

NAME- DEVANSHU SURANA  
ELECTIVE ROLL NO-BT1-18  
PANEL ROLL NO- PC-23  
PRN-1032210755  
PANEL-C

### Lab Assignment - 08

**Title:** - Demonstrate the Blockchain in Python or Java

**Aim:** - To illustrate the fundamental concepts and implementation of a blockchain using Python or Java programming languages.

**Objectives:** - 1) Understand the basic principles of blockchain technology.

- 2) Implement a simple blockchain data structure.
- 3) Create a mechanism for adding new blocks to the blockchain.
- 4) Implement proof-of-work consensus mechanism.
- 5) Demonstrate verification of transactions within the blockchain.
- 6) Explore potential applications and extensions of blockchain technology.

**Theory:** -

Introduction to Blockchain Technology:

Blockchain is a distributed ledger technology that enables secure and transparent record-keeping across multiple participants in a network. It consists of a chain of blocks, each containing a list of transactions. These blocks are cryptographically linked, forming an immutable and tamper-resistant ledger.

Blocks and Transactions:

A block in a blockchain contains a batch of transactions, a timestamp, and a reference to the previous block, forming a linked list structure. Each transaction represents a transfer of assets or data between participants in the network. Blocks are sequentially added to the blockchain, ensuring the ledger's integrity.

Consensus Mechanisms:

Consensus mechanisms are protocols to achieve agreement among nodes in a decentralised network. Proof-of-work (PoW) is a mechanism where participants (miners) compete to solve complex mathematical puzzles to validate transactions and create new blocks. PoW ensures the security and integrity of the blockchain by making it computationally expensive to alter historical records.

Mining and Validation:

Mining is adding new blocks to the blockchain by validating transactions. Miners use computational power to solve cryptographic puzzles and append

new blocks to the chain. Once a miner successfully mines a block, other nodes broadcast it to the network for validation. Validated blocks are added to the blockchain, and miners are rewarded for their efforts.

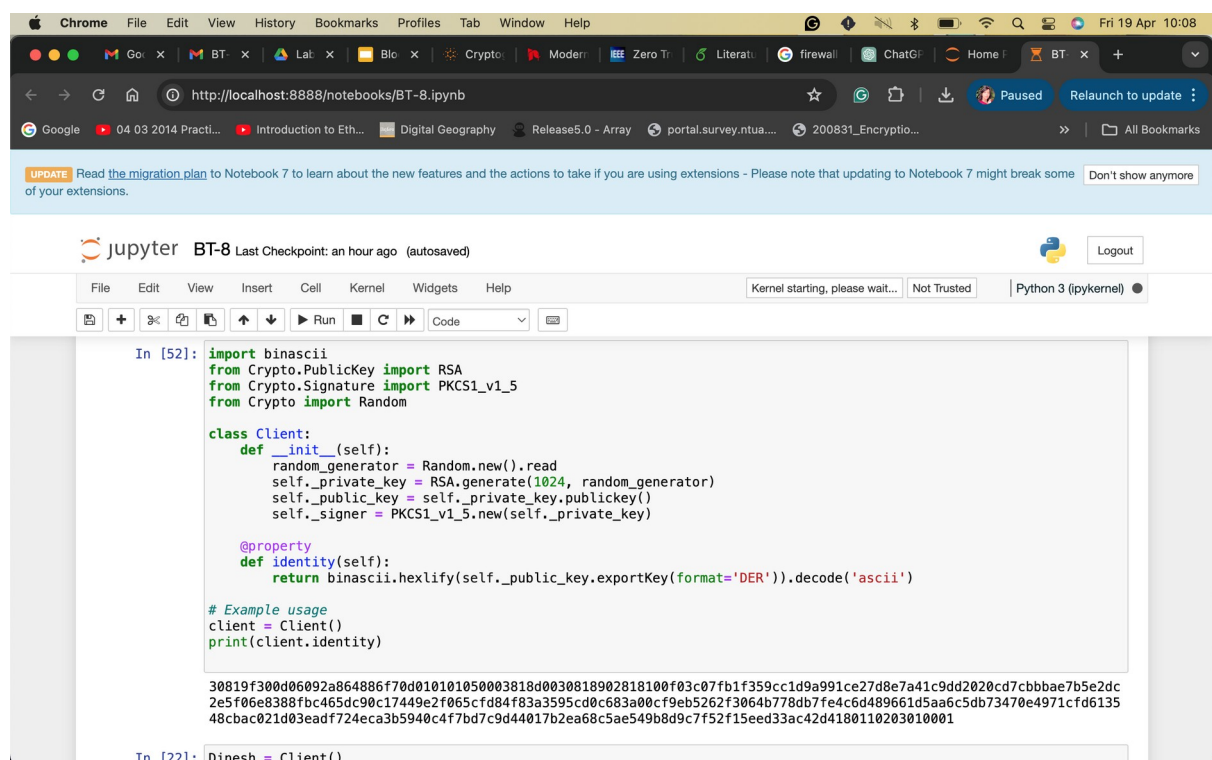
### Security and Immutability:

Blockchain achieves security through cryptographic techniques such as hashing and digital signatures. Each block contains a cryptographic hash of the previous block, making it computationally infeasible to alter historical records without affecting subsequent blocks. Additionally, digital signatures ensure the authenticity and integrity of transactions, preventing unauthorised modifications.

### Smart Contracts:

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They enable automated and trustless transactions between parties, eliminating the need for intermediaries. Smart contracts are deployed on blockchain platforms like Ethereum, allowing for decentralised applications (DApps) and programmable digital assets.

### Screenshots:



The screenshot shows a Jupyter Notebook interface in a Chrome browser. The notebook is titled 'BT-8' and shows the execution of Python code to generate a client identity and keys. The code defines a 'Client' class with methods for generating random keys and a property for the client's identity. The output of the code is a long hexadecimal string representing the client's identity.

```
In [52]: import binascii
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto import Random

class Client:
    def __init__(self):
        random_generator = Random.new().read
        self._private_key = RSA.generate(1024, random_generator)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

# Example usage
client = Client()
print(client.identity)

30819f300d06092a864886f70d010101050003818d0030818902818100f03c07fb1f359cc1d9a991ce27d8e7a41c9dd2020cd7cbbbae7b5e2dc
2e5f06e8388fbc465dc90c17449e2f065cdf84f83a3595cd0c683a00cf9eb5262f3064b778db7fe4c6d489661d5aa6c5db73470e4971cfd6135
48cbac021d03eadf724eca3b5940c4f7bd7c9d44017b2ea68c5ae549b8d9c7f52f15eed33ac42d4180110203010001

In [22]: Dinesh = Client()
```

Figure 1: This image shows the client identity, a hexadecimal value randomly generated of 1024 bits. These are the public and private keys that are generated for the client.

```
In [19]: import binascii
from Crypto.PublicKey import RSA
from Crypto import Random

class MyClass:
    def __init__(self):
        random_generator = Random.new().read
        self._private_key = RSA.generate(1024, random_generator)
        self._public_key = self._private_key.publickey()

    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

# Example usage
my_instance = MyClass()
print(my_instance.identity())
```

```
30819f300d06092a864886f70d010101050003818d0030818902818100bb4810b644753a9ec9b28d56aab8a0a2b650c607e29e683edd6b0a876
3f0f625fac7b11184106230bc42342638c0f5386be149c88b75296fb8a47478c36dfe5be268cd0a7a7179d23f2066eb16ae375385f6bb1bbf9e
819599c374558e16acebb48dcf70b9eb0a001541ea3321099ab94d87edc0e2a7867de11808c5e5e6e5150203010001
```

Figure 2: The public key that is generated will be returned as an instance of the MyClass as the identity of the Client. Anybody could send virtual currency to you using this identity and it will get added to your wallet.

```
In [22]: Dinesh = Client()
print (Dinesh.identity())
```

```
30819f300d06092a864886f70d010101050003818d00308189028181009a65e380aa03b0642ba0e30e236f8b82771bfe2093782355337ff0696
97513a0fba93005193348db71960fdc99462ed252eef13dbd0a712ec774dce849a6e3baec77cc6f858e8cb7b330a4d9cdb8b51acd18e5438798
7d3445f959c747e30e4b826818b277f3dac609e71b1dca4dc212fad71a6d90aa673479fea01e51398c750203010001
```

Figure 3: The public key generated is printed in the figure. We can see that the public key is a hexadecimal value used in the transaction and can be accessed by the receiver.

```
In [59]: t = Transaction(
    client, # Assuming Dinesh is the sender
    Ramesh, # Assuming Ramesh is the recipient
    5.0
)

transaction_dict = t.create_transaction_dict()
print(transaction_dict)
```

```
OrderedDict([('sender', '30819f300d06092a864886f70d010101050003818d0030818902818100f03c07fb1f359cc1d9a991ce27d8e7a4
1c9dd2020cd7cbbbae7b5e2dc2e5f06e8388fbc465dc90c17449e2f065cfd84f83a3595cd0c683a00cf9eb5262f3064b778db7fe4c6d489661d
5aa6c5db73470e4971cfd613548cbac021d03eadf724eca3b5940c4f7bd7c9d44017b2ea68c5ae549b8d9c7f52f15eed33ac42d418011020301
0001'), ('recipient', <__main__.Client object at 0x11668c8d0>), ('value', 5.0), ('time', datetime.datetime(2024, 4,
19, 9, 45, 19, 324386))])
```

Figure 4: The figure shows a sample transaction setting with the client and the recipient Ramesh. The value sent from the client is 5.0. A timestamp is also included in the transaction. The whole transaction is printed.

```
In [60]: signature = t.sign_transaction()
print (signature)
```

```
73649a8af271aec6c2c5f606ba851b4440bed4ae440b747e6d651024e81b808af2f24e892965eab433c172ec8f822a82684125cdcb858f59830
472a58a22591028c7784fed65ad41cada55f3d498cd424f1c02d70b0c4baca83b6aefc081c4322717cfab7a1c2e791bc71e3f74e43918d89136
8176343e67c07732a43983ead2
```

Figure 5: The signature is the main indication that the transaction is valid. It tells both the client and the receiver that the transaction is valid and approved by both parties.

```
In [65]: t1.sign_transaction()
transactions.append(t1)
```

```
In [66]: t2 = Transaction(
    Dinesh,
    Seema.identity,
    6.0
)
t2.sign_transaction()
transactions.append(t2)
t3 = Transaction(
    Ramesh,
    Vijay.identity,
    2.0
)
t3.sign_transaction()
```

Figure 6: Here, multiple transactions are shown. The transactions are displayed in a list. For example, t2 occurs between Dinesh and Seema and transfers the value of 6.0. t3 occurs between Ramesh and Vijay and transfers the value of 2.0.

```
In [67]: for transaction in transactions:
    display_transaction(transaction)
    print('-----')
```

```
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100cfd3a55bf6fc9335a24c9d05a40ecdd043711ba64447cf9e5
05ee72d846e57b05ebd0bc2681f3df91a7619d6a96bb966873b501fe8dc9dc554cbad9df994452de394043b0484b19e4752f36f66be525ae8f4
206429d1196c60ec27aeb32781c537ed894ee9e93835879ba78b7ed776678e217bf6720b70ff77fc54b183cdfb690203010001
-----
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100d0ead92d2a44dd8ac9ec3d59d87687c96d3babe665704b
5f9ab8449de10924b955a5c722a60333008417604b514064ca38610bb48e7f29c5d2dbbfa433b4f65764a02863376c2ed81439b900268506aba
0cff1206dae4c0ef51f9908d0241298f52b7b03d9e1ee06536e6cbcdf71fec8a47759588ba5dd970ca66bf60faadee10203010001
-----
value: 15.0
-----
time: 2024-04-19 09:45:59.370912
-----
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100cfd3a55bf6fc9335a24c9d05a40ecdd043711ba64447cf9e5
05ee72d846e57b05ebd0bc2681f3df91a7619d6a96bb966873b501fe8dc9dc554cbad9df994452de394043b0484b19e4752f36f66be525ae8f4
206429d1196c60ec27aeb32781c537ed894ee9e93835879ba78b7ed776678e217bf6720b70ff77fc54b183cdfb690203010001
-----
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100d97a2ce5a427543ed30d8f8e113e22892ce410715b734d
300f8e02acbd9f969726cbbd221e5d2c433b1322518ee1a464f16f9367af695847aaebca44e5751c21f39088cfcc5b184097ba808006befeb4d5
33c6f66b30445e37e0d77e30443e105f0803400e1000600e3741e03244b3b50e3d410255e64400014000e410203010001
```

Figure 7: Here, all the multiple transactions are displayed. For distinction, the transactions are separated by a dashed line. We print all transactions beginning with the first transaction except for the Genesis transaction, which was never added to this list.

## FAQs:

### 1. What is blockchain technology?

- Blockchain technology is a decentralised ledger system that securely records transactions across multiple parties in a network. It enables transparency, security, and immutability of data without the need for a central authority.

### 2. How does the proof-of-work consensus mechanism work?

- Proof-of-work is a consensus mechanism where miners compete to solve complex mathematical puzzles to validate transactions and create new blocks.

This process involves significant computational power and ensures the integrity and security of the blockchain.

3. What are some real-world applications of blockchain beyond cryptocurrency?

- Blockchain technology has various applications beyond cryptocurrency, including supply chain management, healthcare data sharing, voting systems, decentralised finance (DeFi), and identity verification.

4. Is blockchain technology secure?

- Blockchain technology is inherently secure due to its cryptographic principles, decentralised nature, and consensus mechanisms. It provides tamper-resistant data storage and ensures transaction integrity through cryptographic hashing and digital signatures.

5. What are the limitations of blockchain technology?

- Despite its benefits, blockchain technology faces challenges such as scalability, energy consumption (in proof-of-work systems), regulatory concerns, and interoperability between different blockchain networks.

Conclusion:

In conclusion, blockchain technology represents a transformative innovation with profound implications across various industries. Throughout this exploration, we've delved into the foundational principles, mechanisms, and applications of blockchain in both theory and practice.

Blockchain's decentralized nature, secured through cryptographic techniques and consensus mechanisms like proof-of-work, ensures trust and transparency in transactions without intermediaries. Its immutable ledger provides a tamper-resistant record of data, fostering accountability and reducing the risk of fraud.

Beyond its roots in cryptocurrency, blockchain has applications in supply chain management, healthcare, finance, voting systems, and more. It offers solutions to challenges such as data security, transparency, and efficiency.

However, while blockchain holds immense promise, it also faces hurdles related to scalability, energy consumption, regulatory complexities, and interoperability. Addressing these challenges will be crucial for unlocking blockchain's full potential and realising its benefits on a global scale.

As we continue to explore, innovate, and adapt blockchain technology, it's essential to maintain a nuanced understanding of its capabilities, limitations,

and ethical considerations. By doing so, we can harness the power of blockchain to drive positive change, foster innovation, and build a more transparent and equitable future.