INTRODUCTION TO BLOCKCHAIN

ASSIGNMENT -2 (LEGACY AND SEGWIT TRANSACTION)

SUBMITTED BY-

Kunal Gourv (230002036)

Kumar Ayaman (230001044)

Cheepati gireesh kumar reddy (230005013)

Submitted To-

Dr. Subhra Mazumdar

Legacy (P2PKH) Address Transactions

For this section, we have implemented two Python scripts: **legacy_xyz.py** and **legacy1_xyz.py**.

- **legacy_xyz.py** is responsible for initializing the Bitcoin wallet, generating legacy addresses, funding an address through mining, and creating and broadcasting a transaction. It also extracts relevant transaction details, such as the ScriptPubKey and transaction size.
- **legacy1_xyz.py** builds upon the previous script by fetching existing legacy addresses, retrieving UTXO details, and creating a new transaction using the available UTXO balance. Additionally, it decodes the transaction to extract the ScriptSig and ensures the secure transfer of funds to the intended recipient.

a) legacy_xyz.py

1. **Wallet Initialization**:
   - Creates a new wallet named *Team_xyz* or loads it if it already exists.
2. **Address Generation**:

- Generates three legacy Bitcoin addresses: *A*, *B*, and *C*.

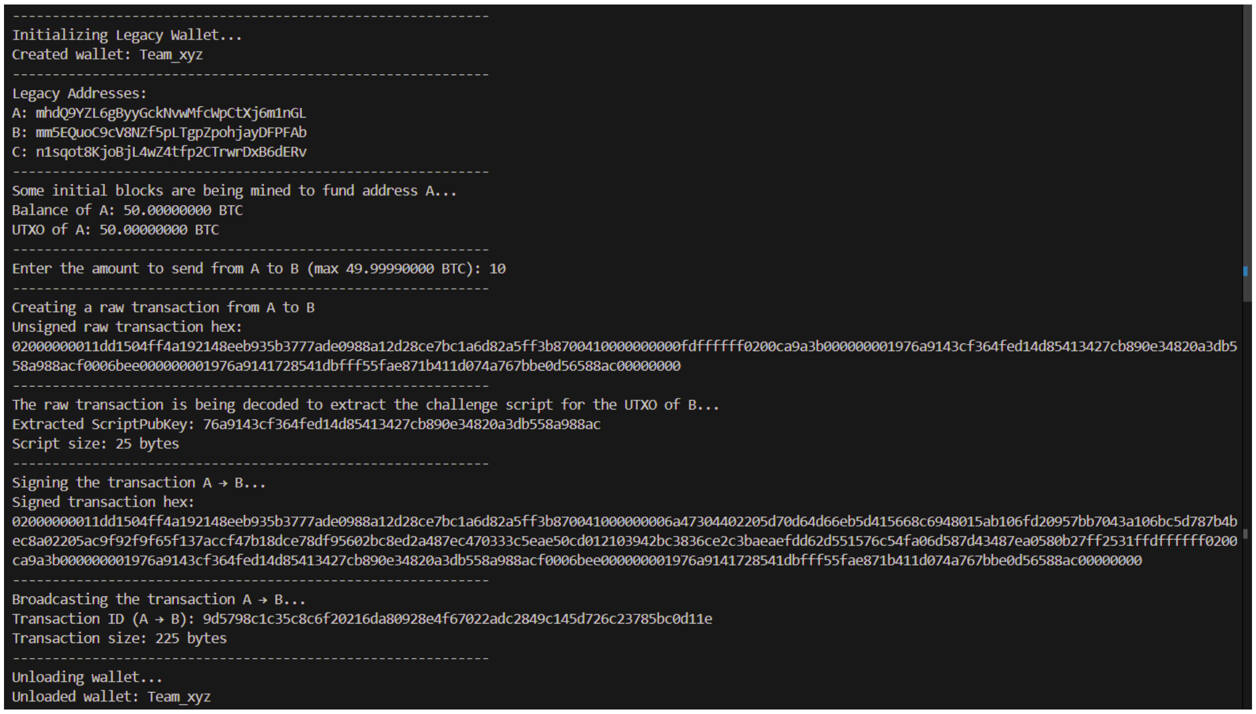
3. **Funding the Wallet**:
   - Mines a series of initial blocks to provide funds for address *A*.
4. **UTXO Balance Display**:
   - Retrieves and displays the available Unspent Transaction Output (UTXO) balance of address *A* once it has received funds.
5. **User Input for Transaction**:
   - Prompts the user to specify the amount to be transferred from *A* to *B*, ensuring it satisfies the constraint:
   - Amount should be less than (UTXO -Minning fees).
6. **Raw Transaction Creation**:
   - Constructs a raw Bitcoin transaction to transfer the specified amount from *A* to *B*.
7. **Transaction Decoding**:
   - Decodes the raw transaction to extract the challenge script (ScriptPubKey) for the newly created UTXO of *B*.
   - Displays the size of the ScriptPubKey in virtual bytes (vbytes).
8. **Transaction Signing and Broadcasting**:

- Signs the transaction transferring funds from *A* to *B*.
- Broadcasts the signed transaction to the Bitcoin network.

9. **Transaction Details**:
   - Displays the transaction ID and the total transaction size in vbytes.

10. **Wallet Cleanup**:
    - Unloads the *Team_xyz* wallet upon completion.

---------------------- Output of legacy1_xyz.py ----------------------



```
------------------------------------------------------------
Initializing Legacy Wallet...
Created wallet: Team_xyz
------------------------------------------------------------
Legacy Addresses:
A: mhdQ9YZL6gByyGckNvwMfcWpCtXj6m1nGL
B: mm5EQuoC9cV8NZf5pLTgpZpohjayDFPFAb
C: n1sqot8KjoBjL4wZ4tfp2CTrwrDxB6dERv
------------------------------------------------------------
Some initial blocks are being mined to fund address A...
Balance of A: 50.00000000 BTC
UTXO of A: 50.00000000 BTC
------------------------------------------------------------
Enter the amount to send from A to B (max 49.99990000 BTC): 10
------------------------------------------------------------
Creating a raw transaction from A to B
Unsigned raw transaction hex:
02000000011dd1504ff4a192148eeb935b3777ade0988a12d28ce7bc1a6d82a5ff3b8700410000000000fdffffff0200ca9a3b000000001976a9143cf364fed14d85413427cb890e34820a3db5
58a988acf0006bee000000001976a9141728541dbfff55fae871b411d074a767bbe0d56588ac00000000
------------------------------------------------------------
The raw transaction is being decoded to extract the challenge script for the UTXO of B...
Extracted ScriptPubKey: 76a9143cf364fed14d85413427cb890e34820a3db558a988ac
Script size: 25 bytes
------------------------------------------------------------
Signing the transaction A → B...
Signed transaction hex:
02000000011dd1504ff4a192148eeb935b3777ade0988a12d28ce7bc1a6d82a5ff3b870041000000006a47304402205d70d64d66eb5d415668c6948015ab106fd20957bb7043a106bc5d787b4b
ec8a02205ac9f92f9f65f137accf47b18dce78df95602bc8ed2a487ec470333c5eae50cd012103942bc3836ce2c3baeaefdd62d551576c54fa06d587d43487ea0580b27ff2531ffdffffff0200
ca9a3b000000001976a9143cf364fed14d85413427cb890e34820a3db558a988acf0006bee000000001976a9141728541dbfff55fae871b411d074a767bbe0d56588ac00000000
------------------------------------------------------------
Broadcasting the transaction A → B...
Transaction ID (A → B): 9d5798c1c35c8c6f20216da80928e4f67022adc2849c145d726c23785bc0d11e
Transaction size: 225 bytes
------------------------------------------------------------
Unloading wallet...
Unloaded wallet: Team_xyz
```

b)legacy1_xyz.py

1. **Wallet Loading**:

   - Loads the existing wallet *Team_xyz*.

2. **Address Retrieval**:

   - Fetches the legacy Bitcoin addresses *B* and *C* that were previously generated by *legacy_xyz.py*.

3. **UTXO Details of Address B**:

   - Retrieves and displays the Unspent Transaction Output (UTXO) details of address *B* from the transaction $A \rightarrow B$.

4. **Transaction Creation (B → C)**:

   - Constructs a new transaction transferring funds from *B* to *C*, utilizing the available UTXO balance.
   - Follows the same procedure as the transaction $A \rightarrow B$.

5. **Transaction Details**:

   - Displays the transaction ID and the total transaction size in virtual bytes (vbytes).

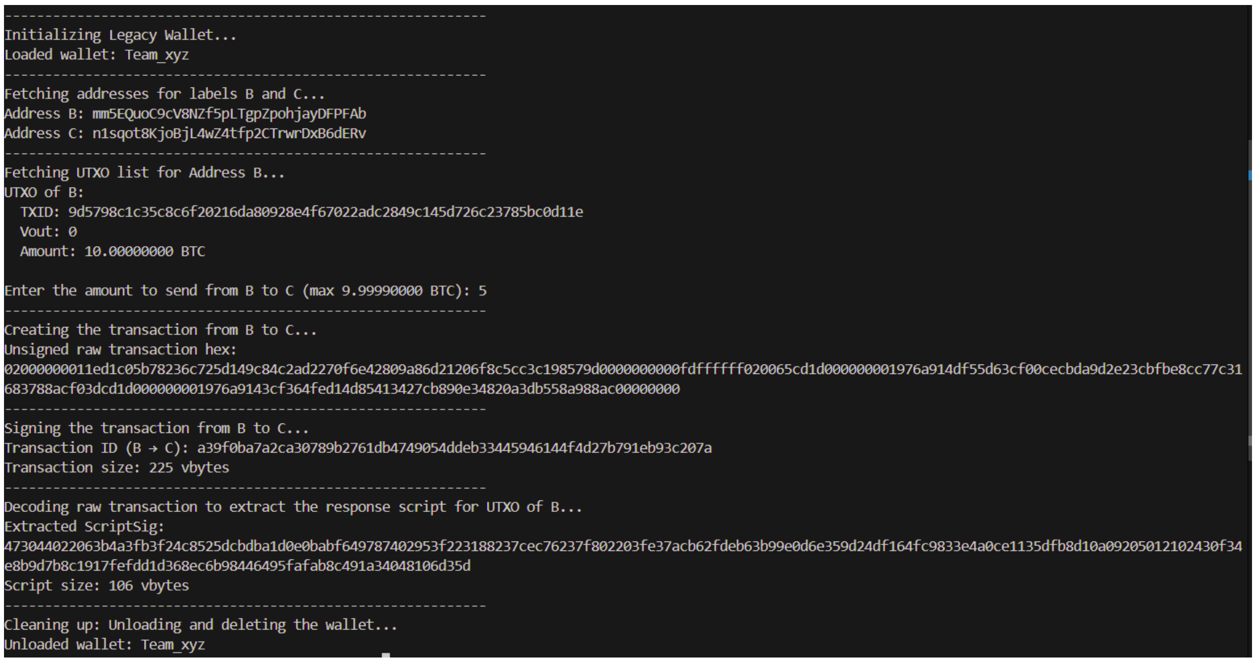
6. **Transaction Decoding**:

- Decodes the transaction $B \longrightarrow C$ to extract the response script (*ScriptSig*) used to unlock the UTXO balance of *B*.
- Displays the size of *ScriptSig* in vbytes.

7. **Wallet Cleanup**:

- Unloads the *Team_xyz* wallet upon completion.

---------------------- Output of legacy1_xyz.py ----------------------



```
------------------------------------------------------------
Initializing Legacy Wallet...
Loaded wallet: Team_xyz
------------------------------------------------------------
Fetching addresses for labels B and C...
Address B: mm5EQuoC9cV8NZf5pLTgpZpohjayDFPFAb
Address C: n1sqot8KjoBjL4wZ4tfp2CTrwrDxB6dERv
------------------------------------------------------------
Fetching UTXO list for Address B...
UTXO of B:
  TXID: 9d5798c1c35c8c6f20216da80928e4f67022adc2849c145d726c23785bc0d11e
  Vout: 0
  Amount: 10.00000000 BTC

Enter the amount to send from B to C (max 9.99990000 BTC): 5
------------------------------------------------------------
Creating the transaction from B to C...
Unsigned raw transaction hex:
02000000011ed1c05b78236c725d149c84c2ad2270f6e42809a86d21206f8c5cc3c198579d0000000000fdffffff020065cd1d000000001976a914df55d63cf00cecbda9d2e23cbfbe8cc77c31
683788acf03dcd1d000000001976a9143cf364fed14d85413427cb890e34820a3db558a988ac00000000
------------------------------------------------------------
Signing the transaction from B to C...
Transaction ID (B → C): a39f0ba7a2ca30789b2761db4749054ddeb33445946144f4d27b791eb93c207a
Transaction size: 225 vbytes
------------------------------------------------------------
Decoding raw transaction to extract the response script for UTXO of B...
Extracted ScriptSig:
473044022063b4a3fb3f24c8525dcbdba1d0e0babf649787402953f223188237cec76237f802203fe37acb62fdeb63b99e0d6e359d24df164fc9833e4a0ce1135dfb8d10a09205012102430f34
e8b9d7b8c1917fefdd1d368ec6b98446495fafab8c491a34048106d35d
Script size: 106 vbytes
------------------------------------------------------------
Cleaning up: Unloading and deleting the wallet...
Unloaded wallet: Team_xyz
```

Work-flow of transactions

1. Transaction A → B

Transaction ID:

9d5798c1c35c8c6f20216da80928e4f67022adc2849c145d726c2378
5bc0d11e

vout	0
Amount	10 BTC
ScriptPubKey	76a9143cf364fed14d85413427cb890e34820a3db558a988ac
Script Size	25 bytes

2. Transaction B → C

Transaction ID:

a39f0ba7a2ca30789b2761db4749054ddeb33445946144f4d27b791eb93c207
a

Transaction size: 25 vbytes

Transfer of 5 BTC from B to C

- The input for this transaction is the UTXO from the previous transaction as:

Referred Transaction ID	9d5798c1c35c8c6f20216da80928e4f67022adc2849c 145d726c23785bc0d11e

vout	0
UTXO Balance unlocked	10 BTC (5 BTC sent to C, remaining coins back to B)
Challenge Script (ScriptPubKey)	76a9143cf364fed14d85413427cb890e34820a3db558a988ac
Response Script (ScriptSig)	473044022063b4a3fb3f24c8525dcbdba1d0e0babf649787402953f223188237cec76237f802203fe37acb62fdeb63b99e0d6e359d24df164fc9833e4a0ce1135dfb8d10a0920501210243 0f34e8b9d7b8c1917fefdd1d368ec6b98446495fafab8c491a34048106d35d
Response Script Size	106 vbytes

Structure of scripts:

Challenge Script (ScriptPubKey)

"76a9143cf364fed14d85413427cb890e34820a3db558a988ac"

- This script ensures that only the owner of Address B (who possesses the corresponding private key) can spend the UTXO
- The structure of this script can be broken down as:

1. **76 (OP_DUP)**
    - Duplicates the top stack item (the public key hash).
2. **a9 (OP_HASH160)**
    - Applies the RIPEMD-160 hash function to the SHA-256 hash of the public key.

3. **14**
    - Indicates that the next 20 bytes represent the **public key hash** (Address B's hash).
4. **3cf364fed14d85413427cb890e34820a3db558a9**
    - The **RIPEMD-160 hash** of Address B's public key.
5. **88 (OP_EQUALVERIFY)**
    - Verifies that the provided public key hash matches the expected value.
6. **ac (OP_CHECKSIG)**
    - Confirms that the provided digital signature is valid for the given public key, ensuring that only the owner of Address B can spend this UTXO.

Response Script (ScriptSig)-

"47304402205d079f3e94e38c77952ce96056d2f8b448f4702278cf918ef0322668637845b90220351
19e712ba40b0cb2bf2be129041204f5a70164c30b132aad6e07136a754ee601210390d7218b8cb6a3e
7aca2c311e1bda33c9afff50b19196b33435686410c35dbdf"

The **ScriptSig** is the unlocking script that provides the necessary cryptographic proof to spend a previously locked UTXO. It includes the following components:

1. **47**
    - Indicates that the following digital signature is **71 bytes** long.
2. **304402205d079f3e94e38c77952ce96056d2f8b448f4702278cf918ef0322668637845b9
022035119e712ba40b0cb2bf2be129041204f5a70164c30b132aad6e07136a754ee6**
    - This is the **DER-encoded digital signature**, composed of two **32-byte** values:
        - R value (first 32 bytes).
        - S value (second 32 bytes).

- This signature is generated by the private key corresponding to **Address B** and proves ownership of the UTXO.

3. **01**
    - The **SIGHASH flag**, which in this case (01), indicates that only this input is signed (SIGHASH_ALL).
4. **21**
    - Specifies that the following public key is **33 bytes** long.
5. **0390d7218b8cb6a3e7aca2c311e1bda33c9afff50b19196b33435686410c35dbdf**
    - This is the **compressed public key** of Address B, which corresponds to the private key that signed the transaction.

Functionality of ScriptSig

The **ScriptSig** is used to unlock the **ScriptPubKey** from the previous transaction. When executed together, they validate that:

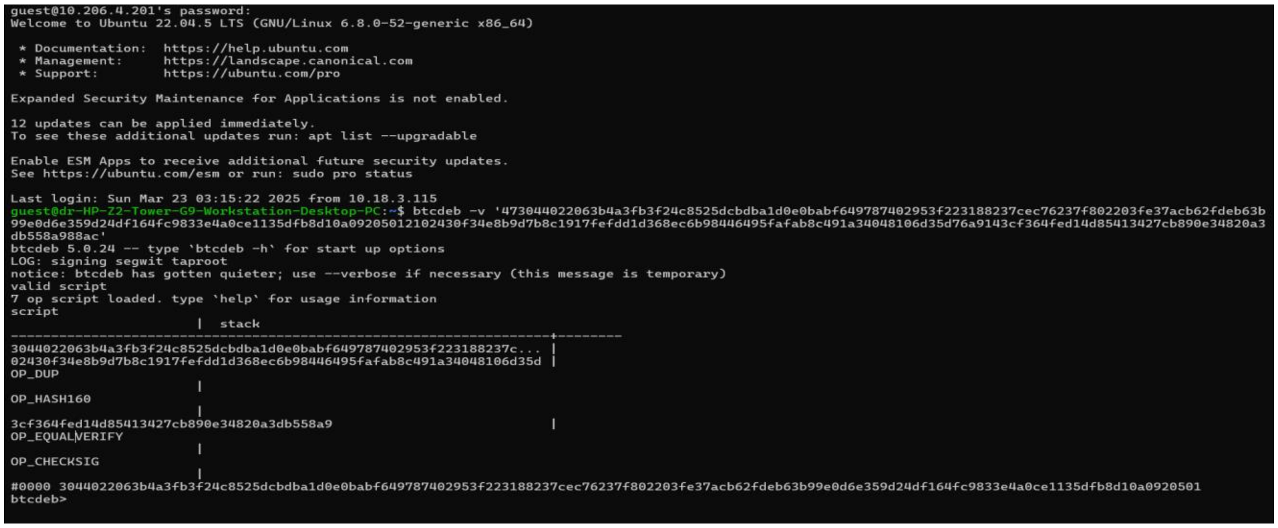
- The provided **public key hash** matches the hash stored in the **ScriptPubKey**.
- The **digital signature** is valid and proves ownership of the UTXO.
  This follows the **Pay-to-PubKey-Hash (P2PKH)** standard, ensuring that only the rightful owner of Address B can spend the funds.
  4o

Validating scripts using Bitcoin Debugger-

1. When spending the UTXO in **Transaction B → C**, the Bitcoin network executes the **combined script**, which consists of:
ScriptSig + ScriptPubKey

2. This script execution follows the **Bitcoin Script** verification process, ensuring that the spender has the correct private key and signature to unlock the UTXO.

3. To validate the scripts, we can use the **btcdeb** (Bitcoin Script Debugger) tool with the following command: *btcdeb -v '<combined_script>'*



```
guest@10.206.4.201's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun Mar 23 03:15:22 2025 from 10.18.3.115
guest@dr-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ btcdeb -v '473044022063b4a3fb3f24c8525dcbdba1d0e0babf649787402953f223188237cec76237f802203fe37acb62fdeb63b
99e0d6e359d24df164fc9833e4a0ce1135dfb8d10a09205012102430f34e8b9d7b8c1917fefdd1d368ec6b98446495fafab8c491a34048106d35d76a9143cf364fed14d85413427cb890e34820a3
db558a988ac'
btcdeb 5.0.24 -- type `btcdeb -h` for start up options
LOG: signing segwit taproot
notice: btcdeb has gotten quieter; use --verbose if necessary (this message is temporary)
valid script
7 op script loaded. type `help` for usage information
script                        |  stack
------------------------------------------------------------------------+--------
3044022063b4a3fb3f24c8525dcbdba1d0e0babf649787402953f223188237c... |
02430f34e8b9d7b8c1917fefdd1d368ec6b98446495fafab8c491a34048106d35d |
OP_DUP
                              |
OP_HASH160
                              |
3cf364fed14d85413427cb890e34820a3db558a9                           |
OP_EQUALVERIFY
                              |
OP_CHECKSIG
                              |
#0000 3044022063b4a3fb3f24c8525dcbdba1d0e0babf649787402953f223188237cec76237f802203fe37acb62fdeb63b99e0d6e359d24df164fc9833e4a0ce1135dfb8d10a0920501
btcdeb>
```

P2SH-SegWit Address Transactions---

Introduction

This report details the process of setting up and executing Segregated Witness (SegWit) transactions using the Bitcoin Core JSON-RPC interface. The script Segwit.py facilitates wallet creation, transaction processing, and script analysis.

Workflow Overview

The script follows these steps:

1. Wallet Initialization

- Connects to the Bitcoin Core node using authproxy.AuthServiceProxy.
- Loads an existing wallet or creates a new one (Team_xyz).
- If the wallet does not exist, it is created and initialized with mined blocks.
- Sets a transaction fee of **0.0001 BTC/kB**.

2. Address Generation

- Generates three P2SH-SegWit addresses labeled A', B', and C'.

3. Funding Address A'

- Funds A' with **10 BTC** using sendtoaddress.
- Mines an additional block to confirm the transaction.

4. Transaction: A' → B'

- Fetches UTXOs for A' and displays the available balance.
- Prompts the user to specify an amount to transfer from A' to B', ensuring:

$$0 < \text{Amount} \leq \text{UTXO(A)} - \text{Mining Fee}$$

- Constructs a raw transaction transferring funds from A' to B'.
- Decodes the raw transaction to extract the scriptPubKey for B'.
- Signs and broadcasts the transaction.
- Displays the transaction ID and size (in vbytes).

5. Transaction: B' → C'

- Fetches UTXOs for B' and displays the available balance.
- Prompts the user to specify an amount to transfer from B' to C'.
- Constructs a raw transaction transferring funds from B' to C'.
- Signs and broadcasts the transaction.
- Displays the transaction ID and size (in vbytes).
- Decodes and analyzes the redeem script and witness data.

6. Cleanup

- Unloads the wallet after transaction processing.

Key Functions

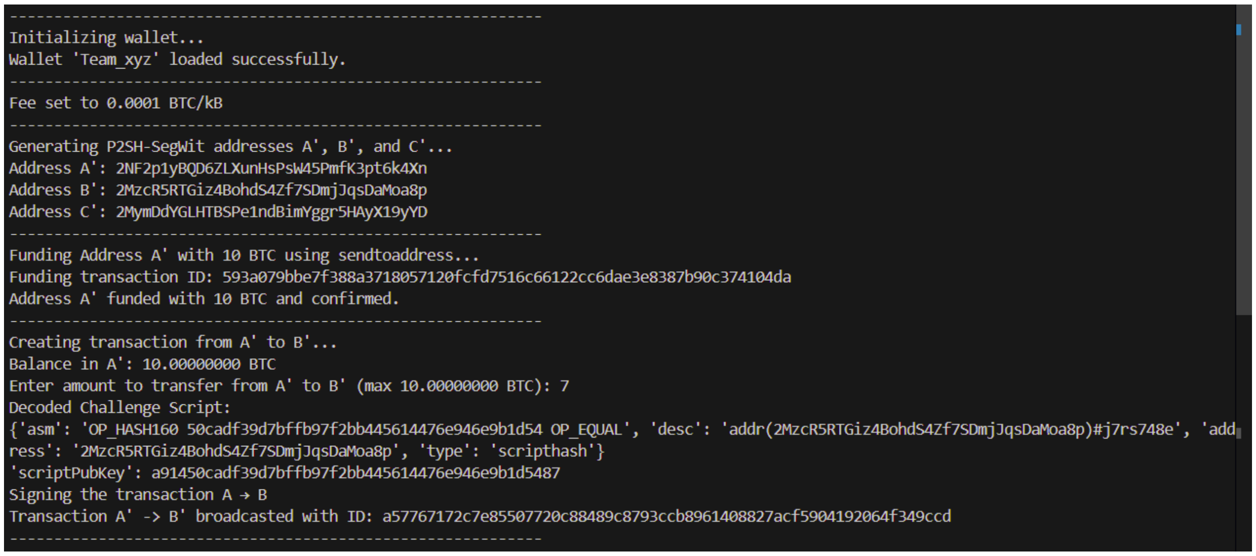
- prompt_for_amount(max_value): Ensures user input adheres to the valid transaction limits.

- compute_script_stats(hex_script): Computes byte size, weight, and virtual size for Bitcoin scripts.
- hash160(data): Computes the RIPEMD160(SHA256(data)) hash, commonly used in Bitcoin addresses.

Conclusion

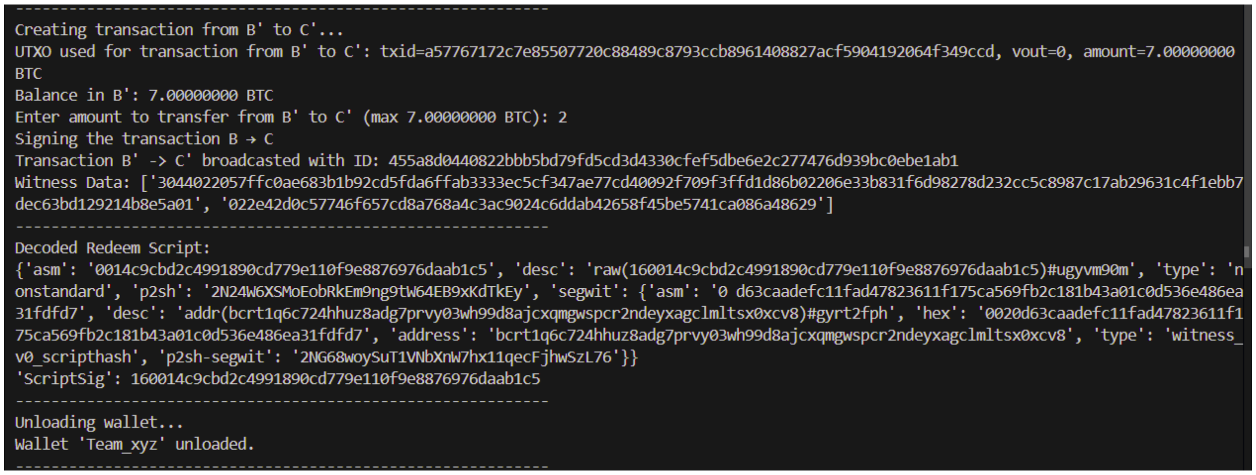
This script efficiently sets up and processes SegWit transactions by leveraging Bitcoin Core's RPC interface. The transactions ensure compliance with UTXO constraints while allowing for effective transaction scripting and verification.

----------------------------------- Output for A’ to B’ ---



```
------------------------------------------------------------
Initializing wallet...
Wallet 'Team_xyz' loaded successfully.
------------------------------------------------------------
Fee set to 0.0001 BTC/kB
------------------------------------------------------------
Generating P2SH-SegWit addresses A', B', and C'...
Address A': 2NF2p1yBQD6ZLXunHsPsW45PmfK3pt6k4Xn
Address B': 2MzcR5RTGiz4BohdS4Zf7SDmjJqsDaMoa8p
Address C': 2MymDdYGLHTBSPe1ndBimYggr5HAyX19yYD
------------------------------------------------------------
Funding Address A' with 10 BTC using sendtoaddress...
Funding transaction ID: 593a079bbe7f388a3718057120fcfd7516c66122cc6dae3e8387b90c374104da
Address A' funded with 10 BTC and confirmed.
------------------------------------------------------------
Creating transaction from A' to B'...
Balance in A': 10.00000000 BTC
Enter amount to transfer from A' to B' (max 10.00000000 BTC): 7
Decoded Challenge Script:
{'asm': 'OP_HASH160 50cadf39d7bffb97f2bb445614476e946e9b1d54 OP_EQUAL', 'desc': 'addr(2MzcR5RTGiz4BohdS4Zf7SDmjJqsDaMoa8p)#j7rs748e', 'add
ress': '2MzcR5RTGiz4BohdS4Zf7SDmjJqsDaMoa8p', 'type': 'scripthash'}
'scriptPubKey': a91450cadf39d7bffb97f2bb445614476e946e9b1d5487
Signing the transaction A → B
Transaction A' -> B' broadcasted with ID: a57767172c7e85507720c88489c8793ccb8961408827acf5904192064f349ccd
------------------------------------------------------------
```

----------------------------------- Output for B’ to C’ ---



```
------------------------------------------------------------
Creating transaction from B' to C'...
UTXO used for transaction from B' to C': txid=a57767172c7e85507720c88489c8793ccb8961408827acf5904192064f349ccd, vout=0, amount=7.00000000
BTC
Balance in B': 7.00000000 BTC
Enter amount to transfer from B' to C' (max 7.00000000 BTC): 2
Signing the transaction B → C
Transaction B' -> C' broadcasted with ID: 455a8d0440822bbb5bd79fd5cd3d4330cfef5dbe6e2c277476d939bc0ebe1ab1
Witness Data: ['3044022057ffc0ae683b1b92cd5fda6ffab3333ec5cf347ae77cd40092f709f3ffd1d86b02206e33b831f6d98278d232cc5c8987c17ab29631c4f1ebb7
dec63bd129214b8e5a01', '022e42d0c57746f657cd8a768a4c3ac9024c6ddab42658f45be5741ca086a48629']
------------------------------------------------------------
Decoded Redeem Script:
{'asm': '0014c9cbd2c4991890cd779e110f9e8876976daab1c5', 'desc': 'raw(160014c9cbd2c4991890cd779e110f9e8876976daab1c5)#ugyvm90m', 'type': 'n
onstandard', 'p2sh': '2N24W6XSMoEobRkEm9ng9tW64EB9xKdTkEy', 'segwit': {'asm': '0 d63caadefc11fad47823611f175ca569fb2c181b43a01c0d536e486ea
31fdfd7', 'desc': 'addr(bcrt1q6c724hhuz8adg7prvy03wh99d8ajcxqmgwspcr2ndeyxagclmltsx0xcv8)#gyrt2fph', 'hex': '0020d63caadefc11fad47823611f1
75ca569fb2c181b43a01c0d536e486ea31fdfd7', 'address': 'bcrt1q6c724hhuz8adg7prvy03wh99d8ajcxqmgwspcr2ndeyxagclmltsx0xcv8', 'type': 'witness_
v0_scripthash', 'p2sh-segwit': '2NG68woySuT1VNbXnW7hx11qecFjhwSzL76'}}
'ScriptSig': 160014c9cbd2c4991890cd779e110f9e8876976daab1c5
------------------------------------------------------------
Unloading wallet...
Wallet 'Team_xyz' unloaded.
------------------------------------------------------------
```

Transaction Workflow for SegWit Implementation

Transaction A' → B'

Transaction ID:
a57767172c7e85507270c88489c8793ccb8961408827acf5904192064f349ccd

Transaction size: 166 vbytes

- **Transfer of 7 BTC from A' to B'**
- **The output (UTXO) of this transaction is stored in Address B'**

Transaction Output Details

Vout	Amount
0	7 BTC

Decoded Challenge Script

ScriptPubKey	Script Size
a91450cadf39d7bffb97f2bb445614476e946e9b1d5487	23 vbytes

Transaction B' → C'

Transaction ID:
455a8d404082bb5bd79fd5cd833cefcf5db6e2c277476d939bc0ebe1ab1

Transaction size: 166 vbytes

- **Transfer of 2 BTC from B' to C'**
- **The input for this transaction is the UTXO from the previous transaction**

Transaction Input Details-

Referred Transaction ID-	Referred Output Index (vout)-
a57767172c7e85507270c88489c8793ccb8961408827acf5904192064f349ccd	0

UTXO Balance Unlocked

- **7 BTC (2 BTC sent to C', remaining 5 BTC sent back to B')**

Challenge Script & Response Script

Challenge Script (ScriptPubKey)	Response Script (ScriptSig)	Script Size
a91450cadf39d7bffb97f2bb445614476e946e9b1d5487	160014c9cbd2c4991890cd779e110f9e8876976daab1c5	23 vbytes

Structure of scripts

- **Challenge Script (ScriptPubKey)**

"a91450cadf39d7bffb97f2bb445614476e946e9b1d5487"

- This script locks funds to a SegWit-compatible redeem script hash. The actual spending requires validation of witness data (signature + public key)
- The structure of this script can be broken down as:

Segment	Label	Instruction
a9	OP_HASH160	Hash the redeem script using SHA-256 + RIPEMD-160
14	N/A	Push 20 bytes, which represents the length of the hashed redeem script.
50cadf39d7bffb97f2bb445614476e946e9b1d54	N/A	Insert a 20-byte hash that corresponds to the redeem script, commonly referred to as the witness program.
87	OP_EQUAL	Confirm that the computed hash is identical to the embedded hash.

- **Response Script (ScriptSig)**

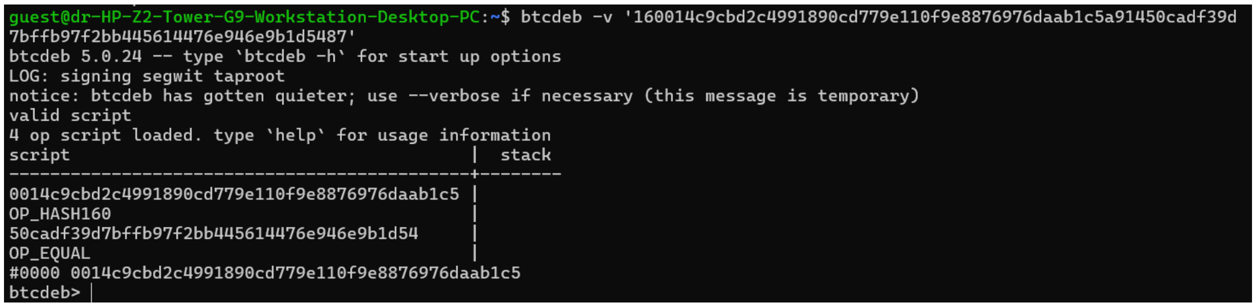
“160014c9cbd2c4991890cd779e110f9e8876976daab1c5”

- This script generates a cryptographic proof, consisting of a signature and a public key, to meet the requirements defined by the ScriptPubKey.
- The composition of this script can be outlined as follows:

Segment	**Instruction**
16	Push 22 bytes, which represent the entire length of the witness program.
0014c9cbd2c4991890cd779e110f9e8876976daab1c5	The witness program begins with 0x00, indicating the SegWit version, followed by 0x14, which signifies that a 20-byte public key hash is included.

Validating scripts using Bitcoin Debugger

The challenge and response scripts for SegWit addresses can be verified using the same process as Legacy address scripts, ensuring that all cryptographic requirements are met.



```
guest@dr-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ btcdeb -v '160014c9cbd2c4991890cd779e110f9e8876976daab1c5a91450cadf39d
7bffb97f2bb445614476e946e9b1d5487'
btcdeb 5.0.24 -- type `btcdeb -h` for start up options
LOG: signing segwit taproot
notice: btcdeb has gotten quieter; use --verbose if necessary (this message is temporary)
valid script
4 op script loaded. type `help` for usage information
script                                       |  stack
---------------------------------------------+--------
0014c9cbd2c4991890cd779e110f9e8876976daab1c5 |
OP_HASH160                                   |
50cadf39d7bffb97f2bb445614476e946e9b1d54     |
OP_EQUAL                                     |
#0000 0014c9cbd2c4991890cd779e110f9e8876976daab1c5
btcdeb>
```