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

Ex2: 涉及四方的多签名交易

姓名：陆皓喆

学号：2211044

专业：信息安全

指导教师：苏明

2024 年 10 月 8 日

目录

1 实验要求

- 生成一个涉及四方的多签名交易，这样交易可以由第一方（银行）与另外三方（客户）中的任何一方（客户）共同赎回，而不仅仅只是客户或银行。对于这个问题，你可以假设是银行的角色，这样银行的私钥就是你的私钥，而银行的公钥就是你的公钥。使用 keygen.py 生成客户密钥并将它们粘贴到 ex2a.py 中。
- 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。

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 对 Ex2a.py 的修改

首先我们打开 keygen.py，**生成三对不同的私钥和地址作为另外三方客户**，如图3.1所示：

```
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex2$ /bin/python3 /home/luhaozhhhe/BlockChain2024/Ex2/keygen.py
Private key: cVDkvR5cVoGWiTtJfiVuTg3zCJWyzUTuoHWdtyFzb2UmCznrP5hu
Address: mmREm2BsBEqSPMGWCBi6n2WaPPJujzPVS8
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex2$ /bin/python3 /home/luhaozhhhe/BlockChain2024/Ex2/keygen.py
Private key: cUTjPkbehgpnRnDHj7saxW1htRWBwB4FnGPw8F1wDF3ARLcPYiEX
Address: mnrrsXSDqERw3Vrce7roVAxqBMz7WVtnJy
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex2$ /bin/python3 /home/luhaozhhhe/BlockChain2024/Ex2/keygen.py
Private key: cSCa4Ddws1DkXfKcN8KFx5n44dQcRZoU318Nk23x9CDjxssLUcUT
Address: mjejyoK9YGNLXyGjtMNBZvvHoUFdEZCtNW
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex2$
```

图 3.1: 生成三个私钥和地址

我们得到了另外三组数据备用，数据具体的值为：

```
Private key: cVDkvR5cVoGWiTtJfiVuTg3zCJWyzUTuoHWdtyFzb2UmCznrP5hu
Address: mmREm2BsBEqSPMGWCBi6n2WaPPJujzPVS8

Private key: cUTjPkbehgpnRnDHj7saxW1htRWBwB4FnGPw8F1wDF3ARLcPYiEX
Address: mnrrsXSDqERw3Vrce7roVAxqBMz7WVtnJy

Private key: cSCa4Ddws1DkXfKcN8KFx5n44dQcRZoU318Nk23x9CDjxssLUcUT
Address: mjejyoK9YGNLXyGjtMNBZvvHoUFdEZCtNW
```

我们根据三个新账户的信息，完成代码段中私钥的填写，如下所示：

```
cust1_private_key = CBitcoinSecret(
    'cVDkvR5cVoGWiTtJfiVuTg3zCJWyzUTuoHWdtyFzb2UmCznrP5hu')
cust1_public_key = cust1_private_key.pub
cust2_private_key = CBitcoinSecret(
    'cUTjPkbehgpnRnDHj7saxW1htRWBwB4FnGPw8F1wDF3ARLcPYiEX')
cust2_public_key = cust2_private_key.pub
```

```
cust3_private_key = CBitcoinSecret(
    'cSCa4Ddws1DkXfKcN8KFx5n44dQcRZoU318Nk23x9CDjxssLUcUT')
cust3_public_key = cust3_private_key.pub
```

接着，我们创建多重签名脚本，完成多签名交易的要求。

首先，我们定义所需要的签名数量，此处为 3。然后我们需要创建对应三个用户的公钥列表，再用公钥列表来创建多签名脚本，来完成多签名交易。

```
required_signatures = 3 #定义所需要的签名数量为3
#定义public_key，内容分别为cust1、cust2和cust3的公钥
public_key=[
    cust1_public_key,
    cust2_public_key,
    cust3_public_key
]
#完成多签名交易
ex2a_txout_scriptPubKey = CScript([required_signatures] + public_key +
    [len(public_key), OP_CHECKMULTISIG])
```

我们详细解释一下最后一行代码的含义：

首先，写入 [required_signatures]，也就是签名的内容和长度；然后写入对应的三个公钥，然后写入公钥的长度，此处为 3；最后写入 OP_CHECKMULTISIG。栈空间在执行 OP_CHECKMULTISIG 前是这个样子的：

```
3
cust3_public_key
cust2_public_key
cust1_public_key
3
sig3
sig2
sig1
OP_0
```

然后执行 OP_CHECKMULTISIG。首先，弹出**公钥的个数 3**，然后分别出栈三**个公钥**，也就是 cust3_public_key、cust2_public_key 和 cust1_public_key；然后出栈**签名的个数**，也就是 3；接着出栈三**个对应的签名** sig3、sig2 和 sig1，最后将 **OP_0** 出栈（历史遗留问题）。然后读取完数据后，进行对应的检查工作，此处就是拿公钥和签名去做分别的是否匹配的判断，如果匹配成功，就返回 1。

这样，我们的多签名交易就成功完成了！

编写完对应的函数后，我们在最后填写自己的 bitcoin 的相关信息：

```
    amount_to_send = 0.00001 #定义花费的bitcoin
    #定义我们的交易tx的值，选取Ex1中分币生成的交易id
    txid_to_spend = (
```

```
    'b95bbc0006084b3edb5d87f55bd05b3696c5d34ca7f7ba74426219d5ebe24615')
#由于我们在Ex1中完成的分币操作还没有使用过，因此我们选择index为0
utxo_index = 0
```

编写完毕后，运行程序，得到 Ex2a 的输出结果。

由于篇幅所限，我们将其存放在附录 1:Ex2a_output 中，仅对其部分内容做一些解释与说明。

```
"hash": "b8c28455c3fa7498d49620dab820d41e1213432a1430ebca6be2418c475e1cb7",
"addresses": [
  "mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM",
  "zByATba1LvxeFfDKjScGMJXS362cPsaDZQ"
],
"total": 1000
```

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；
- addresses 和 Ex1 中有所不同，此处生成了两个地址，分别是**发币的地址**和**收到币的地址**；
- total 为 1000 指的是我们本次传输的 **bitcoin 的数量**，实际上为 0.00001。

进一步前往 bitcoin 网站上查看，**发现确实有了一笔从银行 (自身) 发往其他账户的交易**，如图3.2所示。

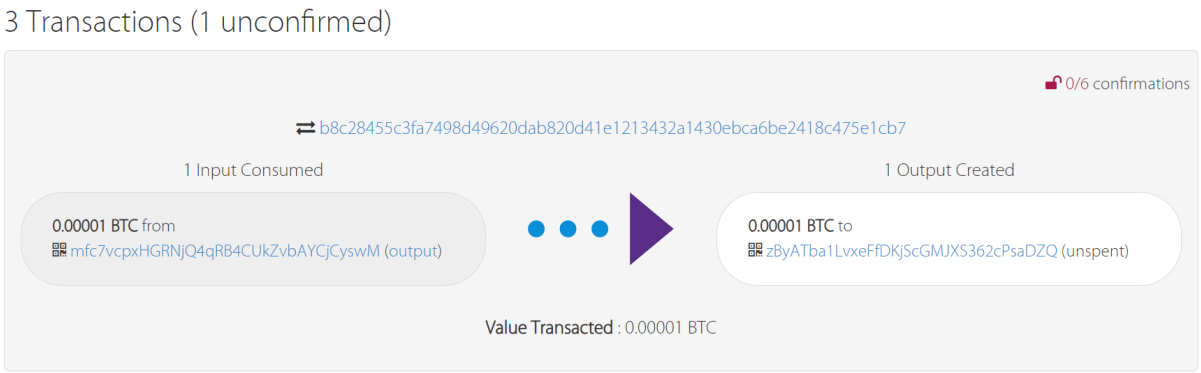

图 3.2: Ex2a.py 完成的任务

我们发现，该笔交易从我们个人的账户 (mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM) 发到了另外一个账户 (zByATba1LvxeFfDKjScGMJXS362cPsaDZQ) 上。

对应的交易 hash 为 b8c28455c3fa7498d49620dab820d41e1213432a1430ebca6be2418c475e1cb7。

4　对 Ex2b.py 的修改

该文件我们需要完成 bitcoin 的赎回工作。首先，我们需要完成多重签名脚本的解锁脚本函数 multisig_scriptSig 的补充。可以发现前面的代码都已经提供给我们了，我们只需要编写对应的返回值即可。

```
def multisig_scriptSig(txin, txout, txin_scriptPubKey):
    bank_sig = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey,
                                        my_private_key)
    cust1_sig = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey,
                                        cust1_private_key)
    cust2_sig = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey,
                                        cust2_private_key)
    cust3_sig = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey,
                                        cust3_private_key)

    return [OP_0, bank_sig, cust1_sig, cust2_sig, cust3_sig]
```

此处的 OP_0 的输出，是为了解决历史遗留问题。通过搜索资料可得，OP_CHECKMULTISIG 交易赎出币时需要如下的 scriptSig，在前文中也已经有所提及。

```
    OP_0 ...signatures...
```

OP_CHECKMULTISIG 中存在 Bug，它在实现的时候多出栈了一个元素，所以只好使用 OP_0 进行填充。

综上所述，我们使用银行的签名 bank_sig 以及三个客户的签名 cust1_sig、cust2_sig、cust3_sig 来完成了多签名解锁的功能。

然后，我们填写一下个人的 bitcoin 相关信息。

```
    amount_to_send = 0.00001
    txid_to_spend =
        'b8c28455c3fa7498d49620dab820d41e1213432a1430ebca6be2418c475e1cb7'
    utxo_index = 0
```

此处，amount_to_senf 我们选择 0.00001，与上一问相同。交易的 txid 我们需要换为上一个发币交易所对应的 hash 值，交易的索引我们也是选择 0。

编写完毕后，运行程序，得到 Ex2b 的输出结果。

由于篇幅所限，我们将其存放在附录 2:Ex2b_output 中，仅对其部分内容做一些解释与说明。

```
    "hash": "95ab400c1865885d72af9da44209e67eff6494ac9aaf1ac7d4bb3fcfc0c038de",
    "addresses": [
      "zByATba1LvxeFfDKjScGMJXS362cPsaDZQ",
      "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
    ],
    "total": 1000
```

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；
- addresses 分别表示收到 bitcoin 的客户的地址以及苏明老师的地址；

- total 为 1000 指的是我们本次传输的 **bitcoin 的数量**，实际上为 0.00001。

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，**发现确实有了一笔从客户账户发往苏明老师账户的交易**，如图4.3所示。

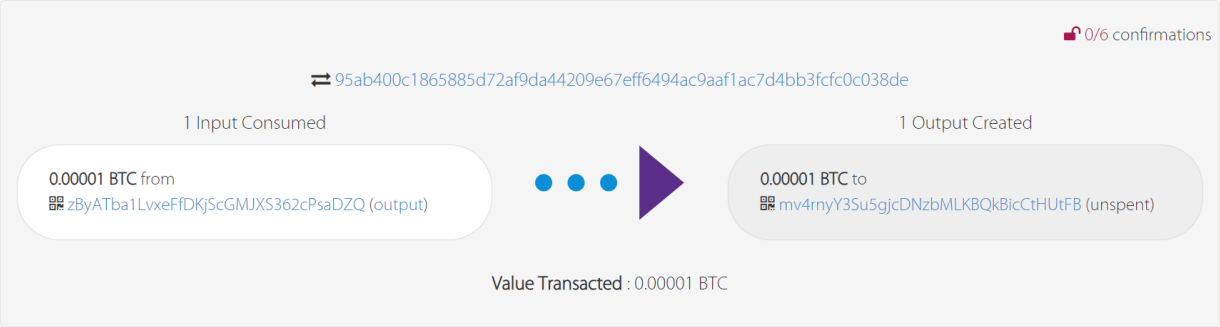

图 4.3: Ex2b.py 完成的任务

我们发现，该笔交易从我们客户的账户 (zByATba1LvxeFfDKjScGMJXS362cPsaDZQ) 发到了苏明老师的账户 (mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB) 上。

对应的交易 hash 为 95ab400c1865885d72af9da44209e67eff6494ac9aaf1ac7d4bb3fcfc0c038de。

5 附录 1:Ex2a_output

```
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex2$ /bin/python3
↪  /home/luhaozhhhe/BlockChain2024/Ex2/ex2a.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "b8c28455c3fa7498d49620dab820d41e1213432a1430ebca6be2418c475e1cb7",
    "addresses": [
      "mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM",
      "zByATba1LvxeFfDKjScGMJXS362cPsaDZQ"
    ],
    "total": 1000,
    "fees": 0,
    "size": 271,
    "vsize": 271,
    "preference": "low",
    "relayed_by": "111.33.78.4",
    "received": "2024-10-08T08:46:22.886784817Z",
    "ver": 1,
    "double_spend": false,
```

```
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash":
        ↪  "b95bbc0006084b3edb5d87f55bd05b3696c5d34ca7f7ba74426219d5ebe24615",
        "output_index": 0,
        "script": "473044022001c00309a8e0b4dacf03c5553a2852d2549903e3a0902c12eb43b
        ↪  1e67f59a4ed02205c868f8894ddf5d66c1ff50aae8e11e825d2eb4aa6cebbb41f471b3
        ↪  5f0a6df2b012103019c64252a509d87deb3ac2592c017b5237d7b49b4b96d75845a9a0
        ↪  253740fd2",
        "output_value": 1000,
        "sequence": 4294967295,
        "addresses": [
          "mfc7vcpxHGRNjQ4qRB4CUkZvbAYCjCyswM"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 3009503
      }
    ],
    "outputs": [
      {
        "value": 1000,
        "script": "53210281db26acfc8c53b647b2c958c0fa73fb6dc3c97c0169c930db726b27b
        ↪  e10703f210357bab7788d238d673fb53685e3f1980c8fcaf3209fdf8dc26a9fe7e1476
        ↪  b0a2c21025676e7dfdedca8fefc3031f842bd0cb56c4dd516f72c64d82403b2ffeaff3
        ↪  46353ae",
        "addresses": [
          "zByATba1LvxeFfDKjScGMJXS362cPsaDZQ"
        ],
        "script_type": "pay-to-multi-pubkey-hash"
      }
    ]
  }
}
```

6 附录 2:Ex2b_output

```
luhaozhhhe@luhaozhhhe-virtual-machine:~/BlockChain2024/Ex2$ /bin/python3 /home/luhaozhhhe/BlockChain2024/Ex2/ex2b.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "95ab400c1865885d72af9da44209e67eff6494ac9aaf1ac7d4bb3fcfc0c038de",
    "addresses": [
      "zByATba1LvxeFfDKjScGMJXS362cPsaDZQ",
      "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
    ],
    "total": 1000,
    "fees": 0,
    "size": 376,
    "vsize": 376,
    "preference": "low",
    "relayed_by": "111.33.78.4",
    "received": "2024-10-08T08:49:56.321205594Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash":
        "b8c28455c3fa7498d49620dab820d41e1213432a1430ebca6be2418c475e1cb7",
        "output_index": 0,
        "script": "0047304402205035c894beaeb8e36f3143eb397438aa1147b42ddbe3871bf27
        8f9547f34468a02202d878d07878278d7444c7bb44f7306172e7829d24f98bf34d0b96
        e5068e6591c0147304402201b96d2581c01f61f217c5a3f292febc8e74775b9e085380
        0899ab3cfa9acfe990220431d674e507f44d08916d5be0d139e9e1660601c41946c350
        0781a844cda41670147304402203af5aca5657a1eaa5f65199cd1c8de3a4849017d71c
        42aa73159be1f67760ede022076633dbd16793961b3ac00462902f56491e541a2c135a
        b70a22c18725a876c420147304402206a6a476cc9c786246e4246d8d55ba6e0ba3b3f8
        f9e38e7a902f85d68c598ac3b02207b021f588f91d1e59011cefca7a367cb1af4112c9
        5e36f79e81f29254a43c15e01",
        "output_value": 1000,
        "sequence": 4294967295,
```

```
        "addresses": [
          "zByATba1LvxeFfDKjScGMJXS362cPsaDZQ"
        ],
        "script_type": "pay-to-multi-pubkey-hash",
        "age": 0
      }
    ],
    "outputs": [
      {
        "value": 1000,
        "script": "76a9149f9a7abd600c0caa03983a77c8c3df8e062cb2fa88ac",
        "addresses": [
          "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}
```