南開大學
Nankai University

南开大学

计算机学院和网络空间安全学院

《区块链基础及应用》实验报告

Ex3: 求解线性方程组谜题

姓名：陆皓喆

学号：2211044

专业：信息安全

指导教师：苏明

2024 年 10 月 22 日

目录

1 实验要求 **2**

2 github 实验仓库 **2**

3 实验过程 **2**

3.1 生成交易 (修改 Ex3a.py) 2

3.2 赎回事务 (修改 Ex3b.py) 5

4 附录 1:Ex3a_output **7**

5 附录 2:Ex3b_output **9**

1 实验要求

- 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]。
- 赎回交易。赎回脚本应尽可能小。也就是说，一个有效的 scriptSig 应该是简单地将两个整数 x 和 y 发送到堆栈中。确保在 scriptPubKey 中使用了 OP_ADD 和 OP_SUB。

2 github 实验仓库

本学期 BlockChain 课程的所有实验代码以及报告，均存放到本人的 github 中。您可以通过**此链接**来查看我的实验项目文件。

3 实验过程

3.1 生成交易 (修改 Ex3a.py)

本次实验我们将在原先的 bitcoin 账户上进行实验，对应的账户信息为：

```
Private key: cW3fxHAhp5RRfDcFkbctRPymyYJZe3AEF7UngN8qfnKhkPnccKNj
Address: mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM
```

我们首先完成生成交易的任务，也就是修改 Ex3a.py 的代码。首先，我们补充 ex3a_txout_scriptPubKey 的内容。该部分我们还是和上一个实验相似的思路，使用堆栈式编程来完成对应的检查工作。代码如下所示：

```
ex3a_txout_scriptPubKey = [
    OP_2DUP,
    OP_ADD,
    2211,
    OP_EQUALVERIFY,
    OP_SUB,
    43,
    OP_EQUAL
]
```

解释一下该段代码：

- **OP_2DUP:** 功能为复制当前堆栈顶部的两个值；
- **OP_ADD:** 首先，弹出当前堆栈的顶部的两个值，进行相加操作，然后将相加所得的结果压入堆栈中；
- **2211:** 这个数字为我的学号 (2211044) 的前四位；

- **OP_EQUALVERIFY:** 检查堆栈的顶部两个值是否相等。如果它们相等，则会被弹出；如果不相等，则脚本执行失败。
- **OP_SUB:** 首先，弹出当前堆栈的顶部的两个值，进行相减操作，然后将相减所得的结果压入堆栈中；
- **43:** 我的学号 (2211044) 的后三位为 044，去掉首部的 0 后为 44，为了保持和 2211 奇偶性相同，减去 1，变为 43；
- **OP_EQUAL:** 检查堆栈的顶部两个值是否相等。如果它们相等，堆栈顶部为 TRUE；否则，为 FALSE。

我们模拟一下运行该程序时，堆栈的变化情况：

1. 首先，初始堆栈里的内容为 x 和 y；

```
y
x
```

2. OP_2DUP 表示复制堆栈中的内容，即变成 x,y,x,y；

```
y
x
y
x
```

3. OP_ADD 弹出堆栈的顶部两个值并将它们相加，然后将结果推到堆栈的顶部；

```
x+y
y
x
```

4. 推送数字 2211 到堆栈顶部；

```
2211
x+y
y
x
```

5. 检查堆栈中顶部的两个元素值是否相同，即检查 2211 和 x+y 是否相同。如果相同，即被弹出栈；此处值确实相同，所以 2211 和 x+y 均被弹出栈；

```
y
x
```

6. 弹出堆栈的顶部两个值并做减法运算，然后将结果推到堆栈的顶部；

```
x-y
```

7. 推送数字 43 到堆栈顶部;

```
43
x-y
```

8. 检查堆栈的顶部两个值是否相等，也就是判断 x-y 和 43 的大小关系；如果它们相等，堆栈顶部为 TRUE；否则，为 FALSE；此处如果通过检查，该函数就会输出 TRUE。

我们接着完成 main 函数的填写。代码如下所示:

```
    amount_to_send = 0.00001
    txid_to_spend = (
        'b95bbc0006084b3edb5d87f55bd05b3696c5d34ca7f7ba74426219d5ebe24615')
    utxo_index = 1
```

首先，我们分币后的每一个交易都只含有 0.00001 的 bitcoin，所以我们选择花费的 bitcoin 为 0.00001；然后我们设置的 txid 为分币交易的 txid 值，索引值为 1(上一次实验中，已经使用掉了索引值为 0 的交易)。

这样我们的生成交易的代码部分就填写完毕了。运行 Ex3a.py 程序，得到对应的输出结果。**由于篇幅有限，我们将其存放在附录 1:Ex3a_output 中**，仅对其部分内容做一些解释与说明。

```
    "hash": "f0b8f633740c034a7efe861804546c98cccbc1d96f758ce15e392ac4133ac580",
    "addresses": [
      "mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM"
    ],
    "total": 1000,
    "fees": 0,
```

上面这段输出的内容，我做一些说明:

- 输出的 hash 表示，我们该笔交易产生了一个新的 hash 值，**我们后续的赎回操作应该在此 hash 基础上进行完成**;
- address 代表了我的地址，因为此处我们只是生成交易，所以并没有第二个地址;
- total 为 1000 指的是我们**本次传输的 bitcoin 的数量**，实际上为 0.00001;
- fee 代表了我们本次**交易的小费**，由于只是进行交易的生成，所以我并没有给一定的小费。

进一步前往 bitcoin 网站上查看，我们发现确实生成了一笔交易，如图3.1所示:

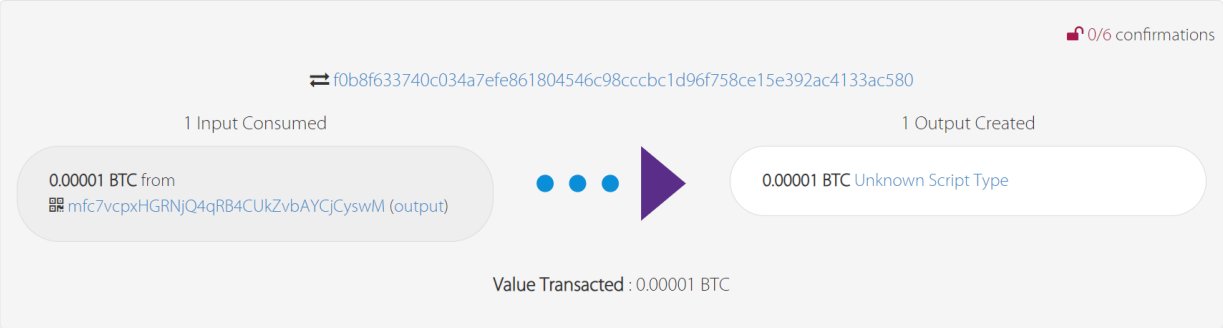

图 3.1: 生成 bitcoin

我们发现，该笔交易从我们个人的账户 (mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM) 发到了另外一个未知的地址类型上。对应的交易 hash 为 f0b8f633740c034a7efe861804546c98cccbc1d96f758ce15e392ac4133ac580。

tips：由于我没有给小费，导致这笔交易等了一个小时也没有人给我确认，我哭死。

3.2 赎回事务 (修改 Ex3b.py)

该文件需要我们完成对上一部分的生成的交易进行赎回的操作。

我们首先完成对线性方程组的求解，实际上就是，已知

$$\begin{cases} x + y = 2211 \\ x - y = 43 \end{cases}$$

我们手动解一下，得到如下结果：

$$\begin{cases} x = 1127 \\ y = 1084 \end{cases}$$

然后我们来完成对应脚本的填写。

首先是对于解决谜题部分的函数的填写，此处较为简单，只需要我们将 x 和 y 的值填上去即可。

```
txin_scriptSig = [1127,1084]
```

我们分析一下执行 send_from_custom_transaction 函数后，函数的输出情况。

1. 首先，初始堆栈里的内容为 x 和 y，值分别为 1127 和 1084；

```
1084
1127
```

2. OP_2DUP 表示复制堆栈中的内容，即变成 1127,1084,1127,1084；

```
1084
1127
1084
1127
```

3. OP_ADD 弹出堆栈的顶部两个值并将它们相加，然后将结果推到堆栈的顶部，也就是 2211；

```
2211
1084
1127
```

4. 推送数字 2211 到堆栈顶部；

```
2211
2211
1084
1127
```

5. 检查堆栈中顶部的两个元素值是否相同，我们发现 2211 和 2211 相同，所以弹出；

```
1084
1127
```

6. 弹出堆栈的顶部两个值并做减法运算，然后将结果推到堆栈的顶部；

```
43
```

7. 推送数字 43 到堆栈顶部；

```
43
43
```

8. 检查堆栈的顶部两个值是否相等，我们发现 43 和 43 相等，**所以堆栈顶部为 TRUE**；

这样我们就能够完成对 bitcoin 的赎回工作。
然后我们完成交易信息的填写，如下所示。

```
amount_to_send = 0.000005
txid_to_spend = 'f0b8f633740c034a7efe861804546c98cccbc1d96f758ce15e392ac4133ac580'
utxo_index = 0
```

此处的 txid 为我们上一步中交易的 hash 值，索引值为 0(因为没用过)，支付的费用为 0.000005，剩下的 bitcoin 当作小费。

这样我们的赎回交易的代码部分就填写完毕了。运行 Ex3b.py 程序，得到对应的输出结果。**由于篇幅有限，我们将其存放在附录 2:Ex3b_output 中**，仅对其部分内容做一些解释与说明。

```
    "hash": "7753049921c0bbac9e44c4edd6c57749ff554c35188b7ea93dd0e4ed4a613ce1",
    "addresses": [
      "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
    ],
    "total": 500,
    "fees": 500,
```

上面这段输出的内容，我做一些说明：

- 输出的 hash 表示，我们该笔交易产生了一个新的 hash 值，**表示赎回操作的 hash 值**；
- address 代表了苏明老师的地址，我们将 bitcoin 赎回到了苏明老师的账户上；
- total 为 500 指的是我们本次传输的 **bitcoin 的数量**，实际上为 0.000005；
- fee 代表了我们本次**交易的小费**，为 500，也就是 0.000005。

这样，我们就完成了上一笔交易的赎回工作，将 bitcoin 打到了苏明老师的账户上。

进一步前往 bitcoin 网站上查看，**我们发现确实有一笔交易打到了苏明老师的账户上**，如图3.2所示：

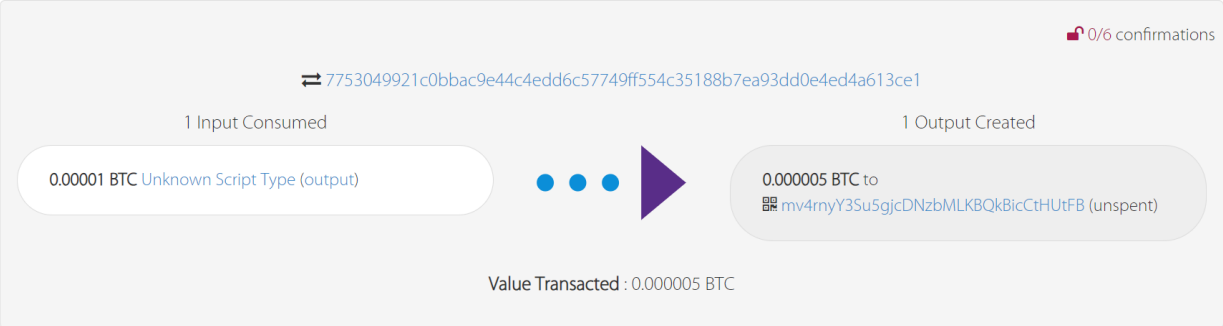

图 3.2: 赎回 bitcoin

我们发现，该笔交易从上一问中的未知地址账户发到了苏明老师的账户上。对应的交易 hash 为 7753049921c0bbac9e44c4edd6c57749ff554c35188b7ea93dd0e4ed4a613ce1。

4 附录 1:Ex3a_output

```
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex3/codes$ /bin/python3
↪ /home/luhaozhhhe/BlockChain2024/Ex3/codes/ex3a.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "f0b8f633740c034a7efe861804546c98cccbc1d96f758ce15e392ac4133ac580",
    "addresses": [
      "mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM"
    ],
    "total": 1000,
    "fees": 0,
    "size": 177,
    "vsize": 177,
    "preference": "low",
```

```
    "relayed_by": "60.29.153.32",
    "received": "2024-10-22T07:58:56.923181574Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash":
        ↪ "b95bbc0006084b3edb5d87f55bd05b3696c5d34ca7f7ba74426219d5ebe24615",
        "output_index": 1,
        "script": "483045022100e160e98d99f5aab8c59d3d1d9104506a5a3bac4f2120cf06ccd」
        ↪ 73d40435395970220 78f4bc2bb22db069c4bebbb2380883fa195b92ddb5ac7de339816」
        ↪ cfe7d49ce9a012103019c64252a509d87deb3ac2592c017b5237d7b49b4b96d75845a9」
        ↪ a0253740fd2",
        "output_value": 1000,
        "sequence": 4294967295,
        "addresses": [
          "mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 3009503
      }
    ],
    "outputs": [
      {
        "value": 1000,
        "script": "6e9302a3088894012b87",
        "addresses": null,
        "script_type": "unknown"
      }
    ]
  }
}
```

5 附录 2:Ex3b_output

```
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex3/codes$ /bin/python3 /home/luhaozhhhe/BlockChain2024/Ex3/codes/ex3b.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "7753049921c0bbac9e44c4edd6c57749ff554c35188b7ea93dd0e4ed4a613ce1",
    "addresses": [
      "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
    ],
    "total": 500,
    "fees": 500,
    "size": 91,
    "vsize": 91,
    "preference": "low",
    "relayed_by": "60.29.153.32",
    "received": "2024-10-22T08:03:37.422689423Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "f0b8f633740c034a7efe861804546c98cccbc1d96f758ce15e392ac4133ac580",
        "output_index": 0,
        "script": "026704023c04",
        "output_value": 1000,
        "sequence": 4294967295,
        "script_type": "unknown",
        "age": 0
      }
    ],
    "outputs": [
      {
        "value": 500,
        "script": "76a9149f9a7abd600c0caa03983a77c8c3df8e062cb2fa88ac",
        "addresses": [
```

```
          "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}
```