南开大学

Nankai University

南　　开　　大　　学

计　算　机　学　院

区块链实验报告

Ex4

徐俊智 2213410

符秀婷 2212939

年级：2022 级

专业：计算机科学与技术

指导教师：苏明

2024 年 11 月 18 日

目录

一、 前期准备

1. 配置环境

```
pip install -r requirements.txt
```

2. 为 Alice 和 Bob 创建 BTC testnet 密钥

```
xu@LENOVO:~/projects/Blockchain/Ex4$ python3 keygen.py
Private key: cMefQbn2xThyva1csJ3V1hhLtP4fZGm18zyCie82qgDNo7zpprDg
Address: mzHSnW6dKgKkWnefjuDRSKgu7aHR8UecyZ

xu@LENOVO:~/projects/Blockchain/Ex4$ python3 keygen.py
Private key: cW6MpGnF7gosYAFopFFYRsbHdJfk4hTTXGyj7yGvTSXzrRN8BhyZ
Address: moZ7a2TCTTE62ZBwYfpWSYfga6KXc5D33G
```

将 Alice 和 Bob 的 BTC testnet 密钥填入 keys.py

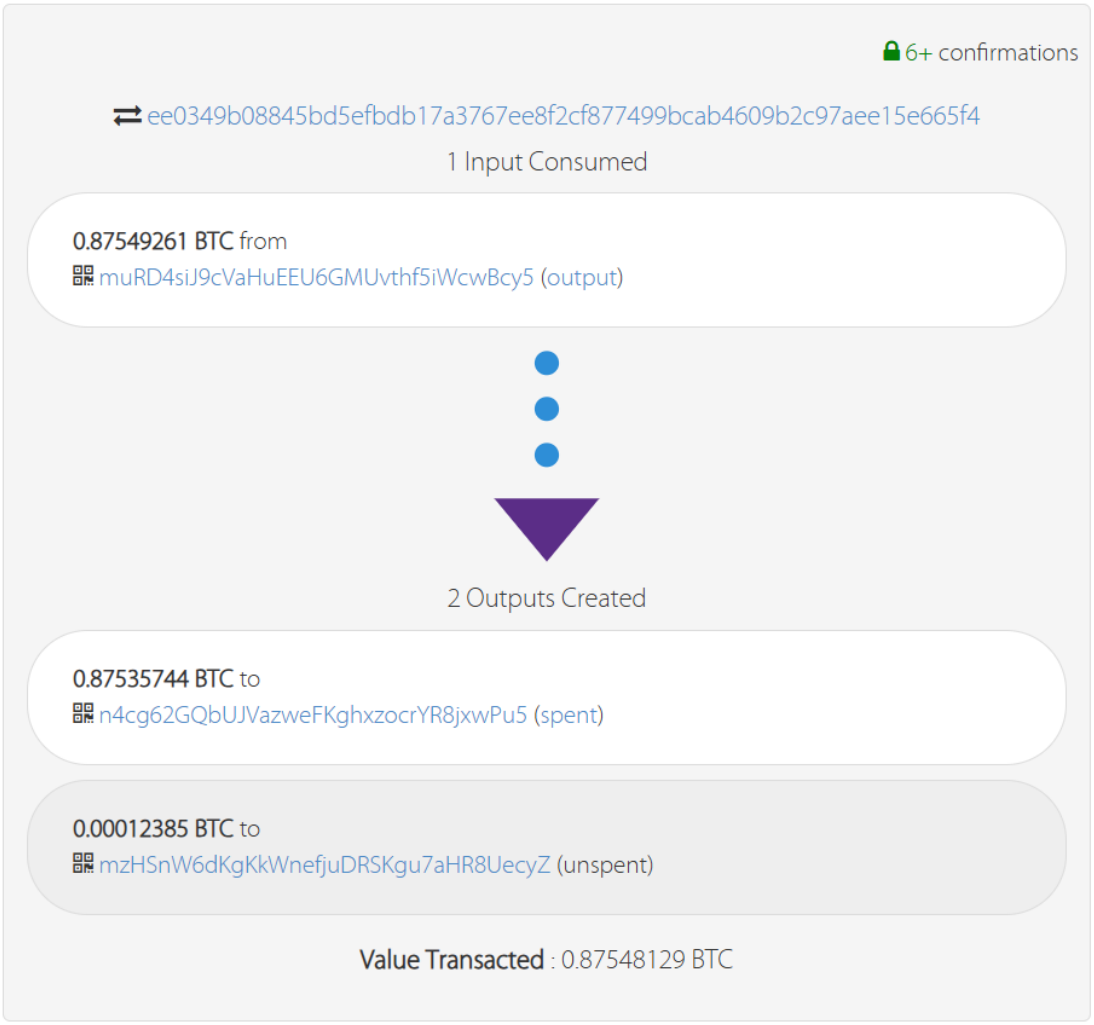
```
# Only to be imported by alice.py
# Alice should have coins!!
alice_secret_key_BTC = CBitcoinSecret(
    'cMefQbn2xThyva1csJ3V1hhLtP4fZGm18zyCie82qgDNo7zpprDg')

# Only to be imported by bob.py
bob_secret_key_BTC = CBitcoinSecret(
    'cW6MpGnF7gosYAFopFFYRsbHdJfk4hTTXGyj7yGvTSXzrRN8BhyZ')
```

3. 为 Alice 的 BTC 地址领取测试币

进入 faucet(https://coinfaucet.eu/en/btctestnet/)，点击 Bitcoin testnet3，粘贴地址 mzHSnW6dKgKkWnefjuDRSKgu7aHR8UecyZ，faucet 将给我们一个可消费的输出。

```
tx: ee0349b08845bd5efbdb17a3767ee8f2cf877499bcab4609b2c97aee15e665f4
```

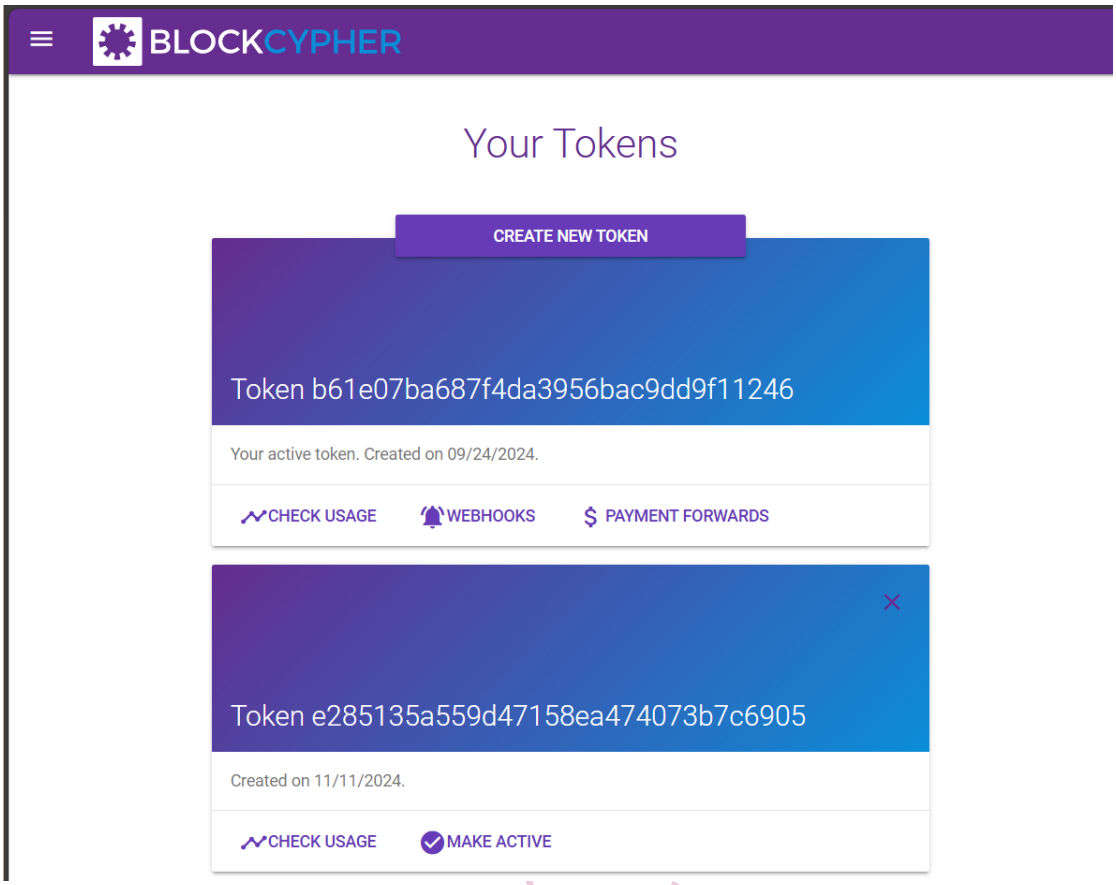
查看 https://live.blockcypher.com/网站的交易记录



4. 在 Blockcypher 注册帐户以获取 API token 在 Blockcypher 注册帐户以获取 API token: https://accounts.blockcypher.com/

点击 create new token

```
Alice:
b61e07ba687f4da3956bac9dd9f11246

Bob:
e285135a559d47158ea474073b7c6905
```



5. 为 Alice 和 Bob 创建 BCY testnet 密钥

```
xu@LENOVO:~/projects/Blockchain/Ex4$ curl -X POST https://api.blockcypher.com/v1/bcy/test/addrs?token=b61e07ba687f4da3956bac9dd9f11246
{
  "private": "1f0e7d70e9a6f778d22880a1e1622994e826497881ba898a7f7e1ad8bd45e61f",
  "public": "031d60feee079e6f210cd10f2163b13174fab56bd68e0a4e8f13c50800463a105c",
  "address": "C8PNMBEqQ5GgoF2kEgeubZk4z4ogqo35V7",
  "wif": "BpNQCNuCG7EuPaZHbVaYEWTu5YAknDuwtZtyj724ZjXzQZDFZ3t9"
}

xu@LENOVO:~/projects/Blockchain/Ex4$ curl -X POST https://api.blockcypher.com/v1/bcy/test/addrs?token=62fcb0c1140b4be2b4ce30baeb23c6a8
{
  "private": "744f8c2629623e3f64a2170afb20a9cc441ac8a9d893d8988cd17c0685d2c2e3",
  "public": "03a84808b962faf7aad486688f99f8adf6862d41f038e1e219a6955461f2b61e21",
  "address": "CAFQph1mS4h2ciwcPgEVMt8vqcSNxiYffg",
  "wif": "BsE88BKTNdgU92Cebmmjs3PxYm7EUDn58A9Sws9PWaPfzqbwfJ2d"
}
```

```
xu@LENOVO:~/projects/Blockchain/Ex4$ curl -X POST https://api.blockcypher.com/v1/b
cy/test/addrs?token=b61e07ba687f4da3956bac9dd9f11246
{
  "private": "1f0e7d70e9a6f778d22880a1e1622994e826497881ba898a7f7e1ad8bd45e61f",
  "public": "031d60feee079e6f210cd10f2163b13174fab56bd68e0a4e8f13c50800463a105c",
  "address": "C8PNMBEqQ5GgoF2kEgeubZk4z4ogqo35V7",
  "wif": "BpNQCNuCG7EuPaZHbVaYEWTu5YAknDuwtZtyj724ZjXzQZDFZ3t9"
}xu@LENOVO:~/projects/Blockchain/Ex4$ curl -X POST https://api.blockcypher.com/v1/
bcy/test/addrs?token62fcb0c1140b4be2b4ce30baeb23c6a8
{
  "private": "744f8c2629623e3f64a2170afb20a9cc441ac8a9d893d8988cd17c0685d2c2e3",
  "public": "03a84808b962faf7aad486688f99f8adf6862d41f038e1e219a6955461f2b61e21",
  "address": "CAFQph1mS4h2ciwcPgEVMt8vqcSNxiYffg",
  "wif": "BsE88BKTNdgU92Cebmmjs3PxYm7EUDn58A9Sws9PWaPfzqbwfJ2d"
}xu@LENOVO:~/projects/Blockchain/Ex4$
```

将 Alice 和 Bob 的 BCY testnet 密钥填入 keys.py

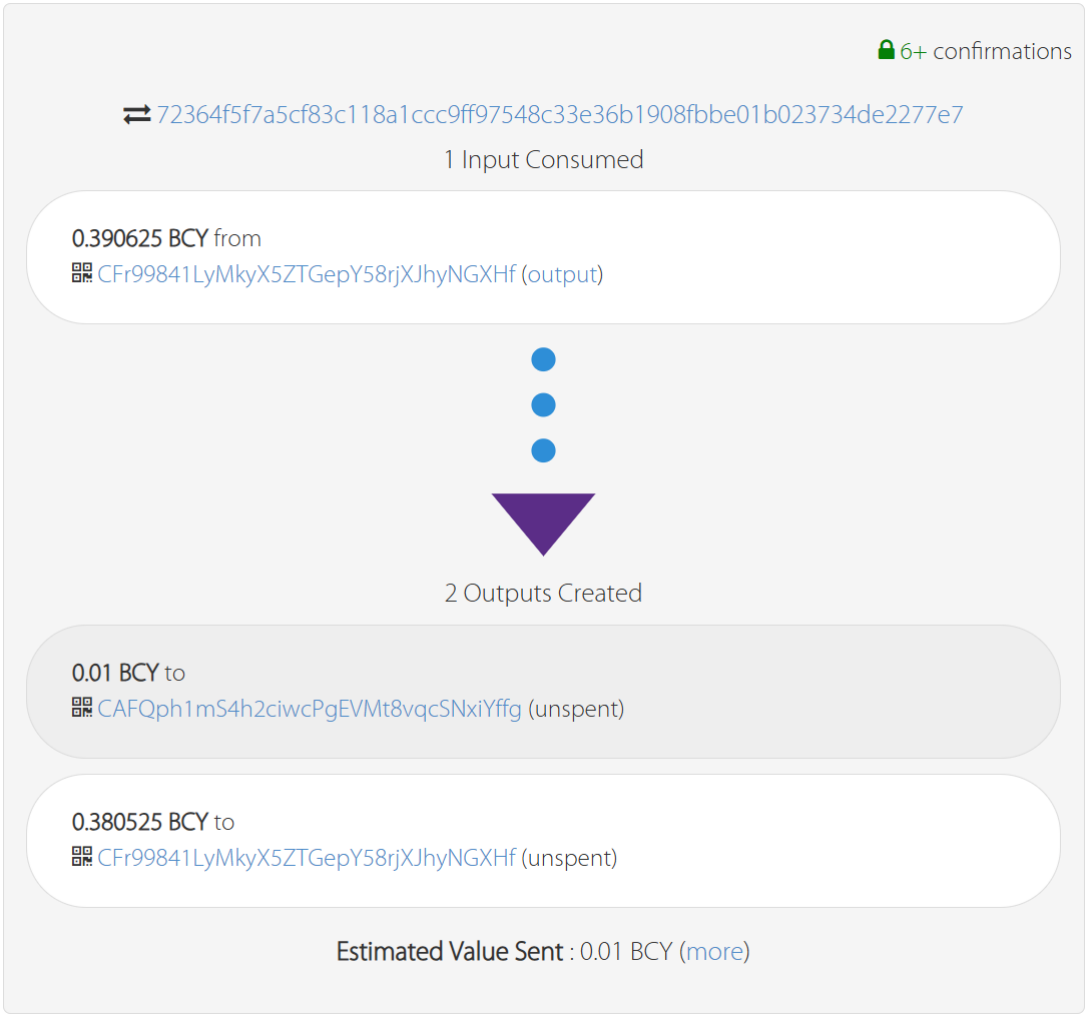
```
# Only to be imported by alice.py
alice_secret_key_BCY = CBitcoinSecret.from_secret_bytes(
    x('1f0e7d70e9a6f778d22880a1e1622994e826497881ba898a7f7e1ad8bd45e61f'))

# Only to be imported by bob.py
# Bob should have coins!!
bob_secret_key_BCY = CBitcoinSecret.from_secret_bytes(
    x('744f8c2629623e3f64a2170afb20a9cc441ac8a9d893d8988cd17c0685d2c2e3'))
```

6. 为 Bob 的 BCY 地址领取测试币

```
xu@LENOVO:~/projects/Blockchain/Ex4$ curl -d '{"address":"
    CAFQph1mS4h2ciwcPgEVMt8vqcSNxiYffg", "amount": 1000000}' https://api.
    blockcypher.com/v1/bcy/test/faucet?token=e285135a559d47158ea474073b7c6905
{
  "tx_ref": "72364f5f7a5cf83c118a1ccc9ff97548c33e36b1908fbbe01b023734de2277e7
      "
}
```

```
xu@LENOVO:~/projects/Blockchain/Ex4$ curl -d '{"address":"CAFQph1mS4h2ciwcPgEVMt8v
qcSNxiYffg", "amount": 1000000}' https://api.blockcypher.com/v1/bcy/test/faucet?to
ken=e285135a559d47158ea474073b7c6905
{
  "tx_ref": "72364f5f7a5cf83c118a1ccc9ff97548c33e36b1908fbbe01b023734de2277e7"
}xu@LENOVO:~/projects/Blockchain/Ex4$
```

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



7. 使用 `split_test_coins.py` 划分领取的币

设置交易参数

```
my_private_key = CBitcoinSecret('ee0349b08845bd5efbdb17a3767ee8f2cf877499bcab4609b2c97aee15e665f4')
amount_to_send = 0.00012 # amount of BTC in the output you're splitting minus fee

txid_to_spend = (
    'ee0349b08845bd5efbdb17a3767ee8f2cf877499bcab4609b2c97aee15e665f4')
```

运行

```
python3 split_test_coins.py
```

命令行输出交易信息

```
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
```

```
    "hash": "52e4e8d41d0e64e129e1537973846241047d1422cbcf63b3ef56d57b77d136b0
        ",
    "addresses": [
      "mzHSnW6dKgKkWnefjuDRSKgu7aHR8UecyZ"
    ],
    "total": 12000,
    "fees": 385,
    ......
```

详情见《split_test_coins 交易结果.md》

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

8. 填写 swap.py

```
##############################################################################
#
# Configured for your addresses
#
# TODO: Fill in all of these fields
#

alice_txid_to_spend      = "52
    e4e8d41d0e64e129e1537973846241047d1422cbcf63b3ef56d57b77d136b0"
alice_utxo_index         = 0
alice_amount_to_send     = 0.0000001

bob_txid_to_spend        = "72364
    f5f7a5cf83c118a1ccc9ff97548c33e36b1908fbbe01b023734de2277e7"
bob_utxo_index           = 0
bob_amount_to_send       = 0.01
```

```
15
16 # Get current block height (for locktime) in 'height' parameter for each
       blockchain (and put it into swap.py):
17 #  curl https://live.blockcypher.com/btc-testnet/
18 btc_test3_chain_height  = 3254861
19
20 #  curl https://live.blockcypher.com/bcy/
21 bcy_test_chain_height   = 1586104
22
23 # Parameter for how long Alice/Bob should have to wait before they can take
       back their coins
24 ## alice_locktime MUST be > bob_locktime
25 alice_locktime = 5
26 bob_locktime = 3
27
28 tx_fee = 0.00000001
29
30 broadcast_transactions = True
31 alice_redeems = True
32
33 #
34 #
35 ################################################################################
```

二、 练习（A）

考虑创建跨链原子交换所需事务所需的 ScriptPubKey。此交易必须可由接收者赎回（如果他们有一个与 Hash（x）对应的秘密 x），或者可以用发送者和接收者的两个签名赎回。完善 `swap_scripts.py` 中的脚本 coinExchangeScript。

锁定脚本：

```
1  OP_IF, # 接收方提供x和签名
2      OP_HASH160,
3      hash_of_secret,
4      OP_EQUALVERIFY,
5      public_key_recipient,
6      OP_CHECKSIG,
7  OP_ELSE, # 双方都提供签名
8      public_key_recipient,
9      OP_CHECKSIGVERIFY,
10     public_key_sender,
11     OP_CHECKSIG,
12 OP_ENDIF
```

三、 练习（B）

完善脚本：

（a）在接收者知道秘密 x 的情况下，编写赎回交易所需的 ScriptSig。在 `swap_scripts.py` 中完善 coinExchangeScriptSig1.

(b)在发送方和接收方都签署事务的情况下,编写赎回事务所需的 ScriptSig 。在 `swap_scripts.py` 中完善 coinExchangeScriptSig2.

解锁脚本 1

```
sig_recipient ,
secret ,
OP_TRUE
```

解锁脚本 2

```
sig_sender ,
sig_recipient ,
OP_FALSE
```

我们分析一下验证脚本的执行堆栈状态图：

1. 解锁脚本 1+ 锁定脚本

（1）当前交易输入的解锁脚本 1 (`txin_scriptSig`) 执行：

1. `sig_recipient`
   - 接收方的签名 `sig_recipient` 入栈
   - 堆栈状态:[`sig_recipient`]
2. secret
   - 秘密 x (secret) 入栈
   - 堆栈状态:[`sig_recipient`, secret]
3. `OP_TRUE`
   - `OP_TRUE` 入栈
   - 堆栈状态:[`sig_recipient`, secret, `OP_TRUE`]

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

1. `OP_IF`
   - 检查栈顶的值。因为是 `OP_TRUE`，所以会执行 `OP_IF` 分支
   - 堆栈状态:[`sig_recipient`, secret]
2. `OP_HASH160`
   - 对栈顶元素进行哈希处理
   - 堆栈状态:[`sig_recipient`, `hash_of_secret`]
3. `hash_of_secret`

- 秘密 x 的哈希值 `hash_of_secret` 入栈
- 堆栈状态:[`sig_recipient`, `hash_of_secret`,`hash_of_secret`]

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:[`sig_recipient`]
5. `public_key_recipient`
   - 接收方的公钥 `public_key_recipient` 入栈
   - 堆栈状态:[`sig_recipient`,`public_key_recipient`]
6. `OP_CHECKSIG`
   - 弹出栈顶的两个元素（公钥和签名），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]

2. 解锁脚本 2+ 锁定脚本

（2）当前交易输入的解锁脚本 2 (`txin_scriptSig`) 执行:

1. `sig_sender`
   - 发送方的签名 `sig_sender` 入栈
   - 堆栈状态:[`sig_sender`]
2. `sig_recipient`
   - 接收方的签名 `sig_recipient` 入栈
   - 堆栈状态:[`sig_sender`,`sig_recipient`]
3. `OP_FALSE`
   - `OP_FALSE` 入栈
   - 堆栈状态:[`sig_sender`,`sig_recipient`, `OP_FALSE`]

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

1. `OP_IF`
   - 检查栈顶的值。因为是 `OP_FALSE`，所以会跳过 `OP_IF` 分支，执行 `OP_ELSE` 分支
   - 堆栈状态:[`sig_sender`,`sig_recipient`]
2. `public_key_recipient`
   - 接收方的公钥 `public_key_recipient` 入栈
   - 堆栈状态:[`sig_sender`,`sig_recipient`,`public_key_recipient`]
3. `OP_CHECKSIGVERIFY`

- 弹出栈顶的两个元素（公钥和签名），检查是否匹配。如果匹配，继续执行；否则，脚本失败。
- 堆栈状态:[sig_sender]

4. public_key_sender
   - 发送方的公钥 public_key_sender 入栈
   - 堆栈状态:[sig_sender,public_key_sender]
5. OP_CHECKSIG
   - 弹出栈顶的两个元素（公钥和签名），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]

四、 练习（C）

运行你的代码 swap.py。注意，文件中需要填写区块高度，代码注释中的方法不可用，请使用以下地址，BTC 高度查询地址：https://live.blockcypher.com/btc-testnet/，BCY 区块高度查询地址：https://live.blockcypher.com/bcy/

我们不需要广播事务，因为这需要等待一些时间来验证。设置 broadcast_transactions=False。将本地验证 ScriptSig+ScriptPK 是否返回 true。将 alice_redeems=True 和 alice_redeems=False 分别尝试此操作。

可选：尝试使用 broadcast_transactions=True，这将使代码休眠一段适当的时间，以便将所有内容发布到区块链并正确验证。需要 20-60 分钟甚至更长时间才能运行。

设置 broadcast_transactions=False,alice_redeems=False。这会在本地模拟 Alice 没有揭示秘密 x，因此 Bob 和 Alice 可以在指定时间后取回他们原始的比特币。

设置 broadcast_transactions=False,alice_redeems=True。这会在本地模拟 Alice 使用秘密 x 赎回 Bob 的比特币，在完成赎回时公开秘密 x 。Bob 获得秘密 x 后，可以用它赎回 Alice 的比特币。

设置 broadcast_transactions=True,alice_redeems=True。将内容广播到区块链。Alice 使用秘密 x 赎回 Bob 的比特币，在完成赎回时公开秘密 x 。Bob 获得秘密 x 后，可以用它赎回 Alice 的比特币。

运行

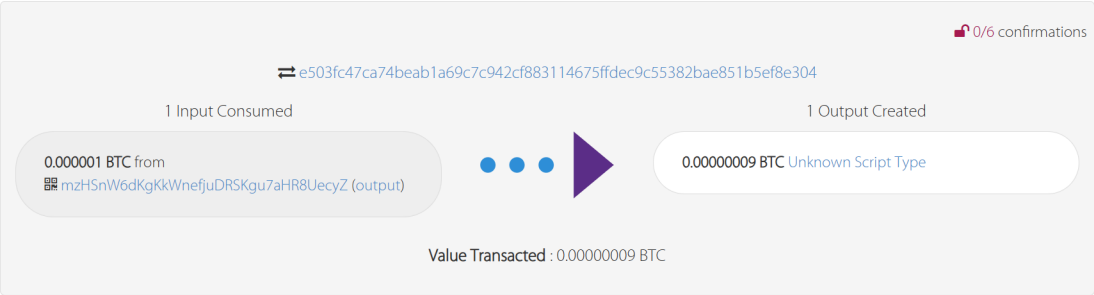
```
python3 ex3b.py
```

命令行输出交易信息

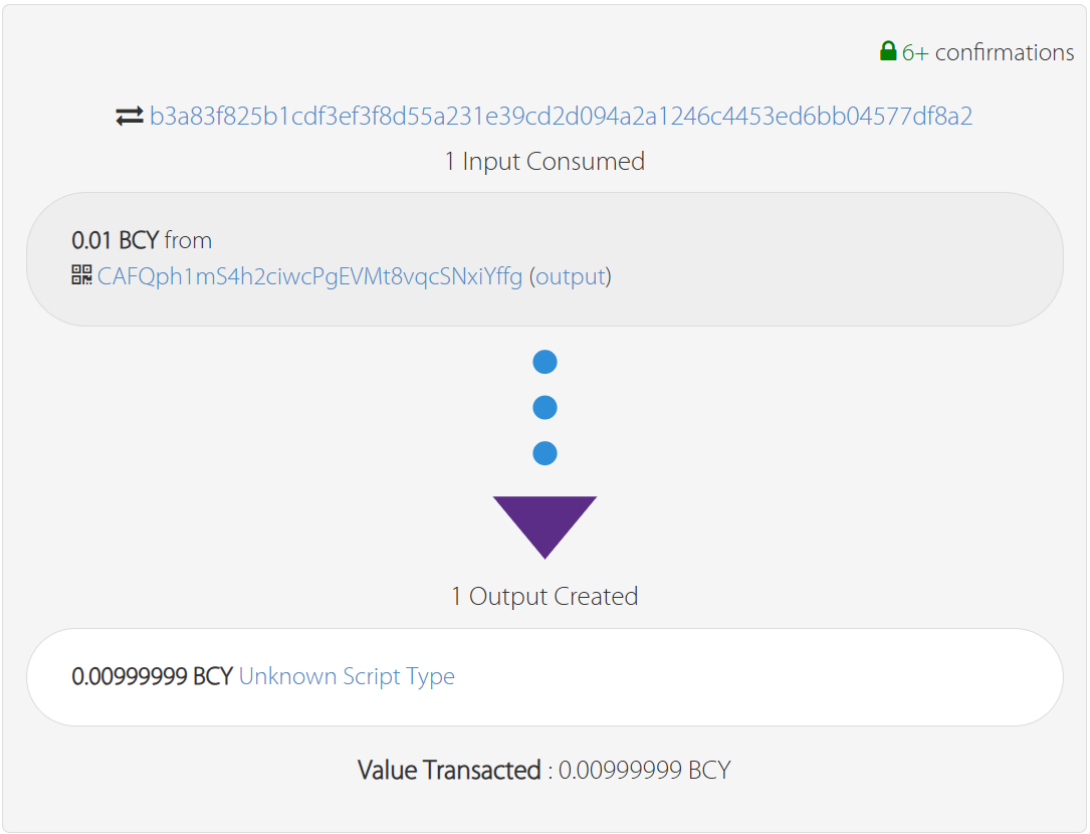
```
Alice swap tx (BTC) created successfully!
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "179aeec771a9c6bfd8833a7508fc17a3960970f2289fda5d4682619419042d2f
        ",
    "addresses": [
```

```
        "mzHSnW6dKgKkWnefjuDRSKgu7aHR8UecyZ"
      ],
      "total": 9,
      "fees": 1191,
      ......
Bob swap tx (BCY) created successfully!
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "b3a83f825b1cdf3ef3f8d55a231e39cd2d094a2a1246c4453ed6bb04577df8a2
        ",
    "addresses": [
      "CAFQph1mS4h2ciwcPgEVMt8vqcSNxiYffg"
    ],
    "total": 999999,
    "fees": 1,
    ......
Sleeping for 20 minutes to let transactions confirm...
Alice redeem from swap tx (BCY) created successfully!
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "9f543ec3f2fec8e710931be0be7cd2b9be706dd62dad7e27ee5669c23e5dfaed
        ",
    "addresses": [
      "C8PNMBEqQ5GgoF2kEgeubZk4z4ogqo35V7"
    ],
    "total": 999998,
    "fees": 1,
    ......
Bob redeem from swap tx (BTC) created successfully!
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "1e960f3fbe164a6d14fa4f8cd701e6c87dbaec8ec6786df26e743afb78960c2d
        ",
    "addresses": [
      "moZ7a2TCTTE62ZBwYfpWSYfga6KXc5D33G"
    ],
    "total": 8,
    "fees": 1,
    ......
```

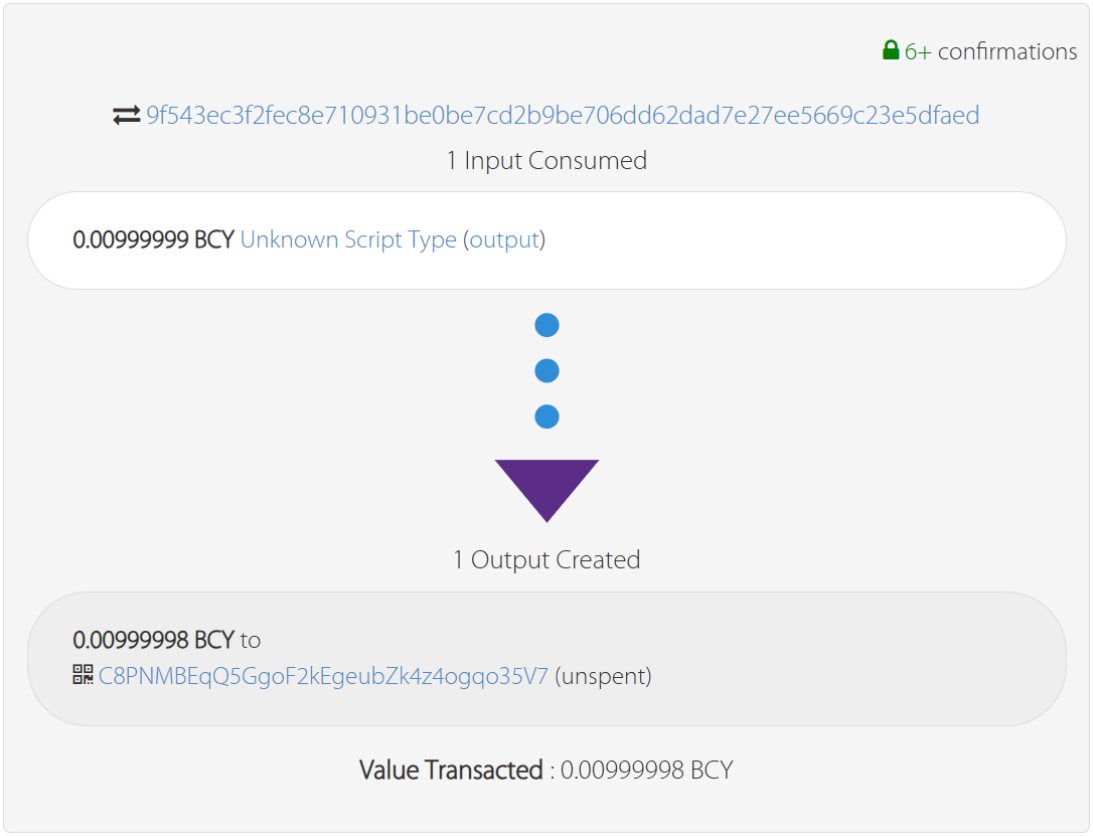
详情见《swap 交易结果.md》
查看 https://live.blockcypher.com/网站的交易记录
Alice 发送比特币到 Bob



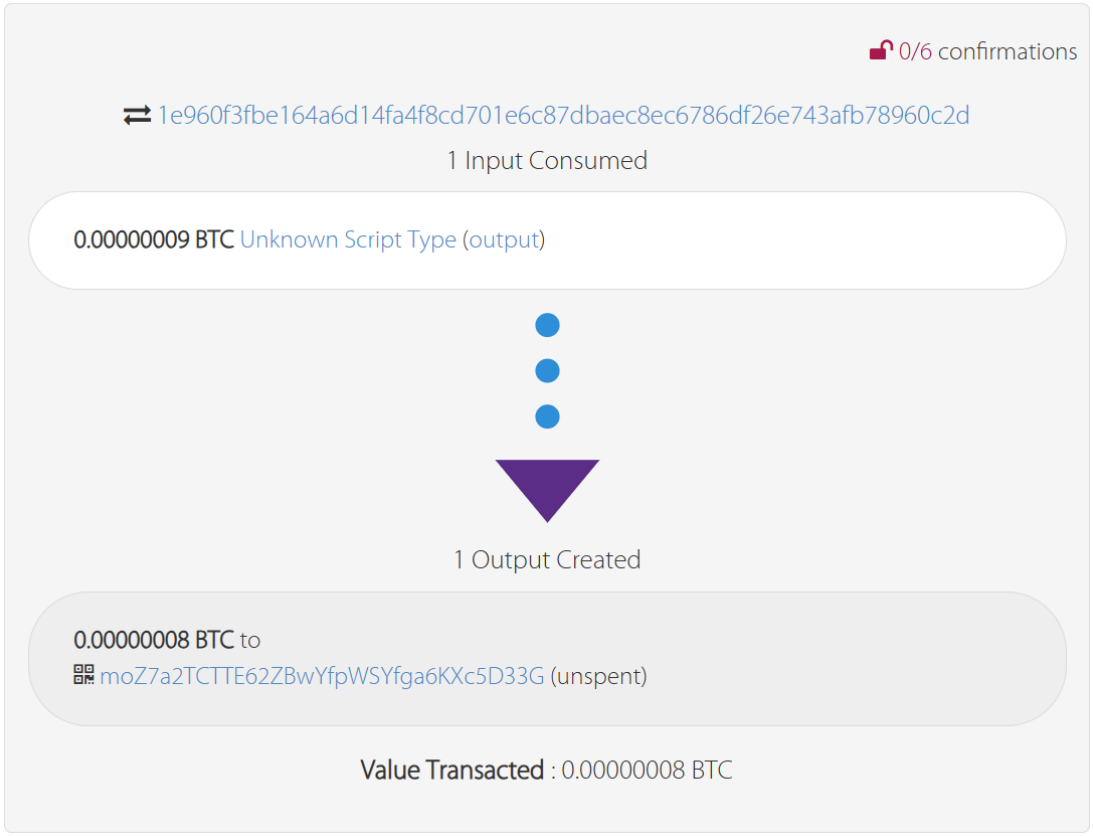
Bob 发送比特币到 Alice



Alice 赎回 Bob 的比特币



Bob 赎回 Alice 的比特币



五、 练习（D）

请写一个简短的关于这个项目的设计文档

（a）解释你写的代码内容，以及 coinExchangeScript 是如何工作的。

```
# This is the ScriptPubKey for the swap transaction
def coinExchangeScript(public_key_sender, public_key_recipient,
    hash_of_secret):
    return [
        OP_IF, # 如果栈顶是"真"值（即接收方提供秘密x和签名）
            OP_HASH160, # 对栈顶元素进行哈希处理
            hash_of_secret, # 秘密 x 的哈希值
            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
            public_key_recipient, # 接收方的公钥（Bob 的公钥）
            OP_CHECKSIG, # 弹出栈顶的两个元素（公钥和签名），检查是否匹配。如
                果匹配，将 True 压入栈顶；否则，将 False 压入栈顶。
        OP_ELSE, # 如果栈顶是"假"值（即双方提供签名）
            public_key_recipient, # 接收方的公钥（Bob 的公钥）
            OP_CHECKSIGVERIFY, # 弹出栈顶的两个元素（公钥和签名），检查是否匹
                配。如果匹配，继续执行；否则，脚本失败。
            public_key_sender, # 发送方的公钥（Alice 的公钥）
            OP_CHECKSIG, # 弹出栈顶的两个元素（公钥和签名），检查是否匹配。如
                果匹配，将 True 压入栈顶；否则，将 False 压入栈顶。
        OP_ENDIF
    ]

# This is the ScriptSig that the receiver will use to redeem coins
def coinExchangeScriptSig1(sig_recipient, secret):
    return [
        sig_recipient, # 接收方的签名（Bob 的签名）
        secret, # 秘密 x
        OP_TRUE
    ]

# This is the ScriptSig for sending coins back to the sender if unredeemed
def coinExchangeScriptSig2(sig_sender, sig_recipient):
    return [
        sig_sender, # 发送方的签名（Alice 的签名）
        sig_recipient, # 接收方的签名（Bob 的签名）
        OP_FALSE
    ]
```

`coinExchangeScript` 实现了两种赎回方案:

1. 使用 `OP_IF` 和 `OP_ELSE` 来判断进行哪种赎回方案。
2. 如果执行 `OP_IF` 分支，表明接收方（Bob）提供秘密 x 和签名。通过 `OP_HASH160` 对秘密 x 进行哈希处理，比较哈希值与锁定脚本中的哈希值是否一致。如果一致，使用 `OP_CHECKSIG` 验证 Bob 的签名，完成赎回。

3. 如果执行 `OP_ELSE` 分支，表明发送方（Alice）和接收方（Bob）共同提供签名。使用 `OP_CHECKSIGVERIFY` 验证 Bob 的签名，然后使用 `OP_CHECKSIG` 验证 Alice 的签名

（b）以 Alice 用 coinExchangeScript 向 Bob 发送硬币为例：

1. 如果 Bob 不把钱赎回来，Alice 为什么总能拿回她的钱?

   Alice 创建交易 `alice_swap_tx` 后,又创建了一个时间锁定交易 `alice_return_coins_tx`,只有当 Bob 签名时间锁定交易 `alice_return_coins_tx` 后,Alice 才会公布交易 `alice_swap_tx`。如果 Bob 不赎回,或者没有签署交易,那么 Alice 会在超时后使用时间锁定交易 `alice_return_coins_tx` 取回自己原来的比特币。

2. 为什么不能用简单的 1/2 multisig 来解决这个问题?

   在 1/2 multisig 条件下，提供任意一方签名就能将钱赎回，那么可能会发生其中一方赎回对方的钱后又赎回自己的钱。

（c）解释 Alice (Bob) 创建的一些交易内容和先后次序，以及背后的设计原理。
跨链原子交换 `atomic_swap` 函数

- `hash_of_secret`：Alice 选择一个秘密 x，计算其哈希值。
- `alice_swap_tx` 和 `alice_swap_scriptPubKey`：Alice 创建一个交易 `alice_swap_tx` 和锁定脚本 `alice_swap_scriptPubKey`，这个脚本在解锁交易时验证 Bob 提供的 x 和签名，或者 Alice 和 Bob 共同提供的签名。
- `alice_return_coins_tx`：Alice 创建一个时间锁定交易 `alice_return_coins_tx`，如果规定时间内交换没有完成，那么 Alice 可以使用时间锁定交易取回自己原来的比特币。
- `bob_signature_BTC`：Bob 对 Alice 的时间锁定交易进行签名。
- 如果 `broadcast_transactions` 为 True，Alice 会将她的交易广播到 BTC Testnet3 上。
- `bob_swap_tx` 和 `bob_swap_scriptPubKey`：Bob 创建一个交易 `bob_swap_tx` 和锁定脚本 `bob_swap_scriptPubKey`，这个脚本在解锁交易时验证 Alice 提供的 x 和签名，或者 Alice 和 Bob 共同提供的签名。
- `bob_return_coins_tx`：Bob 创建一个时间锁定交易 `bob_return_coins_tx`，如果规定时间内交换没有完成，那么 Bob 可以使用时间锁定交易取回自己原来的比特币。
- `alice_signature_BCY`：Alice 对 Bob 的时间锁定交易进行签名。
- 如果 `broadcast_transactions` 为 True，Bob 会将他的交易广播到 BCY Testnet 上。
- 如果 `broadcast_transactions` 为 True，程序会等待 20 分钟，以便交易在各自的区块链上得到确认。
- 如果 `alice_redeems` 为 True，Alice 会使用秘密 x 赎回 Bob 在 BCY Testnet 上的比特币，在完成赎回时公开秘密 x 。Bob 获得秘密 x 后，可以用它在 BTC Testnet3 上赎回 Alice 的比特币。
- 如果 `alice_redeems` 为 False，Alice 和 Bob 将等待直到他们各自的时间锁定期过后，使用他们的时间锁定交易来取回他们的原始比特币。

（d）以该作业为例，一次成功的跨链原子交换中，数字货币是如何流转的？如果失败，数字货币又是如何流转的？

如果交换成功，数字货币的流转：

1. Alice 发送比特币到 Bob，锁定的比特币只有在 Bob 提供秘密 x 时才能解锁。
2. Bob 发送比特币到 Alice，锁定的比特币只有在 Alice 提供秘密 x 时才能解锁。
3. Alice 使用秘密 x 赎回 Bob 的比特币，并公开秘密 x
4. Bob 使用秘密 x 赎回 Alice 的比特币

如果交换失败，数字货币的流转：

1. 如果 Bob 没有提供秘密 x，Alice 可以在设定的时间锁定期结束后，通过时间锁定交易取回她的比特币。
2. 如果 Alice 没有提供秘密 x，Bob 可以在约定的时间锁定期结束后，通过时间锁定交易取回他的比特币。