, utxo_index)
    txin_scriptSig = P2PKH_scriptSig(txin, txout, txin_scriptPubKey)

    new_tx = create_signed_transaction(txin, txout, txin_scriptPubKey,
                                       txin_scriptSig)

    return broadcast_transaction(new_tx)


if __name__ == '__main__':
    ##############################################################################
    # TODO: set these parameters correctly
    amount_to_send = 0.0005
```

```
    txid_to_spend = (
        '75988613a832385a4be56c032e0659419e04e04731437248d9fdd424b3419c8a')
    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)
```

将收入的比特币分成 10 份 spilt.py

```
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 = 0.005# amount of BTC in the output you're splitting
        minus fee
    txid_to_spend = (
        '301e3b4f48daaebc9040debb32f54104b254fd01fca5d33fb6d40889ff37399e')
    utxo_index = 1
    n=10 # number of outputs to split the input into
    ################################################################################

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

config.py

```
  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(
    'cQhtgnRw36AD3bkh1TVorPNzBxo9FWDu3kPUuaCeTMFo4uaMRVL5')
my_public_key = my_private_key.pub
my_address = P2PKHBitcoinAddress.from_pubkey(my_public_key)

faucet_address = CBitcoinAddress('mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB')
```

kengen.py

```
from os import urandom
from bitcoin import SelectParams
from bitcoin.wallet import CBitcoinSecret, P2PKHBitcoinAddress

SelectParams('testnet')

seckey = CBitcoinSecret.from_secret_bytes(urandom(32))

print("Private key: %s" % seckey)
print("Address: %s" %
      P2PKHBitcoinAddress.from_pubkey(seckey.pub))
```

(四) 实验结果

回收 bitcoin 后输出的相关文本:

交易成功输出文本

```
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "8b97e66b4aa46503ae0a5a945ee29e2b4de47b03eec85d916f600b206a9c8dc8
        ",
    "addresses": [
      "mxBXw327nU6AZvPPEND9R74RVTwp332uXo",
      "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
```

```
    ],
    "total": 50000,
    "fees": 0,
    "size": 192,
    "vsize": 192,
    "preference": "low",
    "relayed_by": "2001:250:401:6601:b9b5:cb17:1d39:411d",
    "received": "2021-10-16T13:22:20.877715273Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "75988613
            a832385a4be56c032e0659419e04e04731437248d9fdd424b3419c8a",
        "output_index": 0,
        "script": "4830450221008b8936f50b68222cb7299908326e889
        bb4f325d6d258f6a4e5ab97a5a988
        e161022036fac9a4596fe83db8fcf4a2342921add2894f
        63dcc7b1fec920d0b69d1bb9e6012103ff9a6
        4b008ee38f544499f1d5f452ca09dd0bc2430b010dcaa46b2ea5010a6db",
        "output_value": 50000,
        "sequence": 4294967295,
        "addresses": [
          "mxBXw327nU6AZvPPEND9R74RVTwp332uXo"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 2099053
      }
    ],
    "outputs": [
      {
        "value": 50000,
        "script": "76a9149f9a7abd600c0caa03983a77c8c3df8e062cb2fa88ac",
        "addresses": [
          "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}
>>>
```

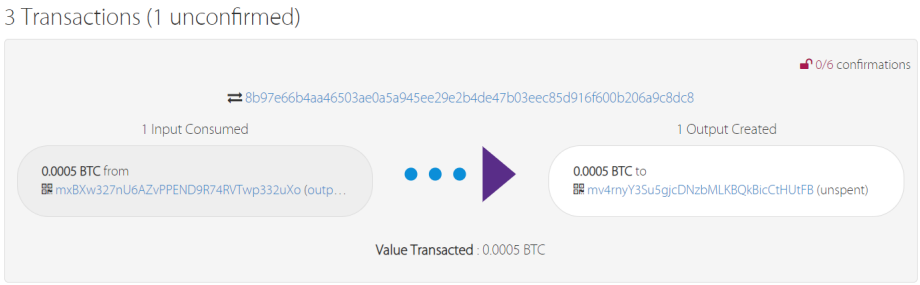

图 4: 回收成功的截屏

二、 遇到的问题与解决

1.scriptPubKey 脚本：

scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

遇到问题：编写时对脚本理解不对，直接把 pubKeyHash 填进去导致报错。

这里的内容是直接填到 return[] 当中的，操作码直接不变，带尖括号的是需要自己填入对应的变量名的。例如，在这里前面声明了一个 address 的变量，那就要把 address 替换到 <pubKeyHash> 这里。

2.scriptSig 脚本

scriptSig: <sig> <pubKey>

```
Traceback (most recent call last):
  File "D:\LGC信息与基本资料\相关学术工具\比特币\Ex1\ex1.py", line 53, in <module>
    amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey)
  File "D:\LGC信息与基本资料\相关学术工具\比特币\Ex1\ex1.py", line 37, in send_from_P2PKH_transaction
    txin_scriptSig)
  File "D:\LGC信息与基本资料\相关学术工具\比特币\Ex1\utils.py", line 37, in create_signed_transaction
    txin.scriptSig = CScript(txin_scriptSig)
  File "D:\python\lib\site-packages\bitcoin\core\script.py", line 563, in __new__
    return super(CScript, cls).__new__(cls, b''.join(coerce_iterable(value)))
TypeError: sequence item 2: expected a bytes-like object, CMutableTxIn found
>>>
```

图 5: 对应变量的理解

同理，这里也是只需要找到文件中和 sig 签名和 pubkey 公钥对应的变量名，将其填入 return[] 即可。

找对应变量时要注意看当前函数生成的是否都已经返回了。那可以发现，这里 signature 声明后没有进行任何操作，因此它就作为这里 sig 出现。而另外一个公钥是通过前面私钥生成的，签名也是由私钥生成的，那么签名对应的公钥应该是私钥对应的公钥。并且生成公钥的过程是在 config 文件中发生的，因此公钥对应的变量一定是一个全局变量，也就是第 4 行的 my_public_key

3. 回收过程：

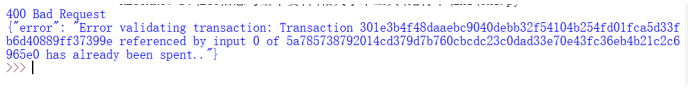

图 6: 回收交易不成功

（1）每次回收的金额不是分发出去的总数（0.005），而是每一次拆分好的数目（0.0005），以后的每笔交易也是只把拆分后的一份进行消费。

（2）回收，不是回收到自己的账户，而是回收到发给你的地址，因此不要把所有的都操作一遍，否则前面的分币操作将毫无意义。

（3）回收的 txid 应该换成分币后每份的那个，75988613a832385a4be56c032e0659419e04e04731437248d9fdd424b3419c8a

（4）下面的 index 代表第几笔交易，10 笔交易，下标则为 0-9。

三、 总结和收获

1. 对比特币的获取，分发和回收等工作原理有了深入的了解，并完成 to do 后，成功尝试完成了相应操作。
2. 对于 python 脚本的调用和操作码级别的编程知识进行了学习和应用。
3. 在此特别致谢袁也老师和王浩宾同学在我完成本次作业过程中提供的指导和帮助！