DOS Project 4.1

Surya Sudharshan(UF ID:5019-3163)

Mrigank Shekhar(UF ID:9428-9219)

Problem Statement:

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

Implementation:

A simple blockchain and its protocol is simulated to support the functionalities of mining, transactions and wallets. The protocol is built using Wallets, Transactions and Mining.

Wallet

The wallet is basically an like an account which abstracts functionalities related with transactions.

The basic core structures for a block is as follows:

1. Header - which holds state data like previous block’s hash,the nonce value, state of the merkle tree, difference for incentive calculation and transactions merkle.
2. Transaction - holds the necessary details to validate and process a transaction, it holds an ID, origin account, destination account, the amount and the digital signature of the transaction.

It supports the following functionalities:

1. Send coins - Wallet needs an origin account, destination account amount and the originator's private key to perform a transaction of sending coins.
2. Check amount - Wallet also supports checking the account balance.

Transaction

A transaction is a transfer of coin value. Transaction happen between wallets.
A transaction can be done using send coin method supported by wallets.
As a security measure, overspending transactions and double spending transactions do not get added to the chain, thus not changing the chain state.

Mining

Mining is the process of adding transaction records to coin's public ledger of past transactions or blockchain. This ledger of past transactions is called the blockchain as it is a chain of blocks. The block chain serves to confirm transactions to the rest of the network as having taken place.

The miner implemented supports two interfaces:

1. Start miner using **Blockchain.Miner.start_mining()**
2. Stop miner using **Blockchain.Miner.stop_mining()**

For keeping the mining operation fast, a difficulty of 4 is selected.
Once the miner is started, it starts the validation of blocks based on the difficulty. It adds transactions in the chain as described above.

The implementation also supports querying the chain state. Chain state can be queried before a transaction or mining and after mining using **Blockchain.Chain.get_state()**

After giving some amount of time to the mining operation, we can see that the chain size gets increased due to the additions of new transactions and blocks

Highlights of Implementation

Apart from standard features, the project also supports additional features and measures to maintain the integrity of chain.

1. Transactions
    a. Transaction verification[**Additional**]: Transaction verification measures like previous block verification, double spending and unauthorized spending are in place.
2. Mining
    a. Difficulty: Difficulty can be changed dynamically.
    b. Proof of Work: “Hashcash” algorithm is implemented.
3. Signatures[**Additional**] (ECDSA to sign transactions): Wallet supports Elliptic Curve Digital Signature Algorithm or ECDSA to ensure that funds can only be spent by their rightful owners.
4. Incentive[**Additional**]: If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction.
5. Common Standard Blockchain Protocol
    a. Hashes
    b. Merkle Trees[**Additional**]: Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree
    c. Addresses
6. Security

    All the below features are taken care as a part of wallet and transaction.
    a. Unauthorized spending
    b. Double spending
    c. Payment verification

Main Functions and their descriptions:

File: miner_server.ex

FunctionName	Description
check_transactions()	This function is used to validate all transactions in the transactions list. First it checks if the digital signature of the transaction is matching with the sender. Secondly, it verifies if the sender has enough balance in his wallet and if the amount is a valid number.
validate_amount_post_transactions()	This function is used to update the wallets of the sender and the receiver post the validity of a transaction.
mine_block()	This function is used to perform the core functionality of mining a new block. First it compiles the list of unprocessed transactions, computes the root of merkle tree which stores the hashes of all the transactions. Gets the most recent block the current chain and starts mining. Once the mining is done, it hashes the new block, updates the root of the merkle tree and then calls for a consensus to add this new block to the chain. Once that's done it adds the new block to the chain

File:wallet.ex

FunctionName	Description
send_coins	This function takes all the required parameters that constitute a transaction and creates a transaction, checks for validity, signs the transaction if valid and adds it to the transaction pool
check_amount()	This function is used to update the wallets of the entire chain in its current state. Takes the list of transactions and updates the wallets according to the transactions.

File:miner_server_impl.ex

FunctionName	Description
mine()	This function performs the same functionality as the mine() function before but also does the additional functionality of calling itself to keep the program running until all transactions in the pool are completely processed.
consensus()	This function implements the consensus algorithm defined by the blockchain protocol
insert_block()	This function is used to add a newly mined block into the chain.

File: merkle_tree.ex

FunctionName	Description
get_merkle_root()	This function returns the root of the merkle tree for a given set of transactions.
get_merkle_tree()	This function returns the merkle tree for a given set of transactions.
group_transactions()	This function recursively combines transactions in pairs
to_map()	This function maintains a map of the grouped transactions and their respective hashes.
hash_keys()	Computes the hash of the given input transaction pair
calculate_map()	This function is used to recursively calculate and reduce the map of the

	merkle tree.

A lot of files are names with tag _server to accommodate the server APIs of that particular genserver. They mainly contain getters,setters etc.

Test Cases:

(Please Note: To check the outputs of every important test case, we have put IO statements in them. But we have commented them out because having all IO statements printing at the same time will be quite impossible to comprehend. So we request that, to view the output of any test case, just navigate to that particular test case file and uncomment all the IO statements and run the program again using the commands mix compile followed by mix test.)

1. Mining a block and adding it to the chain:
   Here a new list of transactions is created and a miner is asked to process the list of those transactions. The transactions are all valid and are processed. The miner is also awarded an incentive for successfully mining and this reward is also stored in the blockchain.

```
.....List of unprocessed transactions
%Blockchain.Core.Transaction{
  amount: 5,
  destination_account: "04D598B4A5D8924BA1A1358A0C421F15F328B73640854E4AFA607549C1933CBBDBBC7AF7EF88D4B0F4B50F64390AC38CC83C364CD6C99DD84EEDFB5DBCAC6
F4AE6",
  digital_signature: "304402204F4644DB43FD0C6E7C5290F7119E287C220C34ACA50E25EECC7568375C8F6105022041A041664C0B5E50AAC9DA18D7996B18EA5ECA19A6750EAC83A
6E9745374469B",
  id: "26F22788D97B7C5C",
  origin_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A33925A5EA5
"
}
Beginning Mining
```

```
    transaction_list: [
      %Blockchain.Core.Transaction{
        amount: 5,
        destination_account: "04D598B4A5D8924BA1A1358A0C421F15F328B73640854E4AFA607549C1933CBBDBBC7AF7EF88D4B0F4B50F64390AC38CC83C364CD6C99DD84EEDFB5
DBCAC6F4AE6",
        digital_signature: "304402204F4644DB43FD0C6E7C5290F7119E287C220C34ACA50E25EECC7568375C8F6105022041A041664C0B5E50AAC9DA18D7996B18EA5ECA19A6750
EAC83A6E97453744698",
        id: "26F22788D97B7C5C",
        origin_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A3392
5A5EA5"
      },
      %Blockchain.Core.Transaction{
        amount: 25,
        destination_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25
A33925A5EA5",
        digital_signature: "",
        id: "7FDF7B268D541E72",
        origin_account: ""
      }
    ]
  }
]
Last block contains the unprocessed transaction along with the reward for mining assigned to the block that mined it.
.....

Finished in 6.3 seconds
```

2. Mining a block and the block is not added to the chain :

Here a list of transactions are created and are processed by a miner. But the block is not added to the chain as the hashes of the transactions are not validated or reached consensus.

```
....List of unprocessed transactions
%Blockchain.Core.Transaction{
  amount: 125,
  destination_account: "04D598B4A5D8924BA1A1358A0C421F15F328B73640854E4AFA607549C1933CBBDBBC7AF7EF88D4B0F4B50F64390AC38CC83C364CD6C99DD84EEDFB5DBCAC6
F4AE6",
  digital_signature: "304402202412CDE5222C52B2FE2FD4CB4261C494E93CF9BC52AC9010CE10ECEEDF98E68302206A164F3720447EE0D577F926CDDE1654471943FA198551EBB77
F9D7C0FD030ED",
  id: "47DAB222FE2AC545",
  origin_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A33925A5EA5
"
}
Beginning Mining
```

```
    ]
  },
  %Blockchain.Core.Block{
    header: %Blockchain.Core.Head{
      chain_state_merkle: "E05CED501AB7C1BC74547F73509A9A6B70C67CB40144E9729062225E2E9F4C07",
      diff_target: 4,
      nonce: 13653,
      prev_block: "9B871512327C09CE91DD649B3F96A63B7408EF267C8CC5710114E629730CB61F",
      transactions_merkle: "A7616B7F01D827012FD1211238243712DEFC5CBC34408B5C0FB861CC3591F6A7"
    },
    transaction_list: [
      %Blockchain.Core.Transaction{
        amount: 25,
        destination_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25
A33925A5EA5",
        digital_signature: "",
        id: "1757A212DBB8F7D6",
        origin_account: ""
      }
    ]
  }
]
Transaction not added as it was not validated
......

Finished in 2.7 seconds
```

3. Transaction with valid digital signature:

Here a transaction is mined and it is validated to be true as it contains the correct digital signature. It is processed and added to the chain. To show that it is added, the original transaction is filtered out from the list of transactions in the chain to show successful addition

```
Transaction with correct private key, valid transaction
Transaction Details
Miner Public Key
"04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A33925A5EA5"
Public Key of transaction
"04D598B4A5D8924BA1A1358A0C421F15F328B73640854E4AFA607549C1933CBBDBBC7AF7EF88D4B0F4B50F64390AC38CC83C364CD6C99DD84EEDFB5DBCAC6F4AE6"
Amount
15
Miner Private Key
"09572E1A9FB5DECE6DA88DE0EE009849"
Beginning mining.
```

```
    }
  ],
  [
    %Blockchain.Core.Transaction{
      amount: 15,
      destination_account: "04D598B4A5D8924BA1A1358A0C421F15F328B73640854E4AFA607549C1933CBBDBBC7AF7EF88D4B0F4B50F64390AC38CC83C364CD6C99DD84EEDFB5DB
CAC6F4AE6",
      digital_signature: "3045022100EC1F2F2CFA03B03F2C2D87019C89A62CF30B48261262BC15623CA816BE7F29B50220728EA5545D1563ADDFB0286CBC4BE37F3490FBFA1DAC8
54BBC11F6102051CCB0",
      id: "59E3D95E4083A313",
      origin_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A33925A
5EA5"
    },
    %Blockchain.Core.Transaction{
      amount: 25,
      destination_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A3
3925A5EA5",
      digital_signature: "",
      id: "1B4E235199ADD92A",
      origin_account: ""
    }
  ]
]
Valid transaction filtered from the list of processed transactions.
["59E3D95E4083A313"]
.........

Finished in 3.9 seconds
```

4.Transaction with invalid digital signature:

Here a faulty transaction is simulated with an invalid digital signature is processed. It does not pass the validation and hence is not added to the chain. To show that the this, the original transaction is filtered out from the list of transactions in the chain and the result turns out to be empty.

```
..Transaction not valid as the digital signatures do not match
Transaction Details
Miner Public Key
"04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A33925A5EA5"
Public Key of transaction
"04D598B4A5D8924BA1A1358A0C421F15F328B73640854E4AFA607549C1933CBBDBBC7AF7EF88D4B0F4B50F64390AC38CC83C364CD6C99DD84EEDFB5DBCAC6F4AE6"
Amount
15
Miner Private Key
"F98B04BC42B472ACC18829FAA979FBA3"
Beginning Mining
```

```
    },
    %Blockchain.Core.Transaction{
      amount: 25,
      destination_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A3
3925A5EA5",
      digital_signature: "",
      id: "878713C83BE63FB2",
      origin_account: ""
    }
  ],
  [
    %Blockchain.Core.Transaction{
      amount: 25,
      destination_account: "04E891784A12EB25E93F0F2A9D74C8AB4A9AD3C5FBF3C84C269EFBFDF78B870DD9DCDCB73EC68D6A17762E3AE778C826E5E90A5A409C3C56C6AAA25A3
3925A5EA5",
      digital_signature: "",
      id: "CDCAA8A79BFEDDA0",
      origin_account: ""
    }
  ]
]
Transaction list filtered for our particular transaction turns out empty as it was not processed
[]
........

Finished in 2.1 seconds
```

5.Sending amount greater than the current balance:

Here an amount greater than the current balance of a miner is tried to process. A miner with initial amount of 25 tries to send to a receiver 20 coins two times. Only the 1st transaction is processed and the balances are reflected. The 2nd transaction is not even processed.

```
........Sending an amount greater than present in the wallet. For eg. Send 20 first then Send 20 again, with an initial balance of only 25
Initial balance of sender:
25
Initial balance of receiver:
25
Transaction 1 of sender sending receiver 20
Transaction 2 of sender sending receiver 20 again
Beginning Mining
Final balance of sender:
5
Final balance of receiver:
45
Second transaction was not processed/valid as the sender had insufficient balance
..

Finished in 5.5 seconds
```

6.Sending an invalid amount of coins:

Here we consider invalidity to be an amount in with decimal points(Can be modified to accommodate any case of invalidity). When a transaction with a faulty amount is initiated, it does not go through.

```
.........Sending an invalid number of coins,for eg: 19.28
Initial balance of sender:
25
Initial balance of receiver:
25
Beginning Mining
Final balance of sender:
25
Final balance of receiver:
25
.

Finished in 2.5 seconds
10 tests, 0 failures
```