Blockchain Technology

Assignment – 06

Aim: Program to generate chain of blocks.

Description:

1. Hashlib module:

This module implements a common interface to many different secure hash and message digest algorithms. Included are the FIPS secure hash algorithms SHA1, SHA224, SHA256, SHA384, and SHA512 (defined in FIPS 180-2) as well as RSA’s MD5 algorithm (defined in internet RFC 1321). The terms “secure hash” and “message digest” are interchangeable. Older algorithms were called message digests. The modern term is secure hash.

1. Datetime module:

Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

1. Proof of Work (PoW) algorithm:

Proof of work is a form of cryptographic proof in which one party proves to others that a certain amount of a specific computational effort has been expended.

Code:

import hashlib

from datetime import datetime

*#defining global variables*

DIFFICULTY\_LEVEL = 0

BLOCKCHAIN = []

class Block:

    def \_\_init\_\_(self, data, prev\_hash=None):

        self.blockId = len(BLOCKCHAIN)

        self.timestamp = datetime.now().strftime("%d-%h-%y  %H:%M:%S.%f")

        self.data = data

        self.prev\_hash = prev\_hash

        self.nonce = 0

        self.hash = self.hashBlock()

    def hashBlock(self):

        data = (str(self.blockId) + str(self.data) + str(self.prev\_hash) + str(self.nonce)).encode()

        return hashlib.sha256(data).hexdigest()

class Chain:

    def \_\_init\_\_(self):

        self.addBlock("Genesis Block")

    def addBlock(self, data):

        if len(BLOCKCHAIN) == 0:

            prev\_hash = "0"\*64

            newBlock = Block(data, prev\_hash)

            newBlock.hash = self.proofOfWork(newBlock)

            BLOCKCHAIN.append(newBlock)

        else:

            prev\_hash = BLOCKCHAIN[-1].hash

            newBlock = Block(data, prev\_hash)

            newBlock.hash = self.proofOfWork(newBlock)

            BLOCKCHAIN.append(newBlock)

    def getChain(self):

        return BLOCKCHAIN

    def proofOfWork(self, block):

        while block.hash[:DIFFICULTY\_LEVEL] != "".zfill(DIFFICULTY\_LEVEL):

            block.nonce += 1

            block.hash = block.hashBlock()

        return block.hash

    def isValid(self):

        for i in range(1, len(BLOCKCHAIN)):

            curntBlock = BLOCKCHAIN[i]

            prevBlock = BLOCKCHAIN[i-1]

            if curntBlock.hash != curntBlock.hashBlock():

                return False

            if curntBlock.prev\_hash != prevBlock.hash:

                return False

        return True

numOfBlocks = int(input("\nHow many blocks do you want to create: "))

*#choose the difficulty leve for the Proof of Work*

DIFFICULTY\_LEVEL = int(input("\nEnter the difficulty level: "))

DATA = []

for block in range(1, numOfBlocks):

    DATA.append(input(f"\nEnter the data for the block-{block}: "))

*# blockchain is initialized and genesis block is created*

blockchain = Chain()

print(f"\n{BLOCKCHAIN[-1].\_\_dict\_\_}")

for num in range(1, numOfBlocks):

    blockchain.addBlock(DATA[num-1])

    print(f"\n\n{BLOCKCHAIN[-1].\_\_dict\_\_}")

print(f"\n\nBlockchain is valid: {blockchain.isValid()}\n")

Output:

How many blocks do you want to create: 4

Enter the difficulty level: 5

Enter the data for the block-1: this my block 1

Enter the data for the block-2: my name is vishal

Enter the data for the block-3: We can put anything in the data field as data instead of only putting strings...okay?

{'blockId': 0, 'timestamp': '20-Aug-22 19:45:29.074431', 'data': 'Genesis Block', 'prev\_hash': '0000000000000000000000000000000000000000000000000000000000000000', 'nonce': 281159, 'hash': '00000de4a2851f8108d9e23ffa6f00d8c5d1b91118bfe6e2ec7cc24bd2712a3b'}

{'blockId': 1, 'timestamp': '20-Aug-22 19:45:29.665594', 'data': 'this my block 1', 'prev\_hash': '00000de4a2851f8108d9e23ffa6f00d8c5d1b91118bfe6e2ec7cc24bd2712a3b', 'nonce': 75483, 'hash': '00000b6356d723f02b991984d6d99b3e624df59548a0852a5f56b6166b6a6a7c'}

{'blockId': 2, 'timestamp': '20-Aug-22 19:45:29.848464', 'data': 'my name is vishal', 'prev\_hash': '00000b6356d723f02b991984d6d99b3e624df59548a0852a5f56b6166b6a6a7c', 'nonce': 384999, 'hash': '00000000121fb7669fffa4238f3e1d654779a28741d5f3204585c3fceeb47ef6'}

{'blockId': 3, 'timestamp': '20-Aug-22 19:45:30.632044', 'data': 'We can put anything in the data field as data instead of only putting strings...okay?', 'prev\_hash': '00000000121fb7669fffa4238f3e1d654779a28741d5f3204585c3fceeb47ef6', 'nonce': 476692, 'hash': '000004dfc6bb107e5b8ec326c0e34a68249034705665dd6bc2dfc73618c487eb'}

Blockchain is valid: True