

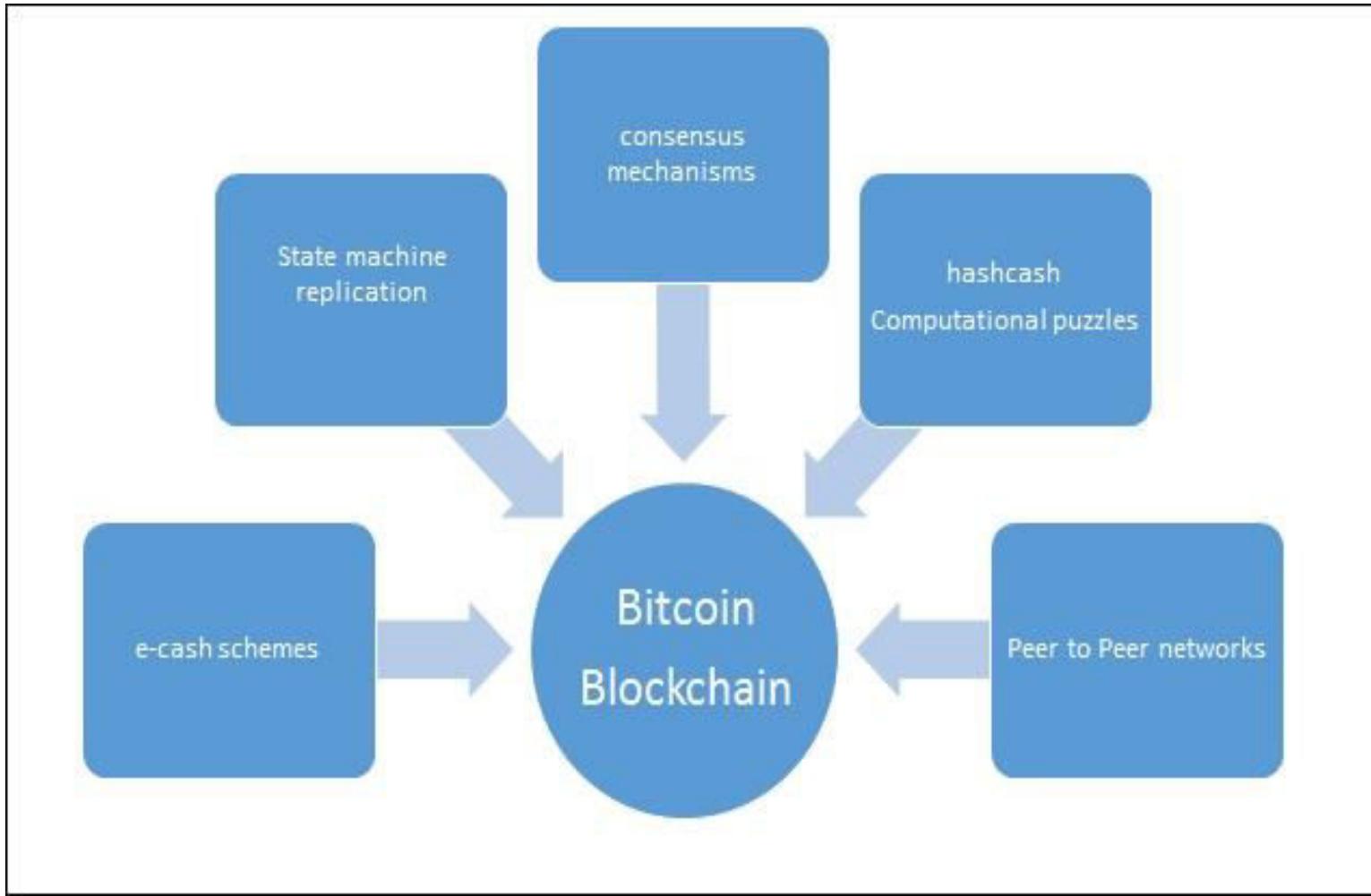
Blockchain Architecture & Block Structure

Dr. Bhawana Rudra
Assistant professor

Department of Information Technology
National Institute of Technology Karnataka

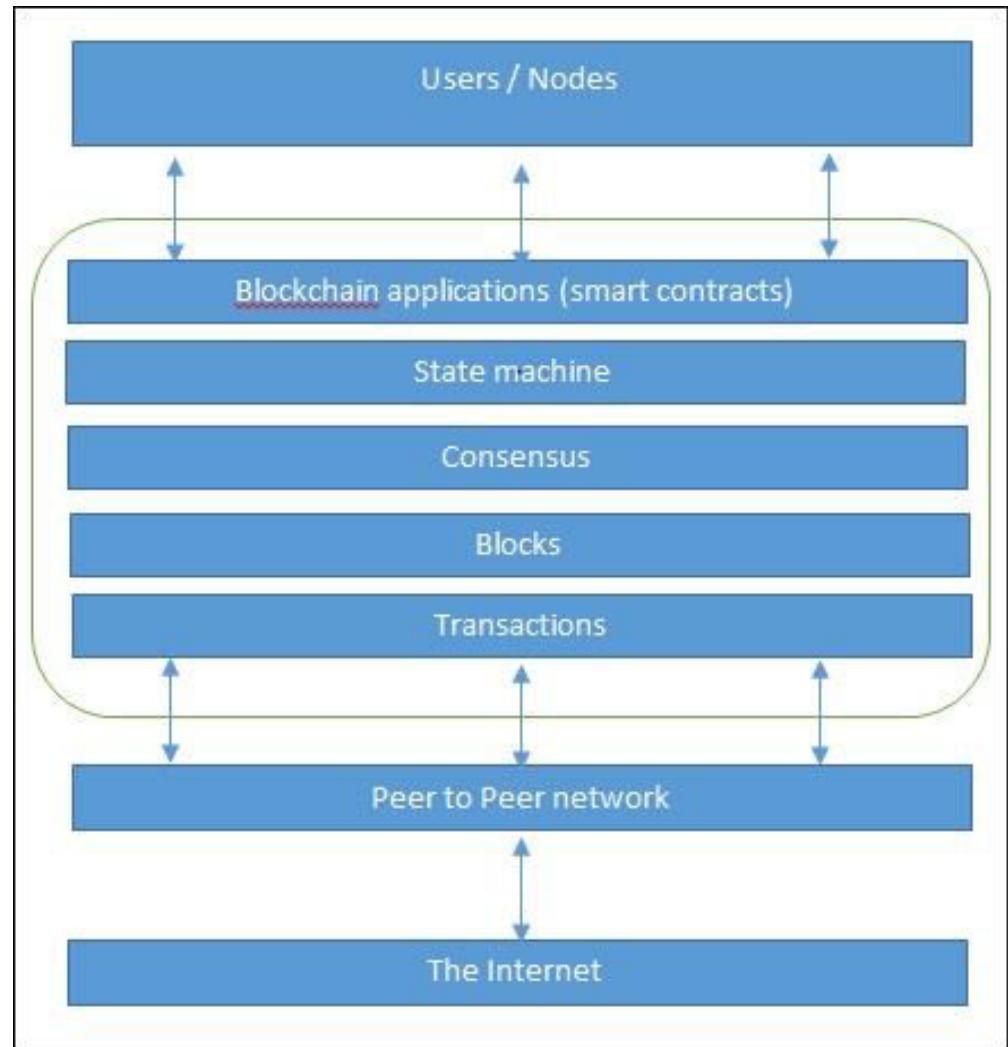
- Disclamier: The information and images are taken from various sources of web.

Invention Of Bitcoin and Blockchain-Various Ideas

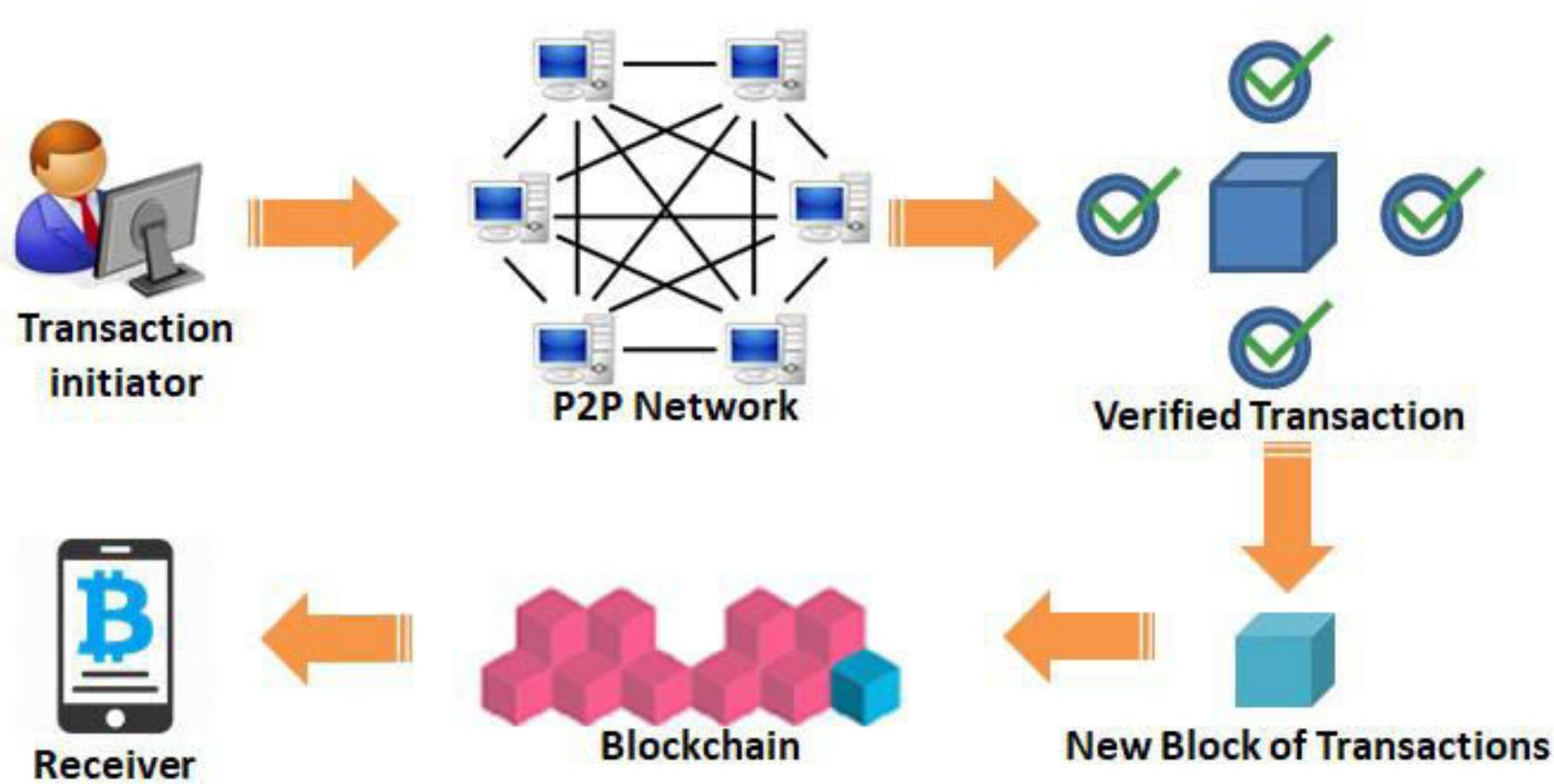


Blockchain

- It is analogous to SMTP, HTTP, or FTP running on top of TCP/IP
- It is a Peer-to Peer Distributed Ledger that is secure, append only, immutable and updateable using agreement among the peers.



How Blockchain Works



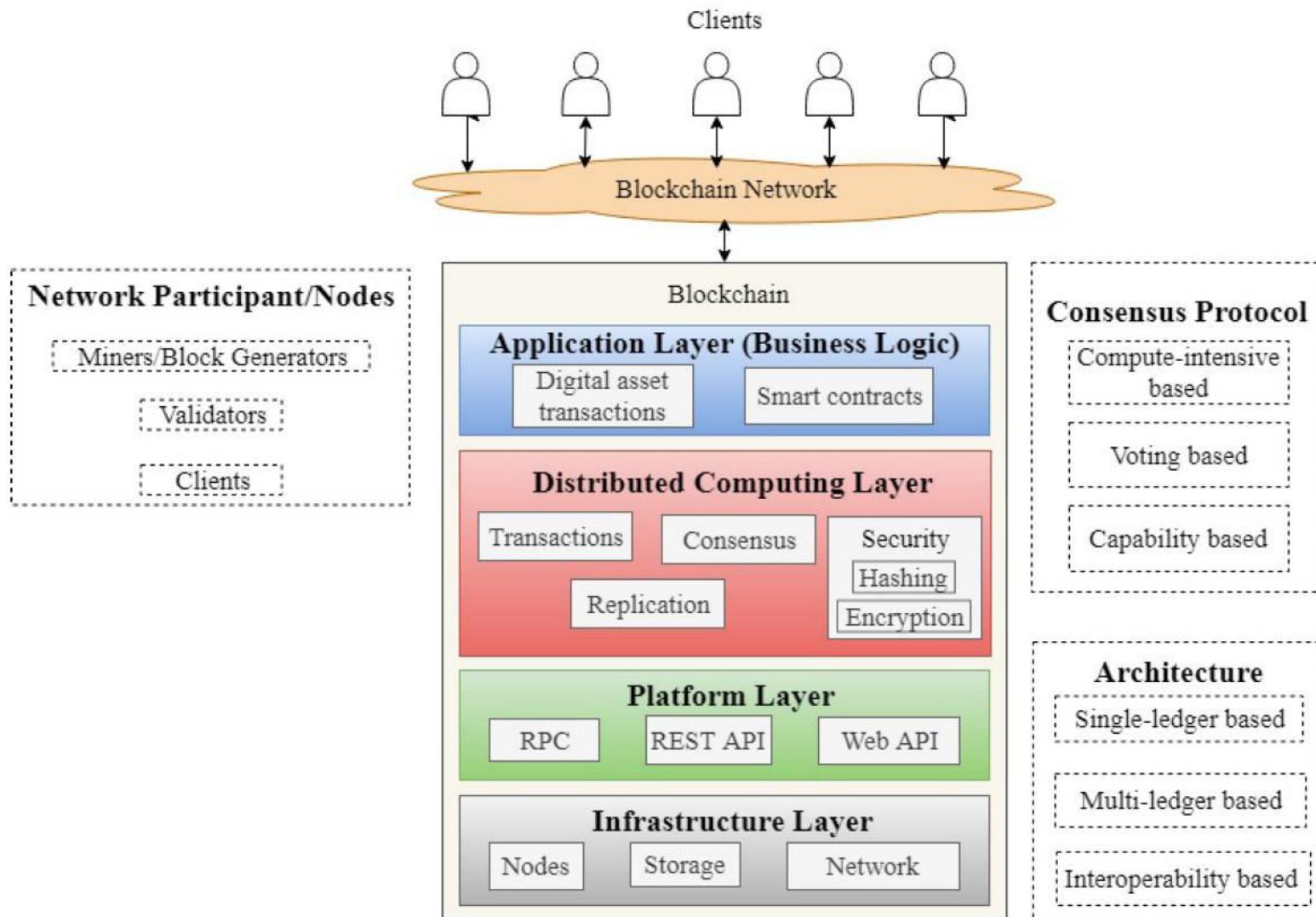
Features of Blockchain

- **Distributed consensus**
- **Transaction verification**
- **Platforms for smart contracts**
- **Transferring value between peers**
- **Generating cryptocurrency**
- **Smart property**
- **Provider of security**
- **Immutability**
- **Uniqueness**
- **Smart contracts**

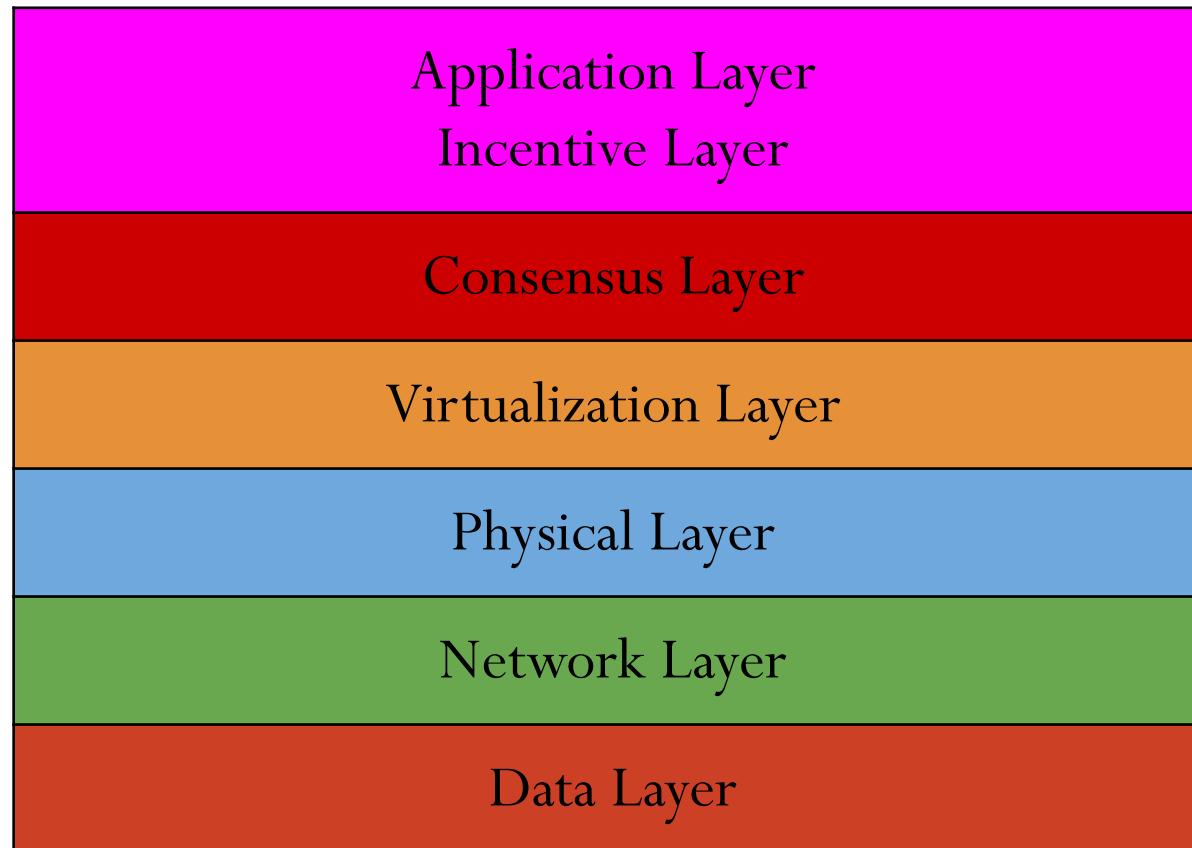
- **Tamper-Proof**
- **Address Space**--consists of 160-bit address space allows $1.46 * 10^{48}$ devices and provides 4.3 billion addresses that provides a scalable solution
- **Authentication and Integrity**
- **Authentication, Authorisation and Privacy**
- **Secure Communications**
- **Scalability:** GHOST is a protocol that improves scalability. The Inter Planetary File System (IPFS) was used to store decentralised and shared files enabling peer-to-peer file system
- **Anonymity and Privacy**—The protocols like Dark Wallet, Dash, MixCoin, CoinShuffle, CoinSwap increase security
- **Persistency**
- **Auditability**

Tiers of Blockchain technology

- **Blockchain 1.0:** basically used for cryptocurrencies
- **Blockchain 2.0:** for financial services and contracts
- **Blockchain 3.0:** general-purpose industries such as government, health, media, the arts, and justice.
- **Generation X (Blockchain X):** public blockchain service available that anyone can use just like the Google search engine



Blockchain Architecture



Data Layer



This layer acts as the **blockchain data structure** and **physical storage**. The ledger is built using a **linked list or Merkle trees**, of blocks, which are encrypted using **asymmetric encryption**. The Data layer consists of the following components such as **Blocks, Merkle Trees and Transactions, Blockchain**.

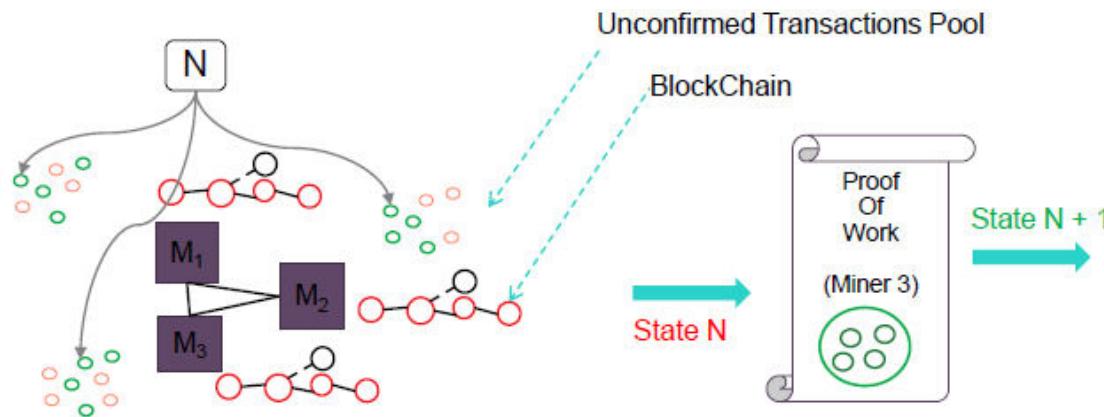
Transactions

- Transactions are the **smallest building blocks** of a blockchain system. They normally consist of a recipient address, a sender address, and a value.
- Transactions are bundled and delivered to each node in the form of a block. As new transactions are distributed throughout the network, they are **independently verified and "processed"** by each node.
- Each transaction is **time-stamped** and collected in a block.
- It is this continuous movement of data that builds up the **blockchain architecture**. Each transaction can have single/multiple inputs and outputs. Here input means the reference value from a previous transaction, and output means the amount and address.

The transaction life cycle

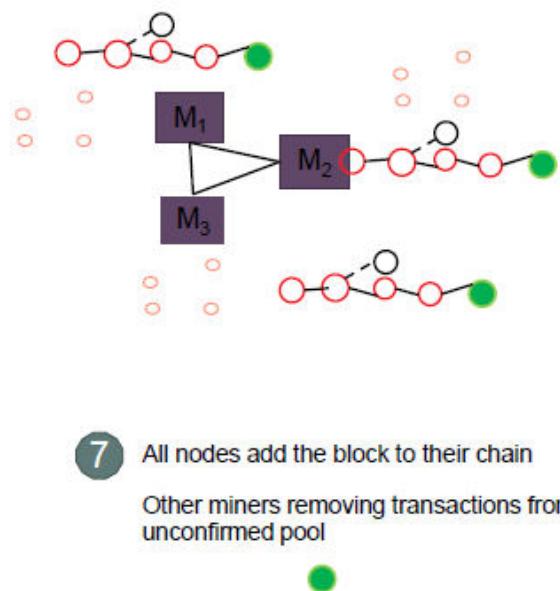
1. A user/sender sends a transaction using wallet software or some other interface.
2. The wallet software signs the transaction using the sender's private key.
3. The transaction is broadcasted to the Bitcoin network using a flooding algorithm.
4. Mining nodes include this transaction in the next block to be mined.
5. Mining starts once a miner who solves the Proof of Work problem broadcasts the newly mined block to the network.
6. The nodes verify the block and propagate the block further, and confirmation starts to generate.
7. Finally, the confirmations start to appear in the receiver's wallet and after approximately six confirmations, the transaction is considered finalized and confirmed.

Transaction Cycle



- 1 Nodes Generate Transactions.
○ ○
- 2 Nodes broadcasts Transactions to network.
- 3 Miner Nodes adds Transactions unconfirmed pool.

- 4 Miners create block and starts work on proof of work.
In this case all Green colored pebbles
- 5 First miner to generate proof of work wins
- 6 Broadcast Block and proof of work to other nodes



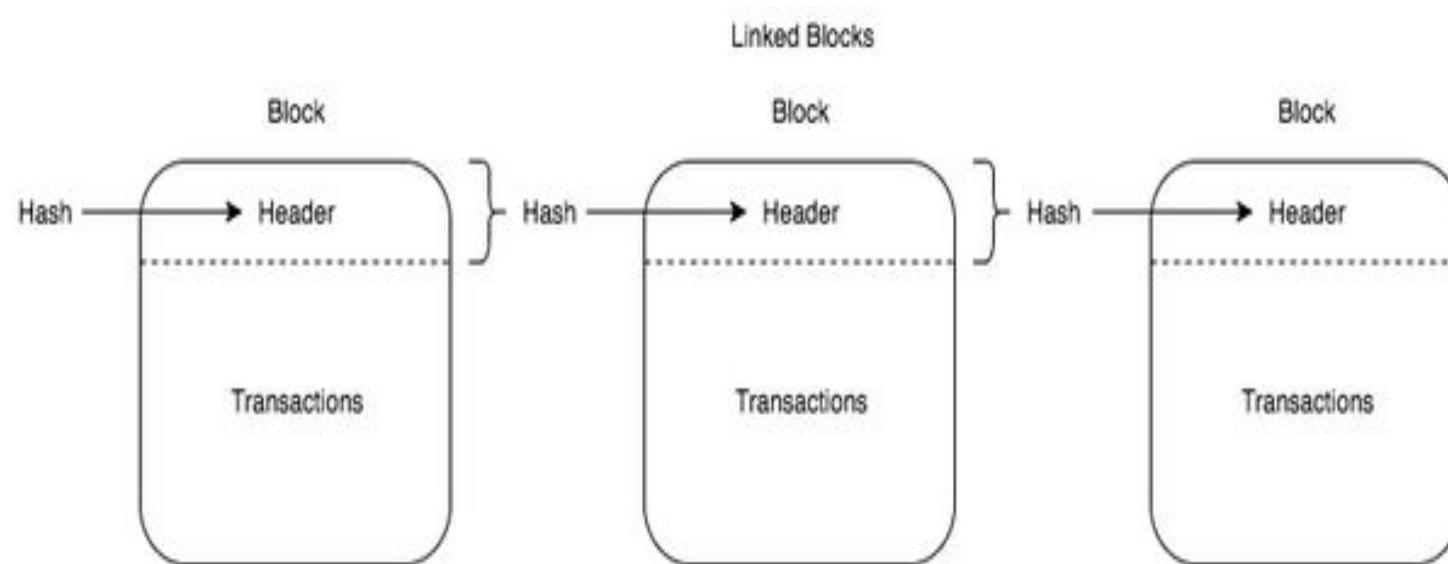
- 7 All nodes add the block to their chain
Other miners removing transactions from unconfirmed pool

Types of Transactions

- **Pay to Public Key Has(P2PKH):**Commonly used to send the transactions to the bitcoin addresses
- **Pay to Script Hash(P2SH):** Along with use of Script hash, a redeem hash is also evaluated and must be valid
- **Multisig:** M of N signatures are to be paid to construct a script and validate with multiple signatures inorder to redeem a transaction
- **Pay to Pubkey:** It is simple and was used in coinbase which is absolete. In this the public key is stored within the script and to unlock we require to sign the transaction with private key
- **Null data/ OP_RETURN:** The script is used to store the arbitrary data for a fee and the limit of data storage is 40bytes

Blocks(Contd...)

- Blockchains are **probabilistic systems**, by design.
- **Nodes**, or **the computers** in the network, independently decide and concur upon which "chain of blocks" is the **longest** and **most valid**.
- As a block is created and set around the network, each node processes the block and decides where it fits into the current overarching **blockchain ledger**.

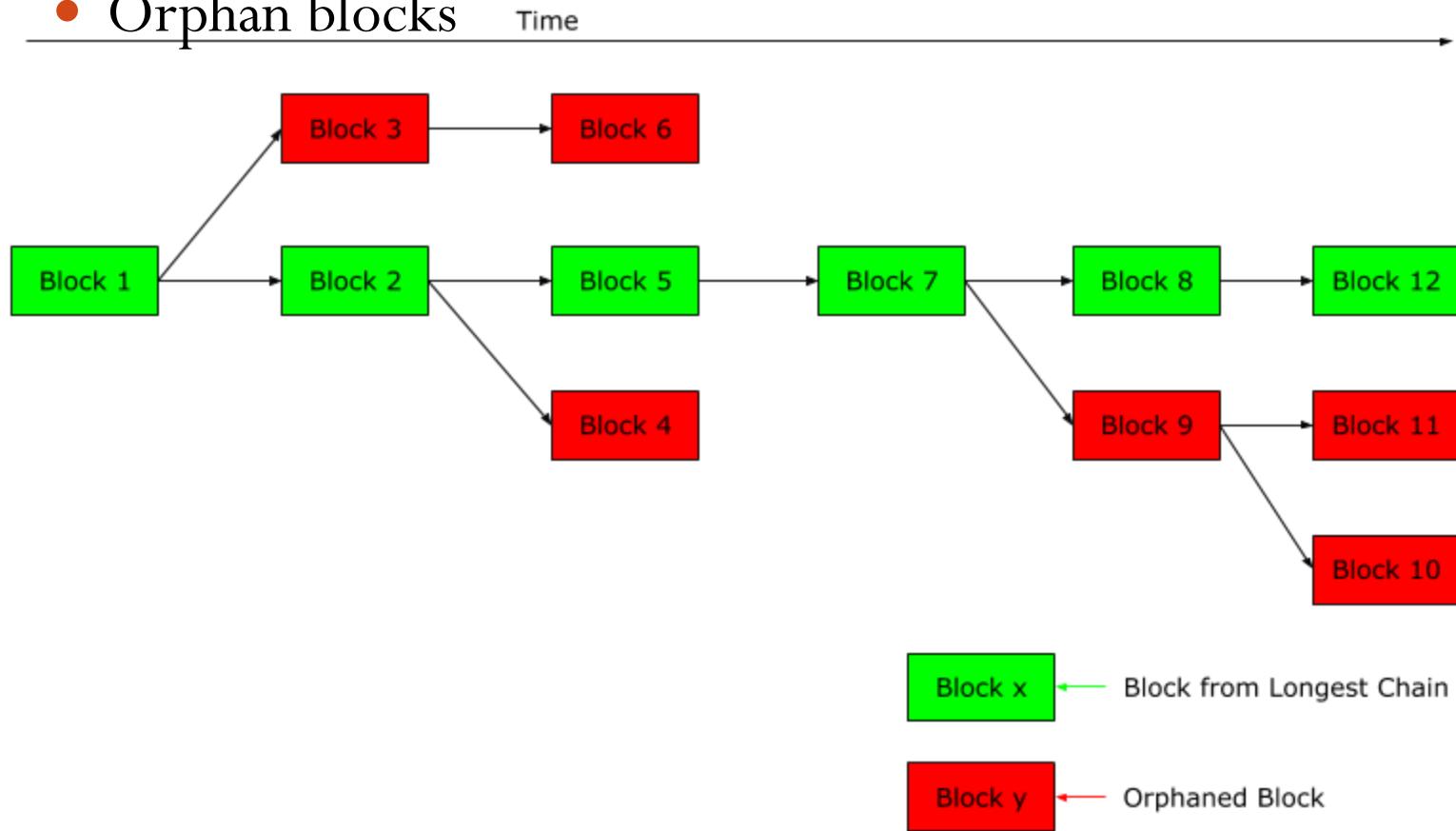


Blocks(Contd...)

In terms of the **blockchain architecture**, there are different types of blocks based on their functionalities:

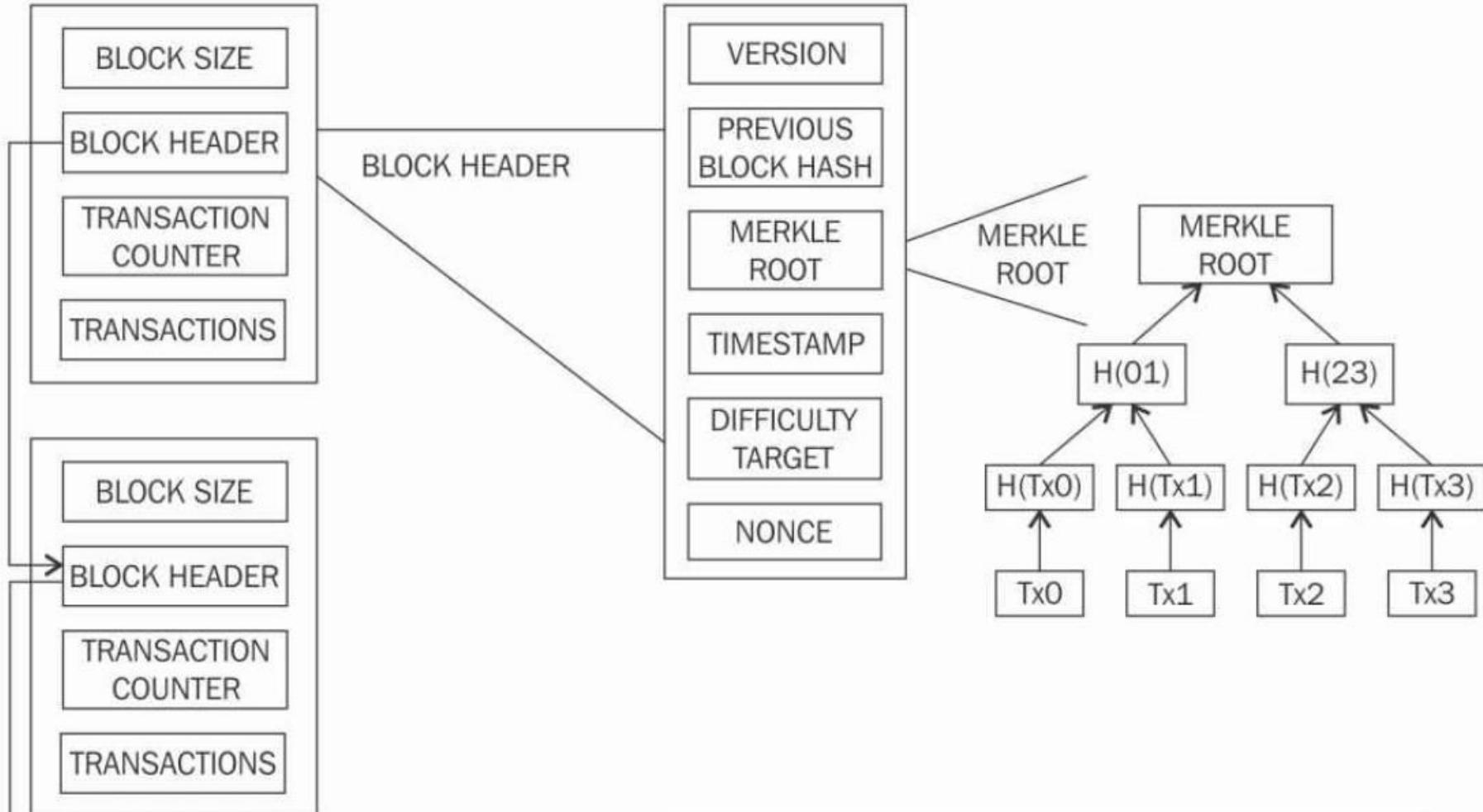
- **Main branch blocks** – The ones that extend the main **blockchain network** in current use.
- **Side branch blocks** – These refer to parent blocks that aren't present in the current **blockchain**.
- **Orphan blocks** – These refer to parent blocks unknown to the node analyzing the current **blockchain**.

- Stale blocks
- Orphan blocks



Block Structure

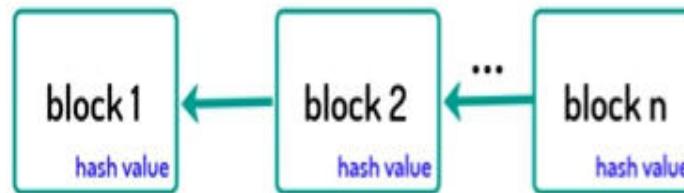
— BLOCK HEIGHT —



Merkle Tree

- Merkle trees are binary trees containing **cryptographic hashes**.
- It is constructed by recursively hashing pairs of nodes until there is only one hash, called the *root*, or *merkle root*. The cryptographic hash algorithm used in merkle trees is **SHA256** applied twice, also known as **double-SHA256**.
- It is a **binary tree**, it needs an even number of leaf nodes. If there is an odd number of transactions to summarize, the last transaction hash will be duplicated to create an even number of leaf nodes, also known as a *balanced tree*.
- **There are two ways the blocks can be identified. These are cryptographic hash and block height.** The Merkle Root summarizes all of the data in the related transactions, and is stored in the block header.

Merkle Tree(Contd...)



Transactions

transaction 1 0A2750887547FA08F762DC34B

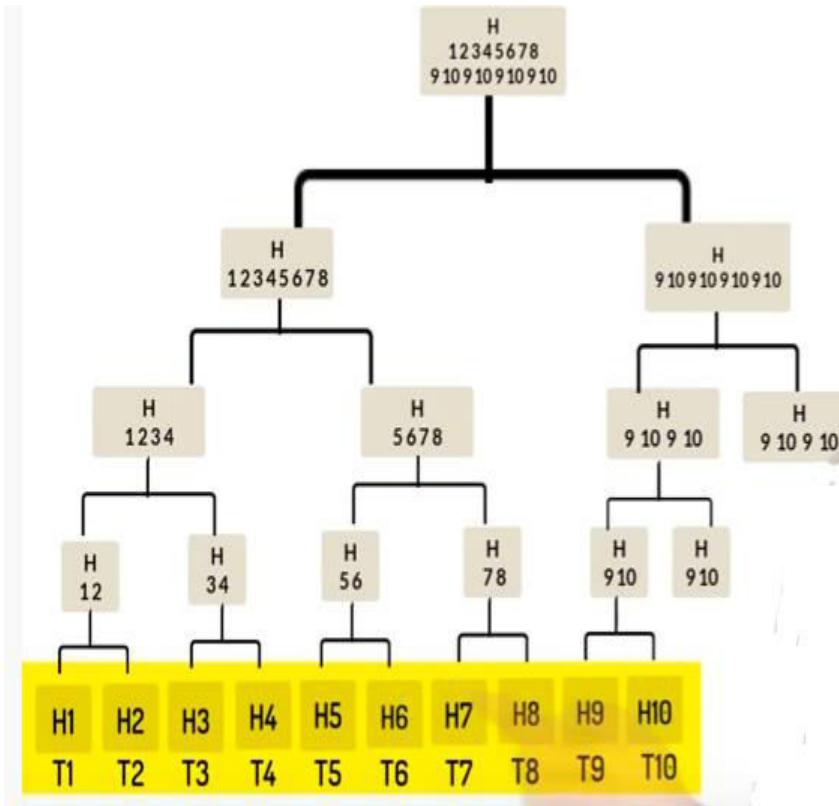
transaction 2 0A2750887547FA08F762DC34B

transaction 3 0A2750887547FA08F762DC34B

.

.

transaction n 0A2750887547FA08F762DC34B

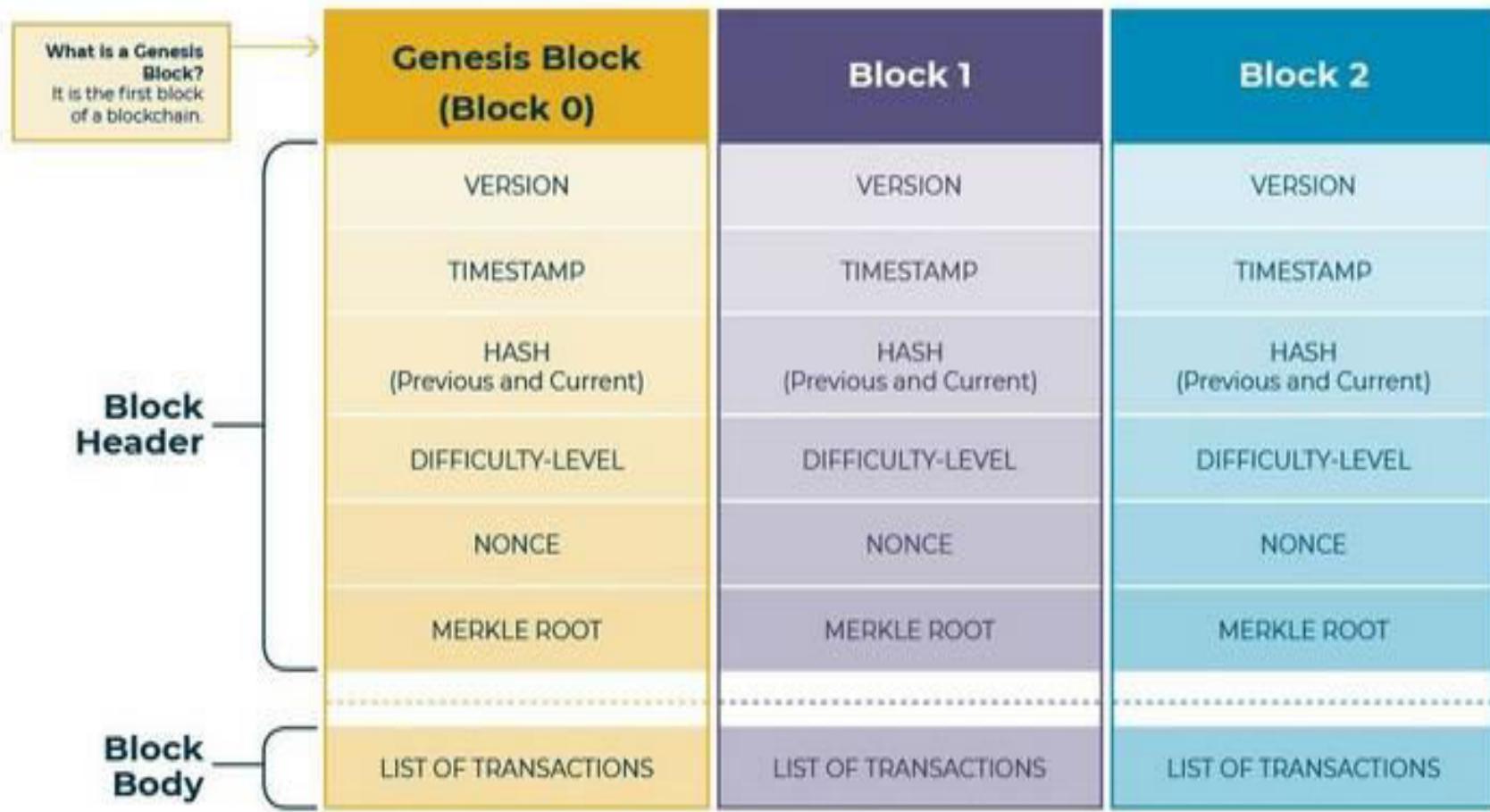


H1 = SHA256(SHA256(Transaction 1))
H12= SHA256(SHA256(H1+H2))

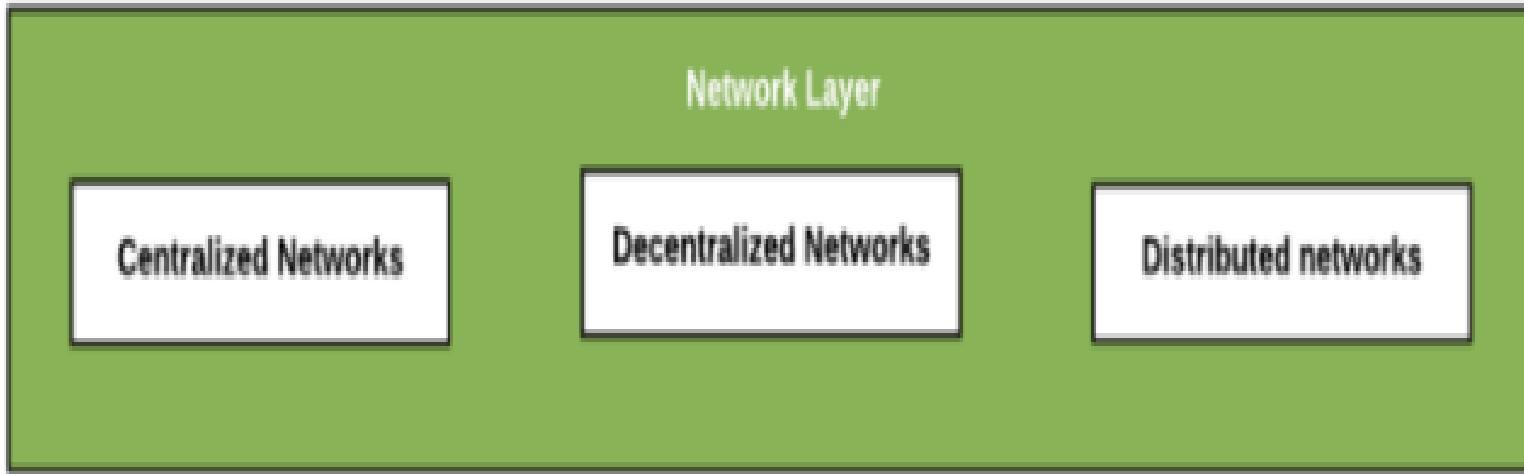
Blockchain

- A blockchain will thus consist all the transactions defined by user as linked together by storing the **hash of the previous block** , where each block stores the **root hash of the Merkle tree** where the actual transactions are stored.
- Transmission of blockchains across a given network is further secured using **asymmetric encryption** or **public private key pairs** .

Blockchain(Contd...)



Network Layer



The blockchain makes use of the **distributed network**, such that everyone downloads and interacts with all the information that is available on the Blockchain.

P2P network is used to propagate/broadcast transactions among nodes.

Network Layer(Contd...)

P2P Network:

- ✖ The blockchain is a **peer to peer (P2P) network** working on the IP protocol.
- ✖ A P2P network is a flat topology with **no centralized node**. All nodes equally provide and can consume services while collaborating via a **consensus algorithm**.
- ✖ Peers contribute to the computing power and storage that is required for the upkeep of the network. **P2P networks** are generally **more secure** because they do not have a single point of attack or failure as in case of a centralized network.

P2P Network(Contd...)

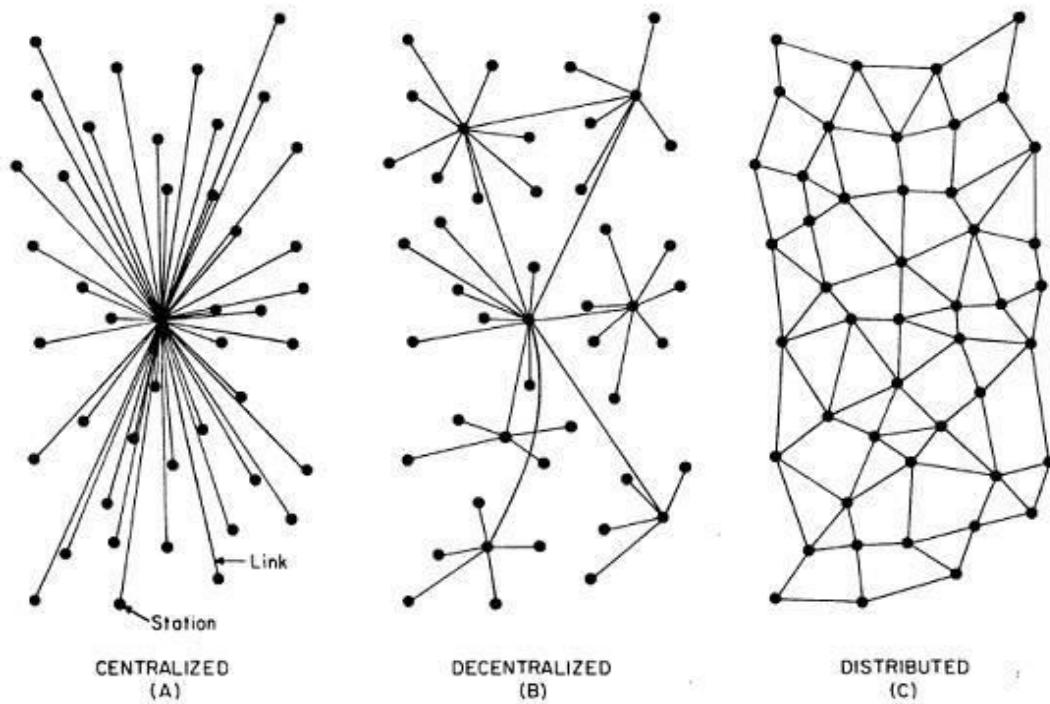
Different types of Blockchain

- ✖️ Public blockchains
- ✖️ Private blockchains
- ✖️ Semi-private blockchains
- ✖️ Sidechains
- ✖️ Permissioned ledger
- ✖️ Distributed ledger
- ✖️ Shared ledger
- ✖️ Fully private and proprietary blockchains
- ✖️ Tokenized blockchains
- ✖️ Tokenless blockchains

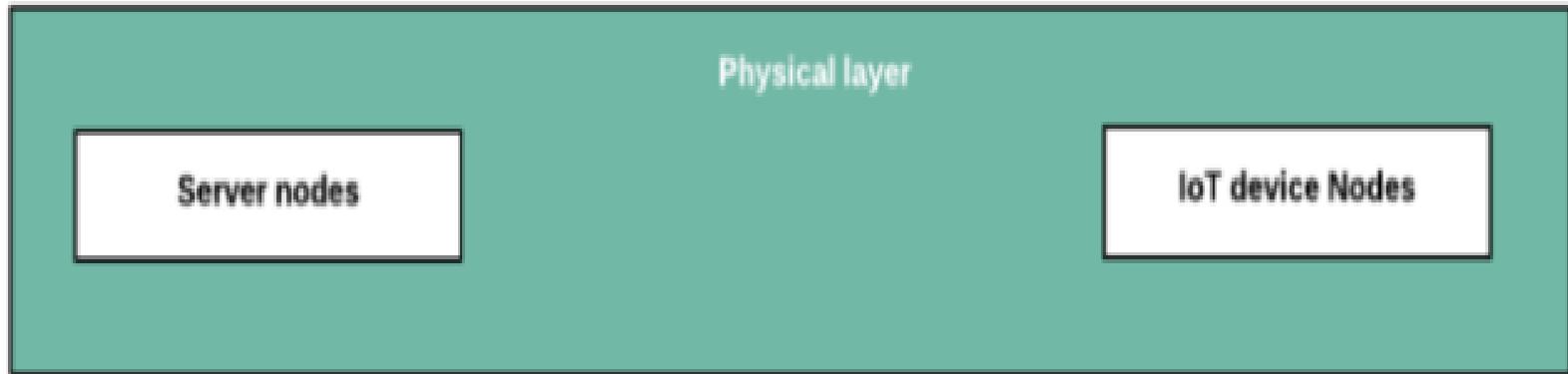
Network Layer(Contd...)

There are mainly 3 types of networks utilized by blockchain platforms

- Centralized networks
- Decentralized networks
- Distributed networks



Physical Layer



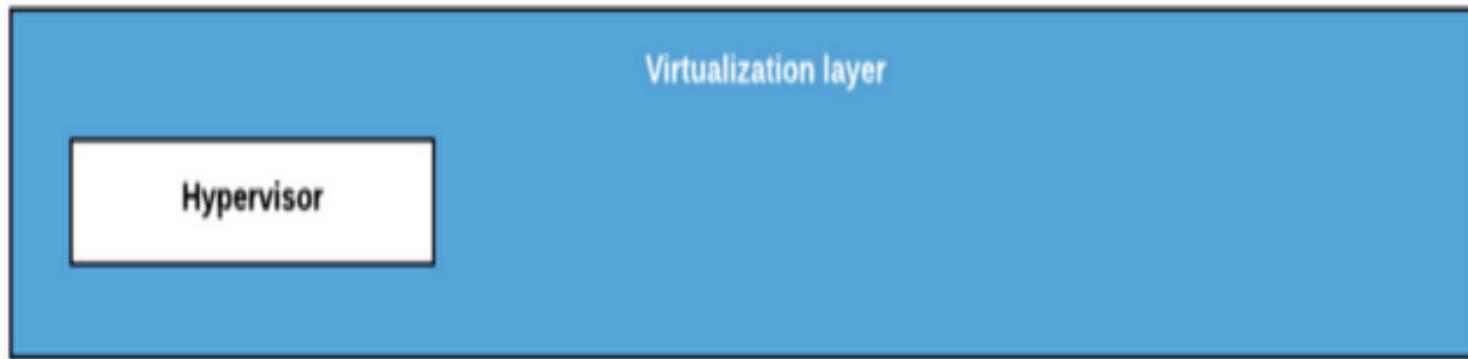
Physical Layer(Contd..)

1. It consists of **servers, edge nodes, IoT devices** which act as nodes on the blockchain network.
2. These are generally connected as a **P2P network** where Peers are equally privileged, equipotent participants in the application.
3. A node can be any **active electronic device, including a computer, phone or even a printer**, as long as it is connected to the internet and as such has an IP address.
4. The role of a node is to **support the network** by maintaining a copy of a blockchain and, in some cases, to process transactions.
5. Nodes are often arranged in the structure of trees, known as **binary trees**.

Physical Layer(Contd..)

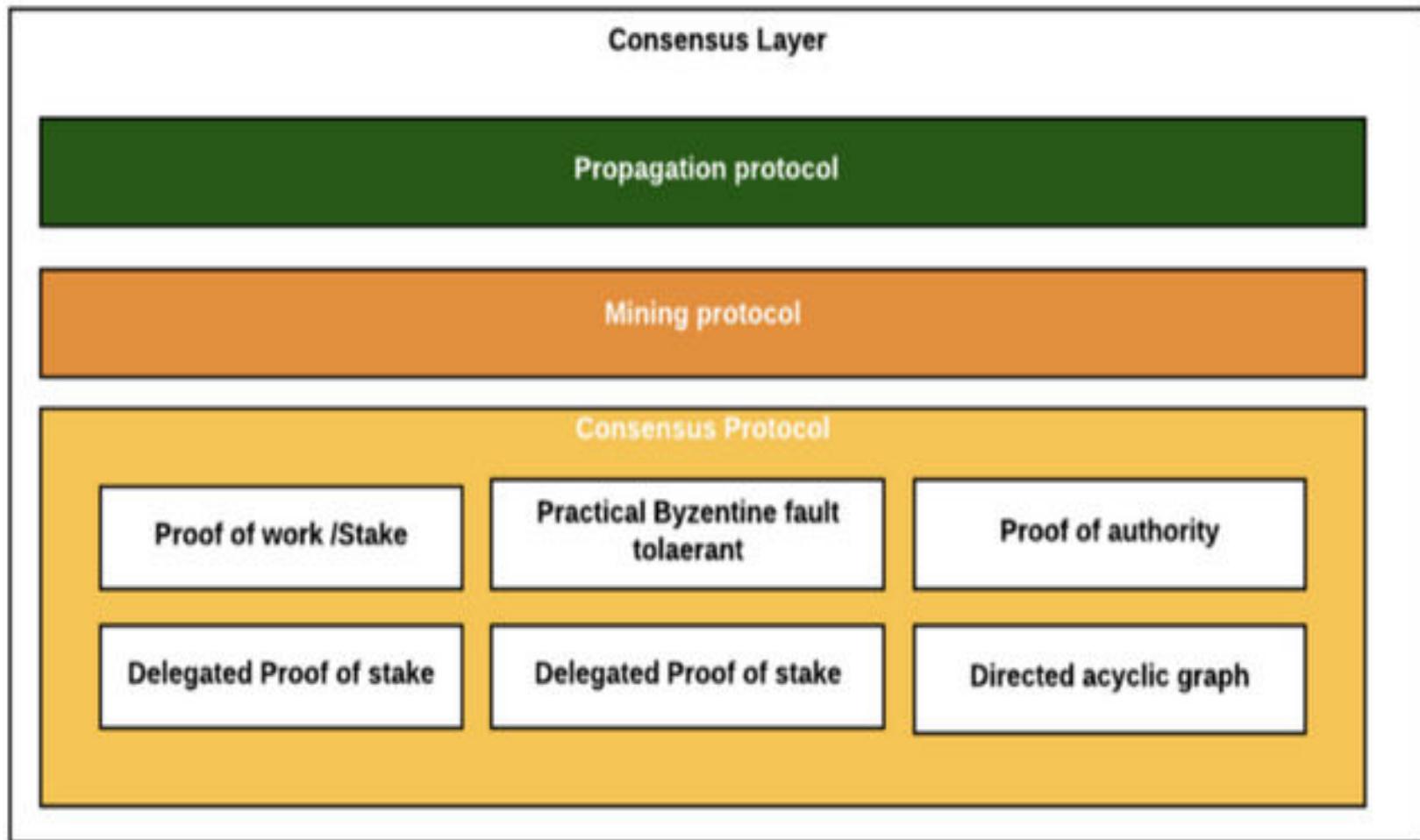
6. Processing these transactions can require **large amounts of computing and processing power**, meaning that the average computer's capabilities are inadequate.
7. A **full node** downloads a **complete copy of a blockchain** and checks any new transactions coming in based on the **consensus protocol** utilized by that particular cryptocurrency or utility token.
8. All nodes use the same consensus protocol to remain compatible with each other. It is the nodes on the network that **confirm and validate transactions**, putting them into blocks.
9. Nodes always come to their own conclusion on whether a transaction is valid and should be added to a block with other transactions, irrespective of how other nodes act.

Virtualization Layer



Hardware virtualization layer used to allocate **hardware and resources to virtual machines**

Consensus Layer



Consensus Layer

- This layer deals with the **enforcement of network rules** that describe what nodes within the network should do to reach consensus about the broadcasted transactions. It also deals with the **generation and verification of blocks**.
- Without consensus, blockchain is simply a way of storing encrypted/unencrypted data.
- Consensus allows it to **decentralized** because all the nodes in network follow the same rules and it will maintain uniformity in all the copies of a blockchain.
- So each and every change in a single blockchain is verified and adopted by another blockchain in the network.

Consensus Layer(Contd...)

- When we speak about consensus, we mean the collaborative process that participating nodes of the network use to agree that a transaction is valid and to keep the **distributed ledger synchronized at all times**.
- These consensus mechanisms **lower the risk of malicious** transactions because they would have to occur (or be executed) across many locations at the same time, or else the tampering will be noticed almost immediately by other nodes.
- To reach consensus, the **majority of the participants** need to agree that the transaction is **valid** before it is permanently recorded in the ledger.

Consensus Layer(Contd...)

- Once a transaction is **permanent**, no one, not even a system administrator, can delete the transaction from the ledger.
- The **cost and time** needed to reach consensus depends on the **mechanism in place** and **the number of nodes participating in the consensus**.
- The process of reaching a consensus on a high level include **mining** and propagation of **blocks** in the network defined by the consensus algorithm.

Consensus Layer(Contd...)

- **Example:** Two miners, **miner A and miner B**. Both miner A and miner B can decide to include **transaction X** into their block.
- A block has a maximum size of data. On the Bitcoin blockchain, the maximum size of a block is data up to **1 MB**.
- But before adding the transaction to their block, a miner needs to check if the transaction is eligible to be executed according to the **blockchain history**.
- If the senders' wallet balance has sufficient funds according to the existing blockchain history, the transaction is considered valid and can be added to the block.

- Miners will usually prioritize transactions that have a high transaction fee set, because this gives them a higher reward.
- This mined block is then propagated to other nodes where the transaction is individually executed until a majority of them are in the same state

There are various consensus models used in blockchains .These Blockchain consensus models consist of some **particular objectives, such as:**

- × **Coming to an agreement:** The mechanism gathers all the agreements from the group as much as it can.
- × **Collaboration:** Every one of the group aims toward a better agreement that results in the groups' interests as a whole.
- × **Co-operation:** Every individual will work as a team and put their own interests aside.
- × **Equal Rights:** Every single participant has the same value in voting. This means that every person's vote is important.

Contd...

- × **Participation:** Everyone inside the network needs to participate in the voting. No one will be left out or can stay out without a vote.
- × **Activity:** every member of the group is equally active. There is no one with more responsibility in the group.

Incentive Layer

Incentive Layer

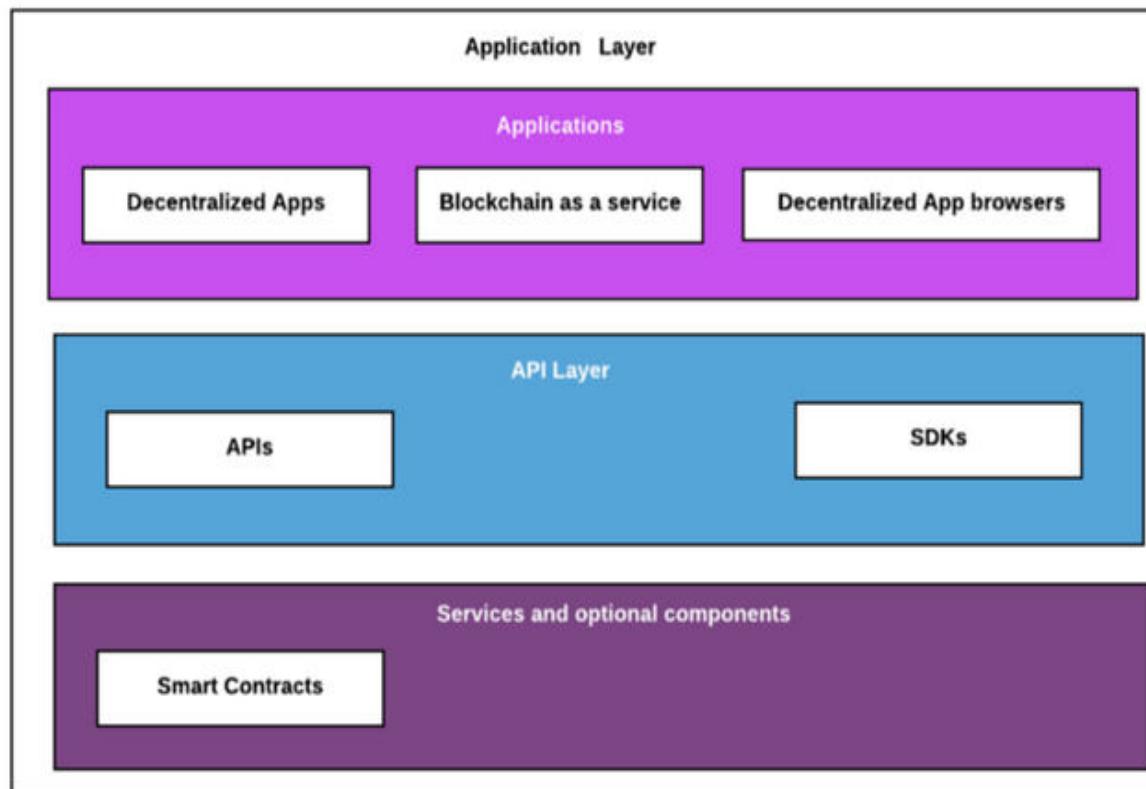
Rewards distribution

Transaction Fee

Incentive Layer

- ✗ This layer deals with the **distribution of rewards** that are earned by nodes in the network for the work they do to reach consensus. Whether this layer is implemented or not depends on the **consensus mechanism** in use.
- ✗ The incentive layer include capabilities that describe what kinds of incentives are given by the network, when and how incentives can be earned by nodes, and the minimum amount of transaction fees needed to perform actions on the blockchain.

Application Layer(Aplications, API Layer and Contract Layer)



Contract Layer

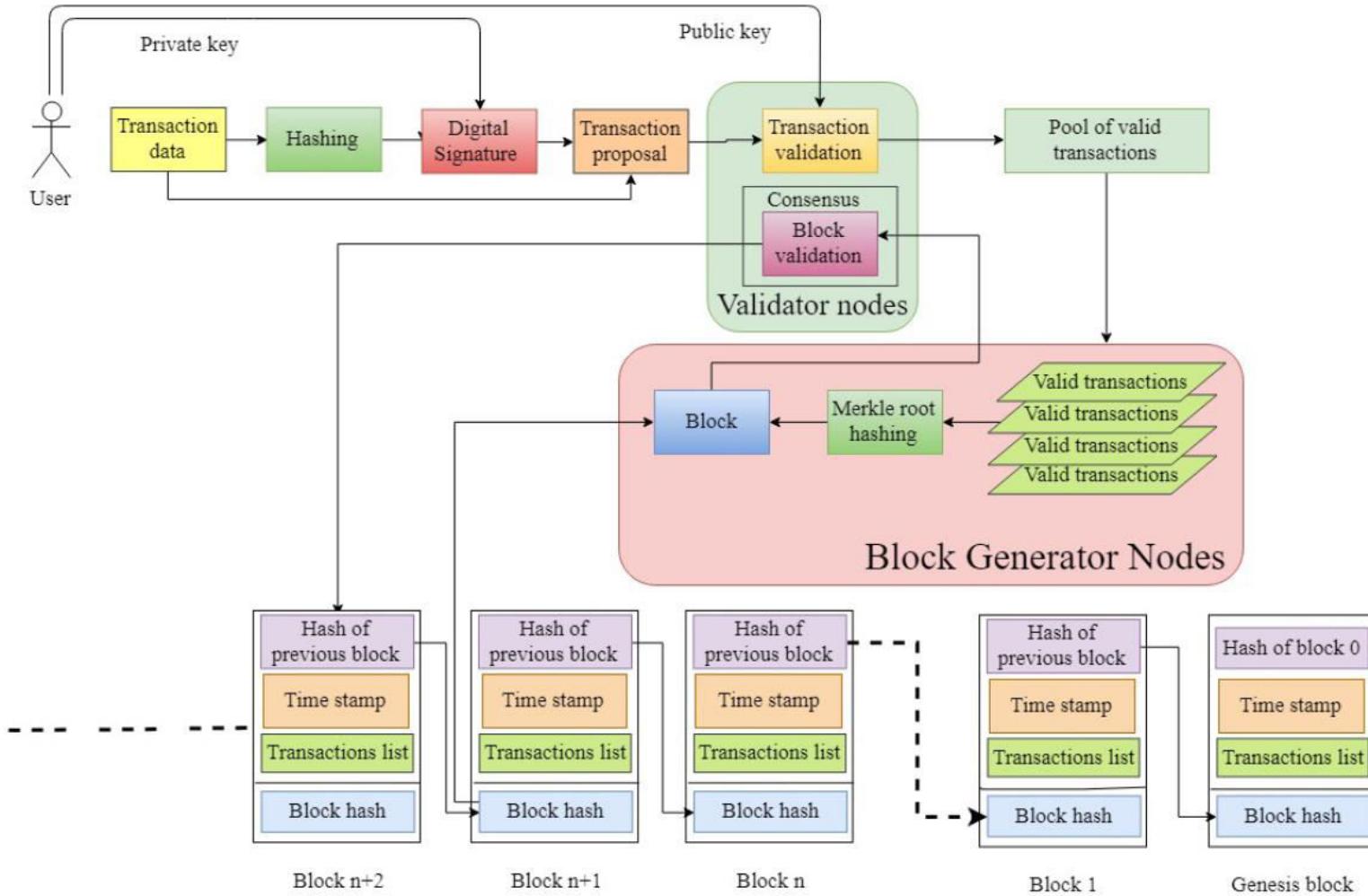
It consists of the **services and optional components** and serves to enable integration of the blockchain platform with other technologies like

- × **Data feeds**
- × **Smart contracts**
- × **On chain and off chain computing**
- × **Digital wallets**
- × **Digital ids**
- × **Multi signatures**
- × **Oracles**
- × **DAOs and governance**
- × **State channels**

API Layer

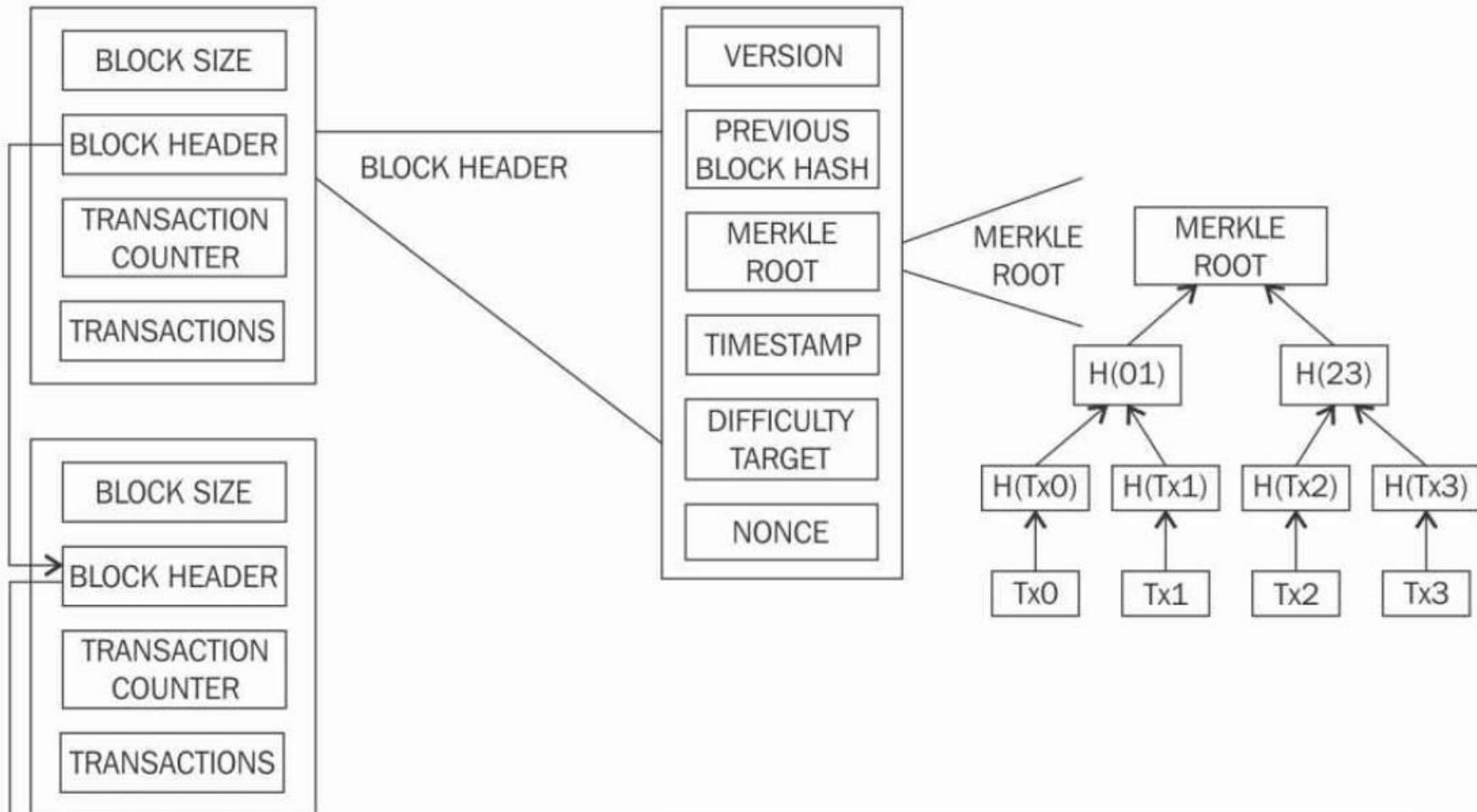
- It provides **application interfaces** on top of the blockchain, and how third party applications can interact with the **digital ledger and smart contracts**.
- The application layer includes **capabilities** that provide applications on top of the blockchain.

Blockchain works...

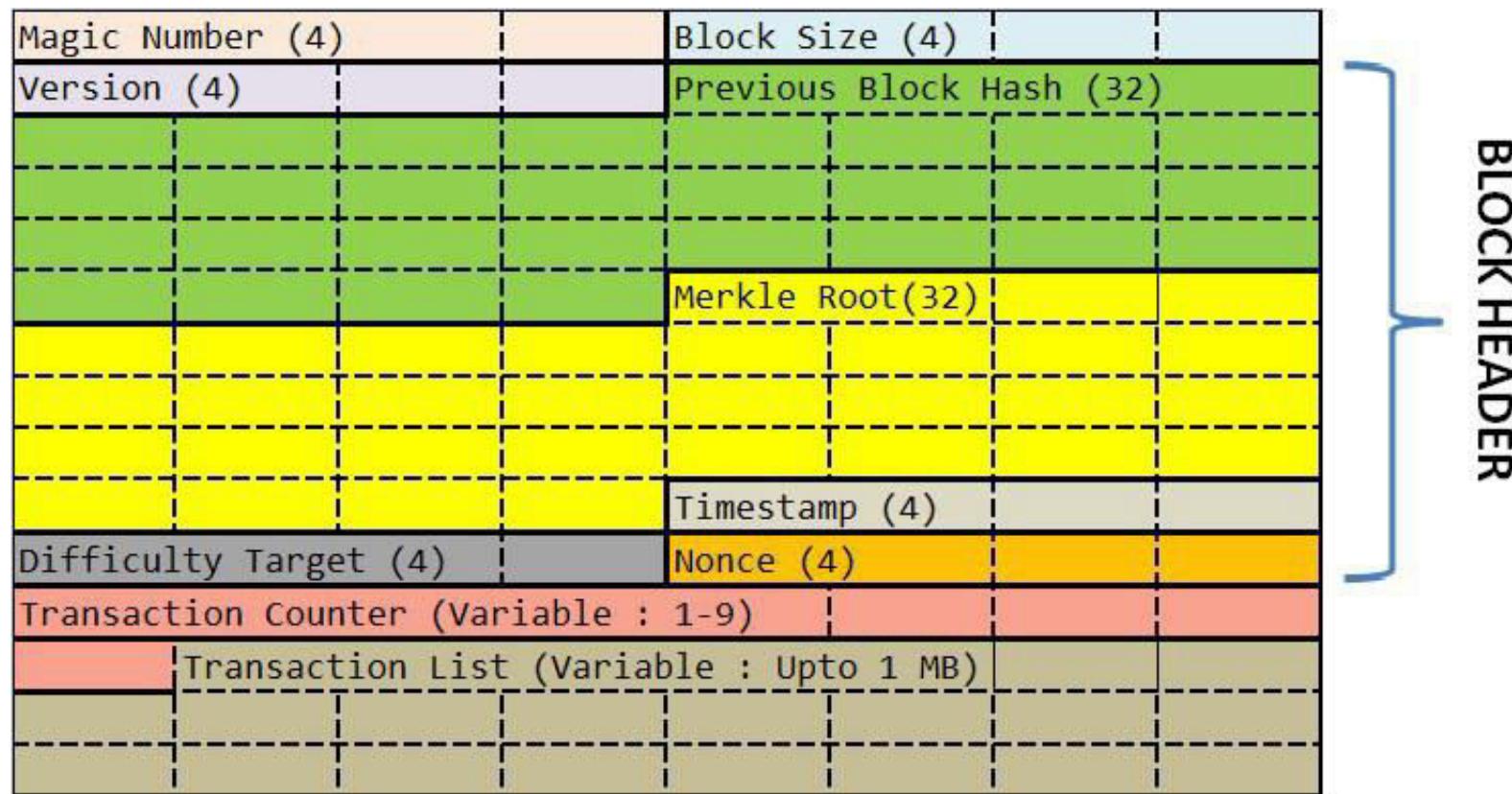


Block Structure

— BLOCK HEIGHT —



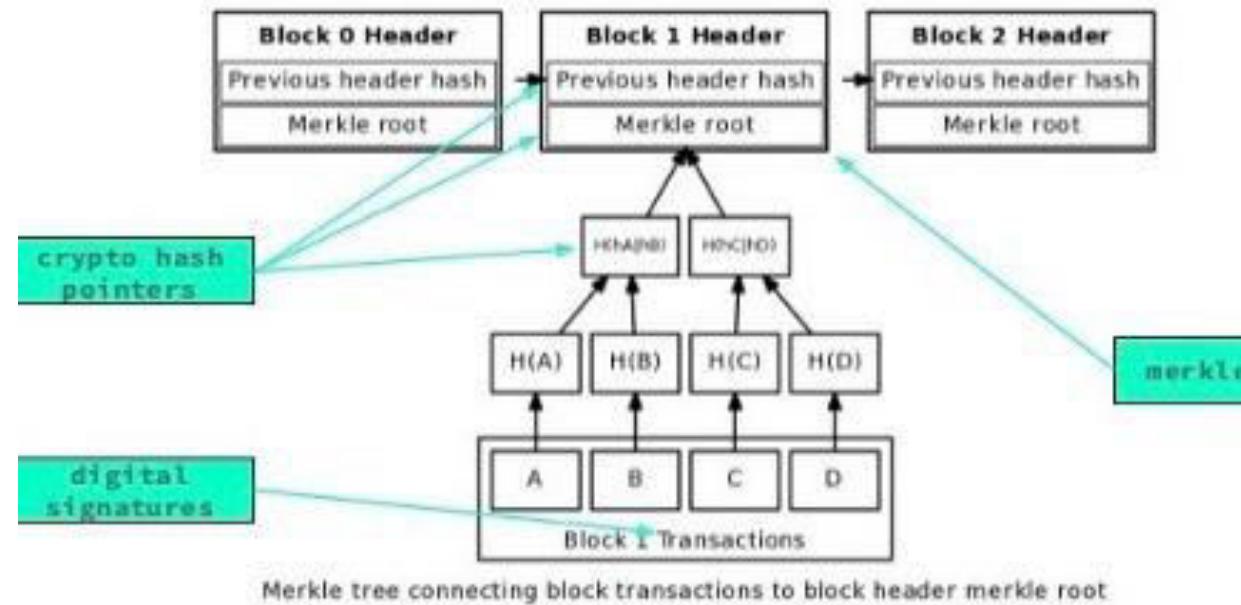
Block Header



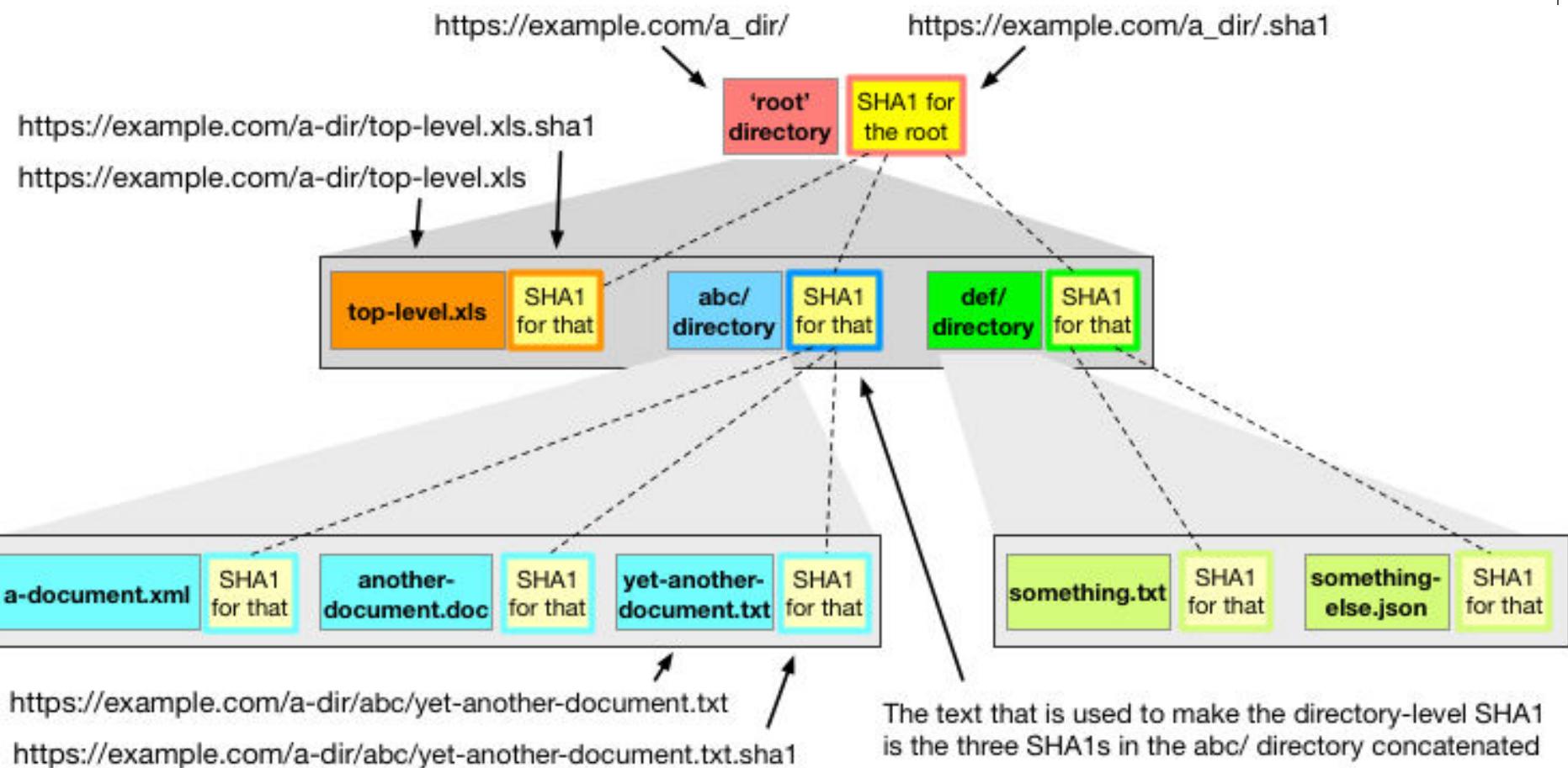
Block-Detail

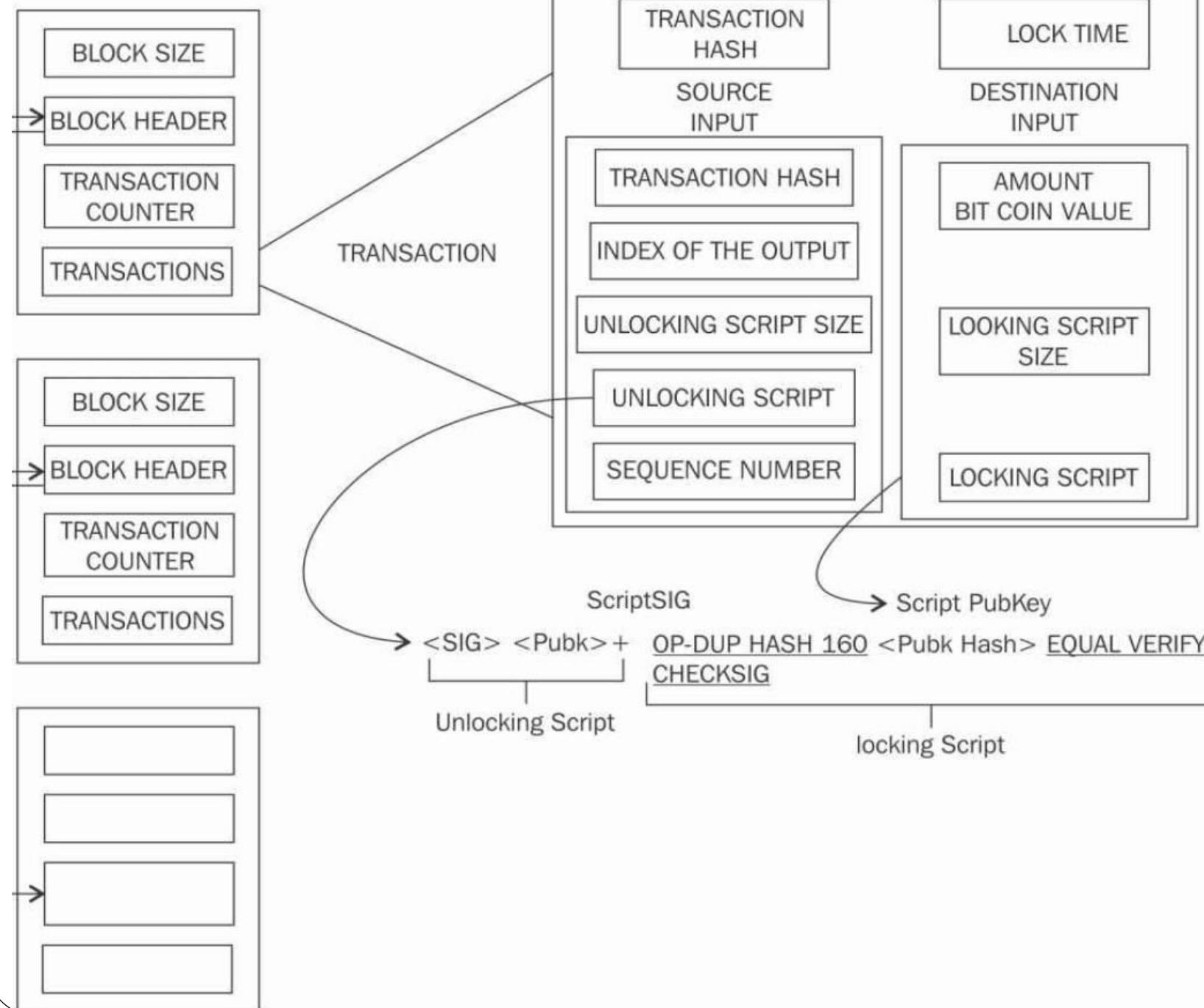
- **Magic number:** an identifier for the Blockchain network, Indicates a)
Start of the block b) Data is from network
- **Block size:** each block is fixed to 1 MB. However will be increased to 2 MB.
The maximum capacity is 2 GB
- **Version:** Each node has to implement
- **Timestamp, Difficulty Target, Nonce:** Time and Input for Hash
- **Hash function: SHA-256, Merkle Tree:** Hash Tree
- Public Key/Private Key & ECDSA
- Nodes/Peers- Peer to Peer Network
- Wallet
- Smart Contract (Optional)
- PoW

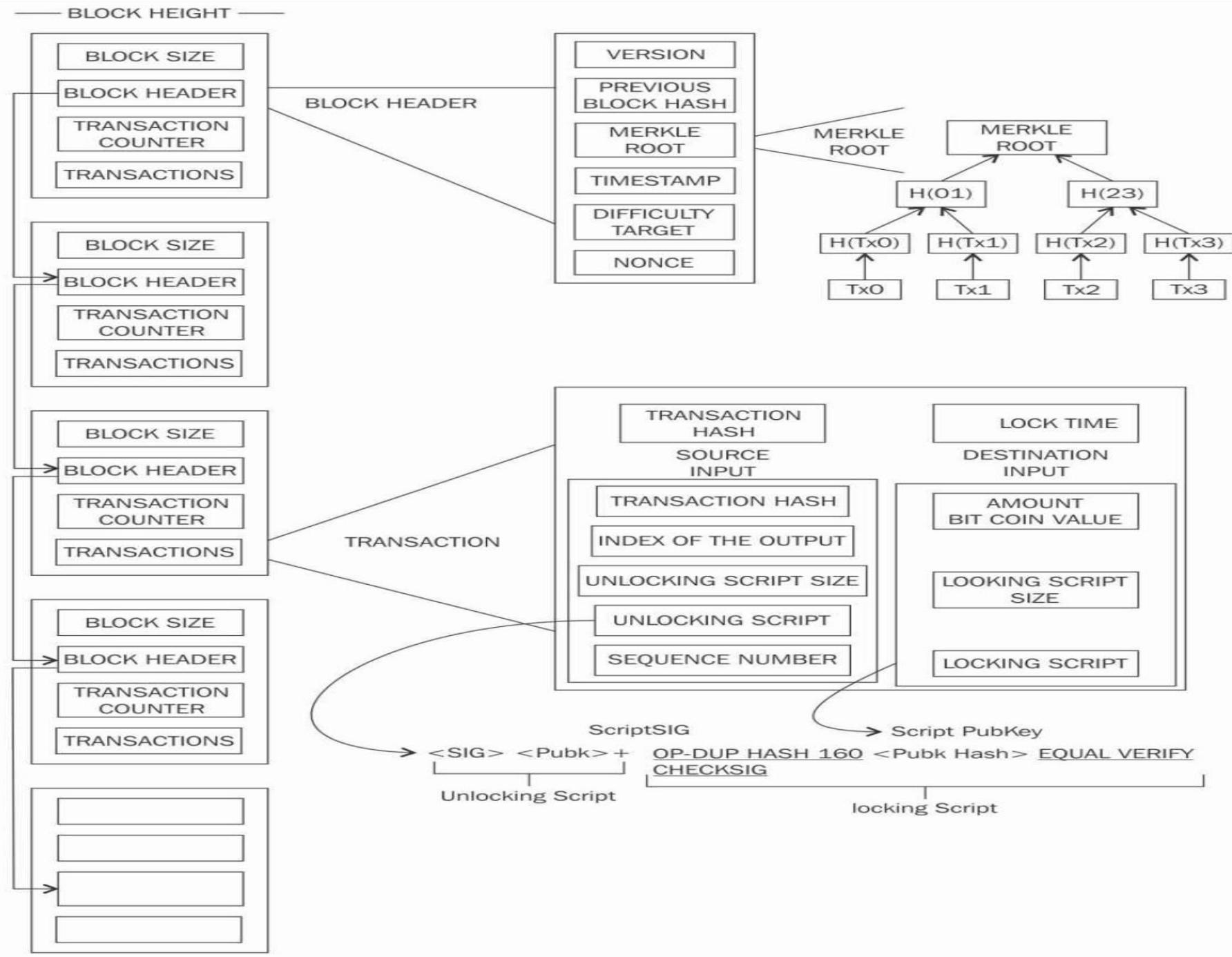
Blockchain Structure



Merkle Tree Structure







- Genesis Block
- This is the first block in the bitcoin blockchain. The genesis block was hardcoded in the bitcoin core software. It is in the chainparams.cpp file.
- <https://github.com/bitcoin/bitcoin/blob/master/src/chainparams.cpp>

```
48     *      CTxOut(nValue=50.00000000, scriptPubKey=0x5F1DF16B2B704C8A578D0B)
49     * vMerkleTree: 4a5e1e
50     */
51 static CBlock CreateGenesisBlock(uint32_t nTime, uint32_t nNonce, uint32_t nBits, int32_t nVersion, const CAmount& genesisReward)
52 {
53     const char* pszTimestamp = "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks";
54     const CScript genesisOutputScript = CScript() << ParseHex("04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649"
55     return CreateGenesisBlock(pszTimestamp, genesisOutputScript, nTime, nNonce, nBits, nVersion, genesisReward);
56 }
57
58 /**
59 * Create a new block with the given timestamp and nonce, and reward the genesis
60 * address with genesisReward.
```

THANK YOU

Blockchain Features

Features of Blockchain

- **Distributed consensus**
- **Transaction verification**
- **Platforms for smart contracts**
- **Transferring value between peers**
- **Generating cryptocurrency**
- **Smart property**
- **Provider of security**
- **Immutability**
- **Uniqueness**
- **Smart contracts**
- **Tamper-Proof**
- **Address Space**--consists of 160-bit address space allows 1.46×10^{48} devices and provides 4.3 billion addresses provides a scalable solution

- **Authentication and Integrity**
- **Authentication, Authorisation and Privacy**
- **Secure Communications**
- **Scalability:** GHOST is a protocol that improves scalability. The Inter Planetary File System (IPFS) was used to store decentralised and shared files enabling peer-to-peer file system
- **Anonymity and Privacy**—The protocols like Dark Wallet, Dash, MixCoin, CoinShuffle, CoinSwap increase security
- **Persistency**
- **Auditability**

Tiers of Blockchain technology

- **Blockchain 1.0:** basically used for cryptocurrencies
- **Blockchain 2.0:** for financial services and contracts
- **Blockchain 3.0:** general-purpose industries such as government, health, media, the arts, and justice.
- **Generation X (Blockchain X):** public blockchain service available that anyone can use just like the Google search engine

Blockchain in the supply chain Management

What is Supply Chain Management?

- The management of the flow of goods, services, and information involving the storage and movement of raw materials, building products as well as full-fledged finished goods from one point to another are known as “supply chain management”.
- Supply chain management includes integrated planning as well as the execution of different processes within the supply chain. These processes include:
 - Material flow
 - Information flow
 - Financial capital flow

Why do we need supply chain management?

- Increase sales
- Decrease frauds and overhead costs
- Improve the quality of improvisation
- Lead to accelerating production and distribution
- Reduce the cost and complexity of the manufacturing process, especially when the process itself is extremely complex.

Problem #1: Hard to Track Down

The lack of transparency in modern supply chains is another major issue. If you have a defective component in your phone, then it is near impossible to pinpoint exactly where that defective piece came from and who was the person(s) responsible for it.

In fact, let's give you an example of how truly dangerous this lack of transparency can be.

On October 6, 2006, multiple states in the US suffered a major E-Coli outbreak. The culprit? Spinach — Around 199 people were affected of whom 22 were children under 5 years old. 31 of the 199 developed a type of kidney failure called hemolytic-uremic syndrome and 3 people died.

Contd...

As a result of this, the entire food industry went into pandemonium. People were desperately trying to trace the source of the infected spinach. Everyone stopped selling spinach immediately from the market. It took the Food and Drug Administration (FDA) a total of 2 weeks to find the source of the contaminated spinach.

Problem #2: Corruption

The problem with running a complex supply chain is that you will need to trust all the participants to do their job. You need to trust them to deliver quality while following standard safety standards. However, human beings are not really that trustworthy and are prone to corruption.

There are **several instances of corruption** that can seep into your traditional supply chains.

- Certain suppliers in your supply chain may get preferential treatment by some of your procurement officers because of a personal relationship.
- Suppose you need a new component/material for a new product. However, that material is a rare asset, and only one supplier can supply it to you. Knowing his monopoly, the supplier can swindle you

Contd...

- One of your procurement officers has a special deal going on with your suppliers. In exchange for a bribe, the supplier can compensate on the quality of their products.
- One of your suppliers is secretly delegating work to sub-suppliers who are not bound by the code of ethics that you have impressed upon your main suppliers.

Problem #3: Costs

There are four main factors that shoot up the cost in traditional supply chains:

- Procurement Costs
- Transportation Costs
- Inventory Costs
- Quality Costs

Procurement Costs

First and foremost, the most obvious cost that you will need to mitigate in your supply chain is the procurement costs i.e. paying for products throughout your supply chain. One thing to keep in mind here, if you have a large company, it won't be possible for you to personally go to the supplier and buy all the components. This is why you will need to hire procurement officers to take care of this for you.

Transportation Costs

When it comes to transportation costs, you will need to make a compromise between speed and expenses. Usually, high-speed transportation choices cost a lot of money. However, going for a financially cheap option can compromise majorly on time, which is almost as important if not more.

Inventory Costs

Inventories are places where you are storing your products. Obviously, that is going to cost a lot of money as well. Now, here is where things become a little messy. Most companies end up borrowing money from banks to pay for the inventories. So, not only do they have to take care of the loan, they have to take care of the interest rates as well.

Quality Costs

You expect any product that you buy to meet a certain level of quality. After all, getting defective products will lead on to problems later on down the chain. However, in order to do these quality checks, you will need to hire properly trained experts. Plus, also keep in mind, that a big chunk of your products will be disposed of at once.

Problem #4: Globalization

Globalization is the process by which your company becomes big enough to have international influence or it starts operating on an international scale. As you can guess, globalization opens up several challenges to supply chain management.

Contd...

So, there are two things that we have learnt so far:

- Supply chains are absolutely critical for the overall well-being of your business.
- The current system of supply chains is outdated and requires a significant reboot.

This is where the **blockchain** comes in.

Contd...

The 3 properties of the blockchain technology that is going to help disrupt the supply chain management system are:

- Decentralization
- Immutable
- Transparency

How Blockchain Technology Will Disrupt the Supply Chain?

Every time a product changes hands, the transaction could be documented in the blockchain, creating a permanent history of a product, from manufacture to sale. What this does, is that it reduces:

- Time Delays
- Human Error
- Added Costs

Blockchain can **definitely improve** the following properties:

- Recording the quantity of the products and its transfer through different parties.
- Tracking all the purchase orders, change orders, receipts, trade-related details
- Verifying the validity of the certification of the products. Eg. this can be used to track whether a particular item meets certain quality standards or not
- It can link various physical items to serial numbers, barcodes, and tags like RFID etc.
- Helps in the sharing all the information about the manufacturing process, assembly, delivery, and maintenance of products with the different parties in the supply chain.

What are the Benefits of Blockchain in Supply Chain Management?

Blockchain technology coupled with the ability to program business logic with the use of smart contracts enables the following:

- Transparency into the provenance of consumer goods— from the source point to end consumption
- Accurate asset tracking
- Enhanced licensing of services, products, and software

How does blockchain technology cut costs from the supply chain infrastructure?

Blockchain has the potential to drive cost-saving efficiencies and to enhance the consumer experience through traceability, transparency, and tradeability.

Blockchain Appliances in Supply Chain Management

Accurate tracking and traceability

Blockchain technology is often used to record product status at each phase of its lifecycle. It helps to track processes starting from the initial stage of production.

The system shows where the meat comes from, processes each step of its journey to a customer, and records everything in the supply chain up to the sale date.

Consensus and permission

Blockchain is often explained as “one version of the truth” for each product. It is a system of records that is aimed to capture proof of money transactions like bills of lading and money transactions.

It covers all stages of the supply chain – from serialization, and shipping to receiving and installation – each is tracked automatically. This system is absolutely built on principles of trust, transparency, and audibility.

Writing smart contracts

A smart contract is a program that utilizes Blockchain to execute the agreement. You can integrate smart contracts into your supply chain management to prevent fraud or other interference.

Establishing trust

Establishing trust between all participants is key to the effective supply chain functioning. In a Blockchain, every participant has a copy of a ledger and knows where each item originated. Everybody has access to information about who has owned it before and when.

Transparent transactions

Now all transactions can be maximally transparent anywhere in the world – no need to use traditional banking. Money transfers can be easily made between a payer and payee within a few minutes. You need not longer wait for days since with a Blockchain-based system, and everything happens much faster and safer.

Monitoring of product conditions

Some kinds of products, like food or medicines, are susceptible and have specific needs. The product storage conditions, such as temperature, humidity, or vibration, can be recorded by sensors and stored on a Blockchain.

RFID Tags

Nowadays, companies actively use RFID tags to store information about products in supply chains. Commonly, they are automatically processed by IT systems and used for smart contracts in logistics.

Here are some industries where blockchain is expected to bring transparency and improve supply chain management.

1. Blockchain and Oil Supply
2. Blockchain and Diamond Supply
3. Blockchain and Food Supply
4. Blockchain and Fashion Supply
5. Blockchain and Wine Supply

Blockchain and Oil Supply

- Abu Dhabi National Oil Company (ADNOC) – the United Arab Emirate's state-owned oil company – has been collaborating with IBM too. Together, they have successfully launched a blockchain supply chain pilot program.
- The program will help to track oil from well to customers, recording transactions every step of the way.
- Even though the program is still in its early stages, ADNOC aims to expand the chain to include customers and investors. This will bring more transparency to its business process. Abu Dhabi's oil company produces about 3 million barrels of oil a day.

Blockchain and Diamond Supply

There is a huge problem in the diamonds industry when it comes to the working conditions under which they are extracted. De Beers, the world's largest diamond producer, wants to put an end to this with the help of a blockchain supply chain program.

Their program, Tracr, successfully tracked 100 diamonds from mine through the cutter and polisher, up to the jeweler. In this program, actors in the blockchain upload photos and information regarding the color, quality and location of a processed diamond. More importantly, introducing Tracr to the diamond industry can prevent illegal actions when mining for diamonds all over the world.

Blockchain and Food Supply

- Interestingly, it seems like every year there is a new epidemic outburst concerning food supplies. Because it takes time to locate the origin of the disease, many retailers need to throw out entire inventories.
- While Walmart and JD.com handle the shipment and production processes, IBM and Tsinghua University are responsible for the research and maintenance of the blockchain. With the progression of the project, other big companies like Nestle, Unilever, and Tyson Foods are expected to jump in.

Blockchain and Fashion Supply

Fashion designer Martine Jarlgaard together with Provenance are campaigning for more transparency in the fashion industry. As a result, in 2017 at a Danish fashion show, they demonstrated their first ever apparel tracked with blockchain.

The aim is for customers to be able to track every aspect of a garment's life. In this way, blockchain will ensure that the purchased clothing is legitimate and produced in factories that do not violate the employees' working conditions.

Blockchain and Wine Supply

- When it comes to the wine industry, China alone sells nearly 30,000 bottles of illegitimate wine every hour. Many of these wines comprise of additives that can be dangerous to consumers' health.
- To solve this, Origintrail and TagItSmart created their blockchain solution. The two companies managed to track more than 15,000 unique wine bottles with the pilot version of their program.
- They aim to stop the production of illegitimate wine with the help of QR codes. In this way, customers will be able to scan the code on the bottle and receive all the information of their purchase.

Blockchain, Challenges and Applications

Dr. Bhawana Rudra
Assistant Professor

Department of Information technology
National Institute of Technology Karnataka

- **Kevin** David Mitnick arrested in 1995 and five years in prison for various computer and communications-related crimes. Now a American computer security consultant, author, and convicted **hacker**.
- Vladimir Leonidovitch Levin is a Russian individual famed for his involvement in the attempt to fraudulently transfer USD 10.7 million via Citibank's computers
- **Gary** McKinnon a **Scottish** systems administrator and **hacker** who was accused in 2002 of perpetrating the biggest military computer **hack** of all time.

History

- Since the 1980s, **e-cash** protocols have existed that are based on a model proposed by *David Chaum*.
- *The issues like accountability and anonymity was solved using blind signature and secret sharing which was faced in e-cash.*
- Chaum, Fiat, and Naor (CFN), **e-cash** schemes that introduced anonymity and double spending detection and named it as Brand e-cash
- **hashcash** was introduced by *Adam Back in 1997 as a PoW system to control e-mail spam.*

History....

- *Hashcash was based on the usage of computing hash functions as PoW.*
- *Wei Dai has introduced **b-money** in 1998 creates money based on solving computational puzzles such as hash cash.*
- *Nick Szabo introduced **BitGold** in 2005 based on solving computational puzzles to mint digital currency*
- In the same year, *Hal Finney introduced the concept of*
- **cryptographic currency** by combining ideas from b-money and hashcash puzzles.
- In 2009 the first practical implementation of cryptocurrency named **bitcoin** was introduced;

- https://www.youtube.com/watch?v=II_Jogdmo88
- <https://www.binance.com/en>
- <http://vlabs.iitb.ac.in/vlabsdev/labs/blockchain/labs/blockers-intro-blockchain-psit/simulation.html>

Blockchain

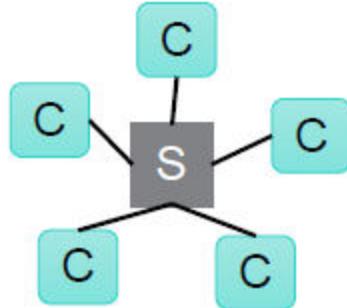
- Transaction systems need to be decentralized, transparent and incorruptible
- Digital currency allows for instantaneous transactions and border-less transfer of ownerships
- Bitcoin: On October 31, 2008, *Satoshi Nakamoto*

A purely peer to peer electronic cash/digital asset transfer system

- First popular implementation of Blockchain and birthing today's Blockchain industry
- Since then, additional Blockchains have been popularized, Ethereum, various Hyperledger project solutions etc

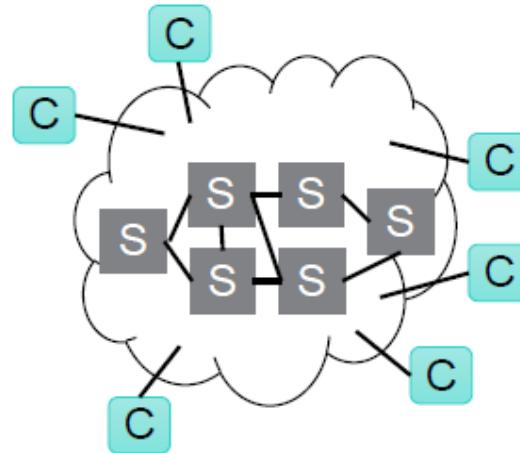
Centralized

- E.g. Simple Web servers, database servers, etc
- Easy to deploy and maintain
- Full control
- Hard to scale
- Single point-of-failure



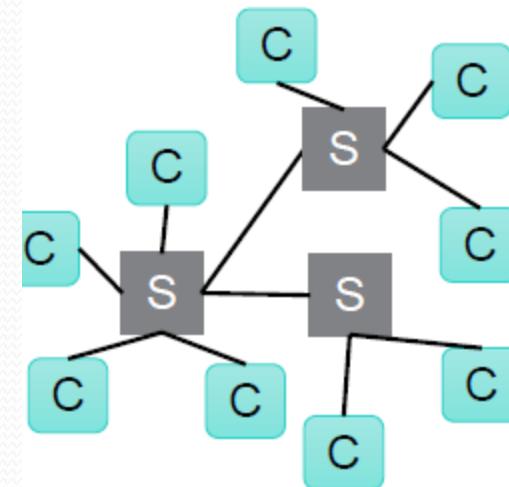
Distributed

- E.g. DNS, AWS, GCP, etc
- System behavior is defined by multiple entities working together
 - Usually well defined roles for participating nodes
- Scales easily
- May have a single point of failure



Decentralized

- Eg: Blockchain, p2p networks
- Multiple masters, need for consensus
 - Moderately tricky to maintain
- Multiple point of control
- Scales easily
- No single point of failure



- *Distributed database that is practically immutable, maintained by decentralized P2P network using consensus mechanism, cryptography and back referencing blocks to order and validate the transactions*

Five common blockchain myths create misconceptions about the advantages and limitations of the technology.

	Myth	Reality
1	 Blockchain is Bitcoin	<ul style="list-style-type: none">● Bitcoin is just one cryptocurrency application of blockchain● Blockchain technology can be used and configured for many other applications
2	 Blockchain is better than traditional databases	<ul style="list-style-type: none">● Blockchain's advantages come with significant technical trade-offs that mean traditional databases often still perform better● Blockchain is particularly valuable in low-trust environments where participants can't trade directly or lack an intermediary
3	 Blockchain is immutable or tamper-proof	<ul style="list-style-type: none">● Blockchain data structure is append only, so data can't be removed● Blockchain could be tampered with if >50% of the network-computing power is controlled and all previous transactions are rewritten—which is largely impractical
4	 Blockchain is 100% secure	<ul style="list-style-type: none">● Blockchain uses immutable data structures, such as protected cryptography● Overall blockchain system security depends on the adjacent applications—which have been attacked and breached
5	 Blockchain is a "truth machine"	<ul style="list-style-type: none">● Blockchain can verify all transactions and data entirely contained on and native to blockchain (eg, Bitcoin)● Blockchain cannot assess whether an external input is accurate or "truthful"—this applies to all off-chain assets and data digitally represented on blockchain

Financial Ledger : Good old method

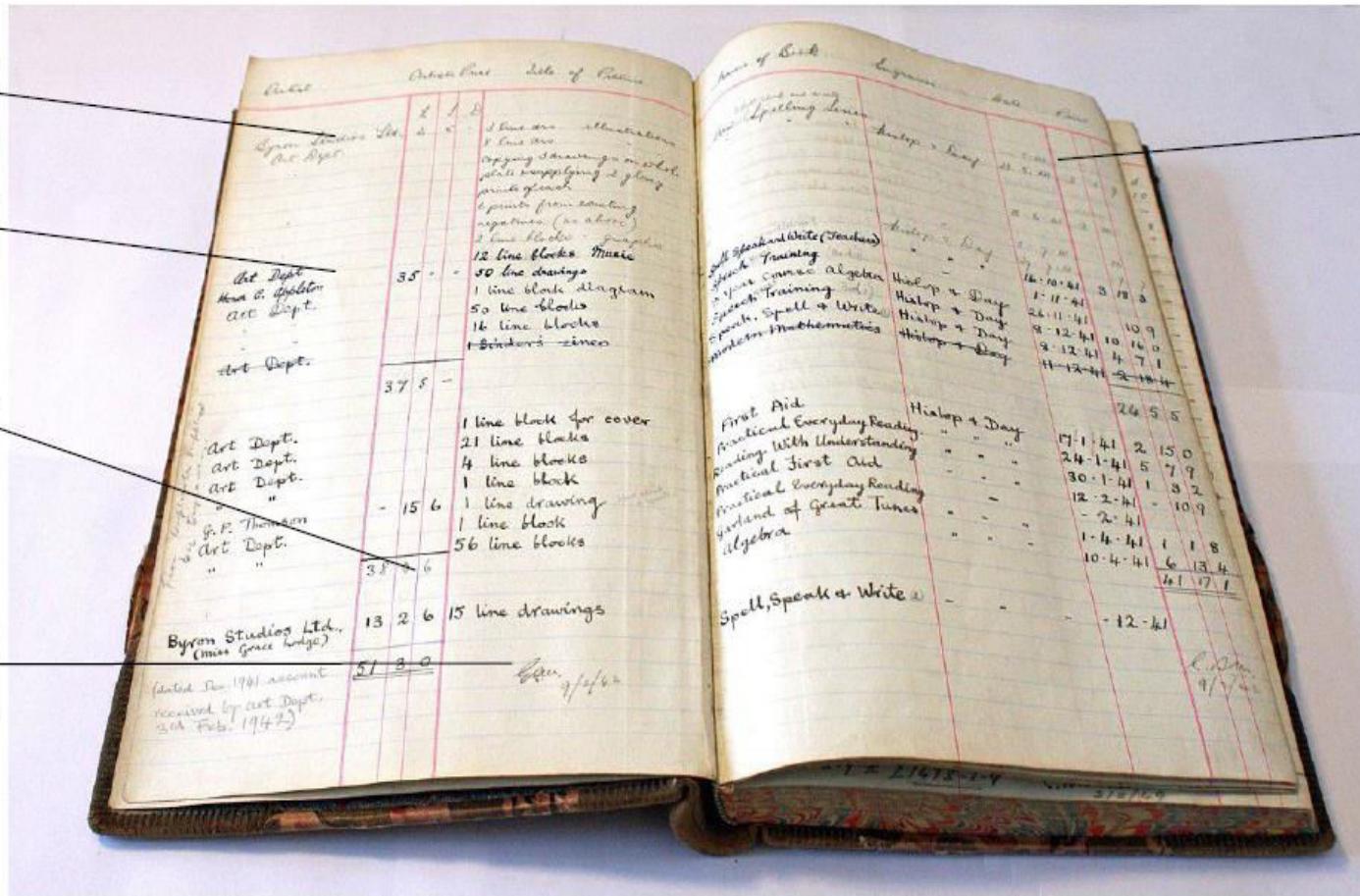
Immutable

Ordering

Consistency

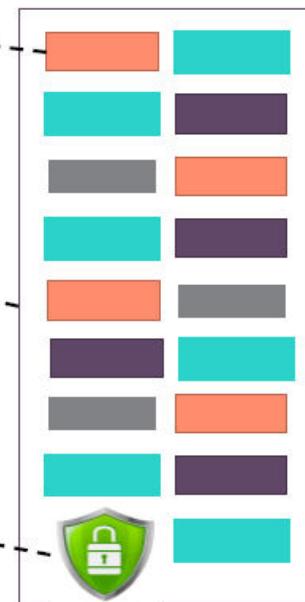
Accountability

Consistency
from Previous
Page



What is Blockchain

Individual Transaction record

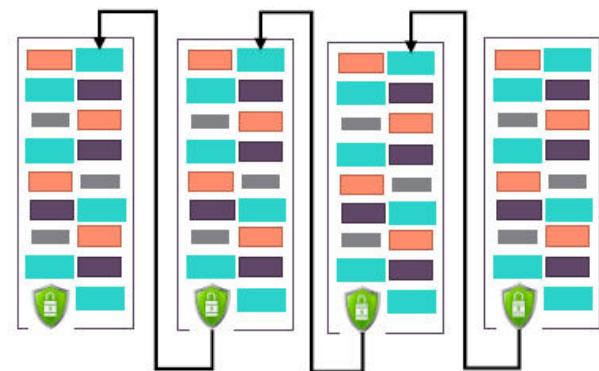


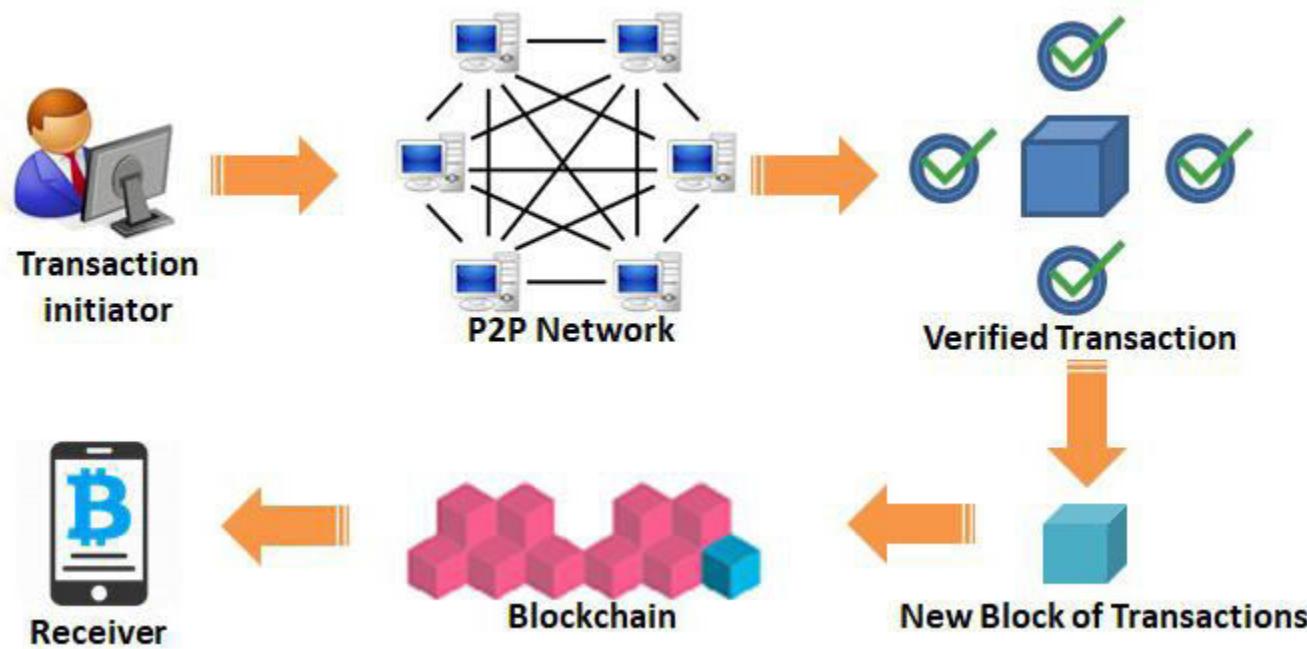
Block:

A collection of transactions that
are all approved at once.

Security and Tamper proofing
using Cryptography

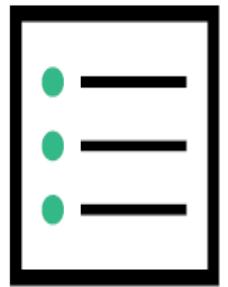
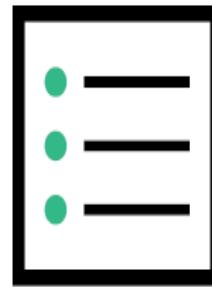
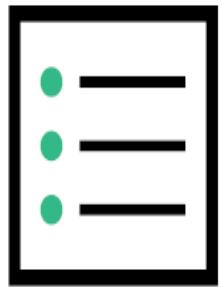
Block Chain.





Blockchain is

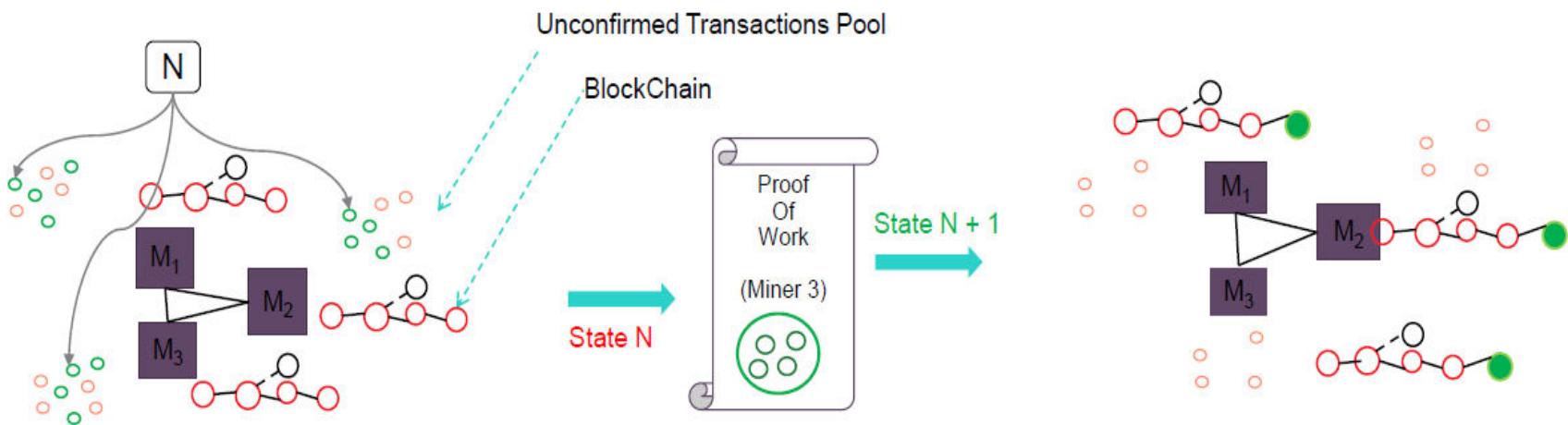
- Typically a **replicated state machine**
- Geographically distributed without any central entity – called **decentralization**
- Multiple copies of the same code – called **smart contract**
- Multiple copies of the same variable values – called **state**
- Dynamic population of nodes
 - Nodes can come and go without constraints
- Dynamic Topology
 - Network partitioning is assumed



- **A transaction network**
 - Decentralized peer-to-peer architecture
 - Nodes consisting of market participants
- **Shared ledger**
 - Members transact through a shared ledger always kept in sync.
 - All transactions are replicated so each participant has their own copy of ledger
 - Ledgers can be public (Consumer) or private (Business).

- **Consensus-based validation**
- •Members validate and commit transactions in order to reach consensus
- •Consensus lowers the risk of fraudulent transactions (tampering would have to occur across many places at the exact same time)

A day in the life of a Blockchain Transaction



1 Nodes Generate Transactions.



2 Nodes broadcasts Transactions to network.

3 Miner Nodes adds Transactions unconfirmed pool.

4 Miners create block and starts work on proof of work.
In this case all Green colored pebbles

5 First miner to generate proof of work wins

6 Broadcast Block and proof of work to other nodes

7 All nodes add the block to their chain

Other miners removing transactions from unconfirmed pool

Block Structure

Magic Number (4)	Block Size (4)
Version (4)	Previous Block Hash (32)
Merkle Root(32)	
Timestamp (4)	
Difficulty Target (4)	Nonce (4)
Transaction Counter (Variable : 1-9)	
Transaction List (Variable : Upto 1 MB)	

BLOCK HEADER

- **Magic number: an identifier for the Blockchain network,**
- Indicates a) Start of the block b) Data is from network
- **Block size: each block is fixed to 1 MB. However will be increased to 2 MB. The maximum capacity is 2 GB**
- Version: Each node has to implement
- **Timestamp, Difficulty Target,Nonce: Time and Input for Hash**
- **Hash function: SHA-256, Merkle Tree: Hash Tree**
- Public Key/Private Key & ECDSA
- Nodes/Peers- Peer to Peer Network
- Wallet
- Smart Contract (Optional)
- PoW

Cryptography use

*Initiation and Broadcasting
of Transaction*

- Digital Signatures
- Private/Public Keys

Validation of Transaction

- Proof of Work and certain alternatives

Chaining Blocks

- Hash Function

Proof of Work

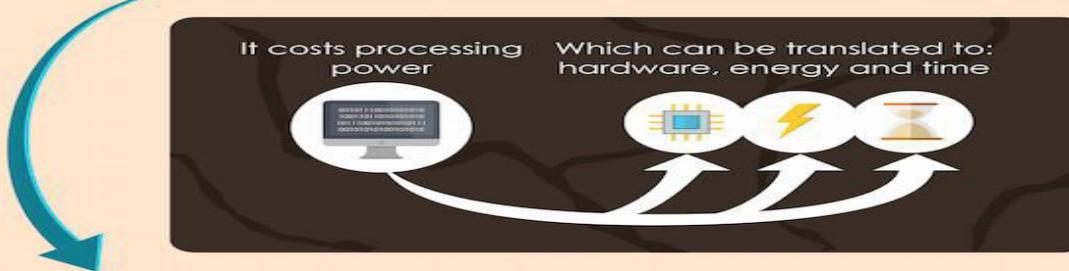
- Producing a proof of work can be a random process with low probability, so that a lot of trial and error is required on average before a valid proof of work is generated
- The most widely used proof-of-work scheme is SHA-256
- Some other hashing algorithms that are used for proof-of-work include scrypt, Blake-256, CryptoNight, HEFTY1, Quark, SHA-3, scrypt-jane, scrypt-n and combinations

THE BITCOIN MINING SAGA - PART II

By Patrícia Estevão

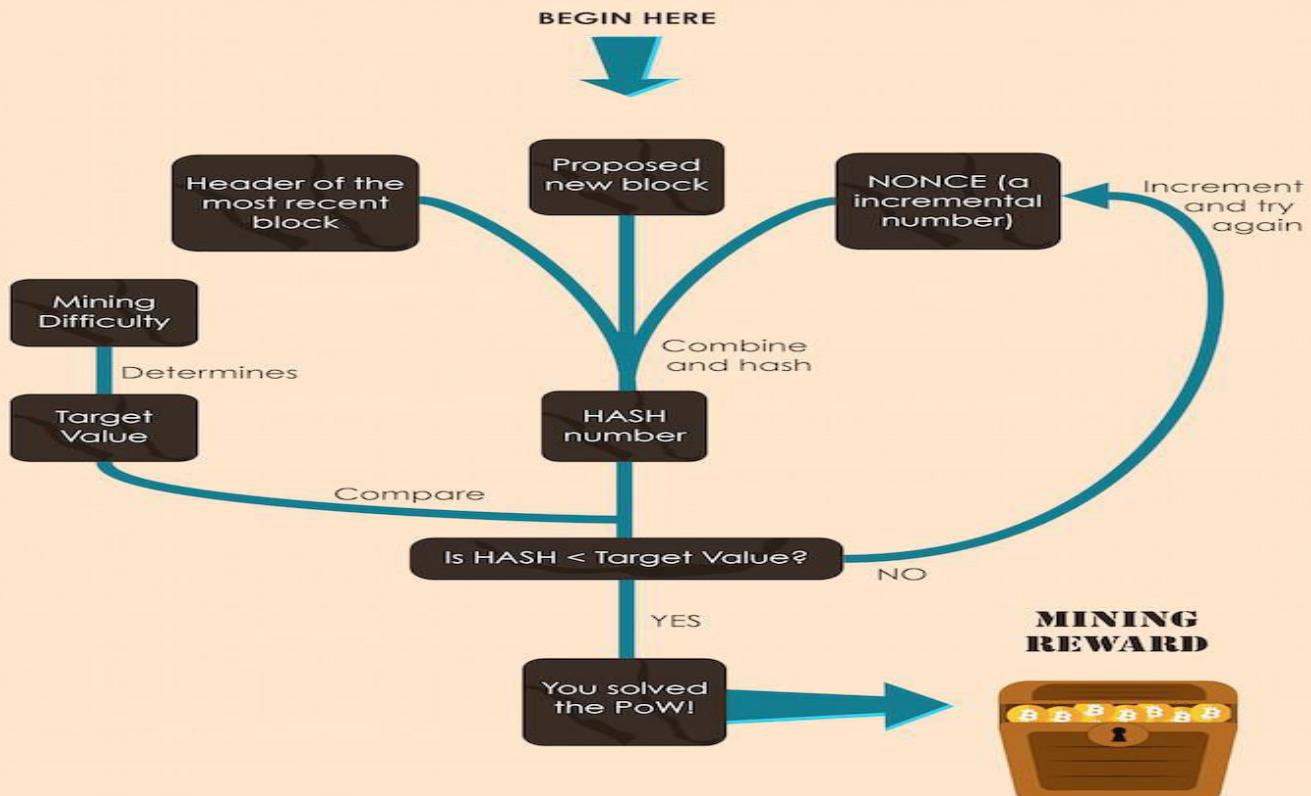
What is Proof of Work (PoW)?

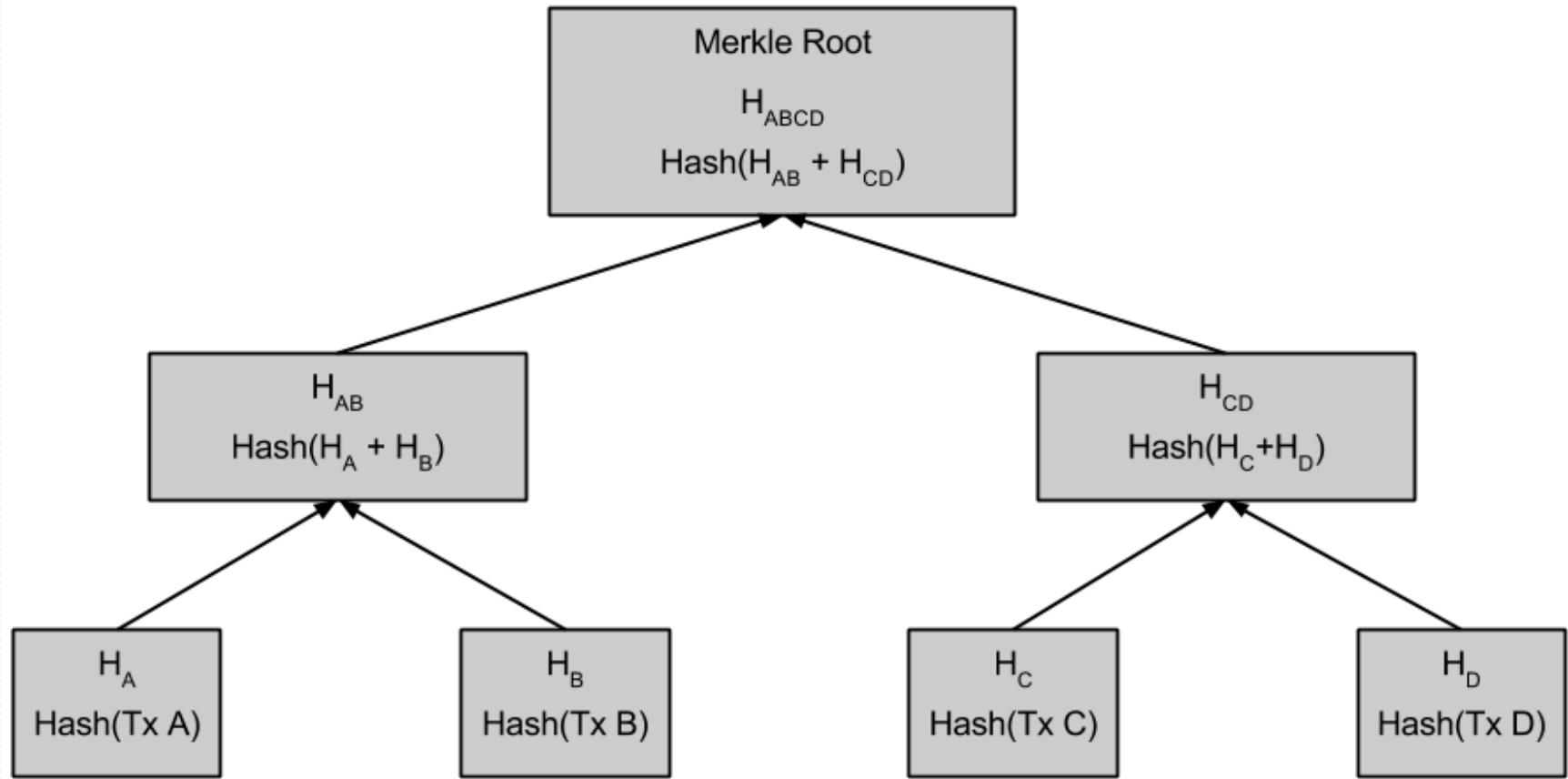
It's a method to ensure that the information (the new block) was difficult (costly, time-consuming) to be made.

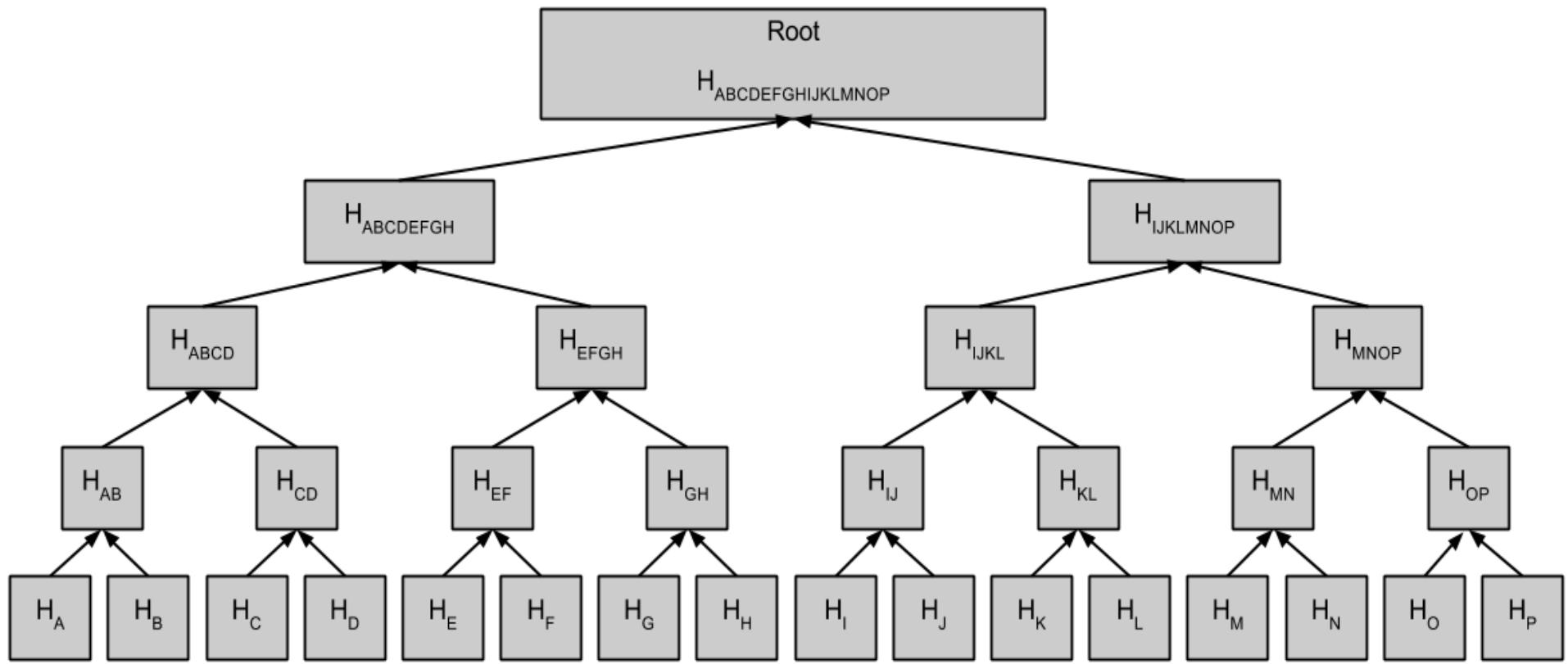


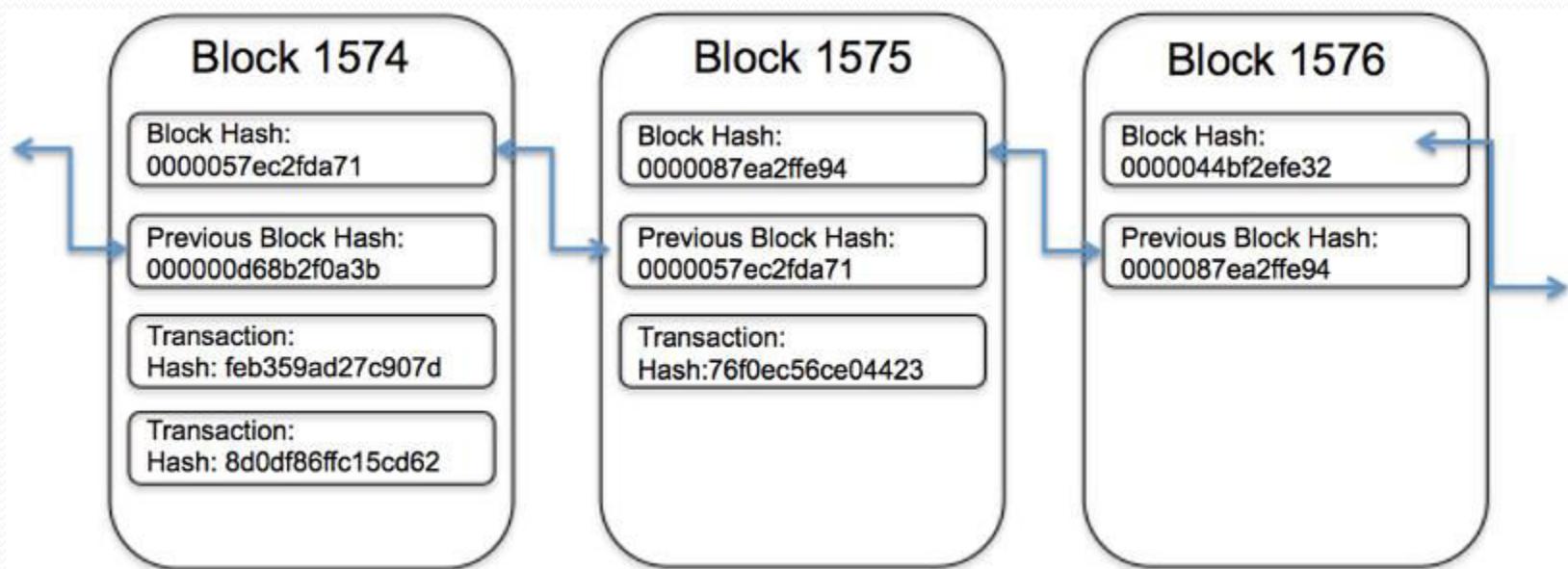
It's easy, on the other hand, for others to check if the requirements were met.

IN PRACTICE (made simple)









Applicable for business

Shared Ledger

- Append-only distributed system of record shared across business network

Permissions

- Ensuring appropriate visibility, transactions are secure, authenticated, and verifiable

Smart Contract

- Business terms embedded in transaction database and executed with transactions

Consensus

- All parties agree to network verified transaction

Tiers of blockchain technology

- **Blockchain 1.0**
 - This was introduced with the invention of bitcoin and is basically used for cryptocurrencies
- **Blockchain 2.0**
 - Generation 2.0 blockchains are used by financial services and contracts are introduced in this generation.
- **Blockchain 3.0**
 - Generation 3 blockchains are used to implement applications beyond the financial services industry and are used in more general-purpose industries such as government, health, media, the arts, and justice.
- **Generation X (Blockchain X)**
 - This is a vision of blockchain singularity where one day we will have a public blockchain service available that anyone can use just like the Google search engine

Types of Blockchain

- **Public Blockchain**
- **Private Blockchain**
- **Semi-private Blockchain**
- **Sidechains**
 - More precisely known as pegged side chains, this is a concept whereby coins can be moved from one blockchain to another and moved back. Common uses include the creation of new altcoins (alternative crypto currencies) whereby coins are *burnt as a proof of adequate stake*. *There are two types of side chain*. The example provided above for *burning coins is applicable to a one-way pegged side chain*. The second type is called a two-way pegged side chain, which allows the movement of coins from the main chain to the side chain and back to the main chain when required.

- **Permissioned ledger**
 - A permissioned ledger is a blockchain whereby the participants of the network are known and already trusted
- **Distributed ledger**
 - As the name suggests, this ledger is distributed among its participants and spread across multiple sites or organizations. It can be Public or Private.
- **Shared ledger**
 - This is generic term that is used to describe any application or database that is shared by the public or a consortium

- **Fully private and proprietary blockchains**
 - These blockchains perhaps have no mainstream application as they deviate from the core idea of decentralization in blockchain technology
- **Tokenized blockchains**
 - These blockchains are standard blockchains that generate cryptocurrency as a result of a consensus process via mining or via initial distribution.
- **Tokenless blockchains**
 - These are probably not real blockchains because they lack the basic unit of transfer of value but are still valuable in situations where there is no need to transfer value between nodes and only sharing some data among various already trusted parties is required.

Platforms

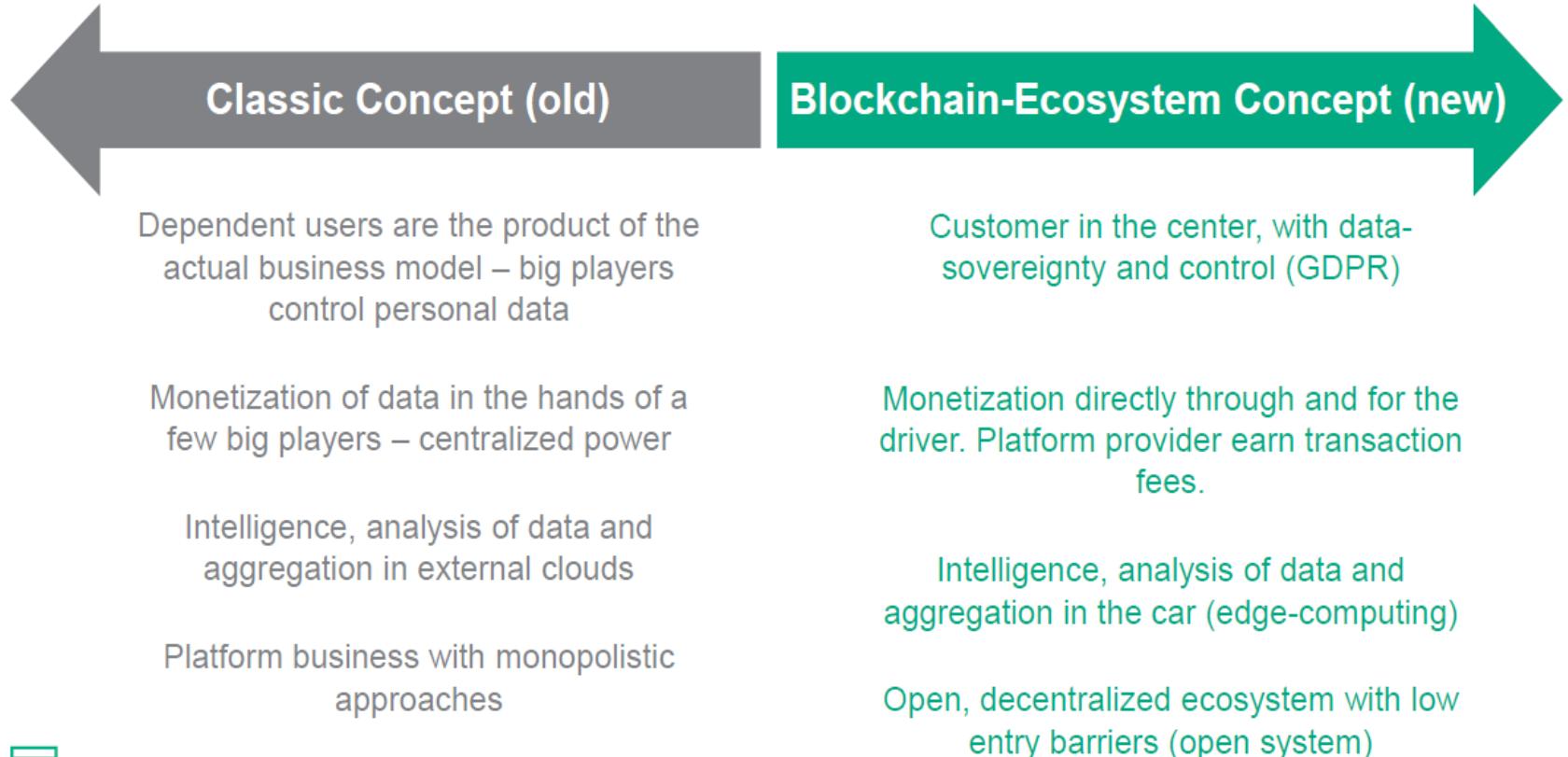
Ethereum

Hyperledger Fabric

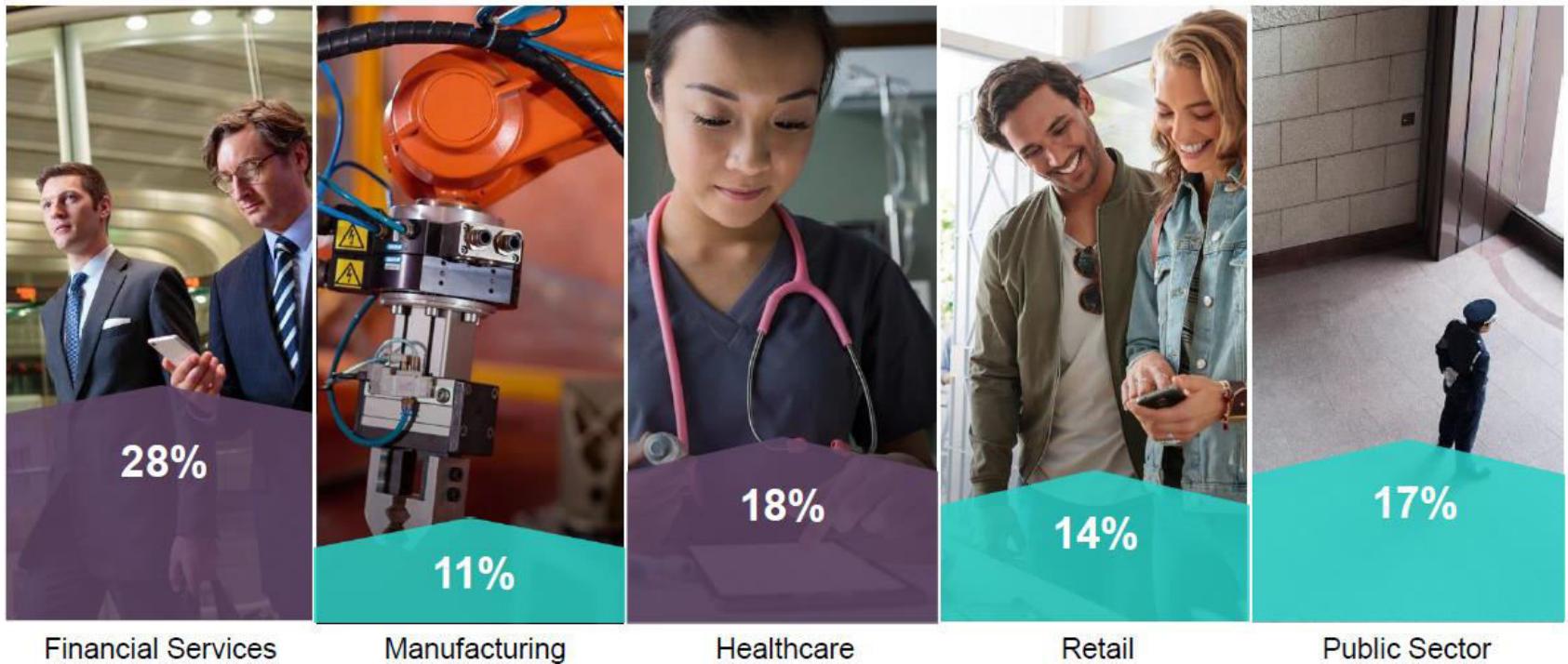
R3 Corda

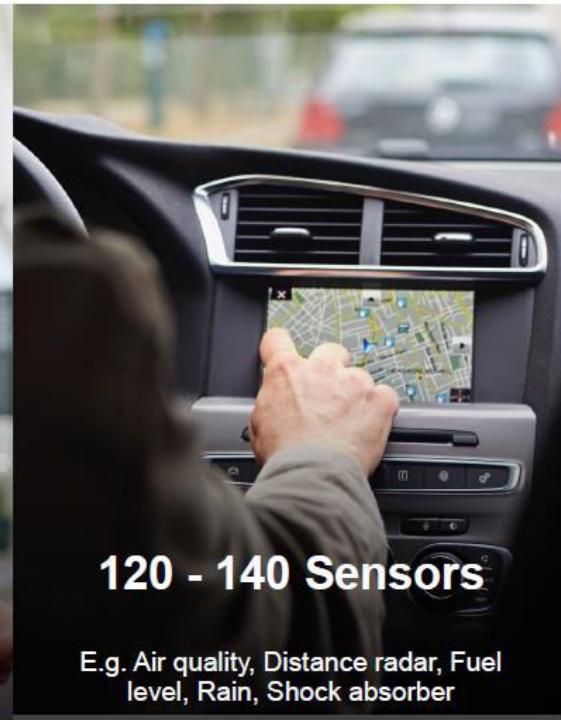
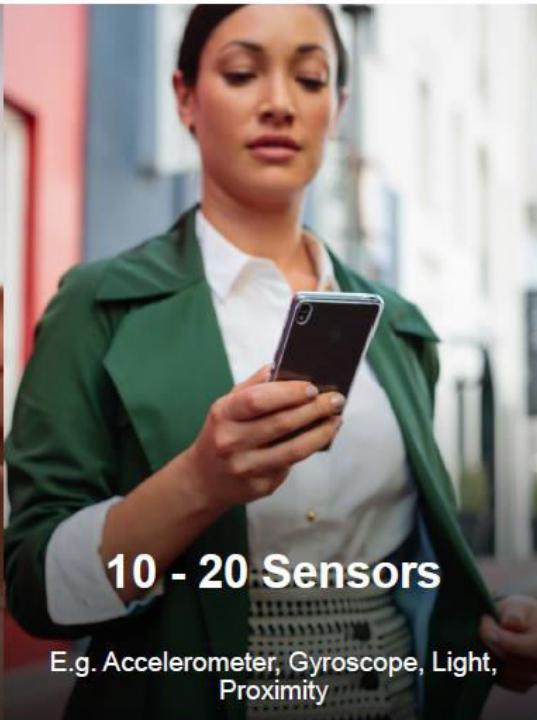
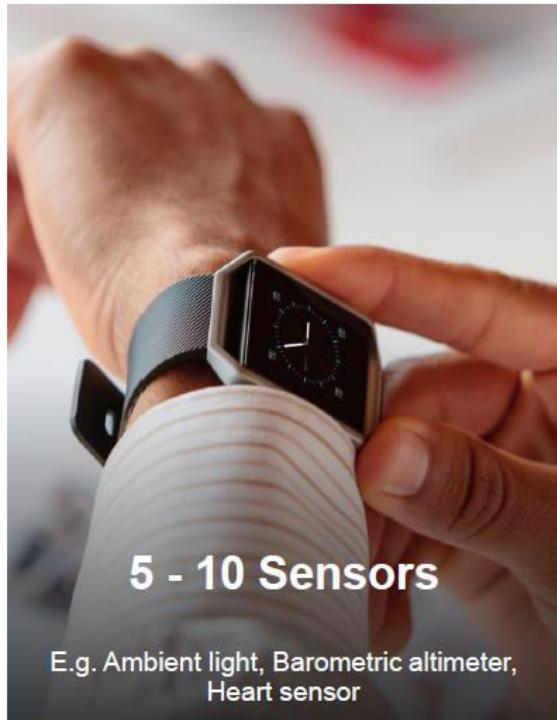
Others ...

It is time to rethink and a new approach

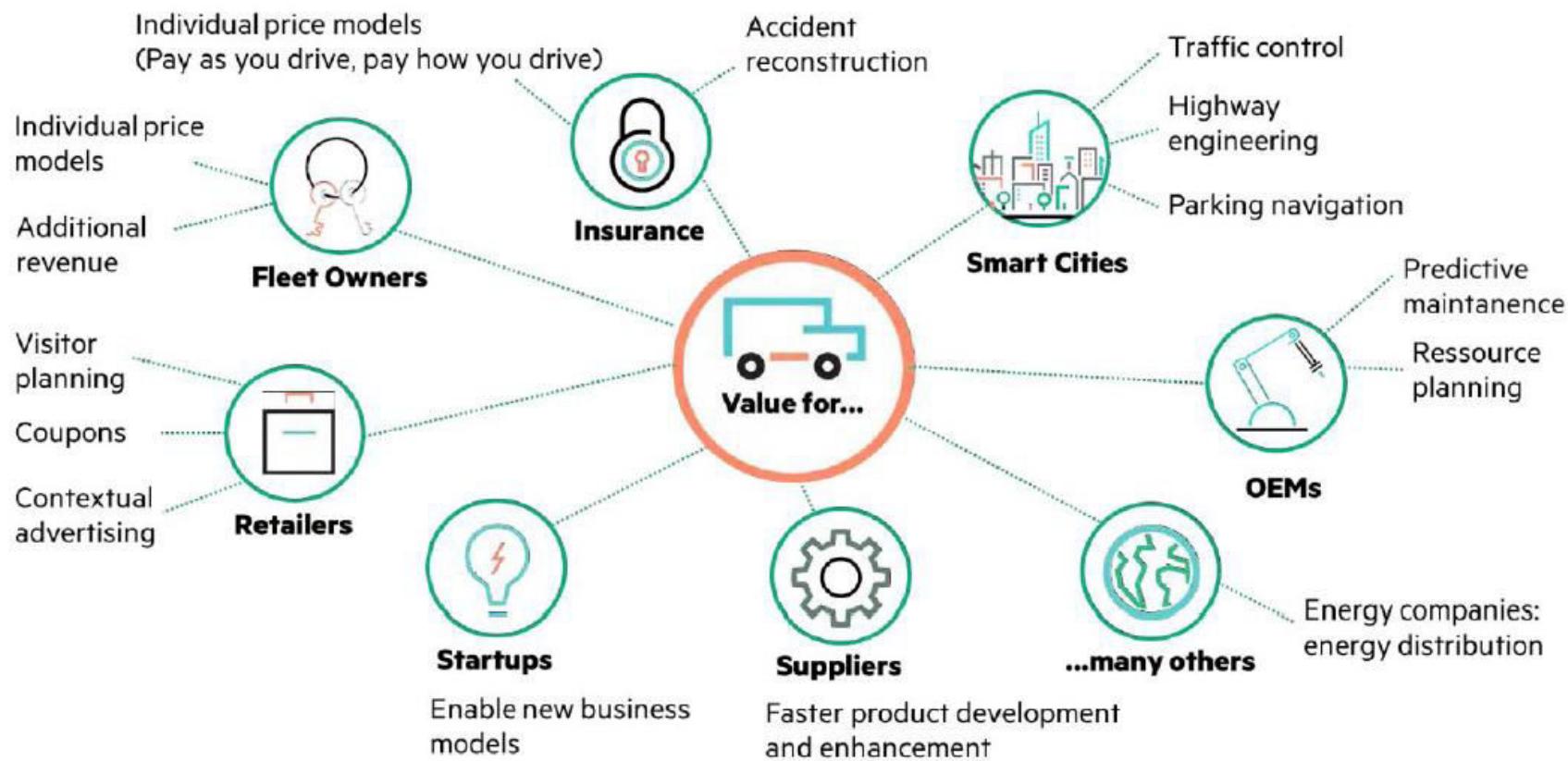


Blockchain in Industries





Vehicle data monetization use cases



Benefits and limitations of blockchain

- **Decentralization**
 - This is a core concept and benefit of blockchain
- **Transparency and trust**
 - As blockchains are shared and everyone can see what is on the blockchain, this allows the system to be transparent and as a result trust is established
- **Immutability**
 - Once the data has been written to the blockchain, it is extremely difficult to change it back.

- **High availability**
 - As the system is based on thousands of nodes in a peer-to-peer network, and the data is replicated and updated on each and every node, the system becomes highly available
- **Highly secure**
 - All transactions on a blockchain are cryptographically secured and provide integrity.
- **Simplification of current paradigms**
 - The current model in many industries such as finance or health is rather disorganized, wherein multiple entities maintain their own databases and data sharing can become very difficult

- **Faster dealings**
 - In the financial industry, especially in post-trade settlement functions, blockchain can play a vital role by allowing the quicker settlement of trades as it does not require a lengthy process of verification
- **Cost saving**
 - As no third party or clearing houses are required in the blockchain model by eliminating the overload

Challenges and limitations of blockchain technology

- As with any technology there are challenges that need to be addressed in order to make a system more robust, useful, and accessible. Blockchain technology is no exception; in fact a lot of effort is being made in Academia and Industry to overcome the challenges posed by blockchain technology. A selection of the most sensitive challenges are presented as follows:
 - Scalability
 - Adaptability
 - Regulation
 - Relatively immature technology
 - Privacy

- **Throughput**

- Bitcoin: 7 TPS
- Ethereum: 15 TPS
- Visa: 1700 TPS

- **Serial Execution**

- To ensure there are no conflicts

- **Restricted features**

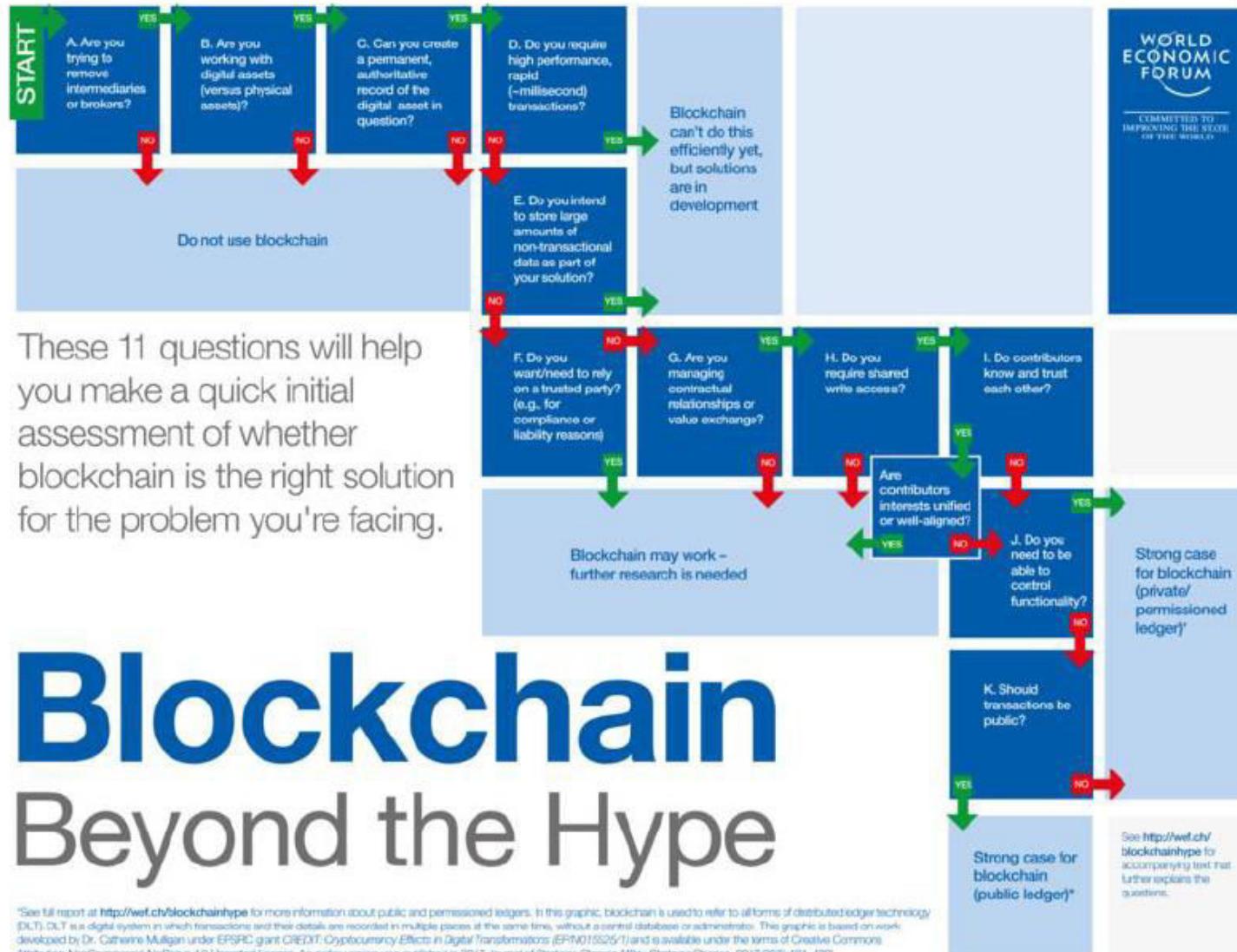
- Random number generator, current time

- **Entire block history needs to be maintained**

- Every new node runs through all the blocks ever produced

- **When can a transaction be considered committed?**

When to use Blockchain



Blockchain Beyond the Hype

Blockchain-types

Dr. Bhawana Rudra

NITK



Disclaimer: The images and Information has been used from various resources of the web.

Types of blockchain

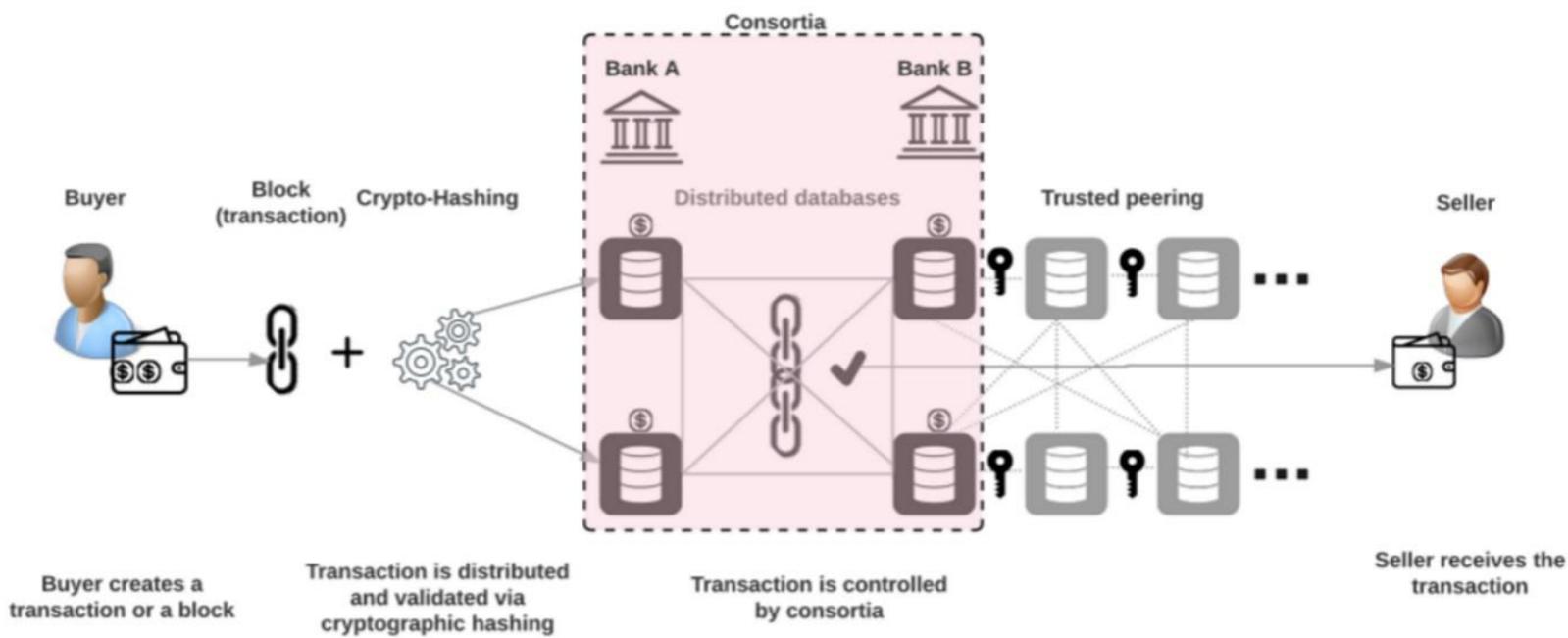
- Public blockchains
- Private blockchains
- Semi-private blockchains
- Sidechains
- Permissioned ledger
- Distributed ledger
- Shared ledger
- Fully private and proprietary blockchains
- Tokenized blockchains
- Tokenless blockchains

- Public Blockchain

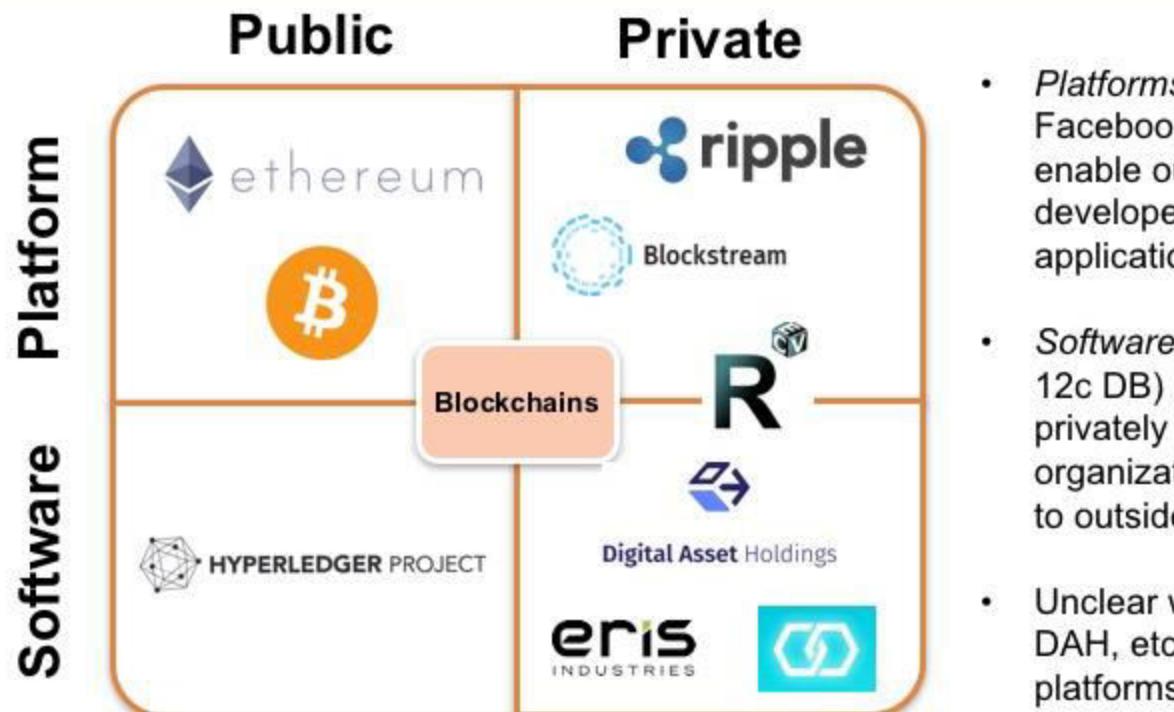


● Private Blockchain

Private Blockchain



Blockchains Can Be Further Distinguished Between 'Platform' and 'Software' Providers



- *Platforms* (ie Facebook, iOS) enable outside developers to build applications on top
- *Software* (eg Oracle 12c DB) is often run privately inside an organization, not open to outside developers
- Unclear whether R3, DAH, etc will become platforms

Sources: Chain, [Chris Skinner's blog](#)

● Semi Private Blockchain

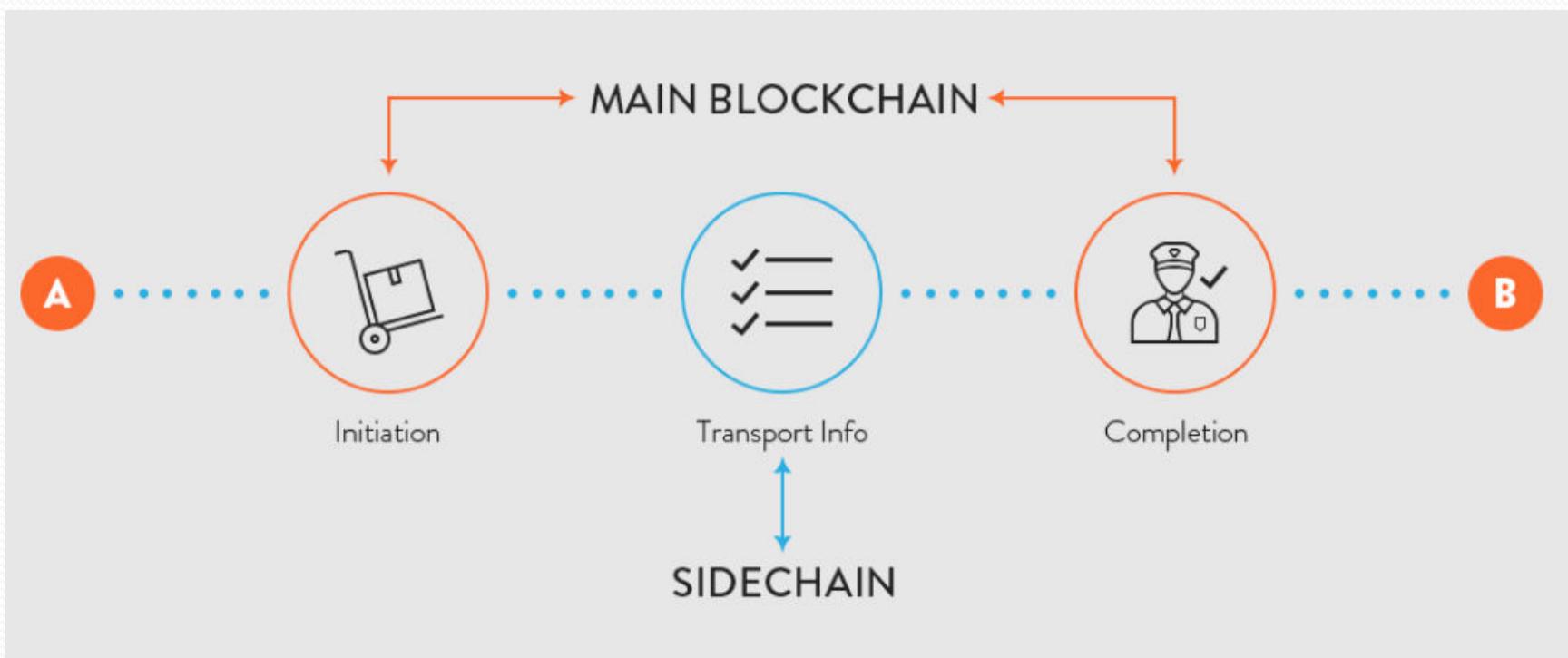
What is Hybrid Blockchain?

The hybrid blockchain is best defined as the blockchain that attempts to use the best part of both private and public blockchain solutions.

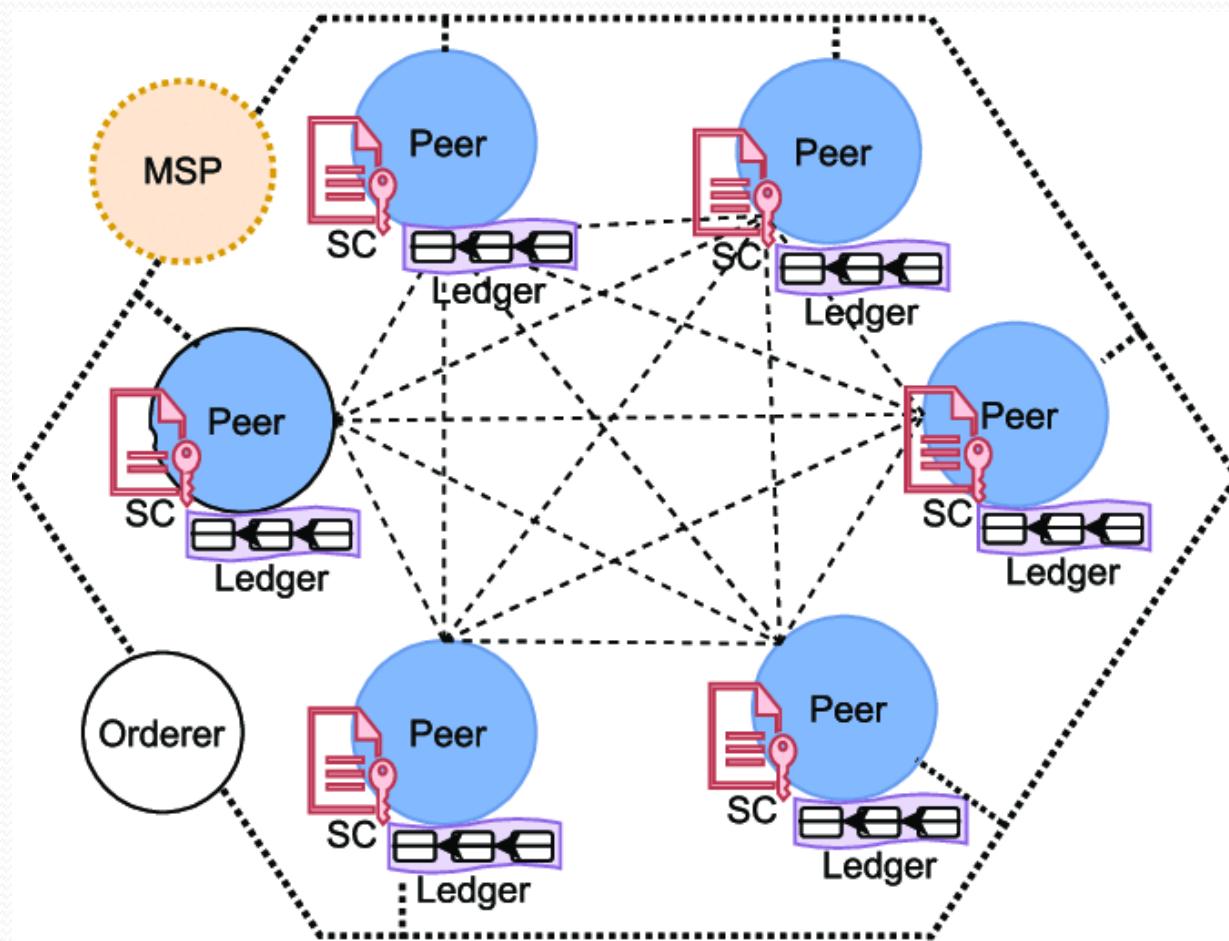
In an ideal world, a hybrid blockchain will mean controlled access and freedom at the same time. In simple terms, some processes are kept private and others public.



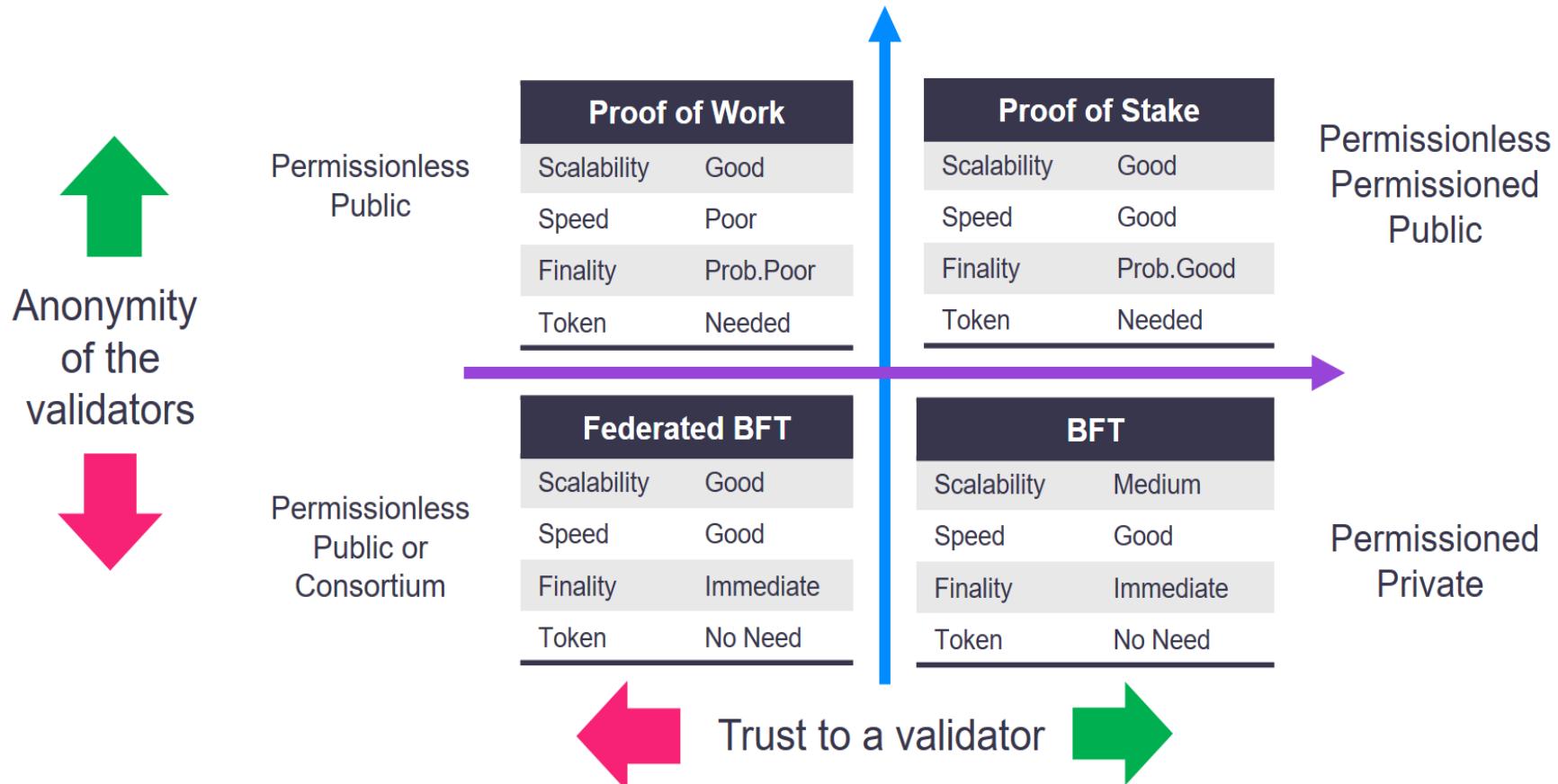
- Sidechain



- Permissioned Ledger



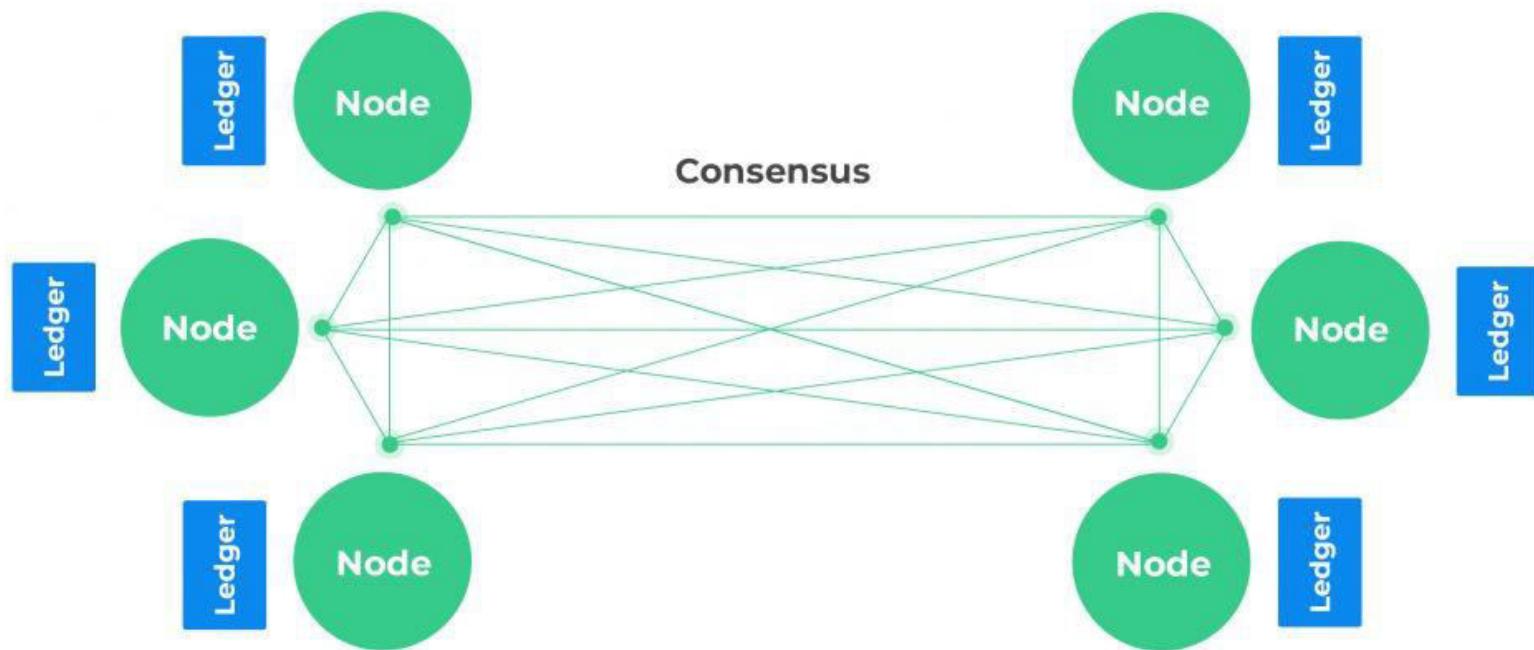
Public, Private and Permissioned Blockchains



- Distributed ledger

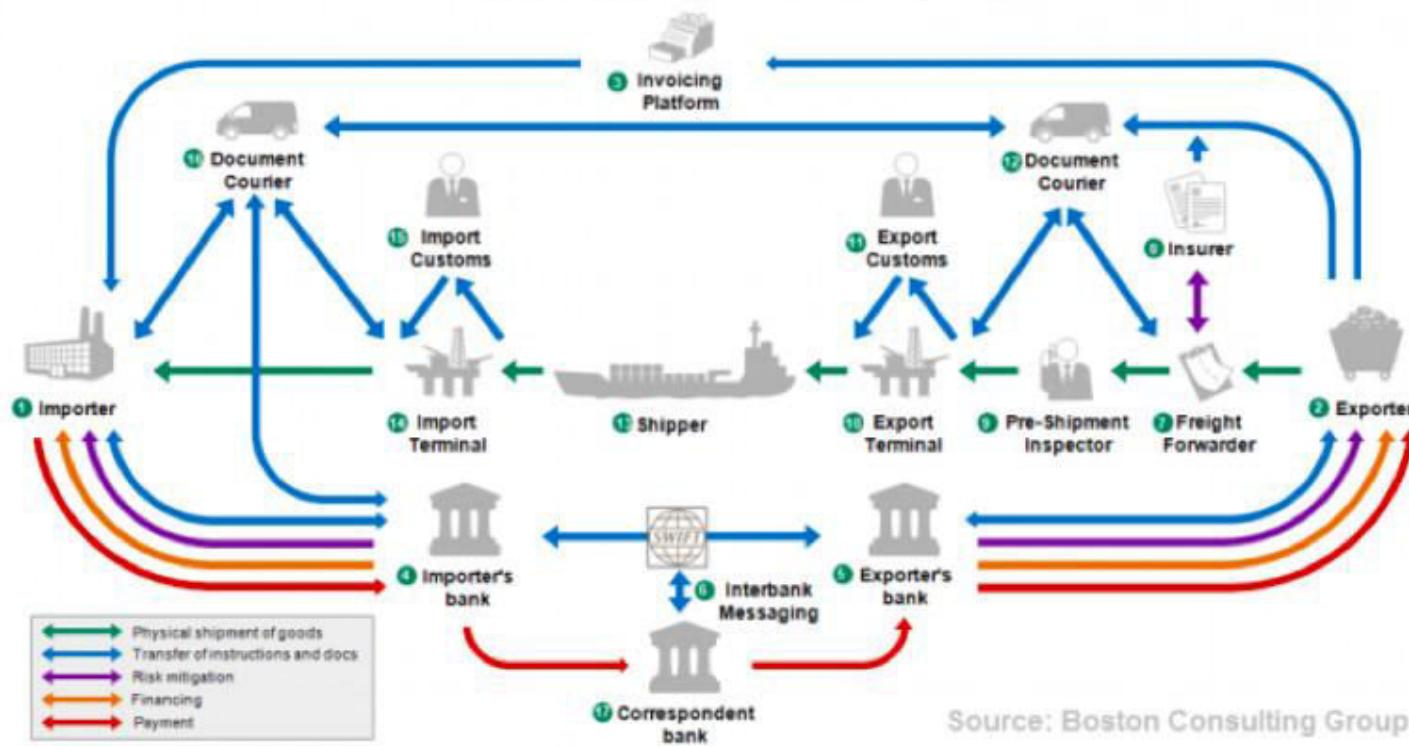


Financial Applications of Distributed Ledger Technology

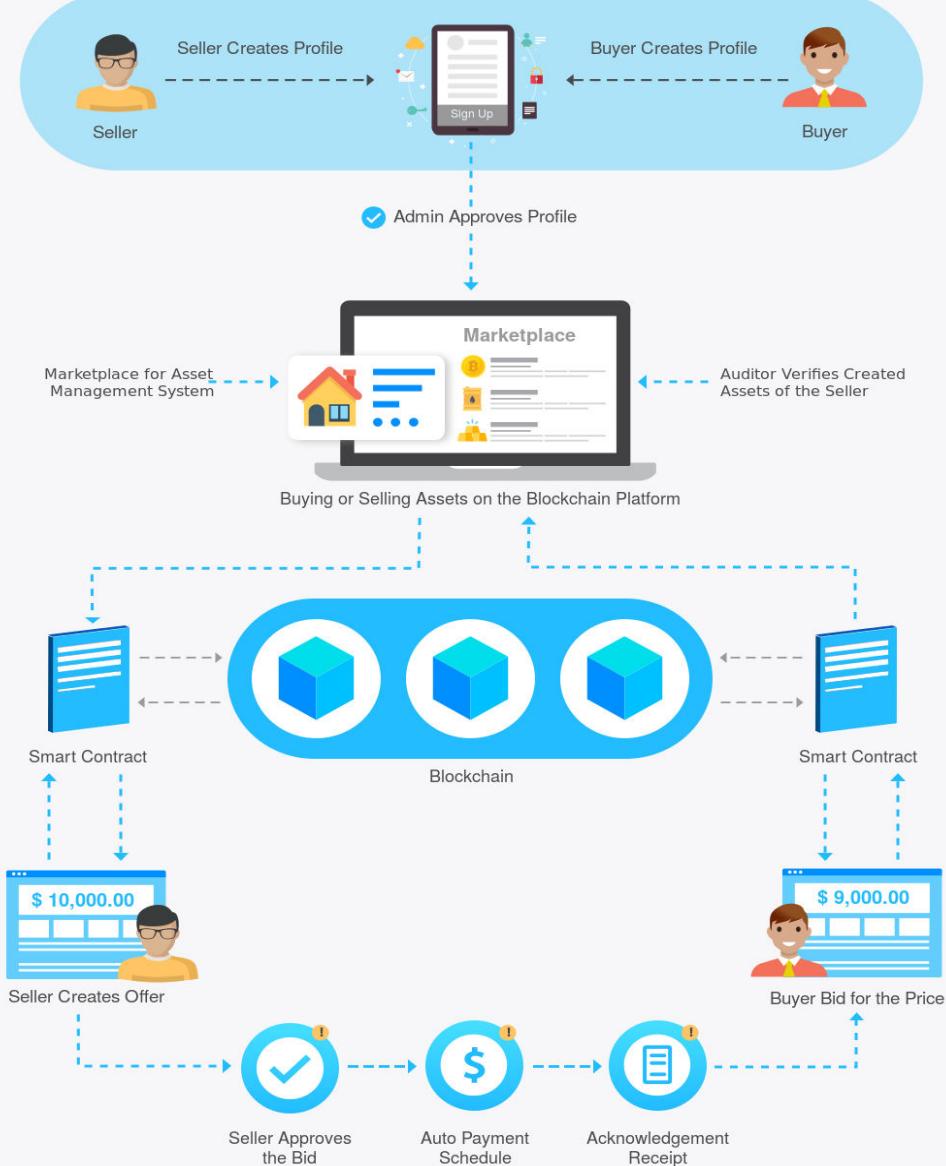


Shared ledgers for supply chains

The international trade “ecosystem”



- Tokenized blockchains

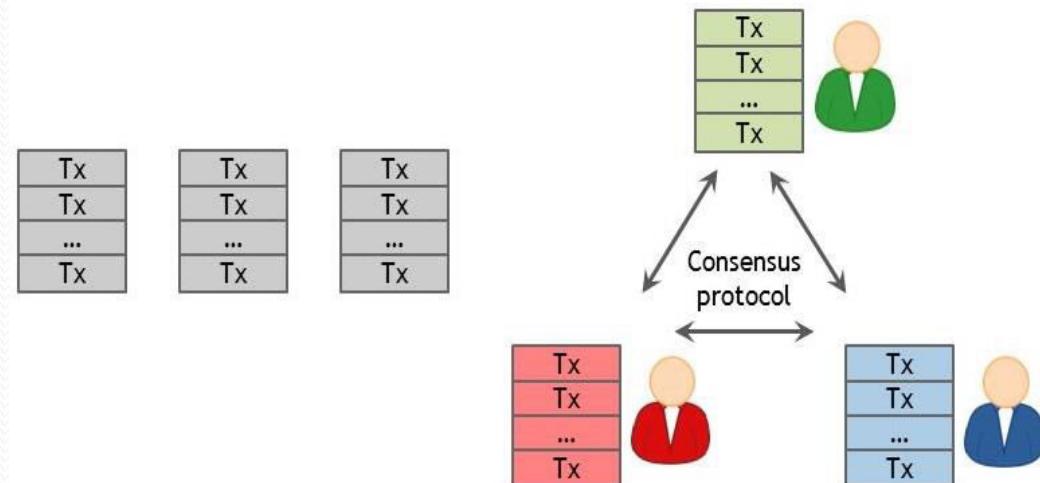


Consensus

Definition :

“Consensus decision-making is a group decision-making process in which group members develop, and agree to support a decision in the best interest of the whole. Consensus may be defined professionally as an acceptable resolution, one that can be supported, even if not the “favourite” of each individual. Consensus is defined by Merriam-Webster as, first, general agreement, and second, group solidarity of belief or sentiment.”

A procedure to reach in a common agreement in a distributed or decentralized multi-agent platform

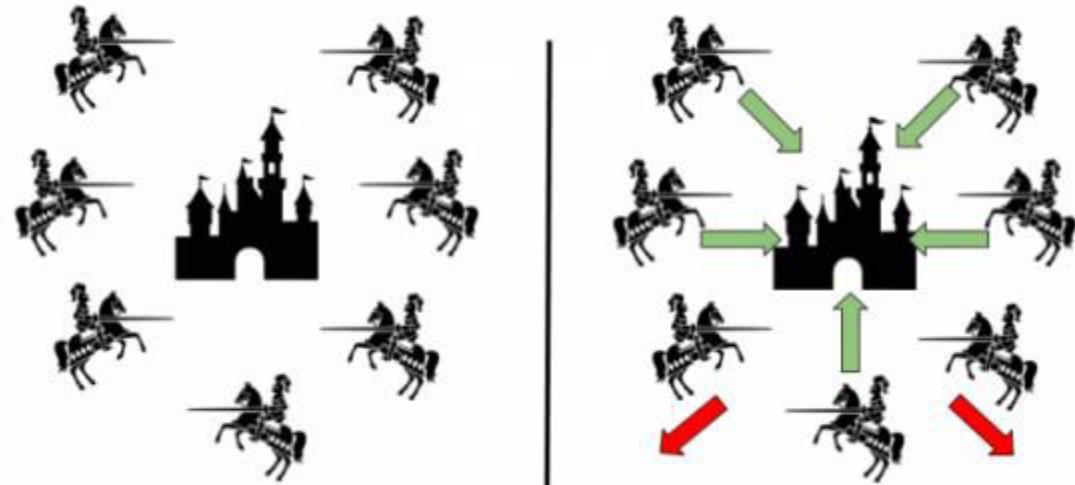


Decision making process

The Byzantine Generals Problem

Described in 1982 by Lamport, Shostak and Pease.

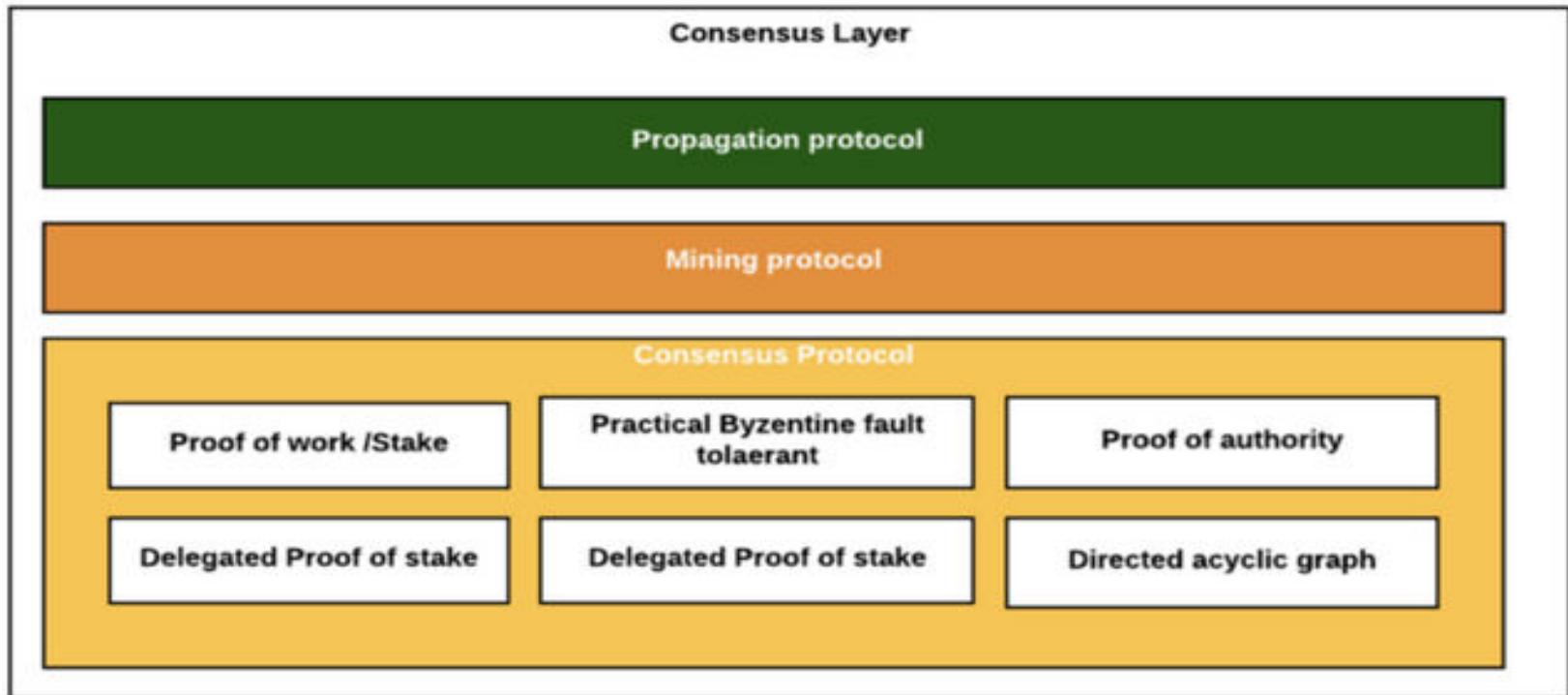
it is a generalized version of the Two Generals Problem with a twist.



It describes the same scenario, where instead more than two generals need to agree on a time to attack their common enemy.

The added complication here is that one or more of the generals can be a traitor, meaning that they can lie about their choice (e.g. they say that they agree to attack at 0900 but instead they do not).

Consenses Algorithms



Blockchain Security

Dr. Bhawana Rudra

NITK

- Disclaimer: The images and content are taken from various sources of web

- The use of blockchain technology does not remove inherent cybersecurity risks that require thoughtful and proactive risk management.
- Many of these inherent risks involve a human element. Therefore, a robust cybersecurity program remains vital to protecting the network and participating organizations from cyber threats, particularly as hackers develop more knowledge about blockchain networks and their vulnerabilities.
- Existing cybersecurity standards and guidance remain highly relevant for ensuring the security of systems that interface and/or rely on blockchain networks.
- Subject to certain adjustments to consider specific attributes of blockchain technology, existing standards and guidance provide a strong foundation for protecting blockchain networks from cyberattacks.

- In addition to general principles and controls, there are specific cybersecurity standards with relevance to blockchain technology which already exist and are in wide use by many industries.
- For instance, the NIST Cybersecurity Framework expressly states that it is “**not a one-size-fits-all approach to managing cybersecurity risk**” because “organizations will continue to have unique risks—different threats, different vulnerabilities, different risk tolerances—and how they implement the practices in the [Framework] will vary.”
- With that said, even though the Framework was not designed for blockchain technology specifically, its standards are broad enough to cover blockchain technology and to help institutions develop policies and processes that identify and control risks affecting blockchain technology.

Why Security?

- Node level security
- Bugs: software and poorly written contracts
- Dishonest miner
- Misuse of consensus protocol
- Network vulnerabilities
- Application vulnerabilities

Security Consideration at different levels

- Network Security
- Configuration of cryptography protocols/algorithms
- Key management
- Security Management
- Node availability and trust management
- Blockchain/DLT Platform layer security concerns (From Next Slide)
- Phishing attack

- **Cyber and Network-based Attacks**

- Blockchain technologies are immutable . This is only true for transactions which have been included in a published block.
- Transactions that have not yet been included in a published block within the blockchain are vulnerable to several types of attacks.
- For blockchain networks which have transactional timestamps, spoofing time or adjusting the clock of a member of an ordering service could have positive or negative effects on a transaction, making time and the communication of time an attack vector.
- Denial of service attacks can be conducted on the blockchain platform or on the smart contract implemented on the platform.
- Blockchain networks and their applications are not immune to malicious actors who can conduct network scanning and reconnaissance to discover and exploit vulnerabilities and launch zero-day attacks.
- In the rush to deploy blockchain-based services, newly coded applications (like smart contracts) may contain new and known vulnerabilities and deployment weaknesses that will be discovered and then attacked through the network just like how websites or applications are attacked today.

- **Malicious Users**
 - While a blockchain network can enforce transaction rules and specifications, it cannot enforce a user code of conduct.
 - This is problematic in permissionless blockchain networks, since users are pseudonymous and there is not a one-to-one mapping between blockchain network user identifiers and users of the system.
 - Permissionless blockchain networks often provide a reward (e.g., a cryptocurrency) to motivate users to act fairly; however, some may choose to act maliciously if that provides greater rewards.
 - The largest problem for malicious users is getting enough power (be it a stake in the system, processing power, etc.) to cause damage

- Once a large enough malicious collusion is created, malicious mining actions can include:
 - Ignoring transactions from specific users, nodes, or even entire countries.
 - Creating an altered, alternative chain in secret, then submitting it once the alternative chain is longer than the real chain.
 - The honest nodes will switch to the chain that has the most “work” done (per the blockchain protocol).
 - This could attack the principle of a blockchain network being tamper evident and tamper resistant .
 - Refusing to transmit blocks to other nodes, essentially disrupting the distribution of information (this is not an issue if the blockchain network is sufficiently decentralized).

- **No Trust**

- Another common misinterpretation comes from people hearing that there is no “trusted third party” in a blockchain and assuming blockchain networks are “trustless” environments.
- While there is no trusted third party certifying transactions in permissionless blockchain networks (in permissioned systems it is less clear, as administrators of those systems act as an administrator of trust by granting users admission and permissions), there is still a great deal of trust needed to work within a blockchain network:
 - There is trust in the cryptographic technologies utilized. For example, cryptographic algorithms or implementations can have flaws.
 - There is trust in the correct and bug free operation of smart contracts, which might have unintended loopholes and flaws.
 - There is trust in the developers of the software to produce software that is as bug-free as possible.
 - There is trust that most users of the blockchain are not colluding in secret. If a single group or individual can control more than 50 percent of all block creation power, it is possible to subvert a permissionless blockchain network. However, generally obtaining the necessary computational power is prohibitively expensive.
 - For blockchain network users not running a full node, there is trust that nodes are accepting and processing transactions fairly.

• **Resource Usage**

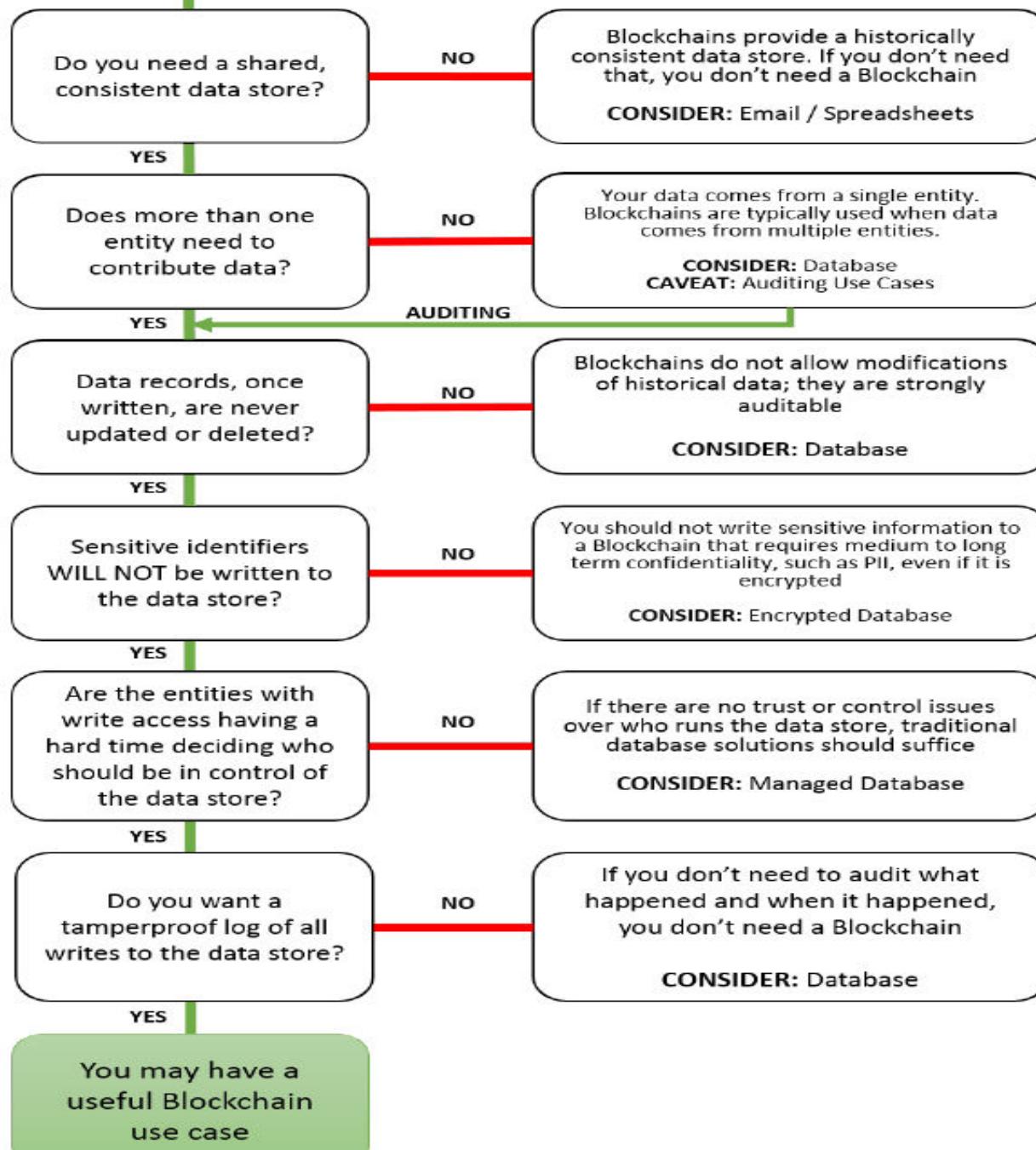
- Blockchain technology has enabled a worldwide network where every transaction is verified and the blockchain is kept in sync amongst a multitude of users.
- For blockchain networks utilizing proof of work, there are many publishing nodes expending large amounts of processing time and, more importantly, consuming a lot of electricity.
- A proof of work method is an effective solution for “hard to solve, easy to verify” proofs; however, it generally requires significant resource usage.
- Because of their different applications, and trust models, many permissioned blockchain technologies do not use a resource intensive proof, but rather they utilize different mechanisms to achieve consensus.
- The proof of work consensus model is designed for the case where there is little to no trust amongst users of the system.
- It ensures that publishing nodes cannot game the system by always being able to solve the puzzles and thereby control the blockchain and the transactions added to it.
- However, a major concern surrounding the proof of work consensus model is its use of energy in solving the puzzles.

- **Inadequate Block Publishing Rewards**

- A potential limitation is the risk of inadequate rewards for publishing a block. The combination of increased competition, increased computational resources needed to have meaningful contributions to pools of publishing nodes, and highly volatile market prices in the cryptocurrency market creates the risk that the expected return for any given cryptocurrency may be less than the power costs needed to run publishing node software. Thus, the expected return for other cryptocurrencies may be more attractive.
- Cryptocurrencies that are not able to consistently and adequately reward publishing nodes risk delays in publishing blocks and processing transactions. These delays could therefore reduce confidence in the cryptocurrency, reducing its market value further.
- It could then become increasingly less attractive for publishing nodes to contribute to that cryptocurrency's publishing efforts.
- Even worse, such weakened cryptocurrencies open themselves up to being attacked by nodes with large amounts of resources that may maliciously alter the blockchain or deny service to users attempting to submit transactions.

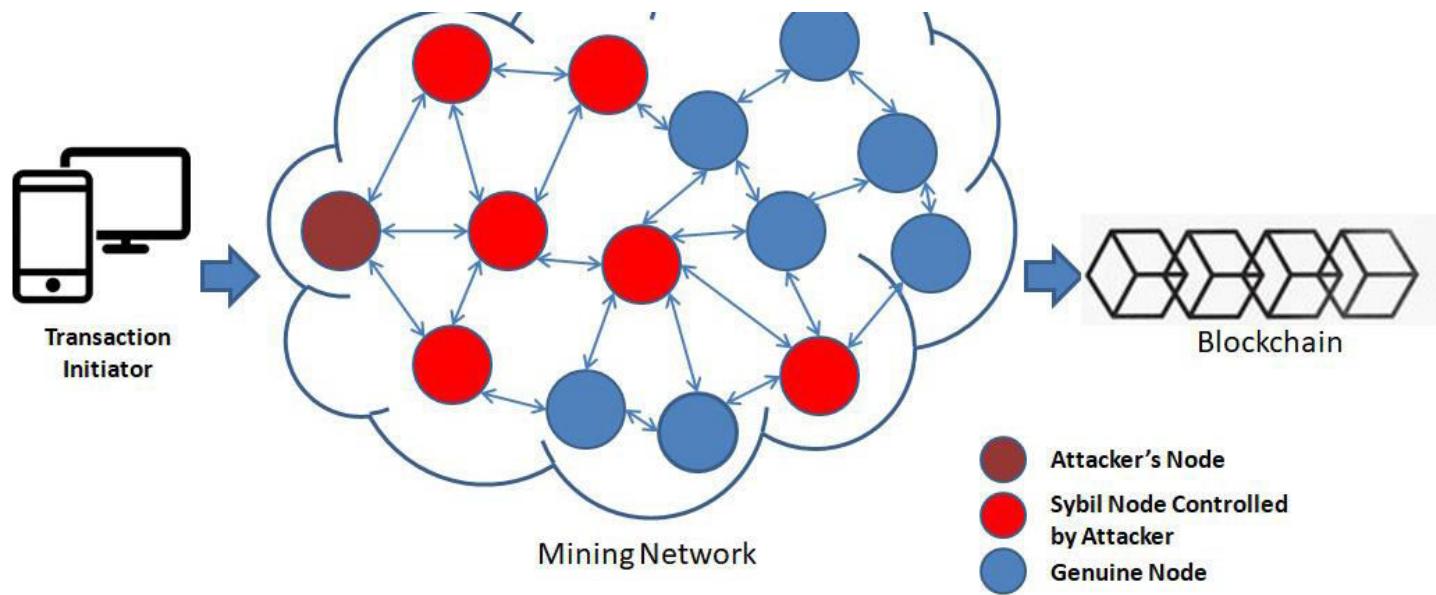
- **Public Key Infrastructure and Identity**

- When hearing that blockchain technology incorporates a public key infrastructure, some people immediately believe it intrinsically supports identity. This is not the case, as there may not be a one-to-one relationship of private key pairs to users (a user can have multiple private keys), nor is there a one-to-one relationship between blockchain addresses and public keys (multiple addresses can be derived from a single public key).
- Digital signatures are often used to prove identity in the cybersecurity world, and this can lead to confusion about the potential application of a blockchain to identity management.
- A blockchain's transaction signature verification process links transactions to the owners of private keys but provides no facility for associating real-world identities with these owners.
- In some cases, it is possible to connect real-world identities with private keys, but these connections are made through processes outside, and not explicitly supported by, the blockchain.
- For example, a law enforcement agency could request records from an exchange that would connect transactions to specific individuals.
- Another example is an individual posting a cryptocurrency address on their personal website or social media page for donations, this would provide a link from address to real world identity.
- While it is possible to use blockchain technology in identity management frameworks that require a distributed ledger component, it is important to understand that typical blockchain implementations are not designed to serve as standalone identity management systems. There is more to having secure digital identities than simply implementing a blockchain.

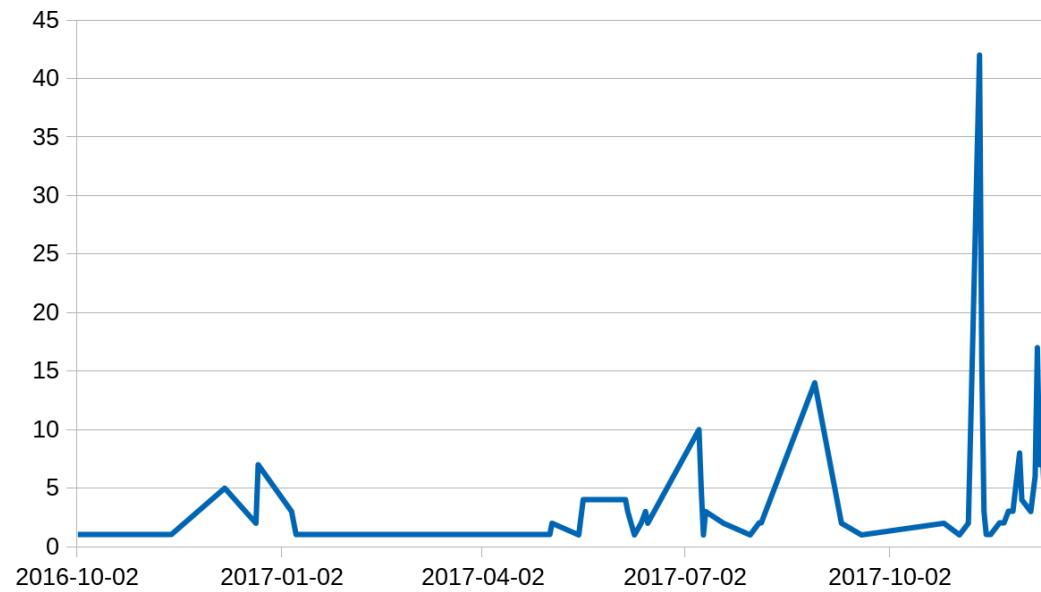


Blockchain Platform layer security concerns

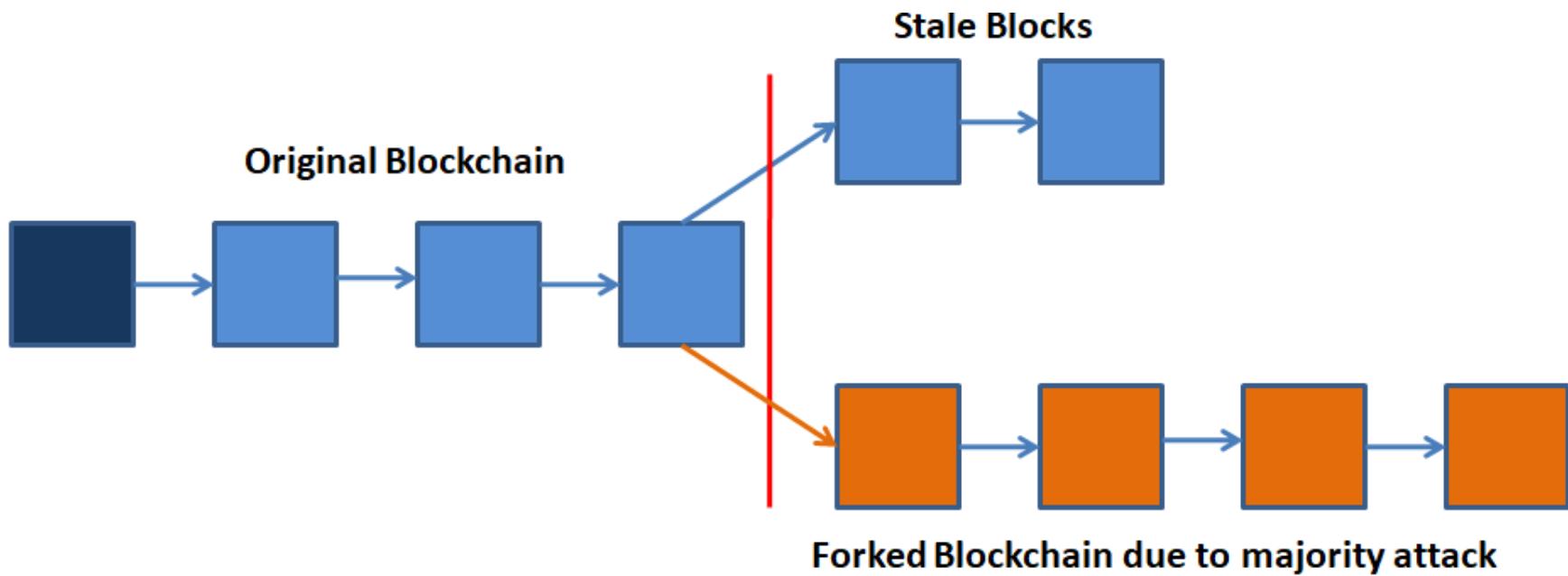
- Sybil Attack- Lack of robust identity management
 - Attacker creates multiple identities (maybe virtual) and take control of the network
 - To forward attackers block faster than the genuine users block
- DoS/DDoS Attack- Inherent from the Sybil attack



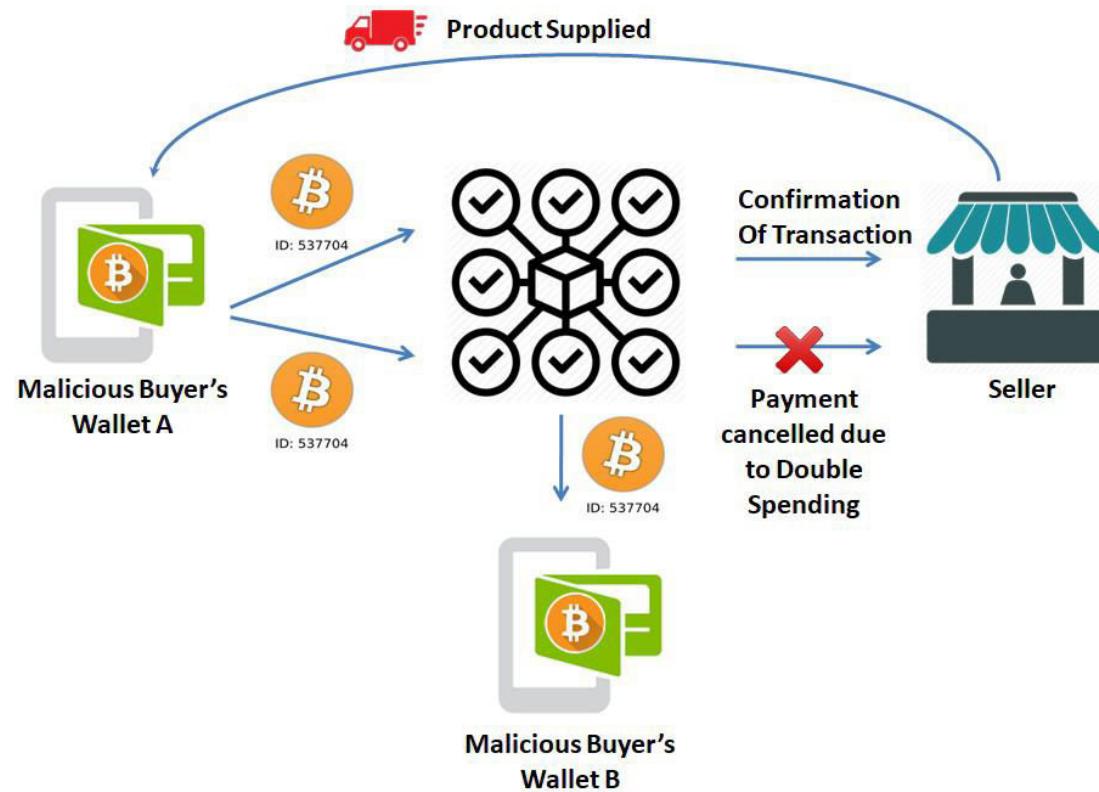
- DoS/DDoS Attack:
- Giulio Prisco, 2017 states that cryptocurrency industry are frequently suffering from DDoS attacks
- In Q3 2017, more than 75% bitcoin sites were attacked through DDoS attack



- Majority Attack



- Double Spending



- **Finnery Attack** : A malicious buyer mines a block of transaction T_2 in private blockchain, and then it performs a transaction T_1 using same bitcoin with a seller
- Seller accepts transaction T_1 after receiving a confirmation from the miners saying that T_1 is valid and appended in Blockchain
- After receiving the product from seller, buyer releases the pre-mined block in the blockchain network, and as a result, a blockchain forks
- If the new mined block added in the forked blockchain, then all the miners will consider the forked blockchain to add further blocks, which makes forked blockchain longest in the network, original blockchain will be ignored to add further blocks
- As a result, transaction T_1 becomes invalid and seller loses the product, and the malicious buyer gets back bitcoin via transaction T_2
- However, an adversary can only perform double spending in the presence of one-confirmation sellers Blockchain/DLT Platform layer security concerns

- **Brute-force Attack:** Here, consider a resourceful buyer has control over some miners in the network, which can collectively perform private mining scheme
- As like Finney Attack, an attacker can continuously work on the extending private blockchain
- Assume that a seller waits for x *confirmations from the original* blockchain to accept a transaction, and then sends a product
- Here, if, an attacker can release the pre-mined the x *number of blocks* in the network to cause fork chain longer than the original chain, then there is a possibility of double spending attack Blockchain/DLT Platform layer security concerns

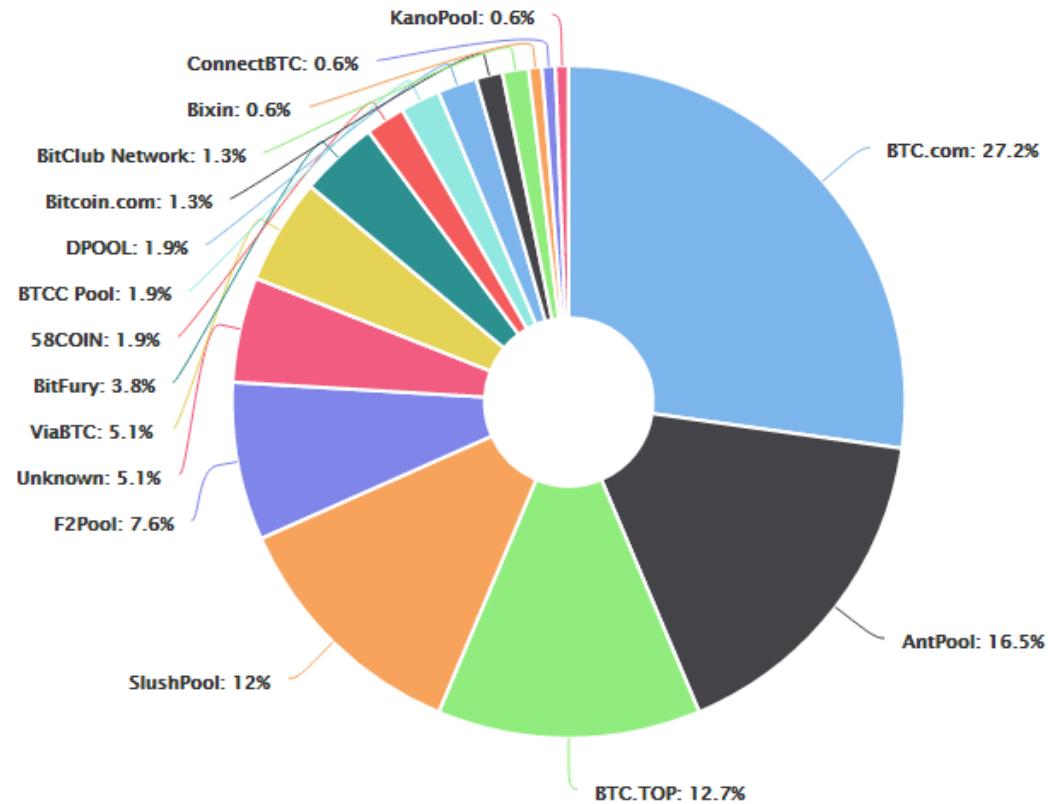
BlockchainPlatform layer security concerns

- **Vector 76 Attack:** Here, a malicious buyer holds a pre-mined block of deposit transaction and waits for the announcement of next block
- He can immediately send the pre-mined block to his peers
- In such case, there is a possibility that the miners consider the forked blockchain caused due to pre-mined block as the original chain
- After this, buyer can immediately send a withdrawal request transaction for the same coins which were deposited in his previous transaction block, mined privately
- Here, if the original chain does not have block of deposit transaction, the deposit request will be invalidated
- However, the withdrawal request transaction will be validated, which causes the loss of coins.

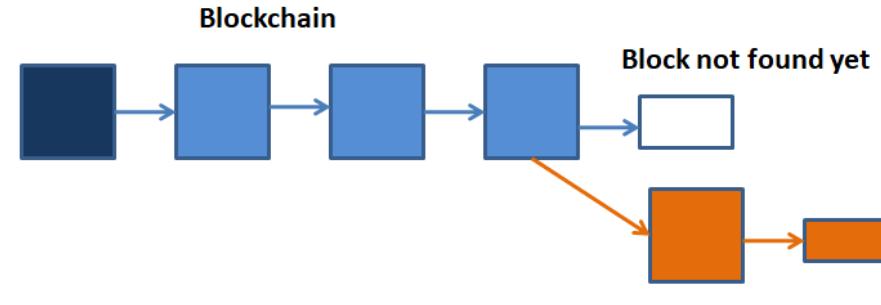
- **Balance Attack:** It aims at delaying communications among pools of the miners with balanced hash power
- It can be performed against the PoW-based consensus mechanism in Bitcoin blockchain
- A trade-off between the communication delay and the mining power required to make double spend attack in Ethereum with high probability
- It can be possible, if attacker's mining pool has higher computation power than the others
- In addition, network delay, connectivity, and ratio of the number of honest miners and malicious miners are affecting parameters
- Majority attack can also cause double spending attack

- **Mining Pool Attacks:**
- Mining pools are formed for increasing the computational power in order to increase the chances of winning the bitcoin rewards.
- Each mining pool is governed by pool manager which gives block mining work to miners in the same pool.
- These miner provides partial proofs-of-work (PPoWs) and full proofsof-work (FPoWs) to the pool manager
- A new block with the FPoW is broadcasted in the network of miners by a pool manager to get the bitcoin reward.
- This reward is then distributed to the pool miners.
- A Bitcoin network is included solo miners, public pools and private pools.

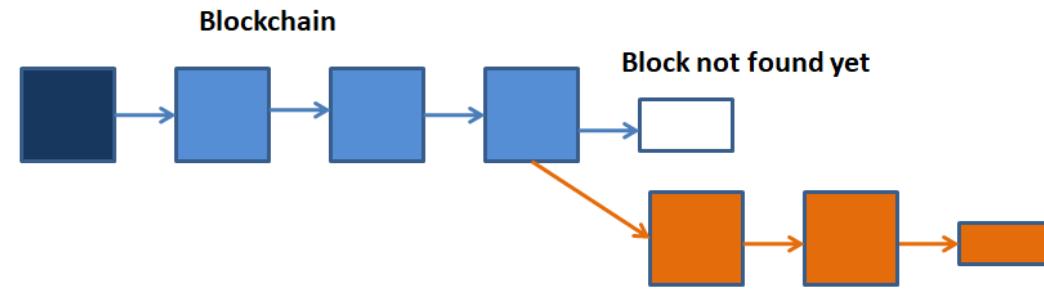
- Mining Pool Attacks



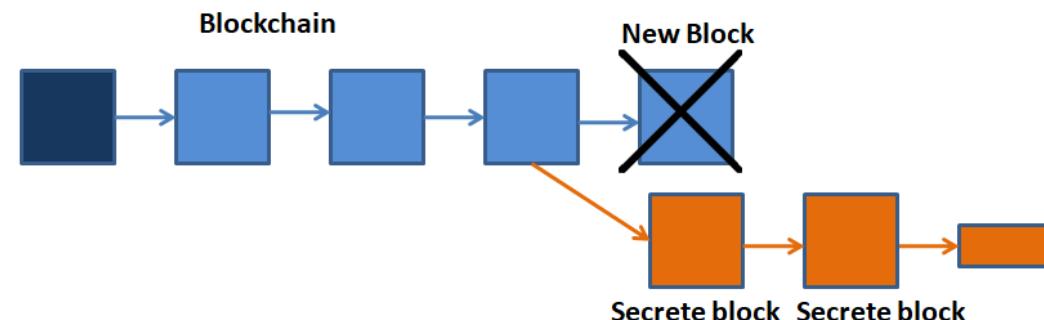
- Selfish Mining



(I) Secret block has caused a fork chain



(II) Secret blocks are continuously added to fork chain before any block mined for original chain has caused a fork chain



(III) New block is added to original chain, however discarded as fork chain become longer

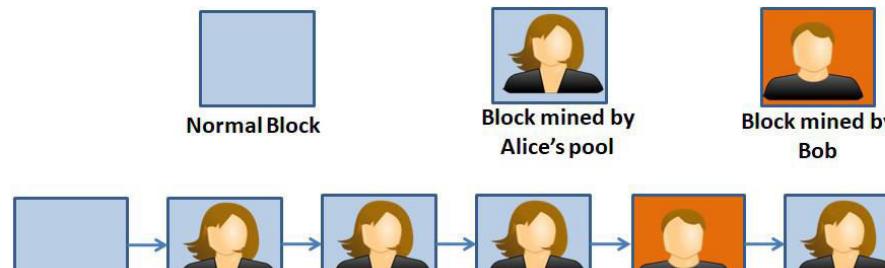
- **Block withholding (BWH):**
 - Consider, a pool member is not revealing a mined block and submits shares with PPoWs
 - Here, an adversary can either make other pool members to lose or he can perform selfish mining

- **Pool Hopping attack:**
 - Here, a malicious miner in mining pool analyzes the number of shares submitted by other miners in same pool to the manager for finding a new block
 - In this case, if a large number of shares are submitted and a new block has not been found, he can get a very small share from the reward since reward is distributed based on the submitted shares
 - As a result, it may be profitable for a malicious miner to jump to another pool or mine independently

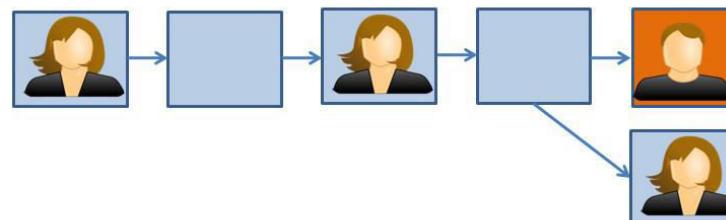
- **Bribery attack:**
 - An attacker may attempt to get the majority of computational power for a short duration through bribery
 - An attacker can perform Out-of-Band Payment, where he pays directly to the owners of the computing resources, who mine the blocks for an attacker
 - In **Negative-Fee Mining Pool**, an attacker can form a mining pool by paying higher return.
 - In addition, an attacker can attempt to bribe through Bitcoin itself, where an attacker's fork involves bribe money freely available to the miners for considering the fork. This is called as In-Band Payment via Forking
 - Thus, an attacker can have majority of hash power which can further lead to double spending and Distributed Denial-of- Service (DDoS)

- **Blacklisting Through Punitive Forking:** In punitive forking, an attacker (e.g. Alice) attempts to censor the Bitcoin addresses owned by certain people (e.g. Bob) and prevents Bob to spend his bitcoins
 - Consider, Alice has more than 50% of network hashrate, then she can refuse to work on a blockchain containing transactions created by Bob's addresses and she will not extend the blockchain by her blocks
 - If some other miners in a network includes a block with Bob's transaction, Alice will perform fork and attempt to make fork chain longer so that the blocks containing Bob's transactions will be invalidated and cannot be published.
 - However, a weak adversary (e.g. Alice) with lower hashrate can also cause delays and inconveniences for Bob's transaction.

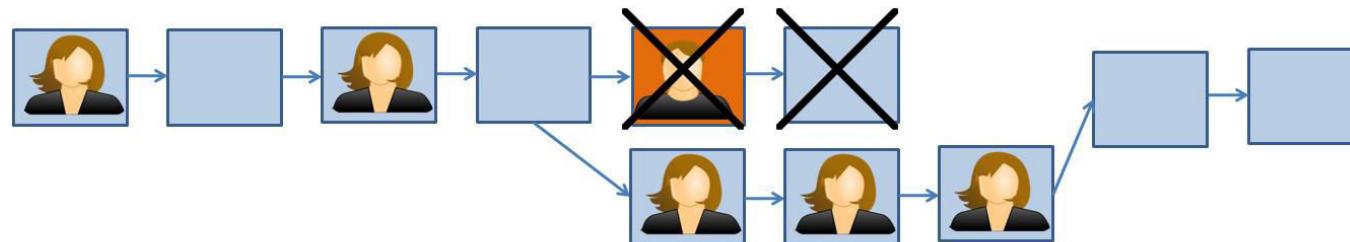
- Blacklisting Through Punitive Forking:



(I) Normal scenario of creating blockchain



(II) Alice and bob both have added their block to the blockchain. Alice has created fork chain



(III) Bob's block is invalidated by adding more number of Alice's blocks and making fork chain longer

- **Feather forking:** To perform punitive forking for blacklisting Bob, Alice should have more than 50% network hashrate.
 - However, an attacker (Alice) with less hashrate can announce that she will fork, if she finds any block having Bob's transaction in the blockchain and then she will give up
 - Here, Alice forks the blockchain and continues to her fork chain until wins
 - Later she gives up and works on original chain to extend it after a Bob's transaction block
 - Alice may lose rewards, but she will be able to block the transaction with some probability

- **Wallet Theft:**
 - The wallet thefts are mainly performed using different methods like phishing, system hacking, installation of buggy software, and incorrect usage of the wallet.
 - Since there is no trusted third party, if a victim lost the private key associated with her wallet due to a hard drive crash or a virus corrupts data or lost the device carrying the key, all the bitcoins in the wallet has been considered lost for forever.
 - There is no mechanism available for recovering the lost Bitcoins.

- How to Set Up a Bitcoin Miner. <http://www.coindesk.com/information/how-to-set-up-a-miner>
- G. Nash. 2017. Build the Tiniest Blockchain. <https://medium.com/crypto-currently/lets-build-the-tiniest-blockchain-e70965a248b>
- D. V. Flymen. 2017. Learn Blockchains by Building One. <https://hackernoon.com/learn-blockchains-by-building-one-117428612f46>
- Build Your Own Blockchain: A Python Tutorial. <http://ecomunsing.com/build-your-own-blockchain>

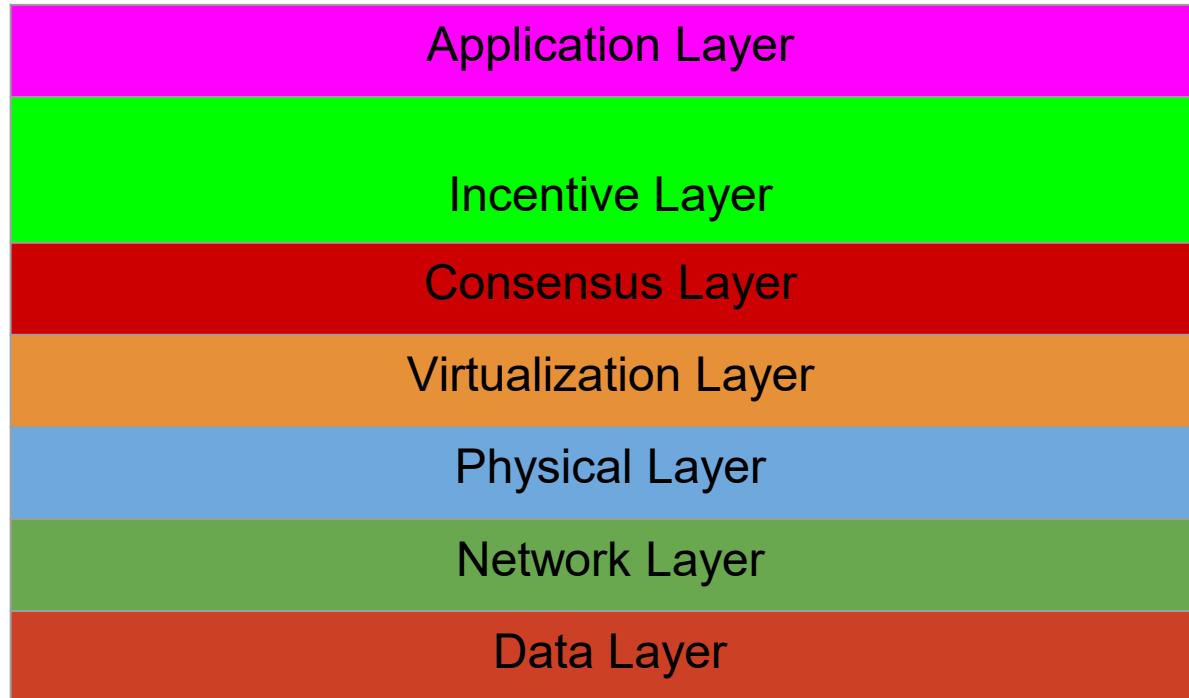
References

- D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, (2018)“Everything you wanted to know about the blockchain: Its promise, components, processes, and problems,” IEEE Consumer Electronics Mag., vol. 7, no. 4, pp. 6–14, July.
- M. Swan, (2015) Blockchain: blueprint for a new economy, 1st ed. O'Reilly Media, Jan.
- F. Tschorß and B. Scheuermann, (2016) “Bitcoin and beyond: A technical survey on decentralized digital currencies,” IEEE Commun. Surveys & Tut., vol. 18, no. 3, pp. 2084–2123, Mar..
- S. Nakamoto, (2018) Bitcoin: A peer-to-peer electronic cash system, 2008. Available online: <https://bitcoin.org/bitcoin.pdf>. (Accessed 1 February 2018).
- A.M. Antonopoulos, (2014) Mastering Bitcoin: Unlocking Digital Cryptocurrencies, O'Reilly Media, Inc.
- Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, (2017).Blockchain challenges and opportunities: a survey, Int. J. Web Grid Serv.
- M. Pilkington ,(2016) Blockchain technology: principles and applications, in: F.X. Olleros, M. Zhegu (Eds.), Research Handbook on Digital Transformations, Edward Elger Publishing, Northampton, MA
- D. Tapscott , A. Tapscott , (2016) Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Penguin Random House LLC, New York, NY.
- A . Kosba, A . Miller, E. Shi, Z. Wen, C. Papamanthou, (2016) Hawk: the blockchain model of cryptography and privacy-preserving smart contracts, in: Proceed- ings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, May 2016, doi: 10.1109/SP.2016.55
- A. Wright, P. De Filippi, (2015) Decentralized blockchain technology and the rise of Lex cryptographia. <https://ssrn.com/abstract=2580664> , March.
- X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A.B. Tran, S. Chen, (2016) The blockchain as a software connector, in: Proceedings of the 2016 Thirteenth Working IEEE/IFIP Conference on Software Architecture (WICSA), Venice, Italy, April 2016, doi: 10.1109/WICSA.2016.21 .
- R.C. Merkle, “A digital signature based on a conventional encryption function”. Proceedings of the Advances in Cryptology, CRYPTO '87. Lecture Notes in Computer Science. vol. 293. p. 369. doi:10.1007/3-540-48184-2_32.

THANKYOU

Blockchain Architecture

Blockchain Architecture



Data Layer

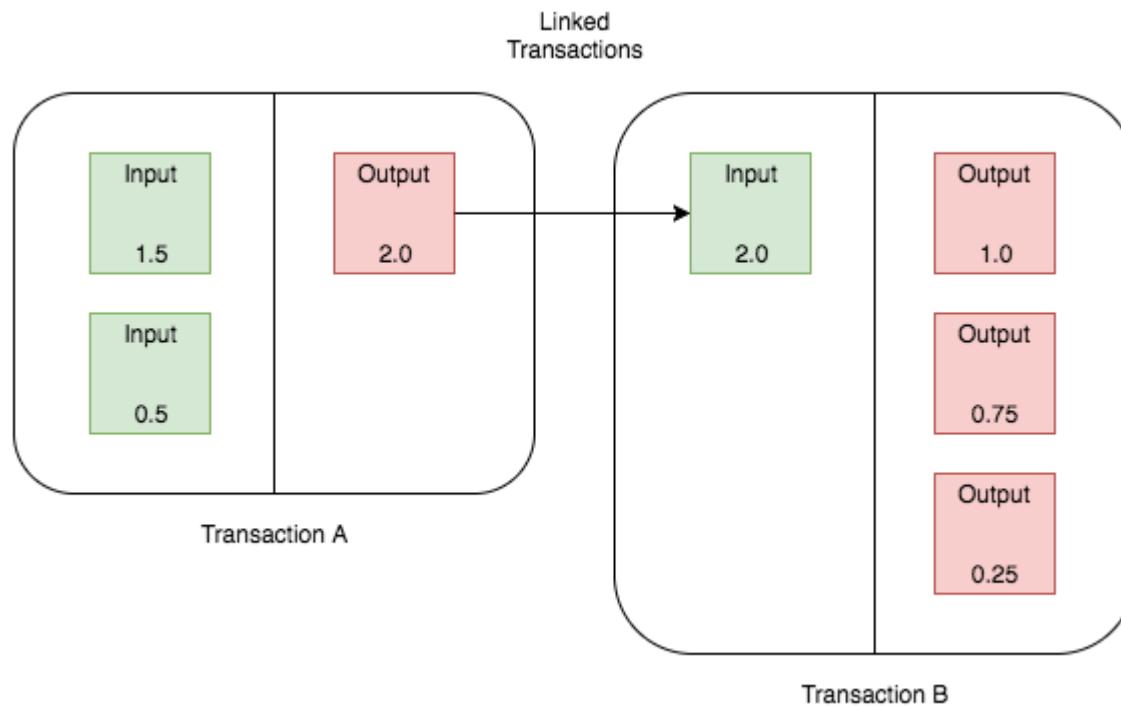


This layer acts as the **blockchain data structure** and **physical storage**. The ledger is built using a **linked list or Merkle trees**, of blocks, which are encrypted using **asymmetric encryption**. The Data layer consists of the following components such as **Blocks, Markle Trees and Transactions, Blockchain**

Transactions

- Transactions are the **smallest building blocks** of a blockchain system. They normally consist of a recipient address, a sender address, and a value.
- Transactions are bundled and delivered to each node in the form of a block. As new transactions are distributed throughout the network, they are **independently verified and "processed"** by each node.

Transactions(Contd...)



Blocks

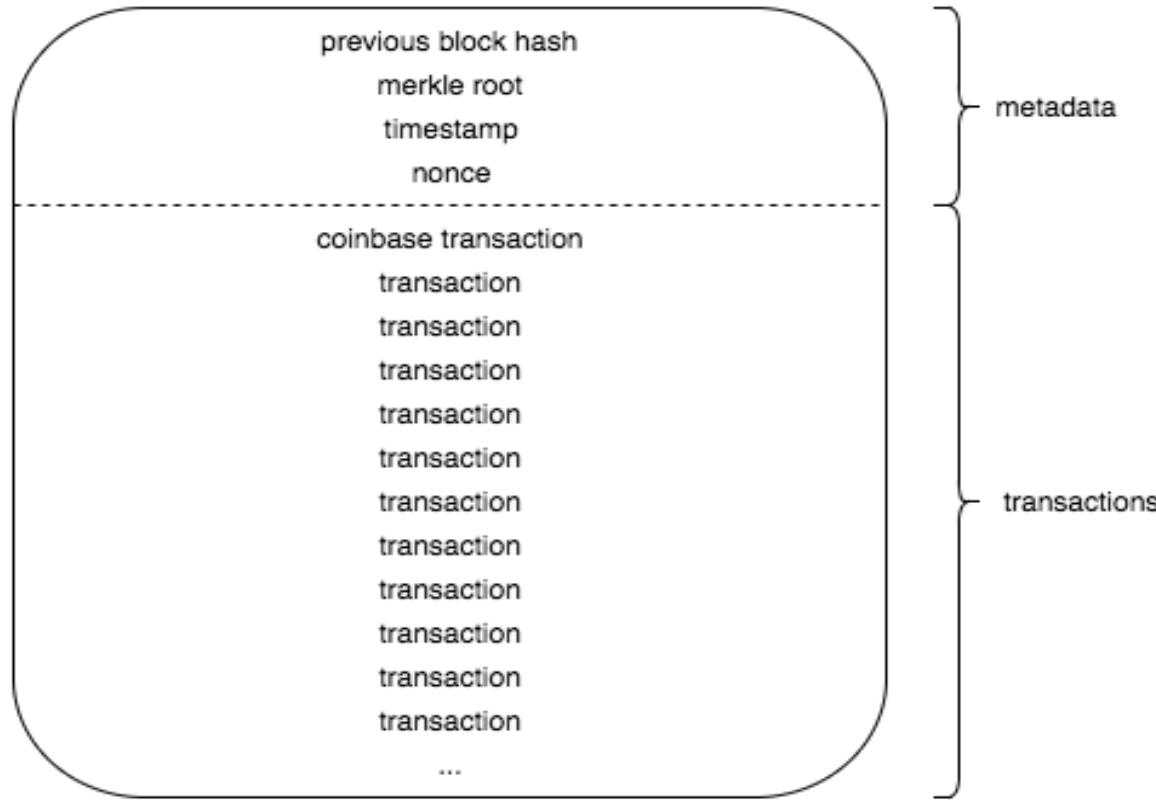
- Blocks are **data structures** whose purpose is to bundle sets of transactions and be distributed to all nodes in the network. Blocks are created by **miners**
- **Miners** are the equivalent to the processing network of a credit card company. They take pending transactions, verify that they are cryptographically accurate, and package them into blocks to be stored on the blockchain.
- Blocks contain a **block header**, which is the metadata that helps verify the **validity** of a block.

Blocks(Contd...)

Typical block metadata contains:

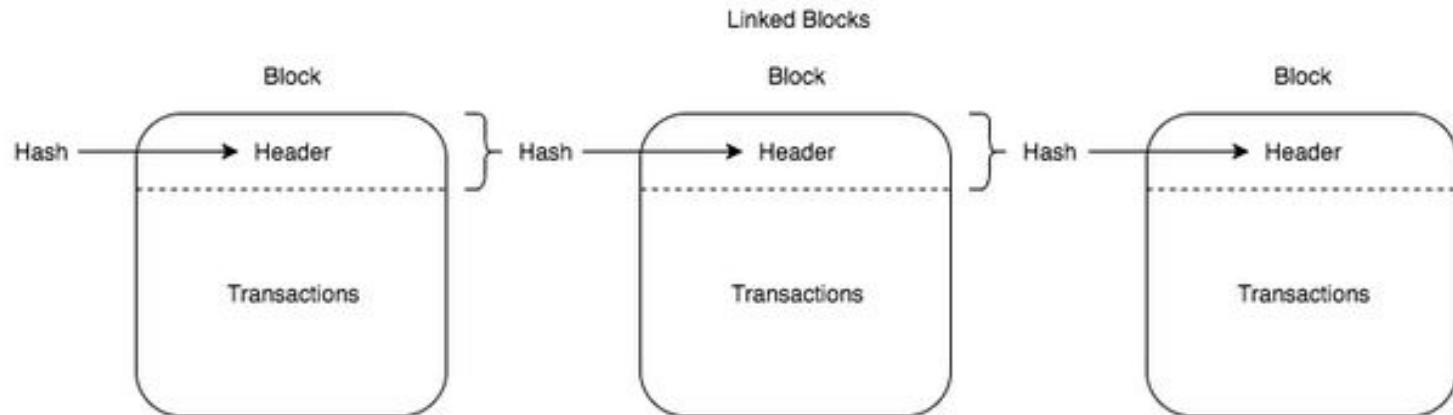
- version
- previous block header hash
- merkle root hash
- time
- nBits
- nonce ("number used once")

Block



Blocks(Contd...)

- Nodes, or the computers in the network, independently decide and concur upon which "chain of blocks" is the longest and most valid.
- As a block is created and set around the network, each node processes the block and decides where it fits into the current overarching blockchain ledger.



Blocks(Contd...)

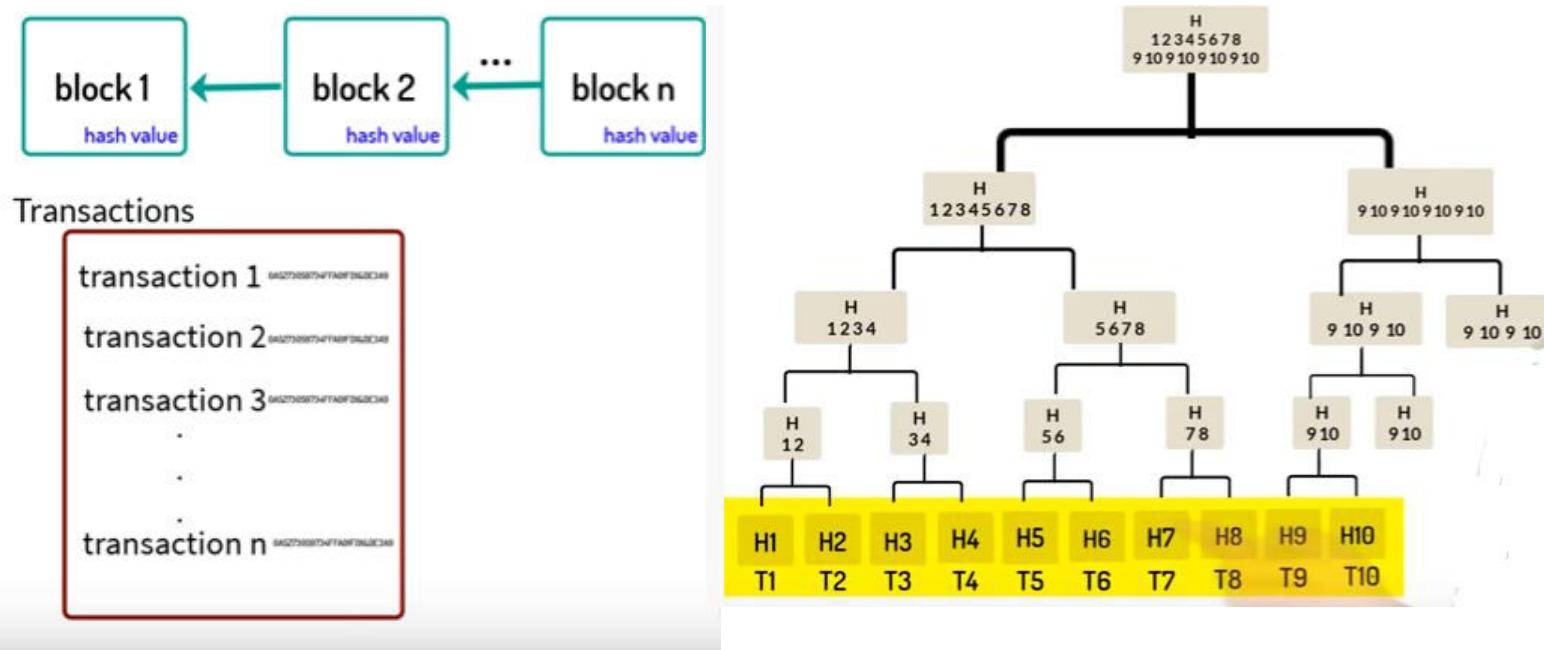
In terms of the **blockchain architecture**, there are different types of blocks based on their functionalities:

- **Main branch blocks**
- **Side branch blocks**
- **Orphan blocks**

Markle Tree

- Merkle trees are binary trees containing **cryptographic hashes**.
- It is constructed by recursively hashing pairs of nodes until there is only one hash, called the *root*, **or merkle root**. The cryptographic hash algorithm used in merkle trees is **SHA256** applied twice, also known as **double-SHA256**.
- It is a **binary tree**, it needs an even number of leaf nodes. If there is an odd number of transactions to summarize, the last transaction hash will be duplicated to create an even number of leaf nodes, also known as a *balanced tree*.

Markle Tree(Contd...)

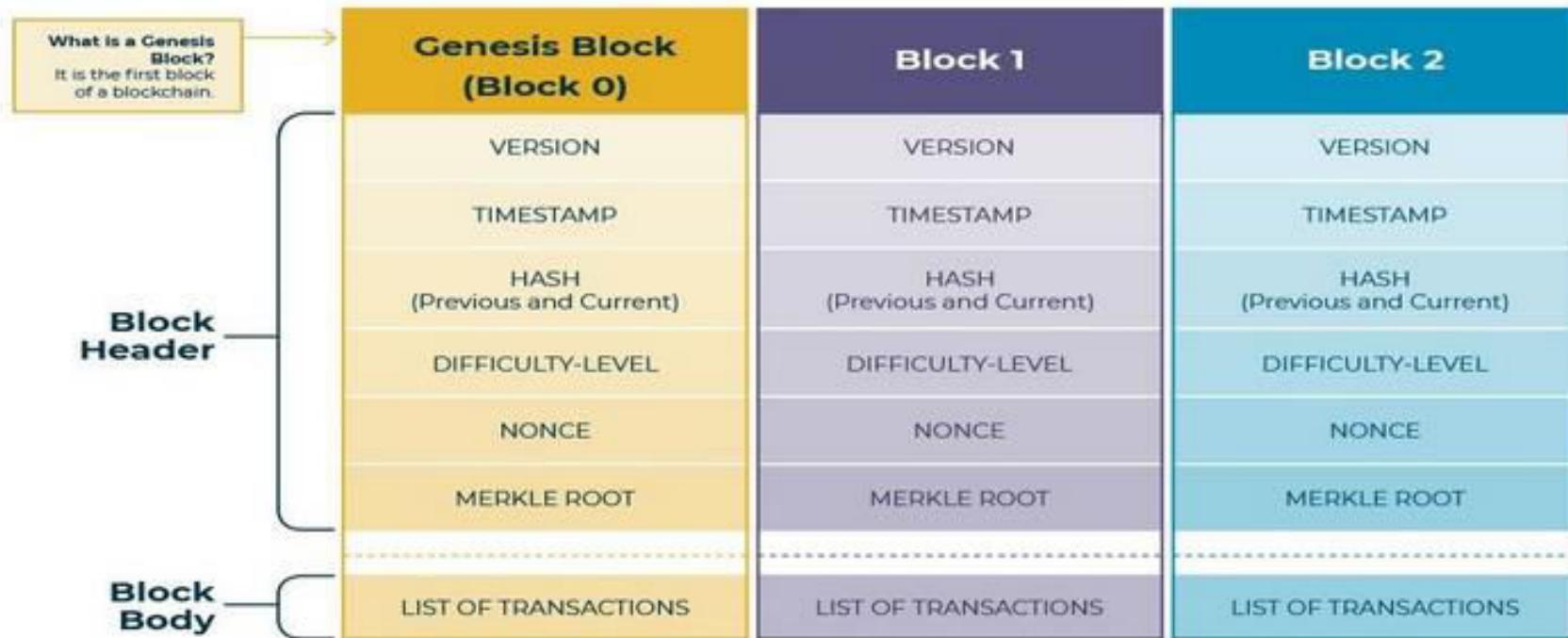


H1 = SHA256(SHA256(Transaction 1))
H12= SHA256(SHA256(H1+H2))

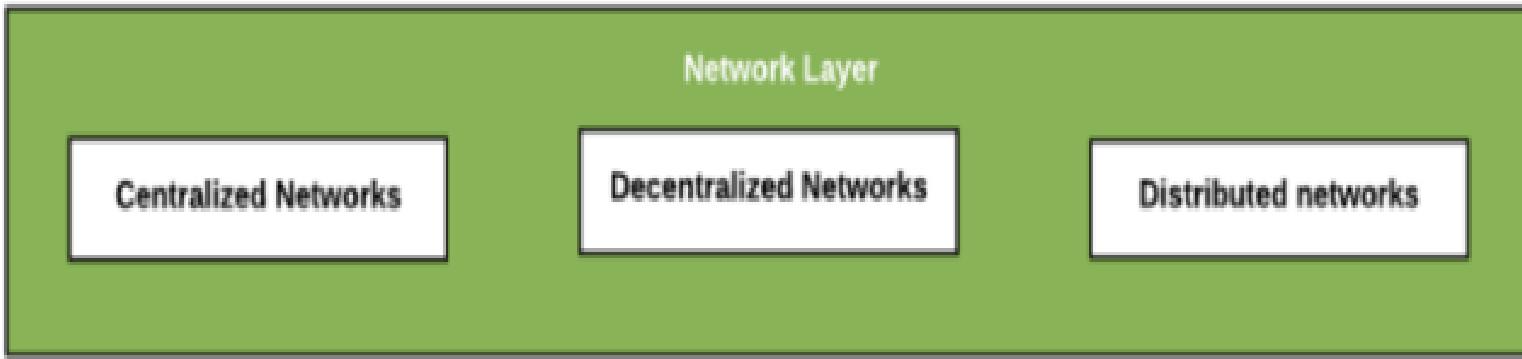
Blockchain

- A blockchain will thus consist of all the transactions defined by user as linked together by storing the **hash of the previous block**, where each block stores the **root hash of the Merkle tree** where the actual transactions are stored.
- Transmission of blockchains across a given network is further secured using **asymmetric encryption** or **public private key pairs**.

Blockchain(Contd...)



Network Layer



The blockchain makes use of the **distributed network**, such that everyone downloads and interacts with all the information that is available on the Blockchain.

P2P network is used to propagate/broadcast transactions among nodes.

Network Layer

- **P2P Network:**
- The blockchain is a **peer to peer (P2P) network** working on the IP protocol.
- A P2P network is a flat topology with **no centralized node**. All nodes equally provide and can consume services while collaborating via a **consensus algorithm**.
- **P2P networks** are generally **more secure** because they do not have a single point of attack or failure as in case of a centralized network.

P2P Network(Contd...)

- Different types of Blockchain
- Public Blockchain
- Private Blockchain
- Consortium / Hybrid Blockchain

P2P Network(Contd...)

- Public Blockchain:
- Publicly accessible blockchains are termed as **Public Blockchain**. These blockchains have no restrictions on the participatory and validator.
- This type of blockchain is the **uncontrollability** of the blockchain.Hence it ensures that the **data is secure** and helps in the **immutability** of the records.
- All the nodes connected to this public blockchain will have equal authority, and hence, these public blockchain becomes **fully distributed**.
- **Bitcoin, Ethereum, and Litecoin** are some of the examples of Public Blockchain being used in real-world scenarios.

P2P Network(Contd...)

- Private Blockchain:
- This particular blockchain requires the participants to be invited before they can be a part of the blockchain.
- As these blockchains are **more centralized**, they can be governed and regulated by someone who can make sure that the governors are guiding participants.
- Private blockchains usually have a **network administrator** who can take care of the user permissions in case any particular user requires additional authority on the go.
- These are typically used in **private organizations** to store sensitive information about the organization.

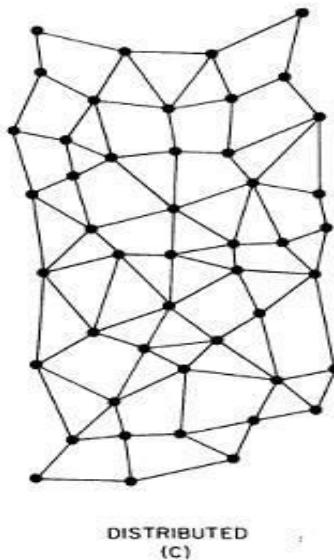
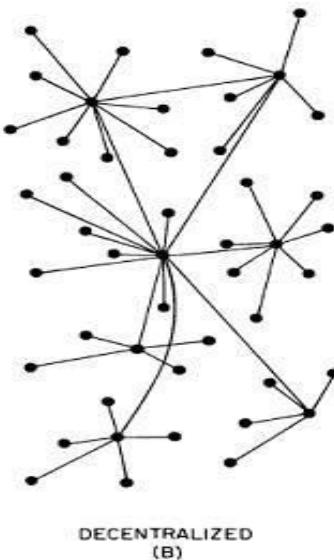
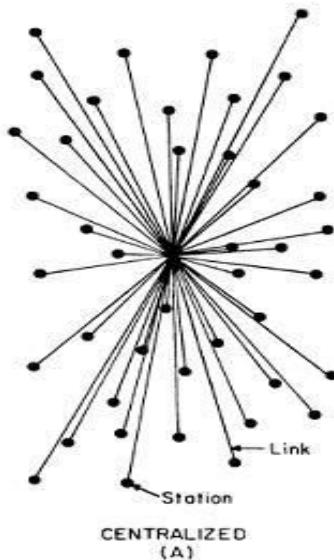
P2P Network(Contd...)

- Consortium / Hybrid Blockchain:
- This blockchain is divided into **two different types**, where some nodes are **private**, while the other nodes are **public**.
- As a result, some of the nodes will be allowed to participate in the transactions. The other nodes are to control the consensus process. This is a **hybrid blockchain** between **private and public blockchains**.

Network Layer(Contd...)

There are mainly 3 types of networks utilized by blockchain platforms

- Centralized networks
- Decentralized networks
- Distributed networks



Network Layer(Contd...)

Centralized Networks:

- A centralized network, on the other hand, is made up of parties whose identities are known. Thus, the system is **valid** because **only credible and reputable participants** can post to the ledger.

Network Layer(Contd...)

Decentralized Networks:

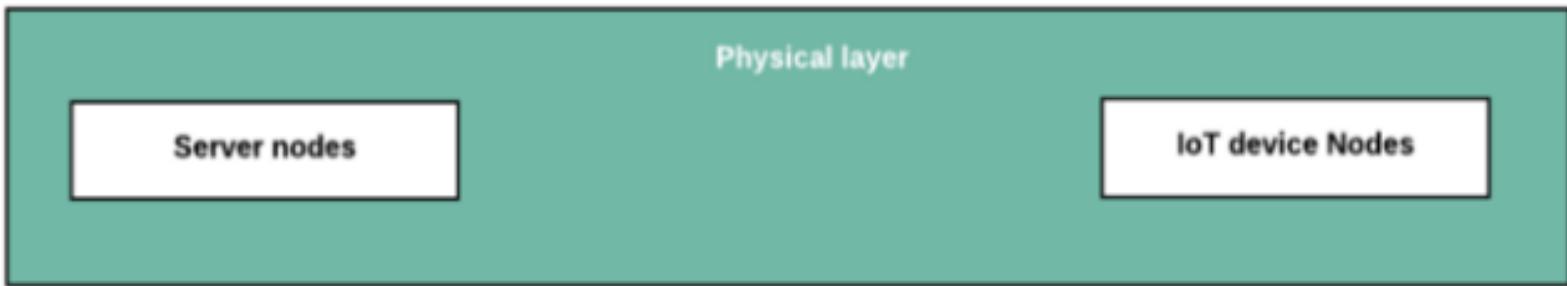
- In a decentralized network, anyone can participate and transact on the ledger. As a result, mechanisms must exist in order to prevent the vulnerabilities that arise from this design and to ensure that transactions are correct.
- Bitcoin, for example, is a **decentralized blockchain** that uses **mining and proof-of-work** to maintain the integrity of the ledger and to prevent people from corrupting the system.

Network Layer(Contd...)

Distributed Networks:

- Many parties hold copies of the ledger

Physical Layer



Physical Layer(Contd..)

- It consists of **servers, edge nodes, IoT devices** which act as nodes on the blockchain network.
- These are generally connected as a **P2P network** where Peers are equally privileged, equipotent participants in the application.
- The role of a node is to **support the network** by maintaining a copy of a blockchain and, in some cases, to process transactions.
- Nodes are often arranged in the structure of trees, known as **binary trees**.
- Processing these transactions can require **large amounts of computing and processing power**.

Physical Layer(Contd..)

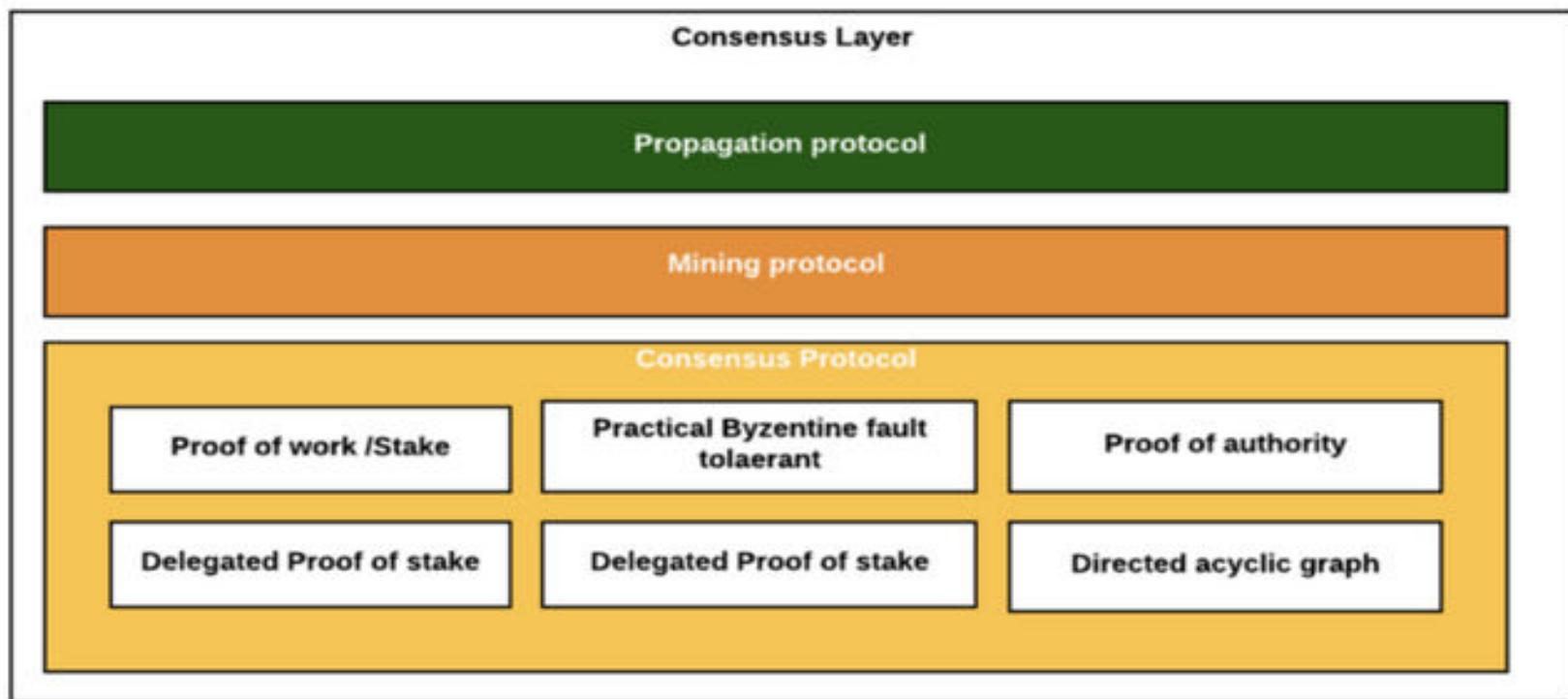
- A **full node** downloads a **complete copy of a blockchain** and checks any new transactions coming in based on the **consensus protocol** utilized by that particular cryptocurrency or utility token.
- All nodes use the same consensus protocol to remain compatible with each other. It is the nodes on the network that **confirm and validate transactions**, putting them into blocks.
- Nodes always come to their own conclusion on whether a transaction is valid and should be added to a block with other transactions, irrespective of how other nodes act.

Virtualization Layer



Hardware virtualization layer used to allocate hardware and resources to virtual machines

Consensus Layer



Consensus Layer

- This layer deals with the **enforcement of network rules** that describe what nodes within the network should do to reach consensus about the broadcasted transactions.
- Consensus allows it to **decentralized**
- So each and every change in a single blockchain is verified and adopted by another blockchain in the network.
- When we speak about consensus, we mean the collaborative process that participating nodes of the network use to agree that a transaction is valid and to keep the **distributed ledger synchronized at all times**.

Consensus Layer(Contd...)

- These consensus mechanisms **lower the risk of malicious** transactions
- To reach consensus, the **majority of the participants** need to agree that the transaction is **valid** before it is permanently recorded in the ledger.
- Once a transaction is **permanent**, no one, not even a system administrator, can delete the transaction from the ledger.

Consensus Layer(Contd...)

- The **cost and time** needed to reach consensus depends on the mechanism in place and the number of nodes participating in the consensus.
- The process of reaching a consensus on a high level include **mining and propagation of blocks** in the network defined by the consensus algorithm.

Consensus Layer(Contd...)

Example: two miners, **miner A** and **miner B**. Both miner A and miner B can decide to include **transaction X** into their block. A block has a maximum size of data. On the Bitcoin blockchain, the maximum size of a block is data up to **1 MB**. But before adding the transaction to their block, a miner needs to check if the transaction is eligible to be executed according to the **blockchain history**. If the senders' wallet balance has sufficient funds according to the existing blockchain history, the transaction is considered valid and can be added to the block.

Miners will usually **prioritize transactions** that have a **high transaction fee** set, because this gives them a **higher reward**. This mined block is then propagated to other nodes where the transaction is individually executed until a majority of them are in the same state.

There are various consensus models used in blockchains .These Blockchain consensus models consist of some particular objectives, such as:

- Coming to an agreement
- Collaboration
- Co-operation
- Equal Rights
- Participation
- Activity

Incentive Layer

Incentive Layer

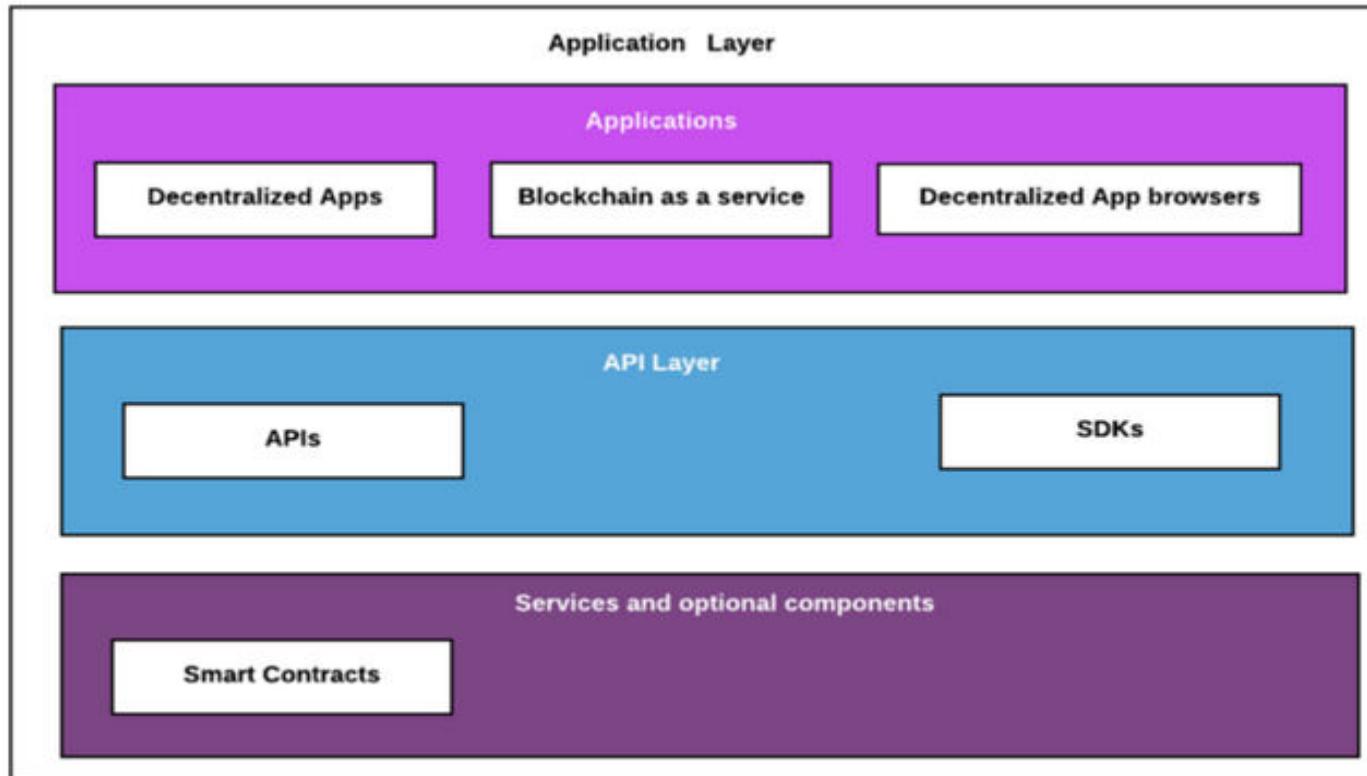
Rewards distribution

Transaction Fee

Incentive Layer

- This layer deals with the **distribution of rewards** that are earned by nodes in the network for the work they do to reach consensus. Whether this layer is implemented or not depends on the **consensus mechanism** in use.
- The incentive layer include capabilities that describe what kinds of incentives are given by the network, when and how incentives can be earned by nodes, and the minimum amount of transaction fees needed to perform actions on the blockchain.

Application Layer(Aplications, API Layer and Contract Layer)



Contract Layer

It consists of the **services and optional components** and serves to enable integration of the blockchain platform with other technologies these include

- **Data feeds**
- **Smart contracts**
- **On chain and off chain computing**
- **Digital wallets**
- **Digital ids**
- **Multi signatures**
- **Oracles**
- **DAOs and governance**
- **State channels**

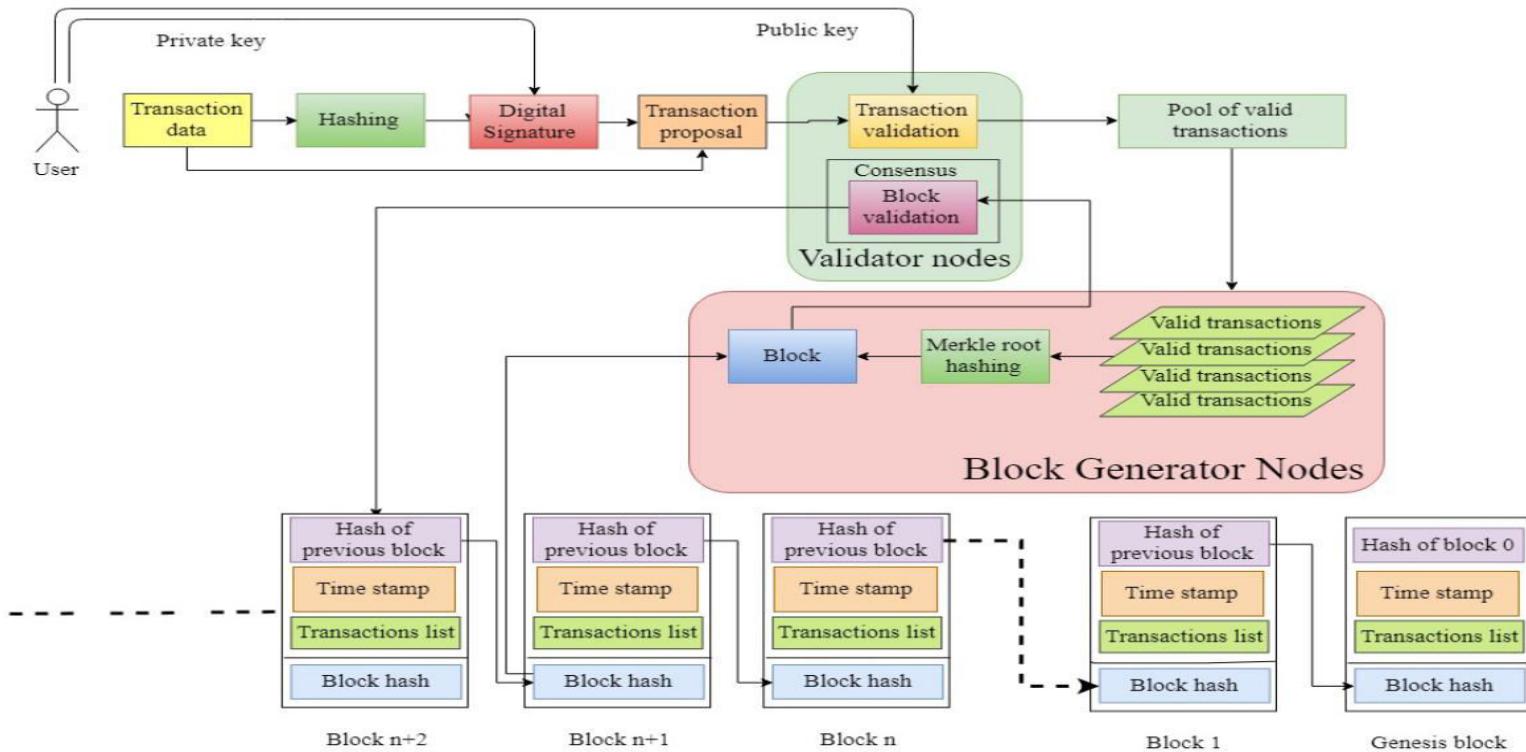
API Layer

It provides **application interfaces** on top of the blockchain, and how third-party applications can interact with the **digital ledger and smart contracts**.

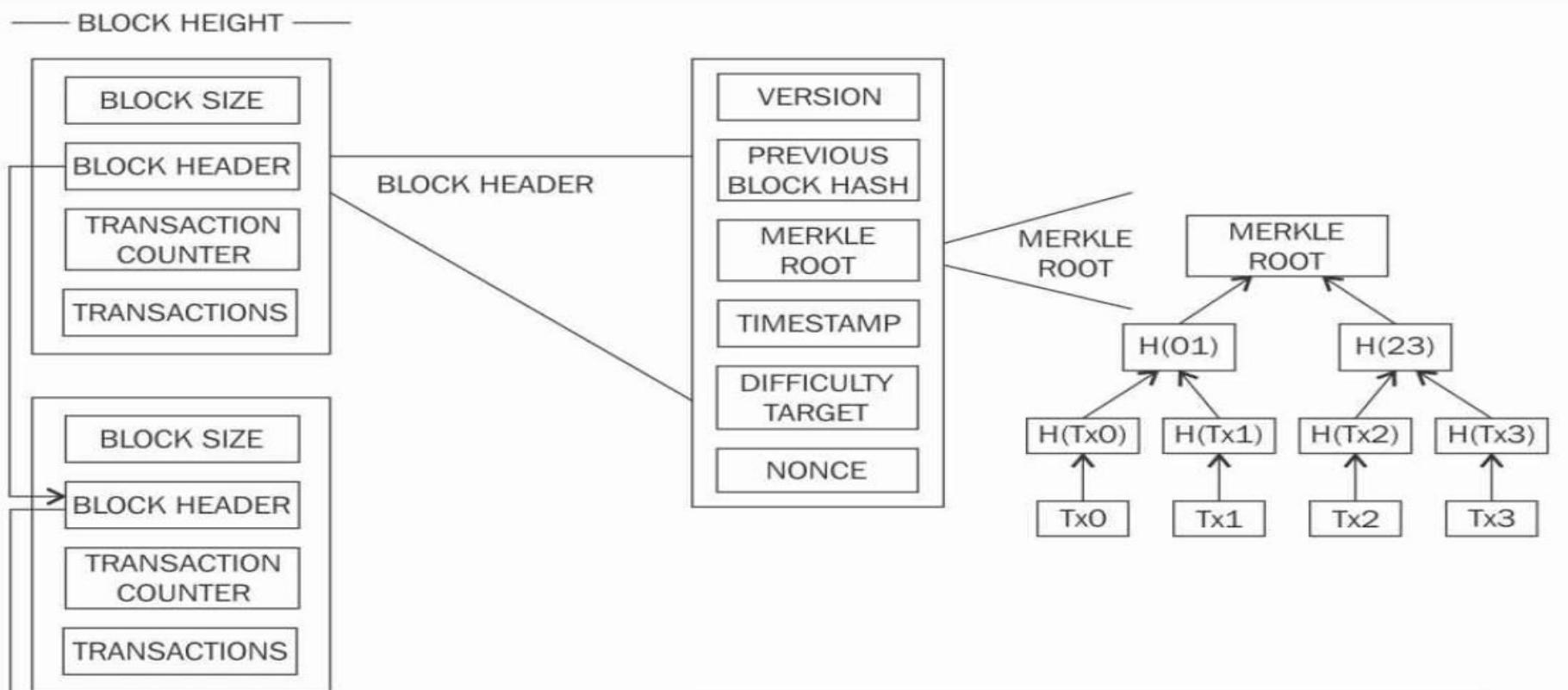
Application Layer

The application layer includes **capabilities** that provide applications on top of the blockchain.

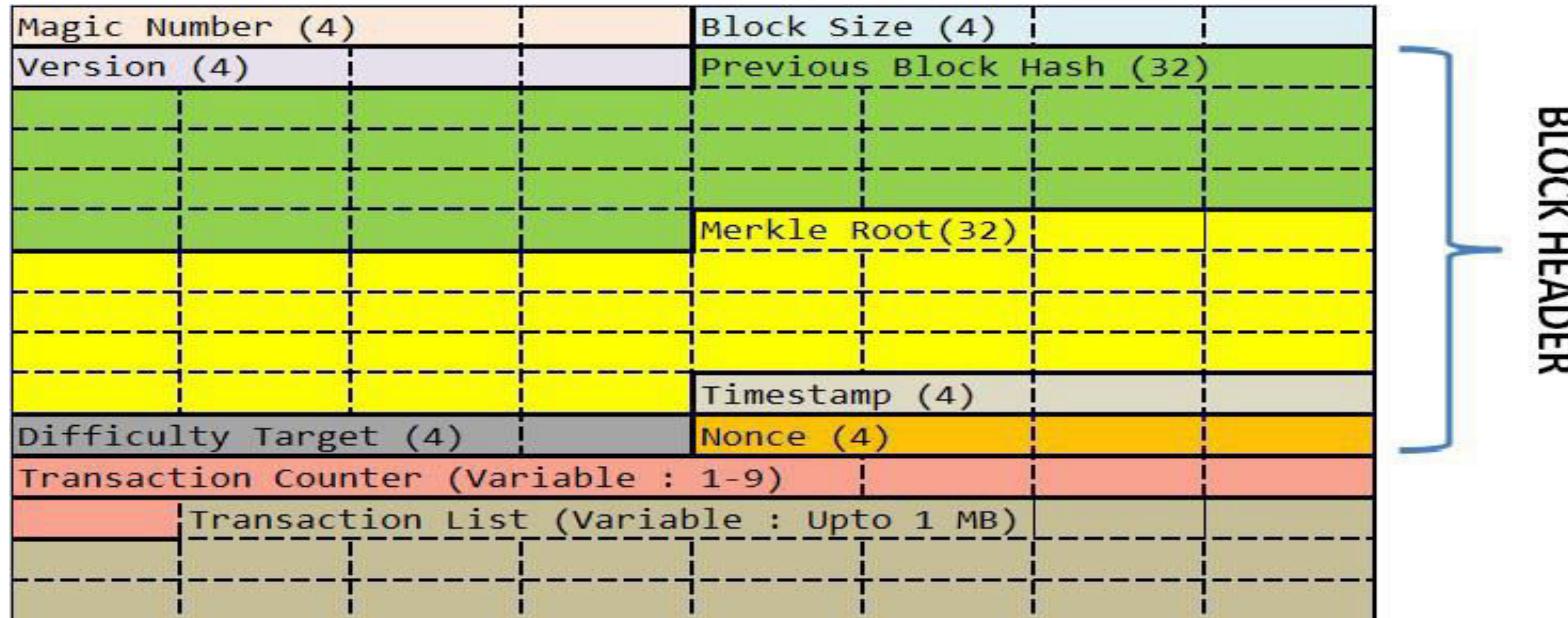
Blockchain works...



Block Structure



Block Header



Block-Detail

Magic number: an identifier for the Blockchain network, Indicates
a) Start of the block b) Data is from network

Block size: each block is fixed to 1 MB. However will be increased to 2 MB. The maximum capacity is 2 GB

Version: Each node has to implement

Timestamp, Difficulty Target, Nonce: Time and Input for Hash

Hash function: SHA-256, Merkle Tree: Hash Tree

Public Key/Private Key & ECDSA

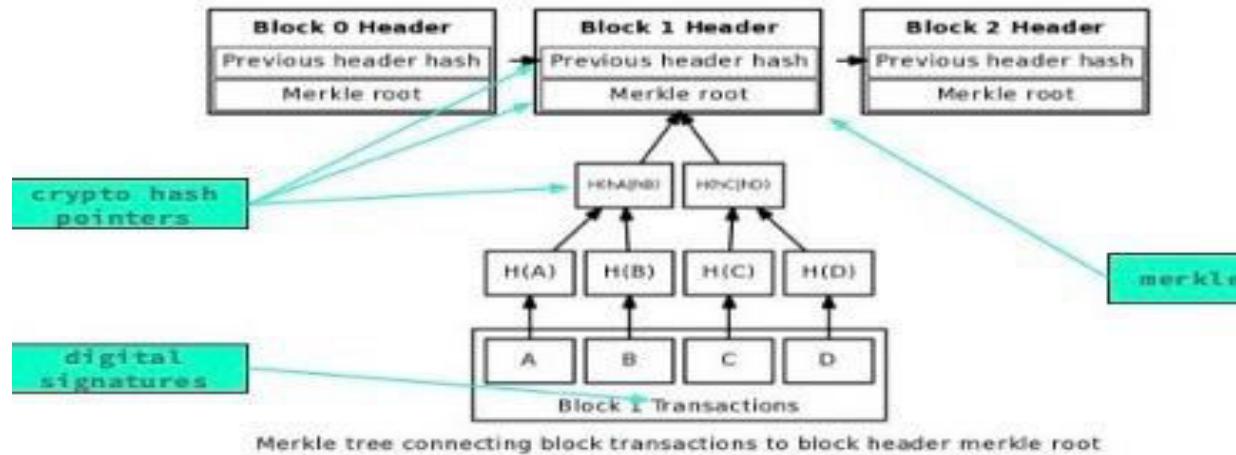
Nodes/Peers- Peer to Peer Network

Wallet

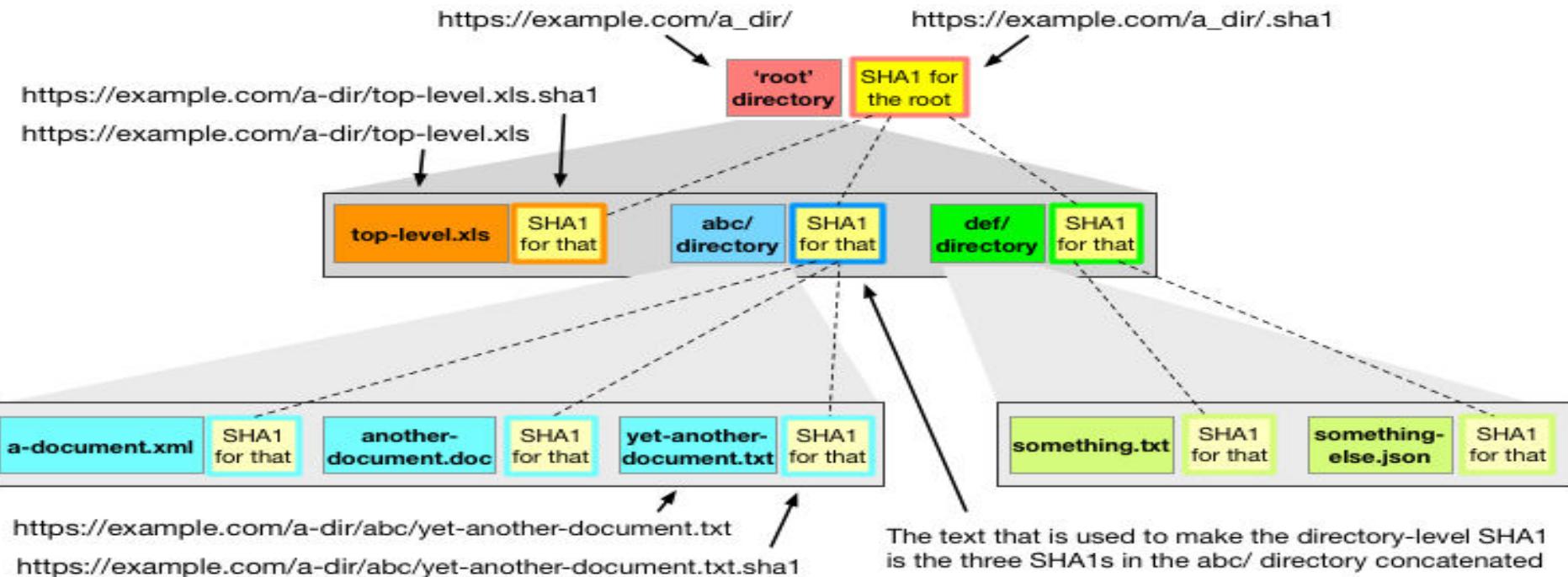
Smart Contract (Optional)

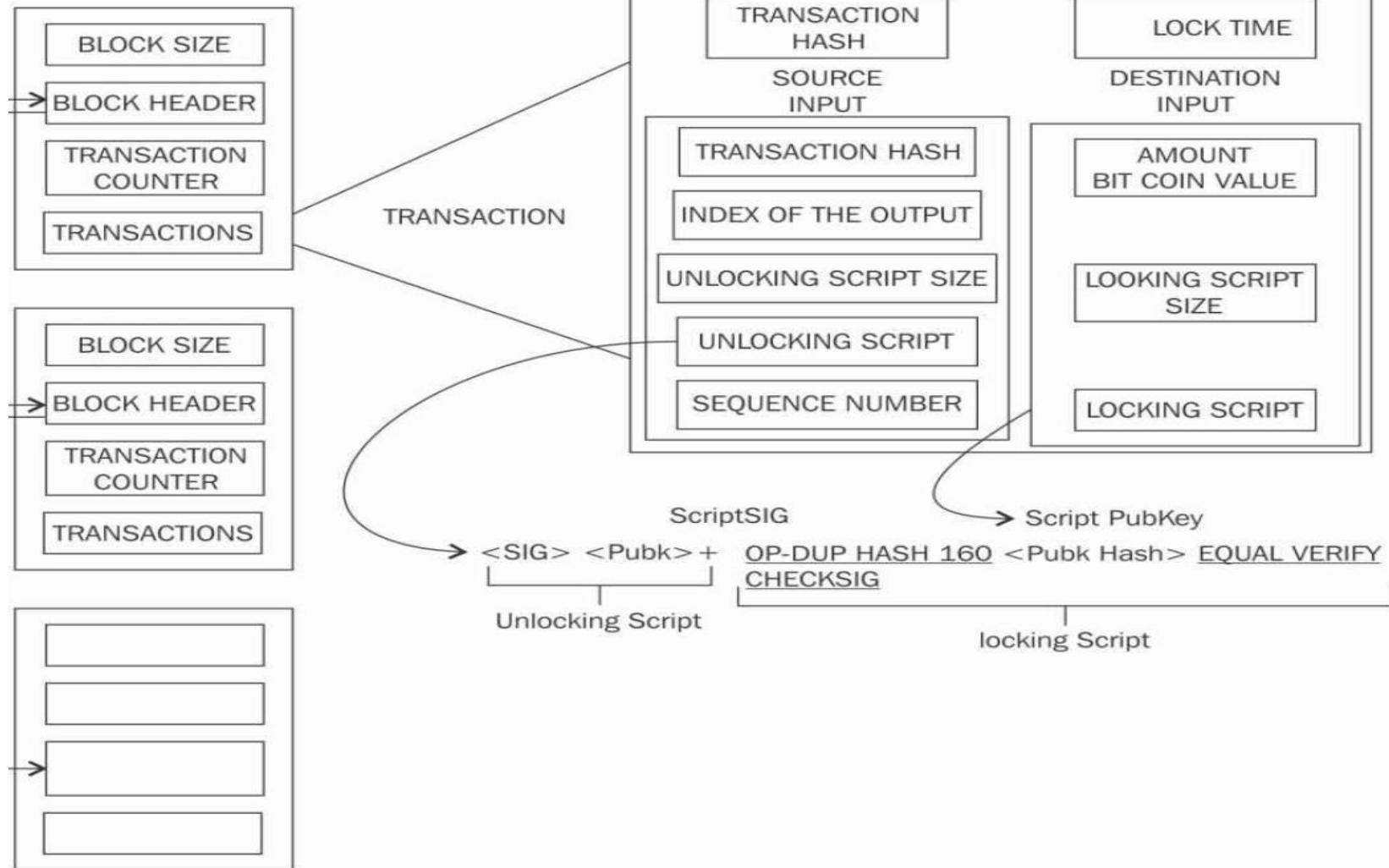
PoW

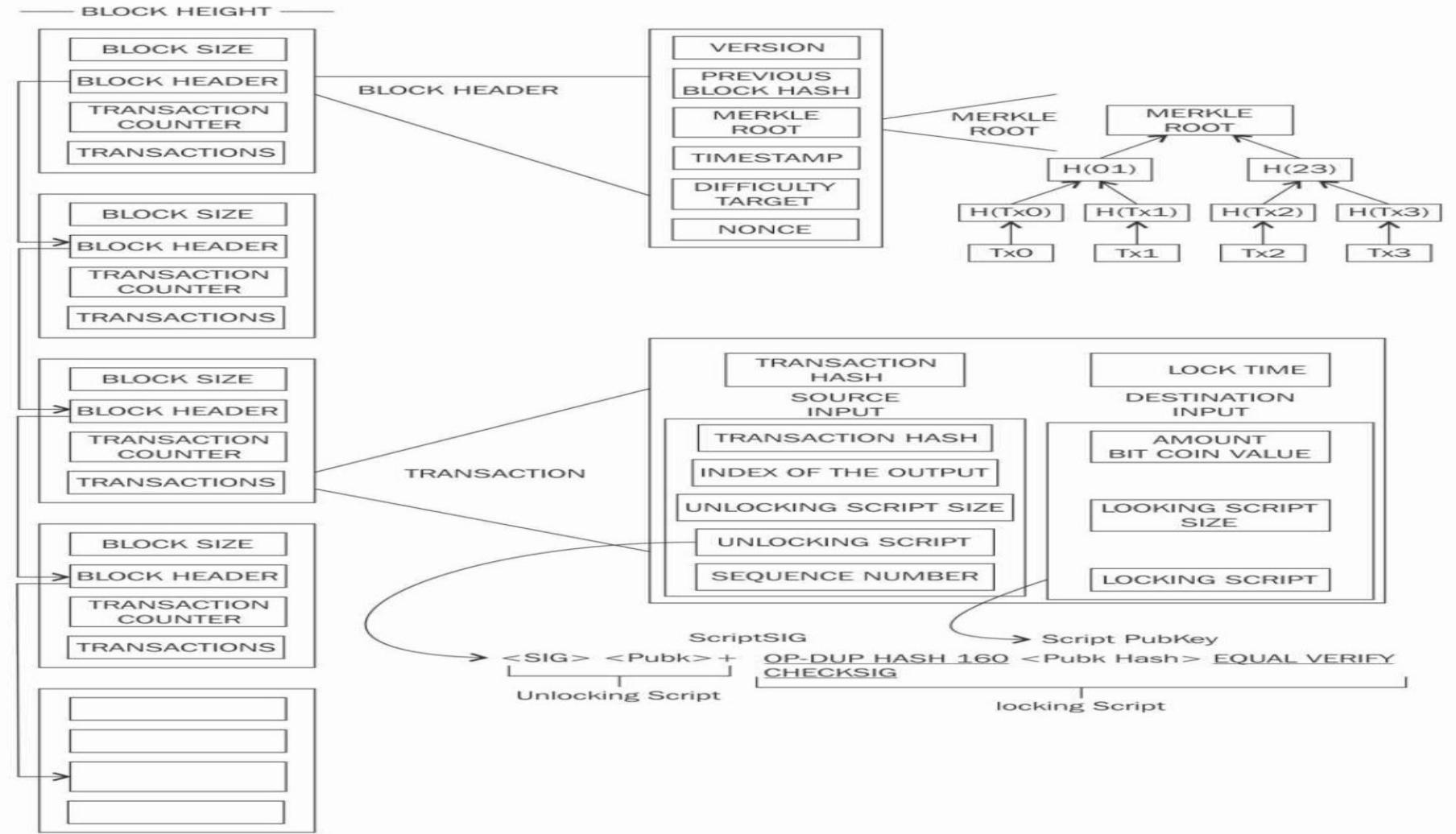
Blockchain Structure



Merkle Tree Structure







Genesis Block

This is the first block in the bitcoin blockchain. The genesis block was hardcoded in the bitcoin core software. It is in the chainparams.cpp file.

<https://github.com/bitcoin/bitcoin/blob/master/src/chainparams.cpp>

```
48 *      CtxOut(nValue=50.00000000, scriptPubKey=0x5F1DF16B2B704C8A578D0B)
49 * vMerkleTree: 4a5e1e
50 */
51 static CBlock CreateGenesisBlock(uint32_t nTime, uint32_t nNonce, uint32_t nBits, int32_t nVersion, const CAmount& genesisReward)
52 {
53     const char* pszTimestamp = "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks";
54     const CScript genesisOutputScript = CScript() << ParseHex("04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649
55     return CreateGenesisBlock(pszTimestamp, genesisOutputScript, nTime, nNonce, nBits, nVersion, genesisReward);
56 }
57 /**
58 */
```

Consensus Algorithms

Leased Proof of Stake, Proof of Capacity, Proof of
Importance, Proof of Authority, Proof of Burn

Introduction

- **Leased Proof of Stake (LPoS)** is an enhanced type of proof of stake consensus algorithm by which the Waves blockchain network aims to achieve the distributed consensus to secure the network.
- In a regular **Proof-of-Stake system**, each node that holds a certain amount of cryptocurrency is eligible to add the next block to the blockchain but in the **LPoS system**, on the Waves Platform, users can lease their balance to full nodes.
- With LPoS, the user will have the ability to **Lease WAVES** from the wallet to different contractors which can pay a percentage as a reward.
- The larger the amount that is leased to a full node, the higher the chances of that full node being selected to produce the next block.
- If that full node is selected to produce the next block, the leaser will then receive a percentage of the transaction fee that is collected by the full node.

Contd...

- This system allows anyone to participate in the Waves network maintenance. User can lease his waves through leasing on any computer or mobile device that has an internet browser

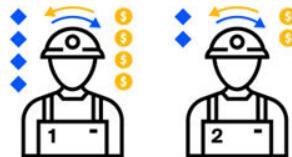
Blockchain Leasing for Proof of Stake

waves[♦]

Leased proof-of-stake (LPoS) allows Waves holders to profit by using their balances to secure the network – whilst retaining full control of their funds.



Miners run nodes to secure the Waves network and are rewarded with fees.



Their rewards depend on the amount of Waves they use to mine. The more Waves, the greater the rewards.



Regular users can lease their Waves to miners from the lite client, thereby increasing the Waves they can use...



...and sharing in the rewards, without giving up control of their Waves.

How It Works?

- In Waves, only full nodes can validate transactions. The platform's lite users can't hold a full node.
- Validators are picked from full node owners based on their stake.
- To participate in mining, a lite user can choose to shift to a full node or help a full node owner get selected as a validator by leasing them WAVES tokens.

Contd...

LPoS involves **two main types of transactions**:

- **Lease transaction** – it activates the leasing process. The token holder initiates a Lease transaction, specifies the node address (recipient address), and the number of funds to lease.
- **Lease cancel transaction** – stops the leasing process.
 - a. The minimum fee for a Lease Cancel transaction is 0.001 WAVES.
 - b. If the transaction sender is a dApp or a smart account, the minimum fee is increased by 0.004 WAVES.

Leasing benefits for the node owner

Nodes can use the leased tokens to generate blocks and get the mining reward. For that purpose, the generating balance of a node must be at least 1000 WAVES.

- In the [node configuration file](#), Use the enable parameter to start generating blocks on your node. By default, it's enabled, but if you disable it your node won't generate blocks.

Leasing benefits for the token holder

- LPoS allows the token holders to lease their tokens to the Waves nodes and earn a percentage of the payout as a reward.
- By using LPoS, leasers will be able to participate in the process of generating new blocks.
- When the user starts leasing the tokens, those leased tokens are locked and remained in the same address with the full control of their owner.

Rewards

- The node owner may send to leaser a part of rewards according to his conditions.
- The more transactions that are made on the network, the more rewards leasers get.
- These rewards mostly are in WAVES but also they can be in the form of different tokens with the unique Waves feature where different tokens can be accepted as a fee.

JSON Representation

```
{ "senderPublicKey": "BEPNBjo9Pi9hJ3hVtxpwyEfXCW3qWUNK5dMD7aFdiHsa",
  "fee": 100000,
  "type": 9,
  "version": 2,
  "leaseId": "BggRaeNCVmzuFGohzF4dQeYXSWr8i5zNSnGtdKc5eGrY",
  "sender": "3PMBXG13f89pq3WyJHHKX2m5zN6kt2CEkHQ",
  "feeAssetId": null,
  "chainId": 87,
  "proofs": [
    "3cqVVsaEDzBz367KTBFggMXEYJ2r3yLWd4Ha8r3GzmAFsm2CZ3GeNW22wqxfK4LNRFgsM5kCWRVhf6gu2Nv6zVqW" ],
  "id": "7siEtrJAvmVzM1WDX6v9RN4qliCtk7qQEeD5ZhE6955E",
  "lease": { "senderPublicKey": "BEPNBjo9Pi9hJ3hVtxpwyEfXCW3qWUNK5dMD7aFdiHsa",
    "amount": 406813214,
    "sender": "3PMBXG13f89pq3WyJHHKX2m5zN6kt2CEkHQ",
    "feeAssetId": null,
    "signature":
    "4u1sBnSrBiLE7DdE3Uj7R8kVqW3BFoBcwnuqDyssqxaYrF5hs6ABbw5hVT9S1AjQmgDr18euS5bYgedy7wdFUBjC",
    "proofs": [
      "4u1sBnSrBiLE7DdE3Uj7R8kVqW3BFoBcwnuqDyssqxaYrF5hs6ABbw5hVT9S1AjQmgDr18euS5bYgedy7wdFUBjC" ]}
```

```
"fee": 100000,  
"recipient": "3PMWRsRDy882VR2viKPrXhtjAQx7ygQcnea",  
"id": "BggRaeNCVmzuFGohzF4dQeYXSWr8i5zNSnGtdKc5eGrY",  
"type": 8,  
"version": 1,  
"timestamp": 1548192718874  
},  
"timestamp": 1548660629957,  
"height": 1370970  
}
```

- The lease structure does not need to be filled when sending a transaction, and it is not stored on the blockchain. The node returns this structure when providing transaction data via the REST API.
- The fields that are common to all types of transactions
- The LeaseCancelTransaction structure is used for transaction handling in smart contracts.

Features of LPoS

Balance Leasing

LPoS allows users to passively make a profit by leasing coins from their wallets or other cold storage to miners.

Fixed Tokens

Mining in LPoS doesn't add any tokens to the network. Tokens are fixed and leasable. Leased tokens are locked in leasers' accounts; therefore, they cannot be traded or transferred unless the leaser stops the leasing.

Contd...

Decentralized

In LPoS, however, rewards are linearly distributed based on the amount of stake; hence no mining pool required.

Transaction Fee As Rewards

Miners in LPoS receive transaction fees as a reward for processed blocks contrary to block rewards given in most blockchains.

Benefits of Leased Proof-of-Stake

Validate With Less Stake: In a PoS network, the validators are picked based on their stake. That may challenge the fairness of the system where some nodes are chosen repeatedly.

Earn with Fewer Tokens: LPoS allows minor token holders to earn by leasing their limited tokens to full node owners.

Control Over Funds: Leased tokens are locked in the leaser's wallet. They can neither be traded nor transferred.

Fewer Energy Consumptions: A lease transaction can be activated using a phone. A handful of nodes can now do the process that requires multiple nodes with high computing power with the support of phone users.

Higher Processing Speed: LPoS-based systems are fast and efficient since a few nodes are involved in validating a transaction at a given time. LPoS is an excellent alternative to Bitcoin's PoW that makes 4.6 transactions per second.

LPoS Weaknesses

- Malicious activities can be orchestrated on the LPoS, where members lease to a single full node. This node will always be in front of the validators' pool, giving it an advantage over other nodes.
- LPoS is still a new technology whose vulnerabilities are not yet fully exposed to help users make sound decisions.

Proof of Elapsed Time

- PoET was introduced by Intel with an intent to take over cryptographic puzzles involved in PoW mechanism by considering the fact that the CPU architecture and the quantity of mining hardware knows when and at what frequency does a miner win the block.
- PoET is a consensus mechanism algorithm that is often used on the **permissioned blockchain networks** to decide the mining rights or the block winners on the network.
- It is based on the idea of fairly distributing and expanding the odds for a bigger fraction of participants. And so, **every participating node** is asked to wait for a particular time to participate in the next mining process. The member with the **shortest hold-up time** is asked to offer a block.
- At the same time, every node also come up with their own waiting time, after which they go into sleep mode.

Contd...

- So, as soon as a node gets active and a block is available, that node is considered as the '**lucky winner**'. This node can then spread the information throughout the network, while maintaining the property of decentralization and receiving the reward.

Proof of Capacity

Another consensus algorithm is **Proof-of-Capacity (PoC)**, also known as **Proof-of-Space (PoSpace)**.

PoC works using the following approach:

- each miner calculates quite a large amount of data, which gets recorded on a disk subsystem of a node: hard drive, cloud storage or other. This initial dataset in PoC is called space.
- per each new block in the blockchain, the miner reads a small data set that equals 1/4096, which is approximately 0.024% of all stored data. Then it returns the result (deadline) as elapsed time since the last block was created, after which the miner can create a new block.
- the miner who received a minimum deadline time signs the block and receives a reward for transactions.

Proof of Importance

- This consensus algorithm is used by the **NEM blockchain platform**. The significance of each user in the NEM network is defined by a number of resources available on their balance and the number of transactions within their wallet.
- Unlike a more common PoS algorithm, which takes into account only user balance, PoI takes into consideration both resources amount and user activity in the blockchain network.
- This approach encourages users not only to keep funds in their accounts but also to use them extensively.

Proof of Authority

The PoA consensus algorithm is somewhat different from the rest of the algorithms since it doesn't require any mining, unlike PoW or PoS. In a PoAuthority-based blockchain network, all transactions and blocks are checked with approved accounts also known as validators. Transaction execution and block generation takes place automatically using just a computing power of the validator.

A positive aspect of this algorithm is:

- no need for mining leads to a significant reduction in maintenance costs.

A negative point of using this algorithm is:

- key persons of the algorithm are validators and that can lead to centralization. This only can make sense in a case, when a private network is deployed and accounts are completely, or as much as possible, trustworthy.

Proof of Burn

- **Proof of burn** is a method for distributed consensus and an alternative to Proof of Work and Proof of Stake. It can also be used for bootstrapping one cryptocurrency off of another.
- A miner sends coins to a random address of a generated hash. Spending the coins transferred to this address is basically impossible since there is a zero chance to find out its private key.
- By burning the coins, the miner gets a chance to find a PoB block and get a reward for it. Chances to mine that block increase as the number of burned coins grows.
- From an economics perspective, this process of burning coins can be thought of as buying a mining rig. Clearly, using such algorithm makes sense only in the later stages of a particular cryptocurrency existence, when there is actually something to burn.

Example of Proof of Burn

- For example, Slimcoin, a virtual currency network that uses POB, allows a miner to burn coins that not only gives them the right to compete for the next block but also gives them the chance to receive blocks during a longer time period, for at least a year.
- Essentially, Slimcoin's POB implementation combines three algorithms: POW, POS, and the core POB concept. The process of burning coins utilizes POW; the more coins one burns the more chances one has to mine, thus ensuring POS; and the whole ecosystem follows the POB concept.

Chapter 9. Hyperledger

Hyperledger is not a blockchain, but it is a project that was initiated by Linux foundation in December 2015 to advance blockchain technology. This project is a collaborative effort by its members to build an open source distributed ledger framework that can be used to develop and implement cross-industry blockchain applications and systems. The key focus is to build and run platforms that support global business transactions. The project also focuses on improving the reliability and performance of blockchain systems.

Projects under Hyperledger undergo various stages of development, starting from **proposal** to **incubation** and graduating to an **active** state. Projects can also be **deprecated** or in **End of Life** state where they are no longer actively developed. In order for a project to be able to move into incubation stage, it must have a fully working code base along with an active community of developers.

Projects

Currently, there are six projects under the Hyperledger umbrella: Fabric, Iroha, Sawtooth lake, blockchain explorer, Fabric chaintool, and Fabric SDK Py. Corda is the most recent addition that is expected to be added to the Hyperledger project. The Hyperledger project currently has 100 members and is very active with more than 120 contributors, with regular meet-ups and talks being organized around the globe.

A brief introduction of all these projects follows, after which we will provide more details around the design, architecture, and implementation of Fabric and Sawtooth lake.

Fabric

Fabric is a blockchain project that was proposed by IBM and **DAH (Digital Asset Holdings)**. This is intended to provide a foundation for the development of blockchain solutions and is based on pluggable architecture where various components, such as consensus algorithm, can be plugged into the system as required. It is available at <https://github.com/hyperledger/fabric>.

Sawtooth lake

Sawtooth lake is a blockchain project proposed by Intel in April 2016 with some key innovations focusing on **decoupling** of ledgers from transactions, flexible usage across multiple business areas using *transaction families*, and **pluggable consensus**. Decoupling can be explained more precisely by saying that the *transactions* are decoupled from the *consensus layer* by making use of a new concept called *Transaction families*. Instead of transactions being individually coupled with the ledger, transaction families are used, which allows for more flexibility, rich semantics and unrestricted design of business logic. Transactions follow the patterns and structures defined in the transaction families. Intel has also introduced a novel consensus algorithm abbreviated as PoET, proof of elapsed time, which makes use of **Intel Software Guard Extensions (Intel's SGX)** architecture's **trusted execution environment (TEE)** in order to provide a safe and random leader election process. It also supports permissioned and permission-less setups. This project is available at <https://github.com/hyperledger/sawtooth-core>.

Iroha

Iroha was proposed by Soramitsu, Hitachi, NTT Data, and Colu in September 2016. Iroha is aiming to build a library of reusable components that users can choose to run on their Hyperledger-based distributed ledgers. Iroha's main goal is to complement other Hyperledger projects by providing reusable components written in C++ with an emphasis on mobile development. This project has also proposed a novel consensus algorithm called Sumeragi, which is a chain based Byzantine fault tolerant consensus algorithm. Iroha is available at <https://github.com/hyperledger/iroha>. Various libraries have been proposed and are being worked on by Iroha, including but not limited to a digital signature library (ed25519), an SHA-3 hashing library, a transaction serialization library, a P2P library, an API server library, an iOS library, an Android library, and a JavaScript library.

Blockchain explorer

This project aims to build a blockchain explorer for Hyperledger that can be used to view and query the transactions, blocks, and associated data from the blockchain. It also provides network information and the ability to interact with chain code.

Currently there are two other projects that are in incubation: Fabric chaintool, and Fabric SDK Py. These projects are aimed at supporting Hyperledger Fabric.

Fabric chaintool

Hyperledger chaincode compiler is being developed to support Fabric chaincode development. The aim is to build a tool that reads in a high-level Google protocol buffer structure and produces a chaincode. Additionally, it packages the chaincode so that it can be deployed directly. It is envisaged that this tool will help developers in various stages of development, such as compiling, testing, packaging, and deployment. It is available at <https://github.com/hyperledger/fabric-chaintool>.

Fabric SDK Py

The aim of this project is to build a python based SDK library that can be used to interact with the blockchain (Fabric). It is available at <https://github.com/hyperledger/fabric-sdk-py>.

Corda

Corda is the latest project that has been contributed by R3 to the Hyperledger project. It was open sourced on November 30, 2016. Corda is heavily oriented towards the financial services industry and has been developed in collaboration with major banks and organizations in the financial industry. At the time of writing it is not yet in incubation under the Hyperledger project. Technically, Corda is not a blockchain but has key features similar to those of a blockchain, such as consensus, validity, uniqueness, immutability, and authentication.

In the following sections of this chapter, Fabric (IBM) and Sawtooth lake (Intel) and Corda (R3) will be discussed in more detail.

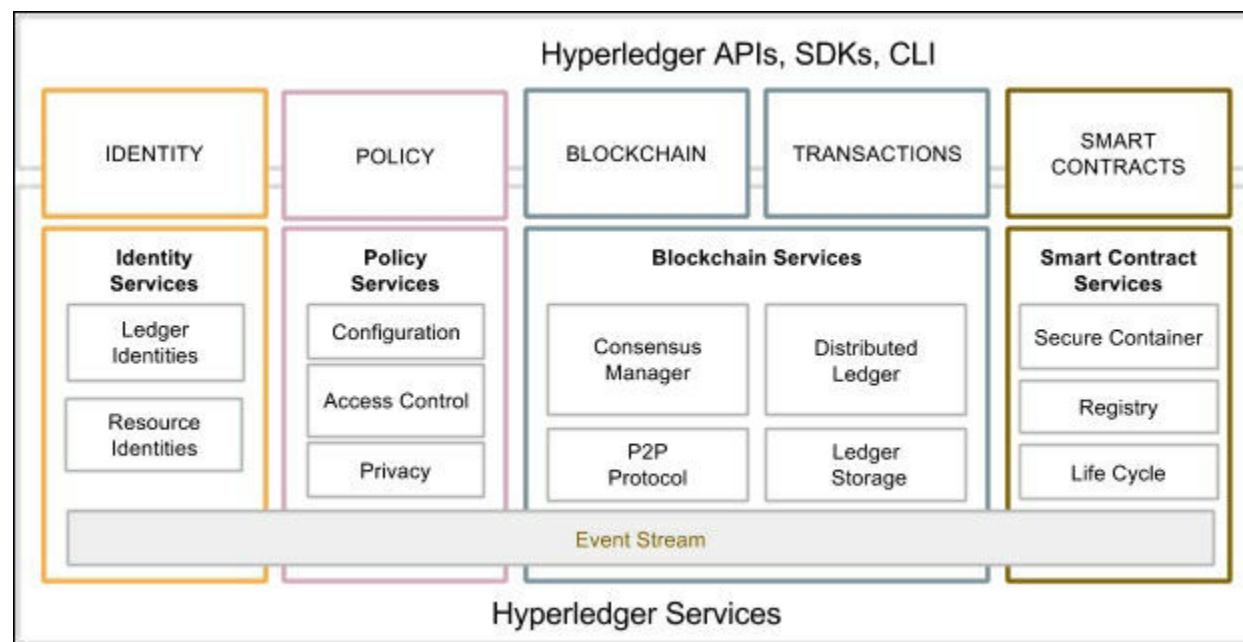
Hyperledger as a protocol

Hyperledger is aiming to build a new blockchain platform that is driven by industry use cases. As there have been number of contributions made to the Hyperledger project by the community, Hyperledger blockchain platform is evolving into a protocol for business transactions. Hyperledger is also evolving into a specification that can be used as a reference to build blockchain platforms as compared to earlier blockchain solutions that address only a specific type of industry or requirement. In the following section, a reference architecture is presented that has been published by the Hyperledger project. As this work is under continuous and rigorous development some changes are expected in this, but core services are expected to remain unchanged.

Reference architecture

Hyperledger has published a white paper with reference architecture that can serve as a guideline to build permissioned distributed ledgers. The reference architecture consists of two main components: Hyperledger services and Hyperledger APIs, SDKs, and CLI. Hyperledger services provide various services such as identity services, policy services, blockchain services, and smart contract services. On the other hand, Hyperledger APIs, SDKs, and CLIs provide an interface into blockchain services via appropriate application programming interfaces, software development kits, or command line interfaces. Moreover, an event stream, which is basically a gRPC channel, runs across all services. It can receive and send events. Events are either pre-defined or custom. Validating peers or chaincode can emit events to which external application can respond or listen to.

The reference architecture that has been published in the Hyperledger white paper at the time of writing is shown in the following diagram. Hyperledger is a rapidly changing and evolving project, and the architecture shown here is expected to change somewhat.



Hyperledger architecture, as proposed in the latest draft V2.0.0 of Hyperledger white paper. (Source: Hyperledger white paper)

Requirements

There are certain requirements of a blockchain service. The reference architecture is driven by the needs and requirements raised by the participants of the Hyperledger project and after studying the industry use cases. There are several categories of requirements that have been deduced from the study of industrial use cases and are discussed in the following sections.

Modular approach

The main requirement of Hyperledger is a modular structure. It is expected that, as a cross-industry fabric (blockchain), it will be used in many business scenarios. As such, functions related to storage, policy, chaincode, access control, consensus and many other blockchain services should be pluggable. The modules should be plug and play and users should be able to easily remove and add a different module that meets the requirements of the business.

For example, if a business blockchain needs to be run only between already trusted parties and performs very basic business operations, then perhaps there is no need to have advanced cryptographic support for confidentiality and privacy, and therefore users should be able to remove that functionality (module) or replace that with a more appropriate module that suits their needs. Similarly, if users need to run a cross-industry blockchain, then confidentiality and privacy can be of paramount importance. In this case, users should be able to plug an advanced cryptographic and access control mechanism (module) into the blockchain (fabric).

Privacy and confidentiality

Privacy and confidentiality of transactions and contracts is of utmost importance in a business blockchain. As such, Hyperledger's vision is to provide a wide range of cryptographic protocols and algorithms and it is expected that users will be able to choose appropriate modules according to their business requirements. The fabric should be able to handle complex cryptographic algorithms without compromising performance.

Identity

In order to provide privacy and confidentiality services, a flexible PKI model that can be used to handle the access control functionality is also required. The strength and type of cryptographic mechanisms is also expected to vary according to the needs and requirements of the users. In certain scenarios it might be required for a user to hide their identity, and as such the Hyperledger is expected to provide this functionality.

Auditability

Auditability is another requirement of a Hyperledger Fabric. It is expected that an immutable audit trail of all identities, related operations and any changes is kept.

Interoperability

Currently there are many blockchain solutions available, but they cannot communicate with each other and this can be a limiting factor in the growth of a blockchain based global business ecosystem. It is envisaged that many blockchain networks will operate in the business world for specific needs, but it is important that they are able to communicate with each other. There should be a common set of standards that all blockchains can follow in order to allow communication between different ledgers. It is expected that a protocol will be developed that will allow the exchange of information between many Fabrics.

Portability

The portability requirement is concerned with the ability to run across multiple platforms and environments without the need to change anything at code level. Hyperledger is envisaged to be portable, not only at infrastructure level but also at code, libraries, and API levels so that it can support uniform development across various implementations of Hyperledger.

Fabric

In order to understand various projects under incubation in Hyperledger project, it is important to understand the foundations of Hyperledger first. A few terminologies that are specific to Hyperledger needs some clarification before readers are introduced to more in-depth material. First there is the concept of Fabric.

Fabric can be defined as a collection of components providing a foundation layer that can be used to deliver a blockchain network. There are various types and capabilities of a fabric network, but all fabrics share common attributes such as immutability and are consensus driven. Some fabrics can provide modular approach towards building blockchain networks. In this case the blockchain network can have multiple pluggable modules to perform various function on the network. For example, consensus algorithms can be a pluggable module in a blockchain network where, depending on the requirements of the network, an appropriate consensus algorithm can be chosen and *plugged* into the network. The modules can be based on some particular specification of the fabric and can include APIs, access control, and various other components. Fabrics can also be designed either to be private or public and can allow the creation of multiple business networks. As an example, bitcoin is an application that runs on top of its fabric (blockchain network). As discussed earlier, blockchain can either be permissioned or permission-less and the same is true for fabric in Hyperledger terminology.

Fabric is also the name given to the code contribution made by IBM to the Hyperledger foundation and is formally called Hyperledger Fabric. IBM also offers blockchain as a service (IBM Blockchain) via its Bluemix cloud service.

Hyperledger Fabric

Fabric is the contribution originally made by IBM to the Hyperledger project. The aim of this contribution is to enable a modular, open and flexible approach towards building blockchain networks. Various functions in the fabric are pluggable, and it also allows use of any language to develop smart contracts. This is possible because it is based on container technology which can host any language. Chaincode (smart contract) is sandboxed into a secure container which includes a secure operating system, chaincode language, runtime environment and SDKs for Go, Java, and Node.js. Other languages can be supported too if required. Smart contracts are called chaincode in the Fabric. This is a very powerful feature compared to domain specific languages in Ethereum, or the very limited scripted language in bitcoin. It is a permissioned network that aims to address issues such as scalability, privacy, and confidentiality. The key idea behind this is modular technology, which would allow for flexibility in design and implementation. This can then result in achieving scalability, privacy and other desired attributes. Transactions in fabric are private, confidential and anonymous for general users, but they can still be traced and linked to the users by authorized auditors. As a permissioned network, all participants are required to be registered with the membership services in order to access the blockchain network. This ledger also provided auditability functionality in order to meet the regulatory and compliance needs.

Fabric architecture

The Fabric is logically organized into three main categories based on the type of service provided. These include membership services, blockchain services, and chaincode services. In the following section, all these categories and associated components are discussed in detail. The current stable version of Hyperledger Fabric is v0.6, however the latest version v1.0 is available but is not yet stable. In version 1.0, many architectural changes have been made, and in later sections of this chapter some changes that have been made in version 1.0 will also be discussed.

Membership services

These services are used to provide access control capability for the users of the fabric network. The following list shows the functions that membership services perform:

1. User identity validation.
2. User registration.
3. Assign appropriate permissions to the users depending on their roles.

Membership services makes use of **Public Key Infrastructure (PKI)** in order to support identity management and authorization operations. Membership services are made up of various components:

- **Registration authority (RA):** A service that authenticates the users and assesses the identity of the fabric participants for issuance of certificates.
- **Enrolment certificate authority:** **Enrolment certificates (Ecerts)** are long term certificates issued by ECA to registered participants in order to provide identification to the entities participating on the network.
- **Transaction certificate authority:** In order to send transactions on the networks, participants are required to hold a transaction certificate. TCA is responsible for issuing transaction certificates to holders of Enrolment certificates and is derived from Ecerts.
- **TLS certificate authority:** In order to secure the network level communication between nodes on the Fabric, TLS certificates are used. TLS certificate authority issues TLS certificates in order to ensure security of the messages being passed between various systems on the blockchain network.

Blockchain services

Blockchain services are at the core of the Hyperledger Fabric. Components within this category are as follows.

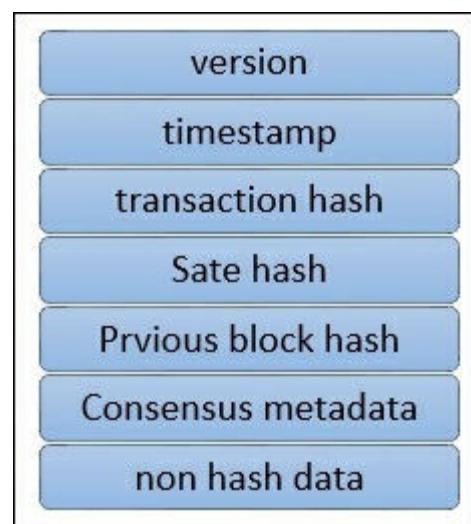
Consensus manager

Consensus manager is responsible for providing the interface to the consensus algorithm. This serves as an adapter that receives the transaction from other Hyperledger entities and executes them under criteria according to the type of algorithm chosen. Consensus is pluggable and currently there are three types of consensus algorithm available in Fabric, namely the batch PBFT protocol, SIEVE

algorithm, and NOOPS.

Distributed ledger

Blockchain and world state are two main elements of the distributed ledger. Blockchain is simply a linked list of blocks (as introduced in earlier chapters) and world ledger is a key-value database. This database is used by smart contracts to store relevant states during execution by the transactions. The blockchain consists of blocks that contain transactions. These transactions contain chaincode, which runs transactions that can result in updating the world state. Each node saves the world state on disk in RocksDB. The following diagram shows a typical block in the Hyperledger Fabric with the relevant fields:



Block structure

The fields shown in the preceding diagram are as follows:

- **Version:** Used for keeping track of changes in the protocol.
- **Timestamp:** Timestamp in UTC epoch time, updated by block proposer.
- **Transaction hash:** This field contains the Merkle root hash of the transactions in the block.
- **State hash:** This is the Merkle root hash of the world state.
- **Previous hash:** This is the previous block's hash, which is calculated after serializing the block message and then creating the message digest by applying the SHA3 SHAKE256 algorithm.
- **Consensus metadata:** This is an optional field that can be used by the consensus protocol to provide some relevant information about the consensus.
- **Non-Hash data:** This is some metadata that is stored with the block but is not hashed. This feature makes it possible to have different data on different peers. It also provides the ability to discard data without any impact on the blockchain.

Peer to Peer protocol

P2P protocol in the Hyperledger Fabric is built using **google RPC (gRPC)**. It uses protocol buffers to define the structure of the messages.

Messages are passed between nodes in order to perform various functions. There are four main types of messages in Hyperledger Fabric: Discovery, transaction, synchronization and consensus. Discovery messages are exchanged between nodes when starting up in order to discover other peers on the network.

Transaction messages can be divided into two types: Deployment transactions and Invocation transactions. The former is used to deploy new chaincode to the ledger, and the latter is used to call functions from the smart contract. Transactions can be public, confidential, and confidential chaincode transactions. Public transactions are open and available to all participants. Confidential transactions are allowed to be queried only by transaction owners and participants. Confidential chaincode transactions have encrypted chaincode and can only be decrypted by validating nodes. Validating nodes run consensus, validate the transactions and maintain the blockchain. Non-validating nodes on the other hand, provide transaction verification, stream server, and REST services. They also act as a proxy between the transactors and the validating nodes. Synchronization messages are used by peers to keep the blockchain updated and in sync with other nodes. Consensus messages are used in consensus management and broadcasting payloads to validating peers. These are generated internally by the consensus framework.

Ledger storage

In order to save the state of the ledger, RocksDB is used, and it is stored at each peer. RocksDB is a high performance database available at <http://rocksdb.org/>.

Chaincode services

These services allow the creation of secure containers that are used to execute the chaincode. Components in this category are as follows:

- **Secure container:** Chaincode is deployed in Docker containers that provide a locked down sandboxed environment for smart contract execution. Currently Golang is supported as the main smart contract language, but any other main stream language can be added and enabled if required.
- **Secure registry:** This provides a record of all images containing smart contracts.

Events

Events on the blockchain can be triggered by validator nodes and smart contracts. External applications can listen to these events and react to them if required via event adapters. They are similar to the concept of events introduced in solidity in the last chapter.

APIs and CLIs

An application programming interface provides an interface into the fabric by exposing various REST APIs. Additionally, command line interfaces that provide a subset of REST APIs and allow for quick testing and limited interaction with the blockchain are also available.

Components of the Fabric

There are various components that can be part of the blockchain. These components include but are not limited to the ledger, chaincode, consensus mechanism, access control, events, system monitoring and management, wallets and system integration components.

Peers or nodes

There are two main types of peers that can be run on a fabric network: Validating and non-validating. Simply put, a validating node runs consensus, creates and validates a transaction, and contributes towards updating the ledger and maintaining the chaincode.

A non-validating peer does not execute transactions and only constructs transactions that are then forwarded to validating nodes.

Both nodes manage and maintain user certificates that have been issued by membership services.

Applications on blockchain

A typical application on Fabric is simply composed of a user interface, usually written in JavaScript/HTML, that interacts with the backend chaincode (smart contract) stored on the ledger via an API layer.

HTML / JS

USER INTERFACE
FRONTEND

A P I

BLOCK

CHAIN
CODE

B

B

B

B

B

LEDGER

Typical blockchain application

Hyperledger provides various APIs and command line interfaces to enable interaction with the ledger. These APIs include interfaces for identity, transactions, chaincode, ledger, network, storage, and events.

Chaincode implementation

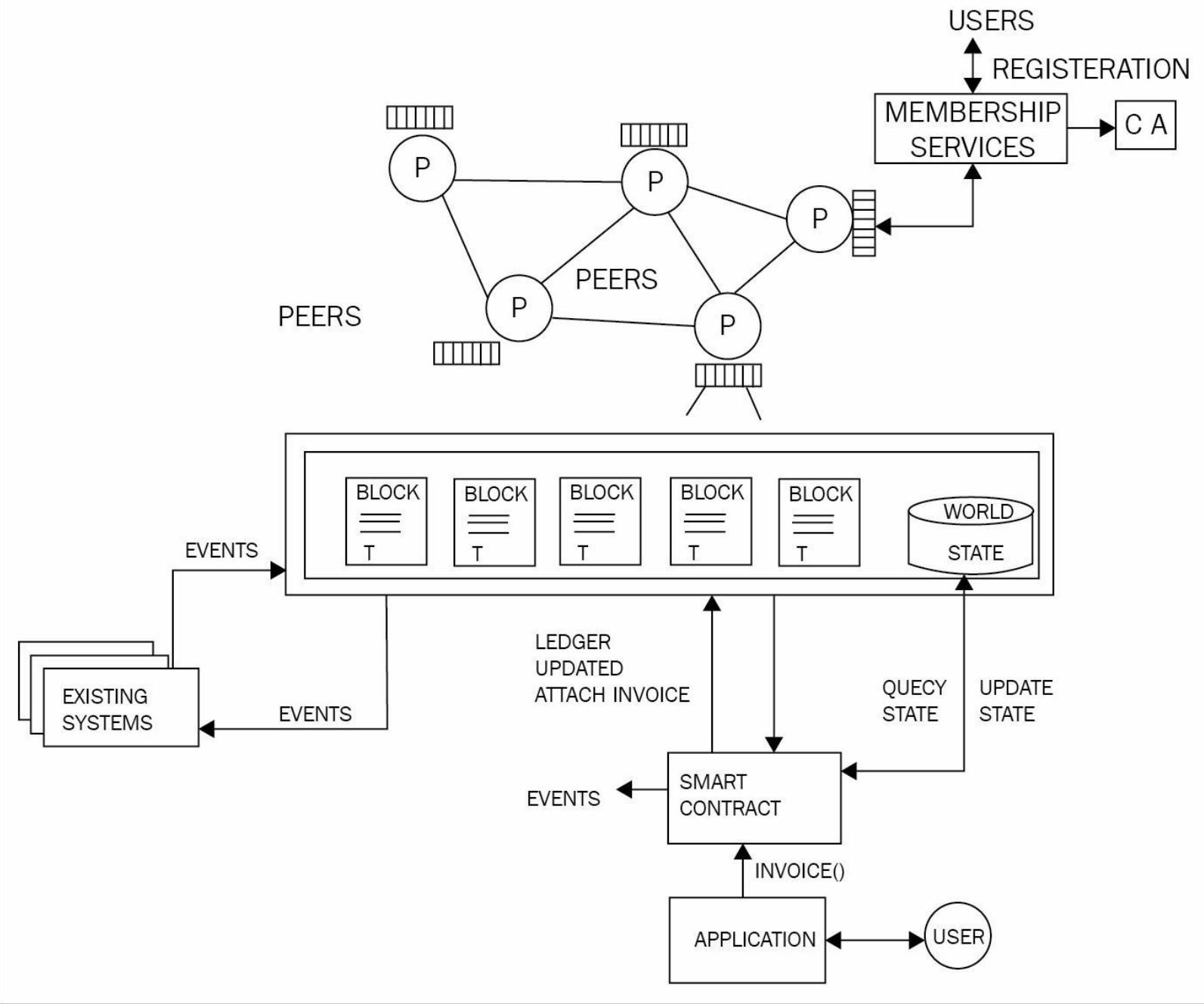
Chaincode is usually written in Golang or Java. Chaincode can be public, confidential or access controlled. These codes serve as a smart contract that users can interact with via APIs. Users can call functions in the chaincode that result in a state change, and consequently updates the ledger. There are also functions that are only used to query the ledger and do not result in any state change.

Chaincode implementation is performed by first creating the chaincode shim interface in the code. It

can either be in Java or Golang code. The following four functions are required in order to implement the chaincode:

- `Init()`: This function is invoked when chaincode is deployed onto the ledger. This initializes the chaincode and results in making a state change, which accordingly updates the ledger.
- `Invoke()`: This function is used when contracts are executed. It takes a function name as parameters along with an array of arguments. This function results in a state change and writes to the ledger.
- `Query()`: This function is used to query the current state of a deployed chaincode. This function does not make any changes to the ledger.
- `Main()`: This function is executed when a peer deploys its own copy of the chaincode. The chaincode is registered with the peer using this function.

The following diagram illustrates the general overview of Hyperledger Fabric:



High-level overview of Hyperledger Fabric

Application model

Any blockchain application for Hyperledger Fabric follows MVC-B architecture. This is based on the popular MVC design pattern. Components in this model are Model, View, Control, and Blockchain:

- **View logic**: This is concerned with the user interface. It can be a desktop, web application or mobile frontend.
- **Control logic**: This is the orchestrator between user interface, data model, and APIs.
- **Data model**: This model is used to manage the off-chain data.
- **Blockchain logic**: This is used to manage the blockchain via the controller and the data model

via transactions.

Due to the fact that Hyperledger current release v0.6 is under heavy refactoring to build V1.0, no practical exercises have been introduced in this section.

It is expected that by the time this book is published, the information regarding practical setup of Hyperledger fabric will be outdated already. As such, readers are encouraged to keep an eye on the updates at <https://hyperledgerfabric.readthedocs.io/en/latest/>.

Moreover, the IBM Bluemix service offers sample applications for blockchain under its blockchain as a service offering. It is available at https://console.ng.bluemix.net/docs/services/blockchain/ibmBlockchain_tutorials.html. This service allows users to create their own blockchain networks in an easy to use environment.

Sawtooth lake

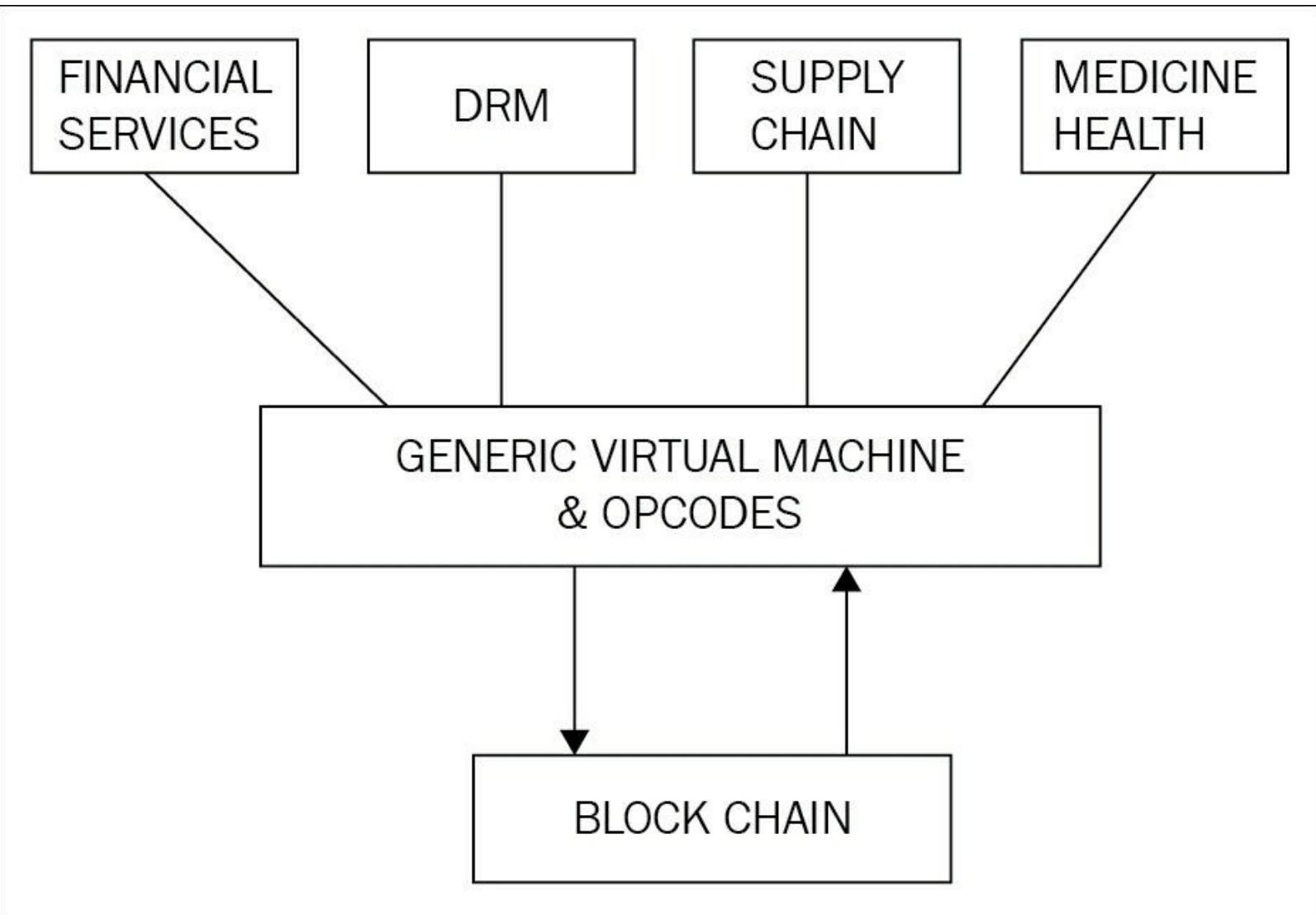
Sawtooth lake can run in both permissioned and non-permissioned modes. It is a distributed ledger that proposes two novel concepts: The first is the introduction of a new consensus algorithm called **Proof of Elapsed Time (PoET)**; and the second is the idea of transaction families. A brief description of these novel proposals is given in the following section.

PoET

PoET is a novel consensus algorithm that allows a node to be selected randomly based on the time that the node has waited before proposing a block. This is in contrast to other leader election and lottery based proof of work algorithms, where an enormous amount of electricity and computer resources are used in order to be elected as a block proposer, for example in the case of bitcoin. PoET is a type of Proof of Work algorithm but, instead of spending computer resources, it uses a trusted computing model to provide a mechanism to fulfill Proof of Work requirements. PoET makes use of Intel's SGX architecture to provide a trusted execution environment to ensure randomness and cryptographic security of the process. It should be noted that the current implementation of Sawtooth lake does not require real hardware SGX based TEE, as it is simulated for experimental purposes only and as such should not be used in production environments.

Transaction families

A traditional smart contract paradigm provides a solution that is based on a general purpose instruction set for all domains. For example, in the case of Ethereum, a set of opcodes has been developed for the **Ethereum virtual machine (EVM)** that can be used to build smart contracts to address any type of requirements for any industry. Whilst this model has its merits, it is becoming clear that this approach is not very secure as it provides a single interface into the ledger with a powerful and expressive language, which potentially offers a larger attack surface for malicious code. This complexity and generic virtual machine paradigm has resulted in several vulnerabilities that were found and exploited recently by hackers. A recent example is the DAO hack and further **Denial of Services (DoS)** attacks that exploited limitations in some EVM opcodes. A model shown in the following figure describes the traditional smart contract model, where a generic virtual machine has been used to provide the interface into the blockchain for all domains:

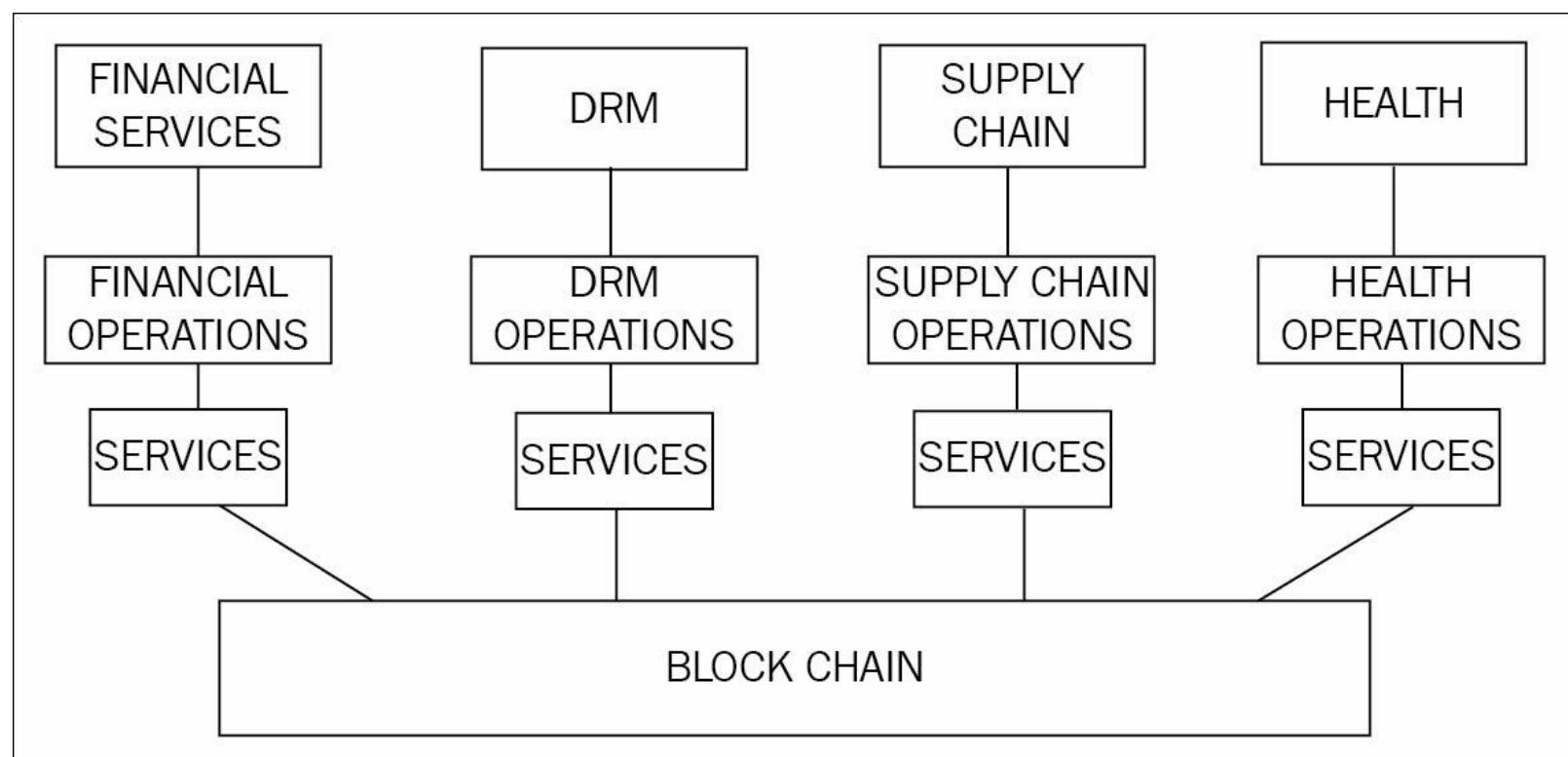


Traditional smart contract paradigm

In order to address this issue, Sawtooth lake has proposed the idea of transaction families. A

transaction family is created by decomposing the logic layer into a *set of rules* and a *composition layer* for a specific domain. The key idea is that business logic is composed within *transaction families*, which provides a more secure and powerful way to build smart contracts. Transaction families contain the domain-specific rules and another layer that allows for creating transactions for that domain. Another way of looking at it is that transaction families are a combination of a data model and a transaction language that implements a logic layer for a specific domain. The data model represents the current state of the blockchain (ledger) whereas the transaction language modifies the state of the ledger. It is expected that users will build their own transaction families according to their business requirements.

The following diagram represents this model, where each specific domain, like financial services, **digital rights management (DRM)**, supply chain, and the health industry, has its own logic layer comprised of operations and services specific to that domain. This makes the logic layer both restrictive and powerful at the same time. Transaction families ensure that operations related to only the required domain are present in the control logic, thus removing the possibility of executing needless, arbitrary and potentially harmful operations.



Sawtooth (transaction families) smart contract paradigm

Intel has provided three transaction families with Sawtooth: Endpoint registry, Integerkey, and MarketPlace.

1. **Endpoint registry** is used for registering ledger services.
2. **Integerkey** is used for testing deployed ledgers.
3. **MarketPlace** is used for selling, buying and trading operations and services.

Sawtooth_bond has been developed as a proof of concept to demonstrate a bond trading platform. It is available at <https://github.com/hyperledger/sawtooth-core/tree/master/extensions/bond>.

Consensus in Sawtooth

Sawtooth has two types of consensus mechanisms based on the choice of network. PoET, as discussed previously, is a trusted executed environment based lottery function that elects a leader randomly based on the time a node has waited for block proposal. There is another consensus type called quorum voting, which is an adaptation of consensus protocols built by Ripple and Stellar. This consensus algorithm allows instant transaction finality, which is usually desirable in permissioned networks.

Development environment

In this section, a quick introduction is given on how to set up a development environment for Sawtooth lake. There are few pre-requisites that are required in order to set up the development environment. Examples in this section assume a running Ubuntu system and the following:

1. vagrant, at least version 1.9.0, available at <https://www.vagrantup.com/downloads.html>.
2. Virtual box, at least 5.0.10 r104061, available at <https://www.virtualbox.org/wiki/Downloads>.

Once both of the above pre-requisites are downloaded and installed successfully, the next step is to clone the repository.

```
$ git clone https://github.com/IntelLedger/sawtooth-core.git
```

This will produce an output similar to the one shown in the following screenshot:

```
drequinox@drequinox-OP7010:~/project$ git clone https://github.com/IntelLedger/sawtooth-core.git
Cloning into 'sawtooth-core'...
remote: Counting objects: 12527, done.
remote: Compressing objects: 100% (964/964), done.
remote: Total 12527 (delta 452), reused 0 (delta 0), pack-reused 11515
Receiving objects: 100% (12527/12527), 9.26 MiB | 1.76 MiB/s, done.
Resolving deltas: 100% (8131/8131), done.
Checking connectivity... done.
```

GitHub Sawtooth clone

Once Sawtooth is cloned correctly, the next step is to start up the environment. First, run the following command to change the directory to the correct location and then start the `vagrant` box:

```
$ cd sawtooth-core/tools
$ vagrant up
```

This will produce an output similar to the following screenshot:

```
drequinox@drequinox-OP7010:~/project/sawtooth-core/tools$ vagrant up
Could not determine vagrant user.
VAGRANT_BOX = ubuntu/xenial64
VAGRANT_FORWARD_PORTS = true
VAGRANT_MEMORY = 2048
VAGRANT_CPUS = 2
Proxyconf plugin not found
Install: vagrant plugin install vagrant-proxyconf
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'ubuntu/xenial64' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'ubuntu/xenial64'
    default: URL: https://atlas.hashicorp.com/ubuntu/xenial64
==> default: Adding box 'ubuntu/xenial64' (v20161221.0.0) for provider: virtualbox
    default: Downloading: https://atlas.hashicorp.com/ubuntu/boxes/xenial64/versions/20161221.0.0/providers/virtualbox.bo
x   default: Progress: 1% (Rate: 1709k/s, Estimated time remaining: 0:04:04)
```

Vagrant up command

If at any point Vagrant needs to be stopped, the following command can be used:

```
$ vagrant halt
```

Or

```
$ vagrant destroy
```

Halt will stop the `vagrant` machine, whereas `destroy` will stop and delete `vagrant` machines.

Finally, the transaction validator can be started by using the following commands. First `ssh` into the `vagrant` Sawtooth box.

```
$ vagrant ssh
```

When the `vagrant` prompt is available, run the following commands.

First build the `sawtooth lake` core using following command:

```
$ /project/sawtooth-core/bin/build_all
```

When the build has completed successfully, in order to run transaction validator issue the following commands:

```
$ /project/sawtooth-core/docs/source/tutorial/genesis.sh
```

This will create the genesis block and clear any existing data files and keys. This should show an output similar to the following screenshot:

```
ubuntu@ubuntu-xenial:/project/sawtooth-core$ /project/sawtooth-core/docs/source/tutorial/genesis.sh
writing file: /home/ubuntu/sawtooth/keys/base000.wif
writing file: /home/ubuntu/sawtooth/keys/base000.addr
```

Genesis block and keys generation

The next step is to run the transaction validator, and change the directory as shown follows:

```
$ cd /project/saw-toothcore
```

Run the transaction validator:

```
$ ./bin/txnvalidator -v -F ledger.transaction.integer_key --config  
/home/ubuntu/sawtooth/v0.json
```

```
ubuntu@ubuntu-xenial:/project/sawtooth-core$ ./bin/txnvalidator -v -F ledger.transaction.integer_key --config /home/ubuntu/sawtooth/v0.json  
[22:08:22 INFO  validator_cli] validator started with arguments: ['./bin/txnvalidator', '-v', '-F', 'ledger.transaction.integer_key', '--config', '/home/ubuntu/sawtooth/v0.json']  
[22:08:22 INFO  validator_cli] read signing key from /home/ubuntu/sawtooth/keys/base000.wif  
[22:08:24 WARNING validator_cli] validator pid is 10937  
[22:08:24 INFO  gossip_core] listening on IPv4Address(UDP, '0.0.0.0', 33713)  
[22:08:24 INFO  global_store_manager] create blockstore from file /home/ubuntu/sawtooth/data/base000_state.dbm with flag c  
[22:08:24 INFO  validator] set administration node to None  
[22:08:24 INFO  validator] starting ledger base000 with id 1K5RNedZ at network address ('127.0.0.1', 33713)  
[22:08:24 INFO  web_api] listen for HTTP requests on (ip='localhost', port=8800)  
[22:08:24 INFO  validator_cli] adding transaction family: ledger.transaction.integer_key  
[22:08:24 INFO  journal_core] restore ledger state from persistence  
[22:08:24 INFO  global_store_manager] add block 60af3ec894fa1cb0 to the queue for loading  
[22:08:24 INFO  global_store_manager] load block 60af3ec894fa1cb0 from storage  
[22:08:24 INFO  journal_core] commit head: 60af3ec894fa1cb0  
[22:08:26 INFO  validator] ledger connections using RandomWalk topology  
[22:08:26 INFO  random_walk] initiate random walk topology update  
[22:08:29 INFO  validator] ledger initialization complete  
[22:08:29 INFO  journal_core] process initial transactions and blocks  
[22:08:29 INFO  validator] register endpoint 1K5RNedZ with name base000  
[22:08:29 INFO  journal_core] build transaction block to extend 60af3ec8 with 1 transactions  
[22:08:29 INFO  wait_timer] wait timer created; TIMER, 5.00, 33.69, HE2DQNJWGI2DCNJQ
```

Running transaction validator

The validator node can be stopped by pressing *Ctrl + C*. Once the validator is up and running, various clients can be started up in another terminal window to communicate with the transaction validator and submit transactions.

For example, in the following screenshot the market client is started up to communicate with the transaction validator. Note that keys under `/keys/mkt.wif` are created by using the following command:

```
./bin/sawtooth keygen --key-dir validator/keys mkt
```

This demonstration is just a basic example derived from Sawtooth lake documentation. However, development using Sawtooth lake is quite an involved process and a full chapter could be dedicated to that.

```
ubuntu@ubuntu-xenial:/project/sawtooth-core$ ./bin/mktclient --name market --keyfile validator/keys/mkt.wif
//UNKNOWN> help

Documented commands (type help <topic>):
=====
EOF      dump      exit      liability    selloffer  tokenstore
account   echo      help      map        session    waitforcommit
asset     exchange  holding  offers     sleep
assettype exchangeoffer holdings participant state

Miscellaneous help topics:
=====
symbols  names

//UNKNOWN> participant reg --name market --description "the market"
transaction ff652e63dadeaf32 submitted
//market> █
```

mktclient for marketplace transaction family

Sawtooth lake is also under continuous development and therefore it is recommended that readers keep an eye on documentation available at <http://intelledger.github.io/> in order to keep up with the latest developments.

Corda

Corda is not a blockchain. Traditional blockchain solutions, as discussed before, have the concept of transactions that are bundled together in a block and each block is linked back cryptographically to its parent block, which provides an immutable record of transactions. This is not the case with Corda: Corda has been designed entirely from scratch with a new model for providing all blockchain benefits, but without a traditional blockchain. It has been developed purely for the financial industry to solve issues arising from the fact that each organization manages their own ledgers and thus have their own view of *truth*, which leads to contradictions and operational risk. Moreover, data is also duplicated at each organization which results in an increased cost of managing individual infrastructures and complexity. These are the types of problems within the financial industry that Corda aims to resolve by building a decentralized database platform.

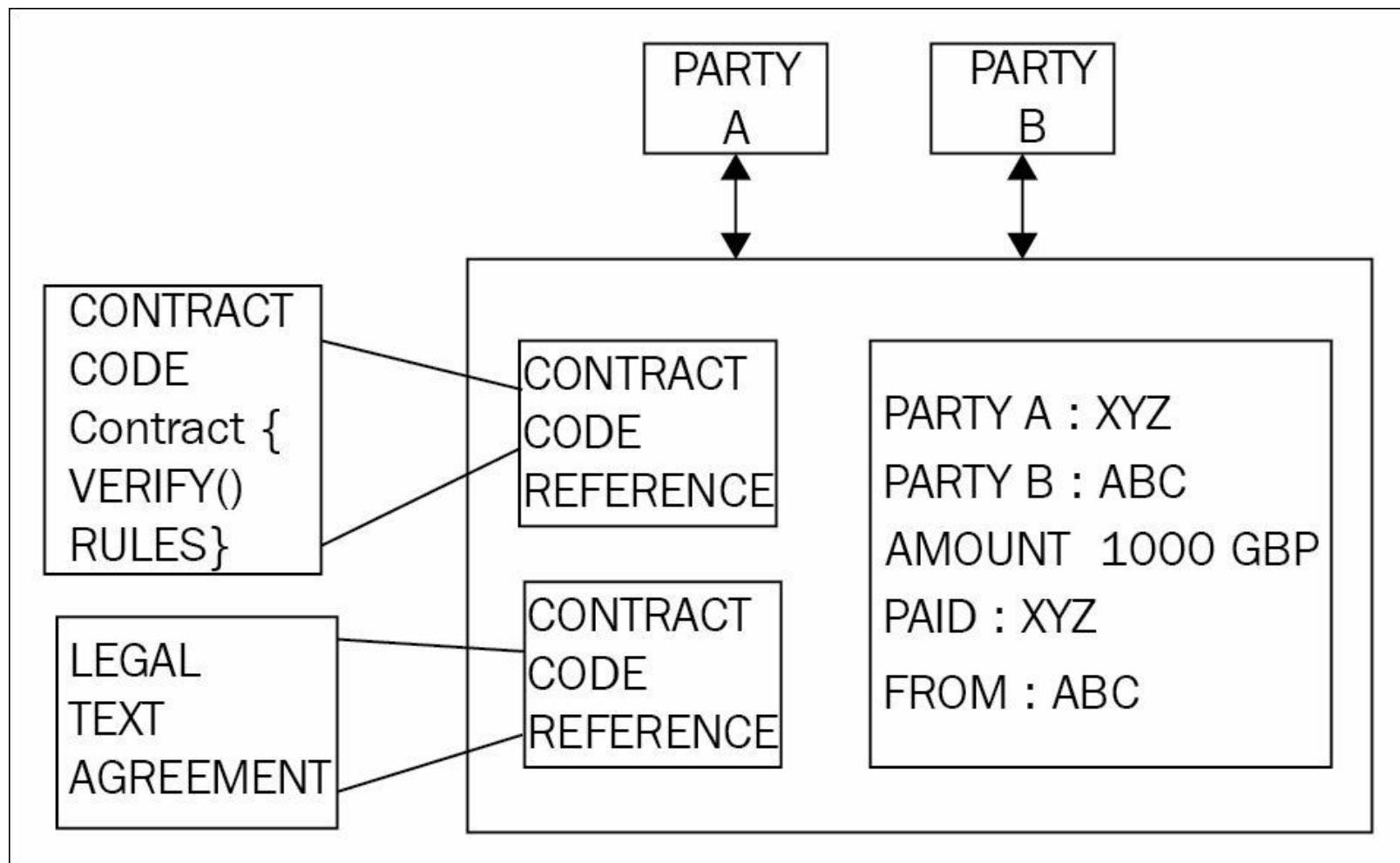
Corda source code is available at <https://github.com/corda/corda>. It is written in a language called Kotlin, which is a statically typed language targeting the **Java Virtual Machine (JVM)**.

Architecture

The main components of the Corda platform include state objects, contract code, legal prose, transactions, consensus, and flows.

State objects

State objects represent the smallest unit of data that represent a financial agreement. They are created or deleted as a result of a transaction execution. They refer to **contract code** and **legal prose**. Legal prose is optional and provides legal binding to the contract. However, contract code is mandatory in order to manage the state of the object. It is required in order to provide a state transition mechanism for the node according to the business logic defined in the contract code. State objects contain a data structure that represent the current state of the object. For example, in the following diagram, a state object represents the current state of the object. In this case, it is a simple mock agreement between **Party A** and **Party B** where **Party ABC** has paid **Party XYZ 1,000 GBP**. This represents the current state of the object; however the referred contract code can change the state via transactions. State objects can be thought of as a state machine, which are consumed by transactions in order to create updated state objects.



An example state object

Transactions

Transactions are used to perform transitions between different states. For example, the state object shown in the preceding diagram is created as a result of a transaction. Corda uses a bitcoin-style UTXO based model for its transaction processing. The concept of state transition by transactions is same as in bitcoin. Similar to bitcoin, transactions can have none, single or multiple inputs, and single or multiple outputs. All transactions are digitally signed. Moreover, Corda has no concept of mining because it does not use blocks to arrange transactions in a blockchain. Instead, notary services are used in order to provide temporal ordering of transactions. In Corda, new transaction types can be developed using JVM bytecode, which makes it very flexible and powerful.

Consensus

The consensus model in Corda is quite simple and is based on notary services that are discussed in a later section. The general idea is that the transactions are evaluated for their uniqueness by the notary service and, if they are unique, they are signed as valid. There can be single or multiple clustered notary services running on a Corda network. Various consensus algorithms like PBFT or Raft can be used by notaries to reach consensus.

There are two main concepts regarding consensus in Corda: Consensus over state validity, and consensus over state uniqueness. The first concept is concerned with the validation of the transaction, ensuring that all required signatures are available and states are appropriate. The second concept is a means to detect double--spend attack and ensures that a transaction has not been already been spent and is unique.

Flows

Flows in Corda are a novel idea that allow the development of decentralized workflows. All communication on the Corda network is handled by these flows. These are transaction-building protocols that can be used to define any financial flow of any complexity using code. Flows run as an asynchronous state machine and they interact with other nodes and users. During the execution, they can be suspended or resumed as required.

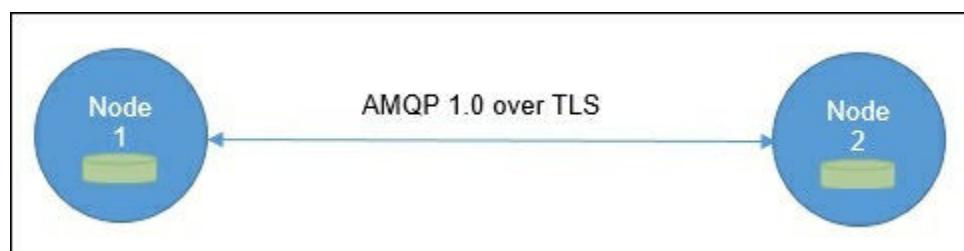
Components

The Corda network has multiple components. All these components are described in the next section.

Nodes

Nodes in a Corda network operate under a trust-less model and run by different organizations. Nodes run as part of an authenticated peer-to-peer network. Nodes communicate directly with each other using the **Advanced Message Queuing Protocol (AMQP)**, which is an approved international standard (ISO/IEC 19464) and ensures that messages across different nodes are transferred safely and securely. AMQP works over **Transport Layer Security (TLS)** in Corda, thus ensuring privacy and integrity of data communicated between nodes.

Nodes also make use of a local relational database for storage. Messages on the network are encoded in a compact binary format. They are delivered and managed by using the **Apache Artemis message broker (Active MQ)**. A node can serve as a network map service, notary, Oracle, or a regular node. The following diagram shows a high-level view of two nodes communicating with each other:



Two nodes communicating in a Corda network

In the preceding diagram, **Node 1** is communicating with **Node 2** over a TLS communication channel using the AMQP protocol, and the nodes have a local relational database for storage.

Permissioning service

A Permissioning service is used to provision TLS certificates for security. In order to participate on the network, participants are required to have a signed identity issued by a root certificate authority. Identities are required to be unique on the network and the Permissioning service is used to sign these identities. The naming convention used to recognise participants is based on the X.500 standard. This ensures the uniqueness of the name.

Network map service

This service is used to provide a network map in the form of a document of all nodes on the network. This service publishes IP addresses, identity certificates and a list of services offered by nodes. All nodes announce their presence by registering to this service when they first start up, and when a connection request is received by a node, the presence of the requesting node is checked on the

network map first. Put another way, this service resolves the identities of the participants to physical nodes.

Notary service

In a traditional blockchain, mining is used to ascertain the order of blocks that contain transactions. In Corda, notary services are used to provide transaction ordering and timestamping services. There can be multiple notaries in a network and they are identified by composite public keys. Notaries can use different consensus algorithms like BFT or Raft depending on the requirements of the applications. Notary services sign the transactions to indicate validity and finality of the transaction which is then persisted to the database.

Notaries can be run in a load-balanced configuration in order to spread the load across the nodes for performance reasons; and, in order to reduce latency, the nodes are recommended to be run physically closer to the transaction participants.

Oracle service

Oracle services either sign a transaction containing a fact, if it is true, or can themselves provide factual data. They allow real world feed into the distributed ledgers.

Transactions

Transactions in a Corda network are never transmitted globally, but in a semi-private network. They are shared only between a subset of participants who are related to the transaction. This is in contrast to traditional blockchain solutions like Ethereum and bitcoin, where all transactions are broadcasted to the entire network globally. Transactions are digitally signed and either consume state(s) or create new state(s).

Transactions on a Corda network are composed of the following elements:

- **Input references:** This is a reference to the states the transaction is going to consume and use as an input.
- **Output states:** These are new states created by the transaction.
- **Attachments:** This is a list of hashes of attached zip files. Zip files can contain code and other relevant documentation related to the transaction. Files themselves are not made part of the transaction, instead, they are transferred and stored separately.
- **Commands:** A command represents the information about the intended operation of the transaction as a parameter to the contract. Each command has a list of public keys which represents all parties that are required to sign a transaction.
- **Signatures:** This represents the signature required by the transaction. The total number of signatures required is directly proportional to the number of public keys for commands.
- **Type:** There are two types of transactions namely, Normal or Notary changing. Notary changing transactions are used for reassigning a notary for a state.
- **Timestamp:** This field represents a bracket of time during which the transaction has taken place. These are verified and enforced by notary services. Also, it is expected that if strict timings are

required, which is desirable in many financial services scenarios, notaries should be synched with an atomic clock.

- **Summaries:** This is a text description that describes the operations of the transaction.

Vaults

Vaults run on a node and are akin to the concept of wallets in bitcoin. As the transactions are not globally broadcast, each node will have only that part of data in their vaults that is considered relevant to them. Vaults store their data in a standard relational database and as such can be queried by using standard SQL. Vaults can contain both on ledger and off ledger data, meaning that it can also have some part of data that is not on ledger.

CorDapp

The core model of Corda consists of state objects, transactions and transaction protocols, which when combined with contract code, APIs, wallet plugins, and user interface components results in constructing a **Corda distributed application (CorDapp)**.

Smart contracts in Corda are written using Kotlin or Java. The code is targeted for JVM. JVM has been modified slightly in order to achieve deterministic results of execution of JVM bytecode. There are three main components in a Corda smart contract as follows:

1. Executable code that defines the validation logic to validate changes to the state objects.
2. State objects represent the current state of a contract and either can be consumed by a transaction or produced (created) by a transaction.
3. Commands are used to describe the operational and verification data that defines how a transaction can be verified.

Development environment

The development environment for Corda can be set up easily using the following steps.

Required software includes the following:

1. JDK 8 which is available
at <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
2. IntelliJ IDEA community edition which is free and available at
<https://www.jetbrains.com/idea/download/>.
3. H2 database platform independent zip, and is available
at <http://www.h2database.com/html/download.html>.
4. Git, available at <https://git-scm.com/downloads>.
5. Kotlin language, which is available for IntelliJ, and more information can be found at
<https://kotlinlang.org/>.
6. Gradle is another component that is used to build Corda.

Once all these tools are installed, smart contract development can be started. CorDapps can be developed by utilizing an example template available at <https://github.com/corda/cordapp-template>. Detailed documentation on how to develop contract code is available at <https://docs.corda.net/>.

Corda can be cloned locally from GitHub using the following command:

```
$ git clone https://github.com/corda/corda.git
```

When the cloning is successful, you should see output similar to the following:

```
Cloning into 'corda'...
remote: Counting objects: 74695, done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 74695 (delta 17), reused 0 (delta 0), pack-reused 74591
Receiving objects: 100% (74695/74695), 51.27 MiB | 1.72 MiB/s, done.
Resolving deltas: 100% (42863/42863), done.
Checking connectivity... done.
```

Once the repository is cloned, it can be opened in IntelliJ for further development. There are multiple samples available in the repository, such as a bank of Corda, interest rate swaps, demo, and traders demo. Readers can find them under the `/samples` directory under `corda` and they can be explored using IntelliJ IDEA IDE.

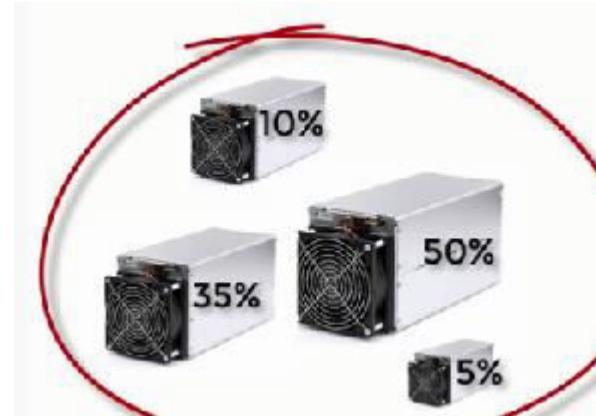
Summary

In this chapter, we've provided an introduction to the Hyperledger project. Firstly, the core ideas behind the Hyperledger project were discussed and a brief introduction to all projects under incubation in Hyperledger was provided. Three main Hyperledger projects were discussed in detail, namely Hyperledger fabric, Sawtooth lake and Corda. All these projects are currently under heavy development and changes are expected in the next releases. Because of this, no in-depth practical exercises were given. However, the core concepts of all the projects mentioned above are expected to remain unchanged or changed only very slightly. Readers are encouraged to visit the relevant links provided within the chapter in order see the latest updates. It is obvious that a lot is going on in this space and projects like Hyperledger from the Linux foundation are playing a key role in the advancement of blockchain technology. Each of the projects discussed in this chapter has novel approaches towards solving the issues faced in various industries, and any current limitations within the blockchain technology are also being addressed, such as scalability and privacy. It is expected that more projects will soon be proposed to the Hyperledger project, and it is envisaged that with this collaborative and open effort blockchain technology will advance tremendously and will benefit the community as a whole.

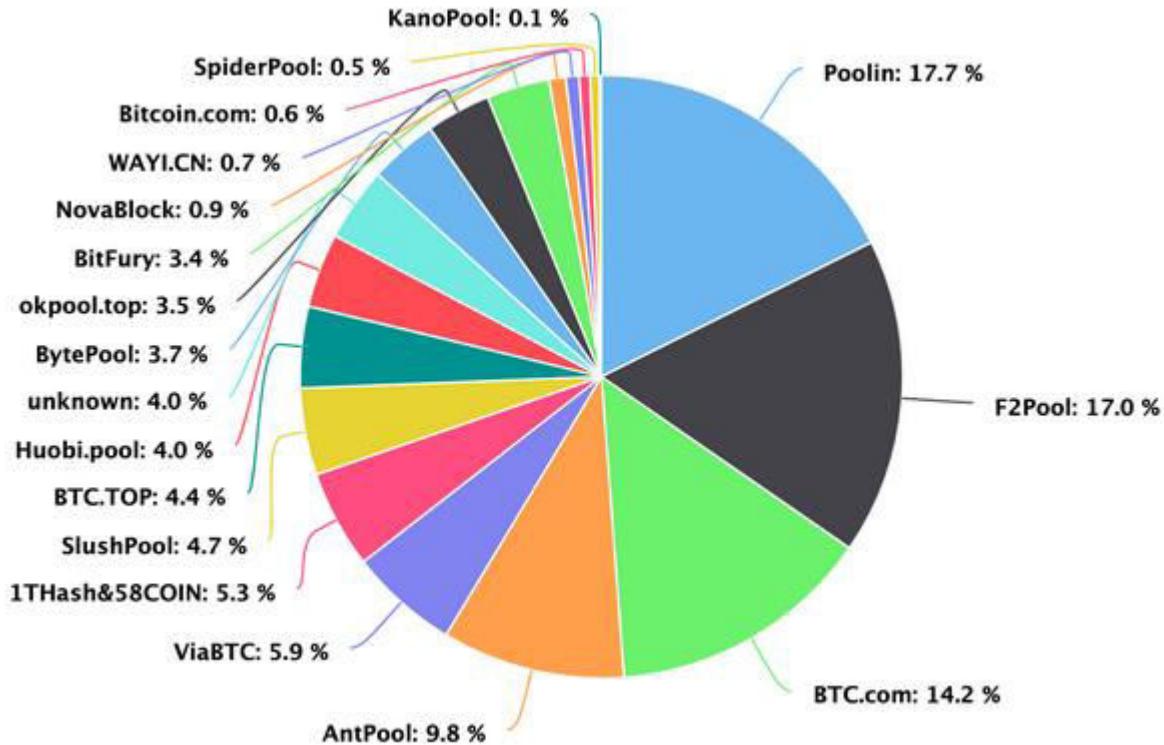
Miner Pools, Types of Mining

Miner Pools

- Miners group together to form a **pool**.
- They combine their mining power to compete more effectively.
- If the pool manager to win the competition, then **reward** is spread out between the pool members.



Miner Pools



Mining Profitability Factors

- Hash Rate
- Block Reward
- Mining Difficulty
- Electricity Cost
- Power Consumption
- Pool Fees
- Bitcoin Price
- Difficulty Increase

Mining Calculator



Bitcoin Mining Calculator

Hashing Power:	Difficulty:	
100	GH/s ▾	7454968648263
Pool Fees:	BTC to USD Price:	
% 3	\$ 6278.77	
Maintenance (GH/s per day):	Hardware/Contract Costs:	
\$ 0	\$ 0	
Power Usage:	Block Reward:	
w 0	12.5	
Power Cost (kw/h):	Calculate	
\$ 0.11	Powered by CryptoRival	

Types of Mining

- Cloud Mining
- Mobile Mining
- Web Mining

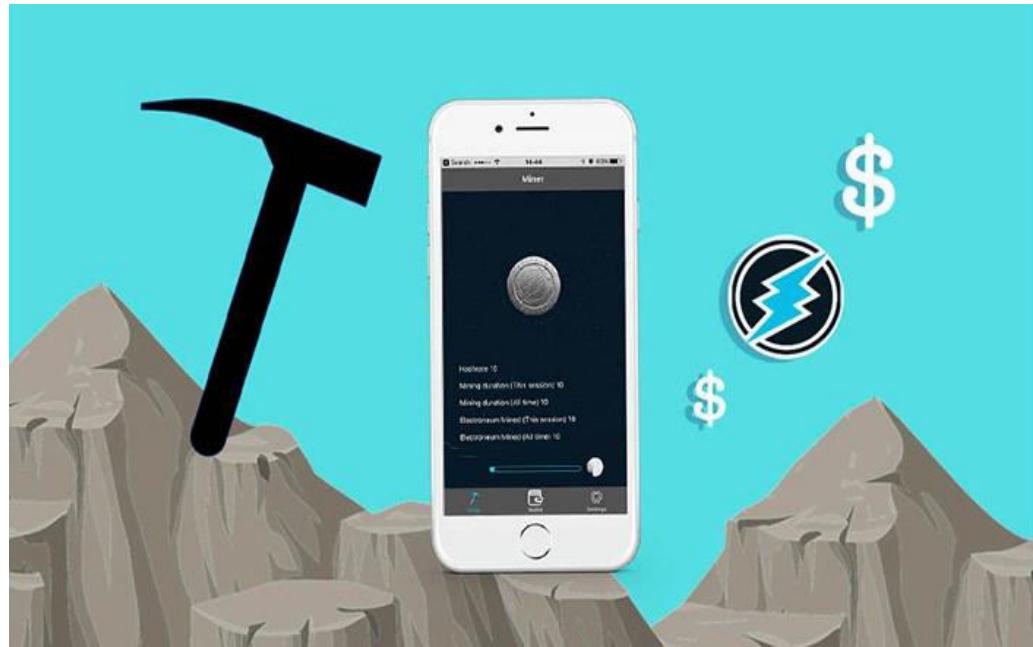
Cloud Mining



Cloud Mining

- Cloud mining is a term describing companies that allow you to rent mining hardware they operate and maintain in exchange for a fixed fee and a share of the revenue you'll make.
- It basically means that you can mine remotely without the need for buying expensive mining hardware.
- Most, if not all, cloud mining companies today are either plain scams or work through an ineffective business model.
- By ineffective I mean that you will either lose money or earn less than you would have by just buying and holding Bitcoins.

Mobile Mining



Mobile Mining

- Cryptocurrency mobile mining is possible, but it comes with a long list of reasons not to do it.
- Moreover, mining on your smartphone doesn't even come close to traditional mining hardware or software.
- In the current state of cryptocurrency mining, doing it on your smartphone might not bring you enough profits to be worth the time and effort.

Which smartphones can be used for mining?

- You can only do **cryptocurrency mobile mining** with smartphones that uses Android
- Moreover, the market is flooded with apps created for Android that allow you to mine Bitcoin directly from your smartphone.
- Cryptocurrency mobile mining leads to **overheating, battery damage, and overall lower performance**.

Popular apps for mining crypto with your smartphone

- **MinerGate Mobile Miner** is an app that enables you to mine for multiple altcoins besides Bitcoin. Among them, you have Monero, Dash, DigitalNote, MonetaVerde, and QuazarCoin.
 - The app also provides a built-in wallet, where users can store their hard-earned coins.
- **Bitcoin Miner** is one of the most popular applications at the moment and is available on most devices. It has a user-friendly interface and its performance often receives good reviews.

Wrapping it up

- Cryptocurrency mobile mining isn't complicated. All you need is a decent **smartphone and a mining app**.
- The app runs in the background while you're using your phone, and you receive rewards for it.
- The **downside** is that all mining apps interfere with the performance of your smartphone and usually end up damaging your device.
- In the long run, what may have seemed like a simple way of making some extra cash could just cause more expenses.

Web Mining

- The idea of cryptocurrency mining in browsers is not something new. In 2013, a group of MIT students founded a company called **Tidbit**, which offered a web service to mine Bitcoins.
- Instead of displaying advertisements, webmasters could include Tidbit's scripts in their websites to earn money via **Bitcoin mining**.
- In this particular case, the mining is performed directly within the browser when the user browses to certain websites. Thus, there is no need to infect the victim's machine or to exploit vulnerabilities.
- All that is needed is a browser with **JavaScript** activated, which is the default state of most browsers.

Is Mining wasteful?

Bitcoin mining ultimately requires less resources than the current banking system

Can't Google start mining Bitcoin and blow out the competition?

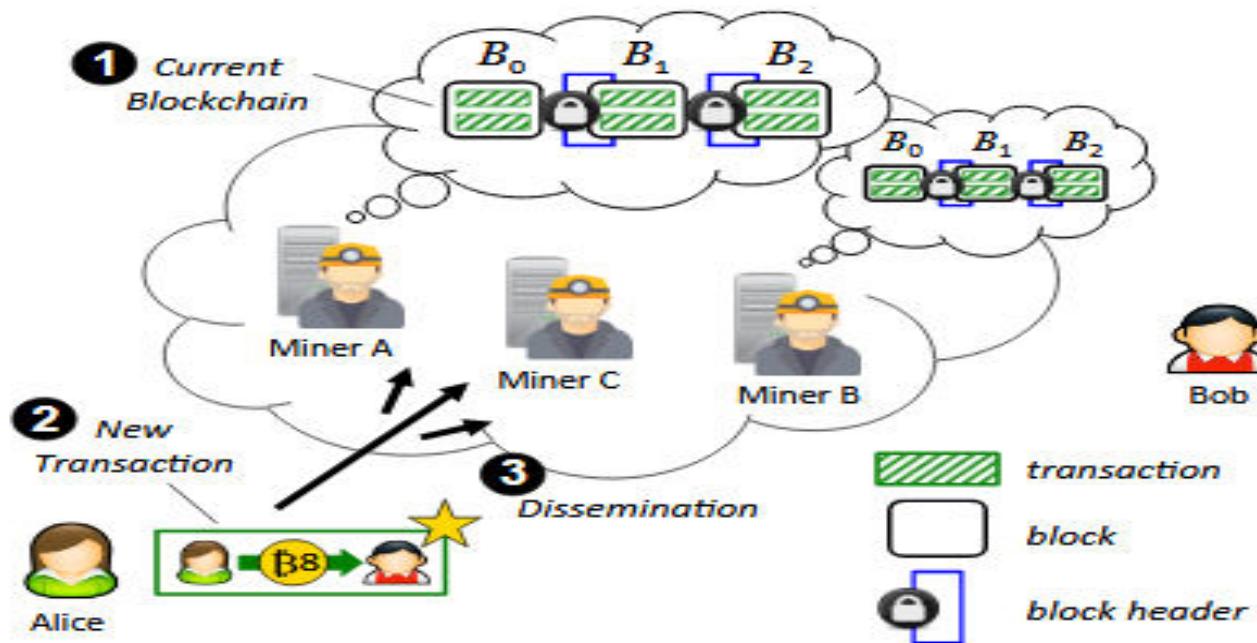
Should you be mining bitcoins?

Miners

Introduction-Miners

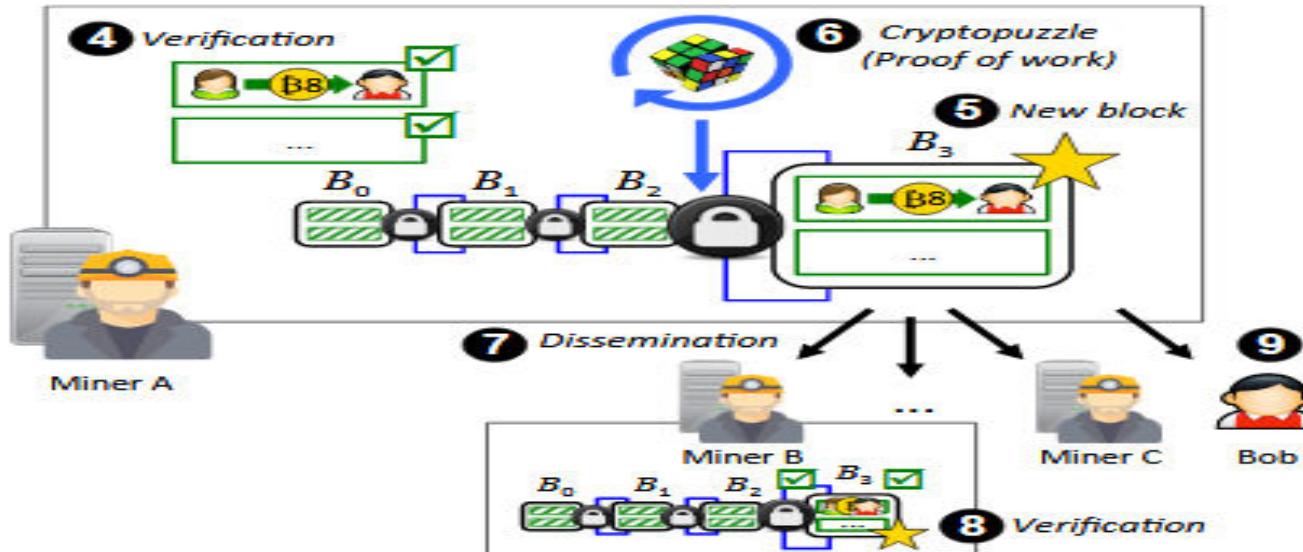
- Miners can be defined as accountants who records every transactions to the blockchain. The concept is simple, a proof of payment is important if you want your payment to be valid.
- The miners are the ones who keep the record of your payment. Hence they are record keepers who keep the system updated of new payments and existing ones.
- Miners are the network participants who have the necessary hardware and computing power to validate the transactions.
- These play an important role in securing and extending the blockchain

Miners



Blockchain is formed of a sequence of blocks containing transactions. The current state of the blockchain (here(B_0 , B_1 , B_2)) is stored by each individual miner.

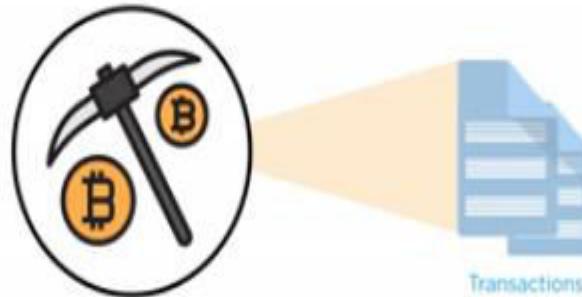
Contd...



To add a new transaction to the current blockchain, a miner first verifies the validity of the transaction. It must then solve a costly crypto puzzle to encapsulate this transaction in a new block (here B_3), before disseminating this block to other miners.

What is Bitcoin Mining?

- Bitcoin mining is the process of verifying bitcoin transactions and recording them in the public blockchain ledger.
- In blockchain, the transactions are verified by bitcoin users, so basically the transactions have to be verified by the participants of the network.
- Those who have the required hardware and computing power are called miners.
- The process is solved based on a difficult mathematical puzzle called proof of work. The proof of work is needed to validate the transaction and for the miner to earn a reward.
- All the miners are competing amongst themselves to mine a particular transaction; the miner who first solves the puzzle gets the reward.



Note: In Blockchain, the transactions are verified by Bitcoin users



Here, any user with mining hardware and internet access can take part



The process is solved based on a difficult mathematical puzzle called proof of work



The miner who first solves the puzzle gets rewarded

Note: Users trying to solve the puzzle are called as miners

Concepts of Bitcoin Mining

In order to understand Bitcoin mining, we need to understand the 3 major concepts of Blockchain



Concepts of Bitcoin Mining(Distributed Ledger)

A **distributed ledger** is a record of all transactions maintained in the blockchain network across the globe

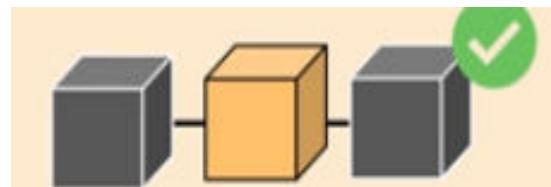


In the network, the validation of transactions is done by Bitcoin users



Concepts of Bitcoin Mining(SHA-256)

Blockchain prevents **unauthorized access** by using a **hash function(SHA-256)** to ensure the blocks are kept secure



It takes an input string of any size and returns a fixed length(256 bit) of output value



Concepts of Bitcoin Mining(Proof of Work)



PoW



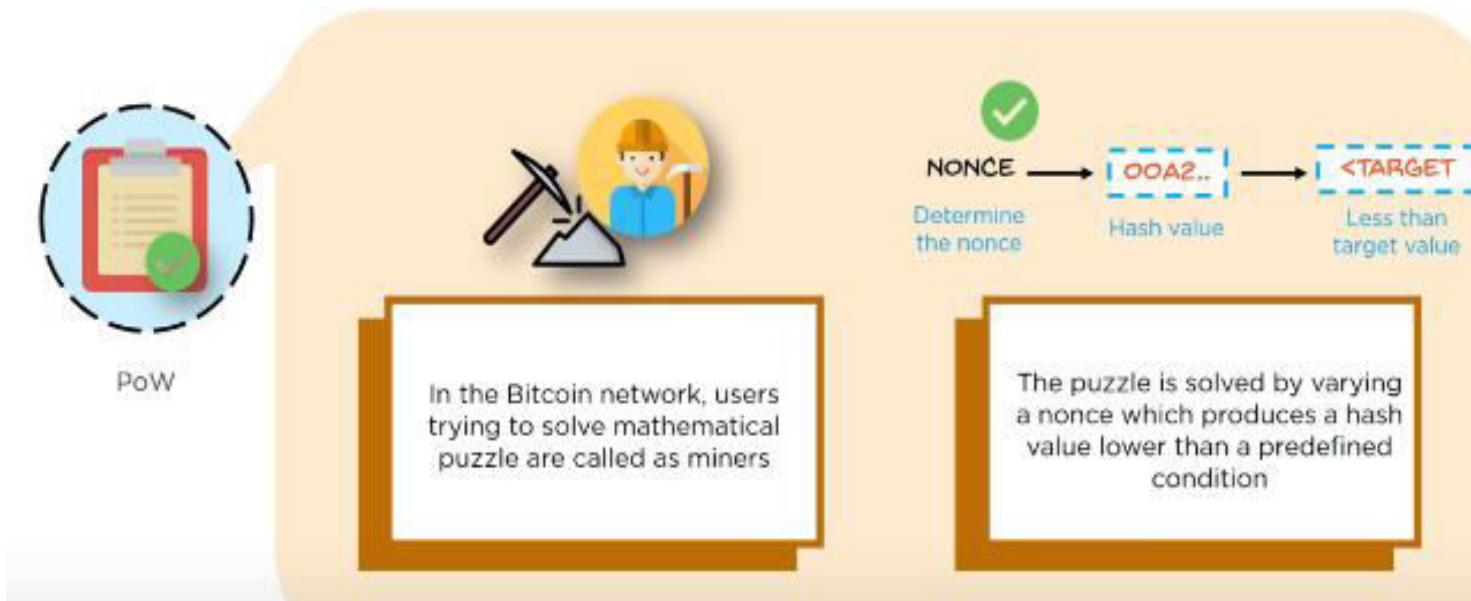
In Blockchain, mining is a process to validate transactions by solving a difficult mathematical puzzle called proof of work

NONCE VALUE: 00AZ...



Determining nonce value is the mathematical puzzle that miners are required to solve

Concepts of Bitcoin Mining(Proof of Work)



Concepts of Bitcoin Mining(Proof of Work)



PoW



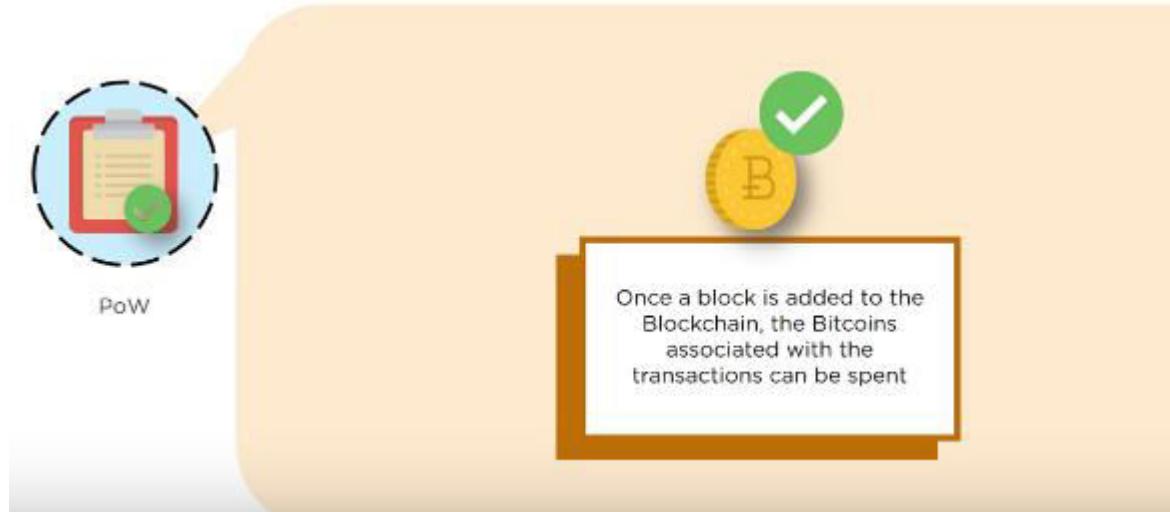
Miners verify the transactions
and add the block to the
Blockchain when confirmed



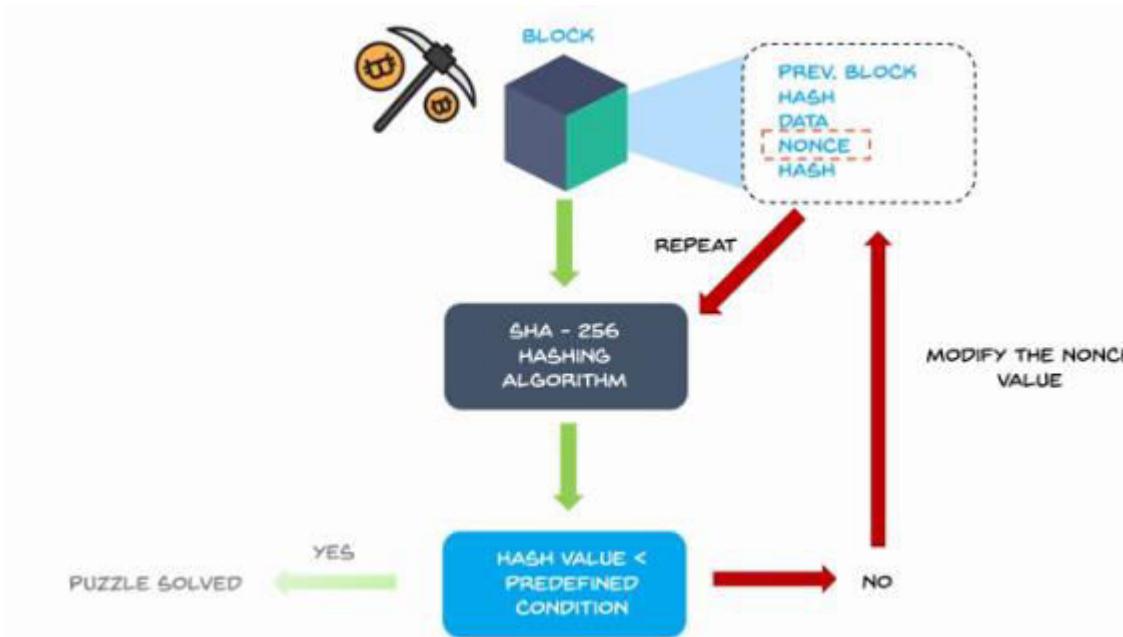
12.5

The first miner who solves the
puzzle gets a reward of 12.5
Bitcoins

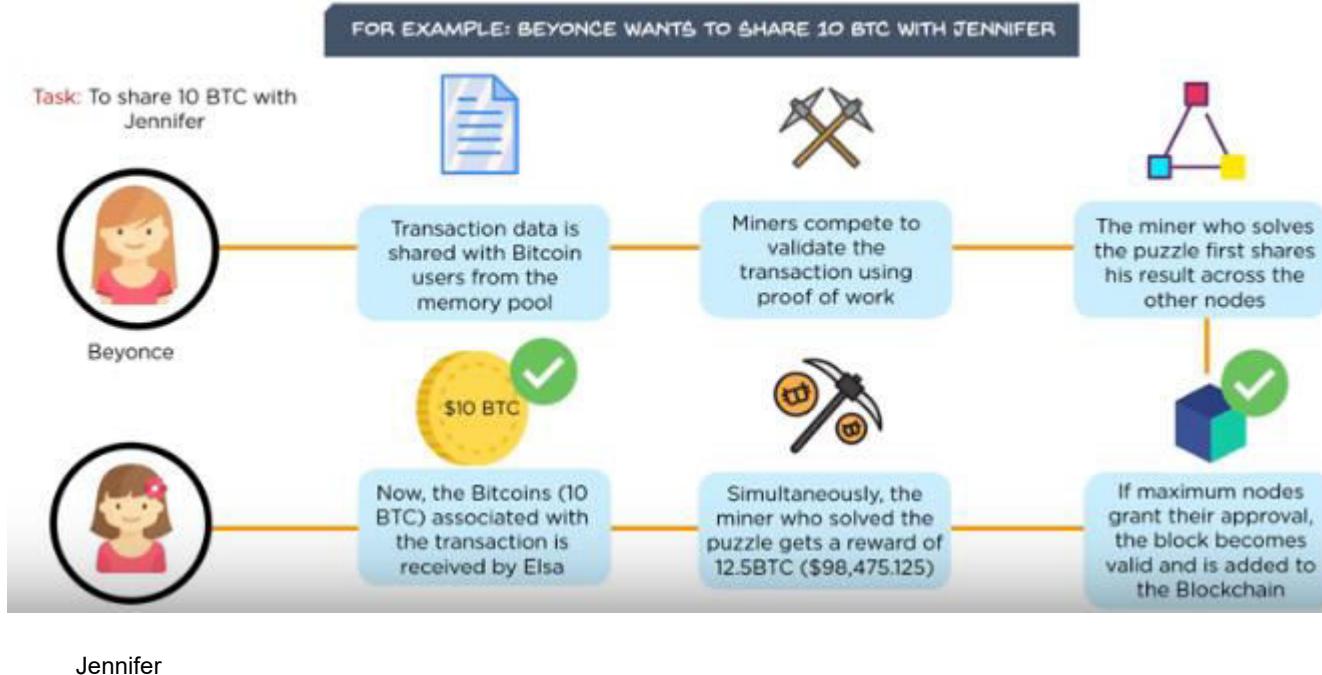
Concepts of Bitcoin Mining(Proof of Work)



Concepts of Bitcoin Mining



Concepts of Bitcoin Mining



Concepts of Bitcoin Mining

DID YOU KNOW?

- ✓ In PoW, a predefined condition is adjusted for every 2016 blocks (approximately every 14 days)
- ✓ An average time to mine a block is 10 minutes

$$\text{Hash Target} = \frac{\text{Hash Target } (current)}{\text{Avg. time taken to generate last 2016 blocks } (mins)} \times 10 \text{ (mins)}$$

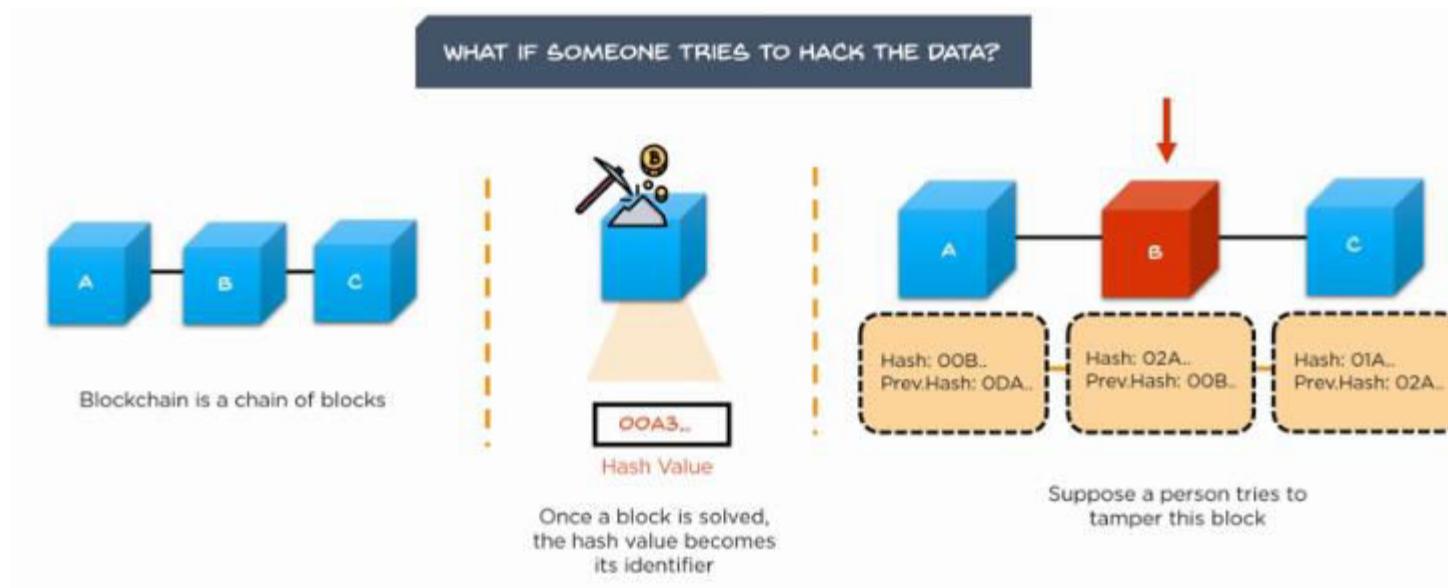
Concepts of Bitcoin Mining

DID YOU KNOW?

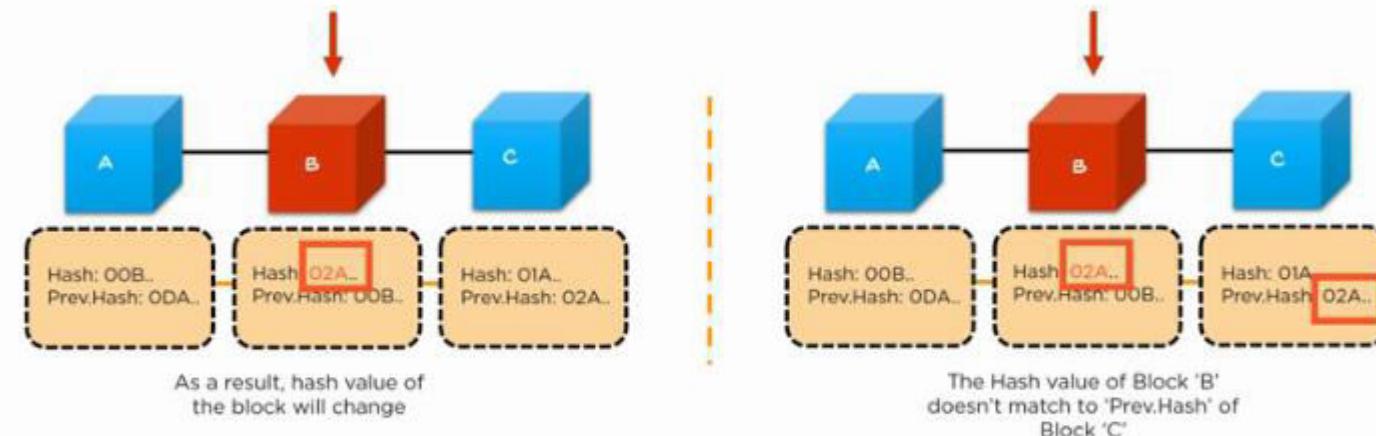
- ✓ The difficulty (condition) of the puzzle changes depending on the time it takes to mine a block

$$\text{Difficulty}_{\text{(new)}} = \frac{\text{Hash Target}_{\text{(genesis block)}}}{\text{Hash Target}_{\text{(current block)}}}$$

Concepts of Bitcoin Mining

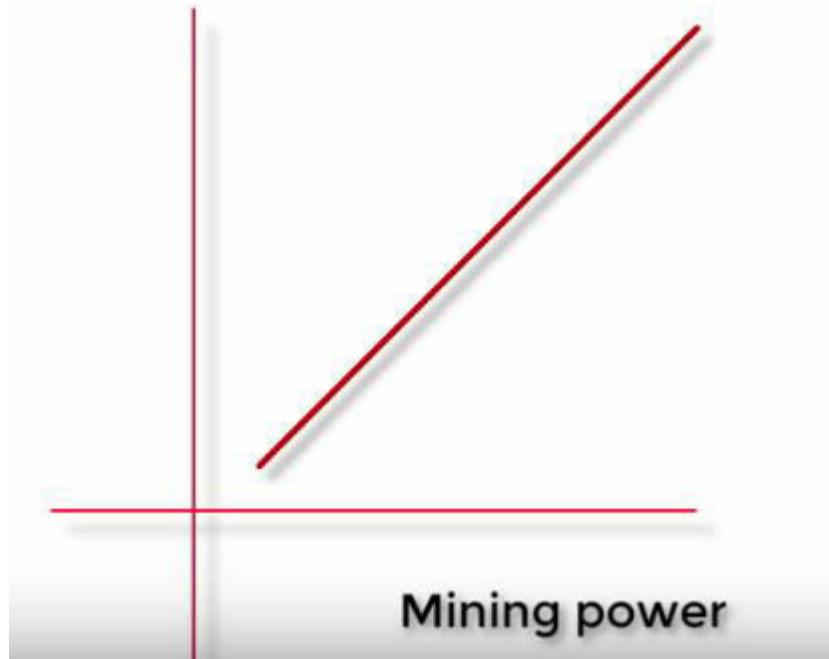


Concepts of Bitcoin Mining



Contd...

Mining difficulty



Mining Difficulty



I guessed the
solution!!

Every 10 minutes
(on average)

Miner Evolution

CPU Mining

GPU Mining

FPGA Mining

ASIC Mining

Hardware for Bitcoin Mining

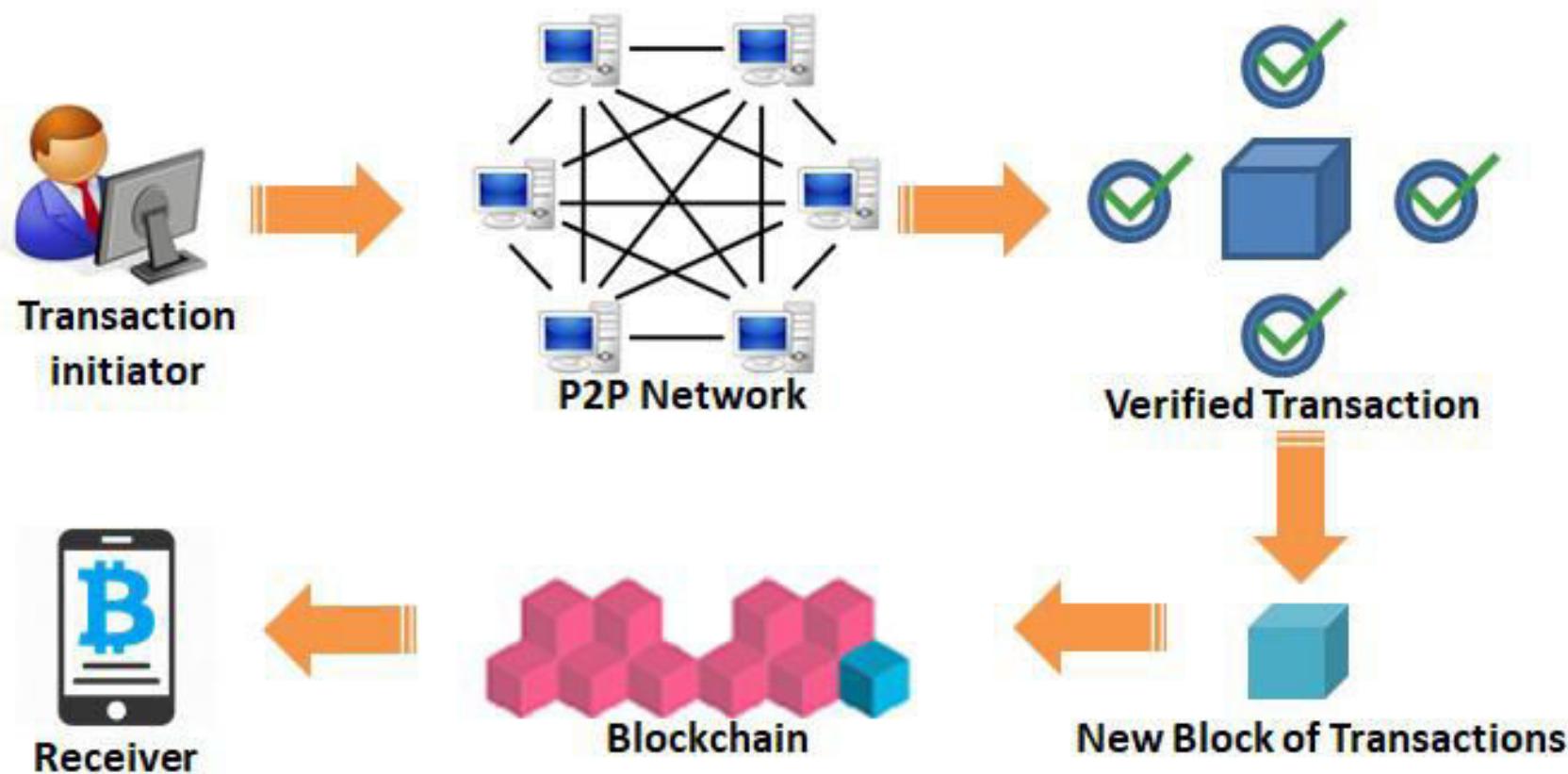
- In the early days of bitcoin, miners used to solve the mathematical puzzles using regular processors—**controlling processor units (CPUs)**.
- They discovered that **graphical processing units (GPUs)** proved to be more efficient than regular CPUs, but this also had the drawback of consuming more electricity.
- Today miners use hardware called **ASIC (application-specific integrated circuit)**, which was specifically introduced for mining Bitcoin and other cryptocurrencies. It consumes less power and has a higher computing power.
- Miners are **profitable** when their cost of resources to mine one block is less than the price of the reward.

Transaction Structure & Block Structure

Dr.Bhawana Rudra

NITK

How Blockchain Works



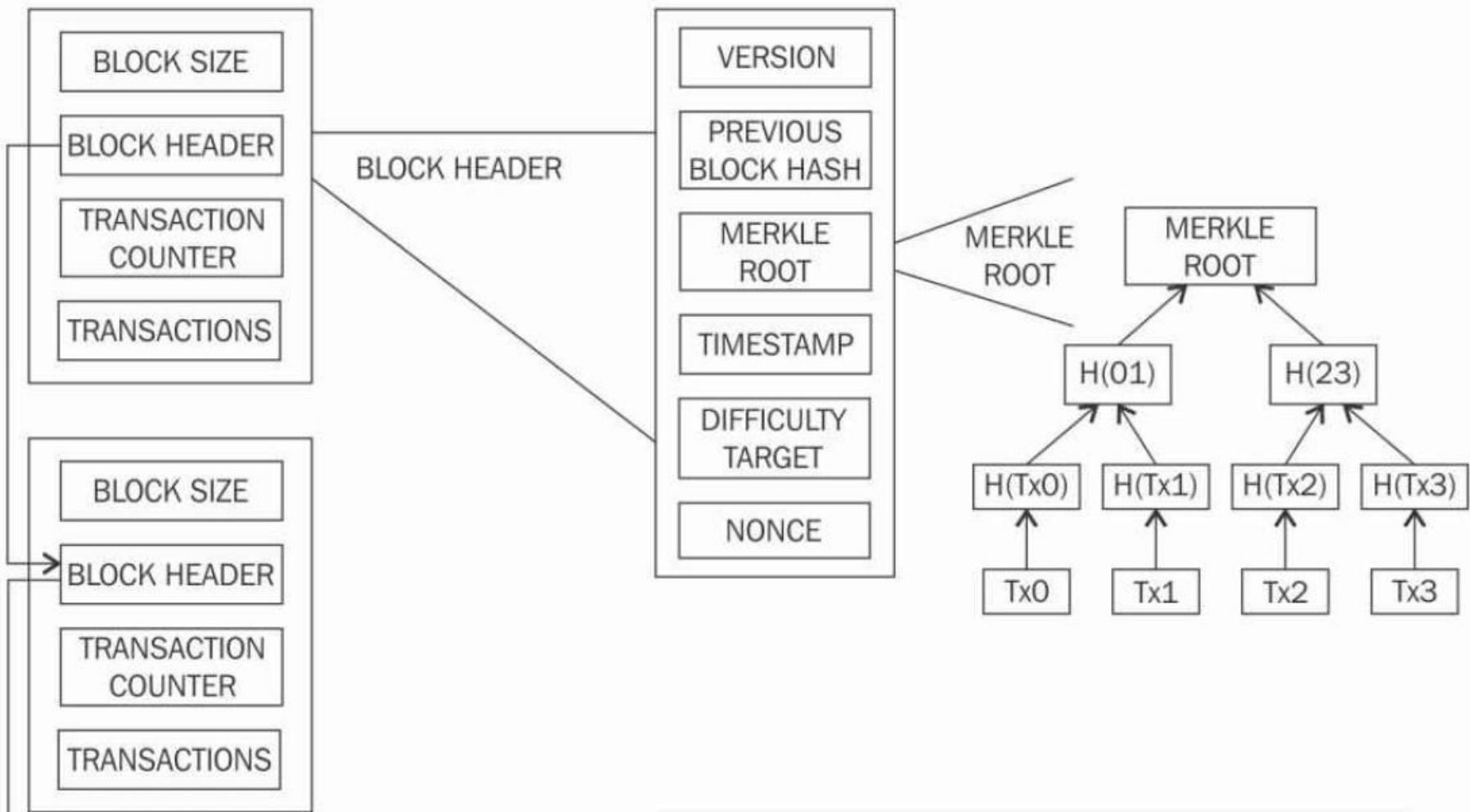
The transaction structure

Field	Size	Description
Version Number	4 bytes	Used to specify rules to be used by the miners and nodes for transaction processing.
Input counter	1 bytes - 9 bytes	The number of inputs included in the transaction.
list of inputs	variable	Each input is composed of several fields, including Previous transaction hash, Previous Txout-index, Txin-script length, Txin-script, and optional sequence number. The first transaction in a block is also called a coinbase transaction. It specifies one or more transaction inputs.
Out-counter	1 bytes - 9 bytes	A positive integer representing the number of outputs.
list of outputs	variable	Outputs included in the transaction.
lock_time	4 bytes	This defines the earliest time when a transaction becomes valid. It is either a Unix timestamp or a block number.

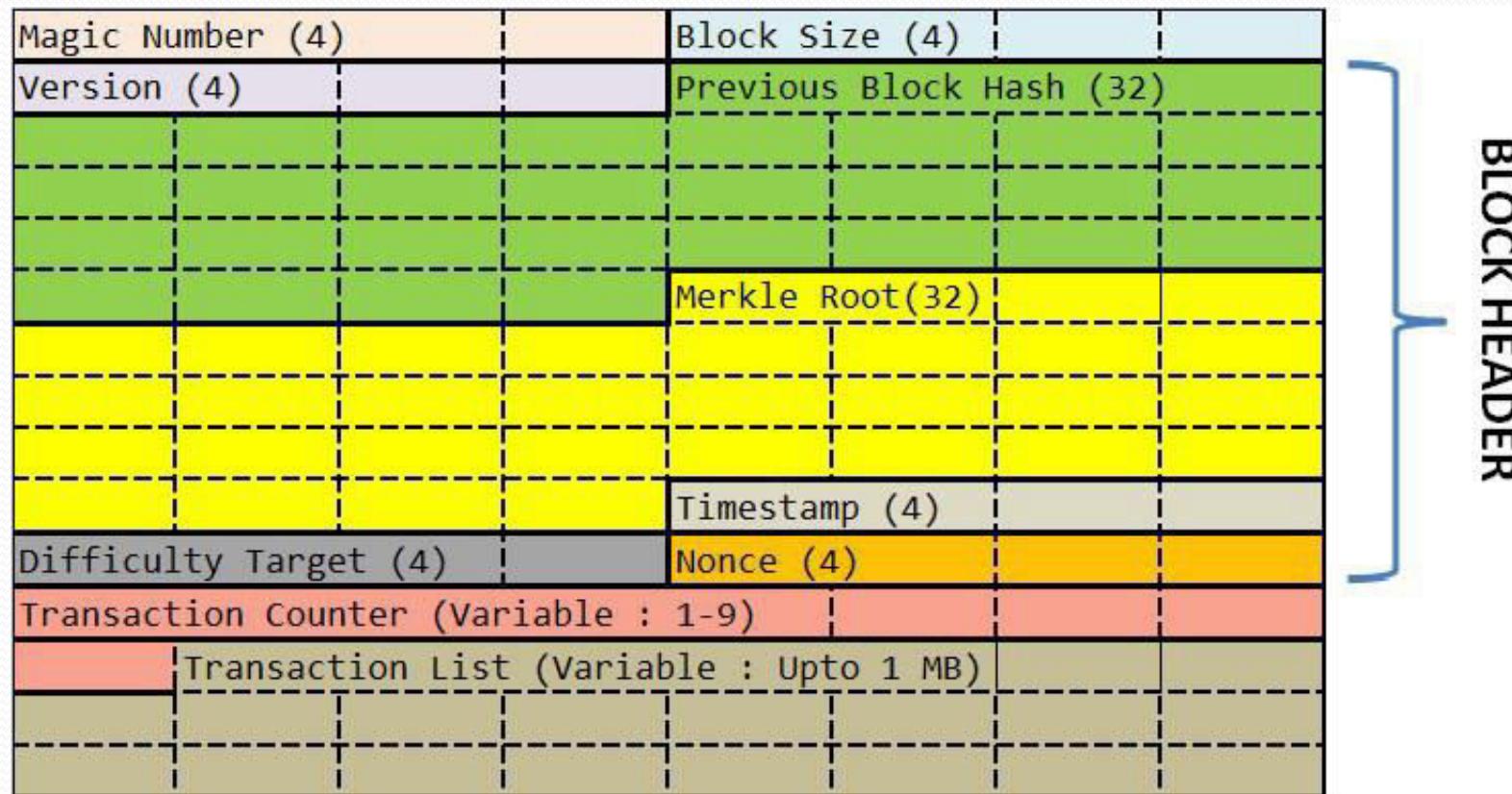
- **MetaData:** This part of the transaction contains some values such as the size of the transaction, the number of inputs and outputs, the hash of the transaction, and a `lock_time` field. Every transaction has a prefix specifying the version number.
- **Inputs:** Generally, each input spends a previous output. Each output is considered an Unspent
- **Transaction Output (UTXO)** until an input consumes it.
- **Outputs:** Outputs have only two fields, and they contain instructions for the sending of bitcoins.
- The first field contains the amount of Satoshis, whereas the second field is a locking script that contains the conditions that need to be met in order for the output to be spent. More information on transaction spending using locking and unlocking scripts and producing outputs
- **Verification:** Verification is performed using bitcoin's scripting language

Block Structure

— BLOCK HEIGHT —



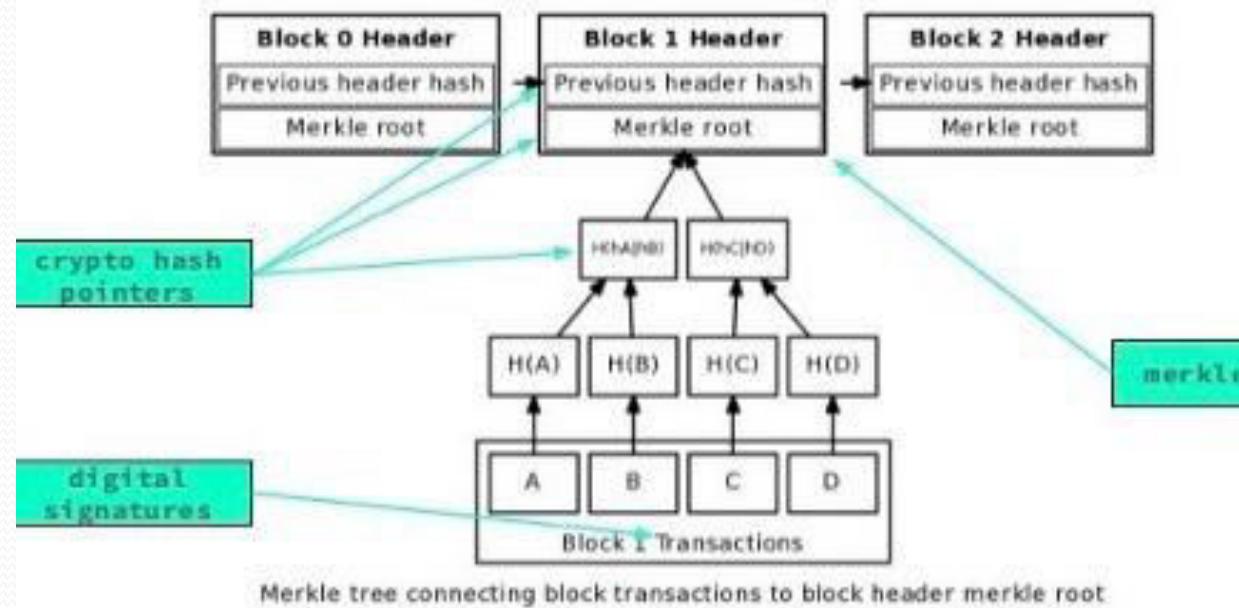
Block Header



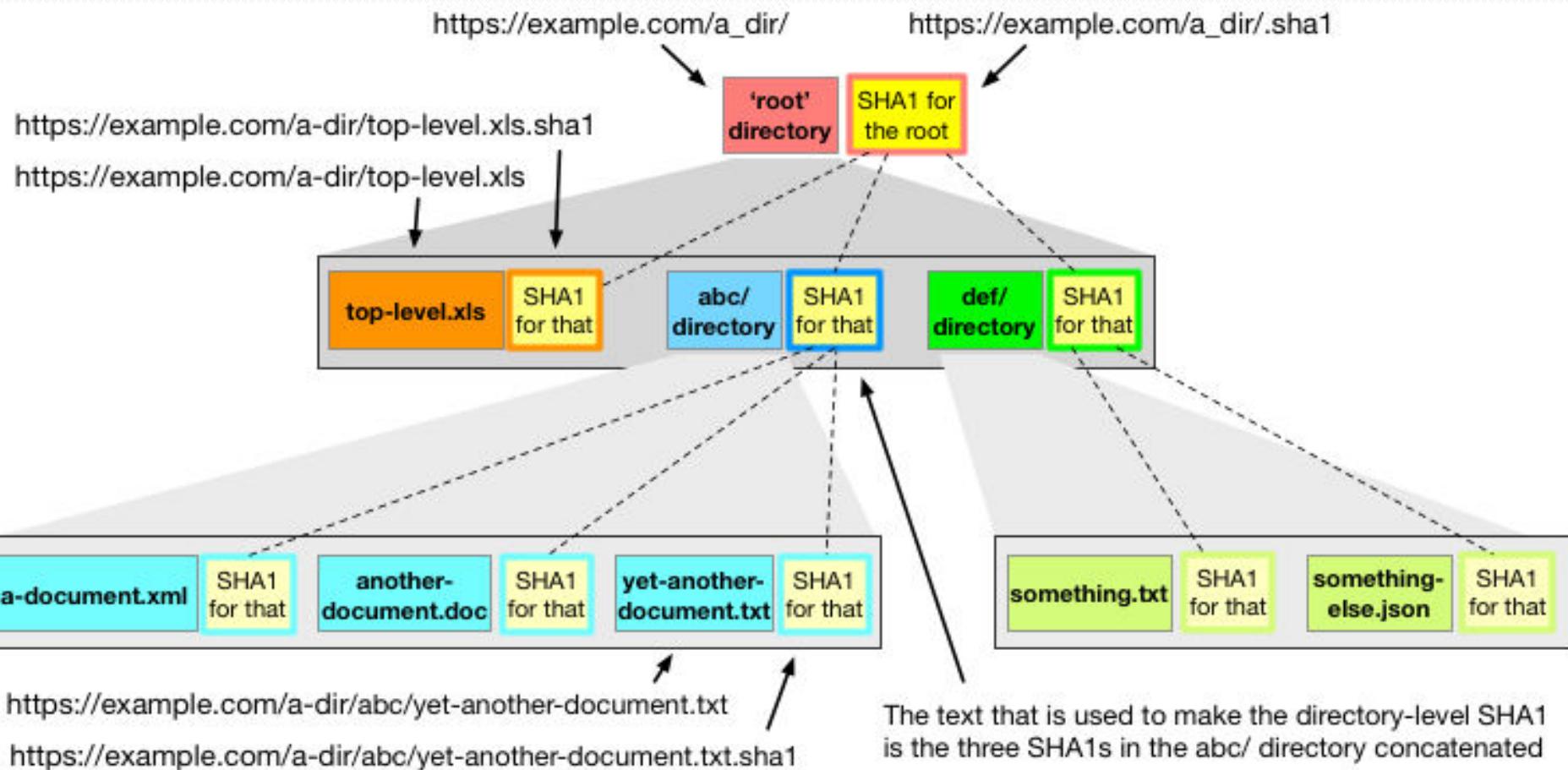
Block-Detail

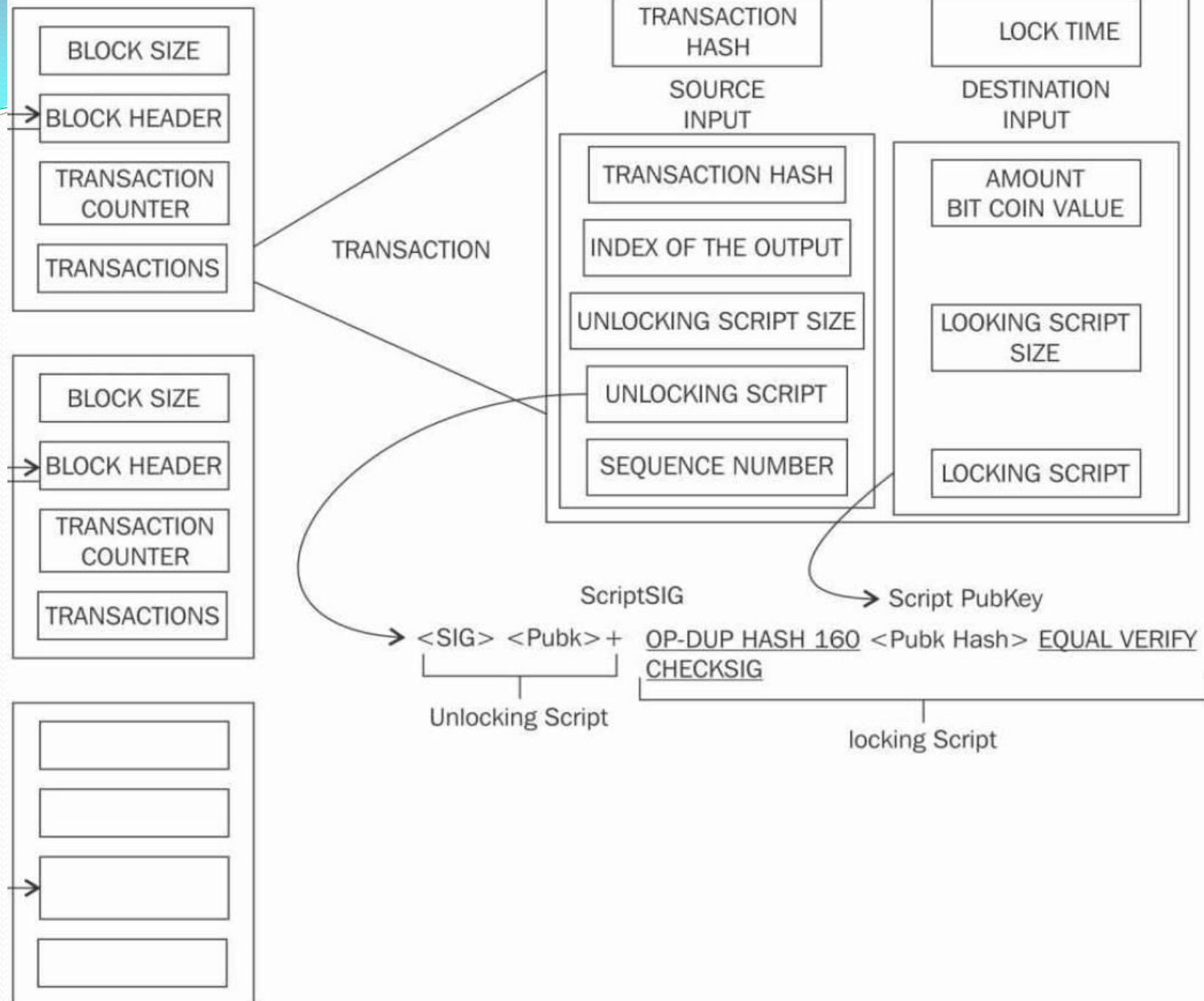
- **Magic number:** an identifier for the Blockchain network, Indicates a) Start of the block b) Data is from network
- **Block size:** each block is fixed to 1 MB. However will be increased to 2 MB. The maximum capacity is 2 GB
- **Version:** Each node has to implement
- **Timestamp, Difficulty Target, Nonce:** Time and Input for Hash
- **Hash function: SHA-256, Merkle Tree:** Hash Tree
- Public Key/Private Key & ECDSA
- Nodes/Peers- Peer to Peer Network
- Wallet
- Smart Contract (Optional)
- PoW

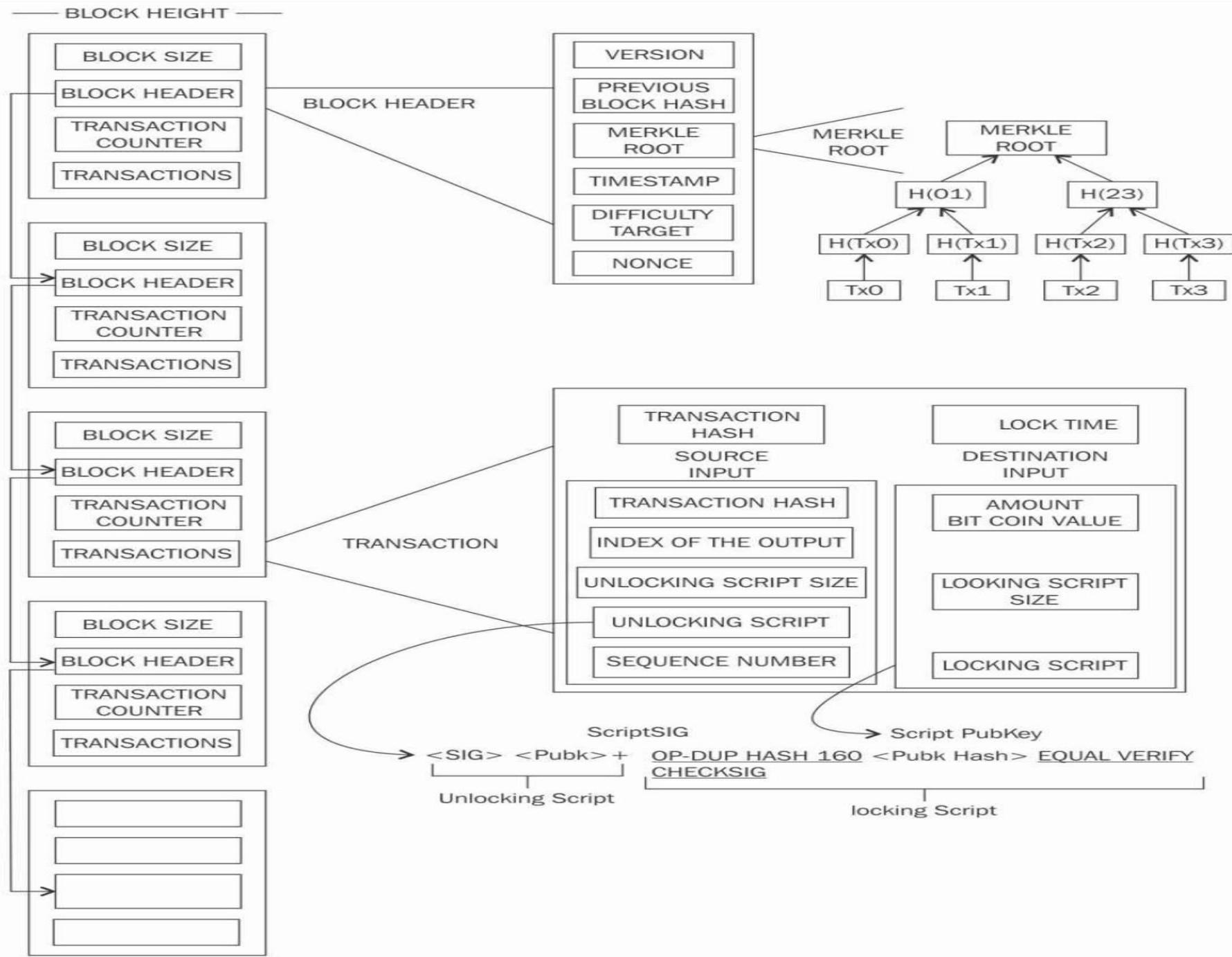
Blockchain Structure



Merkle Tree Structure







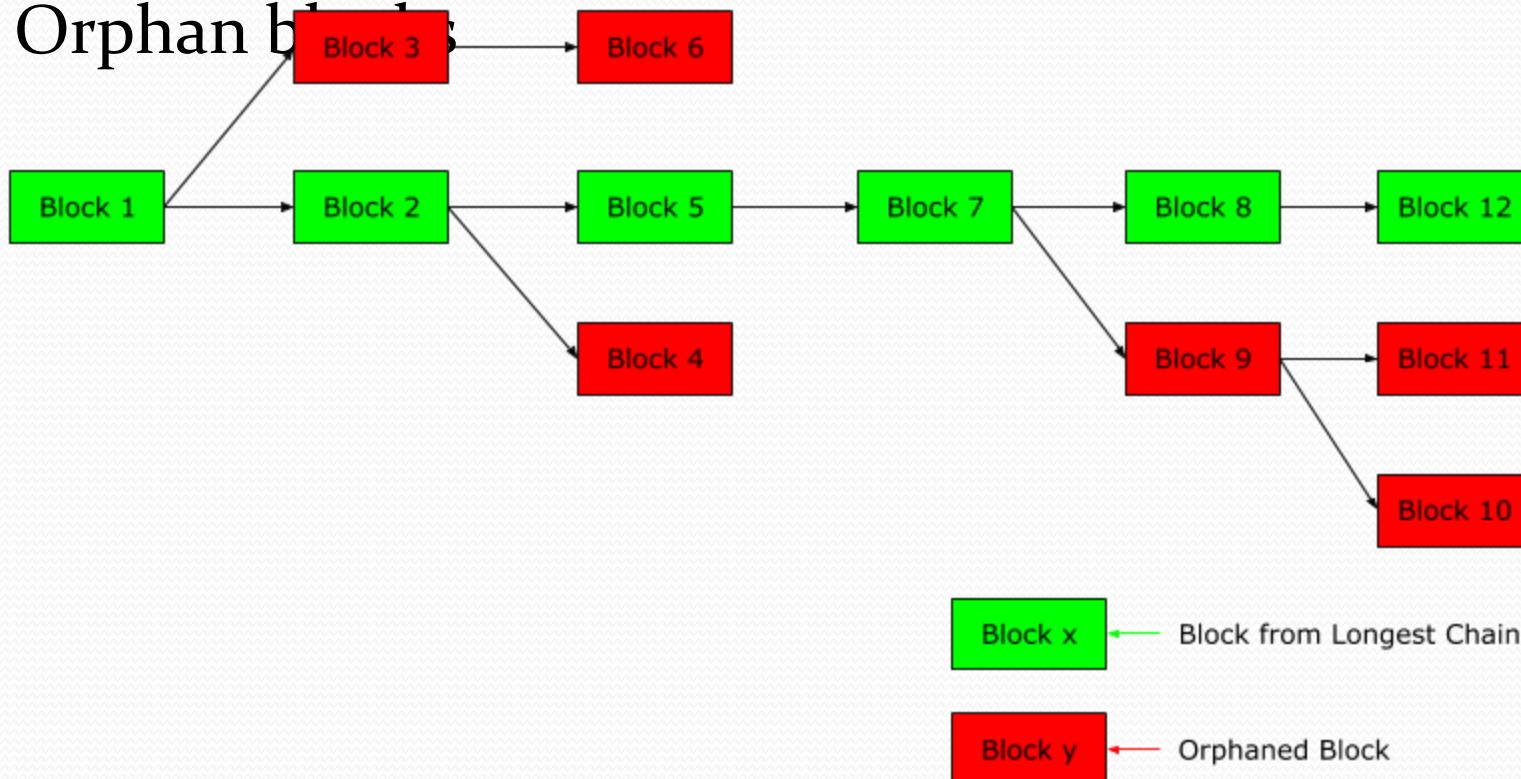
- Genesis Block
- This is the first block in the bitcoin blockchain. The genesis block was hardcoded in the bitcoin core software. It is in the chainparams.cpp file.
- <https://github.com/bitcoin/bitcoin/blob/master/src/chainparams.cpp>

```
48     *      CtxOut(nValue=50.00000000, scriptPubKey=0x5F1DF16B2B704C8A578D0B)
49     *      vMerkleTree: 4a5e1e
50     */
51 static CBlock CreateGenesisBlock(uint32_t nTime, uint32_t nNonce, uint32_t nBits, int32_t nVersion, const CAmount& genesisReward)
52 {
53     const char* pszTimestamp = "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks";
54     const CScript genesisOutputScript = CScript() << ParseHex("04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649
55     return CreateGenesisBlock(pszTimestamp, genesisOutputScript, nTime, nNonce, nBits, nVersion, genesisReward);
56 }
57 /**
58 */
```

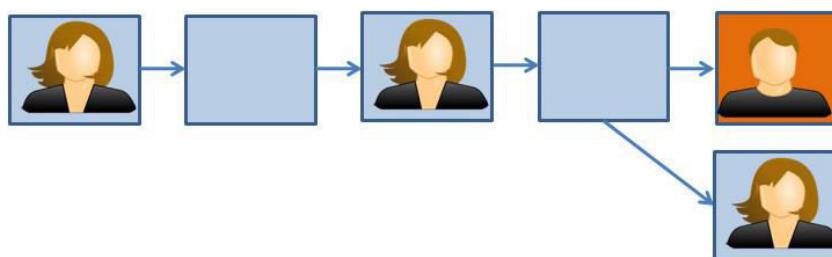
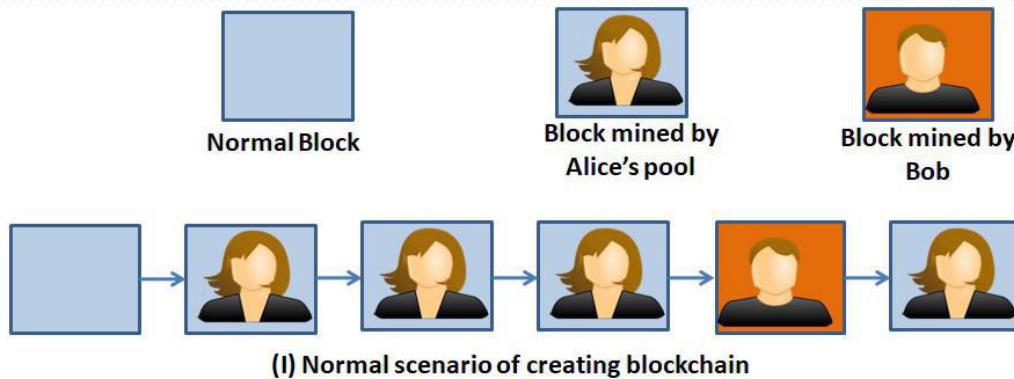
- Stale blocks

Time

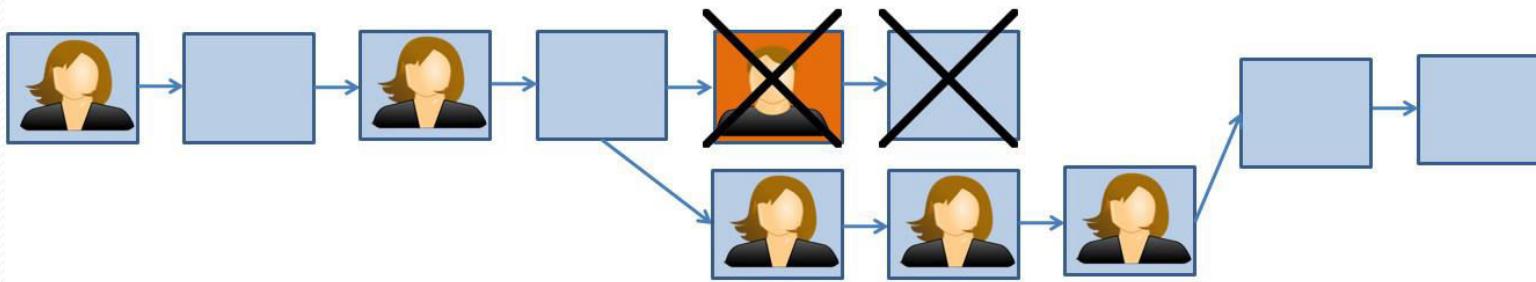
- Orphan blocks



- Because of the distributed nature of bitcoin, network forks can occur naturally. In cases where two nodes simultaneously announce a valid block can result in a situation where there are two blockchains with different transactions
- This is an undesirable situation but can be addressed by the bitcoin network only by accepting the longest chain.
- If an adversary manages to gain 51% control of the network hashrate (computational power), then they can impose their own version of transaction history.

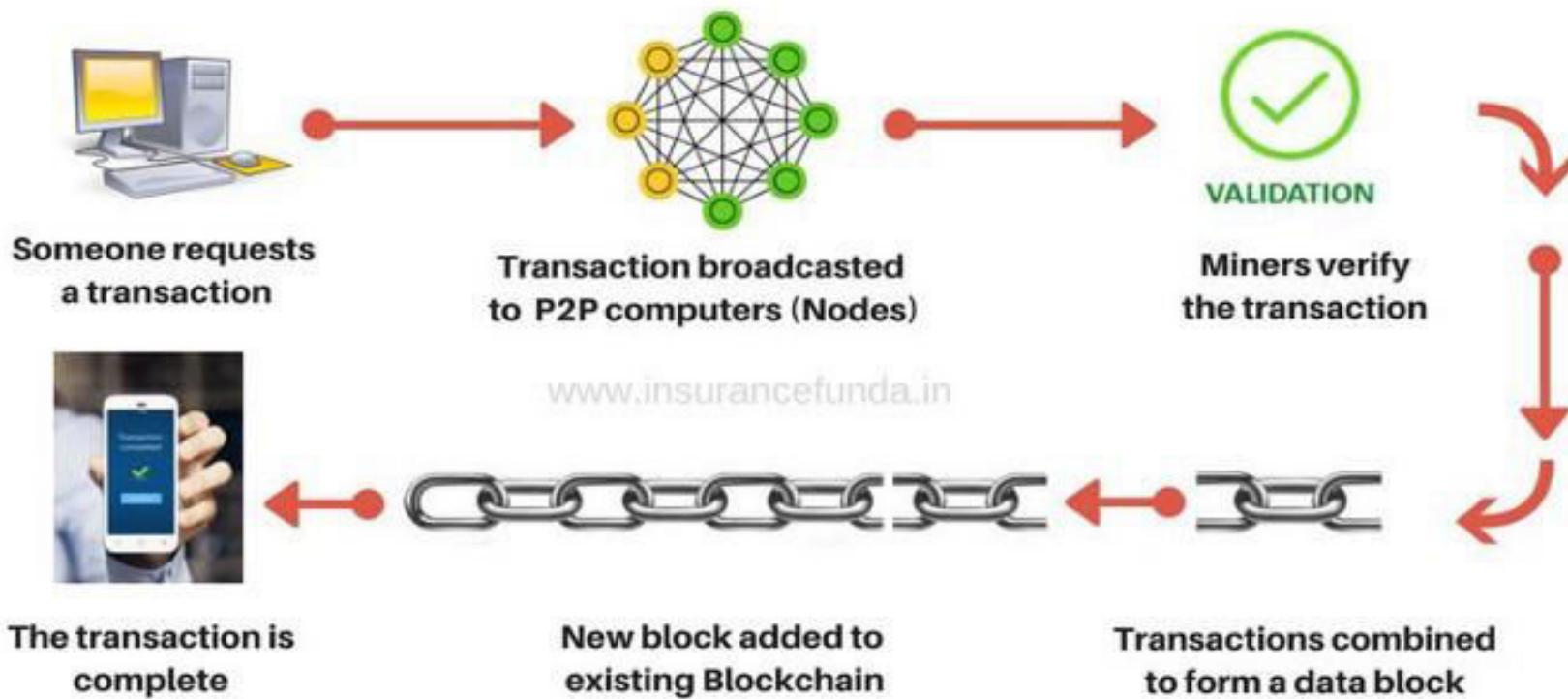


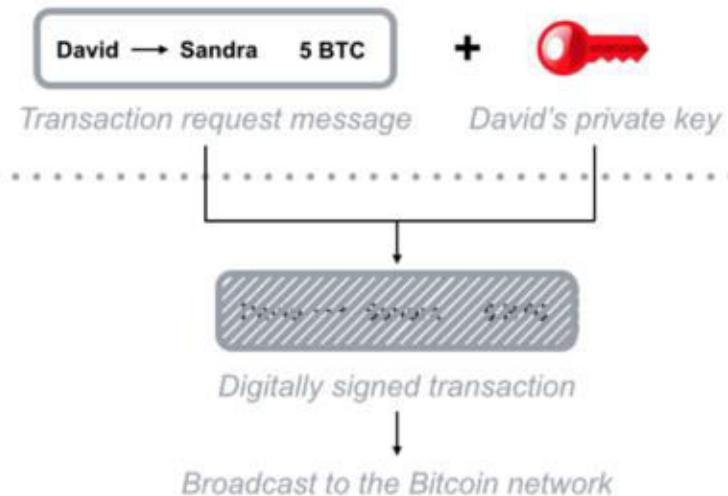
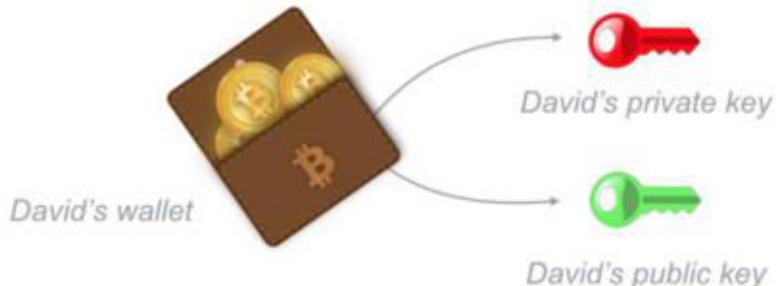
(II) Alice and bob both have added their block to the blockchain. Alice has created fork chain.



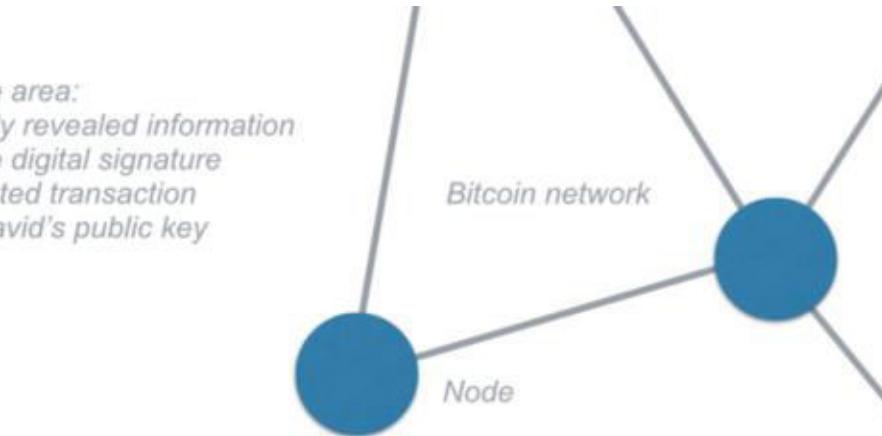
Transaction Life Cycle

HOW BITCOIN TRANSACTION WORKS





Private area:
the only revealed information
are the digital signature
encrypted transaction
and David's public key



Transaction Life Cycle

1. A user/sender sends a transaction using **wallet software** or some other interface.
2. The wallet software signs the transaction using the **sender's private key**.
3. The transaction is broadcasted to the Bitcoin network using a **flooding algorithm**.
4. **Mining** nodes include this transaction in the next block to be mined.
5. Mining starts once a miner who solves the **Proof of Work problem** broadcasts the newly mined block to the network.
6. The nodes verify the block and propagate the block further, and confirmation starts to generate.
7. Finally, the confirmations start to appear in the receiver's wallet and after approximately six confirmations, the transaction is considered finalized and confirmed.

Transaction Life Cycle(Contd...)

The API Client Implementation:

The library that we are going to use is called bitcore-lib and is published at <https://github.com/bitpay/bitcore-lib> by bitpay.

Prerequisites: bitcore-lib and bitcore-explorers

Bitcore-lib is a package that allows you to connect to the Bitcoin node and call the API endpoints. Installation is handled by the npm package manager for linux machines:

```
npm install bitcore-lib
```

Contd...

Bitcore-explorers is responsible for making **bitcoin transactions** and can be installed with:

```
npm install bitcore-explorers
```

Contd...

Codebase — Connect To the Bitcoin Node:

First of all, we need to generate a [WIF hash key \(Wallet Import Format\)](#). The WIF hash number is a way of encoding a private [ECDSA key](#) to make it easier to copy. To do it, we can use the following script:

```
var bitcore = require('bitcore-lib')

var Insight = require('bitcore-explorers').Insight

var privateKey = bitcore.PrivateKey('testnet').toWIF()

console.log(privateKey)
```

Contd...

Connecting WIF hash key to the wallet :

```
var bitcore = require('bitcore-lib');
var Insight = require('bitcore-explorers').Insight;

var privateWIFkey = 'cUw66adsvt6mrpkzqtPueZ6uQQsPURGcc5iK34DUnGqxsKt3FcKh';

var decodedPrivateKey = bitcore.PrivateKey.fromWIF(privateWIFkey)
var senderBitcoinAddress = decodedPrivateKey.toAddress()

console.log(senderBitcoinAddress)
```

And finally, we have generated a new address for the wallet that we will use for our application.

Step 1 — Request Payment:

- When the user wants to pay with Bitcoin, the system should present him an address that he should send the funds to. The **best approach** is to present **the address and the QR code**
- The new address should be generated for each user or even for each order
- The Blockchain does not limit the **number of addresses** assigned to your wallet, so for every new buyer on your website, new address should be generated. This is the application's responsibility to save this address and create a relationship in the database between it and the order.

Contd...

Example:

1. A user U1 places an order, the system generates a special address for him — A1 — and presents it.
2. A user U2 places an order, the system generates a special address for him — A2 — and presents it.
3. And so on...

If any funds are transferred and confirmed to the address A1, we will know that this is a payment which should be assigned to user U1, or somebody else wants to pay for him.

Contd...

Generate a New Wallet:

Before we generate a new address for the user, we need to create a totally new Wallet for our application. We can do it from the code level, by using the REST API which is delivered with BWS at the http address <http://localhost:3232/bws/api>.

```
var Client = require('bitcore-wallet-client');
var fs = require('fs');
var BWS_INSTANCE_URL = 'http://localhost:3232/bws/api'
var client = new Client({baseUrl: BWS_INSTANCE_URL, verbose: false});
client.createWallet("My Wallet", "MyApplication", 2, 2, {network: 'testnet'}, function(err, secret) {
  if (err) {
    console.log('error: ',err);
    return
  };
  // Handle err
  console.log('Wallet Created. Share this secret with your copayers: ' + secret);
  fs.writeFileSync(wallet.dat', client.export()); });

```

Contd...

Generate a New Address:

Now it is time to generate **new random and unique address** for our user. This address will be connected with **our Wallet**, which we created a moment ago, but this connection **will not be visible to anybody else**.

```
var Client = require('bitcore-wallet-client');
var fs = require('fs');
var BWS_INSTANCE_URL = 'http://localhost:3232/bws/api'
//secret saved from the previous exercise
var secret =
'RMDAYDCHnV7RoF9FPs8c7qVL3Eny64yFAFG57suLnwFdCjKHQ9uU5efYbDuHxgMEoWtGQtkw
RQPTbtc'
```

```
var client = new Client({ baseUrl: BWS_INSTANCE_URL, verbose: false, });
client.joinWallet(secret, "MyApplication", {}, function(err, wallet) {
  if (err) {
    console.log('error: ', err);
    return
  };
}
```

```
console.log('Joined ' + wallet.name + '!');
fs.writeFileSync(address.dat', client.export());

client.openWallet(function(err, ret) {
  if (err) { console.log('error: ', err); return };
  if (ret.wallet.status == 'complete') {
    client.createAddress({}, function(err,addr){
      if (err) {
        console.log('error: ', err);
        return;
      };
      //new address
      console.log('Return:', addr)
   });  } });
});
```

Contd...

- **joinWallet** method, which authenticates the application with the secret key. That is the same key we generated in the previous example.
- Another method, **openWallet**, opens this Wallet, and from now on, we will be able to write to the blockchain. As the only argument, it accepts the callback function where we will create a new Address.

Step 2 - Payment (user site):

- The second step is totally on the user's side, and we cannot automate that process easily. At this point, the user scans the **QR code** or copies the **address to his wallet** and makes the payment.
- When that happens, the **payment is published to the Blockchain** but **has not been included in any block yet**. It is on the pending list, and we can get information about its status.
- To do it, we can call an **API method** that delivers a **full list of all pending transactions**. Once, the transaction is mined, it will be moved from this list to the block.

Contd...

We can list all transactions for the address, and before being mined, they will have 0 confirmations. At this point, the transaction is Pending and cannot be treated as safe.

```
var explorers = require('bitcore-explorers');
var insight = new explorers.Insight('http://localhost:3001/insight-api/', 'testnet');

const exampleAddress = 'myqfgu5fZKgPBiqv7gghBPETc4yEnCdocU'

insight.requestGet('txs/?address=' + exampleAddress, function(err, response) {
  console.log(response.body)
});
```

Step 3 - Get the Transaction From the First Block:

There are two ways of getting information about the transaction now — **the first one** is to save the transaction id from the pending list and track its status, or listen to the **last published block** and get a list of transactions included in it.

Get the Latest Block

This is one of the approaches for getting information about the current payment status. If you already got the transaction ID, then you can omit this step and go directly to step four

Contd...

To get information about a block, we can use the block method in the Insight's API. This will return a full set of data including a list of transactions.

```
var explorers = require('bitcore-explorers');
var insight = new explorers.Insight('http://localhost:3001/insight-api/', 'testnet');

const exampleBlock =
'0000000000007435e1e4c72e17ae47fbdd63132950468d3dbd5d732e7fc32d8'

insight.requestGet('block/' + exampleBlock, function(err, response) {
  console.log(response.body)
});
```

Unfortunately, the [list of transactions includes only the IDs](#), so we need to go through all those IDs and get the transaction data from [another method called tx.code](#):

```
var explorers = require('bitcore-explorers');
var insight = new explorers.Insight('http://localhost:3001/insight-api/', 'testnet');
const exampleBlock =
'000000000000592e07543f0da2c21a93095827c4dc03f178da1557057b9795f'
insight.requestGet('block/' + exampleBlock, function(err, response) {
  JSON.parse(response.body).tx.forEach((transaction) => {
    insight.requestGet('tx/' + transaction, function(err, response) {
      const transactionObject = JSON.parse(response.body)
      console.log(transactionObject)
    });
  })
});
```

The code below **lists all receiver addresses** with the amount of funds that were transferred to it. This one list is included into one specific transaction:

```
var explorers = require('bitcore-explorers');
var insight = new explorers.Insight('http://localhost:3001/insight-api/', 'testnet');
const exampleBlock = '000000000000592e07543f0da2c21a93095827c4dc03f178da1557057b9795f'
insight.requestGet('block/' + exampleBlock, function(err, response) {
  JSON.parse(response.body).tx.forEach((transaction) => {
    insight.requestGet('tx/' + transaction, function(err, response) {
      const transactionObject = JSON.parse(response.body)
      transactionObject.vout.forEach((vout) => {
        console.log('TRANSFER ' + vout.value + ' BTC, to ' + vout.scriptPubKey.addresses)
      })
    });
  });
});
```

Get the Block Hash:

Now we can get information about the block, but we need to know the **block hash**, which is pretty hard to guess. That is why blocks also have order numbers.

The API provides a method that returns a block hash for a specific order number. It is called **block-index**:

```
var explorers = require('bitcore-explorers');
var insight = new explorers.Insight('http://localhost:3001/insight-api/', 'testnet');
const exampleBlock = 1261415
insight.requestGet('block-index/' + exampleBlock, function(err, response) {
  console.log('BLOCK HASH: ' + JSON.parse(response.body).blockHash)
});
```

Contd...

- After the transaction is mined in the block, it has the very first confirmation and can be treated with limited trust.
- Some of those blocks may be dropped when the next is mined. Those blocks which have not been accepted after the next block was mined are called **orphaned blocks**. This is why one confirmation may not be enough to trust the payment.

Step 4 — Confirm the Transaction

- The last step is to gather as many confirmations from the blockchain as we need to mark the transaction as secured or trusted.
- After the transaction is mined for the first time and added to the block, every next block on the stack will add one confirmation to the transaction.
- The confirmation number is nothing more than a piece of information about how many blocks have been mined after the block with this transaction.

Contd...

Each block is mined in approximately 10 minutes, the more confirmations we need to trust the transaction the more time we need to wait for it.

1 confirmation (10 minutes) for **small transactions**

3 confirmations (30 minutes) for **usual transactions with bigger value.**

6 confirmations (60 minutes) for **big transactions, mostly used for investing process in ICO or other Blockchain projects**, where the users operate with several tens of thousands of dollars. In this case, time is not as important as security and confidence are.

Contd...

Get Transaction Data

To get information about received confirmation number, we need to know the transaction ID. The tx method is meant to return complete information about the transaction:

```
var explorers = require('bitcore-explorers');
var insight = new explorers.Insight('http://localhost:3001/insight-api/', 'testnet');
```

```
const transactionId =
'84ab8aa62753a29f11adc15b6415c6fc58799e7b6f74b80bc31db2a72f6dd105'
```

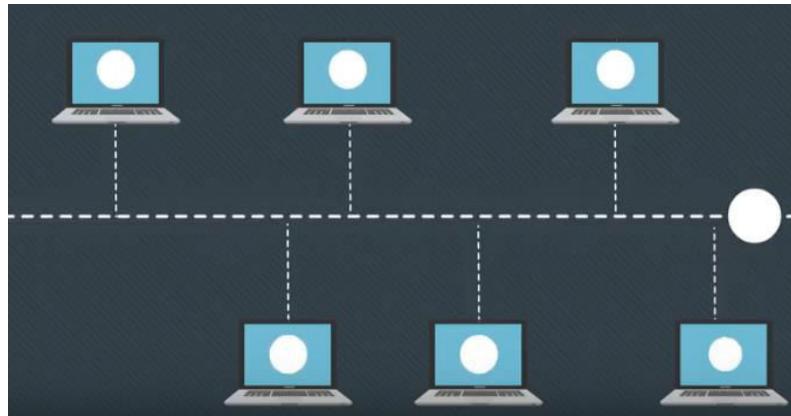
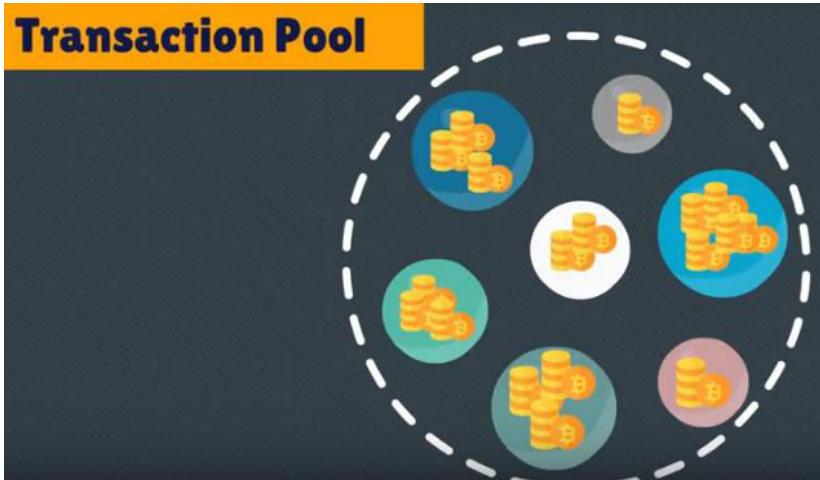
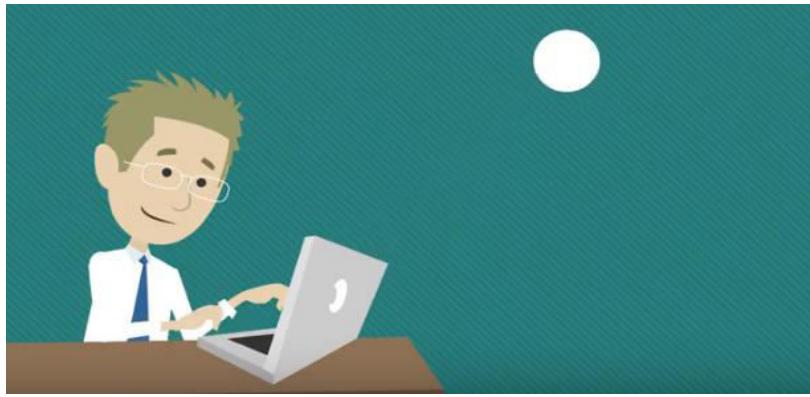
```
insight.requestGet('tx/' + transactionId, function(err, response) {
  console.log('number of confirmations for ' + transactionId + ' is ' +
JSON.parse(response.body).confirmations)
});
```

After the number of confirmations meets our expectations, we can mark the transaction as done.

How transactions are verified in blockchain?

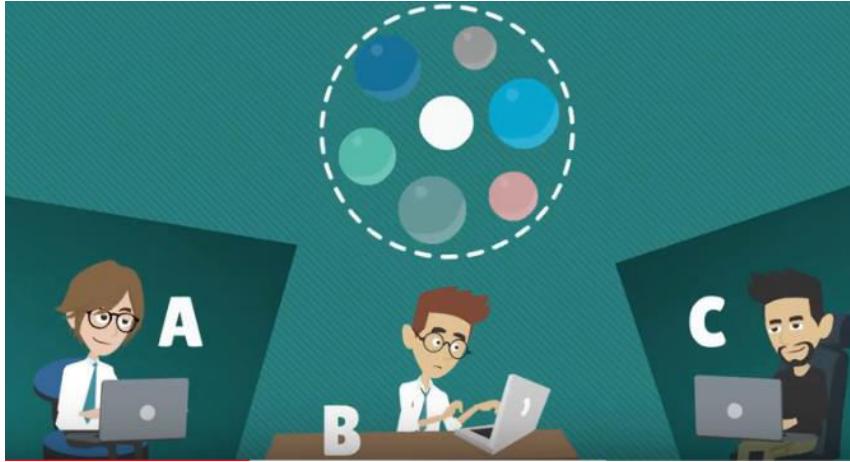
Assume Bob transferred 100\$ from one cryptocurrency exchange to another

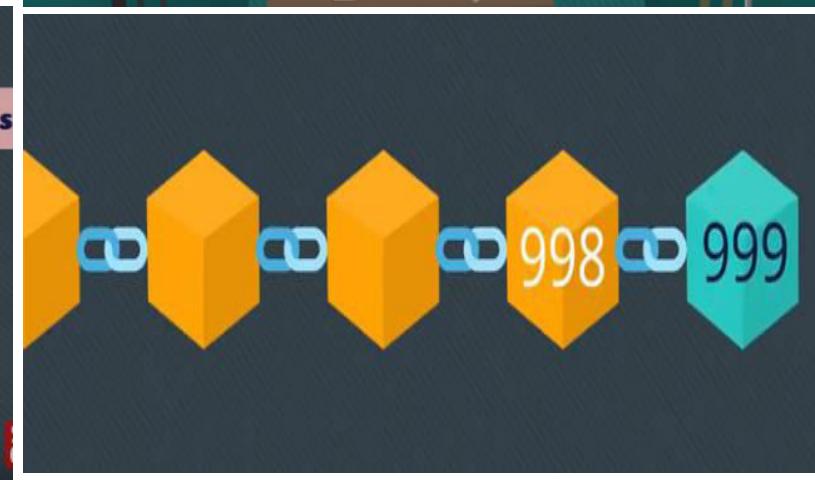


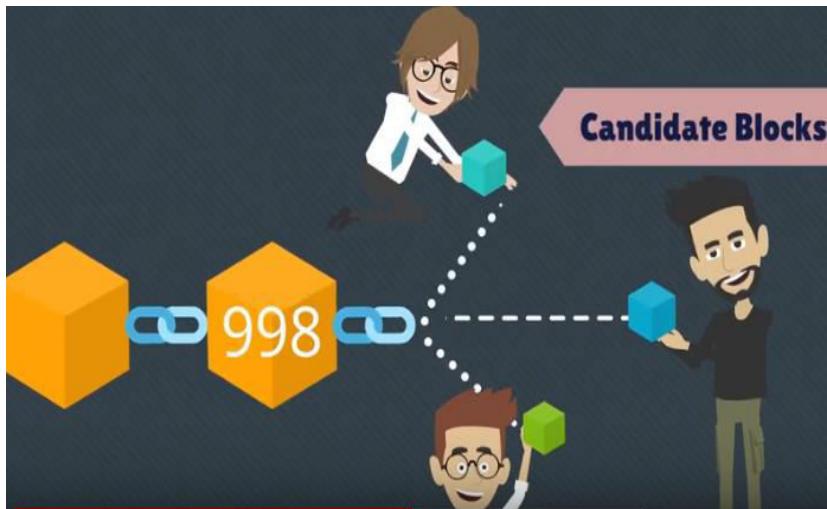
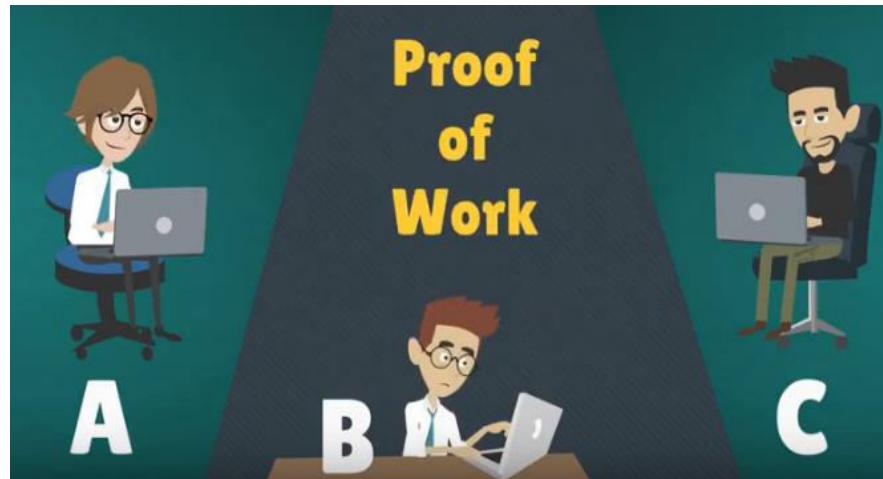


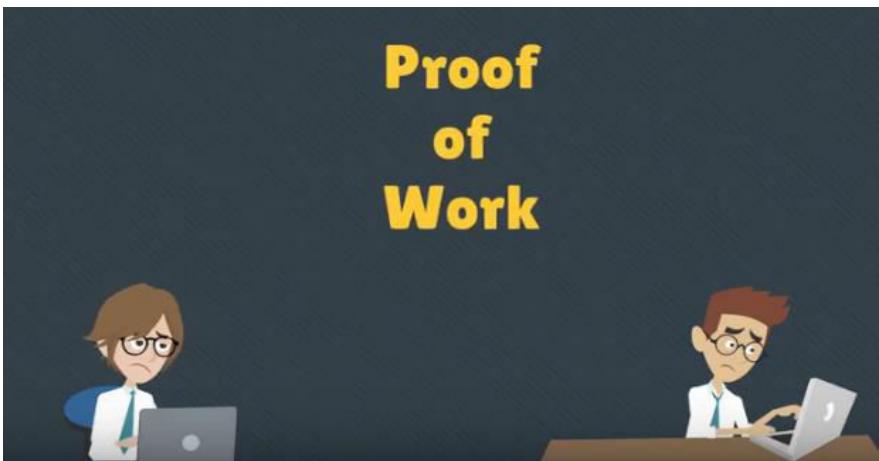
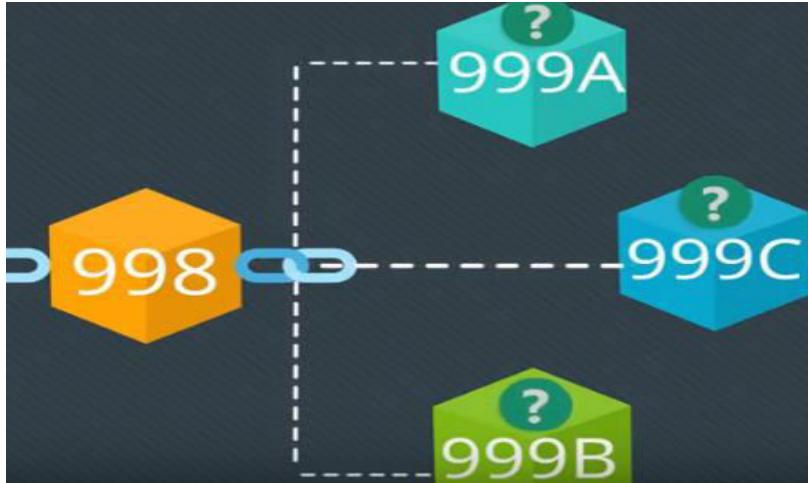
Transaction Pool

Unconfirmed

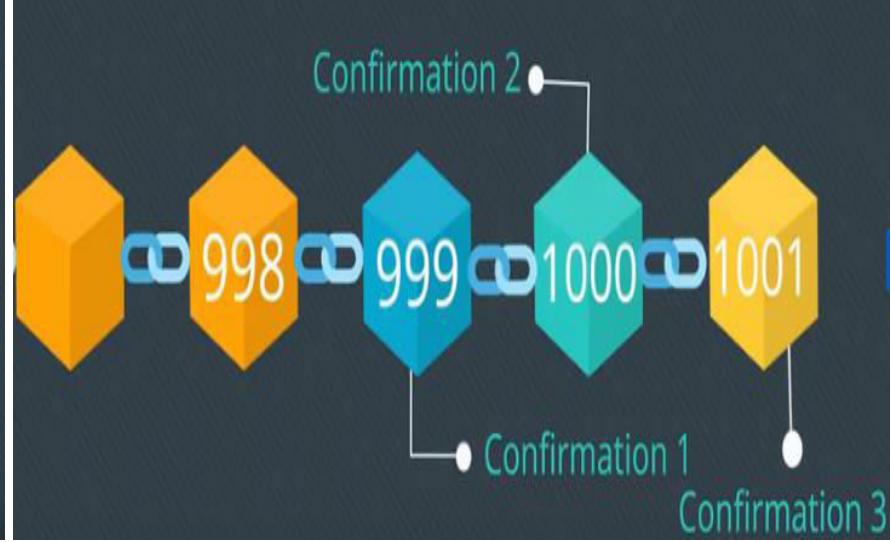








Transaction Pool



Contd...

After applying longest chain rule in the Proof-of-Work,
unconfirmed transactions will become confirmed transactions

