Introduction to Blockchains

Why Blockchains

Traditionally, there have been **middlemen** who have been controlling the data. For instance, Facebook is the middleman between users and advertisers. Banks are middlemen between borrowers and lenders.

The issue here is that the data is controlled by a **central authority** - Facebook and banks. They, therefore, control the prices and also technically own the data. In this scenario, all the power lies with a middleman and users have to trust them to remain ethical and continue to think about user interest. Blockchain was invented to make the **data decentralized** and trust minimized with any centralized party.

If databases could be decentralized. This solves a lot of issues:

- You can directly connect to a borrower without a middleman like a bank. You will get more profits.
- You can connect directly to advertisers to tell them the kind of ads that you like, without going through Facebook.

Blockchain Programming Fundamentals

Digital Signature (Hash)

Digital Signature is basically a **function** that takes a **string** as input and returns a **fixed-size alphanumeric string**. The **output string** is known as the Digital Signature or the Hash of the input message. The important point to note here is that the function via which we obtain the Digital Signature is "**irreversible**" – given an input string, it can compute the Hash. However, given the Hash, it is virtually impossible to compute the input string. Further, it is also virtually impossible to find 2 values that have the same Hash.

Data Structure

In a Blockchain, we have an ordered **chain of blocks** such that each block contains the following information:

- Hash of the previous block.
- List of transactions.
- Hash of itself = `Hash(List of transactions, Hash of the previous block)`

```
e.g. [0, "X paid $100 to Y", 91b452]
     [91b452, "Y paid $20 to Z, X paid $10 to P", 8ab32k]
```

Such design is for data security. As soon as we modify a block, the hashes of all subsequent blocks become invalid and so, the chain collapses.

Code Example

```
def get_parent_hash(block):
    return block[0]


def get_transactions(block):
    return block[1]


def get_hash_itself(block):
    return block[2]


def create_block(parent_hash, transactions):
    hash_itself = hash((transactions, parent_hash))
    return [parent_hash, transactions, hash_itself]


def create_genesis_block(transactions):
    return create_block(0, transactions)


def alter_transaction(block, new_transaction):
    block[1] = new_transaction
    block[2] = hash((block[1], block[0]))


def main():
```

```
    block0 = create_genesis_block("X paid $100 to Y")
    print("First block: %s" % str(block0))

    block1 = create_block(get_hash_itself(block0), "Y paid $20 to Z, X paid $10 to P")
    print("Second block: %s" % str(block1))

    assert get_hash_itself(block0) == get_parent_hash(block1), "Invalid blockchain"

    print("!! Altering the transaction of the first block:")
    alter_transaction(block0, "Y paid $100 to X")

    print("First block: %s" % str(block0))
    print("Second block: %s" % str(block1))

    assert get_hash_itself(block0) == get_parent_hash(block1), "Invalid blockchain"
```

Output:

```
Traceback (most recent call last):
  File "/Users/xinrong.meng/blockchain_toy/blockchain.py", line 45, in <module>
    main()
  File "/Users/xinrong.meng/blockchain_toy/blockchain.py", line 41, in main
    assert get_hash_itself(block0) == get_parent_hash(block1), "Invalid blockchain"
AssertionError: Invalid blockchain
First block: [0, 'X paid $100 to Y', -2083942197780410976]
Second block: [-2083942197780410976, 'Y paid $20 to Z, X paid $10 to P', 1766665025057485219]
!! Altering the transaction of the first block:
First block: [0, 'Y paid $100 to X', 5352704281595772096]
Second block: [-2083942197780410976, 'Y paid $20 to Z, X paid $10 to P', 1766665025057485219]
```

Think of Blockchain as a distributed and secured data structure that can be used in places where no middlemen are involved. The decentralized nature of Blockchain is what helps in removing the middlemen and it comes from the above immutability of Blockchain.

Applications of Blockchains

Cryptocurrency

By implementing Blockchain, parties are able to transact with each other without the involvement of any bank. For instance, a person sitting in the United States can transfer bitcoins to one based out of India without intervention from any bank. This leads to the creation of a lot of cryptocurrencies, Bitcoin being the most popular one.

Supply Chain

An international courier has to go through a lot of steps. For instance, it goes through the courier service provider (like DHL), then goes through customs of the sending country, then through customs of the receiving country, and finally through the local courier service provider at the receiving country. The biggest issue in this supply chain is to **track the status of the shipment**. Companies are planning to implement Blockchain across these parties so that all the parties involved can put status in real-time in the Blockchain which customers can easily track. Using Blockchain eliminates the management onus on one party and helps in decentralizing the load across all the parties.

e.g. https://aws.amazon.com/blockchain/blockchain-for-supply-chain-track-and-trace/

Healthcare

The health records of patients can be securely stored in a Blockchain so that when the patient visits another doctor, he/she can directly share those records with the new doctor. The best part about using Blockchain here is that there is no need for a centralized portal where these records are stored. Therefore, the cost can be lowered significantly

e.g. https://www.ibm.com/blockchain/industries/healthcare

References*

https://hackr.io/blog/blockchain-programming-beginners-guide
https://hackr.io/blog/applications-and-use-cases-of-blockchains
https://aws.amazon.com/managed-blockchain/
https://www.ibm.com/blockchain