MSc CS - Part II
SEM III
E-Journal

Roll No.	054
Name	RAWAL PALAK RAJANIKANT
Subject	BLOCKCHAIN & CRYPTOCURRENCY

Thakur Educational Trust's (Regd.)
THAKUR COLLEGE OF SCIENCE & COMMERCE
AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI
NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)
Best College Award by University of Mumbai for the Year 2018-2019
CELEBRATING 25 YEARS OF GLORY

CERTIFICATE

This is here to certify that Mr./~~Ms~~. RAWAL PALAK RAJNIKANT , Seat Number 054 of M.Sc. Computer Science Part II, has satisfactorily completed the required number of experiments prescribed by the UNIVERSITY OF MUMBAI during the academic year 2021 – 2022.

Date: -09-2022
Place: Mumbai

Teacher In-Charge Head of Department

External Examiner

Index

Practical No.1

Aim: Implement Naive Blockchain construction

Theory:

Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. An *asset* can be tangible (a house, car, cash, land) or intangible (intellectual property, patents, copyrights, branding). Virtually anything of value can be tracked and traded on a blockchain network, reducing risk and cutting costs for all involved. Business runs on information. The faster it's received and the more accurate it is, the better. Blockchain is ideal for delivering that information because it provides immediate, shared and completely transparent information stored on an immutable ledger that can be accessed only by permissioned network members. A blockchain network can track orders, payments, accounts, production and much more. Andbecause members share a single view of the truth, you can see all details of a transaction end to end, giving you greater confidence, as well as new efficiencies and opportunities.

Distributed ledger technology

All network participants have access to the distributed ledger and its immutable record of transactions. With this shared ledger, transactions are recorded only once, eliminating the duplication of effort that's typical of traditional business networks.

Immutable records

No participant can change or tamper with a transaction after it's been recorded to the shared ledger. If a transaction record includes an error, a new transaction must be added toreverse the error, and both transactions are then visible.

Smart contracts

To speed transactions, a set of rules called a smart contract is stored on the blockchain and executed automatically. A smart contract can define conditions for corporate bond transfers, include terms for travel insurance to be paid and much more.

Code:

```
const SHA26 = require('crypto-js/sha256')class
Block{
constructor(index , timestapms , data , previousHash = ''){
this.index = index; this.timestapms =
timestapms; this.data = data;
this.previousHash = previousHash;
this.hash = this.calculateHash();
}
calculateHash(){
return SHA26(this.index + this.previousHash + this.timestapms +
JSON.stringify(this.data)).toString();
}
}
class Blockchain{
constructor(){
this.chain = [this.createGenesisisBlock()];
}
createGenesisisBlock(){
return new Block(0, "01/01/2017", "Genesis Block","0");
}
getLatestBlock(){
return this.chain[this.chain.length - 1]
}
addBlock(newBlock){
newBlock.previousHash = this.getLatestBlock().hash;
newBlock.hash = newBlock.calculateHash();
this.chain.push(newBlock);}
ischainValid(){
for(let i = 1; i < this.chain.length; i++){const
currentBlock = this.chain[i];
const previousBlock = this.chain[i-1]; if(currentBlock.hash !==
currentBlock.calculateHash()){return false;
}
if(currentBlock.previousHash !== previousBlock.hash){return
false;
}
}
return true;
}
}
let ZeeCoin = new Blockchain();
ZeeCoin.addBlock(new Block(1,"08/08/2022",{amount : 4}));
ZeeCoin.addBlock(new Block(2,"09/09/2022",{amount : 10})); console.log("is
Block chain Valid ? " + ZeeCoin.ischainValid()); console.log("Block Chain
Created :\n",JSON.stringify(ZeeCoin, null , 4));
```

```
//ZeeCoin.chain[1].data = { amount: 100}; // tampering the data to check block
chain Validity
//console.log("is Block chain Valid ? " + ZeeCoin.ischainValid());
```

Output:

```
Zeeshan's PC@MSI MINGW64 /d/SEM-3 practicals/Blockchain-practicals
$ node "d:\SEM-3 practicals\Blockchain-practicals\index.js"
is Block chain Valid ? true
Block Chain Created :
 {
    "chain": [
        {
            "index": 0,
            "timestapms": "01/01/2017",
            "data": "Genesis Block",
                "amount": 4
            },
            "previousHash": "2091e5c7e53b2da0bbd7610d1298a2d2ba4091668e8afbc1cadac49c75607ca7",
            "hash": "2351f84b9e1df1f61e779ba00e1bdeb504e2d0e1cde8ee1a12d6fe9fffe9fd2d"
        },
        {
            "index": 2,
            "timestapms": "09/09/2022",
            "data": {
                "amount": 10
            },
            "previousHash": "2351f84b9e1df1f61e779ba00e1bdeb504e2d0e1cde8ee1a12d6fe9fffe9fd2d",
            "hash": "a527690b285e07d02827107e7b0f2fe7eece1f9e050c88ee7d7e45249fddd429"
        }
    ]
}
```

After Tampering Data check validity:

```
ZeeCoin.chain[1].data = { amount: 100}; // tampering the data to check
blockchain Validity
console.log("is Block chain Valid ? " + ZeeCoin.ischainValid());
```

Output:

```
is Block chain Valid ? false
```

Conclusion:

Hence we successfully implemented Blockchain using JavaScript.

Practical No. 2

Aim: Implement Direct Acyclic Graph.

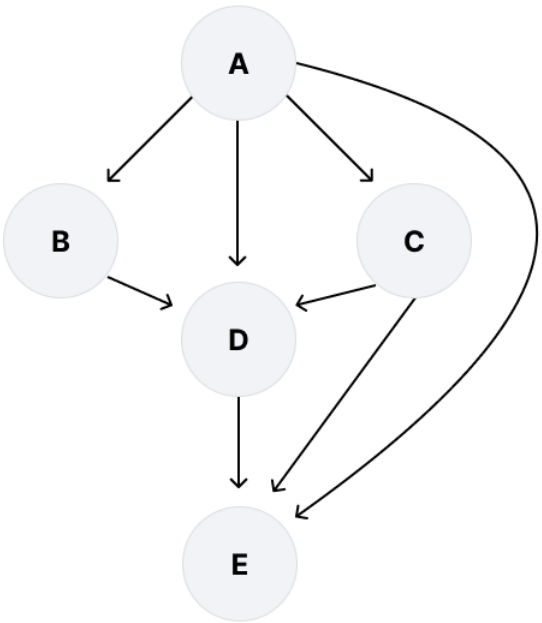
Theory:

A directed acyclic graph (DAG) is a graph that is directed and without cycles connecting the other edges. This means that it is impossible to traverse the entire graph starting at one edge. The edges of the directed graph only go oneway. The graph is a topological sorting, where each node is in a certain order.

In graph theory, a graph is a series of vertexes connected by edges. In a directed graph, the edges are connected so that each edge only goes one way.A directed acyclic graph means that the graph is not cyclic, or that it is impossible to start at one point in the graph and traverse the entire graph.
Each edge is directed from an earlier edge to a later edge. This is also known asa topological ordering of a graph.

A spreadsheet may be represented as a directed acyclic graph, with each cell avertex and an edge connected a cell when a formula references another cell.

Other applications include scheduling, circuit design and Bayesian networks.



```
class AjlistNode
{
	constructor(id)
	{
		// Set value of node key
		this.id = id;
		this.next = null;
	}
}
class Vertices
{
	constructor(data)
	{
		this.data	=	data;
		this.next	=	null;
		this.last = null;
	}
}
class Graph
{
	constructor(size)
	{
		// Set value
		this.size = size;
		this.node = Array(size).fill(null);
		this.setData();
	}
	// Set initial node value
	setData()
	{
		if (this.size <= 0)
		{
			console.log("\nEmpty Graph");
		}
		else
		{
			for (var index = 0; index < this.size; index++)
			{
				// Set initial node value
				this.node[index] = new Vertices(index);
			}
		}
	}
	connection(start, last)
	{
		// Safe connection
		var edge = new AjlistNode(last); if
		(this.node[start].next == null)
```

```
        {
            this.node[start].next = edge;
        }
        else
        {
            // Add edge at the end
            this.node[start].last.next = edge;
        }
        // Get last edge
        this.node[start].last = edge;
    }
    //   Handling the request of adding new edgeaddEdge(start,
    last)
    {
            if (start >= 0 && start < this.size &&
                last >= 0 && last < this.size)
        {
            this.connection(start, last);
        }
        else
        {
            // When invalid nodes console.log("\nHere
            Something Wrong");
        }
    }
    printGraph()
    {
        if (this.size > 0)
        {
            // Print graph ajlist Node value
            for (var index = 0; index < this.size; ++index)
            {
                process.stdout.write("\nAdjacency list of vertex " +
                                                index + " :");var
                edge = this.node[index].next; while (edge
                != null)
                {
                    // Display graph node
                    process.stdout.write("   " +
                                                this.node[edge.id].data);
                    // Visit to next edgeedge
                    = edge.next;
                }
            }
        }
    }
    // Find indegree of each nodes of a given graph
    // Find the incoming edges of each node
```

```
findIndegree(indegree)
{
	if (this.size <= 0)
	{
		return;
	}
	var edge = null;
	for (var i = 0; i < this.size; ++i)
	{
		edge = this.node[i].next;while
		(edge != null)
		{
			// Increase indegree of node
			indegree[edge.id]++;
			// Visit to next edgeedge
			= edge.next;
		}
	}
}
findSequence(indegree, visit, index, result)
{
	if (index == this.size)
	{
		// Display result
		process.stdout.write("\n");var j
		= 0;
		while (j < this.size)
		{
			process.stdout.write(" " + result[j]);j++;
		}
		return;
	}
	var edge = null;var
	i = 0;
	while (i < this.size)
	{
		if (indegree[i] == 0 && visit[i] == false)
		{
			visit[i] = true;
			result[index] = i;
			// Get node edge
			edge = this.node[i].next;
			// Reduce indegree while
			(edge != null)
			{
				indegree[edge.id]--;
				// Visit to next edge
```

```
                edge = edge.next;
            }
            this.findSequence(indegree,
                                visit, index
                                + 1,result);
            visit[i] = false;
            edge = this.node[i].next;
            // Increase indegree
            while (edge != null)
            {
                indegree[edge.id]++;
                edge = edge.next;
            }
        }
        i++;
    }
}
topologicalSort()
{
    if (this.size <= 0)
    {
        return;
    }
    // Use to track node
    var visit = Array(this.size).fill(false);
    // Store indegree of node edges
    var indegree = Array(this.size).fill(0);
    // Store result of topological sort
    var result = Array(this.size).fill(-1);

    // Find indegree of each node in graph
    this.findIndegree(indegree); process.stdout.write("\n");
    this.findSequence(indegree, visit, 0, result);
}

function main()

// 6 implies the number of nodes in graphvar g =
new Graph(6);
// Connect node with an edge
// First and second parameter indicate node index
g.addEdge(1, 0);
g.addEdge(1, 4);
g.addEdge(1, 5);
g.addEdge(2, 0);
g.addEdge(2, 4);
```

```
    g.addEdge(2, 5);
    g.addEdge(3, 4);
    g.addEdge(3, 5);
    g.addEdge(5, 4);
    // Print graph element
    g.printGraph();
    g.topologicalSort();
}
// Start program execution
main();
```

Output:

```
Zeeshan's PC@MSI MINGW64 /d/SEM-3 practicals/Blockchain-practicals
$ node "d:\SEM-3 practicals\Blockchain-practicals\index.js"

Adjacency list of vertex 0 :
Adjacency list of vertex 1 :  0  4  5
Adjacency list of vertex 2 :  0  4  5
Adjacency list of vertex 3 :  4  5
Adjacency list of vertex 4 :
Adjacency list of vertex 5 :  4

  1  2  0  3  5  4
  1  2  3  0  5  4
  1  2  3  5  0  4
  1  2  3  5  4  0
  1  3  2  0  5  4
  1  3  2  5  0  4
  1  3  2  5  4  0
  2  1  0  3  5  4
  2  1  3  0  5  4
  2  1  3  5  0  4
  2  1  3  5  4  0
  2  3  1  0  5  4
  2  3  1  5  0  4
  2  3  1  5  4  0
  3  1  2  0  5  4
  3  1  2  5  0  4
  3  1  2  5  4  0
  3  2  1  0  5  4
  3  2  1  5  0  4
  3  2  1  5  4  0
```

Conclusion:

Hence we successfully implemented DAG using JavaScript.

Practical No.3

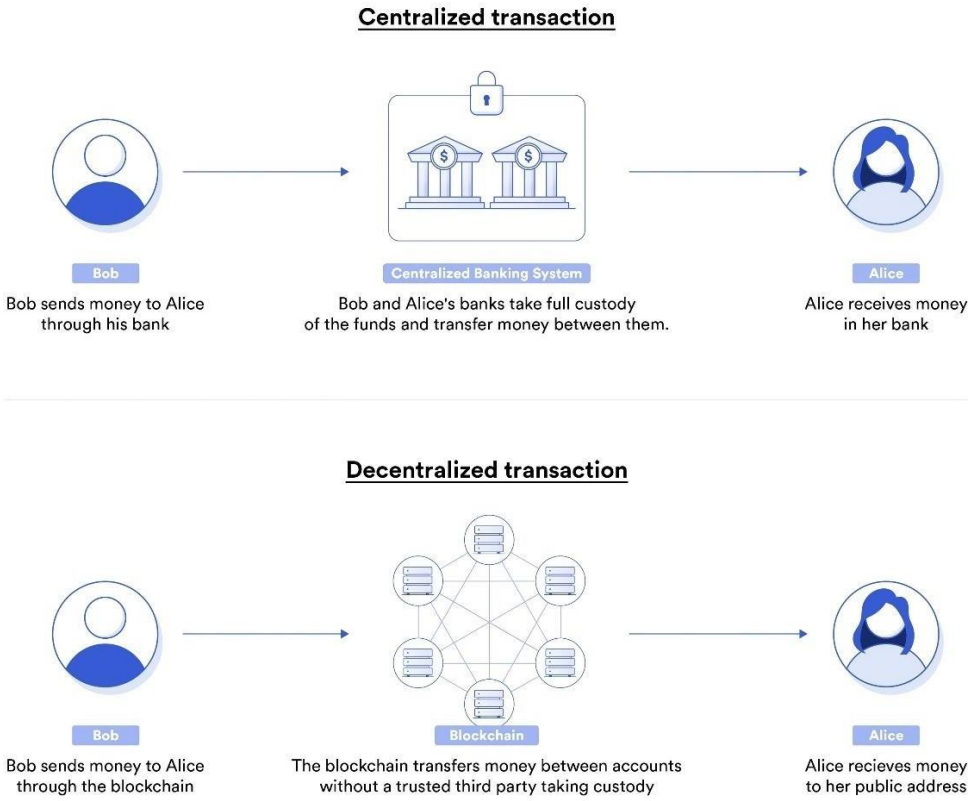
Aim: Explore - Smart Contract Construction (Case Study)

Theory:

What Is Smart Contract?

Smart contracts are computer programs or protocols for automated transactions that are stored on a blockchain and run in response to meeting certain conditions. In other words, smart contracts automate the execution of agreements so that all participants can ascertainthe outcome as soon as possible without the involvement of an intermediary or time delay.

- Smart contracts are self-executing contracts in which the contents of the buyer-seller agreement are inscribed directly into lines of code.
- According to Nick Szabo, an American computer scientist who devised a virtual currency called "Bit Gold" in 1998, Smart contracts are computerized transaction protocols that execute contract conditions.
- Using it makes the transactions traceable, transparent, and irreversible.



Benefits of Smart Contracts
Accuracy, Speed, and Efficiency

- The contract is immediately executed when a condition is met.
- Because smart contracts are digital and automated, there is no paperwork to deal with, and
- No time was spent correcting errors that can occur when filling out documentation by hand.

Trust and Transparency

- There's no need to worry about information being tampered with for personal gain because there's no third party engaged and
- Encrypted transaction logs are exchanged among participants.

Security

- Because blockchain transaction records are encrypted, they are extremely difficult to hack.
- Furthermore, because each entry on a distributed ledger is linked to the entries before and after it, hackers would have to change the entire chain to change a singlerecord.

Savings

- Smart contracts eliminate the need for intermediaries to conduct transactions, as well as the time delays and fees that come with them.

How Do Smart Contracts Work?

A smart contract is a sort of program that encodes business logic and operates on a dedicated virtual machine embedded in a blockchain or other distributed ledger.

Step 1: Business teams collaborate with developers to define their criteria for the smart contract's desired behaviour in response to certain events or circumstances.

Step 2: Conditions such as payment authorization, shipment receipt, or a utility meter reading threshold are examples of simple events.

Step 3: More complex operations, such as determining the value of a derivative financial instrument, or automatically releasing an insurance payment, might be encoded using more sophisticated logic.

Step 4: The developers then use a smart contract writing platform to create and test the logic. After the application is written, it is sent to a separate team for security testing.

Step 5: An internal expert or a company that specializes in vetting smart contract security could be used.

Step 6: The contract is then deployed on an existing blockchain or other distributed ledger infrastructure once it has been authorized.

Step 7: The smart contract is configured to listen for event updates from an "oracle," which is effectively a cryptographically secure streaming data source, once it has been deployed.

Step 8: Once it obtains the necessary combination of events from one or more oracles, the smart contract executes.

Smart Contacts and Flight Insurance

Let's consider a real-life scenario in which smart contracts are used. Rachel is at the airport, and her flight is delayed. AXA, an insurance company, provides flight delay insurance utilizing Ethereum smart contracts. This insurance compensates Rachel in such a case. How? The smart contract is linked to the database recording flight status. The smart contract is created based on terms and conditions.

The condition set for the insurance policy is a delay of two hours or more. Based on the code, the smart contract holds AXA's money until that certain condition is met. The smart contract is submitted to the nodes on EMV (a runtime compiler to execute the smart contract code) for evaluation. All the nodes on the network executing the code must come to the same result. That result is recorded on the distributed ledger. If the flight is delayed inexcess of two hours, the smart contract self-executes, and Rachel is compensated. Smart contracts are immutable; no one may alter the agreement.

Voting and Blockchain Implementation of Smart Contracts

Using Blockchain in the voting process can eliminate common problems. A centralized voting system faces difficulties when it comes to tracking votes – identity fraud, miscounts, or bias by voting officials. Using a smart contract, certain predefined terms and conditions are pre-set in the contract. No voter can vote from a digital identity other than his or her own. The counting is foolproof. Every vote is registered on a blockchain network, and the counting is tallied automatically with no interference from a third party or dependency on a manual process. Each ID is attributed to just one vote. Validation is accomplished by the users on the blockchain network itself. Thus, the voting process can be in a public blockchain, or it could be in a decentralized autonomous organization-based blockchain setup. As a result, every vote is recorded on the ledger, and the information cannot be modified. That ledger is publicly available for audit and verification.

Smart contracts allow you to create voting systems in which you can add and remove members, change voting rules, change debating periods, or alter the majority rule. For

instance, you can create a vote for a decision within a decentralized autonomous organization. Rather than a central authority making a decision, a voting mechanism within the organization can determine whether the proposal is accepted or rejected

Blockchain Implementation of a Smart Contract andCrowdfunding

Ethereum-based smart contracts may be used to create digital tokens for performing transactions. You may design and issue your own digital currency, creating a tradable computerized token. The tokens use a standard coin API. In the case of Ethereum, there are standardizations of ERC 2.0, allowing the contract to access any wallet for exchange automatically. As a result, you build a tradable token with a fixed supply. The platform becomes a central bank of sorts, issuing digital money.

Suppose you want to start a business requiring funding. But who would lend money to someone they don't know or trust? Smart contracts have a major role to play. With Ethereum, you can build a smart contract to hold a contributor's funds until a given date passes or a goal is met. Based on the result, the funds are released to the contract owners or sent back to the contributors. The centralized crowdfunding system has many issues with management systems. To combat this, a DAO (Decentralized Autonomous Organization) is utilized for crowdfunding. The terms and conditions are set in the contract, and every individual participating in crowdfunding is given a token. Every contribution is recorded on the Blockchain.

Limitation of Smart Contracts

- Because smart contracts can't send HTTP queries, they can't acquire information about "real-world" events. This is by design.
- Using external data could jeopardize consensus, which is critical for security and decentralization.

Use Cases of Smart Contracts

- The use cases for smart contracts range from simple to complex.
- They can be used for simple economic transactions, such as moving money from point A to point B, as well as for smart access management in the sharing economy.
- Smart contracts could disrupt many industries.
- Banking, insurance, energy, e-government, telecommunications, the music business, art, mobility, education, and many other industries have use cases.

Conclusion:

Hence we successfully demonstrated case study on smart contracts.

Practical No.4

Aim: Solve Mining puzzles

Theory:

Mining puzzles (Coinbase) refers to the transaction whereby miners receiveBitcoin as a reward for generating a new block through mining.
Bitcoin is only issued through mining and, when it is newly issued, it is given tothe successful miners as a reward.
The reward for mining a block began at 50 BTC. The reward is halved every 210,000 blocks, and the 6,929,999th block will be the last to reward mining.The total amount of Bitcoin to be issued is 21 million BTC -- the maximum amount of Bitcoin that can be issued.
Rewards decrease as explained above, and the speed at which it is issued isadjusted so that one block is generated every 10 minutes. It will take about132 years to mine all 6,929,999 blocks, and the last block will be mined in 2140.

1. **Individual Mining**

When mining is done by an individual, user registration as a miner is necessary. As soon as a transaction takes place, a mathematical problem is given to all the single users in the blockchain network to solve. The first one to solve it gets rewarded.

Once the solution is found, all the other miners in the blockchain network will validate the decrypted value and then add it to the blockchain. Thus, verifying the transaction.

2. **Pool Mining**

In pool mining, a group of users works together to approve the transaction. Sometimes, the complexity of the data encrypted in the blocks makes it difficult for a user to decrypt the encoded data alone. So, a group of miners works as a team to solve it. After the validation of the result, the reward is then split between all users.

3. **Cloud Mining**

Cloud mining eliminates the need for computer hardware and software. It’s a hassle-free method to extract blocks. With cloud mining, handling all the machinery, order timings, or selling profits is no longer a constant worry.

Code:

```
const SHA26 = require('crypto-js/sha256')class
Transaction {
constructor(fromAddress , toAddress , amount){
this.fromAddress  =  fromAddress; this.toAddress =
toAddress;
this.amount = amount;
}
}class Block{
constructor(timestapms , transaction , previousHash = ''){
this.timestapms = timestapms;
this.transaction = transaction;
this.previousHash = previousHash;
this.hash = this.calculateHash();
this.nounce = 0;
}
calculateHash(){
return SHA26(this.index + this.previousHash + this.timestapms +
JSON.stringify(this.data) + this.nounce).toString();
}
mineBlock(difficulty){
while(this.hash.substring(0,difficulty) !== Array(difficulty + 1).join("0")){
this.nounce++;
this.hash = this.calculateHash();
}
console.log("Block mined :" + this.hash);
}
}
class Blockchain{
constructor(){
this.chain = [this.createGenesisisBlock()];
this.difficulty = 2; this.pendingTransaction = [];
this.miningReward = 100;
}
createGenesisisBlock(){
return new Block("05/09/2022", "Genesis Block","0");
}getLatestBlock(){
return this.chain[this.chain.length - 1]
}
minePendingTransactions(miningRewardAdress){
let block = new Block(Date.now() , this.pendingTransaction);
block.mineBlock(this.difficulty);
console.log("Block successfully mined !");
this.chain.push(block); this.pendingTransaction =
[
new Transaction(null,miningRewardAdress,this.miningReward)
```

```
];
}
creatTransaction(transaction){
this.pendingTransaction.push(transaction);
}
getBalanceOfAddress(address){
let balance = 0;
for(const block of this.chain){
for(const trans of block.transaction){
if(trans.fromAddress === address){
balance -= trans.amount;
}
if(trans.toAddress == address){
balance +=trans.amount;
}
}
}
return balance;
} }
let ZeeCoin = new Blockchain();
ZeeCoin.creatTransaction(new Transaction('address1', 'address2',100));
ZeeCoin.creatTransaction(new Transaction('address2', 'address1',50));
console.log("\n satrting the miner ... ");
ZeeCoin.minePendingTransactions('Tommy\'s-address');console.log('\nBalance of
Tommy is ',ZeeCoin.getBalanceOfAddress('Tommy\'s address'));
console.log("\n satrting the miner again ... ");
ZeeCoin.minePendingTransactions('Tommy\'s-address');
console.log('\nBalance of Tommy is ',ZeeCoin.getBalanceOfAddress('Tommy\'s-
address'));
```

Output:

```
Zeeshan's PC@MSI MINGW64 /d/SEM-3 practicals/Blockchain-practicals
$ node "d:\SEM-3 practicals\Blockchain-practicals\index.js"

 satrting the miner....
Block mined :00120b535885eb48eacf7a9e8698dc1013f6c39a4a77fed037c11548c4e434f1
Block successfully mined !

Balance of Tommy is  0

 satrting the miner again....
Block mined :0097aded27bba8e107c72a1c8aa128969b0e591a292aff480db1ca99b6ae054c
Block successfully mined !

Balance of Tommy is  100
```

Conclusion:

Hence we successfully implemented Mining puzzles and Transactions.

Practical No.5

Aim: Implement Public Key Cryptosystems

Theory:
Public Key Cryptography:

When the two parties communicate to each other to transfer the intelligible or sensible message, referred to as plaintext, is converted into apparently random nonsense for security purpose referred to as **ciphertext**.

Encryption:
The process of changing the plaintext into the ciphertext is referred to as **encryption.**
The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext.

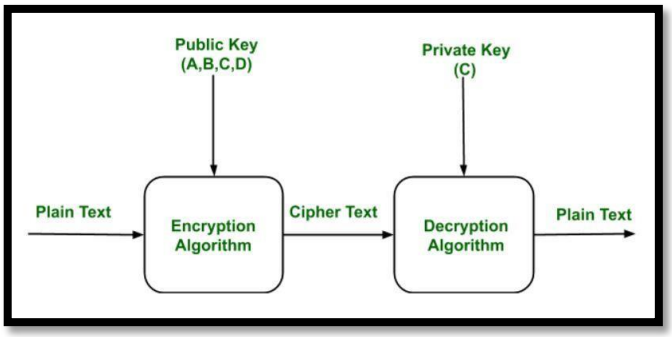
The security of conventional encryption depends on the major two factors:

1. The Encryption algorithm
2. Secrecy of the key

Once the ciphertext is produced, it may be transmitted. The Encryption algorithm will produce a different output depending on the specific key being used at the time. Changingthe key changes the output of the algorithm.
Once the ciphertext is produced, it may be transmitted. Upon reception, the ciphertext canbe transformed back to the original plaintext by using a decryption algorithm and the samekey that was used for encryption.

Decryption:
The process of changing the ciphertext to the plaintext that process is known as **decryption**.

Public Key Encryption : Asymmetric is a form of Cryptosystem in which encryption and



decryption are performed using different keys-Public key (known to everyone) and Private key (Secret key). This is known as **Public Key Encryption.**

Code:

```
const crypto = require("crypto");

// The `generateKeyPairSync` method accepts two arguments:
// 1. The type ok keys we want, which in this case is "rsa"
// 2. An object with the properties of the key
const { publicKey, privateKey } = crypto.generateKeyPairSync("rsa", {
  // The standard secure default length for RSA keys is 2048 bits
  modulusLength: 2048,
});

// use the public and private keys
// ...
// This is the data we want to encryptconst
data = "my secret data";

const encryptedData = crypto.publicEncrypt(
  {
    key: publicKey,
    padding:
    crypto.constants.RSA_PKCS1_OAEP_PADDING,
    oaepHash: "sha256",
  },
  // We convert the data string to a buffer using `Buffer.from`
  Buffer.from(data)
);

// The encrypted data is in the form of bytes, so we print it in base64 format
// so that it's displayed in a more readable form console.log("encypted data: ",
encryptedData.toString("base64"));const decryptedData =
crypto.privateDecrypt(
    {
      key: privateKey,
      // In order to decrypt the data, we need to specify the
      // same hashing function and padding scheme that we used to
      // encrypt the data in the previous step
      padding:
      crypto.constants.RSA_PKCS1_OAEP_PADDING,
      oaepHash: "sha256",
    },
    encryptedData
  );

  // The decrypted data is of the Buffer type, which we can convert to a
  // string to reveal the original data console.log("decrypted data: ",
  decryptedData.toString());
  // Create some sample data that we want to sign
```

```
const verifiableData = "this need to be verified";

// The signature method takes the data we want to sign, the
// hashing algorithm, and the padding scheme, and generates
// a signature in the form of bytes
const signature = crypto.sign("sha256", Buffer.from(verifiableData), {
  key: privateKey,
  padding: crypto.constants.RSA_PKCS1_PSS_PADDING,
});

console.log(signature.toString("base64"));

// To verify the data, we provide the same hashing algorithm and
// padding scheme we provided to generate the signature, along
// with the signature itself, the data that we want to
// verify against the signature, and the public key
const isVerified = crypto.verify(
  "sha256",
  Buffer.from(verifiableData),
  {
    key: publicKey,
    padding: crypto.constants.RSA_PKCS1_PSS_PADDING,
  },
  signature
);

// isVerified should be `true` if the signature is valid
console.log("signature verified: ", isVerified);
```

Output:

```
Zeeshan's PC@MSI MINGW64 /d/SEM-3 practicals/Blockchain-practicals
$ node "d:\SEM-3 practicals\Blockchain-practicals\index.js"
encypted data:  bvybmTfQboSLYRZjht1K65QTkQrMZdFnfbyjRj6p0UE+Zf9rfsZdpQzJIehil9l0nguFzrr6N14qGG5ppVda0vhYOjcSAz5IihVDCF3dyESaSKz5iMjW2NkRnlyTH4QUt1
DgdWD/AcCGYidcFT0KnK95SiET1eAsvou/csAuThS/b705K2QNVo8hgVGtq5ep/OwaQe6B41pz/AObrMgoe4MHVFJV3nOaSWrpfUVRhIHVzrDXs2YKtwXY9snRgM7IBo0W3VKkC+o3KCMMjxM2
/MVvj9vD7z7wESRz4uW56TdKuZGYWf/IYwz+Ya39v+WjYUang+fWgcAfCeG8LXno1Q==
decrypted data:  my secret data
RGvGewXEq/fOwhCYmkfGwH6RMgKsSWzXbLR5X+r7BX4q0Vw0pFGIizAaajU0qEEuoAWs4NHb0Yofs900/a4+/eqw6BcLR91Eyyp/S/CtmjXE5g0KlNloFYqHRRljQ40l7/MNrCdTuBFBdoaE20
A0k4kcMiJ4tGsniy/uv/AVn+EBlrc7xzldDaRK7ZjGv6ck0Iu0Bki41aKxdScA7FmjZlpnvpfjZzAL+rFFalvaBJrzA+bAqy7FF9tkqTekSdzEn9rQBvzeSXT9GLa2qgre4iH+KxnueEKFy05/
tONgdyfexXAC85MDrBwggQeILXkV04q02P6J1BJGfutEtgCV+g==
signature verified:  true
```

Conclusion:

Hence, we successfully implemented Public-key Cryptosystems.

Practical No.6

Aim: Demonstrate cryptocurrency transaction processing

Theory:

Cryptocurrencies run on a distributed public ledger called blockchain, a record of all transactions updated and held by currency holders.

Units of cryptocurrency are created through a process called mining, which involves using computer power to solve complicated mathematical problems that generate coins. Users can also buy the currencies from brokers, then store and spend them using cryptographic wallets.

If you own cryptocurrency, you don't own anything tangible. What you own is a key that allows you to move a record or a unit of measure from one person to another without a trusted third party.

Although Bitcoin has been around since 2009, cryptocurrencies and applications of blockchain technology are still emerging in financial terms, and more uses are expected in the future. Transactions including bonds, stocks, and other financial assets could eventually be traded using the technology.

Transaction speed is one of the prime factors in Blockchains. Cryptocurrency having higher transaction speed results in the **most efficient cryptocurrency**. This means the higher the transaction speed of the Blockchains, the better its ability is to transfer data from one party to the other and confirm transactions. Here it is important to note that its transaction speed depends upon several factors such as block time, block size, transaction fees, and network traffic.

Crypto transaction speed can take a hit if the blockchain you transact has high network congestion, huge volume, and increased transaction fees.

Cryptocurrency with high transaction speed is crucial, and to maintain the speed, some specific factors must be considered.

Code:

```
const SHA26 = require('crypto-js/sha256')class
Block{
constructor(index , timestapms , data , previousHash = ''){
this.index = index; this.timestapms =
timestapms; this.data = data;
this.previousHash = previousHash;
this.hash = this.calculateHash();
this.nounce = 0;
}
calculateHash(){
return SHA26(this.index + this.previousHash + this.timestapms +
JSON.stringify(this.data) + this.nounce).toString();} mineBlock(difficulty){
while(this.hash.substring(0,difficulty) !== Array(difficulty + 1).join("0")){
this.nounce++;
this.hash = this.calculateHash();
}
console.log("Block mined :" + this.hash);
}
}
class Blockchain{
constructor(){
this.chain = [this.createGenesisisBlock()];
this.difficulty = 4;
}
createGenesisisBlock(){
return new Block(0, "09/09/2022", "Genesis Block","0");
}
getLatestBlock(){
return this.chain[this.chain.length - 1]
}
addBlock(newBlock){
newBlock.previousHash = this.getLatestBlock().hash;
newBlock.mineBlock(this.difficulty);
this.chain.push(newBlock);
}
}
let ZeeCoin = new Blockchain();
console.log("mining block 1..");
ZeeCoin.addBlock(new Block(1,"18/09/2021",{amount : 4}));
console.log("mining block 2..");
ZeeCoin.addBlock(new Block(2,"28/09/2021",{amount : 8}));
```

Output:

```
Zeeshan's PC@MSI MINGW64 /d/SEM-3 practicals/Blockchain-practicals
$ node "d:\SEM-3 practicals\Blockchain-practicals\index.js"
mining block 1..
Block mined :0000d974650e46866e99f6564901dcfd30b7d1720bec5dd501f5c58f8c079132
mining block 2..
Block mined :0000b37f9b4cddecf2666bdd5185b06218a5ab01f9e9389f4daed817b886b493
```

Conclusion:

Hence, we successfully implemented cryptocurrency transaction processing.

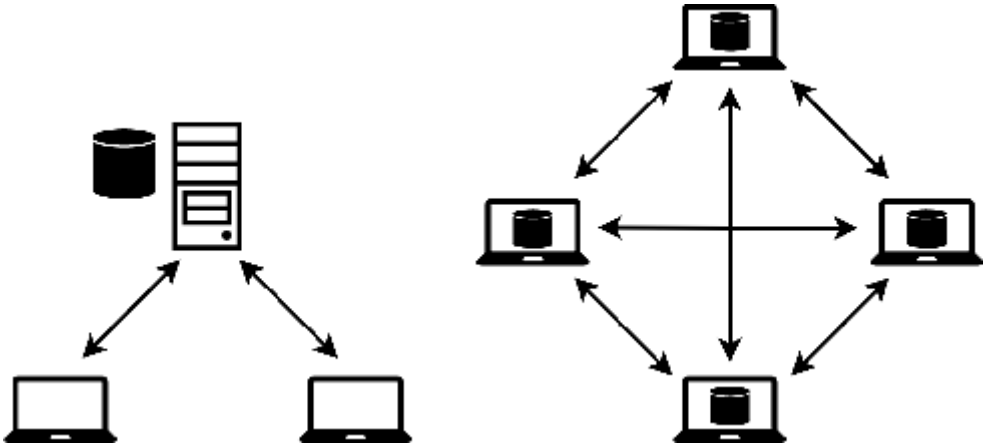
Practical No.7

Aim: Case study on Blockchain Technology and its Applications.

Theory:

1. Introduction

Blockchain is a distributed computing-based technology on a trusted network in which data blocks to be managed are stored in distributed databases hostedon peer-to-peer (P2P) networks. Chains are formed between the blocks so thatthe blocks cannot be arbitrarily revised, and the results of any changes can be accessed. In other words, blockchain is a Distributed Ledger Technology (DLT). The ledger (all the data to be managed) stores and manages transaction information on the computers of participants connected to the P2P network instead of on a centralized server of a certain organization (Kim, 2019; Lin, 2017; IRS, 2019; Nomura Research Institute, 2016)



the left image in Figure 1 illustrates a current and traditional computer network on Internet; the right image shows a blockchain network. Major features of blockchain technology include decentralization, invariance, reliability achieved through consensus processes in untrusted environments, security, economic efficiency, agility, decentralization, the absence of intermediaries, transparency, efficiency, and scalability. Security is achieved as data are commonly owned by multiple participants to prevent hacking. The useof open software and cost reduction related to system development and maintenance, database operations, and security creates economic efficiency.

Decentralization is achieved by storing and managing transaction histories on distributed computers to prevent forgery or illegal use. Non-intermediacy is achieved by enabling transactions between individuals without certification of

any specific third party, and this reduces complexity caused by intermediation and the burden of intermediary fees. When inquiries are allowed to access all information, this results in transparency, which in turn enhances reliability.

Last, scalability is achieved by enabling changes in the blockchain system to fit the use case. Open sources are employed so that the blockchain system can beeasily reconstructed and utilized.

2.Major Technologies in Blockchain :

In blockchain, transaction histories ("blocks") are stored and managed in achain where a cryptographic technique links all past transactions into one sequence.

2.1 Peer-to-peer (P2P) network system

A P2P network is a distributed computer network in which multiple computerscommunicate directly with each other in a one-to-one manner. Information is distributed, stored, and managed on all computers connected to the network.It is a very reliable system because even if several computers were to crash, the entire system is not affected.

2.2 Digital signatures using private keys and public keys

Secret keys in the form of passwords grant users the authority to access theblockchain system, and public keys are used to secure safe, anonymous transactions.

2.3 Hash Function Encryption

This is a core technology used to rapidly detect data manipulation or damage to data using hash values from "hash functions" that map data of arbitrary lengths with fixed data. Put another way, it is a technology that always createsthe same hash values when the input data are the same. It is possible to use the

site http://www.blockchain-basics.com/Hashing.html for a simple test and the site http://www.convertstring.com/ko/Hash/SHA256 to view the SHA256 hashalgorithm.

2.4 Decentralized system

This is a system that can reduce the risk of external hacking attacks because it uses distributed computers instead of a centralized server computer.

2.5 Disintermediation

This is a system that requires no intermediary because all transactions aredone automatically through smart contracts.

2.6 Distributed ledger

All transaction information is stored and managed on individual computers

participating on a P2P network instead of being stored and managed on acentralized server computer.

2.7 Smart contracts

This is a system in which contracting parties preprogram contents agreed uponin advance. The parties then register electronic contracts whose contents are automatically executed when the contract conditions are met so that certain transactions are automatically processed according to the previous arrangements.

2.8 Consensus algorithms

To properly form agreements among an unspecified number of participants,there are consensus algorithms such as Proof of Work (PoW), Proof of Stake(PoS), Delegated Proof of Stake (DPoS), Proof of Importance (PoI), Proof of Authority (PoA), Practical Byzantine Fault Tolerance (PBFT), and so on.

3.Conclusion :

Certainly, blockchain is a technological breakthrough that can deliver and manage critical digital assets on highly reliable network environments; however, if users employ a rudimentary security authentication scheme—suchas passwords— when they want to access a blockchain network, it can cause vulnerability to leakage and illegal or fraudulent use. In such cases, trust in thenetwork will collapse.

It is clear that blockchain technology is emerging as a key infrastructure technology that will lead the Fourth Industrial Revolution. Blockchain infrastructure technology is expected to become very important in promoting mutual harmony and understanding of the complex structure and various social and human phenomena. This technology will lead to a more transparentand fairer society as creators innovate developments in various fields such as politics, economy, culture, education, business, and industry.