CS 216: Introduction to Blockchain
Assignment 3: Bitcoin Scripting
Date: 13.02.2025
Deadline: 23rd March 2025 (Thursday) by 11:30 PM

---

1. Introduction

Bitcoin scripting allows users to create programmable transactions by leveraging a stack-based, Forth-like scripting language. This assignment delves into Bitcoin transactions using both Legacy (P2PKH) and SegWit (P2SH-P2WPKH) address formats. The primary objectives include:

- Using `bitcoind` and RPC commands for transaction processing.
- Writing and signing transactions programmatically.
- Comparing transaction sizes and analyzing efficiency differences between Legacy and SegWit transactions.

We employed `Python` and `C` to interact with `bitcoind`, conducting all transactions in `regtest` mode to simulate a secure and cost-free Bitcoin environment.

---

2. Setup and Tools Required

To execute this assignment, the following tools and configurations were required:

Software and Libraries:

1. **Bitcoin Core (`bitcoind`)** - Bitcoin's full node software.
2. **Bitcoin CLI (`bitcoin-cli`)** - Command-line interface to interact with `bitcoind`.
3. **Programming Languages:**
    - **Python:** Libraries such as `python-bitcoinlib` or `bitcoinrpc` for transaction handling.
    - **C:** Usage of `libbitcoin` or `curl` for RPC interactions.
4. **Bitcoin Debugger** - To verify and debug Bitcoin scripts.

Configuration in `bitcoin.conf`

To ensure smooth transaction execution, the following configurations were added to `bitcoin.conf`:

paytxfee=0.0001
fallbackfee=0.0002
mintxfee=0.00001
txconfirmtarget=6

After configuring, `bitcoind` was launched in `regtest` mode:

bitcoind -regtest -daemon

Its status was verified using:

bitcoin-cli -regtest getblockchaininfo

---

3. Part 1: Legacy Address Transactions (P2PKH)

Step 1: Wallet Setup and Address Generation

1. Wallet creation and initialization:
   bitcoin-cli -regtest createwallet "testwallet"

Generating three Legacy (P2PKH) addresses:
bitcoin-cli -regtest getnewaddress "LegacyA" "legacy"
bitcoin-cli -regtest getnewaddress "LegacyB" "legacy"

2. bitcoin-cli -regtest getnewaddress "LegacyC" "legacy"

Step 2: Funding Transactions

1. Mining test bitcoins:
   bitcoin-cli -regtest generatetoaddress 101 $(bitcoin-cli -regtest getnewaddress)
2. Sending 10 BTC to `LegacyA`:
   bitcoin-cli -regtest sendtoaddress "LegacyA" 10

Step 3: Creating a Transaction from A to B

1. Constructing a raw transaction:
   bitcoin-cli -regtest createrawtransaction '[{"txid":"<TXID>","vout":0}]' '{"LegacyB":5}'

Signing and broadcasting:
bitcoin-cli -regtest signrawtransactionwithwallet "<RAW_TX>"

2. bitcoin-cli -regtest sendrawtransaction "<SIGNED_TX>"

Step 4: Creating a Transaction from B to C

- Identified UTXO using `listunspent`.
- Created and signed a transaction from B to C.
- Verified the transaction using `decoderawtransaction`.

---

4. Part 2: P2SH-SegWit Address Transactions (P2SH-P2WPKH)

Step 1: Generating SegWit Addresses

bitcoin-cli -regtest getnewaddress "SegwitA" "p2sh-segwit"
bitcoin-cli -regtest getnewaddress "SegwitB" "p2sh-segwit"
bitcoin-cli -regtest getnewaddress "SegwitC" "p2sh-segwit"

Step 2: Creating Transactions Similar to Part 1

- Funded Address A'.
- Created a raw transaction from A' to B'.
- Decoded, signed, and broadcasted it.
- Executed another transaction from B' to C'.

---

5. Part 3: Analysis and Explanation

Comparison of Legacy vs. SegWit Transactions

1. **Transaction Size Comparison:**
   - Legacy (P2PKH) transactions are **larger** due to `ScriptSig` containing a full signature and public key.
   - SegWit (P2SH-P2WPKH) transactions **reduce size** by storing witness data separately.
2. **Script Structure Differences:**
   - **P2PKH (Legacy):** Uses `ScriptSig` in input and `ScriptPubKey` in output.
   - **P2SH-P2WPKH (SegWit):** Uses `Witness` data for unlocking, reducing `ScriptSig` size.
3. **Why SegWit is Better:**
   - **Reduces transaction weight** (lower fees).
   - **Fixes malleability issues.**

- **More efficient block usage.**

Debugging and Verification

- Transactions were verified using `decoderawtransaction`.
- Outputs and execution flow were checked using the Bitcoin Debugger.

---

6. Conclusion

This assignment provided valuable experience with Bitcoin scripting, transaction creation, and an in-depth comparison of different address formats. The results demonstrated that SegWit transactions are more efficient due to their reduced size and enhanced security features. Understanding these differences is crucial for blockchain developers aiming to optimize transaction efficiency and security.