PROJECT REPORT

ON

BITCOIN SCRIPTING

Github Link: https://github.com/keshav-singhal04/Bitcoin-Scripting

Team Name: Synergy

Team Members:
- Kartik Hiranandani (230001037)
- Keshav Singhal (230001039)
- Kumar Prince (230008019)

Part 1: Legacy (P2PKH) Address Transactions

For this part, we have created 2 Python codes, Legacy_1.py and Legacy_2.py

- **Legacy_1.py**

  This code will:

  a) Create a new wallet (or load existing wallet) named Synergy_Legacy
  b) Generate 3 legacy addresses A, B and C
  c) Mine some initial blocks in order to fund address A
  d) Display the UTXO balance of A once it is funded
  e) Ask the user the amount to be transferred from A to B, satisfying the condition:

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

  f) Create a raw transaction transferring coins from A to B
  g) Decode the raw transaction to extract the challenge script for freshly created UTXO of B, i.e. ScriptPubKey and also display its size in vbytes
  h) Sign the transaction A → B and broadcast it on the network
  i) Display the transaction ID and transaction size (in vbytes)
  j) Unload the wallet at the end

Output of Legacy_1.py

```
Created wallet: Synergy_Legacy

Legacy Addresses:
A: mwepamnivpckFqwFJMCS3CHMGWPZrtg11Z
B: mxv3hFoHVH9anxtDDFmyLDSiQxUQVqSbGh
C: mrqcv2FbzirjDCY7tkKmkphPKTSuadLNRW

Mining some initial blocks 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): 20

Creating a raw transaction from A to B ...

Unsigned raw transaction hex:
0200000001e21d364af14538a65f743dde4e1d825d70764b956677a723e82a96b2d5c26b580000000000fdffffff0200943
577000000001976a914bed836920f53016a64caa842778ff4f098e8255b88acf036d0b2000000001976a914b0fee71dba42
db838583b6089c2d63d28e1312e188ac00000000

Decoding raw transaction to extract the challenge script ...

Extracted ScriptPubKey: 76a914bed836920f53016a64caa842778ff4f098e8255b88ac
Script size: 25 vbytes

Signing the transaction A → B ...

Signed transaction hex:
0200000001e21d364af14538a65f743dde4e1d825d70764b956677a723e82a96b2d5c26b58000000006a47304402204b48d
3d301eda3772681cb246dee760379b9fe400ec0ae24e6a6b6e78062420f02201670d5107755373a3a3f6ac187b08ceeea2c
21ee47282f2d43026d8666fef33b012103c8f34d7379c5998143ac1716cd67bba325a477ec5594ea721cf5b897ab5992e5f
dfffffff0200943577000000001976a914bed836920f53016a64caa842778ff4f098e8255b88acf036d0b2000000001976a9
14b0fee71dba42db838583b6089c2d63d28e1312e188ac00000000

Broadcasting the transaction A → B ...

Transaction ID (A → B): a5f9f87a48fd07de12fb0bc6ec5cb0400f26026f9d3a4065f09d9dff01065111
Transaction size: 225 vbytes

Unloaded wallet: Synergy_Legacy
```

- **Legacy_2.py**

This code will:

a) Load the wallet Synergy_Legacy

b) Fetch the legacy addresses B and C created by Legacy_1.py

c) Fetch and display the UTXO details of B from the transaction A → B

d) Create a new transaction B → C funded by this UTXO balance by following the same procedure as that for the transaction A → B

e) Display the transaction ID and transaction size (in vbytes)

f) Decode the transaction B → C to extract the response script to unlock the UTXO balance of B, i.e. ScriptSig and also display its size in vbytes

g) Unload the wallet at the end

Output of Legacy_2.py

```
Loaded wallet: Synergy_Legacy

Address B: mxv3hFoHVH9anxtDDFmyLDSiQxUQVqSbGh
Address C: mrqcv2FbzirjDCY7tkKmkphPKTSuadLNRW

Fetching the UTXO list ...

UTXO of B:
TXID: a5f9f87a48fd07de12fb0bc6ec5cb0400f26026f9d3a4065f09d9dff01065111
Vout: 0
Amount: 20.00000000 BTC

Enter the amount to send from B to C (max 19.99990000 BTC): 10

Creating the transaction from B to C ...

Unsigned raw transaction hex:
02000000011151060 1ff9d9df065403a9d6f02260f40b05cecc60bfb12de07fd487af8f9a5000000000fdffffff0200ca9
a3b000000001976a9147c311c02160127d4a35ba7bc6d77497a171b050388acf0a29a3b000000001976a914bed836920f53
016a64caa842778ff4f098e8255b88ac00000000

Signing the transaction B → C ...

Signed transaction hex:
02000000011151060 1ff9d9df065403a9d6f02260f40b05cecc60bfb12de07fd487af8f9a500000006a47304402205d079
f3e94e38c77952ce96056d2f8b448f4702278cf918ef0322668637845b9022035119e712ba40b0cb2bf2be129041204f5a7
0164c30b132aad6e07136a754ee601210390d7218b8cb6a3e7aca2c311e1bda33c9afff50b19196b33435686410c35dbdff
dfffffff0200ca9a3b000000001976a9147c311c02160127d4a35ba7bc6d77497a171b050388acf0a29a3b000000001976a9
14bed836920f53016a64caa842778ff4f098e8255b88ac00000000

Broadcasting the transaction B → C ...

Transaction ID (B → C): 894003738319a374d40b3740cf94ea0c1a058cdcea12624ea0b29ae9810b6f03
Transaction size: 225 vbytes

Decoding raw transaction to extract the response script ...

Extracted ScriptSig:
47304402205d079f3e94e38c77952ce96056d2f8b448f4702278cf918ef0322668637845b9022035119e712ba40b0cb2bf2
be129041204f5a70164c30b132aad6e07136a754ee601210390d7218b8cb6a3e7aca2c311e1bda33c9afff50b19196b3343
5686410c35dbdf
Script size: 106 vbytes

Unloaded wallet: Synergy_Legacy
```

Work-flow of Legacy transactions

- **Transaction A → B**

Transaction ID:

a5f9f87a48fd07de12fb0bc6ec5cb0400f26026f9d3a4065f09d9dff01065111

Transaction size: 225 vbytes

- Transfer of 20 BTC from A to B
- The output (UTXO) of this transaction is stored in Address B's wallet as:

vout	0
Amount	20 BTC
ScriptPubKey	76a914bed836920f53016a64caa842778ff4f098e8255b88ac
Script Size	25 vbytes

- **Transaction B → C**

Transaction ID:

894003738319a374d40b3740cf94ea0c1a058cdcea12624ea0b29ae9810b6f03

Transaction size: 225 vbytes

- Transfer of 10 BTC from B to C
- The input for this transaction is the UTXO from the previous transaction as:

Referred Transaction ID	a5f9f87a48fd07de12fb0bc6ec5cb0400f26026f9d3a4065f09d9dff01065111
Referred Output Index (vout)	0
UTXO Balance unlocked	20 BTC (10 BTC sent to C, remaining coins back to B)
Challenge Script (ScriptPubKey)	76a914bed836920f53016a64caa842778ff4f098e8255b88ac
Response Script (ScriptSig)	47304402205d079f3e94e38c77952ce96056d2f8b448f4702278cf918ef0322668637845b9022035119e712ba40b0cb2bf2be129041204f5a70164c30b132aad6e07136a754ee601210390d7218b8cb6a3e7aca2c311e1bda33c9afff50b19196b33435686410c35dbdf
Response Script Size	106 vbytes

Structure of Legacy scripts

- **Challenge Script (ScriptPubKey)**

"76a914bed836920f53016a64caa842778ff4f098e8255b88ac"

- 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:

Segment	Label	Instruction
76	OP_DUP	Duplicate the public key
a9	OP_HASH160	Hash the duplicated public key using SHA-256 + RIPEMD-160
14	-	Push 20 bytes (length of the hashed public key)
bed836920f53016a64caa842778ff4f098e8255b	-	20-byte hash of Address B's public key
88	OP_EQUALVERIFY	Verify the computed hash matches the embedded hash
ac	OP_CHECKSIG	Validate the cryptographic signature

- **Response Script (ScriptSig)**

“47304402205d079f3e94e38c77952ce96056d2f8b448f4702278cf918ef032266
8637845b9022035119e712ba40b0cb2bf2be129041204f5a70164c30b132aad6e
07136a754ee601210390d7218b8cb6a3e7aca2c311e1bda33c9afff50b19196b33
435686410c35dbdf”

- This script provides a cryptographic proof (signature + public key) to satisfy the conditions set by the ScriptPubKey
- The structure of this script can be broken down as:

Segment	**Instruction**
47	Length of signature
304402205d079f3e94e38c77952ce960 56d2f8b448f4702278cf918ef03226686 37845b9022035119e712ba40b0cb2bf2 be129041204f5a70164c30b132aad6e07 136a754ee6	ECDSA signature (proving ownership of Address B’s private key)
21	Length of public key
0390d7218b8cb6a3e7aca2c311e1bda33 c9afff50b19196b33435686410c35dbdf	Compressed public key of Address B

Validating Legacy scripts using Bitcoin Debugger

- When spending the UTXO (Transaction B → C), the Bitcoin network executes the combined script: ScriptSig + ScriptPubKey
- We can validate the scripts by running command:

  btcdeb -v '*<combined_script>*'

```
PS C:\Users\Kesha> ssh guest@10.206.4.201
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 00:45:59 2025 from 10.18.7.102
guest@dr-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ btcdeb -v '47304402205d079f3e94e
38c77952ce96056d2f8b448f4702278cf918ef0322668637845b9022035119e712ba40b0cb2bf2be12
9041204f5a70164c30b132aad6e07136a754ee601210390d7218b8cb6a3e7aca2c311e1bda33c9afff
50b19196b33435686410c35dbdf76a914bed836920f53016a64caa842778ff4f098e8255b88ac'
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 tem
porary)
valid script
7 op script loaded. type `help` for usage information
script                                                          |  stack
----------------------------------------------------------------+--------
304402205d079f3e94e38c77952ce96056d2f8b448f4702278cf918ef032266... |
0390d7218b8cb6a3e7aca2c311e1bda33c9afff50b19196b33435686410c35dbdf |
OP_DUP                                                          |
OP_HASH160                                                      |
bed836920f53016a64caa842778ff4f098e8255b                        |
OP_EQUALVERIFY                                                  |
OP_CHECKSIG                                                     |
#0000 304402205d079f3e94e38c77952ce96056d2f8b448f4702278cf918ef0322668637845b90220
35119e712ba40b0cb2bf2be129041204f5a70164c30b132aad6e07136a754ee601
btcdeb>
```

Part 2: P2SH-SegWit Address Transactions

For this part, we have created a single Python code SegWit.py which will:

a) Create a new wallet (or load existing wallet) named Synergy_SegWit
b) Generate 3 SegWit addresses A, B and C
c) Mine some initial blocks in order to fund address A
d) Display the UTXO balance of A once it is funded
e) Ask the user the amount to be transferred from A to B, satisfying the condition:

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

f) Create a raw transaction transferring coins from A to B
g) Decode the raw transaction to extract the challenge script for freshly created UTXO of B, i.e. ScriptPubKey and also display its size in vbytes
h) Sign the transaction A → B and broadcast it on the network
i) Display the transaction ID and transaction size (in vbytes)
j) Fetch and display the UTXO details of B from the transaction A → B
k) Create a new transaction B → C funded by this UTXO balance by following the same procedure as that for the transaction A → B
l) Display the transaction ID and transaction size (in vbytes)
m) Decode the transaction B → C to extract the response script to unlock the UTXO balance of B, i.e. ScriptSig and also display its size in vbytes
n) Unload the wallet at the end

Output of SegWit.py

```
Created wallet: Synergy_SegWit

SegWit Addresses:
A: 2MtnJpm9GXHm8adSUTgkN7QGuWqF7sR4wN9
B: 2NDsKuRYKVBorUgkmhhEGk4amEs3AFekh7a
C: 2MwS7ryD9fP1hRYkcapVN8UeMbCtcntjfsq

Mining some initial blocks 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): 20

Creating a raw transaction from A to B ...

Unsigned raw transaction hex:
020000000125f3ac70ccd51a2ad614723d96c6221fd1b83b181d4275f41e4db7e0c7015e030000000000fdffffff0200943577000000
017a914e2366dcc691d7984c9b6d951a54deeeef63ed89387f036d0b20000000017a91410d90eae18aedb77f2a4730d55e70ccc9738fe
de8700000000

Decoding the transaction A → B to extract challenge script ...

Extracted ScriptPubKey: a914e2366dcc691d7984c9b6d951a54deeeef63ed89387
Script size: 23 vbytes

Signing the transaction A → B ...

Signed transaction hex:
020000000010125f3ac70ccd51a2ad614723d96c6221fd1b83b181d4275f41e4db7e0c7015e03000000000171600145e808d9209b0d95
312e6520b9cfe270fd9cf15a3fdfffffff02009435770000000017a914e2366dcc691d7984c9b6d951a54deeeef63ed89387f036d0b200
00000017a91410d90eae18aedb77f2a4730d55e70ccc9738fede87024730440220327bbf1d11b5af5a711d68dd1a6cd7eb74b84374f53
0522cac79addada1bff1a0220185ab305c9f16f57037b975e7c655052be4933eadd4bbbc2f50d53901c2cbbd2012102a693134cfc7f7e
39a5e8a0298ddb7fd9e5dc1f9b5a1c8412b08d231976af8bed00000000

Brodcasting the transaction A → B ...

Transaction ID (A → B): 8b6f3f2359440ad9fca32d97a49bc6f92586b04a0afa67a7c8645593754825b2
Transaction size: 166 vbytes
```

```
Fetching the UTXO list  ...

UTXO of B:
TXID: 8b6f3f2359440ad9fca32d97a49bc6f92586b04a0afa67a7c8645593754825b2
Vout: 0
Amount: 20.00000000 BTC

Enter the amount to send from B to C (max 19.99990000 BTC): 10

Creating the transacation from B to C ...

Unsigned raw transaction hex:
0200000001b2254875935564c8a767fa0a4ab08625f9c69ba4972da3fcd90a4459233f6f8b0000000000fdffffff0200ca9a3b0000000
017a9142deff703c197ecd0ae36bae063a10d952866bf4987f0a29a3b0000000017a914e2366dcc691d7984c9b6d951a54deeeef63ed8
938700000000

Signing the transaction B → C ...

Signed transaction hex:
02000000000101b2254875935564c8a767fa0a4ab08625f9c69ba4972da3fcd90a4459233f6f8b0000000017160014a405007a7d990b3
ea801908a7e0256536b47b395fdffffff0200ca9a3b0000000017a9142deff703c197ecd0ae36bae063a10d952866bf4987f0a29a3b00
00000017a914e2366dcc691d7984c9b6d951a54deeeef63ed893870247304402200806630b23c68ce7239f3f5a37f1fb0ae9dc996a3047
8f29f68a2b897cda709a90220591eeee6b69f1697e17ab0e446ecc3fbcca2abe8f6088b0d604aeb3bb0695cc8012102c0e301a5fe963d
1ccd3a6726288bbda247de54e561c796358e339da541a1f83600000000

Brodcasting the transaction B → C ...

Transaction ID (B → C): 2ce8129c8997cbbdd2e7411ec1454f03f7c22c67a0cef83e57d47cc702af2808
Transaction size: 166 vbytes

Decoding the transaction B → C to extract response script ...

Extracted ScriptSig: 160014a405007a7d990b3ea801908a7e0256536b47b395
Script size: 23 vbytes

Unloaded wallet: Synergy_SegWit
```

Work-flow of SegWit transactions

- **Transaction A → B**

Transaction ID:

8b6f3f2359440ad9fca32d97a49bc6f92586b04a0afa67a7c8645593754825b2

Transaction size: 166 vbytes

- Transfer of 20 BTC from A to B
- The output (UTXO) of this transaction is stored in Address B's wallet as:

vout	0
Amount	20 BTC
ScriptPubKey	a914e2366dcc691d7984c9b6d951a54deeeef63ed89387
Script Size	23 vbytes

- **Transaction B → C**

Transaction ID:

2ce8129c8997cbbdd2e7411ec1454f03f7c22c67a0cef83e57d47cc702af2808

Transaction size: 166 vbytes

- Transfer of 10 BTC from B to C
- The input for this transaction is the UTXO from the previous transaction as:

Referred Transaction ID	a914e2366dcc691d7984c9b6d951a54deeeef63ed89387
Referred Output Index (vout)	0
UTXO Balance unlocked	20 BTC (10 BTC sent to C, remaining coins back to B)
Challenge Script (ScriptPubKey)	a914e2366dcc691d7984c9b6d951a54deeeef63ed89387
Response Script (ScriptSig)	160014a405007a7d990b3ea801908a7e0256536b47b395
Response Script Size	23 vbytes

Structure of SegWit scripts

- **Challenge Script (ScriptPubKey)**

“a914e2366dcc691d7984c9b6d951a54deeeef63ed89387”

- 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	-	Push 20 bytes (length of the hashed redeem script)
e2366dcc691d7984c9b6 d951a54deeeef63ed893	-	20-byte hash of the redeem script (witness program)
87	OP_EQUAL	Verify the computed hash matches the embedded hash

- **Response Script (ScriptSig)**

“160014a405007a7d990b3ea801908a7e0256536b47b395”

- This script provides a cryptographic proof (signature + public key) to satisfy the conditions set by the ScriptPubKey
- The structure of this script can be broken down as:

Segment	**Instruction**
16	Push 22 bytes (length of the witness program)
0014a405007a7d990b3ea8 01908a7e0256536b47b395	Witness program: 0x00 (SegWit version), 0x14 (20-byte public key hash)

Validating SegWit scripts using Bitcoin Debugger

We can validate the challenge and response scripts of SegWit addresses using the same procedure followed for validation of Legacy address scripts.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PS
Windows

PS C:\Users\Kesha>  ssh guest@10.206.4.201
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 02:41:37 2025 from 10.18.4.229
guest@dr-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ btcdeb -v '160014a405007a7d990b3
ea801908a7e0256536b47b395a914e2366dcc691d7984c9b6d951a54deeeef63ed89387'
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 tem
porary)
valid script
4 op script loaded. type `help` for usage information
script                                      |  stack
--------------------------------------------+--------
0014a405007a7d990b3ea801908a7e0256536b47b395 |
OP_HASH160                                  |
e2366dcc691d7984c9b6d951a54deeeef63ed893     |
OP_EQUAL                                    |
#0000 0014a405007a7d990b3ea801908a7e0256536b47b395
btcdeb>
```

Part 3: Analysis and Explanation

Size Comparison

Size (in vbytes)	Legacy Addresses	SegWit Addresses
Transaction size	225	166
ScriptPubKey size	25	23
ScriptSig size	106	23

Evidently, SegWit addresses led to a reduction in transaction size and script size.

Script Structure Comparison

Legacy Addresses	SegWit Addresses
- Signatures and public keys are embedded directly in the transaction's ScriptSig, bloating the transaction size - Both the sender and receiver's public key hashes are stored in the transaction body	- Critical validation data (signatures, public keys) is stored in a separate *witness field*, not counted as heavily toward transaction size - Only the redeem script hash is embedded in the transaction body, reducing redundancy

Why SegWit transactions are smaller?

- **Witness Discount**: Signature data (witness) is counted at 1/4th the weight of non-witness data
- **Simpler Scripts**: Eliminates redundant opcodes like OP_DUP and OP_CHECKSIG
- **Data Separation**: Moves signatures/public keys to the witness field, reducing ScriptSig size

Benefits of SegWit transactions

- **Lower Fees**: Smaller size → reduced transaction costs
- **Scalability**: Increased block capacity (more transactions per block)
- **Security**: Fixes transaction malleability by isolating witness data