南开大学

Nankai University

南　　开　　大　　学

计　算　机　学　院

区块链实验报告

Ex3

徐俊智

年级：2022 级

专业：计算机科学与技术

指导教师：苏明

2024 年 10 月 22 日

目录

一、 实验要求

（a）生成可通过以下两个线性方程组的解（x，y）赎回的交易：x+y=（StudentID 前 4 位）和 x-y=（StudentID 后 3 位）[为确保存在整数解，请必要时调整（顺序减 1）你的 StudentID 后 3 位，使 StudentID 前 4 位和 StudentID 后 3 位奇偶性相同]。

（b）赎回事务并确保 scriptPubKey 尽可能小。可以使用任何合法的签名组合来赎回交易至 faucet 地址，但要确保所有组合都有效。

二、 练习（a）

StudentID 前 4 位：2213

StudentID 后 3 位：410（调整为 409）

$$\begin{cases} x + y = 2213 \\ x - y = 409 \end{cases} \qquad \begin{cases} x = 1311 \\ x = 902 \end{cases}$$

ex3a.py 脚本的目的是从我的一个 UTXO 中拿出 0.00000050 BTC，创建一个基于线性方程组的锁定脚本，要求在解锁交易时验证（x，y）。

基于线性方程组的锁定脚本：

```
ex3a_txout_scriptPubKey = [
    OP_2DUP,
    OP_ADD,
    2213,
    OP_EQUALVERIFY,
    OP_SUB,
    409,
    OP_EQUAL
]
```

- 复制栈顶的两个元素 (x, y)，结果是 (x, y, x, y)
- 将栈顶的两个元素相加，结果是 (x, y, x + y)
- 将 2213 压入栈，结果是 (x, y, x + y, 2213)
- 弹出栈顶的两个元素，验证 (x + y == 2213)，如果相等，继续执行；否则，脚本失败
- 计算 (x - y)，结果是 (x - y)
- 将 409 压入栈，结果是 (x - y, 409)
- 验证 (x - y == 409)，如果相等则交易成功

设置交易参数。lab1 分割了 10 个 UTXO，并且用掉了第 1 个 UTXO。本次实验使用第 4 个 UTXO。设置发送金额为 0.00000050 BTC，那么那么手续费为 0.00000050 BTC 。

```
    amount_to_send = 0.00000050
    txid_to_spend = (
'387f1017c5f92552d1104c57a923bd932db5f03729aba5609394c2296ed5c6d0')
    utxo_index = 3
```

调用 send_from_P2PKH_transaction 函数进行比特币交易。首先创建一个交易输出，指定发送的比特币数量和锁定脚本。然后创建一个交易输入，引用 UTXO，并生成交易输入的解锁脚本。最后通过 create_signed_transaction 函数将输入和输出组合成一个完整的交易，并通过网络广播，使其能够被确认并记录在区块链上。

在 create_signed_transaction 函数中，通过 VerifyScript 函数进行验证。

当有人试图花费前一笔交易的 UTXO 时，验证脚本涉及当前交易输入的解锁脚本和前一笔交易输出的锁定脚本的结合。

```
# 当前交易输入的解锁脚本
signature,
my_public_key,
# 前一笔交易输出的锁定脚本
OP_DUP,
OP_HASH160,
address_hash,
OP_EQUALVERIFY,
OP_CHECKSIG
```

我们分析一下验证脚本的执行堆栈状态图：

（1）当前交易输入的解锁脚本 (txin_scriptSig) 执行：

1. signature
   - signature 入栈
   - 堆栈状态:[signature]
2. my_public_key
   - my_public_key 入栈
   - 堆栈状态:[signature, my_public_key]

（2）前一笔交易输出的锁定脚本 (txin_scriptPubKey) 执行：

1. OP_DUP
   - 复制栈顶元素
   - 堆栈状态:[signature, my_public_key, my_public_key]
2. OP_HASH160
   - 计算栈顶元素的 SHA-256 哈希然后 RIPEMD-160 哈希
   - 堆栈状态:[signature, my_public_key, my_public_key_hash]
3. address_hash
   - 我的地址的哈希值 address_hash 入栈
   - 堆栈状态:[signature, my_public_key, my_public_key_hash, address_hash]
4. OP_EQUALVERIFY
   - 弹出栈顶的两个元素，比较它们是否相等。如果相等，继续执行；否则，脚本失败。

- 堆栈状态:[signature, my_public_key]

5. OP_CHECKSIG

- 弹出栈顶的两个元素（签名和公钥），检查是否匹配。如果匹配，将 True 压入栈顶；否则，将 False 压入栈顶。
- 堆栈状态:[TRUE]

运行

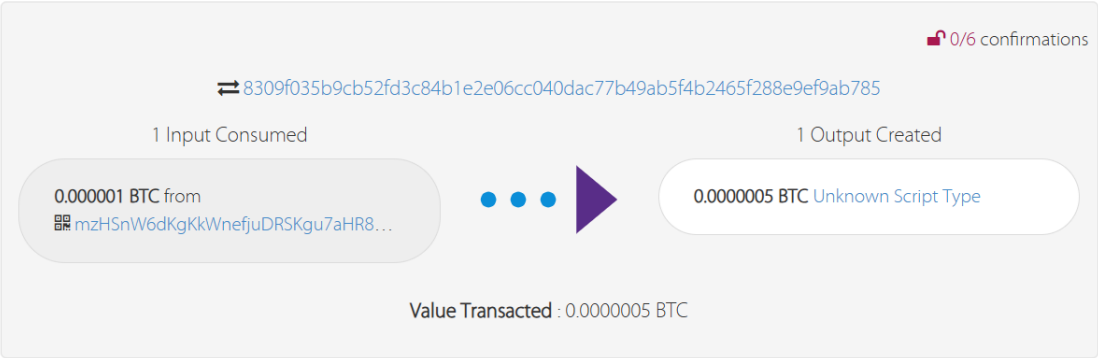
```
python3 ex3a.py
```

命令行输出交易信息

```
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "8309f035b9cb52fd3c84b1e2e06cc040dac77b49ab5f4b2465f288e9ef9ab785
        ",
    "addscriptSig     "mzHSnW6dKgKkWnefjuDRSKgu7aHR8UecyZ"
    ],
    "total": 50,
    "fees": 50,
    ......
```

详情见《ex3a 交易输出结果.txt》

查看 https://live.blockcypher.com/网站的交易记录



三、 练习（b）

ex3b.py 脚本的目的是从上一个基于线性方程组的锁定脚本中赎回 UTXO ，要求在解锁交易时验证 faucet_address 的签名。

解锁脚本:

```
txin_scriptSig = [1311, 902]
```

设置交易参数，这里的 txid_to_spend 要用上一步新生成的。

```
    amount_to_send = 0.00000050
    txid_to_spend = (
'8309f035b9cb52fd3c84b1e2e06cc040dac77b49ab5f4b2465f288e9ef9ab785')
    utxo_index = 0
```

调用 `send_from_P2PKH_transaction` 函数进行比特币交易。首先创建一个交易输出，指定发送的比特币数量和锁定脚本。然后创建一个交易输入，引用 UTXO，并生成交易输入的解锁脚本。最后通过 `create_signed_transaction` 函数将输入和输出组合成一个完整的交易，并通过网络广播，使其能够被确认并记录在区块链上。

在 `create_signed_transaction` 函数中，通过 `VerifyScript` 函数进行验证。

当有人试图花费前一笔交易的 UTXO 时，验证脚本涉及当前交易输入的解锁脚本和前一笔交易输出的锁定脚本的结合。

```
# 当前交易输入的解锁脚本
1311,
902,
# 前一笔交易输出的锁定脚本
OP_2DUP,
OP_ADD,
2213,
OP_EQUALVERIFY,
OP_SUB,
409,
OP_EQUAL
```

我们分析一下验证脚本的执行堆栈状态图：

（1）当前交易输入的解锁脚本 (`txin_scriptSig`) 执行：

1. 1311
   - 1311 入栈。
   - 堆栈状态:[1311]
2. 902
   - 902 入栈。
   - 堆栈状态:[1311, 902]

（2）前一笔交易输出的锁定脚本 (`ex3a_txout_scriptPubKey`) 执行：

1. `OP_2DUP`
   - 复制栈顶的两个元素。
   - 堆栈状态:[1311, 902, 1311, 902]
2. `OP_ADD`
   - 将栈顶的两个元素相加。
   - 堆栈状态:[1311, 902, 2213]

3. 2213
   - 2213 入栈。
   - 堆栈状态:[1311, 902, 2213, 2213]
4. `OP_EQUALVERIFY`
   - 弹出栈顶的两个元素，比较它们是否相等。如果相等，继续执行；否则，脚本失败。
   - 堆栈状态:[1311, 902]
5. `OP_SUB`
   - 计算栈顶的第二个元素减去栈顶的第一个元素。
   - 堆栈状态:[409]
6. 409
   - 409 入栈。
   - 堆栈状态:[409, 409]
7. `OP_EQUAL`
   - 弹出栈顶的两个元素，比较它们是否相等。如果相等，将 True 压入栈顶；否则，将 False 压入栈顶。
   - 堆栈状态:[TRUE]

运行

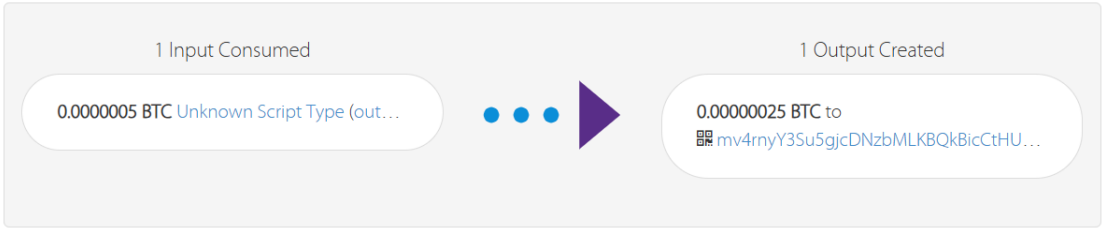
```
python3 ex3b.py
```

输出交易信息

```
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "797e403ab02438eb2611e862f2b1550c9a448e2b3db6ac2b544542efb80ef734
        ",
    "addresses": [
      "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
    ],
    "total": 25,
    "fees": 25,
    ......
```

详情见《ex3b 交易输出结果.txt》

查看 https://live.blockcypher.com/网站的交易记录


1 Input Consumed
0.0000005 BTC Unknown Script Type (out…
1 Output Created
0.00000025 BTC to
mv4rnyY3Su5gjcDNzbMLKBQkBicCtHU…