CS60 21S: Final Project

“Dartmouth Extra Credit Points”: P2P Blockchain Cryptocurrency Wallet

Group 3: Error 404

Jason Chen, Grace King, Norman Kuang, Rehoboth Okorie

Blockchain API

The fundamental building components of the blockchain are blocks, which are linked together like a linked list to form the “blockchain”. In addition, because we are building a crypto wallet as our application, there is a third building component - transactions - which are included in blocks.

The blockchain API design contains three classes:

1. **Transaction** - the Transaction object that make up the data stored in blocks (stored in block.transactions class variable) that comprise the blockchain program
2. **Block** - the Block object that are linked to form the blockchain, and associated functions
3. **Blockchain** - the Blockchain that stores the chain of blocks, pending transactions and contains the majority of the functions that control the

Transaction Class

The transaction class contains 4 objects:

1. **sender_addr** - the address of the peer sending the transaction
2. **sender_priv_key** - the private key of the sending peer, which is used to validate transactions
3. **receiver_addr** - the address of the peer receiving the transaction
4. **amount** - the amount of currency being sent in the transaction
5. **timestamp** - the date and time (in UTC) that the transaction was created in the format: YYYY-MM-DD HH:MM:SS.µS
    - Example: 2021-05-31 15:38:58.931800

The transactions are time stamped always in Coordinated Universal Time (UTC) to maintain consistencies across all users regardless of location.

Block Class

The block class contains 6 objects:

1. **index** - an integer that serves as the block ID and its index in the chain
2. **nonce** - a positive integer that forces the hash of the block to begin with a certain number of 0 bits, also known as the difficulty; if the nonce does not force the block to begin with the specified number of 0 bits, then the block and its transactions are invalid
3. **transactions** - an array containing all the pending transactions at the time the block was created
4. **hash** - the hash of the block
5. **prev_hash** - the hash of the previous block
6. **timestamp** - the date and time (in UTC) that the block was created

Block Functions

- Hash_block
- Num_transactions
- block_transactions

Blockchain Class

The blockchain class contains 4 objects:

1. **chain** - an array that each block is added to; the blockchain
2. **pending_transactions** - an array of all transactions that have not yet been added to a block
3. **nodes** - gives the chain the ability to store multiple nodes
4. **build_genesis** - creates the initial block in the chain

Blockchain Functions

- init
- build_genesis
- add_block
- create_trans
- mine
- validate_proof_of_work
- proof_of_work
- block_validity
- chain_validity
- replace_chain
- last_block
- last_transaction
- pending_transactions
- full_chain

P2P Network

Tracker.py

- Create a tracker object
- Continuously listen for incoming connection requests from newly joining peers
    - Once connection is established, start a new thread to communicate with the peer
        - Receive an initial message from peer containing the peer's name and port number
        - Broadcast message to all other peers in network to add this newly joined peer
        - Create a `peer` object (a helper object) to track information regarding the peer, and the object into a dictionary containing all peers in network

P2P Network

Peer.py

- Create a peer object
- Start a thread that receives messages from the tracker
  - Send the tracker its name and port number to be officially added
  - Receive and process requests from the tracker
- Continuously listen for incoming connection requests from other peers
  - When connection is established, create a thread to receive messages from the peer

P2P Protocol

Messages sent from peer-to-peer and peer-to-tracker will be formatted in a dictionary.

Each message will have a "type" key in the dictionary to denote the purpose of the message. Some "types" include:

- `Block` to send a block
- `blockchain` to send a blockchain
- `add` to add a peer
- `remove` to remove a peer
- `change` to change name or port number
- `joined` to denote successful joining

P2P Protocol

Each message will also have a `name` key in the dictionary to track the sender of the message.

Each dictionary may have more keys and values depending on the type of message. For example, if the type is "blockchain", there will be a key and value to store the blockchain of the peer. If the type is "remove", there will be a key and value to track the name of the peer that should be removed.

Blockchain Synchronization Between Peers

There are three possible scenarios in which the peer blockchains can be desynchronized:

1) A new peer joins the network.
2) Two blocks are mined at the same time by two different peers. (Forking)
3) Corruption of one peer's blockchain, either modifying nonce/hash/transactions/index.

Scenario 1

When a new peer joins the network, it will request a copy of the current blockchain from another peer. Theoretically, all of the peers will have the same copy of the same blockchain; if they don't, then scenarios 2 and 3 will detect the inconsistency and synchronize the blockchains before sending an updated copy to the new peer.

Scenario 2

When two blocks are mined at the same time by two different peers, these blocks will contain the exact same index and timestamp. Thus, when the neighboring peers add these blocks to their chain, the first add will be successfully (regardless which block it adds first) while the addition of the second block signifies an invalid chain. In this case, the blockchain adds the block anyway while correcting the indices such that they are sequential.

When a peer detects invalidity, it will attempt to update its blockchain to a copy of the network's majority. This is done by requesting all of its neighboring peers to broadcast their last block; we can tally up the various chains in the network by looking at the hash of each peer's final block. Equivalent blockchains will have equivalent hashes in the last block. Once a majority has been determined, if the peer observed invalidity, it will request a copy of the blockchain from any one of the majority chain holders. In the event of mining two blocks at the same time, all peers will observe invalidity.

Scenario 3

This scenario is dealt with similar to scenario 2, except this time only one chain is invalid instead of all. Once all of the neighbor peers broadcast their chain, the peer can then request a blockchain from one of the majority peers that have the correct blo

Cryptocurrency Wallet

Wallet Features

- RSA generated public and private key pair for secure transactions
  a. The public key stored in your wallet will be used to receive the funds and can be searched for in the distributed ledger
  b. The private key stored in your wallet is used to sign transactions and prove that the user owns the associated public key. It is also used to match the public address the currency to unlock and utilize funds
- Signing of transactions and validation of signatures for additional security
- Initial balance of 100 coins

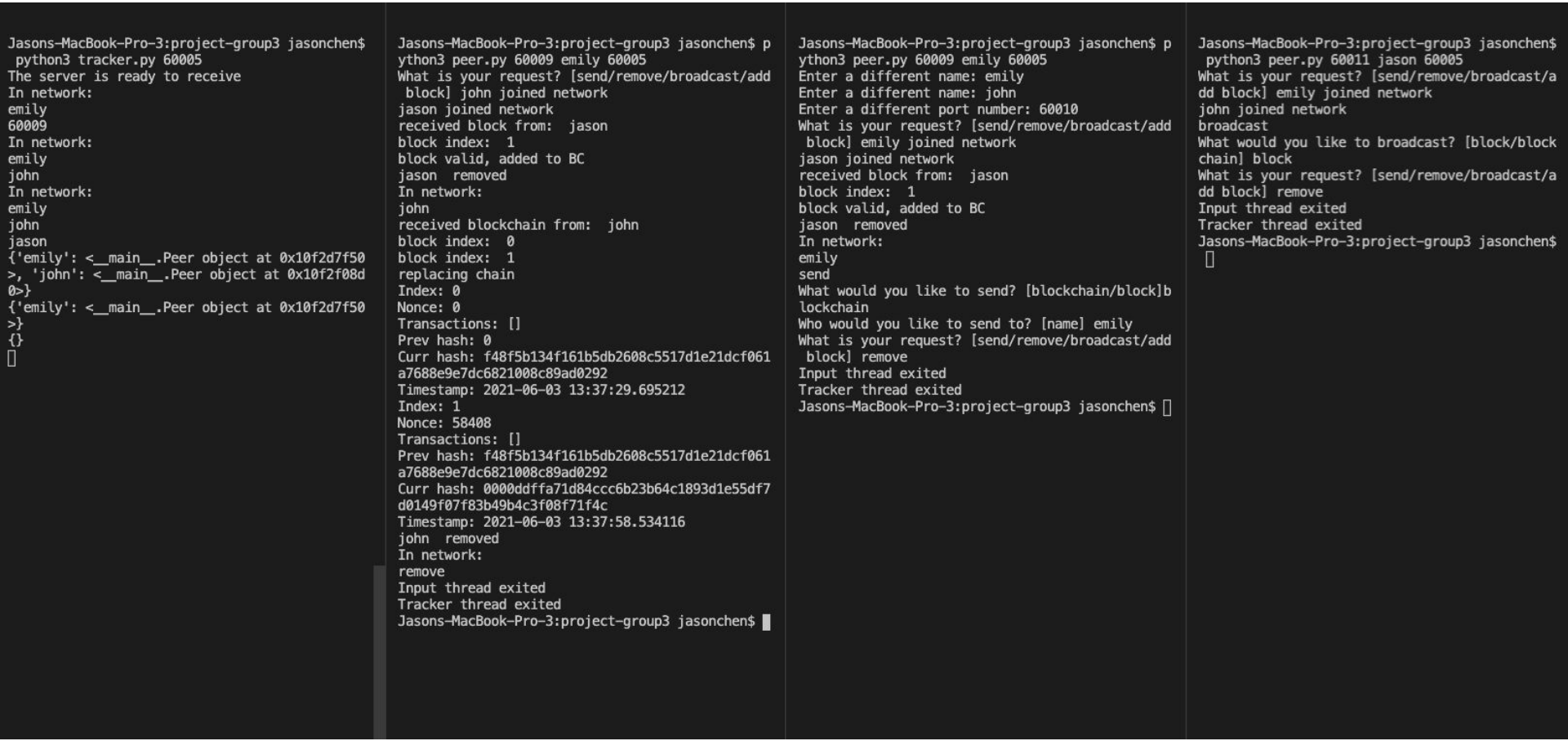
Cryptocurrency Wallet

Wallet Functions

- init
- address
- sign
- verify_signature
- getKeys
- deposit
- withdraw

Demo

P2P Testing


Jasons-MacBook-Pro-3:project-group3 jasonchen$ python3 tracker.py 60005
The server is ready to receive
In network:
emily
60009
In network:
emily
john
In network:
emily
john
jason
{'emily': <__main__.Peer object at 0x10f2d7f50>, 'john': <__main__.Peer object at 0x10f2f08d0>}
{'emily': <__main__.Peer object at 0x10f2d7f50>}
{}
Jasons-MacBook-Pro-3:project-group3 jasonchen$ python3 peer.py 60009 emily 60005
What is your request? [send/remove/broadcast/add block] john joined network
jason joined network
received block from: jason
block index: 1
block valid, added to BC
jason removed
In network:
john
received blockchain from: john
block index: 0
block index: 1
replacing chain
Index: 0
Nonce: 0
Transactions: []
Prev hash: 0
Curr hash: f48f5b134f161b5db2608c5517d1e21dcf061a7688e9e7dc6821008c89ad0292
Timestamp: 2021-06-03 13:37:29.695212
Index: 1
Nonce: 58408
Transactions: []
Prev hash: f48f5b134f161b5db2608c5517d1e21dcf061a7688e9e7dc6821008c89ad0292
Curr hash: 0000ddffa71d84ccc6b23b64c1893d1e55df7d0149f07f83b49b4c3f08f71f4c
Timestamp: 2021-06-03 13:37:58.534116
john removed
In network:
remove
Input thread exited
Tracker thread exited
Jasons-MacBook-Pro-3:project-group3 jasonchen$
Jasons-MacBook-Pro-3:project-group3 jasonchen$ python3 peer.py 60009 emily 60005
Enter a different name: emily
Enter a different name: john
Enter a different port number: 60010
What is your request? [send/remove/broadcast/add block] emily joined network
jason joined network
received block from: jason
block index: 1
block valid, added to BC
jason removed
In network:
emily
send
What would you like to send? [blockchain/block]blockchain
Who would you like to send to? [name] emily
What is your request? [send/remove/broadcast/add block] remove
Input thread exited
Tracker thread exited
Jasons-MacBook-Pro-3:project-group3 jasonchen$
Jasons-MacBook-Pro-3:project-group3 jasonchen$ python3 peer.py 60011 jason 60005
What is your request? [send/remove/broadcast/add block] emily joined network
john joined network
broadcast
What would you like to broadcast? [block/blockchain] block
What is your request? [send/remove/broadcast/add block] remove
Input thread exited
Tracker thread exited
Jasons-MacBook-Pro-3:project-group3 jasonchen$

Wallet Testing

```
Jasons-MacBook-Pro-3:project-group3 jasonchen$
 python3 Tracker.py 60005
The server is ready to receive
In network:
emily
In network:
emily
jack
In network:
emily
jack
andy
```

```
Jasons-MacBook-Pro-3:project-group3 jasonchen$ p
ython3 peer.py 60009 emily 60005
Your balance is 100
What is your request? deposit / transfer / withd
rawal / mine / exit: jack joined network
andy joined network
withdrawal
How much would you like to withdraw? eighty
Error: not a number
What is your request? deposit / transfer / withd
rawal / mine / exit: withdrawal
How much would you like to withdraw? 800
ERROR: Your balance is 100. Please re-enter amou
nt:  20
Your balance is now: 80.0
To process transaction, enter 'mine'.
What is your request? deposit / transfer / withd
rawal / mine / exit: mine
Block mined and broadcasted
What is your request? deposit / transfer / withd
rawal / mine / exit: received block from:  jack
block valid, added to BC
received block from:  andy
block valid, added to BC
Received 40.0 from andy. Your balance is: 120.0
```

```
Jasons-MacBook-Pro-3:project-group3 jasonchen$ p
ython3 peer.py 60010 jack 60005
Your balance is 100
emily joined network
What is your request? deposit / transfer / withd
rawal / mine / exit: andy joined network
received block from:  emily
block valid, added to BC
deposit
How much would you like to deposit? twenty
Error: not a number
What is your request? deposit / transfer / withd
rawal / mine / exit: deposit
How much would you like to deposit? 200
Your balance is now: 300.0
To process transaction, enter 'mine'.
What is your request? deposit / transfer / withd
rawal / mine / exit: mine
Block mined and broadcasted
What is your request? deposit / transfer / withd
rawal / mine / exit: received block from:  andy
block valid, added to BC
Received 15.0 from andy. Your balance is: 315.0
```

```
Jasons-MacBook-Pro-3:project-group3 jasonchen$
 python3 peer.py 60011 andy 60005
Your balance is 100
emily joined network
jack joined network
What is your request? deposit / transfer / wit
hdrawal / mine / exit: received block from:  e
mily
block valid, added to BC
received block from:  jack
block valid, added to BC
transfer
Please enter: [recipient] [amount] emily 10
Your balance is now: 90.0
To process transaction, enter 'mine'.
What is your request? deposit / transfer / wit
hdrawal / mine / exit: transfer
Please enter: [recipient] [amount] jason 2000
Recipient not in network. Please enter a diffe
rent recipient: jack
Cannot transfer 2000.0. Your balance is 90.0.
Enter a smaller amount: 15
Your balance is now: 75.0
To process transaction, enter 'mine'.
What is your request? deposit / transfer / wit
hdrawal / mine / exit: transfer
Please enter: [recipient] [amount] emily 30
Your balance is now: 45.0
To process transaction, enter 'mine'.
What is your request? deposit / transfer / wit
hdrawal / mine / exit: mine
Block mined and broadcasted
What is your request? deposit / transfer / wit
hdrawal / mine / exit:
```

Graphical User Interface

Wallet

Transactions Signout

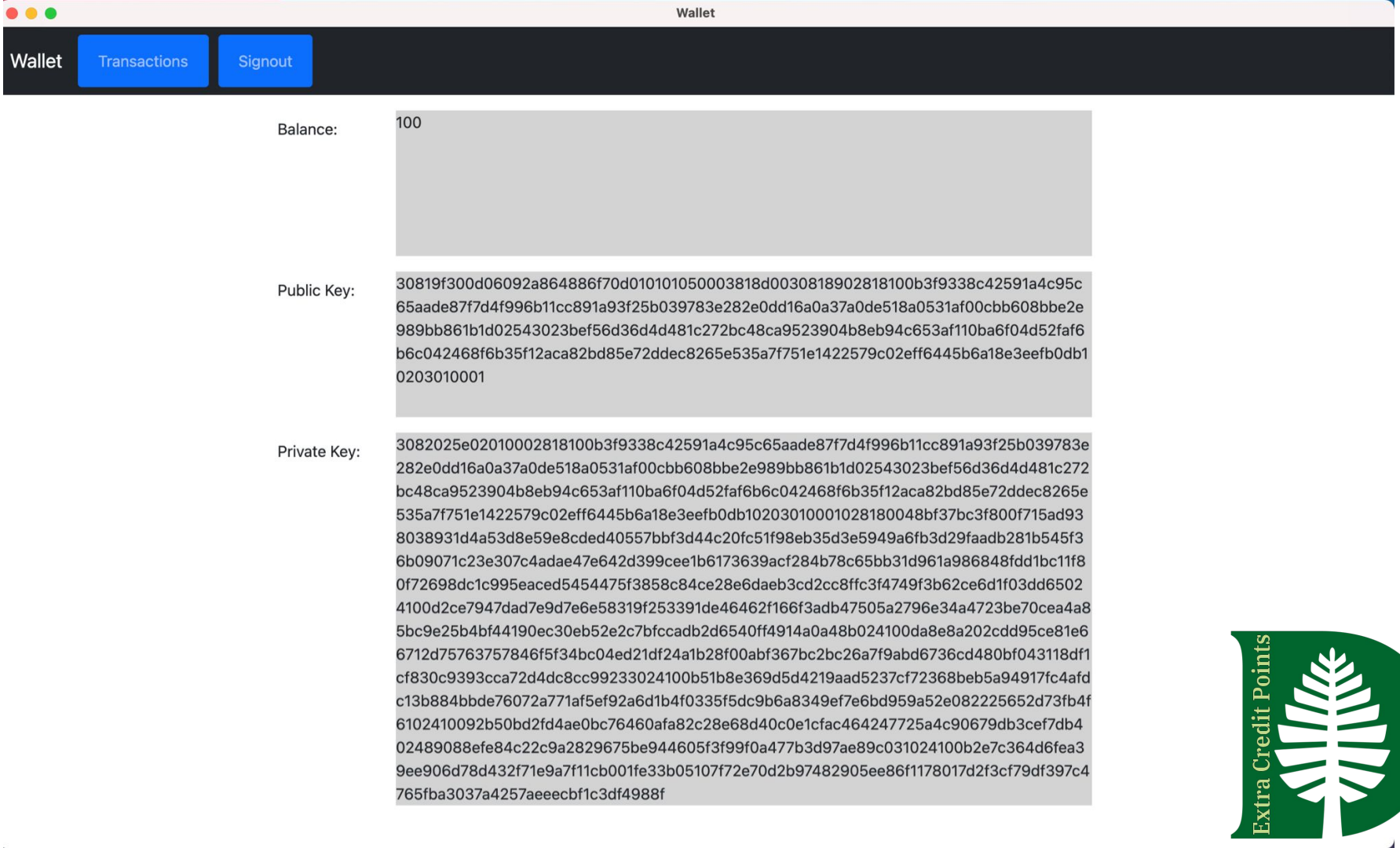
Wallet Generator

Click on the button bellow to generate your block chain wallet

Generate Wallet

Public Key:

Private Key:


Wallet
Wallet
Transactions
Signout
Balance:
100
Public Key:
30819f300d06092a864886f70d010101050003818d0030818902818100b3f9338c42591a4c95c
65aade87f7d4f996b11cc891a93f25b039783e282e0dd16a0a37a0de518a0531af00cbb608bbe2e
989bb861b1d02543023bef56d36d4d481c272bc48ca9523904b8eb94c653af110ba6f04d52faf6
b6c042468f6b35f12aca82bd85e72ddec8265e535a7f751e1422579c02eff6445b6a18e3eefb0db1
0203010001
Private Key:
3082025e02010002818100b3f9338c42591a4c95c65aade87f7d4f996b11cc891a93f25b039783e
282e0dd16a0a37a0de518a0531af00cbb608bbe2e989bb861b1d02543023bef56d36d4d481c272
bc48ca9523904b8eb94c653af110ba6f04d52faf6b6c042468f6b35f12aca82bd85e72ddec8265e
535a7f751e1422579c02eff6445b6a18e3eefb0db1020301000102818004 8bf37bc3f800f715ad93
8038931d4a53d8e59e8cded40557bbf3d44c20fc51f98eb35d3e5949a6fb3d29faadb281b545f3
6b09071c23e307c4adae47e642d399cee1b6173639acf284b78c65bb31d961a986848fdd1bc11f8
0f72698dc1c995eaced5454475f3858c84ce28e6daeb3cd2cc8ffc3f4749f3b62ce6d1f03dd6502
4100d2ce7947dad7e9d7e6e58319f253391de46462f166f3adb47505a2796e34a4723be70cea4a8
5bc9e25b4bf44190ec30eb52e2c7bfccadb2d6540ff4914a0a48b024100da8e8a202cdd95ce81e6
6712d75763757846f5f34bc04ed21df24a1b28f00abf367bc2bc26a7f9abd6736cd480bf043118df1
cf830c9393cca72d4dc8cc99233024100b51b8e369d5d4219aad5237cf72368beb5a94917fc4afd
c13b884bbde76072a771af5ef92a6d1b4f0335f5dc9b6a8349ef7e6bd959a52e082225652d73fb4f
6102410092b50bd2fd4ae0bc76460afa82c28e68d40c0e1cfac464247725a4c90679db3cef7db4
02489088efe84c22c9a2829675be944605f3f99f0a477b3d97ae89c031024100b2e7c364d6fea3
9ee906d78d432f71e9a7f11cb001fe33b05107f72e70d2b97482905ee86f1178017d2f3cf79df397c4
765fba3037a4257aeeecbf1c3df4988f
Extra Credit Points

Wallet

Wallet

Wallet Signout

Transactions to be added to the next block

Show 10 entries

#	Recipient Address	Sender Address	Value
No data available in table			

Showing 0 of 0 entries

Mine

Transactions to be added to the next block

Show 10 entries

#	Recipient Address	Sender Address	Value	Timestamp	Block
No data available in table					

Showing 0 of 0 entries