区块链lab4

Lab4 设计文档

2112209 张佳璐 计算机科学与技术

2112060 孙蕗 信息安全

准备工作

1. 为Alice 和 Bob 创建 BTC testnet 密钥。你可以用 keygen.py 生成密钥，把它填入keys.py 中合适的地方。

Alice BTC账户

```
Private key: cSji1idwDiE6UcCCCsaHMoPawoELhTvNUS8a6BXcDmEU2VGWGHuS
Address: n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk
```

Bob BTC

```
Private key: cVGneoryx5qMQpd8e1aFeggJKdU2muJaKBLjagG8mR9AeNr5U1rD
Address: monqWwckKskrYXDBto1a3zzL9ePxC9cFzx
```

2. 在Project1 中相同的 coinfaucet 上，https://coinfaucet.eu/en/btc-testnet/，为 Alice的 BTC 地址领取测试币。

Alice 领取bitcoin

```
tx: d9314b8d4cd4127619ab87416ec51e430e3a0358d5b6d31b53186a4e7dc28720
0.01343801
```

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

```
Alice token: 0cba5c8b52c9406bb1f3dd8a63b8e0b7
Bob token: 6b848750c13c4577a456cc1f695e60f1
```

4. 为 Alice 和 Bob 创建 BCY testnet 密钥并填入 keys.py.

Alice BCY账户

```
zjl@vm:~/blockchain/Exercise4$ curl -X POST https://api.blockcypher.com/v1/bcy/t
{
  "private": "72e9a3846c06634892db9a2278999466cdc9878507a6d6165c77301c8fc90a02",
  "public": "0326c1341605523fdec666e3b60e83ddbe70c489c1c8be0c42e4d084cec5f25692"
  "address": "BsWfitMMv4CTb8CTxRR4MxWbSYN2XfXb1h",
  "wif": "BsBQVt3Y3KM89rjrUFSbcD1GVvN9MYS8Cp9YK2Fe17jCxiyC4EQD"
}
```

Bob BCY账户

```
judy@judy-virtual-machine:~/桌面/UpdateExercise4-Bob$ curl -X POST https://api.b
{
  "private": "76e2297bda40e4824fc090d581655b4496552df198598ad61766169450e50598",
  "public": "0234483cb7b056f2f9290ef1f693299237a31bdca8b0e20a466041f2d98231b0e1"
  "address": "Bz8DDXh2W6RzyNptFP6L6kc4Xjzkpmupkj",
  "wif": "BsK8Bc1Zk4CvdUjfm71KY5oSjmrfmhpuDb5j7aBgDkTxCjTtnTVi"
}
```

keys.py

```
# Only to be imported by alice.py
# Alice should have coins!!
alice_secret_key_BTC = CBitcoinSecret(
    'cSji1idwDiE6UcCCCsaHMoPawoELhTvNUS8a6BXcDmEU2VGWGHuS')

# Only to be imported by bob.py
bob_secret_key_BTC = CBitcoinSecret(
    'cVGneoryx5qMQpd8e1aFeggJKdU2muJaKBLjagG8mR9AeNr5U1rD')

# Only to be imported by alice.py
alice_secret_key_BCY = CBitcoinSecret.from_secret_bytes(
    x('72e9a3846c06634892db9a2278999466cdc9878507a6d6165c77301c8fc90a02'))

# Only to be imported by bob.py
# Bob should have coins!!
bob_secret_key_BCY = CBitcoinSecret.from_secret_bytes(
    x('76e2297bda40e4824fc090d581655b4496552df198598ad61766169450e50598'))
```

5. 在 Blockcypher 测试网（BCY）上为 Bob 的 BCY 地址领取测试币。

```
judy@judy-virtual-machine:~/桌面/UpdateExercise4-Bob$ curl -d '{"address":"Bz8DD
{
  "tx_ref": "3dc37acdbf64dcc45363055af737b219891824269b10b29ec5785edaef68267a"
}
```

6. 使用 split_test_coins.py（填写文件中的相关字段）划分领取的币。

Alice（BTC）

```
    my_private_key = CBitcoinSecret('cSji1idwDiE6UcCCCsaHMoPawoELhTvNUS8a6BXcDmE

    my_public_key = my_private_key.pub
    my_address = P2PKHBitcoinAddress.from_pubkey(my_public_key)

    amount_to_send = 0.01 # amount of BTC in the output you're splitting minus f
    txid_to_spend = (
        'd9314b8d4cd4127619ab87416ec51e430e3a0358d5b6d31b53186a4e7dc28720')
    utxo_index = 0
    n = 10 # number of outputs to split the input into
    network = 'btc-test3' # either 'btc-test3' or 'bcy-test'
```

```
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "979c0254f32bd0d6d812468bfc60faf5d29f3596cc6af29e50d0e63cff3a2a6d",
    "addresses": [
      "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
    ],
    "total": 1000000,
    "fees": 343801,
    "size": 497,
    "vsize": 497,
    "preference": "high",
    "relayed_by": "163.123.192.136",
    "received": "2023-11-08T01:07:41.052266888Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
```

```
    "vout_sz": 10,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "d9314b8d4cd4127619ab87416ec51e430e3a0358d5b6d31b53186a4e7d
        "output_index": 0,
        "script": "473044022021b56d66be033c1be40bd460feb6a80fae29c9a57e4f9e60e54
        "output_value": 1343801,
        "sequence": 4294967295,
        "addresses": [
          "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 2537423
      }
    ],
    "outputs": [
      {
        "value": 100000,
        "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
        "addresses": [
          "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
        ],
        "script_type": "pay-to-pubkey-hash"
      },
      {
        "value": 100000,
        "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
        "addresses": [
          "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
        ],
        "script_type": "pay-to-pubkey-hash"
      },
      {
        "value": 100000,
        "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
        "addresses": [
          "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
        ],
        "script_type": "pay-to-pubkey-hash"
      },
      {
        "value": 100000,
        "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
        "addresses": [
          "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
        ],
```

```
      "script_type": "pay-to-pubkey-hash"
    },
    {
      "value": 100000,
      "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
      "addresses": [
        "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
      ],
      "script_type": "pay-to-pubkey-hash"
    },
    {
      "value": 100000,
      "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
      "addresses": [
        "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
      ],
      "script_type": "pay-to-pubkey-hash"
    },
    {
      "value": 100000,
      "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
      "addresses": [
        "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
      ],
      "script_type": "pay-to-pubkey-hash"
    },
    {
      "value": 100000,
      "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
      "addresses": [
        "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
      ],
      "script_type": "pay-to-pubkey-hash"
    },
    {
      "value": 100000,
      "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
      "addresses": [
        "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
      ],
      "script_type": "pay-to-pubkey-hash"
    },
    {
      "value": 100000,
      "script": "76a914d7fd1c8db992aad785d14e40573ac26f3cfedde888ac",
      "addresses": [
        "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
```

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

```

7. 填写swap.py

“curl https://api.blockcypher.com/v1/btc/test3”，得到btc_test3_chain_height = 2537460

```
zjl@vm:~/blockchain/Exercise4$ curl https://api.blockcypher.com/v1/btc/test3
{
  "name": "BTC.test3",
  "height": 2537460,
  "hash": "000000000000000daf8227e130aa21ef37fd2e0ff02b80bf8d5b377b171f7279",
  "time": "2023-11-08T04:48:30.944983729Z",
  "latest_url": "https://api.blockcypher.com/v1/btc/test3/blocks/000000000000000
  "previous_hash": "00000000f427d705c8f02f83c629a6453bafd34120dfad9c9e94406a8b04
  "previous_url": "https://api.blockcypher.com/v1/btc/test3/blocks/00000000f427d
  "peer_count": 300,
  "unconfirmed_count": 119,
  "high_fee_per_kb": 41442,
  "medium_fee_per_kb": 29944,
  "low_fee_per_kb": 14139,
  "last_fork_height": 2535938,
  "last_fork_hash": "000000000000277a0446852804a8b6c22cd69225755736606e1804b4de3
}

```

“curl https://api.blockcypher.com/v1/bcy/test”，得到bcy_test_chain_height = 1058748

```
judy@judy-virtual-machine:~/桌面/UpdateExercise4-Bob$ curl https://api.blockcyph
{
  "name": "BCY.test",
  "height": 1058748,
  "hash": "000096629610b696267d02823caf2b52ad00f7dcc4c2d594256fb21719021b00",
  "time": "2023-11-08T05:24:29.081477633Z",
  "latest_url": "https://api.blockcypher.com/v1/bcy/test/blocks/000096629610b696
  "previous_hash": "0000b1662d7fa1db2e14ada8bcc88ebd62fb3f26c1d0895c8f8b35ecb840
  "previous_url": "https://api.blockcypher.com/v1/bcy/test/blocks/0000b1662d7fa1
  "peer_count": 0,
  "unconfirmed_count": 1,
```

```
  "high_fee_per_kb": 111235,
  "medium_fee_per_kb": 67114,
  "low_fee_per_kb": 45000,
  "last_fork_height": 1048976,
  "last_fork_hash": "000022cd98b80a82ddbcf9ded4ae99d75ffb326c6fc74919b18f54a5f29
}
```

swap.py

```
alice_txid_to_spend     = "979c0254f32bd0d6d812468bfc60faf5d29f3596cc6af29e50d0e
alice_utxo_index        = 3
alice_amount_to_send    = 0.000111

bob_txid_to_spend       = "3dc37acdbf64dcc45363055af737b219891824269b10b29ec5785
bob_utxo_index          = 0
bob_amount_to_send      = 0.0001

# Get current block height (for locktime) in 'height' parameter for each blockch
#  curl https://api.blockcypher.com/v1/btc/test3
btc_test3_chain_height  = 2537460

#  curl https://api.blockcypher.com/v1/bcy/test
bcy_test_chain_height   = 1058748

# Parameter for how long Alice/Bob should have to wait before they can take back
## alice_locktime MUST be > bob_locktime
alice_locktime = 5
bob_locktime = 3

tx_fee = 0.00001

broadcast_transactions = True
alice_redeems = True
```

```
Alice swap tx (BTC) created successfully!
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "ce157e2cc849d0a040f2e36c0f0882a45f3a151f5c08080014b8b8e3a6cf0721",
    "addresses": [
      "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
```

```
    ],
    "total": 10100,
    "fees": 89900,
    "size": 267,
    "vsize": 267,
    "preference": "high",
    "relayed_by": "163.123.192.135",
    "received": "2023-11-08T05:25:15.76468833Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "979c0254f32bd0d6d812468bfc60faf5d29f3596cc6af29e50d0e63cff
        "output_index": 3,
        "script": "4830450221009650ea6b46b710ef1ae7a056105da7ead78c2c4729db74d47
        "output_value": 100000,
        "sequence": 4294967295,
        "addresses": [
          "n1Czno6KhSRa6qdE4Ed7tJWVwsn4FAywgk"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 2537424
      }
    ],
    "outputs": [
      {
        "value": 10100,
        "script": "21030e0fcc01bd78c258b00eec01670f8dec84ceedb2e4faf22aaecf93d60
        "addresses": null,
        "script_type": "unknown"
      }
    ]
  }
}
Bob swap tx (BCY) created successfully!
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "70ef4153be61b1eb370cbafd6605cb321204aa513a5ad7c0ed6ccaa107c181d5",
    "addresses": [
      "Bz8DDXh2W6RzyNptFP6L6kc4Xjzkpmupkj"
    ],
```

```
    "total": 9000,
    "fees": 991000,
    "size": 267,
    "vsize": 267,
    "preference": "high",
    "relayed_by": "163.123.192.136",
    "received": "2023-11-08T05:25:17.009141033Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "3dc37acdbf64dcc45363055af737b219891824269b10b29ec5785edaef
        "output_index": 0,
        "script": "48304502210095e80620d449da5aa09567127e826b0d95994c9bef5d9f4e6
        "output_value": 1000000,
        "sequence": 4294967295,
        "addresses": [
          "Bz8DDXh2W6RzyNptFP6L6kc4Xjzkpmupkj"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 0
      }
    ],
    "outputs": [
      {
        "value": 9000,
        "script": "210326c1341605523fdec666e3b60e83ddbe70c489c1c8be0c42e4d084cec
        "addresses": null,
        "script_type": "unknown"
      }
    ]
  }
}
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": "a716d75d4acd000894bb3adae663af097df86c30199a8f138eaee669335c420c",
    "addresses": [
      "BsWfitMMv4CTb8CTxRR4MxWbSYN2XfXb1h"
    ],
```

```
    "total": 8000,
    "fees": 1000,
    "size": 182,
    "vsize": 182,
    "preference": "low",
    "relayed_by": "163.123.192.135",
    "received": "2023-11-08T05:45:18.632182221Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "70ef4153be61b1eb370cbafd6605cb321204aa513a5ad7c0ed6ccaa107
        "output_index": 0,
        "script": "1874686973497341536563726574506173737 76f726431323347304402204
        "output_value": 9000,
        "sequence": 4294967295,
        "script_type": "unknown",
        "age": 1058749
      }
    ],
    "outputs": [
      {
        "value": 8000,
        "script": "76a914043ba15ba01903c723e06cbd147ab607970baf8e88ac",
        "addresses": [
          "BsWfitMMv4CTb8CTxRR4MxWbSYN2XfXb1h"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}
Bob redeem from swap tx (BTC) created successfully!
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "c9d0100fa85f964cd54b22d2a87338b8afbd6c4d4c1e05a0e38bea5917aee9d6",
    "addresses": [
      "monqWwckKskrYXDBto1a3zzL9ePxC9cFzx"
    ],
    "total": 9100,
    "fees": 1000,
```

```
    "size": 183,
    "vsize": 183,
    "preference": "low",
    "relayed_by": "163.123.192.136",
    "received": "2023-11-08T05:45:20.090604499Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "ce157e2cc849d0a040f2e36c0f0882a45f3a151f5c08080014b8b8e3a6
        "output_index": 0,
        "script": "18746869734973415365637265745061737377 6f726431323348304502210
        "output_value": 10100,
        "sequence": 4294967295,
        "script_type": "unknown",
        "age": 2537462
      }
    ],
    "outputs": [
      {
        "value": 9100,
        "script": "76a9145ac200fc39d372e019a5c26fb509e5f1eecdc36788ac",
        "addresses": [
          "monqWwckKskrYXDBto1a3zzL9ePxC9cFzx"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}
```


8. 完善swap_scripts.py

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。

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。

```
1
2  # This is the ScriptPubKey for the swap transaction
3  def coinExchangeScript(public_key_sender, public_key_recipient, hash_of_secret):
4      return [
5          # fill this in!
6          public_key_recipient,#将接收者的公钥放在栈上
7          OP_CHECKSIG,#验证解锁脚本是否包含接受者签名
8          OP_DROP,#丢弃栈顶的true,继续下一步
9          OP_DUP,#复制栈顶的secret或sig_sender（秘密或发送者的签名）进行后续判断
10         public_key_sender,#发送者的公钥放在栈上
11         OP_CHECKSIG,#是否为发送者签名
12         OP_IF,#引入条件块，如果是
13         OP_DROP,# 清空站内元素
14         OP_1,#推送 OP_1（True）
15         OP_ELSE,#如果不是
16         OP_HASH160,#使用 OP_HASH160 hash栈顶的secret
17         hash_of_secret,#将其与给定的 hash_of_secret 进行比较(OP_EQUAL)
18         OP_EQUAL,#该元素是不是秘密x
19         OP_ENDIF
20     ]
21
22 # This is the ScriptSig that the receiver will use to redeem coins
23 #接收者在知道秘密 x 时赎回币的 ScriptSig
24 def coinExchangeScriptSig1(sig_recipient, secret):
25     return [
26         # fill this in!
27         secret,# 秘密 x
28         sig_recipient# 接收者的签名
29     ]
30
31 # This is the ScriptSig for sending coins back to the sender if unredeemed
32 #将币发送回发送者的 ScriptSig，如果未被赎回
33 def coinExchangeScriptSig2(sig_sender, sig_recipient):
34     return [
35         # fill this in!
36         sig_sender,#发送者的签名
37         sig_recipient#接收者的签名
38     ]
39
```

问题

1. 解释代码内容，以及 coinExchangeScript 是如何工作的。

```
1
2  # This is the ScriptPubKey for the swap transaction
3  def coinExchangeScript(public_key_sender, public_key_recipient, hash_of_secret):
4      return [
5          # fill this in!
6          public_key_recipient,#将接收者的公钥放在栈上
7          OP_CHECKSIG,#验证解锁脚本是否包含接受者签名
8
9          OP_DROP,#丢弃栈顶的true,继续下一步
10         OP_DUP,#复制栈顶的secret或sig_sender（秘密或发送者的签名）进行后续判断
11         public_key_sender,#发送者的公钥放在栈上
12         OP_CHECKSIG,#验证是否是发送者签名
13
14         OP_IF,#开始条件块，如果是
15         OP_DROP,# 清空栈顶元素（此时是True），继续下一步
16         OP_1,# 推送数字1到栈上，表示条件为真
17
18         OP_ELSE,#如果不是（Bob 未赎回）
19         OP_HASH160,#使用 OP_HASH160 hash栈顶的secret，对栈顶的秘密进行哈希运算
20         hash_of_secret,#放置预定的hash_of_secret哈希值
21         OP_EQUAL,#检查是否与预定哈希值匹配，即该元素是不是秘密x
22
23         OP_ENDIF#结束条件部分
24     ]
25
26 # This is the ScriptSig that the receiver will use to redeem coins
27 #接收者在知道秘密 x 时赎回币的 ScriptSig
28 def coinExchangeScriptSig1(sig_recipient, secret):
29     return [
30         # fill this in!
31         secret,# 推送秘密x到栈
32         sig_recipient# 推送接收者的签名到栈
33     ]
34
35 # This is the ScriptSig for sending coins back to the sender if unredeemed
36 #将币发送回发送者的 ScriptSig，如果未被赎回
37 def coinExchangeScriptSig2(sig_sender, sig_recipient):
38     return [
39         # fill this in!
40         sig_sender, # 推送发送者的签名到栈
41         sig_recipient# 推送接收者的签名到栈
42     ]
43
```

2. 以 Alice 用 coinExchangeScript 向 Bob 发送硬币为例:如果 Bob 不把钱赎回来，Alice 为什么总能拿回她的钱?为什么不能用简单的 1/2 multisig 来解决这个问题?

```
Q1:
coinExchangeScript 中使用了条件分支（OP_IF 和 OP_ELSE），允许 Alice 在两种情况下都能够取回她的资金：
Bob赎回：如果 Bob提供了秘密x并签署了交易，Alice可以在coinExchangeScript中的条件块中第一个分支取回资金。这是因为条件块允许在Bob签署的情况下Alice取回资金。
未赎回：如果 Bob没有在合理时间内提供秘密 x 或未签署交易，Alice可以在coinExchangeScript中的条件块的第二个分支中取回资金。条件块允许在未赎回的情况下Alice取回资金，只需要她自己的签名，不需要Bob的签名。
```

```
Q2:
1/2 多重签名通常由两个签名密钥（或公钥）中的一个来控制资金，需要两方都签署交易才能花费资金。
1/2多重签名对于一方来说需要依赖另一方的合作，而在跨链原子交换中，其中一方不一定可信。如果Bob不愿意签署，Alice就不能取回资金，即使她有合法的理由。这在跨链原子交换中可能会引发信任问题，因为双方不一定始终可信。
跨链原子交换通常旨在减少信任和依赖合作。如果Alice和Bob不互相信任或出于安全原因不愿意共享秘密，那么使用 1/2 多重签名可能会变得困难。
```

3. 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。

```
Alice:

Alice 创建了一个原子交换的交易（alice_swap_tx），Alice指定了txid_to_spend（Alice 的 BTC UTXO 的交易 ID）、utxo_index（Alice 的 BTC UTXO 的输出索引）、amount_to_send（Alice 想要发送的金额）作为参数。该交易的输出包含一个脚本（alice_swap_scriptPubKey），要求接受者提供x，使得 hash(x) 等于她的秘密密码的哈希，同时需要提供有效的签名才能取走资产。这个交易中，Alice 的 BTC 被锁定，等待 Bob 完成后才能被取走。

Alice 创建了一个返回交易（alice_return_coins_tx），该交易允许她在 48 小时后从原子交换中取回 BTC。这个交易要求 Alice 提供有效的签名。函数的参数为amount_to_send（Alice 想要发送的金额）、last_tx（上一个交易的信息）、lock_time（交易的锁定时间，即必须等待一定数量的区块确认后才能取走资产）、script（脚本，定义了取回资产的条件，这个脚本由 P2PKH_scriptPubKey 生成），但 Alice 将资产发送到她自己的地址。

complete_return_tx 函数是 Alice 用于在原子交换完成后立即取回资产的工具，她需要提供 Bob 的签名。
Alice 可以选择在原子交换完成后，即在Bob向她提供了x并签署了相应的原子交换交易之后，直接取回她的资产。在这种情况下，她不需要等待时间锁定的到期，而可以立即取回资产。为了提前取回资
```

产，Alice 需要签署一笔返回交易（return transaction）。为了完成这笔交易，她需要提供一个有效的签名，以证明她有资格取回这些资产。

redeem_swap 函数则是 Alice 用于在交换开始后的一段时间内（根据时间锁定条件）提前取回资产的工具，
她需要提供有效签名和正确的秘密 x。Alice 也可以选择在交换开始后的一段时间内，提前取回资产，即在交换尚未完成前。这就是为什么返回交易具有时间锁定条件的原因。如果 Alice 决定在提前取回资产，她可以使用 redeem_swap 函数。这个函数会帮助她创建一笔取回比特币的交易。这笔交易要求她提供有效的签名以及秘密 x。Alice 需要提供有效签名以证明她有资格取回资产，同时也需要提供秘密 x。秘密 x 在整个原子交换过程中一直保密，直到 Alice 选择提前取回资产或交换成功后才会被公开。提供正确的 x 是取回资产的关键。

Alice 使用 broadcast_BTC 函数将交易广播到比特币 Testnet3 网络中，从而使交易被网络确认。这笔交易中包含了锁定条件，要求 Bob 提供有效签名和 x 的有效签名。这确保了交易的完成。

Bob:
Bob 创建了一个原子交换的交易（bob_swap_tx），该交易的输出包含一个脚本，要求接受者提供 x 以及有效的签名。这个交易的目的是将Bob 的 BCY 被锁定，等待 Alice 完成后才能被取走。参数与 alice_swap_tx 类似，包括 txid_to_spend、utxo_index 和 amount_to_send。

Bob 创建了一个返回交易（bob_return_coins_tx），允许他在 24 小时后从原子交换中取回 BCY。这个交易同样要求 Bob 提供有效的签名。参数包括 amount_to_send、last_tx、lock_time 和 script，类似于 alice_return_coins_tx，但 Bob 将资产发送到他自己的地址。

complete_return_tx 函数:
如果 Bob 决定在原子交换开始后的一段时间内提前取回 BCY，他可以使用 complete_return_tx 函数来创建一笔已签名的返回交易。这笔交易允许他提前取回资产，无需等待时间锁定的到期。为了提前取回资产，Bob需要签署这笔返回交易，并提供有效的签名（bob_signature_BCY）。签名证明了他有资格取回这些资产。Bob 可以在交换完成后或一段时间后取回资产，具体取决于情况。如果他决定在交换完成后取回资产，他需要使用redeem_swap函数来创建一个取回 BCY 的交易，其中他提供了有效的签名和秘密 x。

redeem_swap 函数:
Bob 有两种选择来取回 BCY。如果 Bob 决定在交换完成后立即取回 BCY，他可以使用 redeem_swap 函数。这个函数帮助他创建一笔取回 BCY 的交易。在这种情况下，他需要提供有效的签名（bob_signature_BCY），以证明他有资格取回资产。如果 Bob 选择在交换开始后的一段时间内取回 BCY，他可以使用 redeem_swap 函数来创建另一笔取回 BCY 的交易。这次，他需要提供有效签名和秘密 x，以满足时间锁定条件。

Bob 使用 broadcast_BTC 函数将交易广播到BCY Testnet网络中，从而使交易被网络确认。这笔交易中包含了锁定条件，要求 Alice 提供有效签名和 x 的有效签名。

如果 Alice 或 Bob 想要取回资产，他们可以等待到时间锁定过后，然后广播相应的取回交易。这将

```
允许他们取回资产，而不需要提供 x。

如果 Alice 想要取回资产，她可以提供 x 并广播 BCY 上的取回交易。这将公开 x。

如果 Bob 想要取回资产，他可以提供 x 并广播 BTC 上的取回交易。这将公开 x。

在任何情况下，双方都可以取回资产，但只有在满足条件和时间锁定后才能这样做。这确保了原子交换的安全性，资产要么全部成功交换，要么全部失败，不会出现资产被一方提前取回的情况。
```

```
设计原理：
原子交换的基本原理是通过锁定资产并要求对方提供某个秘密 x 来实现资产交换。
原子交换的核心思想是分别创建多个交易，其中一个包含锁定资产的脚本，另一个包含返回资产的脚本。这两个交易都要求对方提供 x 和有效的签名来取走资产。
Alice 和 Bob 在不知道 x 的情况下创建了原子交换交易，只有在其中一方提供 x 后，另一方才能取走资产。这种方式保证了资产的安全性，因为只有当交换完成时，资产才会离开原始所有者的控制。
使用不同的锁定时间（locktime）来确保资产的安全，即资产在交换开始后的一定时间内无法被取走，从而为双方提供了足够的时间来完成交换或取回资产。
通过使用比特币的脚本语言来创建脚本，可以定义交易的条件和规则，从而实现复杂的交易逻辑。这是原子交换的关键。
```

4. 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？

```
一次成功的跨链原子交换

Alice 拥有 BTC（Bitcoin）在 Testnet3，Bob 拥有 BCY（BlockCypher's Bitcoin）在 BCY Testnet。
Alice 创建了一笔交易（alice_swap_tx），其中她发送了一定数量的 BTC 到一个多重签名脚本，该脚本要求使用她的公钥和 Bob 的公钥签署交易才能取回资产。这笔交易的解锁脚本包含了一个哈希值，对应她保密的秘密 x。
Bob 创建了一笔交易（bob_swap_tx），将一定数量的 BCY 发送到一个多重签名脚本，要求使用 Alice 的公钥和 Bob 的公钥签署交易才能取回资产。解锁脚本中也包含了与 Alice 保密秘密 x 对应的哈希值。
Alice 和 Bob 签署各自的交易，但他们不会广播这些交易，因为还没有揭示秘密 x。
Alice 决定在一定时间后（48小时）取回 BCY，她创建了一笔已签名的交易（alice_return_coins_tx），其中包含 Bob 的签名。这笔交易允许 Alice 取回 BCY。
Bob 也可以在一定时间后（24小时）取回 BTC，他创建了一笔已签名的交易（bob_return_coins_tx），其中包含 Alice 的签名。这笔交易允许 Bob 取回 BTC。
一旦 Alice 决定取回 BCY，她会广播 alice_return_coins_tx，揭示了秘密 x。
一旦秘密 x 被揭示，Bob 可以取回 BTC，这时他会广播 bob_return_coins_tx。
```

一次失败的跨链原子交换：

如果一方没有按照约定的时间取回资产，另一方可以在超过约定时间后（24小时或48小时）取回对方未取回的资产。这是通过 complete_return_tx 函数实现的。

如果 Bob 没有在约定时间内取回 BCY，Alice 可以使用 complete_return_tx 创建一笔已签名的交易，以取回 BCY。

如果 Alice 没有在约定时间内取回 BTC，Bob 可以使用 complete_return_tx 创建一笔已签名的交易，以取回 BTC。

这些超时的交易允许资产返回给原始的所有者，且不需要对方的签名。