**CS 216 Scripting Assignment**

**Team Name - Blockfinity**

**Members -**

1. **Subhankar Das (230001073)**
2. **Salaj Bansal (230002063)**
3. **Nipun Samal (230041025)**

**Objective:**

The objective of this assignment is to understand the process of creating and validating Bitcoin transactions using Legacy (P2PKH) and SegWit (P2SH-P2WPKH) address formats.

**Setup:**

We have used Python to interact with bitcoind.

The following parameters were set to the corresponding values in ‘bitcoin.conf’ file:

paytxfee - 0.0001 (BTC/kB)

fallbackfee - 0.0002 (BTC/kB)

mintxfee - 0.00001 (BTC/kB)

txconfirmtarget - 6

**Legacy Address Transaction:**

Addresses generated are:

Address A: moFFJr31DfkuKJMhGJVRLXd4TBw8XPaHbT

Address B: n2CTex4hHhmMPK3MGD5SyjKopHGS8MkY1Y

Address C: mj5mFK1PmTJS4NsWCYgvNFwyoPHFs3CCqi

Initially, A is given 1 BTC.

Then, 0.5 BTC is sent from A to B, with a fee of 0.0001 BTC, leaving A with 0.4999 BTC.

TXID for A to B -

*3048109b7c2fbee5d7e901d750d6e7b8f9f335c6eb1877a544af76dd2515f4d9*

Thus, B has one UTXO containing 0.5 BTC.

UTXO set of B -

*{'txid': '3048109b7c2fbee5d7e901d750d6e7b8f9f335c6eb1877a544af76dd2515f4d9', 'vout': 0, 'address': 'n2CTex4hHhmMPK3MGD5SyjKopHGS8MkY1Y', 'label': '', 'scriptPubKey': '76a914e2db3f359ce01b1db3b69eb5a9f631f38af5ea1e88ac', 'amount': Decimal('0.50000000'), 'confirmations': 1, 'spendable': True, 'solvable': True, 'desc': 'pkh([d48c3dee/44h/1h/0h/0/1]03a1f80d1fdf46751ac19be52c30adfc62b681d7a625e35b93f525e50eeebd7d4a)#7rlleftz', 'parent\_descs': ['pkh(tpubD6NzVbkrYhZ4YTiPYoTnmQn3VwYXBcgPM9dENFpsijkoiReXJB4WM77q89ZYVW9f253r1WkACKMsRWQ6Syyj9c6r1iTAWxu9s6t1aScxm3D/44h/1h/0h/0/\*)#wqquptwx'], 'safe': True}*

This UTXO refers to the TXID for A to B and has the address of B along with 1 confirmation, making this UTXO spendable provided the correct response script is provided to the challenge script (scriptPubKey mentioned in the UTXO). Thus, when B spends this UTXO, it uses the previous transaction (from A to B).

Now, 0.25 BTC is sent from B to C along with a fee of 0.0001 BTC, leaving B with 0.2499 BTC. The following command refers to this transaction.

*createrawtransaction [[{"txid": "3048109b7c2fbee5d7e901d750d6e7b8f9f335c6eb1877a544af76dd2515f4d9", "vout": 0}], {"mj5mFK1PmTJS4NsWCYgvNFwyoPHFs3CCqi": 0.25, "n2CTex4hHhmMPK3MGD5SyjKopHGS8MkY1Y": 0.2499}]*

TXID for B to C -

*e12541902639b5992110235a9b538477b650745ef7041385cf1769222c72d860*

Decoded Script for A to B -

*OP\_DUP OP\_HASH160 77f8a0ca27b5e20f226754836ff2f266d9e329ae OP\_EQUALVERIFY OP\_CHECKSIG*

Decoded Script for B to C -

*OP\_DUP OP\_HASH160 149b6ecd97dc692a405919c69f640d61b23c7b04 OP\_EQUALVERIFY OP\_CHECKSIG*

**Procedure**:

In Legacy address transactions, the challenge script and valid response script structures are as follows:

Challenge Script: OP\_DUP OP\_HASH160 <Public Key Hash> OP\_EQUALVERIFY OP\_CHECKSIG

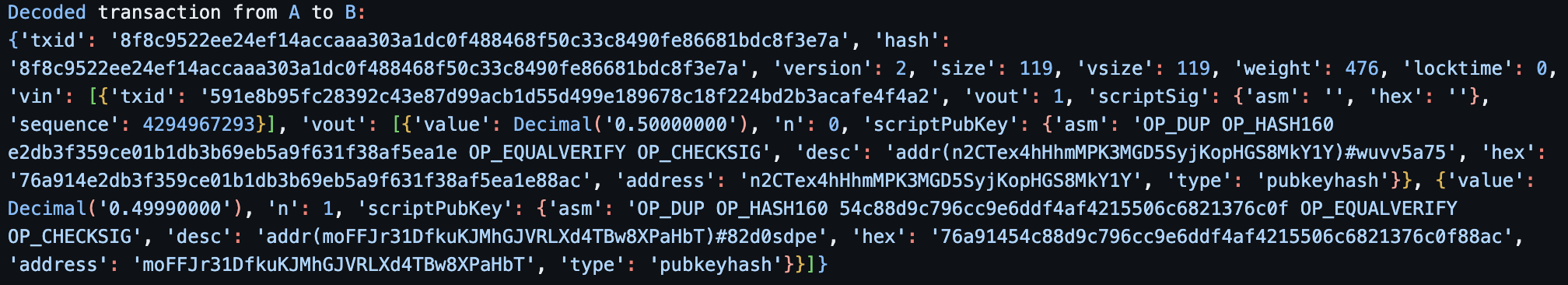
Response Script: <Signature> <Public Key>

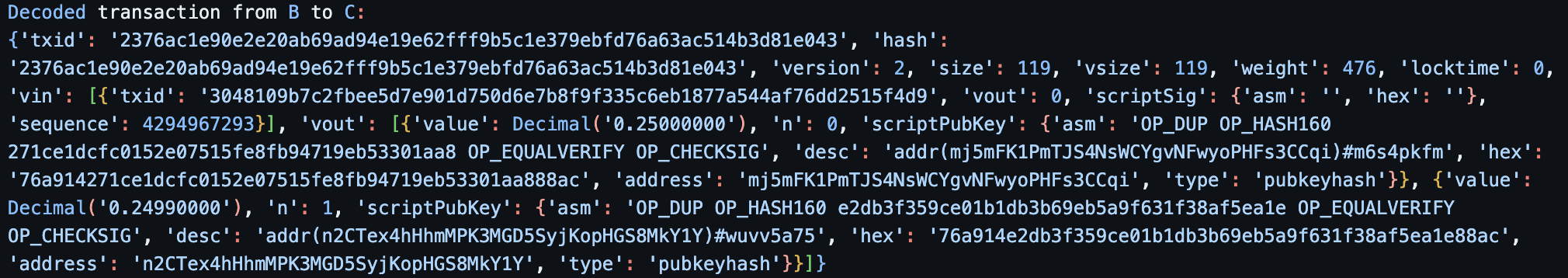
First, the response script is pushed on the stack and then the challenge script is placed on the stack. Then, one by one the operations are executed

1. Public key is duplicated and hashed.
2. This hash is compared with the hash in the challenge script with OP\_EQUALVERIFY.
3. If they are equal, then it is checked if the signature is valid using the public key. If true, then the transaction is valid.

**Screenshots:**

Decoded Scripts -

****

****

Bitcoin Debugger Steps -











**P2SH-SegWit Address Transactions:**

Addresses generated are

SegWit Address A': 2MwbY56NSb27hDgoy6MArCSNDSjZ54gMvfG

SegWit Address B': 2NA5G4fvdTpkELh4R75G3GfYNhTG5YHeLS9

SegWit Address C': 2MxgETHK7EMk1zhMhj8NZbSf6ZMtg7pXQdU

Initially, A is given 1 BTC.

Then, 0.5 BTC is sent from A to B, with a fee of 0.0001 BTC, leaving A with 0.4999 BTC.

TXID for A to B -

*6aeb004b08abccaa83711adffd47421acd55b81305ac2df27bed8c35187a4c8c*

Thus, B has one UTXO containing 0.5 BTC.

Like in the Legacy Address case, here also the UTXO refers to the TXID for A to B and has the address of B along with 1 confirmation, making this UTXO spendable provided the correct response script is provided to the challenge script (scriptPubKey mentioned in the UTXO). Thus, when B spends this UTXO, it uses the previous transaction (from A to B).

Now, 0.25 BTC is sent from B to C along with a fee of 0.0001 BTC, leaving B with 0.2499 BTC. The following command refers to this transaction.

*createrawtransaction [[{"txid": "6aeb004b08abccaa83711adffd47421acd55b81305ac2df27bed8c35187a4c8c", "vout": 0}], {"2MxgETHK7EMk1zhMhj8NZbSf6ZMtg7pXQdU": 0.25, "2NA5*G4fvdTpkELh4R75G3GfYNhTG5YHeLS9": 0.2499}]

TXID for B to C -

*502e5ed2dd0d4ceb2bc3977c06664befca03cbb86bdc9ba37f24919562ca5834*

Decoded Script for A to B -

*OP\_HASH160 b897b61cb7cdb951b2a61e0fd8b49ea163c4ef67 OP\_EQUAL*

Decoded Script for B to C -

*OP\_HASH160 3b936aa1c056f0418b0ce445d26975977a490780 OP\_EQUAL*

**Procedure**:

In P2SH-SegWit address transactions, the challenge script and valid response script structures are as follows:

Challenge Script: OP\_HASH160 <Redeem Script Hash> OP\_EQUAL

Response Script: <Signature> <Public Key> <Redeem Script>

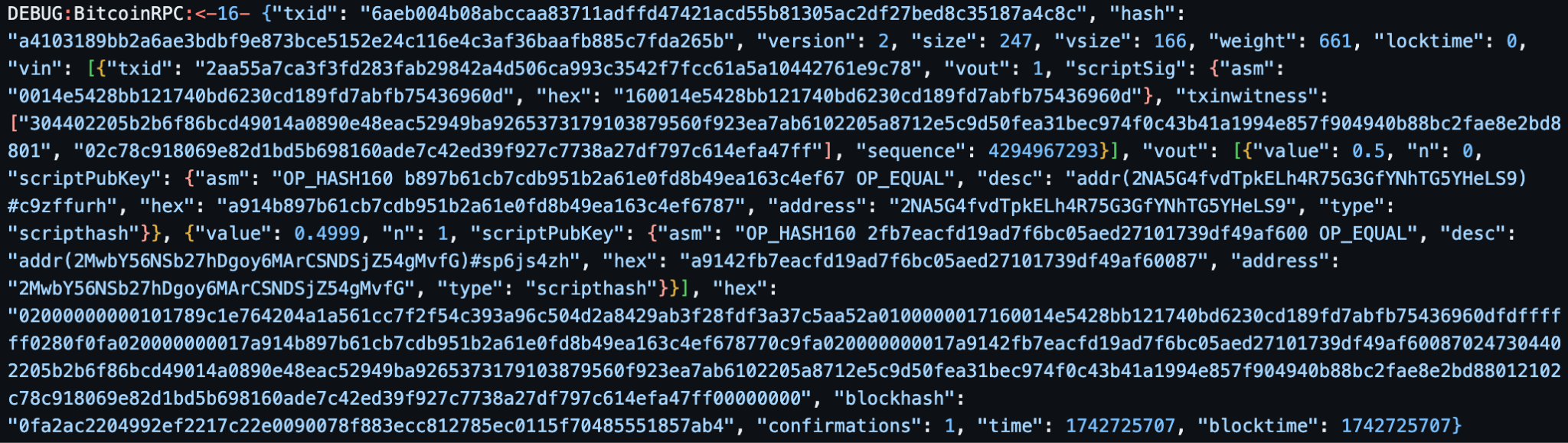
First, the response script is pushed on the stack and then the challenge script is placed on the stack. Then, one by one the operations are executed

1. Redeem Script is hashed and compared with the hash in challenge script using OP\_EQUAL
2. If they are equal, then it is checked if the signature is valid using the public key. If true, then the transaction is valid.

**Screenshots:**

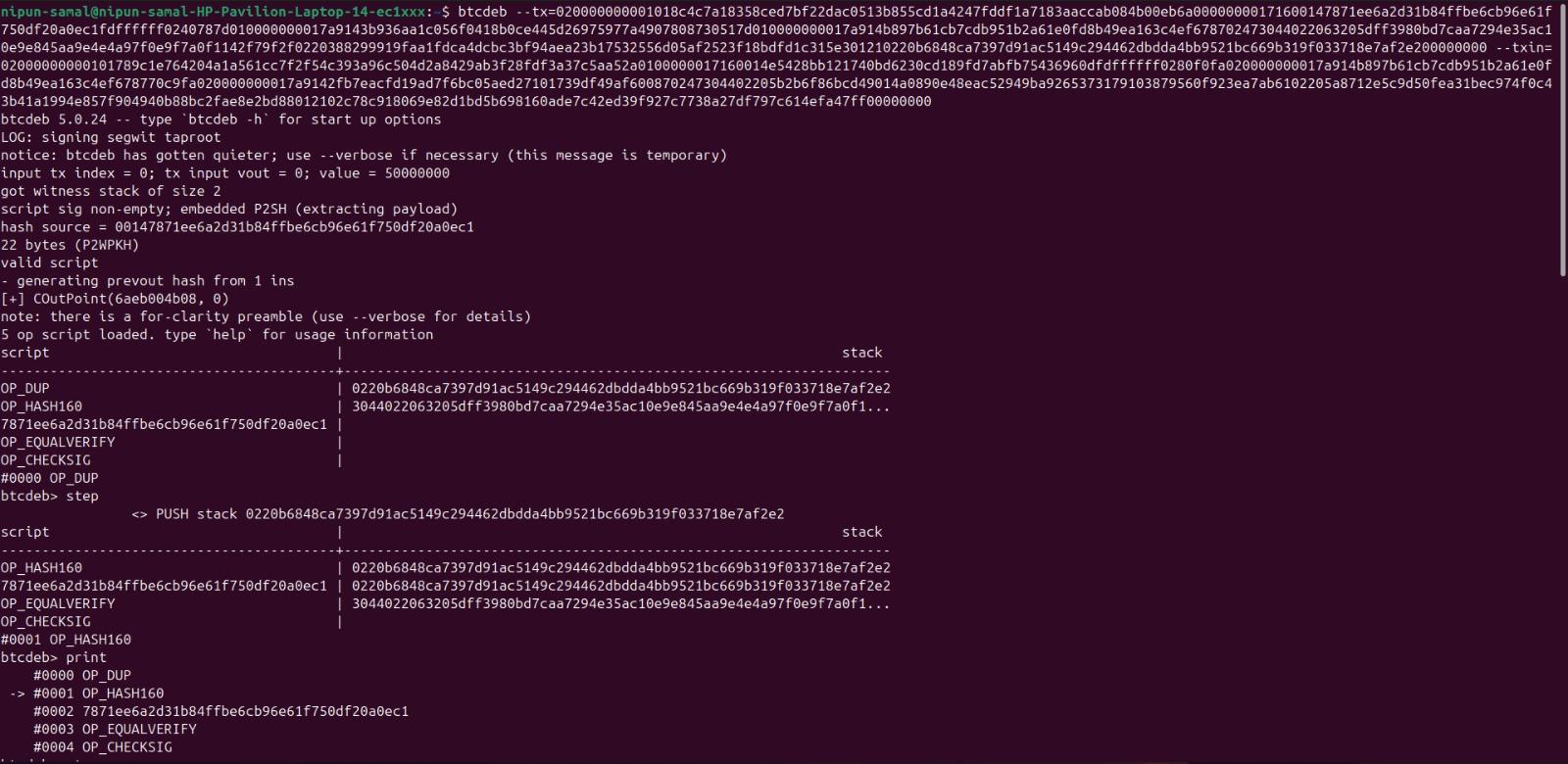
Decoded Scripts -

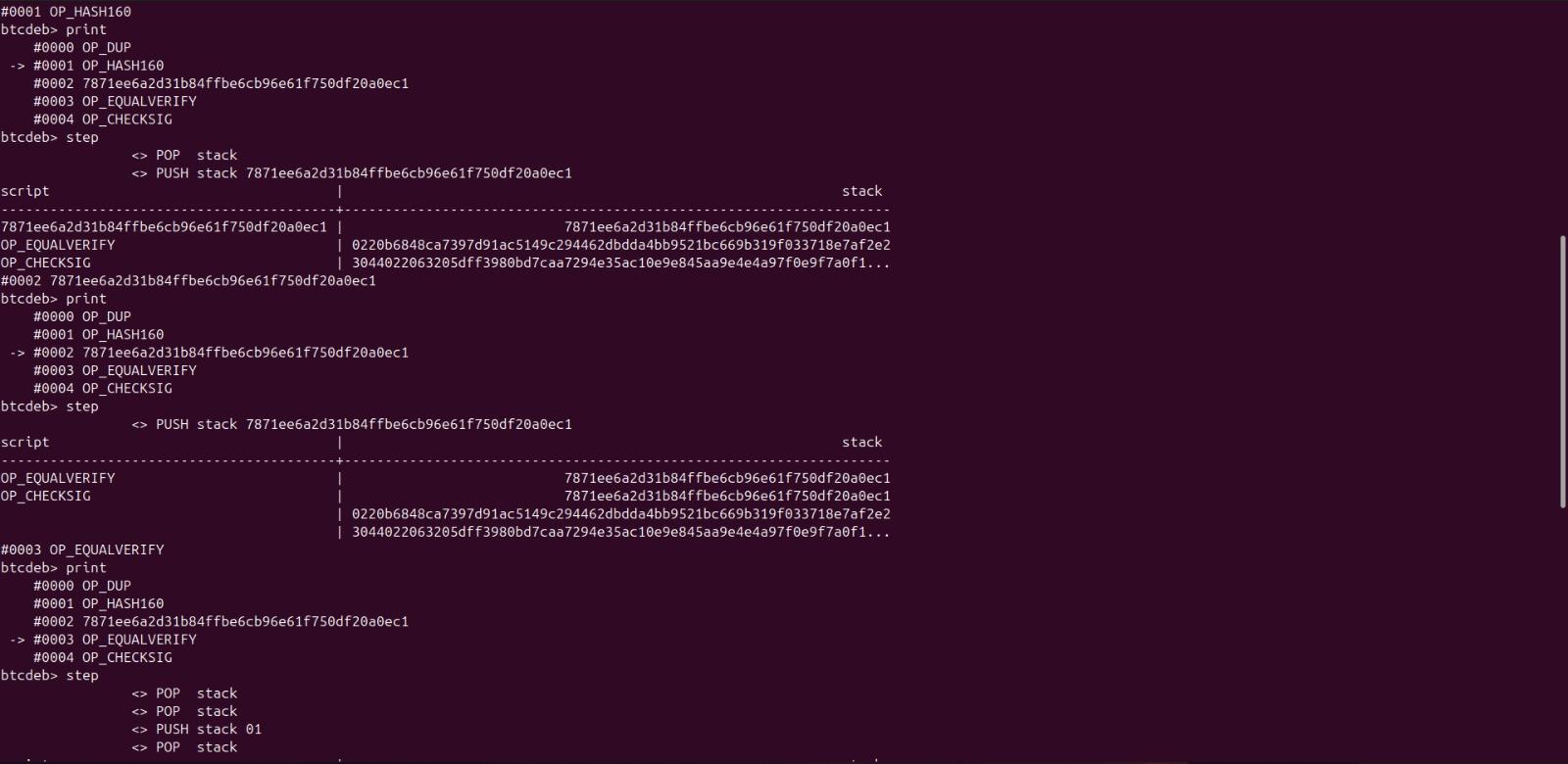
Transaction from A’ to B’ -

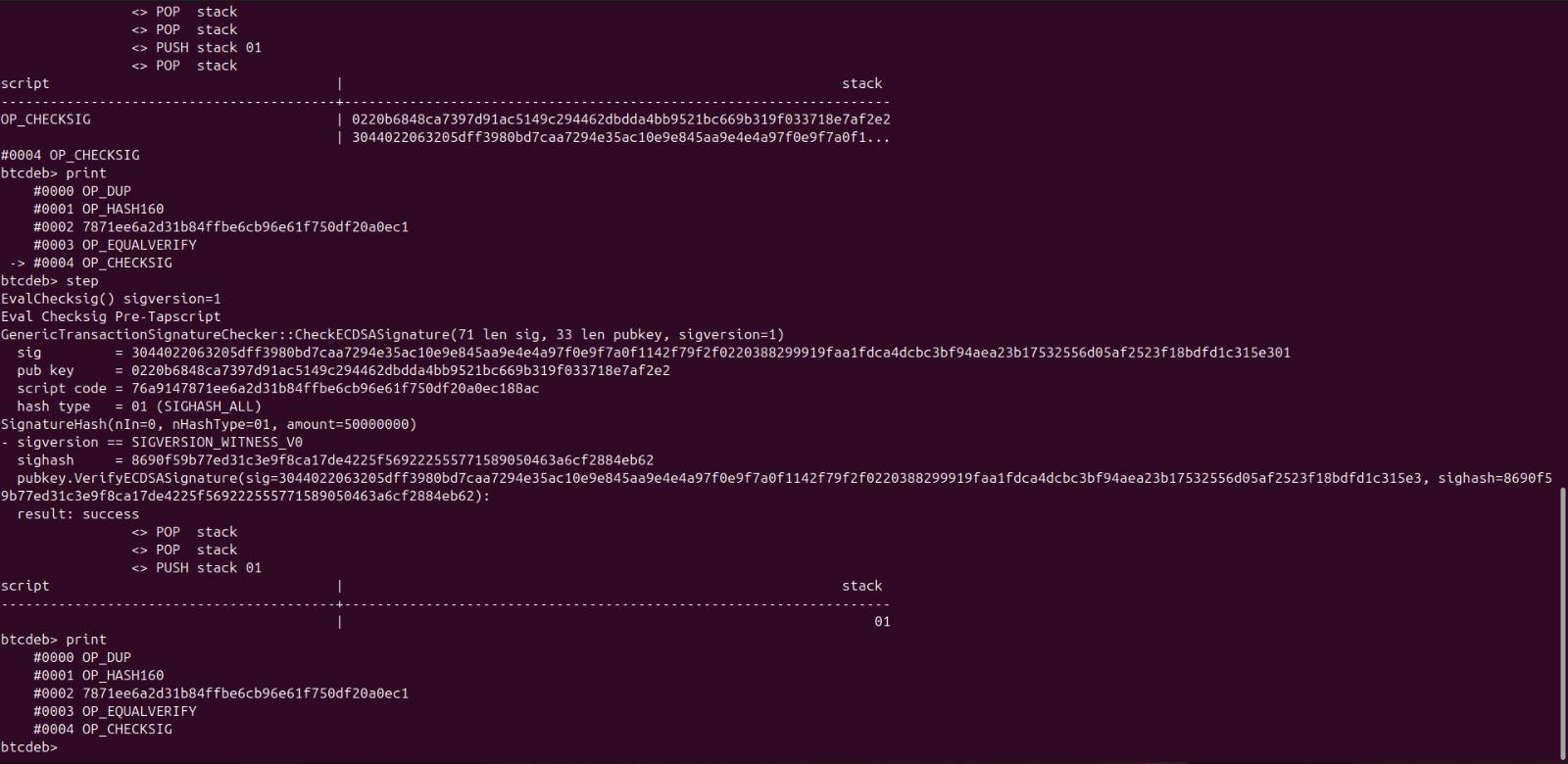


Transaction from B’ to C’ -

Bitcoin Debugger Steps -







**Comparing Legacy and P2SH-SegWit Address Transactions:**

1. **Size:**
   1. P2PKH - 119 bytes or 119 vbytes or 476 weight units
   2. P2SH-SegWit - 247 bytes or 166 vbytes or 661 weight units
2. **Script Structures:**

As mentioned above, the structures of P2PKH scripts and P2SH-SegWit scripts are as follows:

**P2PKH -**

Challenge Script: OP\_DUP OP\_HASH160 <Public Key Hash> OP\_EQUALVERIFY OP\_CHECKSIG

Response Script: <Signature> <Public Key>

**P2SH-SegWit -**

Challenge Script: OP\_HASH160 <Redeem Script Hash> OP\_EQUAL

Response Script: <Signature> <Public Key> <Redeem Script>