

به نام خدا



گزارش کار پروژه اول رمز ارز

شماره دانشجویی:  
۸۱۰۱۹۹۵۵۴

نام و نام خانوادگی: دانشکده مهندسی برق و کامپیوتر  
سینا طبسی

## قسمت اول

### ۱. تولید آدرس و کلید خصوصی به فرم WIF:

ابتدا کلید خصوصی رندوم ۳۲ بایتی را با استفاده از `secret.tokens.hex` تولید می کنیم. در ادامه در تابع `address_generator` به ترتیب عملیات های زیر را انجام می دهیم:  
۱- تولید کلید عمومی شده و سپس `comperess` کردن آن به صورت `byte` که این کلید عمومی در تابع `public_key_compressed` تولید می شود. عکس این تابع در ادامه آورده شده است:

```
def public_key_generator(private_key):  
    private_key_bytes = bytes.fromhex(private_key)  
    public_key_bytes = ecdsa.SigningKey.from_string(private_key_bytes, curve=ecdsa.SECP256k1).verifying_key  
    return public_key_bytes.to_string().hex()
```

```
def public_key_compressed(random_private_key):  
    public_key_compressed = '04' + public_key_generator(random_private_key)  
    public_key_compressed_bytes = bytes.fromhex(public_key_compressed)  
    return public_key_compressed_bytes
```

۲- در ادامه ابتدا این کلید عمومی تولید شده را با استفاده از تابع رمزنگاری درهم ساز `sha256` رمز کرده و سپس با استفاده از تابع رمزنگاری درهم ساز `ripemd_160` بار دیگر آن را رمز می کنیم. توابع و کد استفاده شده در این قسمت در ادامه قابل مشاهده است:

```
sha256_1 = hashlib.sha256(public_key_compressed_bytes).digest()
```

```
def ripemd_hash(sha256_1):  
    ripemd = hashlib.new('ripemd160')  
    ripemd.update(sha256_1)  
    ripemd_key = ripemd.hexdigest()  
    return b"\x6f".hex() + ripemd_key
```

در ادامه با اضافه کردن checksum و تبدیل آن به base58 ما تولید خواهد شد:

```
def checksum(key):  
    return hashlib.sha256(hashlib.sha256(key).digest()).digest()[0:4]
```

```
address = bytes.fromhex(rip_key_extended) + checksum(bytes.fromhex(rip_key_extended))  
  
final_base58 = base58.b58encode(address).decode('utf-8')
```

تمامی توابع بالا در تابع address\_generator قرار دارد. در نهایت نیز آدرس خصوصی خود را به فرم WIF در می آوریم. نمونه از تولید آدرس و تولید کلید خصوصی آورده شده است:

```
/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/generate_address.py  
Private Key: 73665cc24205c9f72ee495c40cf66605fb9ac583bcc6731fc7565de62b98305f  
address: mhjCYwJ4sH5VWx3bChaCiDcguFx2FfUhcG  
WIF private: 92Tju1etbeG49svUGf7S4i3DkwxT1Av8KjHKJr1APh9ZwcEkK1D
```

## 2. تولید آدرس Vanity:

این بخش مانند بخش قبلی است با این تفاوت که تابعی به اسم generate\_vanity\_address به صورت brute force به دنبال آدرس با سه char مشخص شده می گردد. با پیدا کردن آن را چاپ می نماید. کد و نمونه ای از اجرای این کد آورده شده است:

```
def generate_vanity_address(chars):  
    while True:  
        private_key = secrets.token_hex(32)  
        address = address_generator(private_key)  
        if address[1:4] == chars:  
            print('vanity private key:' + private_key)  
            return address
```

```
/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/generate_vanity_address.py  
vanity private key:88ee9fa8f04293f65052a5dabf10a88cbaf8340a7fbf2225e7150b0c1f16989d  
Vanity Address: msinRni7NQ35dXTYftF2dQyMGLq1kZu6s
```

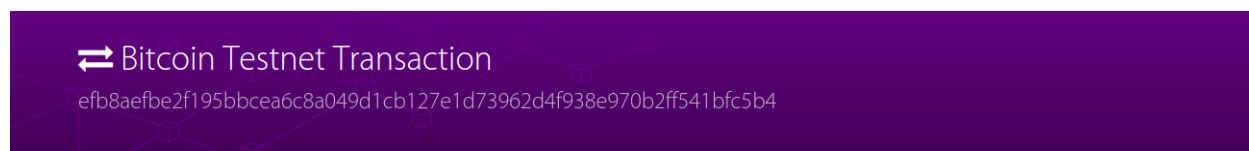
## قسمت دوم

در ابتدای این سوال ما کمی بیت کوین از faucet دریافت می کنیم. برای این کار باید آدرس عمومی P2PKh ساخته شده توسط کد قسمت اول را وارد می کنیم. با این کار یک transaction با آدرس داده شده ساخته شده و به آن آدرس بیت کوین فرستاده می شود. عکس این transaction در ادامه قابل مشاهده می باشد. همانطور مشاهده می شود مقدار بیت کوین آزمایشی برابر 0.01450576 می باشد:

TXID: efb8aefbe2f195bbcea6c8a049d1cb127e1d73962d4f938e970b2ff541bfc5b4

Address: msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ

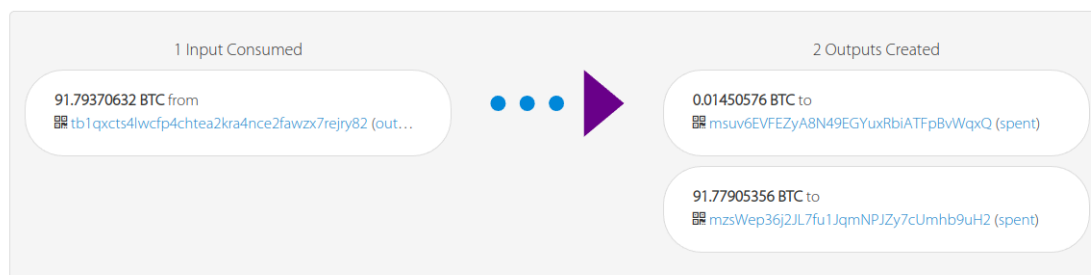
PrivateKey: 91wCkGpbGK42f3RXVUJ4utnewk3SoPnMDUq7JJPjXgZr6QA3MHb



AMOUNT TRANSACTED	FEES	RECEIVED	CONFIRMATIONS ⓘ
91.79355932 BTC	0.000147 BTC	⌚ about 2 hours ago	🔒 6+

Advanced Details ▾

### Details



در این transaction یک ورودی و دو خروجی خواهیم داشت. اولین خروجی مقدار بیت کوینی می باشد که به آدرس ما تعلق گرفته است. دومین خروجی مقدار اضافه ای است که به faucet برگردانده می شود.

سوال اول:

بخش اول

کد این بخش در ادامه آورده شده است:



در بخش int\_main مقدار پرداختی خروجی اول و مقدار خروجی مقدار دوم را مشخص کرده و سپس txid را مشخص می کنیم. txout\_scriptPubKey برای خروجی اول را برابر OP\_RETURN قرار می دهیم تا نتوان آن را خرج نمود و txout\_scriptPubKey برای خروجی دوم را برابر OP\_1 قرار می دهیم تا بتوان آن را خرج کرد. عکس نتیجه کد به همراه عکس transaction خواسته شده آورده شده است:

```
/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/transactionQ1P1.py
msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ
0451ae5229942b6107546db0e97c911ee29e869791437b5e35848714f38960d81e8a40ef94717277c0778342ba28f5523be231571f7cab83d2e9b6f71f3788a83b
2e0fca01abf6b94573762c857b4ccf4bd789bdb1ed9ab9ff7cc5289c2888643a
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "9bcfda8a676eaffd3859d1990d983b552e22b49c454560648ddf8d91ceee239e",
    "addresses": [
      "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
    ],
    "total": 1388100,
    "fees": 150476,
    "size": 210,
    "vsize": 210,
    "preference": "high",
    "relayed_by": "91.133.210.198",
    "received": "2023-05-27T12:02:02.770118788Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 2,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "efb8aefbe2f195b5bceac8a849d1cb127e1d73962d4f938e970b2ff541bfc5b4",
        "output_index": 0,
        "script": "483045022108974dcdb46710d989bfa684c9f2e73b4293bd67c7e5de80811f339d2ec3f718bf0228aefeb518ea9595e9fee6039d18cc5fd874115817a57cde3a2fbfc3fd84dbc44501410451ae5229942b6107546db0e97c911ee29e869791437b5e35848714f38960d81e8a40ef94717277c0778342ba28f5523be231571f7cab83d2e9b6f71f3788a83b",
        "output_value": 1458576,
        "sequence": 4294967295,
        "addresses": [
          "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 2435649
      }
    ],
    "outputs": [
      {
        "value": 100,
        "script": "6a",
        "addresses": null,
        "script_type": "null-data"
      },
      {
        "value": 1388000,
        "script": "51",
        "addresses": null,
        "script_type": "unknown"
      }
    ]
  }
}
```

**TXID:** 9bcfda8a676eaffd3859d1990d983b552e22b49c454560648ddf8d91ceee239e

**Address:** msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ

**PrivateKey:** 91wCkGpbGK42f3RXVUJ4utnewk3SoPnMDUq7JJPjXgZr6QA3MHb

9bcfda8a676eaffd3859d1990d983b552e22b49c454560648ddf8d91ceee239e

AMOUNT TRANSACTED

0.013001 BTC

FEES

0.00150476 BTC

RECEIVED

⌚ about 2 hours ago

CONFIRMATIONS ⓘ

🔒 6+

Advanced Details ▾

## Details

1 Input Consumed

0.01450576 BTC from  
msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ (output)



2 Outputs Created

0.000001 BTC Null Data Transaction

0.013 BTC Unknown Script Type

بخش دوم

کد این بخش در ادامه آورده شده است:

```
import bitcoin.wallet
from bitcoin.core import COIN, b2lx, serialize, x, lx, b2x
from utils import *

# 0.013001
# txid = 9bcfda8a676eaffd3859d1990d983b552e22b49c454560648ddf8d91ceee239e
# msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ
# 91wCkGpb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJXgZr6QA3MHb

bitcoin.SelectParams("testnet")
my_private_key = bitcoin.wallet.CBitcoinSecret("91wCkGpb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJXgZr6QA3MHb")
my_public_key = my_private_key.pub
my_address = bitcoin.wallet.P2PKHBitcoinAddress.from_pubkey(my_public_key)

1 usage
def P2PKH_scriptPubKey(address):
    return [OP_DUP, OP_HASH160, Hash160(my_public_key), OP_EQUALVERIFY, OP_CHECKSIG]

def P2PKH_scriptSig(txin, txout, txin_scriptPubKey):
    signature = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey, my_private_key)
    return [signature, my_public_key]

1 usage
def send_from_P2PKH_transaction(amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey):

    txout = create_txout(amount_to_send, txout_scriptPubKey)

    txin_scriptPubKey = [OP_1]
    txin = create_txin(txid_to_spend, utxo_index)
    txin_scriptSig = []

    new_tx = create_signed_transaction(txin, txout, txin_scriptPubKey,
                                       txin_scriptSig)

    return broadcast_transaction(new_tx)
```

```

if __name__ == '__main__':
    # 0.013001
    amount_to_send = 0.0128
    txid_to_spend = ('9bcfda8a676eaffd3859d1990d983b552e22b49c454560648ddf8d91ceee239e') # TxHash of UTXO
    utxo_index = 1

    print(my_address) # Prints your address in base58
    print(my_public_key.hex()) # Print your public key in hex
    print(my_private_key.hex()) # Print your private key in hex

    txout_scriptPubKey = P2PKH_scriptPubKey(my_address)
    response = send_from_P2PKH_transaction(amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey)

    print(response.status_code, response.reason)
    print(response.text) # Report the hash of transaction which is printed in this section result

    #transaction spent: "c9420f311b98fec5e98adee5c7e38954500629ea263a031e5142ca1170440f40"

```

کد بالا همانند بخش اول این سوال می باشد. با این تفاوت که باید transaction ای ایجاد کنیم تا مقدار اضافی transaction را به حساب خود برگردانیم.

در تابع `send_from_P2PKH_transaction` مانند قبل `txout` را ایجاد می کنیم و مقادیر `scriptPubKey` و `scriptSig` را برای `txin` کامل می کنیم. در نهایت `new_tx` را ایجاد کرده و آن را broadcast می کنیم. برگشت بیت کوین در ابتدای `int_main` آورده شده است.

در ادامه نتیجه کد و transaction آورده شده است:

```

/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/transactionQ1P2.py
msuv6EVFEZyABN49EGYuxRb1ATFpBVWqxQ
0451ae5229942b6107546db0e97c911ee29e869791437b5e35848714f38960d81e8a40ef94717277c0778342ba20f5523be231571f7cab83d2e9b6f71f3700a03b
2e0fca01abf6094573762c857b4ccf4bd789bdb1ed9ab9ff7cc5289c2088643a
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "c9420f311b98fec5e98adee5c7e38954500629ea263a031e5142ca1170440f40",
    "addresses": [
      "msuv6EVFEZyABN49EGYuxRb1ATFpBVWqxQ"
    ],
    "total": 1280000,
    "fees": 20000,
    "size": 85,
    "vsize": 85,
    "preference": "high",
    "relayed_by": "01.133.210.198",
    "received": "2023-05-27T12:21:16.742930454Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "9bcfda8a676eaffd3859d1990d983b552e22b49c454560648ddf8d91ceee239e",
        "output_index": 1,
        "output_value": 1300000,
        "sequence": 4294967295,
        "script_type": "unknown",
        "age": 2435651
      }
    ],

```

```

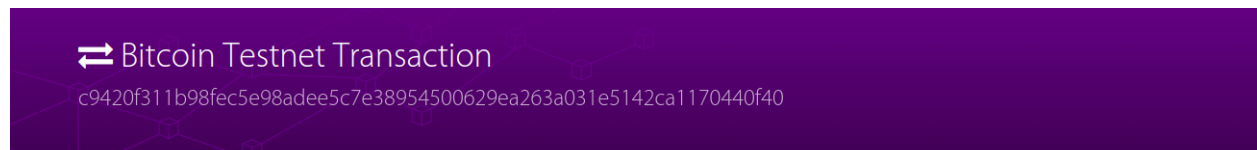
    "outputs": [
      {
        "value": 1280000,
        "script": "76a91487f93ca45d48079f8862a11bf7b7c1e7a46f8edc88ac",
        "addresses": [
          "msuv6EVFEZyABN49EGYuxRb1ATFpBVWqxQ"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}


```

**TXID:** c9420f311b98fec5e98adee5c7e38954500629ea263a031e5142ca1170440f40

**Address:** msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ

**PrivateKey:** 91wCkGpbGK42f3RXVUJ4utnewk3SoPnMDUq7JJPjXgZr6QA3MHb



AMOUNT TRANSACTED <b>0.0128 BTC</b>	FEES <b>0.0002 BTC</b>	RECEIVED <b>about 2 hours ago</b>	CONFIRMATIONS  <b>6+</b>
--	---------------------------	--------------------------------------	--

Advanced Details ▾

#### Details



## سوال دوم

### بخش اول

در این بخش ما سه کلید خصوصی جدید تولید کردیم تا بتوان با استفاده از کلید خصوصی و signature دو تا از این سه کلید خصوصی transaction مورد نظر که شامل یک ورودی و یک خروجی می شود را انجام داد. کد این قسمت به همراه سه کلید خصوصی جدید در ادامه آورده شده است:

First\_private\_key = 92UN41zMS2NiGy6gMaGER5iq5fwthC2uVY64eNAtmARCpDiyOMA

Second\_private\_key = 929D9AU2Vh8msPzdwtYEwo1c9LD8tgz7o8uo6Uq1HAstWsvwDec

Third\_private\_key = 91fuVHjXcY8kmXoao97BPVvCrLZMpcvy8MaXfGgKJgX4KjiuVtH



```

import bitcoin.wallet
from utils import *

# 0.0128
# txid = c9420f311b98fec5e98adee5c7e38954500629ea263a031e5142ca1170440f40
# msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ
# 91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJXgZr6QA3MHb

bitcoin.SelectParams("testnet")

my_private_key = bitcoin.wallet.CBitcoinSecret("91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJXgZr6QA3MHb")
first_private_key = bitcoin.wallet.CBitcoinSecret("92UN41zMS2N16y6gMa6ER5iq5fwtHC2uVY64eNAtmARcpDiyvMA")
second_private_key = bitcoin.wallet.CBitcoinSecret("929D9AU2Vh8msPzdwTYEwo1c9LD8tgz7o8Uo6Uq1HastWsvwDec")
third_private_key = bitcoin.wallet.CBitcoinSecret("91fuVHjXcY8kmXoao97BPVvCrLZMpcvy8MaXf6gK3gX4KjiuVtH")

my_public_key = my_private_key.pub
first_person_public_key = first_private_key.pub
second_person_public_key = second_private_key.pub
third_person_public_key = third_private_key.pub

1 usage
def P2PKH_txin_scriptPubKey():
    return [OP_DUP, OP_HASH160, Hash160(my_public_key), OP_EQUALVERIFY, OP_CHECKSIG]

2 usages
def P2PKH_scriptSig(txin, txout, txin_scriptPubKey):
    signature = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey, my_private_key)
    return [signature, my_public_key]

```

```

def make_multisig_transaction(amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey):

    txout = create_txout(amount_to_send, txout_scriptPubKey)

    txin_scriptPubKey = P2PKH_txin_scriptPubKey()

    txin = create_txin(txid_to_spend, utxo_index)

    txin_scriptSig = P2PKH_scriptSig(txin, txout, txin_scriptPubKey)

    new_tx = create_signed_transaction(txin, txout, txin_scriptPubKey, txin_scriptSig)

    return broadcast_transaction(new_tx)

if __name__ == '__main__':
    # 0.0128
    amount_to_send = 0.0124
    txid_to_spend = ('c9420f311b98fec5e98adee5c7e38954500629ea263a031e5142ca1170440f40')
    utxo_index = 0

    txout_scriptPubKey = [OP_2, first_person_public_key, second_person_public_key, third_person_public_key, OP_3, OP_CHECKMULTISIG]
    response = make_multisig_transaction(amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey)
    print(response.status_code, response.reason)
    print(response.text)

    # result = '6863e1ba33f20e3dcb8286dc09999ca019e64595645768826c6f3424d212877e'

```

در کد بالا قسمت P2PKH\_txin\_scriptPubKey و P2PKH\_scriptSig مانند قسمت قبل می باشد. تابع make\_multisig\_transaction نیز مانند تابع make\_transaction می باشد و نیاز به توضیح ندارد. تفاوت این قسمت در txout\_scriptPubKey می باشد که در main آورده شده است. در واقع با عملیات OP\_2 دو کلید خصوصی از سه کلید عمومی که با OP\_3 در استک قرار دارند را انتخاب کرده و با دستور OP\_CHECKMULTISIG آن دو را بررسی و در صورت درست بودن مقدار True را برمی گرداند و در صورت غلط بودن مقدار False را برمی گرداند.

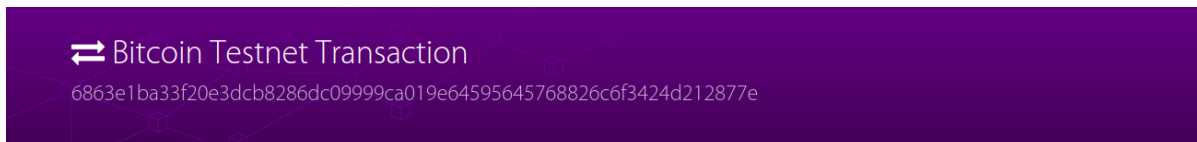
نتیجه کد به همراه transaction ایجاد شده در ادامه آورده شده است:

```
/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/transactionQ2P1.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "6863e1ba33f20e3dcb8286dc09999ca019e64595645768826c6f3424d212877e",
    "addresses": [
      "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ",
      "zG9mCjVbdYomasQwZk3xoN2uRMDa1HVo1z"
    ],
    "total": 1240000,
    "fees": 40000,
    "size": 399,
    "vsize": 399,
    "preference": "high",
    "relayed_by": "188.118.96.41",
    "received": "2023-05-27T15:38:32.920749842Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "c9420f311b98fec5e98adee5c7e38954508629ea263a031e5142ca1170440f40",
        "output_index": 0,
        "script": "473044022000f5e27476388f5e49c0ec53ade3e0cc0abe118dc39328f8e69f413accbe32180220269b2e5bad4f23b05ae011422c346e5af2e1484a45f903f505770505757967d901410451ae5229942b6107546db0e97c911ee29e869791437b5e358a",
        "output_value": 1280000,
        "sequence": 4294967295,
        "addresses": [
          "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 2435653
      }
    ],
    "outputs": [
      {
        "value": 1240000,
        "script": "5241040ba5b011a805097504ea1bc35d9b61b7e4477741038ced0df5a7853b8b964b96d54b5b9d6781d0f5f175af8ea6432832173f8d6b2dc629555e3bf7f8c0d18f84104b378335a178730e91d148fb8a5f7e31b7734db882d75df09ec29b207232b",
        "addresses": [
          "zG9mCjVbdYomasQwZk3xoN2uRMDa1HVo1z"
        ],
        "script_type": "pay-to-multi-pubkey-hash"
      }
    ]
  }
}
```

TXID: 6863e1ba33f20e3dcb8286dc09999ca019e64595645768826c6f3424d212877e

Address: msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ

PrivateKey: 91wCkGpbGK42f3RXVUJ4utnewk3SoPnMDUq7JJPjXgZr6QA3MHb



AMOUNT TRANSACTED <b>0.0124 BTC</b>	FEES <b>0.0004 BTC</b>	RECEIVED ⌚ about 12 hours ago	CONFIRMATIONS ⓘ <b>6+</b>
--	---------------------------	----------------------------------	------------------------------

Advanced Details ▾

## Details



## بخش دوم

در این قسمت باید transaction ای با یک ورودی و یک خروجی مقدار بتی کوین فرستاده شده در قسمت قبل را به خود برگردانیم. کد این قسمت در ادامه آورده شده است:

```
from transactionQ2P1 import *

# 0.0124
# txid = 6863e1ba33f20e3dcb8286dc09999ca019e64595645768826c6f3424d212877e
# msuv6EVFEZyA8N49EGYuxRb1ATFpBvWqxQ
# 91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJXgZr6QA3MHb

bitcoin.SelectParams("testnet")
my_private_key = bitcoin.wallet.CBitcoinSecret("91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJXgZr6QA3MHb").# Private key in WIF format XXXXXXXXXXXXXXXXXXXX
my_public_key = my_private_key.pub

1 usage
def P2PKH_output_scriptPubKey():
    return [OP_DUP, OP_HASH160, Hash160(my_public_key), OP_EQUALVERIFY, OP_CHECKSIG]

1 usage
def P2PKH_scriptSig(txin, txout, txin_scriptPubKey):
    first_person_signature = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey, first_private_key)
    second_person_signature = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey, second_private_key)

    return [OP_0, first_person_signature, second_person_signature]

1 usage
def make_P2PKH_transaction(amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey):
    txout = create_txout(amount_to_send, txout_scriptPubKey)

    txin_scriptPubKey = [OP_2, first_person_public_key, second_person_public_key, third_person_public_key, OP_3, OP_CHECKMULTISIG]

    txin = create_txin(txid_to_spend, utxo_index)

    txin_scriptSig = P2PKH_scriptSig(txin, txout, txin_scriptPubKey)

    new_tx = create_signed_transaction(txin, txout, txin_scriptPubKey, txin_scriptSig)

    return broadcast_transaction(new_tx)

if __name__ == '__main__':
    # 0.0124
    amount_to_send = 0.0120
    txid_to_spend = ('6863e1ba33f20e3dcb8286dc09999ca019e64595645768826c6f3424d212877e')
    utxo_index = 0
    txout_scriptPubKey = P2PKH_output_scriptPubKey()
    response = make_P2PKH_transaction(amount_to_send, txid_to_spend, utxo_index, txout_scriptPubKey)
    print(response.status_code, response.reason)
    print(response.text)

#transaction_hash: fb444f9092ea24dcb9ee48519dcaadfcfe8d7ac2b89a25032149d065919e9467
```


در کد بالا در قسمت P2PKH\_scriptSig ما دو signature را با دستور OP\_CHECKSIG بررسی می نماییم. همچنین در قسمت txin\_scriptPubKey همانند قسمت قبل با استفاده از دستور OP\_2 دو تا از سه کلید عمومی داخل استک را با دستور OP\_CHECKMULTISIG بررسی می نماییم. در صورت درست بود True و در صورت غلط بودن False برگردانده می شود. نتیجه کد و transaction مربوط به آن در ادامه آورده شده است:

```
/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/transactionQ2P2.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "fb444f9092ea24dcb9ee48519dcaadfcfe8d7ac2b89a25032149d065919e9467",
    "addresses": [
      "zG9mCjVbdYomasQwZk3xoN2uRMDa1HVo1z",
      "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
    ],
    "total": 1200000,
    "fees": 40000,
    "size": 230,
    "vsize": 230,
    "preference": "high",
    "relayed_by": "188.118.96.41",
    "received": "2023-05-27T16:02:35.633478997Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "6863e1ba33f20e3dcb8286dc09999ca019e64595645768826c6f3424d212877e",
        "output_index": 0,
        "script": "004730440220457be6d22a080e065b47f6f0eb00fef13241c23143ac2f0b5ea9f72f4f79fbce022065af3b0f6ae74d52837061a8c9f450a21cbb44511b8f7c43cbce66bc4c25164f0147304402203a9cf99e0e0",
        "output_value": 1240000,
        "sequence": 4294967295,
        "addresses": [
          "zG9mCjVbdYomasQwZk3xoN2uRMDa1HVo1z"
        ],
        "script_type": "pay-to-multi-pubkey-hash",
        "age": 2435665
      }
    ],
    "outputs": [
      {
        "value": 1200000,
        "script": "76a91487f93ca45d40079f8862a11bf7b7c1e7a66f8edc88ac",
        "addresses": [
          "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}
```

**TXID:** fb444f9092ea24dcb9ee48519dcaadfcfe8d7ac2b89a25032149d065919e9467

**Address:** msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ

**PrivateKey:** 91wCkGpbGK42f3RXVUJ4utnewk3SoPnMDUq7JJPjXgZr6QA3MHb

 **Bitcoin Testnet Transaction**

fb444f9092ea24dcb9ee48519dcaadfcfe8d7ac2b89a25032149d065919e9467

AMOUNT TRANSACTED	FEES	RECEIVED	CONFIRMATIONS ⓘ
0.012 BTC	0.0004 BTC	🕒 about 11 hours ago	🔒 6+

Advanced Details ▾

#### Details



## سوال سوم

در این بخش لازم است که دو عدد اول را انتخاب کرده و پس از بدست آوردن حاصل جمع و تفریق این دو عدد مقادیر حاصل را در scriptPubKey قرار بدهیم تا شخص دیگر تنها با دانستن این دو عدد اول بتواند آن را خرج نماید. اعداد اول ما ۲۹۰۲ و ۲۴۹ می باشد.

### بخش اول

کد این قسمت در ادامه آورده شده است:

```
import bitcoin.wallet
from utils import *
PRIME_NUM1 = 2909
PRIME_NUM2 = 249
# 0.0120
# txid = fb444f9092ea24dcb9ee48519dcaadfcfe8d7ac2b89a25032149d065919e9467
# msuv6EVFEZyA8N49EGYuxRb1ATFpBvWqxQ
# 91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJxgZr6QA3MHb

bitcoin.SelectParams("testnet")

my_private_key = bitcoin.wallet.CBitcoinSecret("91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJxgZr6QA3MHb")
my_public_key = my_private_key.pub

# usage
def P2PKH_txin_scriptPubKey():
    return [OP_DUP, OP_HASH160, Hash160(my_public_key), OP_EQUALVERIFY, OP_CHECKSIG]

# usage
def P2PKH_scriptSig(txin, txout, txin_scriptPubKey):
    signature = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey, my_private_key)
    return [signature, my_public_key]

# usage
def make_transaction(amount_to_send, txid_to_spend, utxo_index):
    sub = PRIME_NUM1 - PRIME_NUM2
    add = PRIME_NUM1 + PRIME_NUM2
    txout_scriptPubKey = [OP_2DUP,
                          OP_SUB, OP_HASH160, Hash160(sub.to_bytes(2, byteorder="little")), OP_EQUALVERIFY,
                          OP_ADD, OP_HASH160, Hash160(add.to_bytes(2, byteorder="little")), OP_EQUAL]

    txout = create_txout(amount_to_send, txout_scriptPubKey)

    txin_scriptPubKey = P2PKH_txin_scriptPubKey()
    txin = create_txin(txid_to_spend, utxo_index)
    txin_scriptSig = P2PKH_scriptSig(txin, txout, txin_scriptPubKey)

    new_tx = create_signed_transaction(txin, txout, txin_scriptPubKey, txin_scriptSig)

    return broadcast_transaction(new_tx)

if __name__ == '__main__':
    # 0.0120
    amount_to_send = 0.0116
    txid_to_spend = ('fb444f9092ea24dcb9ee48519dcaadfcfe8d7ac2b89a25032149d065919e9467')
    utxo_index = 0

    response = make_transaction(amount_to_send, txid_to_spend, utxo_index)
    print(response.status_code, response.reason)
    print(response.text)

#result = 8b9c71c4d0794e308fe24c3fa0610d1769051108946101b8c7acd8443adc389b
```

در کد بالا قسمت txout\_scriptPubKey به صورتی تعیین می گردد که ابتدا با دستور OP\_2DUP دو عدد اول را از استک برداشته و از آن دو کپی می گیرد. سپس با دستور OP\_SUB مقدار عملیات تفرق بر روی این دو عدد

بدست آمده از استک انجام می گیرد و از نتیجه آن hash گرفته می شود و سپس با دستور OP\_EQUALVERIFY مقدار دو حاصل تفریق با هم مقایسه می شوند. در صورت درست بودن ا برگردانده می شود. حال نوبت بررسی حاصل جمع این دو است. دو عدد خارج شده از استک را با دستور OP\_ADD با هم جمع می کند سپس از آن hash گرفته می شود و با دستور OP\_EQUAL دو مقدار جمع با هم مقایسه می شوند. در صورت درست بودن ا برگردانده می شود. اگر دو مقدار OP\_EQUAL و OP\_EQUALVERIFY هر دو برابر یک باشد. شخص می تواند آن را خرج نماید.

در ادامه نتیجه کد و transaction مربوطه آورده شده است:

```
/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/transactionQ3P1.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "8b9c71c4d0794e308fe24c3fa0610d1769051108946101b8c7acd8443adc389b",
    "addresses": [
      "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
    ],
    "total": 1160000,
    "fees": 40000,
    "size": 247,
    "vsize": 247,
    "preference": "high",
    "relayed_by": "188.118.96.41",
    "received": "2023-05-27T16:49:11.557148187Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "fb444f9092ea24dcb9ee48519dcaadfcfe8d7ac2b89a25032149d0865919e9467",
        "output_index": 0,
        "script": "4739440220749cabf0c71bab320aedd9a02d3835b46da35ffe51e815ad032465fc9d06a6e302200fa3636d73f66d7d73866d397e076baeb05239e9ee78fc6e53f49c424f007a3c01418451ae5229942b0107566db0e97c911ee29e869791437b5e358d",
        "output_value": 1200000,
        "sequence": 4294967295,
        "addresses": [
          "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
        ],
        "script_type": "pay-to-pubkey-hash",
        "age": 2435666
      }
    ],
    "outputs": [
      {
        "value": 1160000,
        "script": "6e94a9146184ecb299607a233b6f7a61a6567c46f5c9a4458893a914da95619fbb5f2fcb68608bf655a0ecb47b52813687",
        "addresses": null,
        "script_type": "unknown"
      }
    ]
  }
}
```

**TXID:** 8b9c71c4d0794e308fe24c3fa0610d1769051108946101b8c7acd8443adc389b

**Address:** msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ

**PrivateKey:** 91wCkGpbGK42f3RXVUJ4utnewk3SoPnMDUq7JJPjXgZr6QA3MHb

## Bitcoin Testnet Transaction

8b9c71c4d0794e308fe24c3fa0610d1769051108946101b8c7acd8443adc389b

AMOUNT TRANSACTED  
**0.0116 BTC**

FEES  
**0.0004 BTC**

RECEIVED  
**⌚ about 11 hours ago**

CONFIRMATIONS ⓘ  
**🔒 6+**

Advanced Details ▾

### Details

1 Input Consumed

0.012 BTC from  
msuv6EVEZyA8N49EGYuxRbiATFpBvWqxQ (output)



1 Output Created

0.0116 BTC Unknown Script Type

### بخش دوم

در این بخش با دانستن دو عدد اول مربوطه در قسمت قبل می توانیم آن را خرج نماییم. در ادامه کد این قسمت آورده شده است:

```
import bitcoin.wallet
from utils import *

PRIME_NUM1 = 2909
PRIME_NUM2 = 249

# 0.0116
# txid = 8b9c71c4d0794e308fe24c3fa0610d1769051108946101b8c7acd8443adc389b
# msuv6EVEZyA8N49EGYuxRbiATFpBvWqxQ
# 91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJxgZr6QA3MHb

bitcoin.SelectParams("testnet")

my_private_key = bitcoin.wallet.CBitcoinSecret("91wCk6pb6K42f3RXVUJ4utnewk3SoPnMDUq7JJpJxgZr6QA3MHb")
my_public_key = my_private_key.pub

1 usage
def P2PKH_txin_scriptPubKey():
    return [OP_DUP, OP_HASH160, Hash160(my_public_key), OP_EQUALVERIFY, OP_CHECKSIG]

def P2PKH_scriptSig(txin, txout, txin_scriptPubKey):
    signature = create_OP_CHECKSIG_signature(txin, txout, txin_scriptPubKey, my_private_key)
    return [signature, my_public_key]

1 usage
def make_multisig_transaction(amount_to_send, txid_to_spend, utxo_index):

    sub = PRIME_NUM1 - PRIME_NUM2
    add = PRIME_NUM1 + PRIME_NUM2

    txout_scriptPubKey = P2PKH_txin_scriptPubKey()

    txout = create_txout(amount_to_send, txout_scriptPubKey)

    txin_scriptPubKey = [OP_2DUP,
        OP_SUB, OP_HASH160, Hash160(sub.to_bytes(2, byteorder="little")), OP_EQUALVERIFY,
        OP_ADD, OP_HASH160, Hash160(add.to_bytes(2, byteorder="little")), OP_EQUAL]

    txin = create_txin(txid_to_spend, utxo_index)
```

```

    txin_scriptSig = [PRIME_NUM1, PRIME_NUM2]

    new_tx = create_signed_transaction(txin, txout, txin_scriptPubKey, txin_scriptSig)

    return broadcast_transaction(new_tx)

if __name__ == '__main__':
    # 0.0116
    amount_to_send = 0.0112
    txid_to_spend = ('8b9c71c4d0794e308fe24c3fa0610d1769051108946101b8c7acd8443adc389b')
    utxo_index = 0

    response = make_multisig_transaction(amount_to_send, txid_to_spend, utxo_index)
    print(response.status_code, response.reason)
    print(response.text)

```

این بخش مانند بخش قبلی می باشد با این تفاوت که مقدار txout\_scriptPubKey با txin\_scriptPubKey عوض شده است و همچنین مقدار txin\_scriptSig به [prime\_num1,prime\_num2] تغییر کرده است. نتیجه کد به همراه transaction مربوطه در ادامه آورده شده است:

```

/home/sina/PycharmProjects/pythonProject1/venv/bin/python /home/sina/PycharmProjects/pythonProject1/transactionQ3P2.py
201 Created
{
  "tx": {
    "block_height": -1,
    "block_index": -1,
    "hash": "54750f60bec7f093e9c6be9b4d398c62042b4830da498897d8d436319bcde627",
    "addresses": [
      "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
    ],
    "total": 1120000,
    "fees": 40000,
    "size": 91,
    "vsize": 91,
    "preference": "high",
    "relayed_by": "100.118.96.41",
    "received": "2023-05-27T17:08:52.867394476Z",
    "ver": 1,
    "double_spend": false,
    "vin_sz": 1,
    "vout_sz": 1,
    "confirmations": 0,
    "inputs": [
      {
        "prev_hash": "8b9c71c4d0794e308fe24c3fa0610d1769051108946101b8c7acd8443adc389b",
        "output_index": 0,
        "script": "025d0b02f900",
        "output_value": 1160000,
        "sequence": 4294967295,
        "script_type": "unknown",
        "age": 2435669
      }
    ],
    "outputs": [
      {
        "value": 1120000,
        "script": "76a91487f93ca45d48079f8862a11bf7b7c1e7a46f8edc8bac",
        "addresses": [
          "msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ"
        ],
        "script_type": "pay-to-pubkey-hash"
      }
    ]
  }
}

```

**TXID:** 54750f60bec7f093e9c6be9b4d398c62042b4830da498897d8d436319bcde627

**Address:** msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ

**PrivateKey:** 91wCkGpbGK42f3RXVUJ4utnewk3SoPnMDUq7JJPjXgZr6QA3MHb



## Bitcoin Testnet Transaction

54750f60bec7f093e9c6be9b4d398c62042b4830da498897d8d436319bcde627

AMOUNT TRANSACTED

**0.0112 BTC**

FEES

**0.0004 BTC**

RECEIVED

**about 11 hours ago**

CONFIRMATIONS ⓘ

**6+**

Advanced Details ▾

### Details

1 Input Consumed

0.0112 BTC Unknown Script Type (output)

1 Output Created

0.0112 BTC to  
msuv6EVFEZyA8N49EGYuxRbiATFpBvWqxQ (unspent)

## قسمت سوم

کد این قسمت به صورت زیر می باشد:

```
import bitcoin.wallet
from bitcoin.core import b2lx
from utils import *
import time
import struct
from hashlib import sha256

bitcoin.SelectParams("mainnet")

private_key = "KxeN3BxxGstQhXSsd8zfy3Wpfc58KnDSmAjpL6g4SJekBdo1eTwN"
my_private_key = bitcoin.wallet.CBitcoinSecret(private_key)
my_public_key = my_private_key.pub

! usage
def P2PKH_scriptPubKey(key):
    return [OP_DUP, OP_HASH160, Hash160(key), OP_EQUALVERIFY, OP_CHECKSIG]

! usage
def create_transaction():

    coinbase_data = '8101995548inaTabassi'.encode('utf-8').hex()
    block_reward = 6.25

    txout_scriptPubKey = P2PKH_scriptPubKey(my_public_key)
    txid_to_spend, index = (04*'0'), int('0xFFFFFFFF', 16)
    txin = create_txin(txid_to_spend, index)
    txout = create_txout(block_reward, txout_scriptPubKey)
    txin.scriptSig = CScript([int(coinbase_data, 16).to_bytes(len(coinbase_data)//2, 'big')])

    return CMutableTransaction([txin], [txout])
```

```

def target_calc(bits):

    target = int(bits[4:], 16) * (int('2', 16) ** (8 * (int(bits[2:4], 16) - 3)))
    target_hex = format(target, 'x')
    target_hex = str(target_hex).zfill(64)
    print("Target in hex: ", target_hex)
    return bytes.fromhex(target_hex)

1 usage
def create_block(header, block_body):

    block_size = len(header) + len(b'\x01') + len(block_body)
    magic_number = 0xd9848ef9.to_bytes(4, "little")
    block = magic_number + struct.pack("<L", block_size) + header + b'\x01' + block_body
    return block

1 usage
def print_block_inforamtion(nonce, hash, header, hash_rate, time_stamp, block, version, block_body, merkle_root):

    print("##### Mining Successfully Done! #####")
    print("Merkle root: ", merkle_root)
    print("Block body: ", block_body.hex())
    print("Block header:", header.hex())
    print("Block hash:", b2lx(hash))
    print("Hash rate:", hash_rate, "H/s")
    print("Block in hex:", b2x(block))
    print("Version:", version)
    print("Time stamp:", time_stamp)
    print("Nonce:", nonce)

3 usages
def reverse(x):
    return x[::-1]

```

```

def header_maker(version, merkle_root, time_stamp, bits, prev_block_hash):
    return struct.pack("<L", version) + reverse(bytes.fromhex(prev_block_hash)) + reverse(
        bytes.fromhex(merkle_root)) + struct.pack('<LL', time_stamp, int(bits, 16))

1 usage
def bitcoin_mining(prev_block_hash):

    version = 2
    bits = '0x1f010000'

    tx = create_transaction()
    block_body = tx.serialize()
    merkle_root = b2lx(sha256(sha256(block_body).digest()).digest())
    target = target_calc(bits)
    time_stamp = int(time.time())
    partial_header = header_maker(version, merkle_root, time_stamp, bits, prev_block_hash)
    nonce = 0

    start_time = time.time()
    while nonce <= 2**32:

        header = partial_header + struct.pack('<L', nonce)
        hash = sha256(sha256(header).digest()).digest()

        if reverse(hash) < target:

            block = create_block(header, block_body)
            hash_rate = nonce / (time.time() - start_time)
            print_block_inforamtion(nonce, hash, header, hash_rate, time_stamp, block, version, block_body, merkle_root)

            return

        nonce = nonce + 1

```

```
if __name__ == '__main__':  
  
    # prev_block_hash = '00000000691b6242b2224682ba60c7f4af8bb2d1d412f47d9e4c50e8aaf99fbc'  
    # n = 9554  
  
    number = input("enter the block number: ")  
    prev_block_hash = input("enter pre block hash: ")  
    bitcoin_mining(prev_block_hash)
```

اطلاعات اولیه برای ایجاد بلوک به صورت زیر می باشد:

Block\_height = 9554

```
Block_hash = 0000000069106242b2224082ba60c7f4af8bb2d1d412f47d9e4c50e8aaf99fbe
```

حال باید coinbase را ایجاد کنیم که اطلاعات آن به صورت زیر می باشد:

Txid\_to\_spend = 64\* '0'

Txid\_to\_spend\_index = 0xFFFFFFFF

```
ScriptSig_input = 810199554SinaTabassi
```

حال باید merkel tree را ایجاد کنیم که برای این لازم است تا stream format را بدست بیاوریم. از آنجا که در اینجا یک transaction داریم بنابراین merkel root برابر coinbase می باشد. بنابراین از تابع serialize استفاده می نماییم.

حال در ادامه باید مقدار target را بدست بیاوریم تا بتوان عملیات mine را انجام داد. مقدار target با n بیتی که با توجه به تعداد صفرهایی که برای سختی انتخاب می شود معین می شود. برای ۴ تا صفر از 0x1f010000 به عنوان بیت ها استفاده می کنیم.

در ادامه باید بخشی از header بلوک را تشکیل دهیم. زیرا بیشتر header دست نخورده باقی می ماند به جز nounce. در ادامه باید در عمل mine کردن nounce را به header متصل کرده و سپس hash می کنیم. در نهایت ما شروع به mining می کنیم. از nounce صفر شروع می کنیم. در نهایت nounce را به header متصل می کنیم و بعد دو بار sha256 را بر روی header انجام می دهیم. حال باید ببینیم که کوچکتر از target می باشد یا خیر. اگر کوچک تر بود به این معنی است که mine بلوک به درستی انجام شده است. تمامی موارد را می توان در مد بالا مشاهده نمود. در ادامه نتیجه کد آمده است:

[illegible]

Result Hash: 00003101fbc22e83bd8700416041af0b467f50aaa1094677ab29f8928d5f6a53