ITU COMPUTER ENGINEERING

2020 FALL COMPUTER PROJECT 1

PROJECT 3 REPORT

28.12.2020

Sercan AYDIN **150170707**

Index

1) INTRODUCTION

In this project, it is asked to create a cryptocurrency, a wallet and build blockchain model. Public keys, private keys are stored by wallet. In this block chain system, transactions are stored in a secure way. The signatures of the transaction made checked by the system and then added to the blocks after their accuracy is confirmed. The system checks whether the sender has sufficient balance. And also, different transaction in the system is checked.

2) TECHNOLOGY THAT I'VE USED

In this week, I decided to implement a web based application. I choosed to use Python programming language and Flask framework for this application. Because, Flask framework is user- and developer-friendly, has tons of useful libraries, a solid community and a great toolset. Python enables me clean and rapid development environment using Flask framework. Also, this framework is easy to learn and it's flexible by nature.

PYCRYPTO 2.6.1

3) TECHNICAL DEEP DIVE ON BLOCKCHAIN

The goal of this section is to go deeper into the technical building blocks that power the blockchain.

3.1) Public Key Cryptography

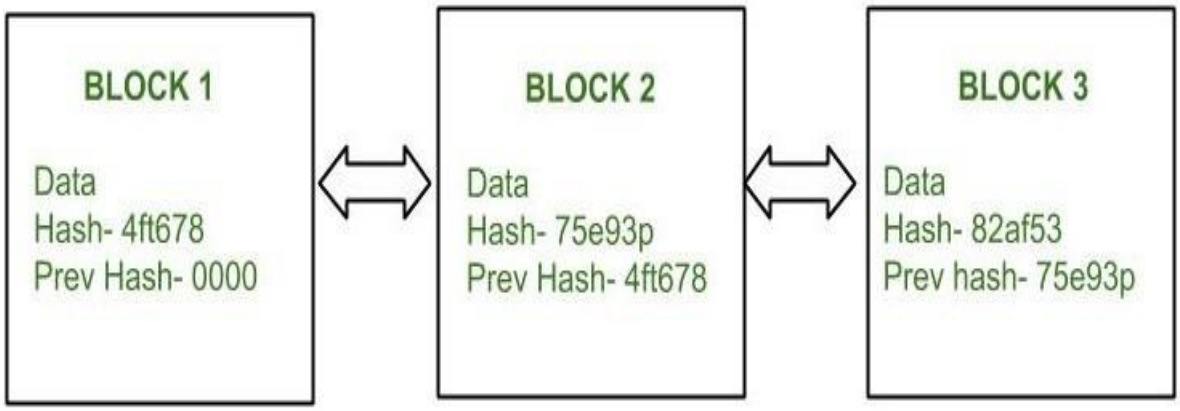
Public-key cryptography is any cryptographic system that uses pairs of keys: public keys which may be disseminated widely, and private keys which are known only to the owner. This accomplishes two functions: authentication, where the public key verifies a holder of the paired private key sent the message, and encryption, where only the paired private key holder can decrypt the message encrypted with the public key [1]. In order to send or receive BTCs, a user starts by generating a wallet which contains a pair of private and public keys. For example, If Sercan wants to send Ali some BTCs, he creates a transaction in which he enters both his and Ali's public keys, and the amount of BTCs he wants to send. He then sign the transaction using his private key. A computer on the blockchain uses Sercan's public key to verify that the transaction is authentic and adds the transaction to a block that will be later added to the blockchain.

3.2) Hashing Functions and Mining

All Bitcoin transactions are grouped in files called blocks. Bitcoin adds a new block of transactions every 10 minutes. Once a new block is added to the blockchain, it becomes immutable and can't be deleted or modified. A special group of participants in the network called miners (computers connected to the blockchain) are responsible for creating new blocks of transactions. A miner has to authenticate each transaction using the sender's public key, confirm that the sender has enough balance for the requested transaction, and add the transaction to the block. Miners are completely free to choose which transactions to include in the blocks, therefore the senders need to include a transaction fee to incentivise the miners to add their transactions to the blocks. For a block to be accepted by the blockchain, it needs to be "mined". To mine a block, miners need to find an extremely rare solution to a cryptographic puzzle. If a mined block is accepted by the blockchain, the miner receive a reward in bitcoins which is an additional incentive to transaction fees. The mining process is also referred to as Proof of Work (POW), and it's the main mechanism that enables the blockchain to be trustless and more secure.

3.3) From Blocks to Blockchain

Transactions are grouped in blocks and blocks are appended to the blockchain. In order to create a chain of blocks, each new block uses the previous block's hash as part of its data. To create a new block, a miner selects a set of transactions, adds the previous block's hash and mines the block in a similar fashion described above. Any changes to the data in any block will affect all the hash values of the blocks that come after it and they will become invalid. This give the blockchain its immutability characteristic.



3.4) Adding Blocks to the Blockchain

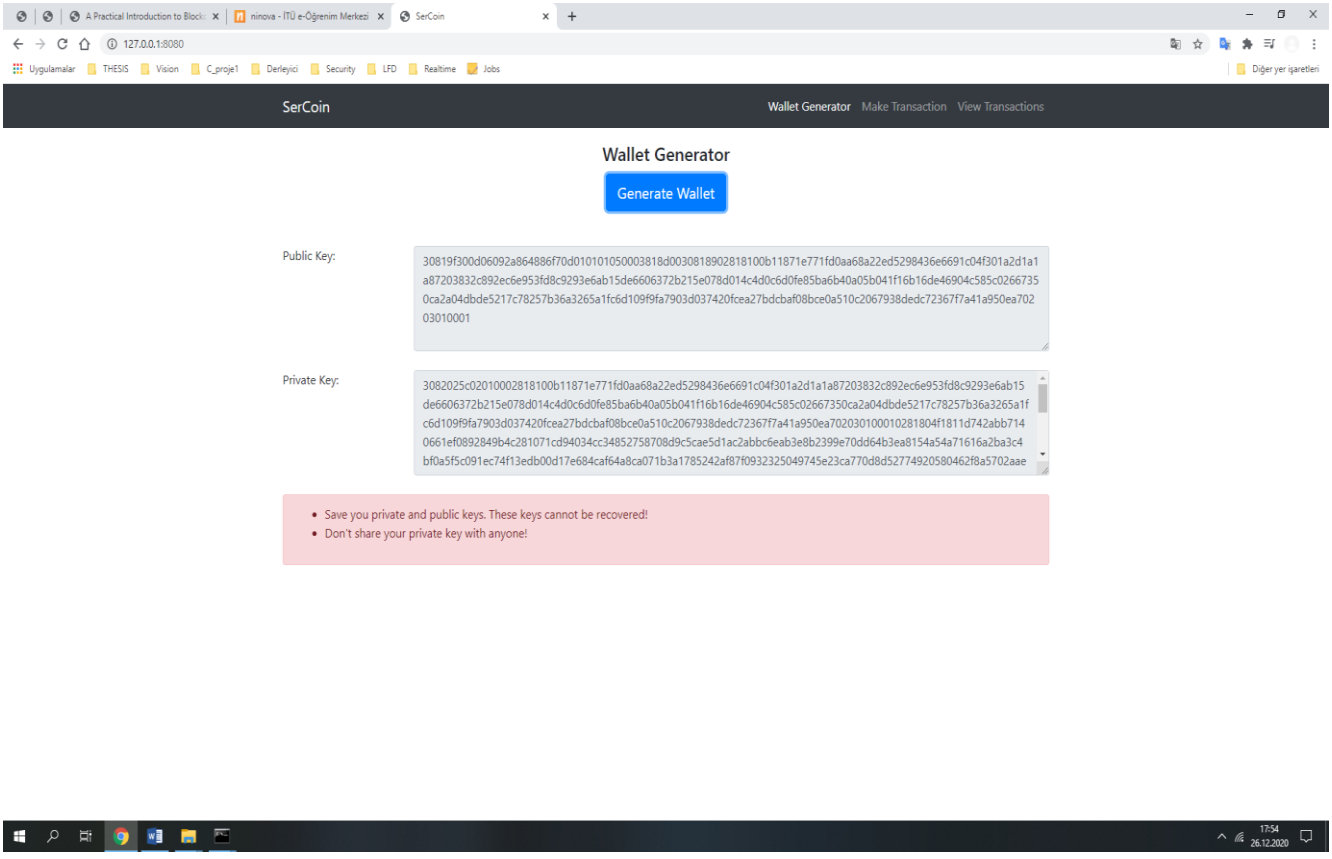
All the miners in the Bitcoin network compete with each other to find a valid block that will be added to the blockchain and get the reward from the network. Finding a nonce that validated a block is rare, but because of the number of miners, the probability of a miner in the network validating a block is extremely high. The first miner to submit a valid block gets his block added to the blockchain and receives the reward in bitcoins.

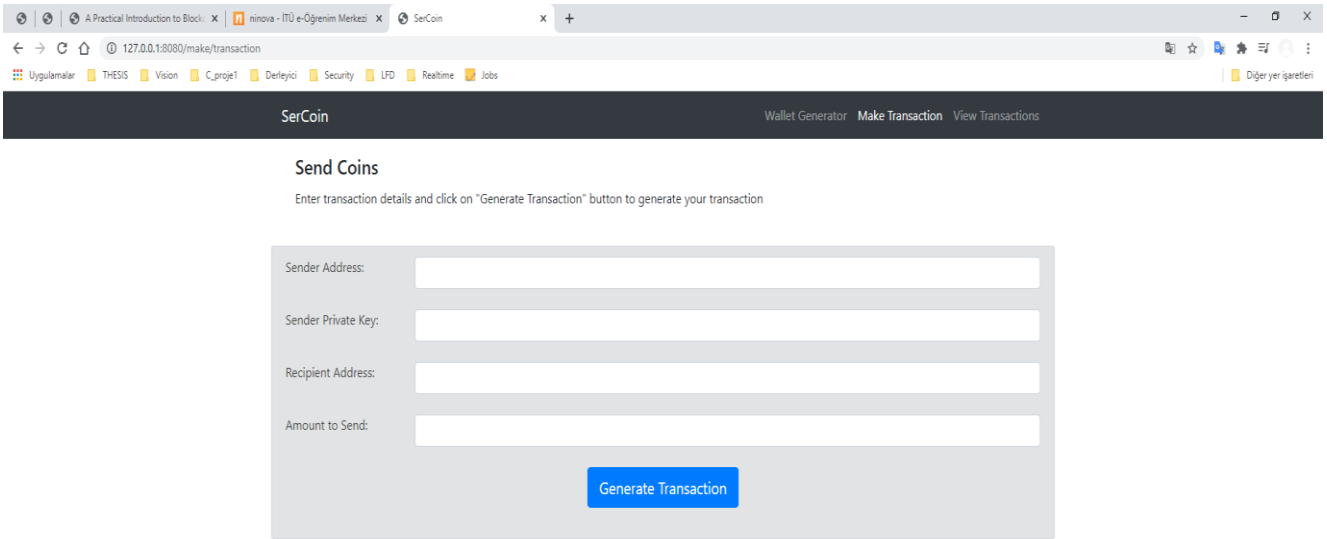
4) IMPLEMENTATION

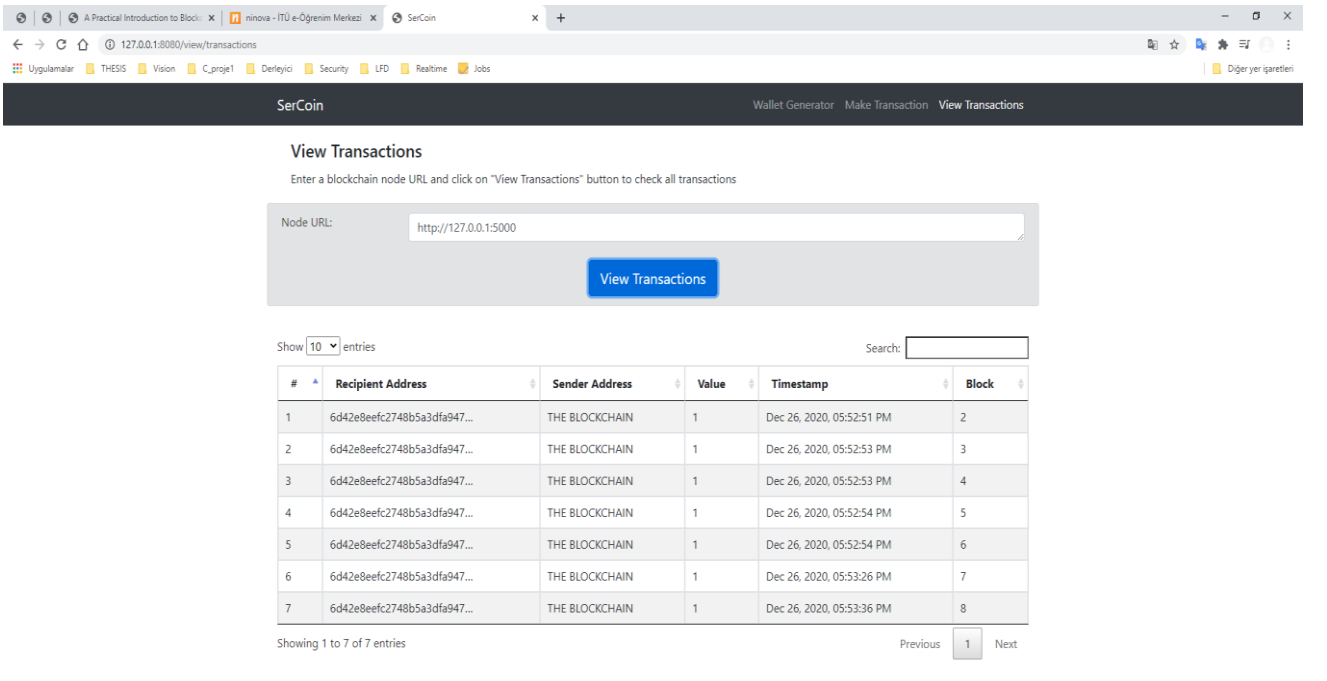
I implemented a basic blockchain and a blockchain client using Python. Our blockchain consists of features like that Possibility of adding multiple nodes to the blockchain, proof of work, transactions with RSA encryption, Public/Private keys generation. Wallet Generator button provide generate wallets (Public/Private keys pair) using RSA encryption algorithm. Make Transaction button provide generate transactions and send them to a blockchain node. View Transasctions button provide view the transactions that are on the blockchain. I defined a python class that we name Transaction that has 4 attributes sender_address, sender_privatekey, receipent_adress, value. These are the 4 pieces of information that a sender needs to create a transaction.The to_dict method returns the transaction information in a Python dictionary format (without the sender's private key). The signtransaction() method takes the transaction information (without the sender's private key) and signs it using the sender's private key.

```python
class Transaction:

    def __init__(self, sender_address, sender_private_key, recipient_address, value):
        self.sender_address = sender_address
        self.sender_private_key = sender_private_key
        self.recipient_address = recipient_address
        self.value = value

    def __getattr__(self, attr):
        return self.data[attr]

    def to_dict(self):
        return OrderedDict({'sender_address': self.sender_address,
                            'recipient_address': self.recipient_address,
                            'value': self.value})

    def sign_transaction(self):
        """
        Sign transaction with private key
        """
        private_key = RSA.importKey(binascii.unhexlify(self.sender_private_key))
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')
```

Blockchain client:




SerCoin
Wallet Generator
Make Transaction
View Transactions
Send Coins
Enter transaction details and click on "Generate Transaction" button to generate your transaction
Sender Address:
Sender Private Key:
Recipient Address:
Amount to Send:
Generate Transaction


SerCoin
Wallet Generator
Make Transaction
View Transactions
View Transactions
Enter a blockchain node URL and click on "View Transactions" button to check all transactions
Node URL:
http://127.0.0.1:5000
View Transactions
Show 10 entries
Search:
Recipient Address Sender Address Value Timestamp Block
1 6d42e8eefc2748b5a3dfa947... THE BLOCKCHAIN 1 Dec 26, 2020, 05:52:51 PM 2
2 6d42e8eefc2748b5a3dfa947... THE BLOCKCHAIN 1 Dec 26, 2020, 05:52:53 PM 3
3 6d42e8eefc2748b5a3dfa947... THE BLOCKCHAIN 1 Dec 26, 2020, 05:52:53 PM 4
4 6d42e8eefc2748b5a3dfa947... THE BLOCKCHAIN 1 Dec 26, 2020, 05:52:54 PM 5
5 6d42e8eefc2748b5a3dfa947... THE BLOCKCHAIN 1 Dec 26, 2020, 05:52:54 PM 6
6 6d42e8eefc2748b5a3dfa947... THE BLOCKCHAIN 1 Dec 26, 2020, 05:53:26 PM 7
7 6d42e8eefc2748b5a3dfa947... THE BLOCKCHAIN 1 Dec 26, 2020, 05:53:36 PM 8
Showing 1 to 7 of 7 entries
Previous
1
Next

Blockchain:

The dashboard has 2 tabs in the navigation bar:

- Mine: For viewing transactions and blockchain data, and for mining new blocks of transactions.
- Configure: For configuring connections between the different blockchain nodes.

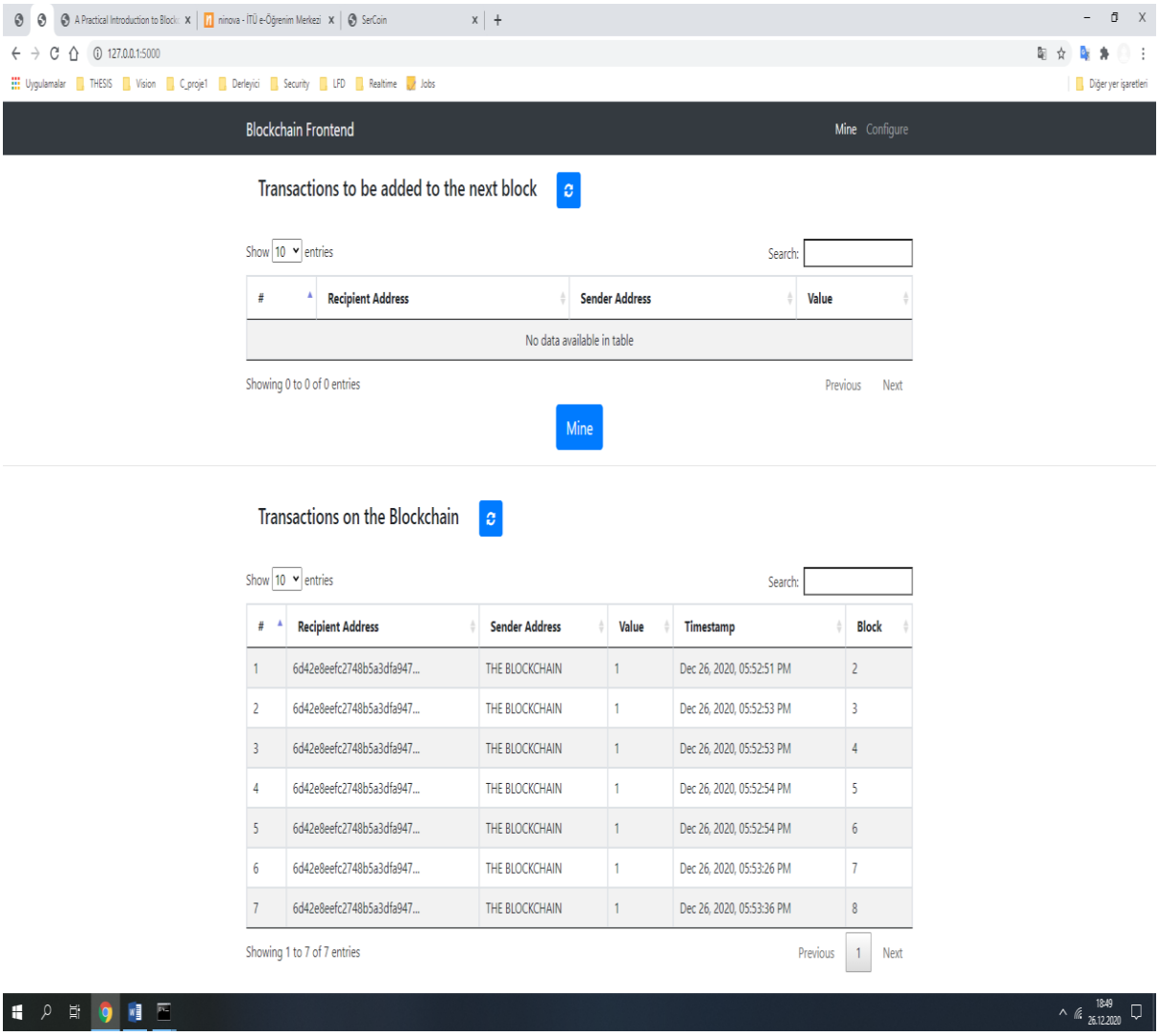
I started by defining a Blockchain class:

- transactions: List of transactions that will be added to the next block.
- chain: The actual blockchain which is an array of blocks.
- nodes: A set containing node urls. The blockchain uses these nodes to retrieve blockchain data from other nodes and updates its blockchain if they're not in sync.
- node_id: A random string to identify the blockchain node.

Some methods that I have implemented:

- register_node(node_url): Adds a new blockchain node to the list of nodes.
- verify_transaction_signature(sender_address, signature, transaction): Checks that the provided signature corresponds to transaction signed by the public key (sender_address).
- submit_transaction(sender_address, recipient_address, value, signature): Adds a transaction to list of transactions if the signature verified.
- create_block(nonce, previous_hash): Adds a block of transactions to the blockchain.
- hash(block): Create a SHA-256 hash of a block.
- proof_of_work(): Proof of work algorithm. Looks for a nonce that satisfies the mining condition.
- valid_proof(transactions, last_hash, nonce, difficulty=MINING_DIFFICULTY): Checks if a hash value satisfies the mining conditions. This function is used within the proof_of_work function.
- valid_chain(chain): checks if a bockchain is valid.
- resolve_conflicts(): Resolves conflicts between blockchain's nodes by replacing a chain with the longest one in the network.

```python
class Blockchain:

    def __init__(self): ...

    def register_node(self, node_url): ...

    def verify_transaction_signature(self, sender_address, signature, transaction): ...

    def submit_transaction(self, sender_address, recipient_address, value, signature): ...

    def create_block(self, nonce, previous_hash): ...

    def hash(self, block): ...

    def proof_of_work(self): ...

    def valid_proof(self, transactions, last_hash, nonce, difficulty=MINING_DIFFICULTY): ...

    def valid_chain(self, chain): ...

    def resolve_conflicts(self): ...
```



5) References

- 1 - Wikipedia - Public-key cryptography