Blockchain Technology

19BCE248

Practical 3

AIM: To perform thorough study and installation of Anaconda 5.0.1 and Python 3.6 and perform proof of work (POW) consensus mechanism. Also, notice the changes in mining rewards and nonce requirement.

Code:

```python
import hashlib
from select import select
import time
import json
import threading
import random

class Block:
    def __init__(self):
        self.chain=[]
        self.difficulty=4

    def addNewBlock(self,msg):
        data={}
        data["time"]=time.time()
        data["id"]=len(self.chain)
        data["message"]=msg
        if len(self.chain)==0:
            data["prevHash"]=0
        else:
            data["prevHash"]=self.chain[-1]["currHash"]
        str=json.dumps(data).encode()
        data["currHash"]=hashlib.sha256(str).hexdigest()
        data["nonce"]=0
        self.chain.append(data)
        self.getSHA()

    def validate_blocks(self):
        while True:
            ind=-1
            no_blocks=len(self.chain)
            for i in range(1,no_blocks):
                if(self.chain[i]["prevHash"]!=self.chain[i-1]["currHash"]):
                    print("Tampering Found!!")
                    ind=i
                    break
            if ind !=-1:
                for i in range(ind,no_blocks):
                    self.chain[i]["prevHash"]=self.chain[i-1]["currHash"]
    def getSHA(self):
        data=self.chain[-1]
        str=json.dumps(data).encode()
        hash=hashlib.sha256(str).hexdigest()
        while not self.isAchieved(hash):
            data["nonce"]=data["nonce"]+1
            str=json.dumps(data).encode()
            hash=hashlib.sha256(str).hexdigest()
        self.chain[-1]["currHash"]=hash


    def isAchieved(self,hash):
        str=hash[0:self.difficulty]
        count=str.count('0')
        return count==self.difficulty
```

```python
def validate():
    validate_thread = threading.Thread(target=block.validate_blocks,
name="Validate Blocks")
    validate_thread.start()


if __name__=="__main__":
    block=Block()
    for i in range(5):
        block.addNewBlock("My current count is"+str(i))
    # block.chain[2]["currHash"]='1'
    validate()


print(block.chain[2])
```

Explanation of Implementation:

- The basic implementation structure remains same as practical 2.
- The only change is addition of nonce in each block.
- It basically identifies the functionality of miners who competes during the addition of any transaction block in existing blockchain.
- Here it will continuously check whether first n bits are being 0 or not where n is given by user which represents the difficulty level.
- Once the goal is achieved it will just store the value.

Learning:

From these practical we learned about how we can implement the functionality of miners from scratch. Also we can understand how difficult it is for miners to achieve goal state.