2CSDE93 - Blockchain Technology

Practical 2

Aim: To create a blockchain and implement replay attacks on blockchain.

Author: Darshil Maru 20BCE514

Date: September 7, 2022

Explanation:

I have defined three functions, namely create_block, validate_blockchain, and show_blockchain. The **create_block** function will be used to create a new block and append it to the block's property in the blockchain.

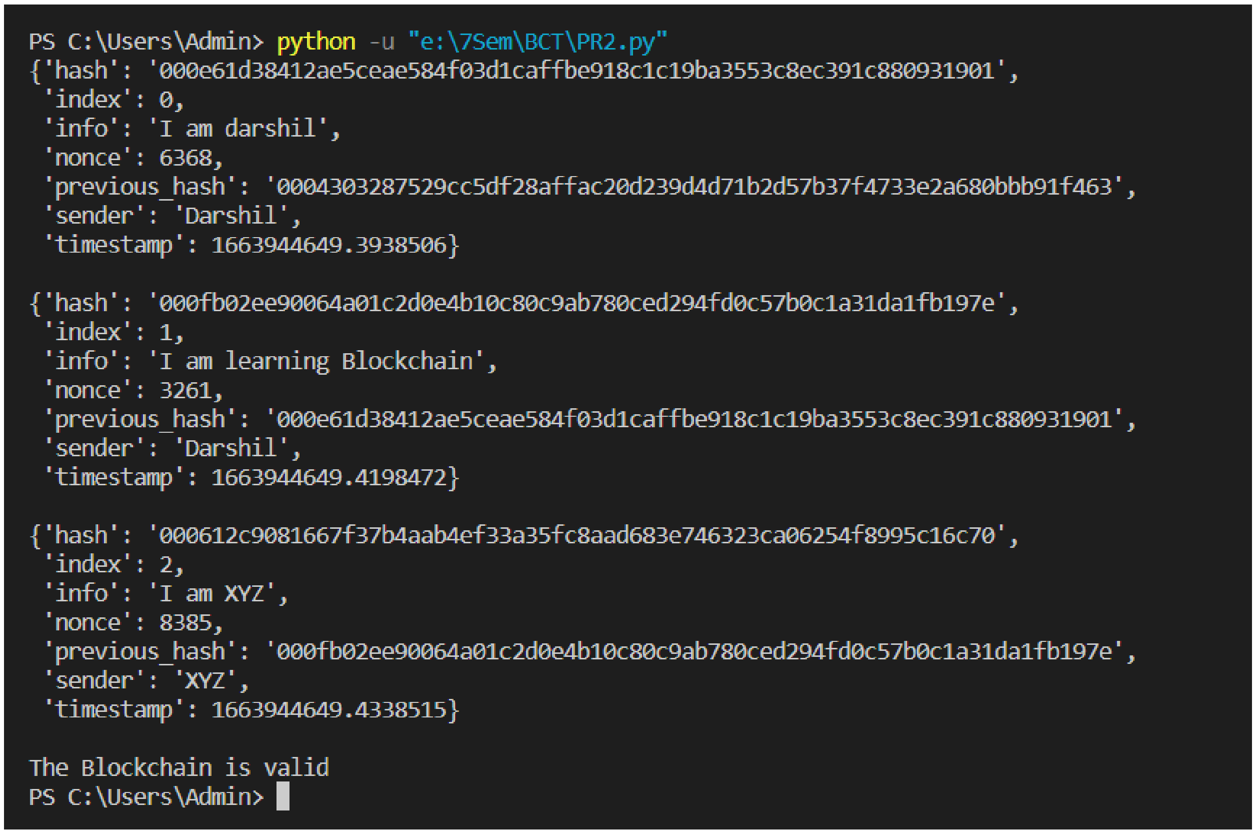
The **validate_blockchain** function will be used to validate the integrity of the blockchain.

The **show_blockchain** function will be used to display all the blocks on the blockchain.

Code:

```
import hashlib
from re import I
from time import time
from pprint import pprint


class blockchain():
    def __init__(self):
        self.blocks = []
        self.__secret = ''
        self.__difficulty = 3

        i = 0
        secret_string = '/*SECRET*/'

        while True:
            _hash = hashlib.sha256(
                str(secret_string+str(i)).encode('utf-8')).hexdigest()
            if(_hash[:self.__difficulty] == '0'*self.__difficulty):
                self.__secret = _hash
                break
            i += 1
    def create_block(self, sender: str, information: str):
        block = {
            'index': len(self.blocks),
            'sender': sender,
            'timestamp': time(),
            'info': information
        }

        if(block['index'] == 0):
            block['previous_hash'] = self.__secret
        else:
            block['previous_hash'] = self.blocks[-1]['hash']

        i = 0
        while True:
            block['nonce'] = i
            _hash = hashlib.sha256(str(block).encode('utf-8')).hexdigest()
            if(_hash[:self.__difficulty] == '0'*self.__difficulty):
                block['hash'] = _hash
                break
            i += 1
        self.blocks.append(block)

    def show_blockchain(self):
        for block in self.blocks:
            pprint(block)
            print()

    def validate_blockchain(self):
        valid = True
        n = len(self.blocks)-1
        i = 0
        while(i < n):
            if(self.blocks[i]['hash'] !=
self.blocks[i+1]['previous_hash']):
                valid = False
                break
            i += 1
        if valid:
            print('The Blockchain is valid')
        else:
            print('The Blockchain is invalid')


b = blockchain()
b.create_block('Darshil', information='I am darshil')
b.create_block('Darshil', information='I am learning Blockchain')
b.create_block('XYZ', information='I am XYZ')
b.show_blockchain()
b.validate_blockchain()
```

Output:



```
PS C:\Users\Admin> python -u "e:\7Sem\BCT\PR2.py"
{'hash': '000e61d38412ae5ceae584f03d1caffbe918c1c19ba3553c8ec391c880931901',
 'index': 0,
 'info': 'I am darshil',
 'nonce': 6368,
 'previous_hash': '0004303287529cc5df28affac20d239d4d71b2d57b37f4733e2a680bbb91f463',
 'sender': 'Darshil',
 'timestamp': 1663944649.3938506}

{'hash': '000fb02ee90064a01c2d0e4b10c80c9ab780ced294fd0c57b0c1a31da1fb197e',
 'index': 1,
 'info': 'I am learning Blockchain',
 'nonce': 3261,
 'previous_hash': '000e61d38412ae5ceae584f03d1caffbe918c1c19ba3553c8ec391c880931901',
 'sender': 'Darshil',
 'timestamp': 1663944649.4198472}

{'hash': '000612c9081667f37b4aab4ef33a35fc8aad683e746323ca06254f8995c16c70',
 'index': 2,
 'info': 'I am XYZ',
 'nonce': 8385,
 'previous_hash': '000fb02ee90064a01c2d0e4b10c80c9ab780ced294fd0c57b0c1a31da1fb197e',
 'sender': 'XYZ',
 'timestamp': 1663944649.4338515}

The Blockchain is valid
PS C:\Users\Admin>
```