Bitcoin Scripting Assignment Report

CS 216: Introduction to Blockchain - Assignment 3

Team : wasd

Submitted by:

- Rohan Dhiman (230001069)
- Mani Kaustubh Mathur (230001050)
- Siddharth Singh (230002068)

1. Introduction

This report presents our implementation and analysis of Bitcoin transactions using both Legacy (P2PKH) and SegWit (P2SH-P2WPKH) address formats. We developed Python scripts to interact with bitcoind in regtest mode, create and broadcast transactions, and analyze the resulting scripts.

2. Environment Setup

We configured Bitcoin Core to run in regtest mode with the following settings in bitcoin.conf:

```
regtest=1
server=1
rpcuser=Wasd
rpcpassword=8520
paytxfee=0.0001
fallbackfee=0.0002
mintxfee=0.00001
txconfirmtarget=1
```

For our implementation, we used Python with the python-bitcoinrpc and simplejson libraries to interact with the Bitcoin daemon.

3. Legacy Address Transactions (P2PKH)

3.1 Workflow

1. **Address Generation**: We generated three legacy addresses (A, B, and C) using the Bitcoin Core wallet.

```
"address_a": "mo3Gb91CZh3MtPGdzJbBw8cFNZW4xN8ihT",
"address_b": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
```

```
"address_c": "mgrxiR3r336phqmN2EGuTtMCF25PJsBK4u",
```

2. **Funding Address A**: We funded address A using the sendtoaddress command.

```
print("Funding Address A")
 txid_funding = rpc.sendtoaddress(address_a, 1.0)
 print(f"Funding transaction ID: {txid_funding}")
```

3. **Transaction from A to B**: We created a raw transaction sending coins from A to B, signed it, and broadcast it to the network.

```
{
        "address_a": "mo3Gb91CZh3MtPGdzJbBw8cFNZW4xN8ihT",
        "address_b": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
        "address_c": "mgrxiR3r336phqmN2EGuTtMCF25PJsBK4u",
        "tx_a_to_b":
"07b3c2c3e791e11c3faffb0333e207bb016633cf4f727621e56874d1e0f48c9f"
    }
```

4. **Transaction from B to C**: We used the UTXO from the previous transaction to create a new transaction from B to C.

```
{
        "address_a": "mo3Gb91CZh3MtPGdzJbBw8cFNZW4xN8ihT",
        "address_b": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
        "address_c": "mgrxiR3r336phqmN2EGuTtMCF25PJsBK4u",
        "tx_a_to_b":
"07b3c2c3e791e11c3faffb0333e207bb016633cf4f727621e56874d1e0f48c9f",
        "tx_b_to_c":
"bae2c3576128e35c649ba30538b9e24aecf300aad7063a0e230a344fd35ffa23",
        "scriptSig": {
            "asm":
"3044022073fdb36a203bc0eb4ffa8d2545f26cb4aeec78ff31ae92d79a937073793f1bfb02205
d4441054c0f984cf795deac8e9034e68b20cc323cd3332d45f00666811f8315[ALL]
02a6b47f2ea4d66df2b14b9878c53fe58fec9724500f74390ec45b43994c374bfb",
            "hex":
"473044022073fdb36a203bc0eb4ffa8d2545f26cb4aeec78ff31ae92d79a937073793f1bfb022
05d4441054c0f984cf795deac8e9034e68b20cc323cd3332d45f00666811f8315012102a6b47f2
ea4d66df2b14b9878c53fe58fec9724500f74390ec45b43994c374bfb"
        },
        "previousScriptPubKey": {
            "asm": "OP_DUP OP_HASH160 0308be3f7ded2074b14c477fa18ec0ecfe3e3dfc
OP_EQUALVERIFY OP_CHECKSIG",
```

```
            "desc": "addr(mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv)#5d7z884z",
            "hex": "76a9140308be3f7ded2074b14c477fa18ec0ecfe3e3dfc88ac",
            "address": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
            "type": "pubkeyhash"
        }
    }
```

3.2 Script Analysis

Transaction from A to B

The locking script (ScriptPubKey) for address B follows the P2PKH format:

```
text
OP_DUP OP_HASH160 <PubKeyHash of B> OP_EQUALVERIFY OP_CHECKSIG
```

This script locks the funds such that only the owner of the private key corresponding to address B can spend them.

```
"tx_a_to_b": {
            "size": 191,
            "vsize": 191,
            "weight": 764,
            "scriptPubKey": "OP_DUP OP_HASH160
00dd19fff02ba87f0568da1b282a93b1c2d598dc OP_EQUALVERIFY OP_CHECKSIG"
        }
```

```
{
        "address_a": "mo3Gb91CZh3MtPGdzJbBw8cFNZW4xN8ihT",
        "address_b": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
        "address_c": "mgrxiR3r336phqmN2EGuTtMCF25PJsBK4u",
        "tx_a_to_b":
"07b3c2c3e791e11c3faffb0333e207bb016633cf4f727621e56874d1e0f48c9f"
    }
```

Transaction from B to C

The unlocking script (ScriptSig) for spending from address B contains:

```
text
<Signature> <Public Key of B>
```

When executed with the locking script, this proves ownership of the private key corresponding to address B.

```
"tx_b_to_c": {
            "size": 191,
```

```
            "vsize": 191,
            "weight": 764,
            "scriptSig":
"3044022016fd32832079dc113843014b53f3439300eabff1836346e11d30b911da90ec1d02203
851e58d65bff9c29860f051bacb17eff6cf73a22b406566d5e2e8f096a32a7f[ALL]
02cae87b2b5a19279fee686c92490e919662dfc152faabf3049403b18a1ab35133"
        }
```


3.3 Script Validation

The validation process works as follows:

1. The unlocking script provides the signature and public key
2. The locking script verifies that:
    - The hash of the public key matches the expected hash
    - The signature is valid for the transaction and public key


```
{
        "address_a": "mo3Gb91CZh3MtPGdzJbBw8cFNZW4xN8ihT",
        "address_b": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
        "address_c": "mgrxiR3r336phqmN2EGuTtMCF25PJsBK4u",
        "tx_a_to_b":
"07b3c2c3e791e11c3faffb0333e207bb016633cf4f727621e56874d1e0f48c9f",
        "tx_b_to_c":
"bae2c3576128e35c649ba30538b9e24aecf300aad7063a0e230a344fd35ffa23",
        "scriptSig": {
            "asm":
"3044022073fdb36a203bc0eb4ffa8d2545f26cb4aeec78ff31ae92d79a937073793f1bfb02205
d4441054c0f984cf795deac8e9034e68b20cc323cd3332d45f00666811f8315[ALL]
02a6b47f2ea4d66df2b14b9878c53fe58fec9724500f74390ec45b43994c374bfb",
            "hex":
"473044022073fdb36a203bc0eb4ffa8d2545f26cb4aeec78ff31ae92d79a937073793f1bfb022
05d4441054c0f984cf795deac8e9034e68b20cc323cd3332d45f00666811f8315012102a6b47f2
ea4d66df2b14b9878c53fe58fec9724500f74390ec45b43994c374bfb"
        },
        "previousScriptPubKey": {
            "asm": "OP_DUP OP_HASH160 0308be3f7ded2074b14c477fa18ec0ecfe3e3dfc
OP_EQUALVERIFY OP_CHECKSIG",
            "desc": "addr(mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv)#5d7z884z",
            "hex": "76a9140308be3f7ded2074b14c477fa18ec0ecfe3e3dfc88ac",
            "address": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
            "type": "pubkeyhash"
        }
    }
```

4. SegWit Address Transactions (P2SH-P2WPKH)

4.1 Workflow

1. **Address Generation**: We generated three P2SH-SegWit addresses (A', B', and C').
2. **Funding Address A'**: We funded address A' using the sendtoaddress command.
3. **Transaction from A' to B'**: We created a raw transaction sending coins from A' to B', signed it, and broadcast it.
4. **Transaction from B' to C'**: We used the UTXO from the previous transaction to create a new transaction from B' to C'.


```
{
        "address_a": "mo3Gb91CZh3MtPGdzJbBw8cFNZW4xN8ihT",
        "address_b": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
        "address_c": "mgrxiR3r336phqmN2EGuTtMCF25PJsBK4u",
        "tx_a_to_b":
"07b3c2c3e791e11c3faffb0333e207bb016633cf4f727621e56874d1e0f48c9f",
        "tx_b_to_c":
"bae2c3576128e35c649ba30538b9e24aecf300aad7063a0e230a344fd35ffa23",
        "scriptSig": {
            "asm":
"3044022073fdb36a203bc0eb4ffa8d2545f26cb4aeec78ff31ae92d79a937073793f1bfb02205
d4441054c0f984cf795deac8e9034e68b20cc323cd3332d45f00666811f8315[ALL]
02a6b47f2ea4d66df2b14b9878c53fe58fec9724500f74390ec45b43994c374bfb",
            "hex":
"473044022073fdb36a203bc0eb4ffa8d2545f26cb4aeec78ff31ae92d79a937073793f1bfb022
05d4441054c0f984cf795deac8e9034e68b20cc323cd3332d45f00666811f8315012102a6b47f2
ea4d66df2b14b9878c53fe58fec9724500f74390ec45b43994c374bfb"
        },
        "previousScriptPubKey": {
            "asm": "OP_DUP OP_HASH160 0308be3f7ded2074b14c477fa18ec0ecfe3e3dfc
OP_EQUALVERIFY OP_CHECKSIG",
            "desc": "addr(mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv)#5d7z884z",
            "hex": "76a9140308be3f7ded2074b14c477fa18ec0ecfe3e3dfc88ac",
            "address": "mfnzo7byi9R76pf4Ct1eUH5cgf9QSA6qnv",
            "type": "pubkeyhash"
        }
    }
```


4.2 Script Analysis

Transaction from A' to B'

The locking script for a P2SH-P2WPKH address B' has the format:

```
OP_HASH160 <Hash of redeemScript> OP_EQUAL
```

Where the redeemScript is:

```
0 <PubKeyHash of B'>

"tx_a_to_b": {
            "size": 215,
            "vsize": 134,
            "weight": 533,
            "scriptPubKey": "OP_HASH160
4f7e3fbf192f9e4838ceb2232f46d13df9694023 OP_EQUAL"
        }


{
    "address_a_prime": "2N5YVgeVK8JaxDNeEja4vPMSXDDruPaSbe8",
    "address_b_prime": "2MxN4iffihUoJ8WkqgnDCHpdTu6vrGnzeks",
    "address_c_prime": "2N6RSDWaE9FmzrYAn8rA2mzK4PHuAapS3LD",
    "tx_a_to_b":
"15f00e0da6d82c54054ea3939fce946338d03ac306fe87172271759b00871c30"
}
```

Transaction from B' to C'

For a P2SH-P2WPKH transaction, the unlocking script is:

```
<redeemScript>
```

The witness data (not part of the scriptSig) contains:

```
<Signature> <Public Key of B'>

"tx_b_to_c": {
            "size": 215,
            "vsize": 134,
            "weight": 533,
            "scriptSig": "0014f004744611840f3536b244c8b29d5e3b0b5852f4"
        }
```

4.3 Script Validation

The validation process for P2SH-P2WPKH works as follows:

1. The unlocking script provides the redeemScript

2. The locking script verifies that the hash of the redeemScript matches the expected hash
3. The witness data provides the signature and public key
4. The redeemScript is executed with the witness data to verify ownership


```
{
        "address_a_prime": "2N5YVgeVK8JaxDNeEja4vPMSXDDruPaSbe8",
        "address_b_prime": "2MxN4iffihUoJ8WkqgnDCHpdTu6vrGnzeks",
        "address_c_prime": "2N6RSDWaE9FmzrYAn8rA2mzK4PHuAapS3LD",
        "tx_a_to_b":
"15f00e0da6d82c54054ea3939fce946338d03ac306fe87172271759b00871c30",
        "tx_b_to_c":
"229ed081cf9af34e05122018084139b7e05d40ae4b3893357e99ca6c555e5916",
        "scriptSig": {
            "asm": "0014263e7f5fbb9155682a1af5621c00937d01b4bc5d",
            "hex": "160014263e7f5fbb9155682a1af5621c00937d01b4bc5d"
        },
        "previousScriptPubKey": {
            "asm": "OP_HASH160 3823cd961dce36366cbcdc63ace32b1fe5bb0ec0
OP_EQUAL",
            "desc": "addr(2MxN4iffihUoJ8WkqgnDCHpdTu6vrGnzeks)#x23gvyn5",
            "hex": "a9143823cd961dce36366cbcdc63ace32b1fe5bb0ec087",
            "address": "2MxN4iffihUoJ8WkqgnDCHpdTu6vrGnzeks",
            "type": "scripthash"
        }
    }
```


5. Comparison of Legacy and SegWit Transactions

5.1 Transaction Size Comparison

Transaction Type	Size (bytes)	Weight Units	Virtual Bytes
P2PKH (Legacy)	191 + 191	764 + 764	191 + 191
P2SH-P2WPKH	215 + 215	533 + 533	134 + 134


```
"comparison": {
        "size_reduction": "-12.57%",
        "vsize_reduction": "29.84%"
```

```
    }
```

The size in bytes increased by 12.5% but the vsize reduced by about 30%.

5.2 Script Structure Comparison

P2PKH (Legacy):

- ScriptPubKey: `OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`
- ScriptSig: `<Signature> <Public Key>`

P2SH-P2WPKH (SegWit):

- ScriptPubKey: `OP_HASH160 <Hash of redeemScript> OP_EQUAL`
- ScriptSig: `<redeemScript>`
- Witness: `<Signature> <Public Key>`

5.3 Benefits of SegWit Transactions

1. **Reduced Transaction Size**: By moving signature data to the witness, SegWit transactions are smaller in terms of virtual bytes, resulting in lower fees.
2. **Malleability Fix**: SegWit addresses the transaction malleability issue by separating the witness data from the transaction hash calculation.
3. **Increased Block Capacity**: SegWit effectively increases the block capacity without changing the block size limit.
4. **Script Versioning**: SegWit introduces a version field that allows for future script upgrades.