# 区块链基础及应用 LAB4 设计文档

- 姓名：卢麒萱 雷贺奥
- 学号：2010519 2013551

## 作业内容

- 解释你写的代码内容，以及 `coinExchangeScript` 是如何工作的。
  - 修改部分：

    在 `alice.py` 和 `bob.py` 中，赎回函数和交换函数中都使用到了 `GetTxid()` 函数，并使用 `b2x()` 函数将其进行格式的转换，但是由于网络编址大小端的问题，所以使得最终得到的 `transaction id` 不匹配，无法交易。将此函数改为 `b2lx()` 将大小端问题解决，顺利运行代码。
  - 准备工作：
    - 为 Alice 和 Bob 创建 BTC testnet 密钥，填入 `keys.py`。

```
# Only to be imported by alice.py
alice_secret_key_BCY = CBitcoinSecret.from_secret_bytes(
 x('79314503f75bd90ec333f8d02b9af77e7d4ea4ec5c078453cdab3a974865c412'))

# Only to be imported by bob.py
# Bob should have coins!!
bob_secret_key_BCY = CBitcoinSecret.from_secret_bytes(
 x('630e5e3a3d3c64eb65389a45ea8bfe602016599688880f3e452f9e60f0ae4eae'))
```

    - 为 Alice 的 BTC 地址领取测试币，在 Blockcypher 测试网（BCY）上为 Bob 的 BCY 地址领取测试币。
    - 使用 `split_test_coins.py` 分别划分 Alice 和 Bob 领取的币。（以Bob为例）

```
my_private_key = CBitcoinSecret.from_secret_bytes(
x('630e5e3a3d3c64eb65389a45ea8bfe602016599688880f3e452f9e60f0ae4eae'))

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

amount_to_send = 0.01 - 0.0001 # amount of BTC in the output you're splitting minus fee
txid_to_spend = (
 '213a0f5950e1f92eaf27f74c1ff4645c013d38062a5e0dd1d01b2815beb8ae35')
utxo_index = 0
n = 10 # number of outputs to split the input into
network = 'bcy-test' # either 'btc-test3' or 'bcy-test'
```

- 填写`swap.py`中相关信息：

```
alice_txid_to_spend     = "b4912f62006bc990c436196b4cab5f64f1c75152bb2cde08d469d1e92c3cc830"
alice_utxo_index        = 0
alice_amount_to_send    = 0.00128463

bob_txid_to_spend       = "1a16aaa142fb8ec267357e3c92f2565dfb16337ec51596b983f3269579c772dc"
bob_utxo_index          = 0
bob_amount_to_send      = 0.00099

# Get current block height (for locktime) in 'height' parameter for each blockchain (and put it into swap.py):
#  curl https://api.blockcypher.com/v1/btc/test3
btc_test3_chain_height  = 2409581

#  curl https://api.blockcypher.com/v1/bcy/test
bcy_test_chain_height   = 571979

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

tx_fee = 0.0001

broadcast_transactions = True
alice_redeems = False
```

- 核心的代码文件调用关系是，`alice.py`和`bob.py`调用`swap_script.py`文件中的比特币脚本来实现原子交换的基本操作，`swap.py`文件调用`alice.py`和`bob.py`两个文件来实现原子交换。

  `coinExchangeScript`函数在`alice.py`和`bob.py`中被调用，用于创建`TX1`中的输出脚本。

  `swap_script.py`中核心代码及注释如下：

```
# This is the ScriptPubKey for the swap transaction
def coinExchangeScript(public_key_sender, public_key_recipient, hash_of_secret):
    return [
        # 首先检查是否包括接收者的签名，如果接收者的签名都没有，那么一定不对
        public_key_recipient,
        OP_CHECKSIGVERIFY,
        # 对于剩余的元素进行复制，从而检测其是否为secret或者是另一个签名
        OP_DUP,
        OP_HASH160,
        hash_of_secret,
        OP_EQUAL,
        # 如果不是秘密，检查是否为自己的签名
```

```
        OP_NOTIF,
        public_key_sender,
        OP_CHECKSIG,
        OP_ELSE,
        OP_DROP,
        OP_TRUE,
        OP_ENDIF
    ]

# This is the ScriptSig that the receiver will use to redeem coins
def coinExchangeScriptSig1(sig_recipient, secret):
    return [
        # 交换，输入secret和签名
        secret,
        sig_recipient
    ]

# This is the ScriptSig for sending coins back to the sender if
unredeemed
def coinExchangeScriptSig2(sig_sender, sig_recipient):
    return [
        # 赎回，两方的签名
        sig_sender,
        sig_recipient
    ]
```

- 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？
  - 只有当 Bob 将 `TX2` 签名后，Alice 才会公布 `TX3`，在 Bob 没给 `TX2` 签名之前，一切都没有 `broadcast`，相当于 Alice 的钱根本没有花出去；当 Bob 签名后，Alice就拥有了通过 `TX2` 赎回钱的能力，但此时脚本还是锁定状态，还没有办法赎回。但是一旦时间超时，Alice 就可以直接赎回自己的钱。
  - 为了实现既能对方得到，又能自己赎回而使用 `1/2 multisig`，在两方都是可信的状态下是可以实现的。但由于最初始的模型建立中就将交换货币的两方视为不可信的，如果使用 `1/2 multisig`，那么每一方都有同时兑换回两笔交易的能力，失去了公平性，也不具备操作的原子性了。
- 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。

  在 `swap.py` 中可以得知交易内容及先后次序。
  - Alice 创建 `TX1`，`TX1` 满足只有输入 `秘密 x` 和 `Bob 的签名` 或者是 `Alice 的签名` 和 `Bob 的签名` 时才能兑换此笔交易。

    > 此时只是准备好 `TX1`，并没有令该交易立即广播出去，因为 Alice 还没有获得 Bob 的签名，所以其一旦广播出去可能这笔钱就再也无法赎回。这个交易的目的是在后续能够顺利地兑换和赎回。

  - Alice 创建 `TX2`，`TX2` 可以用来赎回 `TX1` 的这笔钱。

    > 用 Alice 签名是为了之后 Alice 能够顺利花费这笔钱，将这笔交易锁定是为了令 Alice 过一段时间才能执行这个赎回脚本，给了 Bob 充足的时间去兑换这笔交易。

  - Bob 对此笔交易进行签名，Alice 获得 Bob 的签名，Alice 把 `TX1` 广播出去。

> 如果 Bob 不签名，那么前面的消息还都没有广播，所以并不会产生任何实质的影响。当 Bob 签名以后，Alice 利用自己的签名和 Bob 的签名，就拥有了自己赎回货币的能力，如果当赎回脚本解锁的时候，Bob 还未进行兑换，那么 Alice 可以直接赎回这笔钱。 然后 Alice 把 `TX1` 广播出去， `TX1`被广播出去后就具备了不可篡改性，这笔钱就只能是 Bob 兑换走或当赎回脚本锁定结束后 Alice 将其赎回。

- Bob 创建类似意义的交易 `TX3`。

> Bob 创建 `TX3` 与 Alice 相同，除了依靠两方的签名 Bob 可以将其赎回以外，这笔交易还可以通过 `Alice 的secret` 和 `Alice 的签名` 直接兑换，也就是说一旦这个交易广播出去，Alice 可以直接兑换此交易，所以暂时不广播。

- Bob 创建 `TX4` ， `TX4` 可以用来赎回 `TX3` 的这笔钱。

> Bob 创建 `TX4`与 `TX2` 的意义相同，创建`TX4` 后，Bob 就具备了赎回自己交易的能力，如果 Alice 不签名，那么 Bob 也不公开 `TX3`，对于 Bob 而言自己的货币交易并没有写入区块链所以不会有损失。

- Alice 对此笔交易进行签名，Bob 获得 Alice 的签名，Bob 将 `TX3` 广播出去。

> 一旦广播出去后，Alice 知道`secret`和自己的签名，所以其可以轻松兑换货币，一旦 Alice 兑换货币，其`secret`也将泄露出来，Bob 得知了`secret`后，Bob 也能够轻松兑换 `TX1`，至此两方成功完成兑换。Alice 若不兑换，超时后两方都可以赎回交易。

- 本次作业中，一次成功的跨链原子交换中，资金是如何流转的?
  - Alice 先创建交易 `TX1`，生成 `secret x`，其用于交换的资金还未公布，锁定在 Alice 的输出脚本中。Bob 创建交易 `TX2`，使用同样的 `Hash(x)` 将自己的 BCY 锁定在输出脚本中。
  - Alice 通过生成的 `x` 来 解锁 Bob 的输出脚本，在获得 `TX2` 的签名后，此交易被公布到了链上，此时这笔资金不属于任何人。
  - Bob 同理，创建 `TX3`时并没有资金流动，将 `TX3`公布后，Alice 可以直接兑换此笔交易，资金可以流转到 Alice 手中。Alice 兑换后，Bob就能得知`secret`，在 `TX2`解锁时间没到时，`TX1`这笔钱属于 Bob，及时转移，资金会成功流转到 Bob 手中。通过这种方式，Bob 的 BCY 流转到 Alice，而 Alice 的 BTC 流转给 Bob。
  - 如果双方都没有兑换，两笔交易超时后资金都会流回自己的账户。

## 交易信息

- 基本信息

```
2010519
alice:
Private key: cN7JSzNEjxerSZDa68pgTXA4ZeBPX5ViHKKi7e6jLapeKvHHGg3s
Address: mj2BraQyDZbsFMAFddCfrKay3aguWhgk1N
{
  "private":
"79314503f75bd90ec333f8d02b9af77e7d4ea4ec5c078453cdab3a974865c412",
  "public":
"03b337bb013b4742699562de2c7ed4e3feb3fcb0cba352de54f58e11f2bc52bdcd",
  "address": "C6im25EWC5K1H9f86cJgSZJeQycjtV9Sno",
  "wif": "BsPcWer5XqVQm7RsPdCqbLjdwqDy3UjBCxCRtquursGhjzTH8VFN"
}#
bob:
Private key: cV4w7wALiiHKFyvJSrX6j9fLQ23nLTyYdNwApmR9oS6qnKtSvsNg
```

```
Address: mgx4NHaSJieGQEBg2kiBbjQ2914Pn9vNkW
{
  "private":
"630e5e3a3d3c64eb65389a45ea8bfe6020165996888880f3e452f9e60f0ae4eae",
  "public":
"0212bd91133fa873eb6cd7120de2855c21c8b9729cc3bbbad36d7b8006bc4e25f7",
  "address": "C4yKg3LTW8eTSVueDK8o7qunyYMdJRfHXo",
  "wif": "BreamG2acfMGq2XT3bPwf8SBYboZ5ZLAUp8CGkMi1jrufuPNvHSm"
}#
ef15bf4c473587
{
  "tx_ref":
"28d63c3e5746b17d5e3a02df4d130ce9fff06b8195a5934c40c0d98c998d36b6"
}#

2013551
alice:
Private key: cSvCXVBrB8rKGgRiCHkAzYj4LA4d9KjxPo9YceetiAtMZMSy3Jp7
Address: myE4U3c79CnNkdjGx1EANQLiC5Ck2tDiq6
{
"private":
"434343384cee8a506c06235c743133f487ae1f96295d2f24de66af461de68781",
"public":
"02c49f69a5e864d7942b8d103681ad83980c560947d549e0958c5bbd8f7b913c9e",
"address": "C8CT2woapDiDJjTG8SLNc3bLMu38jqv6Bc",
"wif": "BqanEjWiJDHWJgnsc22sCwK8MiNWXkhWTyZz6xd9JTMp3EP8ZmXY"
}#
bob:
Private key: cSPyXDMAoMb7Vta1otMaMtDRrtbpTcyZ2mpTQ6LWibj8z7GcaiZW
Address: mq87epoGxXd8NL6jdoLHLW7v7yZ7CRF3u8
{
"private":
"19522dc2c7dc804ccf46466ffb86edc62c3351448ac15d18a8f9f28e54552c08",
"public":
"03a7f7cdef6785af38c3e1ef0b7603bbdf1fa9ea424186ab8291f77fdfe711823a",
"address": "C768DzwH2G8CjP8e5pVmBJ4pAeQS1LNGDt",
"wif": "BpBFYLZtyXmLG7VwhJPPdrC624z3r8PxMe5X5iZYWa5AyrPt7nwM"
}#
{
"tx_ref":
"a13cc75493da052d873f030efaaa94f23ae968eef46749b6b1465bd9b5d9e6e6"
}#
```

- 本地验证

`swap.py`中使用：

```
broadcast_transactions=False
alice_redeems = False
```

在本地进行验证：

```
┌─root at DESKTOP-IKOECBQ in ~/src/blockchain/Ex4 22-12-03 - 15:57:12
└─o python swap.py
Alice swap tx (BTC) created successfully!
Bob swap tx (BCY) created successfully!
Bob return coins (BCY) tx created successfully!
Alice return coins tx (BTC) created successfully!
```

`swap.py`中使用：

```
broadcast_transactions=False
alice_redeems = True
```

在本地进行验证：

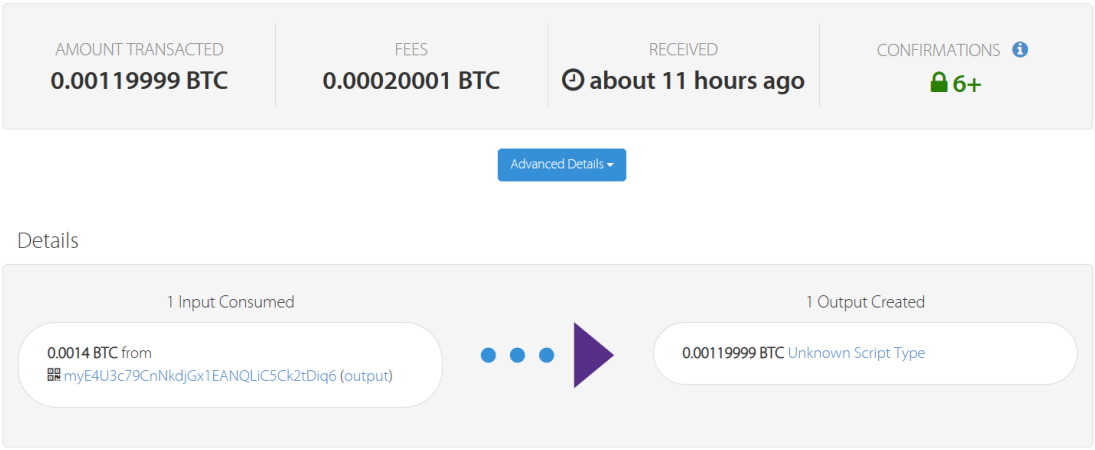
```
┌─root at DESKTOP-IKOECBQ in ~/src/blockchain/Ex4 22-12-03 - 16:27
:00
└─o python swap1.py
Alice swap tx (BTC) created successfully!
Bob swap tx (BCY) created successfully!
Alice redeem from swap tx (BCY) created successfully!
Bob redeem from swap tx (BTC) created successfully!
```

- 广播验证交换

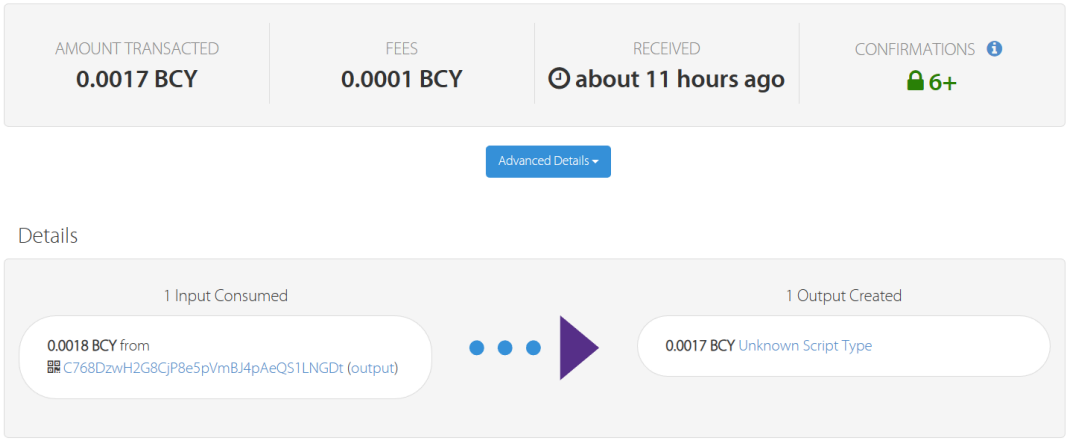
  `swap.py`中使用：

  ```
  broadcast_transactions=True
  alice_redeems = False
  ```

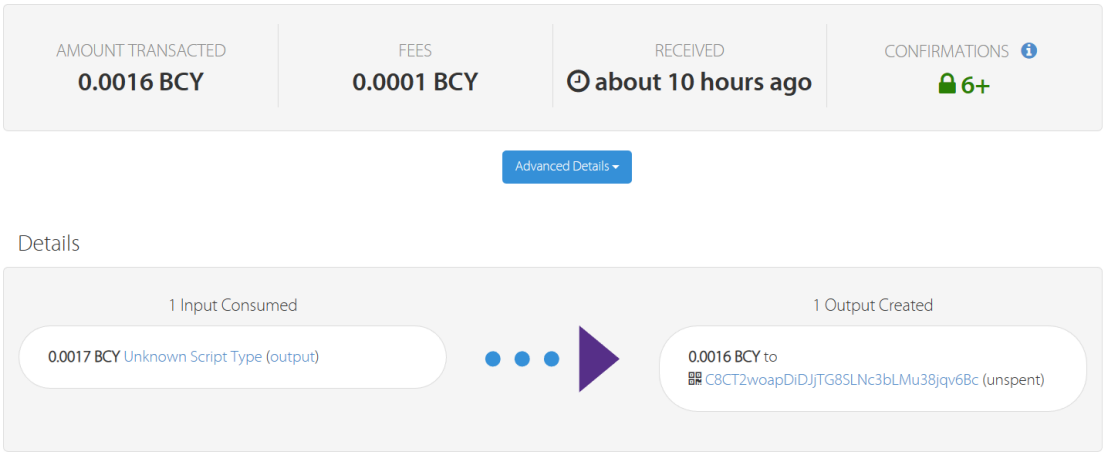
  Alice 创建`TX1`：

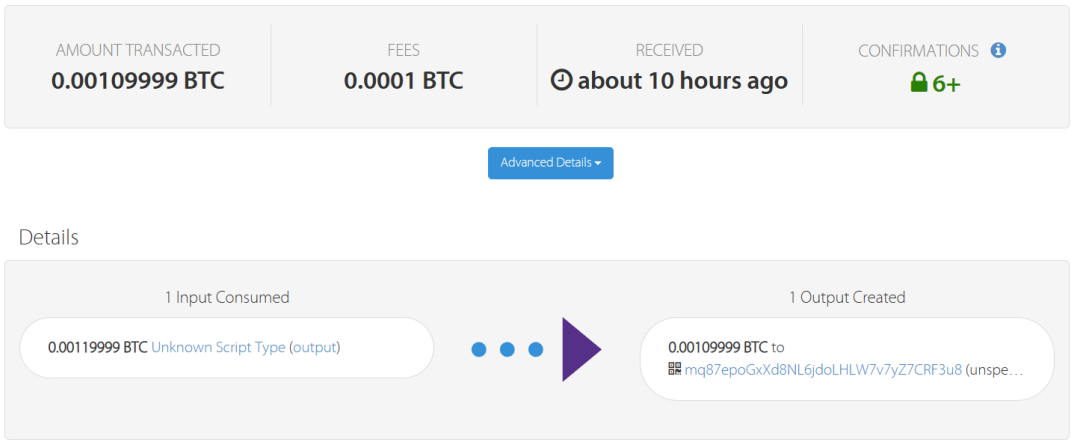
  Bob 创建`TX3`：

Alice 的交换，`TX3`中的钱转移到 Alice 的 BCY 账户：




Bob 的交换，`TX1`中的钱转移到 Bob的 BTC 账户：

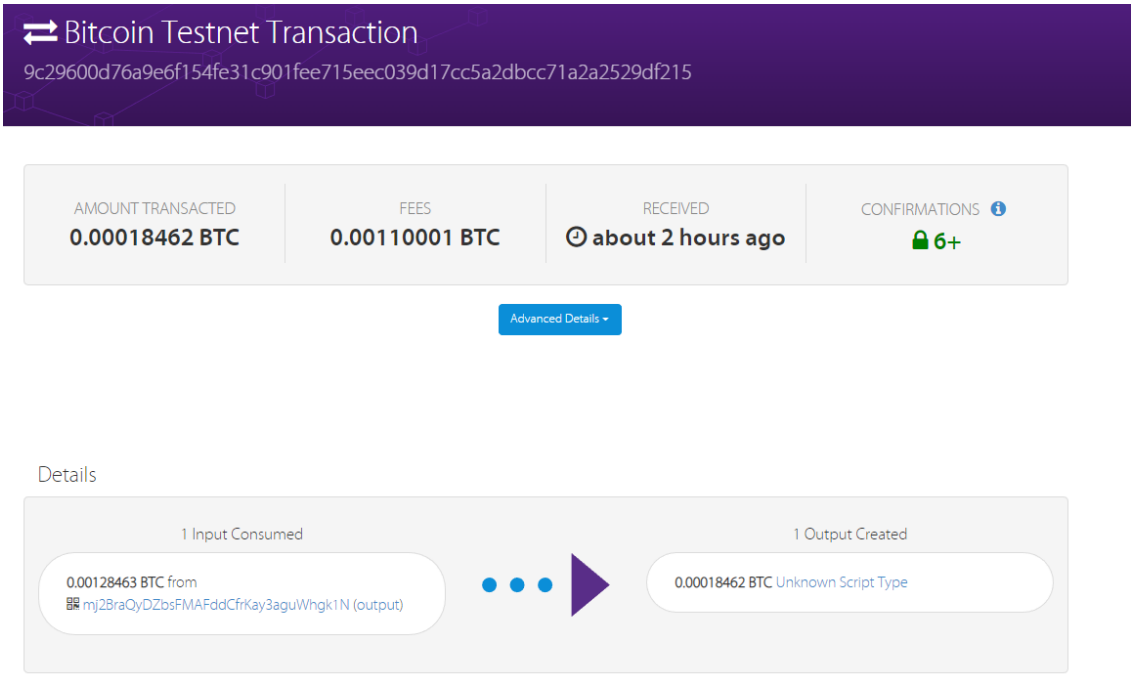



- 广播验证赎回

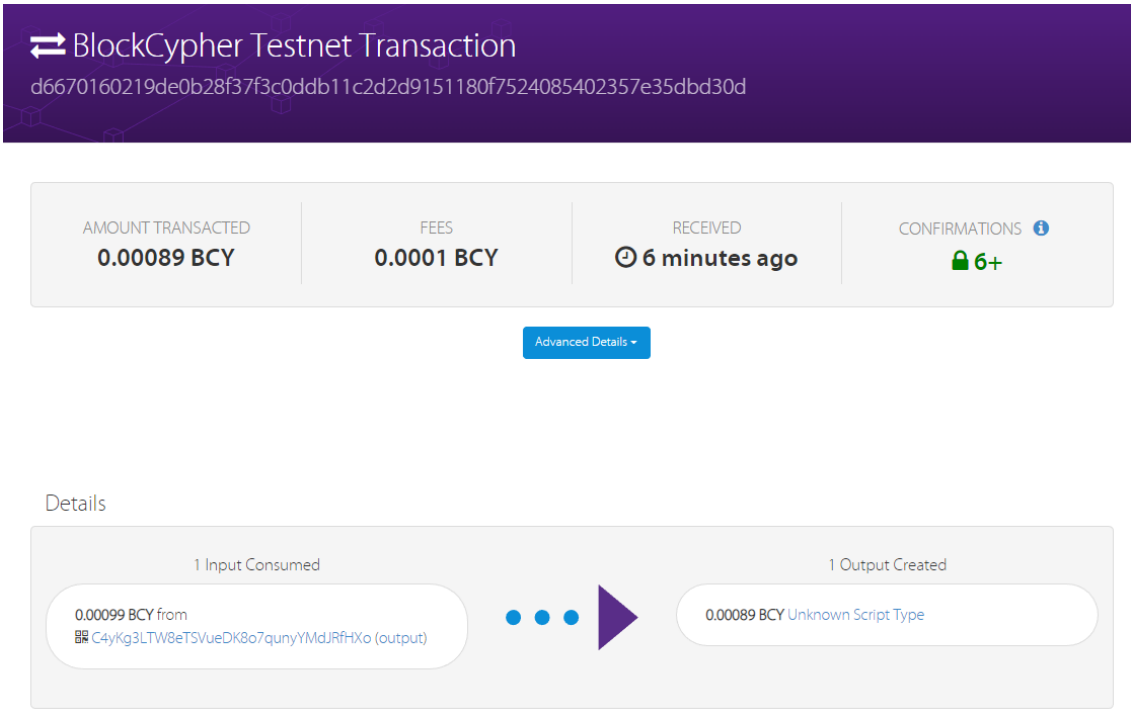
  `swap.py`中使用：

```
broadcast_transactions=True
alice_redeems = True
```

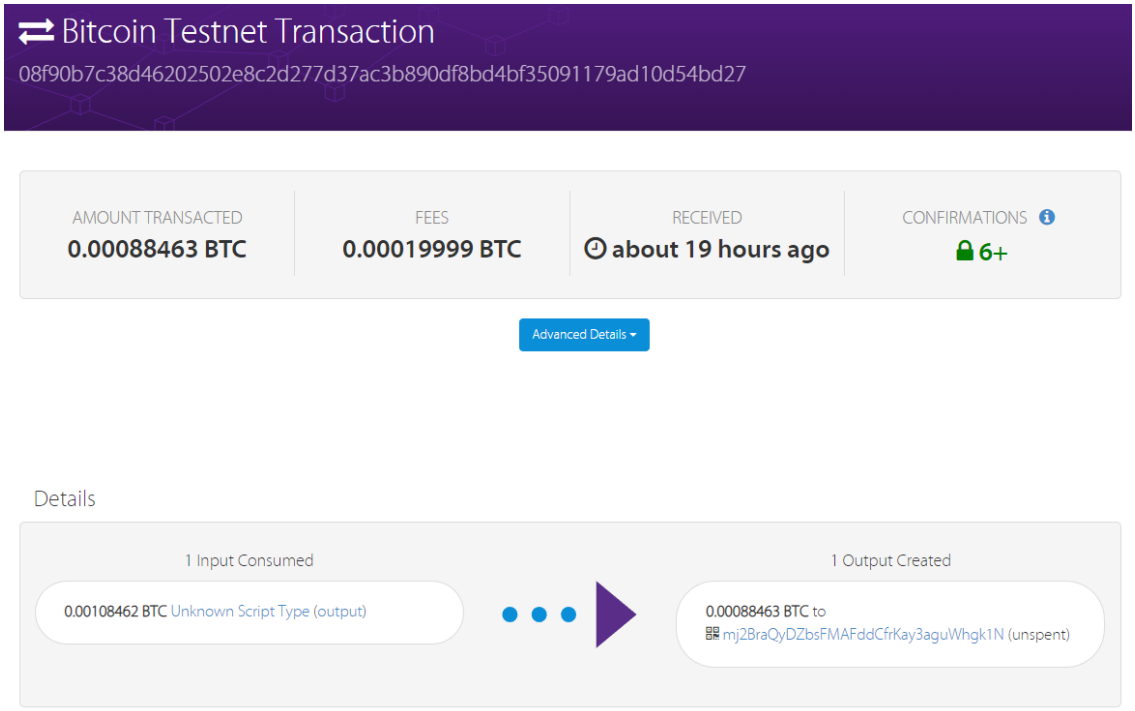
Alice 创建`TX1`：




Bob 创建 `TX3`：




Alice 赎回`TX3`，并且将自己的secret交给Bob，在Alice的BCY地址中查看赎回交易：




Bob通过Alice给的secret 赎回 `TX1`，在Bob的BTC地址中查看赎回交易：