



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

# **BLOCKCHAIN TECHNOLOGY**

**BITS F452**

**1<sup>st</sup> Sem 2022-23**

**Lecture 1**

## **Overview of Blockchain Technology**



# Course objectives & Approach

- ❖ foundational and other essential concepts in blockchain technology
- ❖ development process of decentralized applications pertaining to verticals such as finance, supply chain, governance etc.
- ❖ current advancements in the Blockchain Technology and DAG-based distributed ledgers

## Text Book

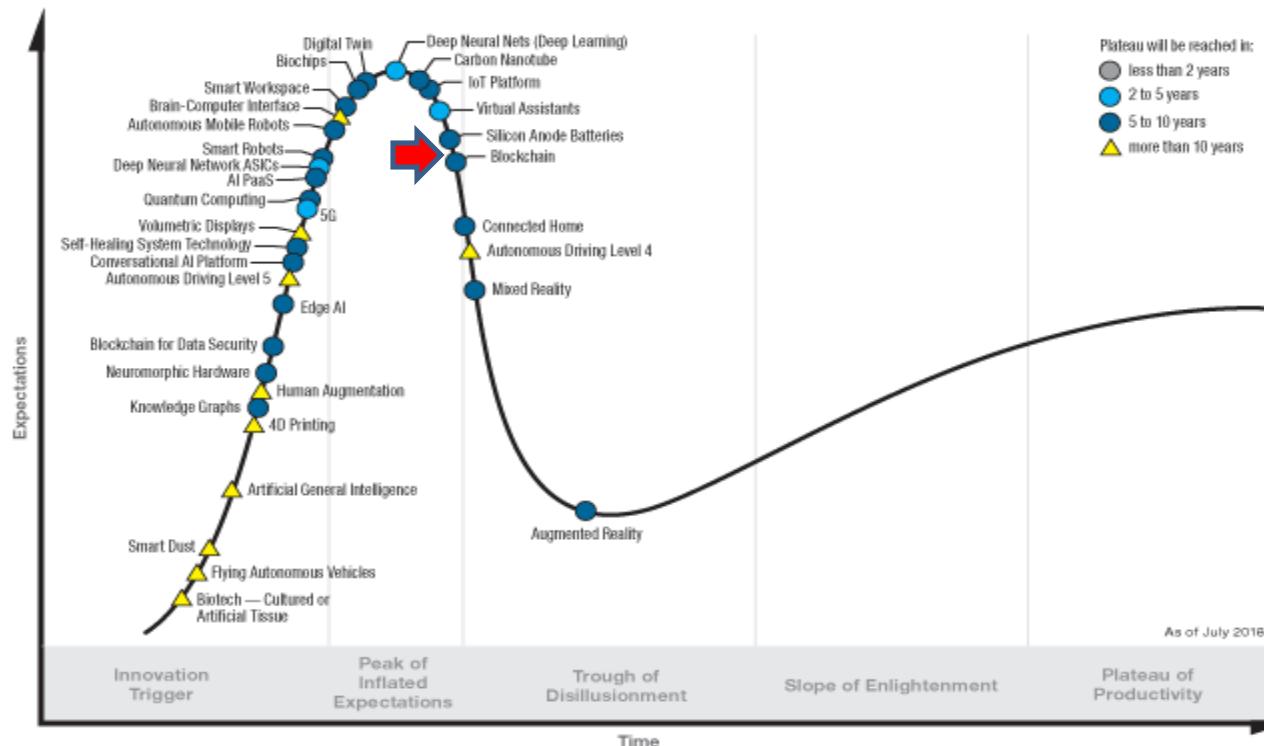
Imran Bashir, Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained, 2nd Edition, Packt Publishing, 2018.

# Overview of Blockchain Technology

- ❖ Defining Blockchain and Distributed Ledger
- ❖ Blockchain Properties: Decentralized, Transparent, Immutable and secure
- ❖ Blockchain Applications
- ❖ Types of Blockchain: public, private, and consortium based blockchains
- ❖ When to use, and when not to use Blockchain



## Hype Cycle for Emerging Technologies, 2018



[gartner.com/SmarterWithGartner](http://gartner.com/SmarterWithGartner)

Source: Gartner (August 2018)  
© 2018 Gartner, Inc. and/or its affiliates. All rights reserved.

**Gartner**

# Blockchain: The new trust paradigm

[https://en.wikipedia.org/wiki/I,\\_Pencil](https://en.wikipedia.org/wiki/I,_Pencil)



- ❖ The need for trust: extremely complex and large-scale human collaboration needed to manufacture even something as simple as a [pencil](#)
- ❖ **Transactions today are facilitated by ‘trust systems’ and intermediaries**
- ❖ These ‘trust systems’ have become increasingly complex
- ❖ **Blockchain – a new paradigm of trust**

# Blockchain: the new trust model

A <b>Database</b>	A list of records / transactions, like a ledger, that keeps growing as more entries are added;
Which is <b>Distributed</b>	Copies of the entire database are stored on multiple computers on a network, syncing within minutes / seconds;
adjustably <b>Transparent</b>	Records stored in the database may be made visible to relevant stakeholders without risk of alteration;
highly <b>Secure</b>	Malicious actors (hackers) can no longer just attack one computer and change any records;
and <b>Immutable</b>	The mathematical algorithms make it impossible to change / delete any data once recorded and accepted.

# Blockchain is a technology that...



A technology that:

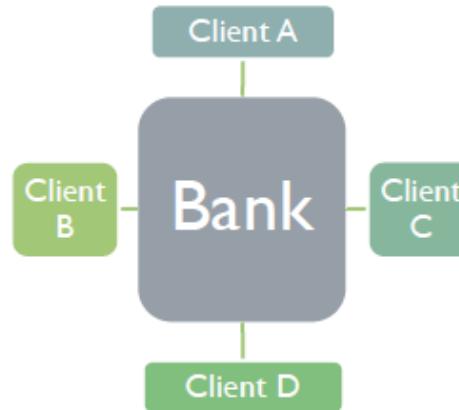
permits transactions to be gathered into blocks and recorded;

allows the resulting ledger to be accessed by different servers.

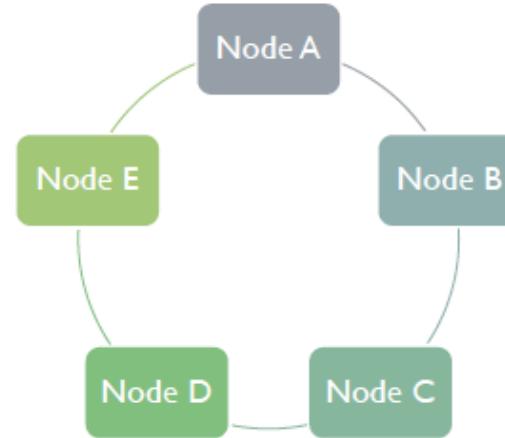
cryptographically chains blocks in chronological order; and

# What is a distributed ledger?

Centralized Ledger



Distributed Ledger



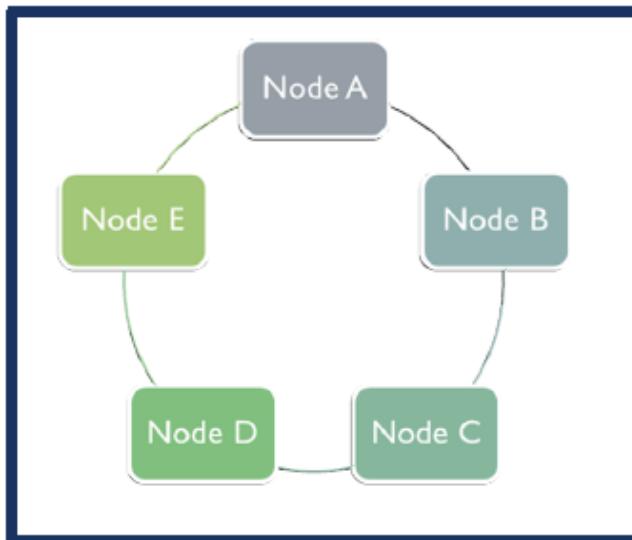
- There are multiple ledgers, but Bank holds the “golden record”
- Client B must reconcile its own ledger against that of Bank, and must convince Bank of the “true state” of the Bank ledger if discrepancies arise

- There is one ledger. All Nodes have some level of access to that ledger.
- All Nodes agree to a protocol that determines the “true state” of the ledger at any point in time. The application of this protocol is sometimes called “achieving consensus.”

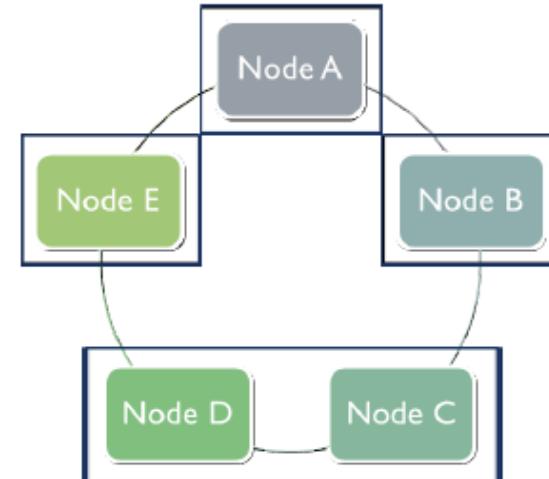
**Ledger – the technology uses an append only ledger to provide full transactional history**

# How does a distributed ledger work?

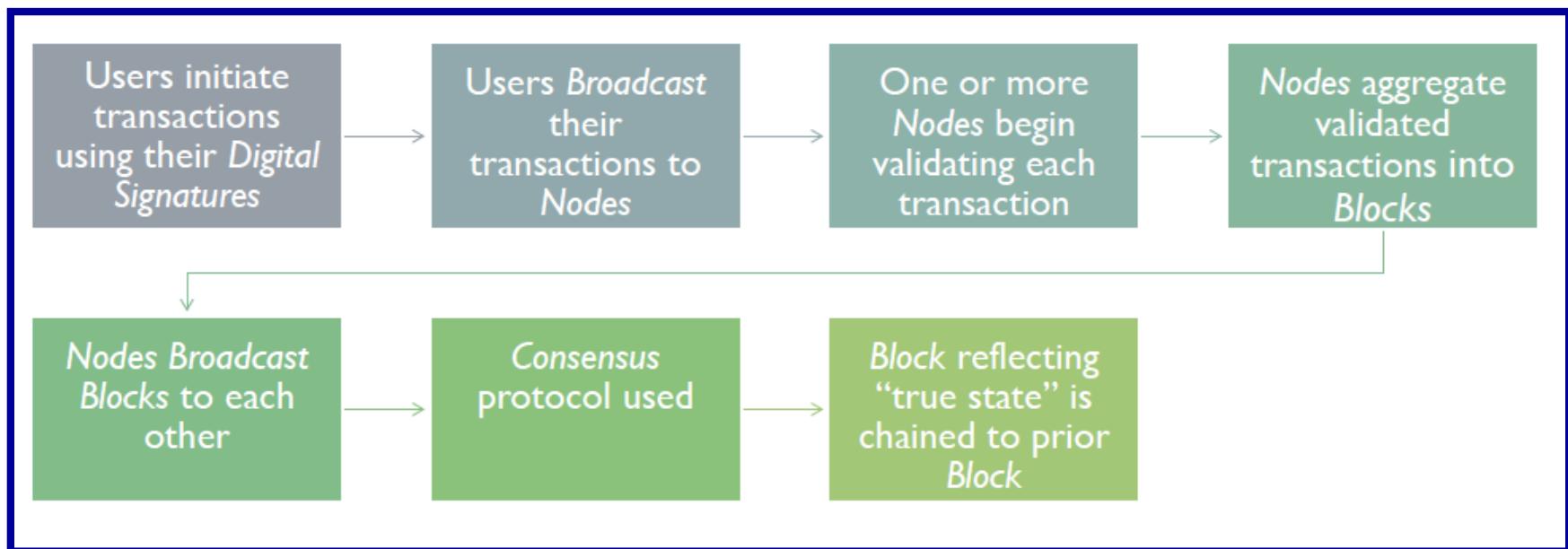
Single Entity



Multiple Entities



# Distributed ledger



# The power of distributed ledger

It can be used without a central authority by individuals or entities with no basis to trust each other

It can be used to create value or issue assets

It can be used to transfer value or the ownership of assets  
A human being or a Smart Contract can initiate the transfer

It can be used to record those transfers of value or ownership of assets  
These records may be very difficult to alter, such that they are sometimes called effectively immutable

It can be used to allow owners of assets to exercise certain rights associated with ownership, and to record the exercise of those rights.  
• Proxy Voting

# Features of blockchain

Fundamental characteristics..

.. translating into generic features

**Electronic**



**Digital or electronic in use, and easily accessible**

**Not liable to anyone**



**Doesn't require a party establish trust (entity less trust)**

**Peer to peer exchange**



**Decentralized in nature**

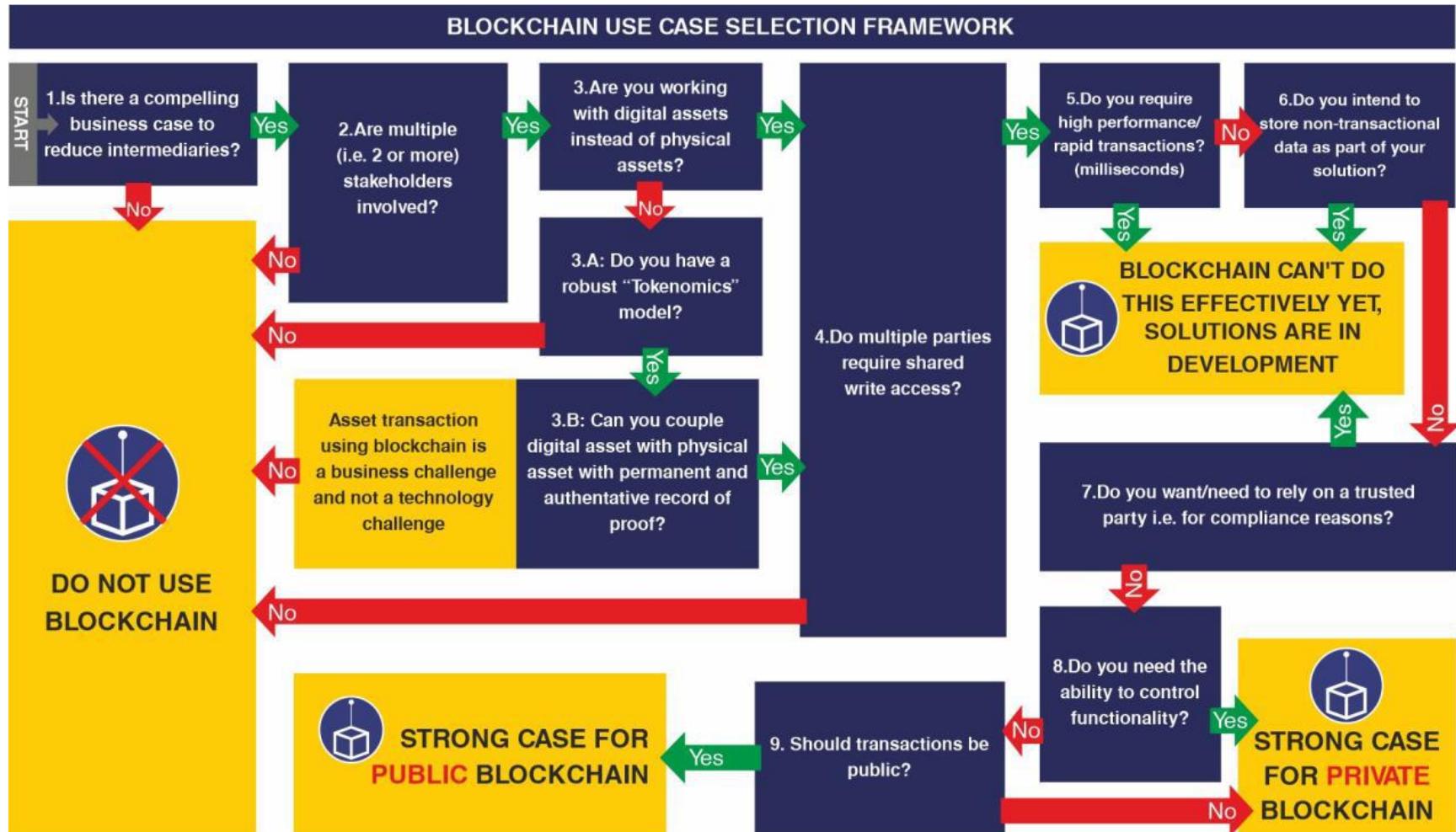
**Decentralized + entity-less trust Systems – why are they important and who should pay attention?**

# Blockchain based direct communication model

decentralized



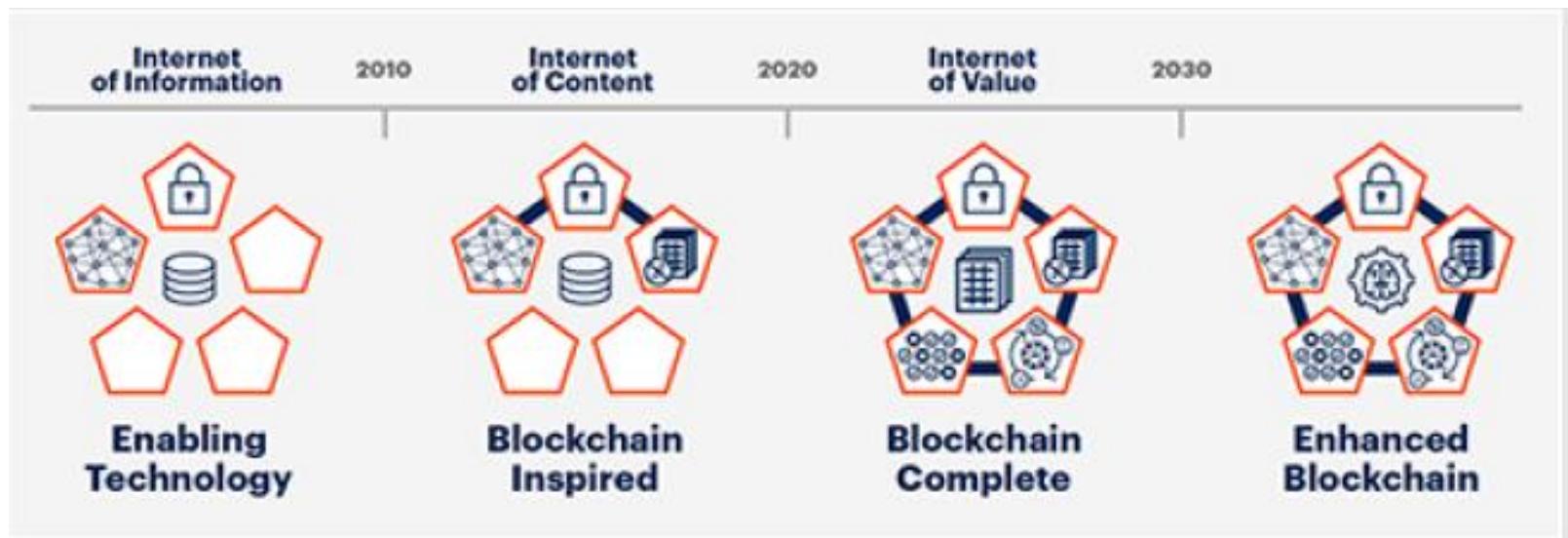
# Framework for blockchain use case evaluation



# Blockchain and India's foundational digital infrastructures

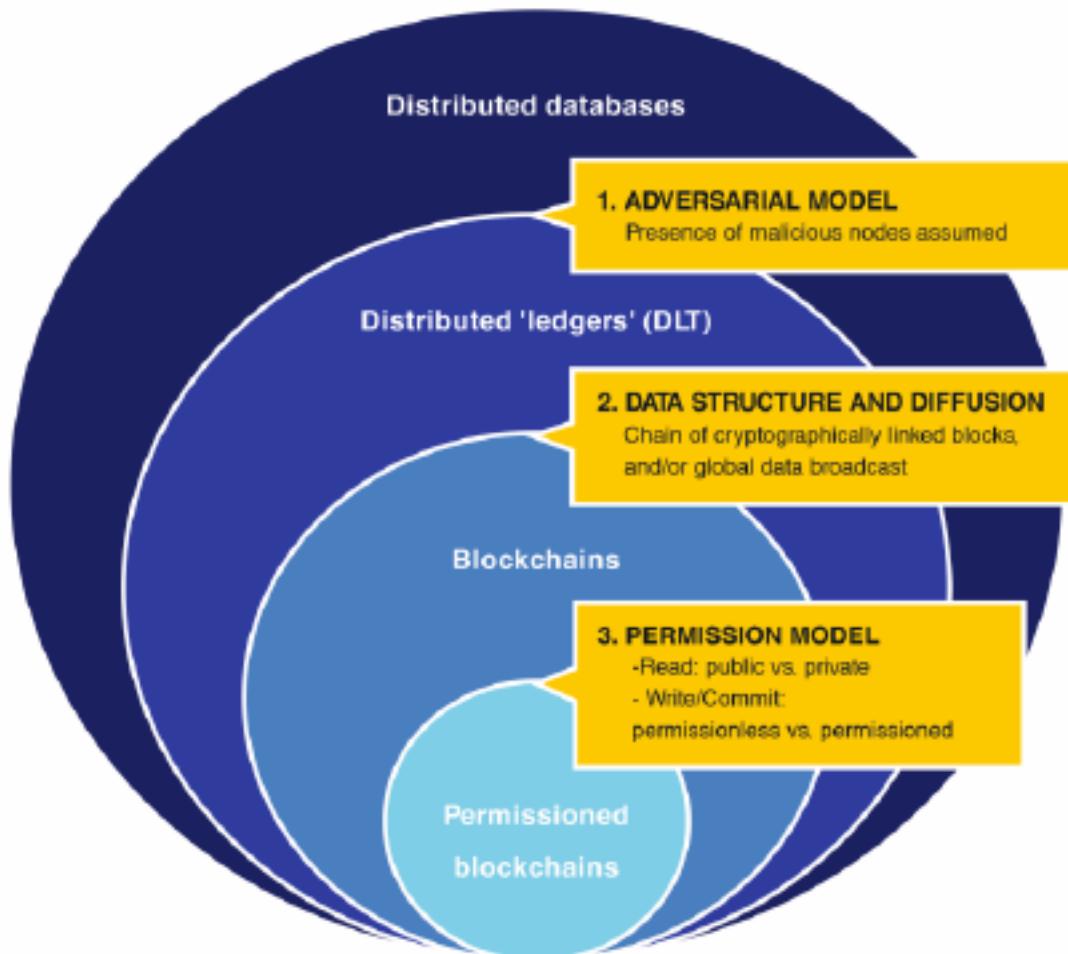
Aadhaar	<ul style="list-style-type: none"><li>• World's largest identity database with more than 1.2bn biometric identities</li><li>• More than 25 million authentications per day</li></ul>
Unified Payments Interface (UPI)	<ul style="list-style-type: none"><li>• World's most sophisticated digital payments system</li><li>• 1.3bn transactions processed in December 2019</li></ul>
Goods and Services Tax Network (GSTN)	<ul style="list-style-type: none"><li>• More than 400 million returns filed</li><li>• More than 800 million invoices uploaded</li></ul>
PM-JAY	<ul style="list-style-type: none"><li>• World's largest healthcare initiative with ~500 million beneficiaries covered</li><li>• ~119 million e-cards issued so far, ~8 million hospital admissions</li></ul>

# Gartner Blockchain spectrum



Over the decades

# Distributed Technologies



# Blockchain Technology - components



<b>Consensus Protocols</b>	Computer algorithms that define the modality of how the blockchain based system defines what is the correct updated state of the database. The simplest version of this would be a simple majority amongst nodes.
<b>Cryptography</b>	Method of protecting information and communications through the use of codes so that only the recipient can read it. In computer science, cryptography refers to secure communication techniques based on algorithms which transforms confidential messages (like email, credit card transactions, web browsing) in ways that are hard to decipher by third parties.
<b>Hashes</b>	Mathematical functions that convert data of indeterminate length to a 'fingerprint' of a fixed length. It is astronomically unlikely for two different sets of data to have the same 'hash'.
<b>Merkle Tree</b>	Structure (also used in BitTorrent, Git, Bitcoin and Ethereum) that summarises data of all related transactions in a block by producing a digital fingerprint in the form of a hash (or a transaction ID) for each transaction and thereafter for every pair of transactions until only one unique ID/hash is left (called the Root Hash/Merkle Root). The structure is built from the bottom up from hashes/ Transaction IDs of individual transactions and tests whether a specific transaction is included in the block or not. It records transactions in a chronological order and can verify whether the record has been altered or tampered with, or whether the record has been branched or forked.
<b>Mining</b>	Refers to the actions nodes take to authenticate transactions. Miners are economically incentivized to spend resources for maintaining the network by a reward of tokens which are generated by the distributed network.
<b>Nodes</b>	Entities a blockchain system that maintain a copy of the most updated state of the blockchain database and participate in the authentication of new changes. However, all nodes may not store a full copy of a blockchain or validate transactions.
<b>Smart Contracts</b>	Represent an if-else construct which enable blockchain based systems to autonomously record a transaction if certain pre-requisites are completed.
<b>Token(s)</b>	Essential components of most public blockchains. They are a unique asset class which not only denote ownership of the network (like shares of a company) but also form the basic unit of value exchange.

# What is Blockchain?

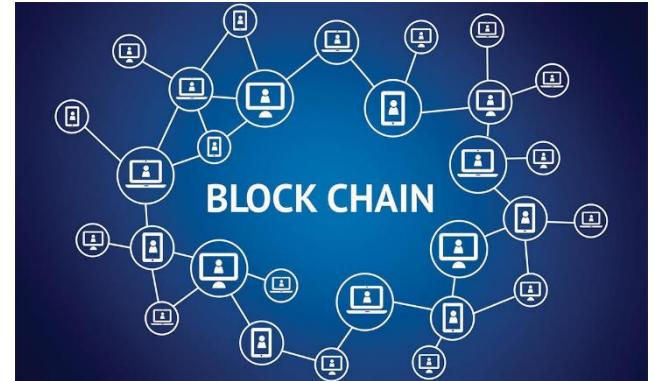
---

- ❖ Blockchains are tamper evident and tamper resistant **digital ledgers** implemented in a distributed fashion (i.e., without a central repository) and usually without a central authority (i.e., a bank, company or government)
- ❖ Blockchains enable a community of users to record transactions in a shared ledger within that community, such that under normal operation of the blockchain network no transaction can be changed once published

# Blockchain is

## ❖ A Linked List

- ❖ Replicated
- ❖ Distributed
- ❖ Consistency maintained by Consensus
- ❖ Cryptographically linked
- ❖ Cryptographically assured integrity of data



## ❖ Used as

- ❖ Immutable Ledger of events, transactions or time stamped data
- ❖ Tamper resistant log
- ❖ Platform to Create and Transact in Cryptocurrency
- ❖ log of events/transactions unrelated to currency
- ❖ SCM
- ❖ healthcare
- ❖ IoT

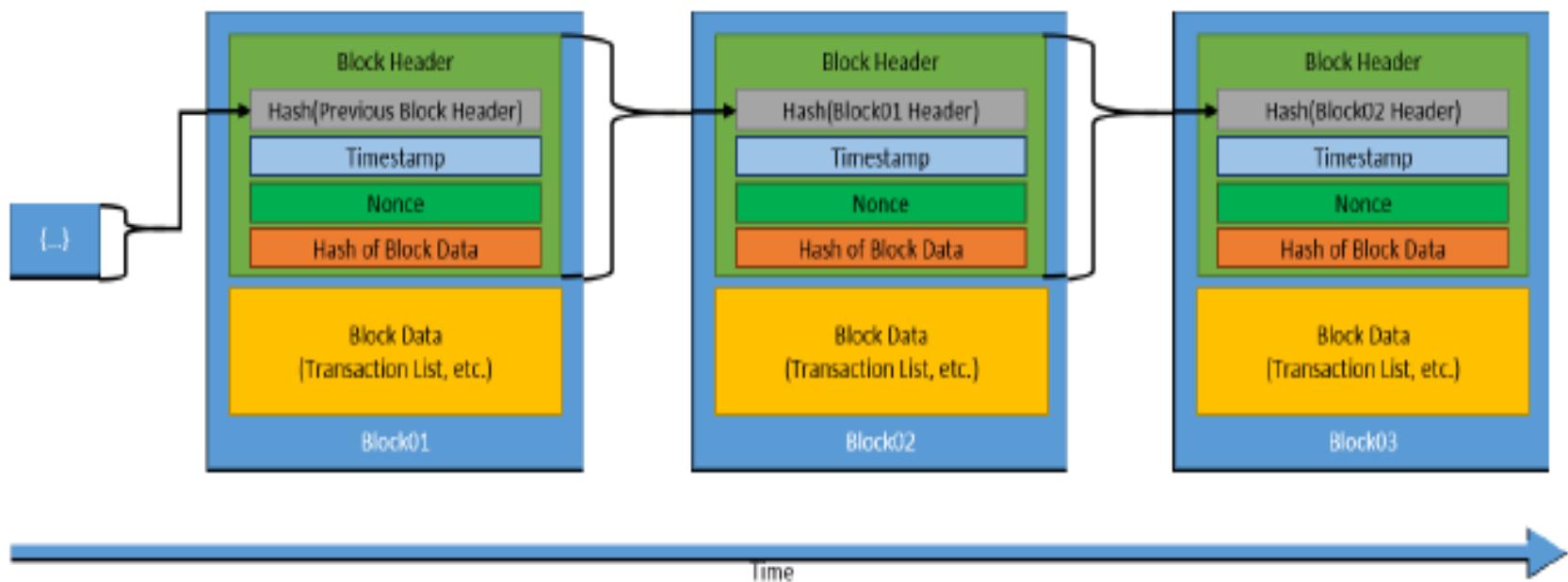
# Blockchain

---

- ❖ Blockchains are distributed digital ledgers of cryptographically signed transactions that are grouped into blocks
  - ❖ Each block is cryptographically linked to the previous one (making it tamper evident) after validation and undergoing a consensus decision
  - ❖ As new blocks are added, older blocks become more difficult to modify (creating tamper resistance)
  - ❖ New blocks are replicated across copies of the ledger within the network, and any conflicts are resolved automatically using established rules
-

# How do we depict a blockchain?

<http://dailyblockchain.github.io/>



# Blockchain Terminology

## Block

- ❖ Block is the most fundamental component in the structure of a blockchain and serves as the single section comprising discrete data.
- ❖ The blocks generally include a list of actions or transactions that should be performed during data processing in the block.

## Block (Canonical)

- ❖ A canonical block is one that has been incorporated in the primary blockchain. The canonical block is referenced either directly or indirectly by future blocks.
- ❖ Non-canonical blocks which might have validity could be rejected in favor of canonical blocks.

## Block (Genesis)

- ❖ The first block in a blockchain structure is known as the genesis block. The block height for the genesis block is zero. All the other blocks in the blockchain are linked intrinsically to the genesis block.
- ❖ It is possible to configure genesis blocks for creating a fork of a chain of purposes, including specification of different block parameters or pre-loading accounts with tokens for test networks.

# Blockchain Technology: Generations

## Blockchain 1.0

- ❖ The first generation of blockchain technology, known as Blockchain 1.0, emphasized particularly executing simple token transactions
- ❖ The chains in Blockchain 1.0 are restricted in terms of scope and ability

## Blockchain 2.0

- ❖ The second generation of blockchain technology, i.e., Blockchain 2.0, focused on enabling the functionalities of smart contracts and generalized processing.
- ❖ The chains in Blockchain 2.0 are developed with Turing-complete programming languages with a broad range of capabilities, other than the basic peer-to-peer (P2P) value exchange

## Blockchain 3.0

- ❖ The new generation of blockchain technology presently emphasizes achieving better interoperability and scalability with blockchain applications
- ❖ With Blockchain 3.0, the chains under development have the potential for improving the use of smart contracts

## Generation X (Blockchain X)

- ❖ Blockchain singularity

# Some more terms

## Block Depth

- ❖ Block depth refers to the **position index of a block in the blockchain with respect to the most recently added block.**
- ❖ For example, a block that is 3 blocks before the last added block will have a block depth of 3.

## Block Height

- ❖ Block height is the **position index of a block with respect to the genesis block.**
- ❖ For example, the second block added to a chain will have a block height of 2.

## Block Explorer

- ❖ Block explorer is the software or GUI graphical user interface, which helps users in reading and analyzing data on a blockchain.

## Block Reward

- ❖ Blockchains with native cryptocurrency allow miners to allocate a specific number of tokens for generating spontaneously and sending to desired address.
- ❖ The reward compensates for the miner's support in building a block and the network alongside incentivizing other miners for joining the network.

# What blockchains do?

---

- ❖ removes central control while maintaining data integrity to have a large distributed network of independent users
-

# Why blockchains matter?

---

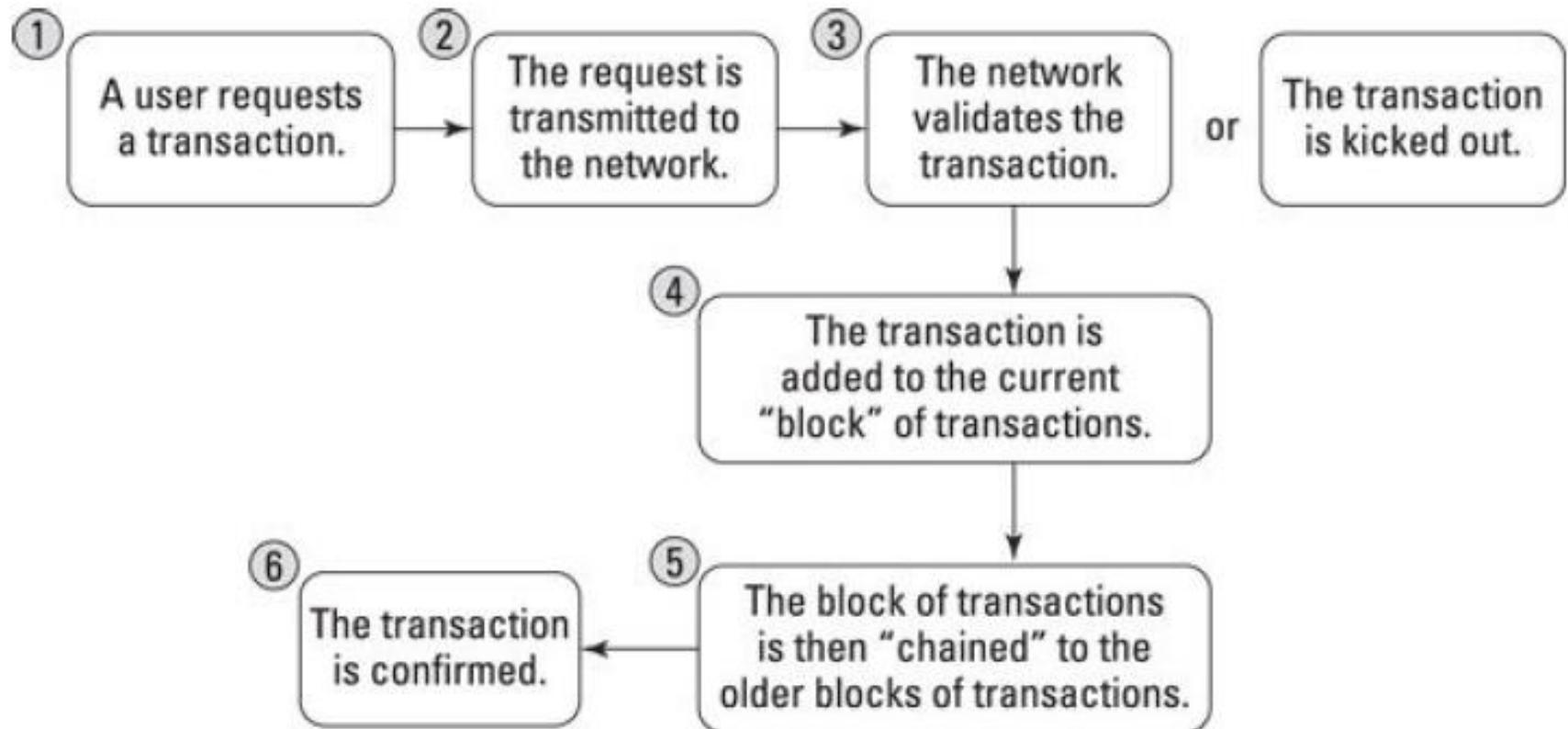
- ❖ Blockchains are now recognized as the “[fifth evolution](#)” of computing, the missing trust layer for the Internet
  - ❖ Blockchains can create trust in digital data (written into a blockchain database, it is nearly impossible to remove or change it)
-

# The Structure of Blockchains

- ❖ **Block:** A list of transactions recorded into a ledger over a given period; size, period, and triggering event for blocks being different for every blockchain
- ❖ **Chain:** A hash that links one block to another, mathematically “chaining” them together
- ❖ **Network:** The network is composed of “full nodes” located all over the world and can be operated by anyone

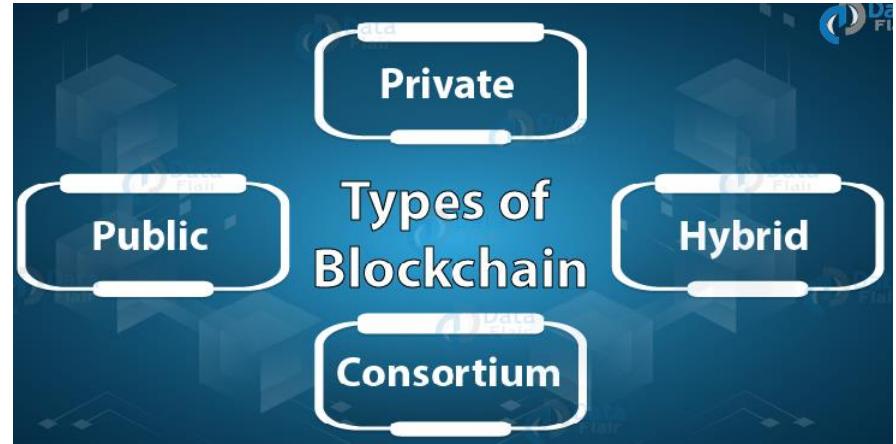
*A full node is a node that fully validates transactions and blocks. Almost all full nodes also support the network by accepting transactions and blocks from other full nodes, validating those transactions and blocks, and then relaying them to further full nodes.”*

# Blockchain Life Cycle



**Actors: users, miners, developers, or community members**

# Types of blockchain



- ❖ Every blockchain consists of a cluster of nodes functioning on a peer-to-peer (P2P) network system
- ❖ Every node in a network has a copy of the shared ledger which gets updated timely
- ❖ Each node can verify transactions, initiate or receive transactions and create blocks

# What is a public blockchain?

---

- ❖ designed to be **fully decentralized**, with no one individual or entity controlling which transactions are recorded in the blockchain or the order in which they are processed
  - ❖ highly **censorship-resistant**, since anyone is open to join the network, regardless of location, nationality, etc.,
  - ❖ have a **token** associated with them that is typically designed to incentivize and reward participants in the network
-

# What is a private blockchain?

## (permissioned blockchains)



- ❖ Participants need **consent** to join the networks
- ❖ Transactions are private and are only available to ecosystem participants that have been given permission to join the network
- ❖ more **centralized** than public blockchains

## WHAT IS PUBLIC BLOCKCHAIN?



In a public blockchain network everyone has access to the network and can take part in consensus.



## WHAT IS PRIVATE BLOCKCHAIN?



In a private blockchain only a single organization will have access and authority over the network. It's basically a partial decentralized system.



## THE COMPARISON TABLE

	PUBLIC BLOCKCHAIN	PRIVATE BLOCKCHAIN
Access	Anyone	Single organization
Authority	Decentralized	Partially decentralized
Transaction Speed	Slow	Fast
Consensus	Permissionless	Permissioned
Efficiency	Low	High
Data Handling	Read and Write access for anyone	Read and write for a single organization
Immutability	Full	Partial

# What is a consortium blockchain?

---

- ❖ **collaborative** model; offers some of the best use cases for the benefits of blockchain, bringing together a group of "frenemies"- businesses who work together but also compete against each other
- ❖ more **efficient**, both individually and collectively, by collaborating on some aspects of their business
- ❖ **participants** could include anyone from central banks, to governments, to supply chains

# What is a hybrid blockchain?

---

- ❖ connect with other blockchain protocols; allowing for a **multi-chain network of blockchains**
  - ❖ **transparency** without having to sacrifice **security** and **privacy**
  - ❖ multiple public blockchains at once increases the security of transactions, as they benefit from the combined **hashpower** being applied to the public chains
-

# Permissionless and Permissioned Blockchains

Blockchain-  
architecture options

Architecture based on read, write, or commit  
permissions granted to the participants

Architecture based on ownership of the data infrastructure	Architecture based on read, write, or commit permissions granted to the participants		
	Public	Permissionless	Permissioned
Private	<ul style="list-style-type: none"> <li>● Only authorized participants can join, read, and write</li> <li>● Hosted on private servers</li> <li>● <b>High scalability</b></li> </ul>	<ul style="list-style-type: none"> <li>● Anyone can join, read, write, and commit</li> <li>● Hosted on public servers</li> <li>● Anonymous, highly resilient</li> <li>● <b>Low scalability</b></li> </ul>	<ul style="list-style-type: none"> <li>● Anyone can join and read</li> <li>● Only authorized and known participants can write and commit</li> <li>● <b>Medium scalability</b></li> </ul>

---

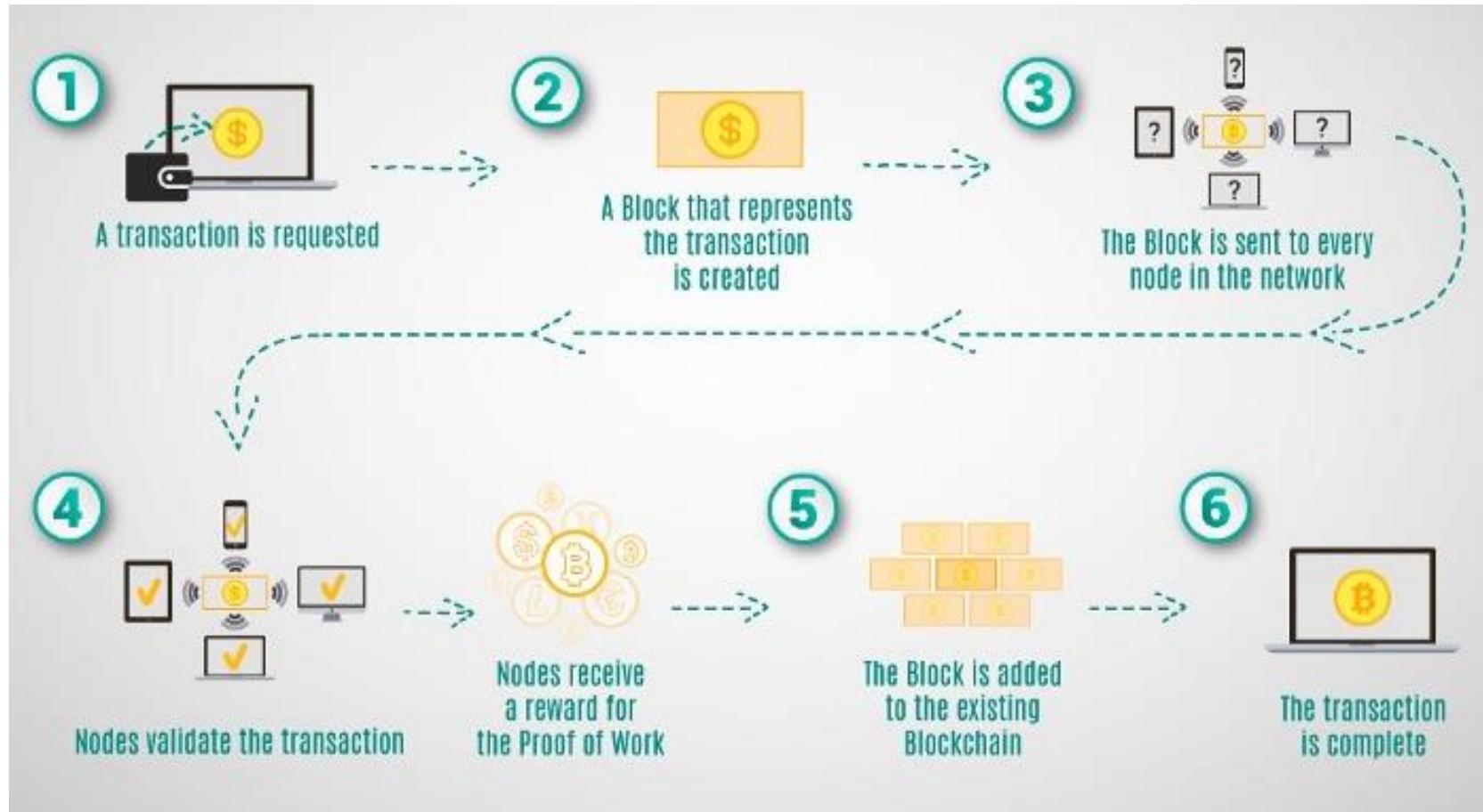
# Blockchain: The technology

# Core Components of Blockchain Architecture

---

- ❖ **Node** — user or computer within the blockchain
- ❖ **Transaction** — smallest building block of a blockchain system
- ❖ **Block** — a data structure used for keeping a set of transactions which is distributed to all nodes in the network
- ❖ **Chain** — a sequence of blocks in a specific order
- ❖ **Miners** — specific nodes which perform the block verification process
- ❖ **Consensus**— a set of rules and arrangements to carry out blockchain operations

# How the blockchain works?



# What is actually a blockchain “block”?



- ❖ certain data
- ❖ the hash of the block
- ❖ the hash from the previous block

- ❖ **Blockchain network** — refers to the application's infrastructure placed within a particular environment inside one, or a few, organizations.
- ❖ **Blockchain code** — refers to the tasks and goals this blockchain solution has been developed to perform.

# Key Characteristics of Blockchain

- ❖ **Cryptography** (secure communication/info techniques)
- ❖ **Immutability** (ability to remain unchanged, unaltered and indelible)
- ❖ **Provenance** (refers to the tracking mechanism where every single change made to the original data is tracked and versioned)
- ❖ **Decentralization** (transfer of control to multiple locations/entities)
- ❖ **Anonymity** (the ability for parties to exchange data without disclosing any off-chain identity information or other transactions they have done)
- ❖ **Transparency** (entirely traceable and easy to maintain)

# Top blockchain platforms - 2022

- ❖ [XDC Network](#)
- ❖ [Tezos](#)
- ❖ [Hyperledger Fabric](#)
- ❖ [Hyperledger Sawtooth](#)
- ❖ [Stellar](#)
- ❖ [EOS](#)
- ❖ [Corda](#)
- ❖ [Klaytn](#)
- ❖ [Tron](#)
- ❖ [Hedera Hashgraph](#)
- ❖ [Ethereum](#)



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

# **BLOCKCHAIN TECHNOLOGY**

**BITS F452**

**1<sup>st</sup> Sem 2022-23**

**Lecture 2**

**Centralized, Decentralized and  
Distributed Systems....**



# Blockchain Environment

- 
- ❖ Centralized, Decentralized and Distributed Systems
  - ❖ Decentralization vs distributed
  - ❖ P2P systems, properties of P2P systems, P2P communication architecture
  - ❖ P2P network applications: File sharing, P2P network for blockchain

# Distributed Systems

- ❖ At the core of blockchain technology is distributed systems
- ❖ Precisely blockchain is a decentralized distributed system

Distributed systems are a computing paradigm whereby two or more nodes work with each other in a coordinated fashion in order to achieve a common outcome and it is modeled in such a way that end users see it as a **single logical platform**.

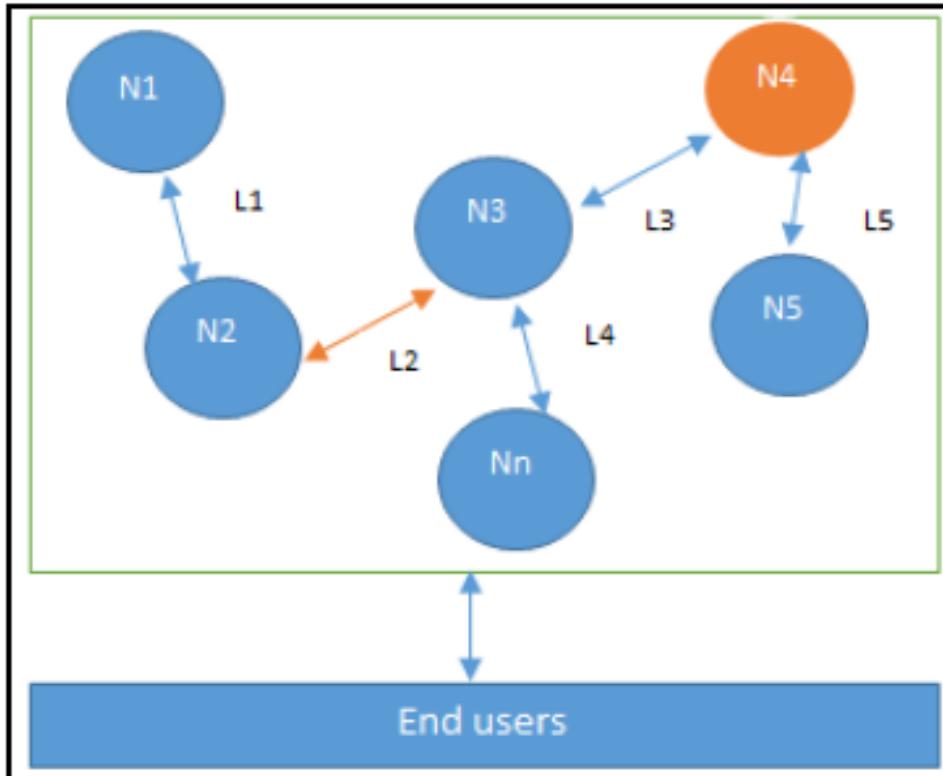
# Distributed systems and Byzantine nodes



- ❖ A node can be defined as an individual player in a distributed system
- ❖ All nodes are capable of sending and receiving messages to and from each other
- ❖ Nodes can be honest, faulty, or malicious and have their own memory and processor
- ❖ A node that can exhibit arbitrary behavior is also known as a Byzantine node
- ❖ This arbitrary behavior can be intentionally malicious, which is detrimental to the operation of the network
- ❖ Generally, any unexpected behavior of a node on the network can be categorized as **Byzantine**, a term which **arbitrarily encompasses any behavior that is unexpected or malicious**

# Design of a distributed system

The main challenge in distributed system design is coordination between nodes and fault tolerance.



Design of a distributed system; N4 is a Byzantine node, L2 is broken or a slow network link

# CAP Theorem (Consistency, Availability, Partition Tolerance)



- ❖ CAP theorem states that a distributed system cannot have all much desired properties simultaneously
- ❖ Also known as **Brewer's theorem**, introduced originally by Eric Brewer as a conjecture in 1998; in 2002 it was proved as a theorem by Seth Gilbert and Nancy Lynch.

The theorem states that any distributed system cannot have Consistency, Availability, and Partition tolerance simultaneously:

- Consistency** is a property that ensures that all nodes in a distributed system have a single latest copy of data
- Availability** means that the system is up, accessible for use, and is accepting incoming requests and responding with data without any failures as and when required
- Partition tolerance** ensures that if a group of nodes fails the distributed system still continues to operate correctly

# CAP Theorem in the blockchain context

---

- ❑ It has been proven that a distributed system cannot have all the afore mentioned three properties at the same time
  - ❑ This is strange because somehow **blockchain** manages to achieve all these properties (which we will see later)
  - ❑ To achieve **fault tolerance**, **replication** is used
  - ❑ **Consistency** is achieved using **consensus algorithms** to ensure that all nodes have the same copy of data. This is also called **state machine replication**
  - ❑ **Blockchain** is basically a method to achieve state machine replication
-

# Two types of faults

---

In general there are two types of fault that a node can experience:

- ❖ a faulty node has simply crashed
- ❖ the faulty node can exhibit malicious or inconsistent behavior arbitrarily;

The second type is difficult to deal with since it can cause confusion due to misleading information

# Byzantine Generals problem

---

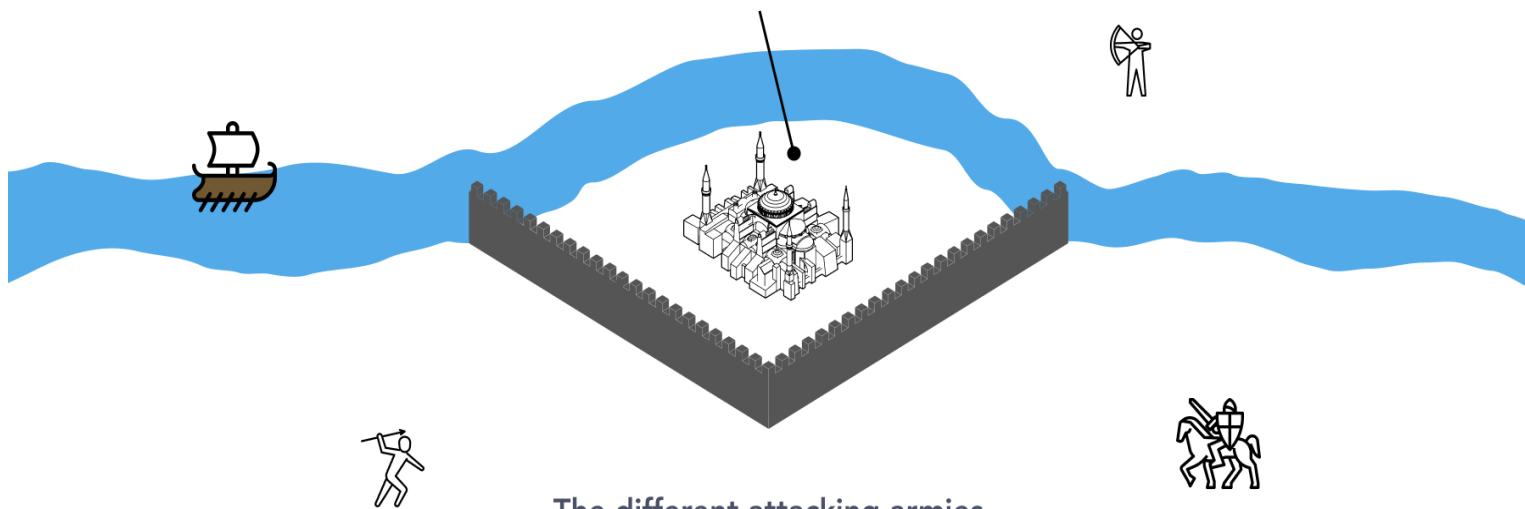
The scenario:

- ❖ A group of army generals who are leading different parts of the Byzantine army are planning to attack or retreat from a city.
  - ❖ The only way of communication between them is a messenger and they need to agree to attack at the same time in order to win.
  - ❖ The issue is that one or more generals can be traitors and can communicate a misleading message.
  - ❖ Therefore there is a need to find a viable mechanism that allows agreement between generals even in the presence of treacherous generals so that the attack can still take place at the same time
-

# The Byzantine Generals Problem

A game theory problem: How do decentralized parties arrive at consensus without a trusted central party?

The defender can intercept communication between attackers and send false messages.



The different attacking armies only win if they all attack at the same time, but they have no secure communications.

# Analogy with distributed systems

As an analogy with distributed systems, generals can be considered as **nodes**, traitors can be considered **Byzantine** (malicious) nodes, and the messenger can be thought of as a channel of communication between the generals.

# Solution to Byzantine Generals problem

---

- ❖ Solved in 1999 by Castro and Liskov who presented the **Practical Byzantine Fault Tolerance (PBFT)** algorithm
  - ❖ In 2009, the first practical implementation was made with the invention of bitcoin where the **Proof of Work (PoW)** algorithm was used
  - ❖ **PoW** developed as a mechanism to achieve consensus
-

# Consensus

---

- ❖ process of agreement between distrusting nodes on a final state of data
  - ❖ concept of achieving consensus between multiple nodes when they need to agree on a single value is known as **distributed consensus**
-

# Consensus mechanisms



- ❖ A consensus mechanism is a set of steps that are taken by all, or most, nodes in order to agree on a proposed state or value

## Requirements of consensus mechanisms

- Agreement:** All honest nodes decide on the same value.
- Termination:** All honest nodes terminate execution of the consensus process and eventually reach a decision.
- Validity:** The value agreed upon by all honest nodes must be the same as the initial value proposed by at least one honest node.
- Fault tolerant:** The consensus algorithm should be able to run in the presence of faulty or malicious nodes (Byzantine nodes).
- Integrity:** This is a requirement where no node makes the decision more than once. The nodes make decisions only once in a single consensus cycle.

# Types of consensus mechanisms

---

- ❖ **Byzantine fault tolerance-based:**

No compute intensive operations such as partial hash inversion; relies on a simple scheme of nodes that are publishing **signed messages**; eventually, when a certain number of messages are received, then an agreement is reached

- ❖ **Leader-based consensus mechanisms:**

nodes compete for the ***leader-election lottery*** and the node that wins it proposes a final value

# Practical implementations

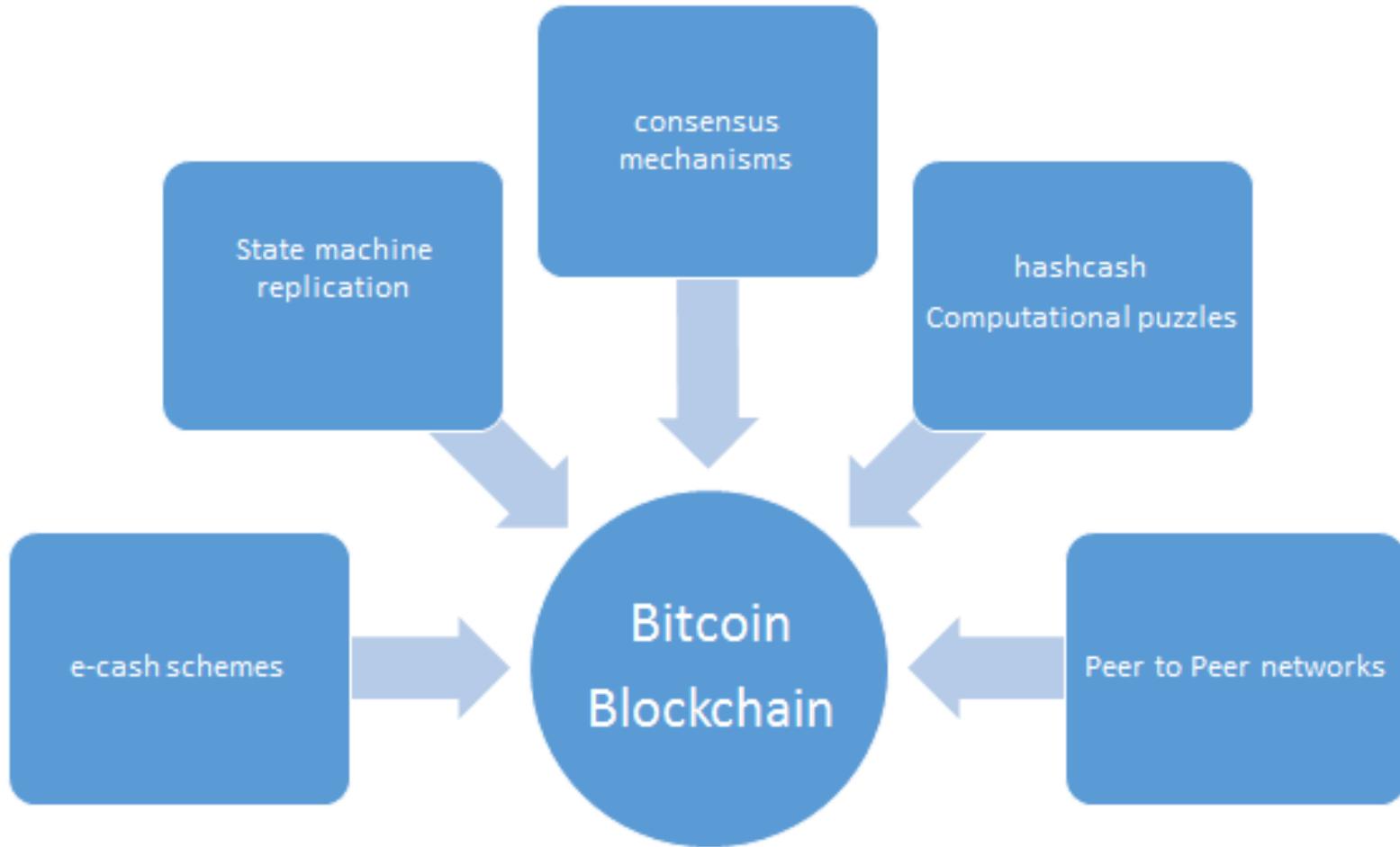
## ❖ Paxos:

Protocol introduced by Leslie Lamport. In Paxos nodes are assigned various *roles* such as **Proposer**, **Acceptor**, and **Learner**. Nodes or processes are named replicas and consensus is achieved in the presence of faulty nodes by agreement among a majority of nodes

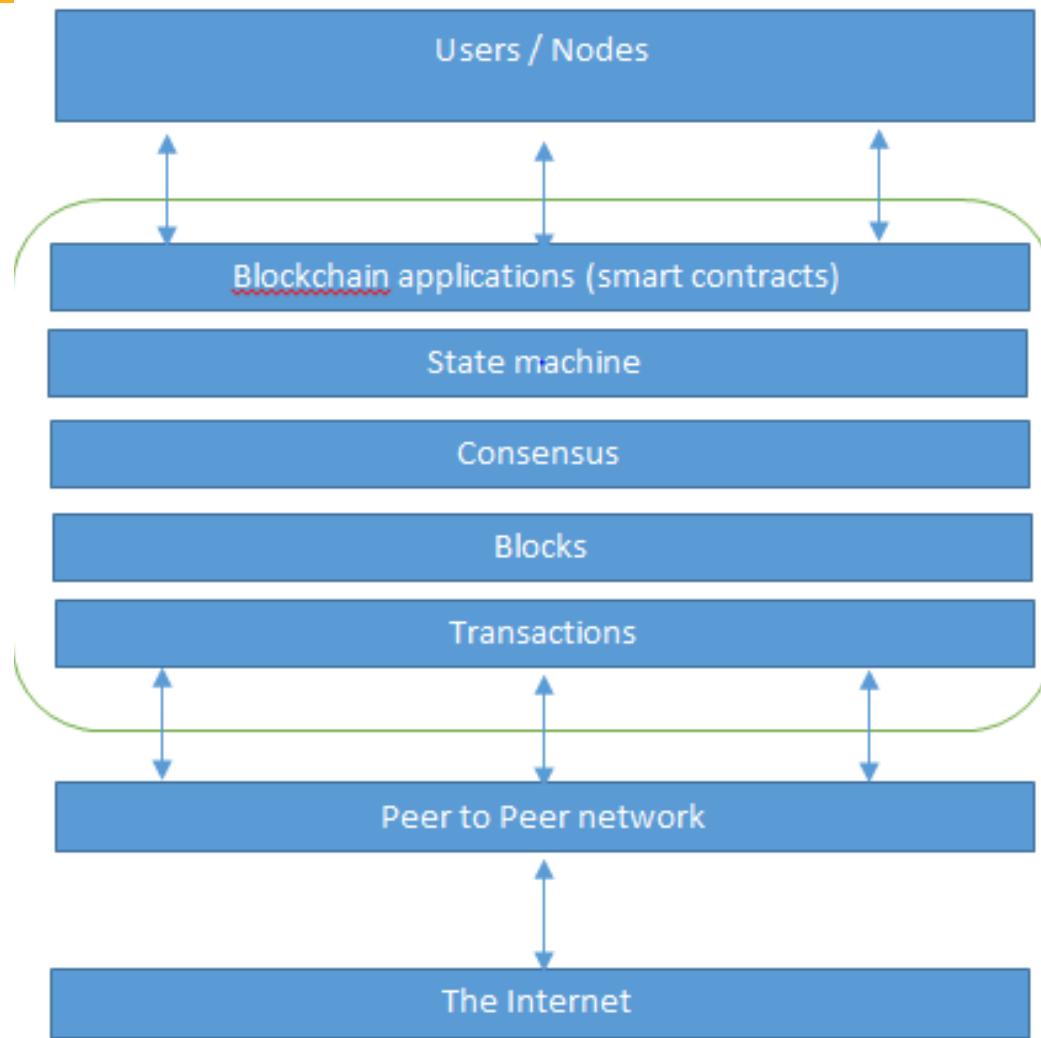
## ❖ RAFT: [Reliable, Replicated, Redundant, And Fault-Tolerant]

Assigning any of three *states*, **Follower**, **Candidate**, or **Leader**, to the nodes. A Leader is elected after a candidate node receives enough votes and all changes now have to go through the Leader, who commits the proposed changes once replication on the majority of follower nodes is completed

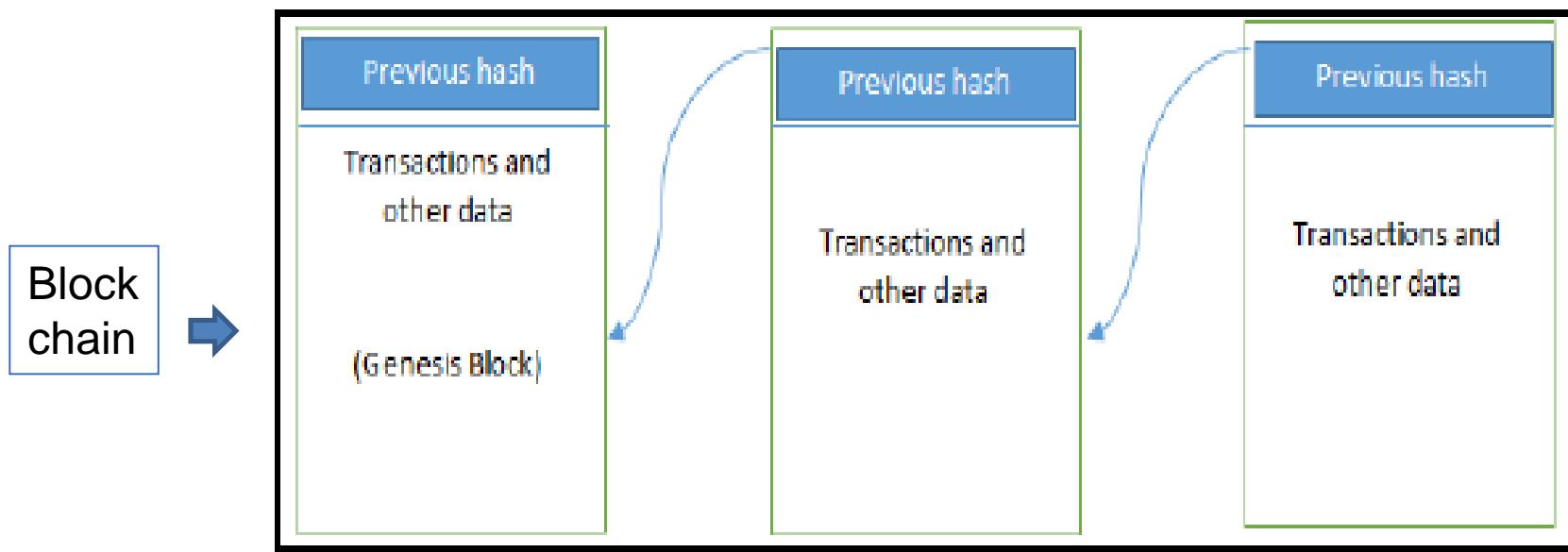
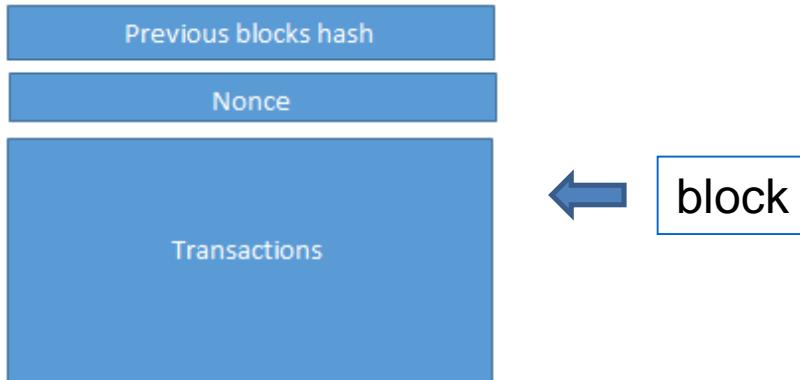
# Moving to blockchain



# Blockchain Network



# Block and blockchain structure



# Generic elements of a blockchain

- ❖ **Addresses:** senders & recipients
- ❖ **Transaction:** transfer of values
- ❖ **Virtual machine:** Ethereum Virtual Machine (EVM) and Chain Virtual Machine (CVM)

**Smart contracts:** programs which run on top of the blockchain and encapsulate the business logic to be executed when certain conditions are met

# Essential Features of Blockchain



- ❖ **Distributed consensus**
- ❖ **Transaction verification**
- ❖ **Platforms for smart contracts**
- ❖ **Transferring value between peers**
- ❖ **Generating cryptocurrency**
- ❖ **Smart property**
- ❖ **Provider of security**
- ❖ **Immutability**
- ❖ **Smart contracts (automated autonomous programs)**
- ❖ **Uniqueness**

# Consensus protocols in blockchain



*agreeing to a single version of truth by all peers on the blockchain network*

---

## Categories:

1. Proof-based, leader-based, or the *Nakamoto consensus* whereby a leader is elected and proposes a final value
2. Byzantine fault tolerance-based, which is a more traditional approach based on rounds of votes

# Important consensus algorithms

- Proof of Work
- Proof of Stake
- Delegated Proof of Stake
- Proof of Elapsed Time
- Deposit-based consensus
- Proof of importance
- Federated consensus or federated Byzantine consensus
- Reputation-based mechanisms
- Practical Byzantine Fault Tolerance

# Proof of Work

---

- ❖ relies on proof that enough computational resources have been spent before proposing a value for acceptance by the network
  - ❖ used in bitcoin and other cryptocurrencies like **Litecoin (LTC)**, **Binance Coin (BNB)** etc.
  - ❖ only algorithm that has proven successful against **Sybil attack** (a type of attack seen in peer-to-peer networks in which a node in the network operates multiple identities actively at the same time and undermines the authority/power in reputation systems).
-

# Proof of Work

- ❖ PoW, is the original consensus algorithm in a Blockchain network which is used in Bitcoin
- ❖ PoW is : difficult to produce but easy to verify by others
- ❖ The Proof of Work consensus algorithm involves solving a computational challenging puzzle in order to create new blocks in the Bitcoin blockchain
- ❖ The process is known as ‘mining’, and the nodes in the network that engage in mining are known as ‘miners’  
**Example:** Find a number x, such that SHA256(text+x) has 10 leading zeroes.
- ❖ **10 leading zeroes = Network difficulty**

# Proof of Stake



- ❖ A node or user has to have enough stake in the system
- ❖ Ex: the user has invested enough in the system so that any malicious attempt would outweigh the benefits of performing an attack on the system
- ❖ First introduced by Peercoin and used in the Ethereum blockchain
- ❖ PoS and coin age - derived from the amount of time and the number of coins that have not been spent
- ❖ chances of proposing and signing the next block increase with the coin age

# Proof of Stake

- ❖ PoS is designed to increase network security and reduce resource wasting.
- ❖ The creator of the next block is chosen in
  - ❖ Combinations of random selection and wealth
  - ❖ E.g. holding 1% of the coins gives the chance to verify (mine) 1% of the "Proof of Stake blocks"
- ❖ The "Monopoly Problem": a monopolist (holder of the most coins) could double spend or deny / filter other's transactions
  - ❖ Executing a monopoly attack is much more expensive than in PoW
- ❖ Used by [Cardano](#), [Bitconnect](#)

# Delegated Proof of Stake

---

- ❖ innovation over standard PoS
- ❖ each node that has a stake in the system can delegate the validation of a transaction to other nodes by **voting**
- ❖ used in the **bitshares** blockchain



- ❖ BitShares is based on [Graphene](#), an open source C++ blockchain implementation, which acts as a consensus mechanism

# Delegated Proof of Stake

- ❖ Stakeholders vote for delegates in democratic way
  - ❖ Every wallet holding coins can vote for delegates
  - ❖ Votes weight is proportional to the wallet's stake in the network
- ❖ Delegates generate new blocks (like miners in PoW)
  - ❖ Validate transactions and take the fees as profit
  - ❖ Maintain the blockchain, e.g. vote for changing the network parameters like block intervals, transaction fees, others
  - ❖ Very fast confirmation of transactions (< 1 sec)
- ❖ Used by [Lisk](#), [EOS](#),[ARK](#).

# Proof of Elapsed Time

- ❖ Introduced by Intel
- ❖ uses **Trusted Execution Environment (TEE)** to provide randomness and safety in the leader election process via a **guaranteed wait time**
- ❖ requires the **Intel SGX (Software Guard Extensions) processor** in order to provide the security guarantee and
- ❖ for it to be secure
- ❖ Intel Sawtooth Lake blockchain project



# Proof of Elapsed Time

- ❖ PoET relies on a trusted execution environment (TEE)
  - ❖ Supported by modern CPUs from Intel, AMD and ARM
  - ❖ No cryptographic puzzle (like in PoW algorithms)
- ❖ PoET ensures blocks get produced in a random lottery fashion
  - ❖ Generates securely the next block + a proof of the waiting time inside the TEE.
  - ❖ The proof of waiting time can be verified by all other nodes.
- ❖ Problem: a small subset of compromised nodes can compromise the entire system

# Deposit-based consensus

- ❖ Nodes that wish to participate on the network have to put in a **security deposit** before they can propose a block
  
- ❖ The protocol governs through the controlling of these security deposits, which implicitly governs the incentives of validators

# Deposit based consensus

---

- ❖ Nodes in this consensus protocol have to place a security deposit in order to serve the consensus by producing blocks
  - ❖ The protocol governs through the controlling of these security deposits, which implicitly governs the incentives of validators
  - ❖ The “arbitrator” is based on a new chain selection rule called **GHOST** (Greedy Heaviest Observed Sub Tree) (Sompolsky & Zohar, 2013).
  - ❖ The advantage of GHOST is a strong convergence of history; meaning that every block would either be fully abandoned or fully adopted
  - ❖ When a validator validates a transaction that GHOST considers invalid, the validator **loses their deposit** and forfeits the privilege of participating in the consensus process
-

# Proof of importance

---

- ❖ important and different from Proof of Stake
- ❖ Proof of importance not only relies on **how much stake** a user has in the system
- ❖ also monitors the **usage and movement of tokens** by the user to establish a level of **trust** and **importance**
- ❖ used in Nem coin



# Proof of Importance

- ❖ PoI is similar to PoS, where stakes are based on coins + activity
- ❖ The mining power calculated by the importance in the network
- ❖ More coins hold for long time -> bigger importance (like a stake)
- ❖ More transactions / activities ->bigger importance
  
- ❖ Used by <https://nem.io/>

# Federated consensus (federated Byzantine consensus)

- ❖ nodes in this protocol keep a group of publicly trusted peers and propagates only those transactions that have been validated by the majority of trusted nodes
- ❖ A Federated Byzantine Agreement (FBA) is a form of Byzantine fault tolerance where each byzantine general is responsible for their own blockchain
- ❖ A quorum is the minimum number of nodes required for a solution to be correct and after a quorum forms, the block is validated and included on the blockchain
- ❖ consensus is a process of that seeks widespread agreement among group members
- ❖ agreement is (countable) an understanding between entities to follow a specific course of conduct

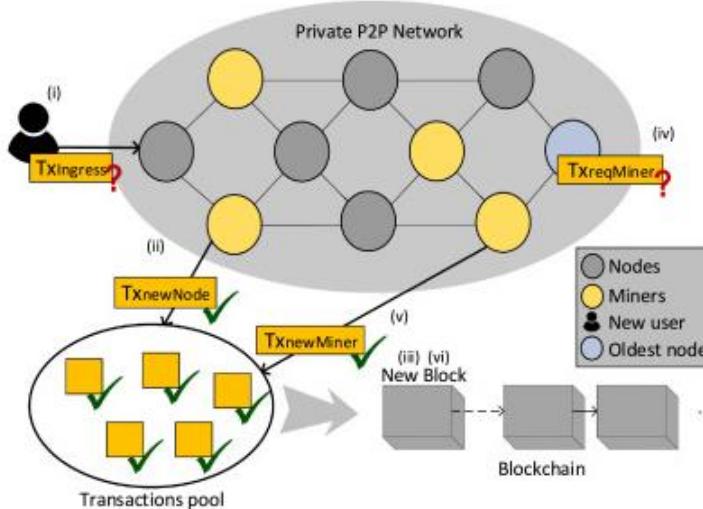
# Federated Byzantine Consensus

---

- ❖ FBC relies on small sets of trusted parties. These sets are from nodes that trust each other's information
- ❖ It assumes that participants know other participants, and can distinguish which it considers important
- ❖ The important participants mentioned do not agree to the transaction until the participants they consider important agree as well, so on and so forth
- ❖ Eventually, enough of the network would accept the transaction, such that it becomes infeasible for the attacker to roll it back
- ❖ Good behaviour of nodes would ascend nodes into small sets of trusted parties, with the level of trust built over time
- ❖ Used by Stellar.

# Reputation-based mechanisms

- ❖ a leader is elected on the basis of the reputation it has built over time on the network
- ❖ Can be based on the voting from other members
- ❖ <https://www.sciencedirect.com/science/article/pii/S1389128620300360>



# Reputation Based Mechanisms

---

- ❖ In Reputation based consensus algorithms, the miners reputation is at stake instead of coins
  - ❖ A reputation threshold may be set for nodes to add a block to the chain
  - ❖ A set of nodes(validators) monitor the behaviour of each node involved in the consensus and updates the node reputation score
  - ❖ **Every cooperative behaviour results in a reward, and a non-cooperative or malicious behaviour results in a punishment(reduction of reputation)**
  - ❖ Nodes with very low reputation are removed
-

# Practical Byzantine Fault Tolerance

---

- ❖ PBFT as blockchain consensus algorithm:
  - ❖ Nodes collecting transactions, select a leader for the next block
    - ❖ Can be random (deterministic) or random based on a stake
  - ❖ The leader orders the transactions + broadcasts the ordered list
  - ❖ Each node validates / executes the transactions + broadcasts the calculated hash of the new block
  - ❖ When 2/3 of the nodes have the same hash, the new block is published (mined)
  - ❖ Transactions are very fast
- ❖ Used by Hyperledger Fabric, Neo.

# How blockchains accumulate blocks?

1. A node starts a transaction by signing it with its private key.
2. The transaction is propagated (flooded) by using much desirable Gossip protocol to peers, which validates the transaction based on pre-set criteria. Usually, more than one node is required to validate the transactions
3. Once the transaction is validated, it is included in a block, which is then propagated on to the network. At this point, the transaction is considered confirmed
4. The newly created block now becomes part of the ledger and the next block links itself cryptographically back to this block. This link is a hash pointer. At this stage, the transaction gets its second confirmation and the block gets its first
5. Transactions are then reconfirmed every time a new block is created. Usually, six confirmations in the bitcoin network are required to consider the transaction final

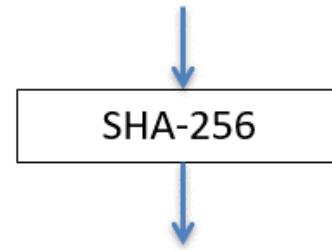
---

**Let us see how to add a block in a blockchain!**

# Hashing: Quick revision..

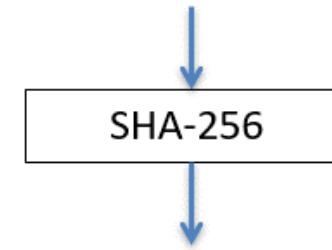
- ❖ A hash function (like SHA-256) takes a block of data in, and produces an effectively random fixed size integer
- ❖ Any change to the input randomizes it
- ❖ We cannot compute an input from an output

“The quick brown fox did some crypto”



410312395834291203...

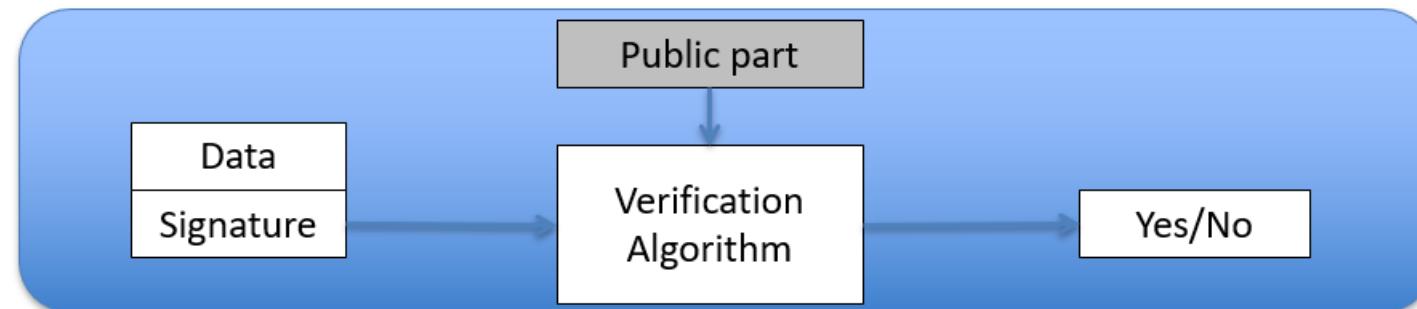
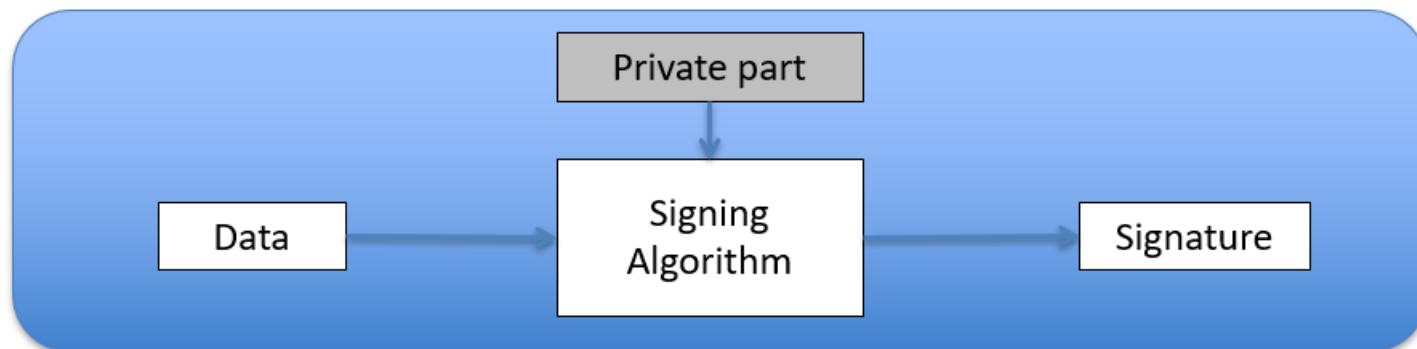
“The quick brown Fox did some crypto”



983249120432492340...

# Signatures & Authentication

Signing key	
Public part	454F4D3E1..
Private part	56F23F2D..



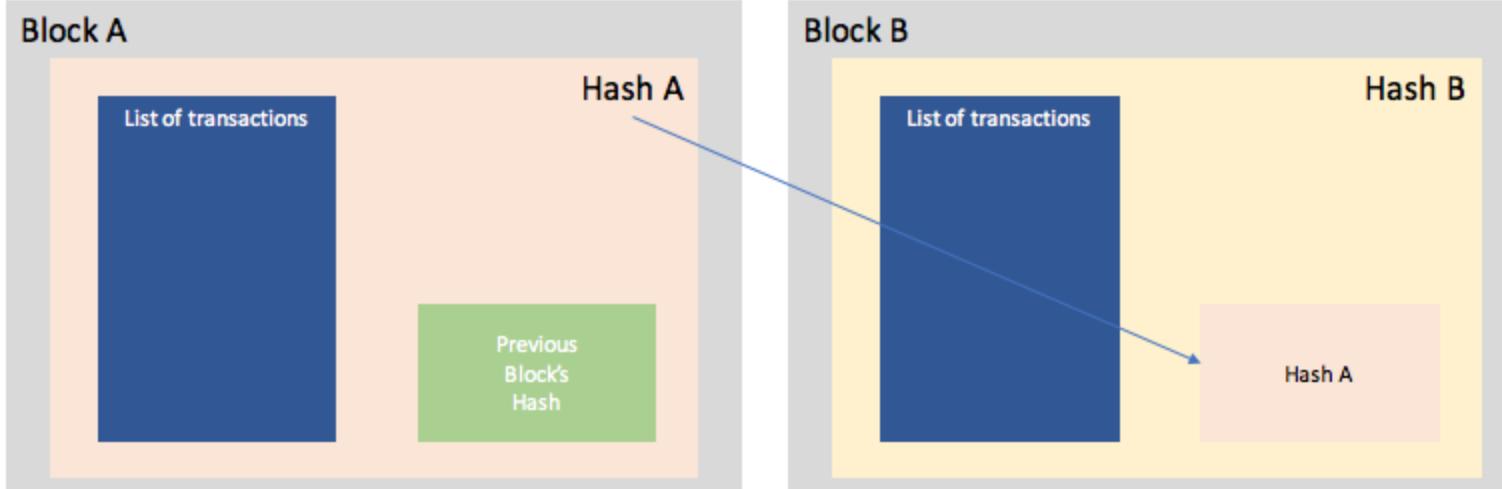
# Transaction Broadcast

- ❖ The transaction is then broadcast to all the nodes which are part of the blockchain
- ❖ The transaction is then added to the unverified pool of the respective nodes
- ❖ In the Proof of Work algorithm (BITCOIN uses POW), it requires a verification of minimum 6 nodes
- ❖ These nodes then try to solve a **computationally hard mathematical puzzle**
- ❖ The node which solves this **puzzle** first, adds the block to the chain and gets paid as per the protocol rules

# What is this puzzle?

- Information contained in Block header: Version No, Merkle root, Prev Block Hash, Timestamp, Difficulty Target,Nonce etc.,
- Essentially, you can continuously change the Nonce until the SHA256 hash function results in a hash with a certain amount of leading 0s
- The amount of leading 0s is based on the Difficulty target which is predetermined by the protocol. Currently it is set such that it takes around 10 minutes to solve this puzzle
- The first miner to find a solution wins the round and publishes that block into the blockchain
- Reward of Miner = Puzzle Solving Reward (6.25BTC) + Transaction Charges

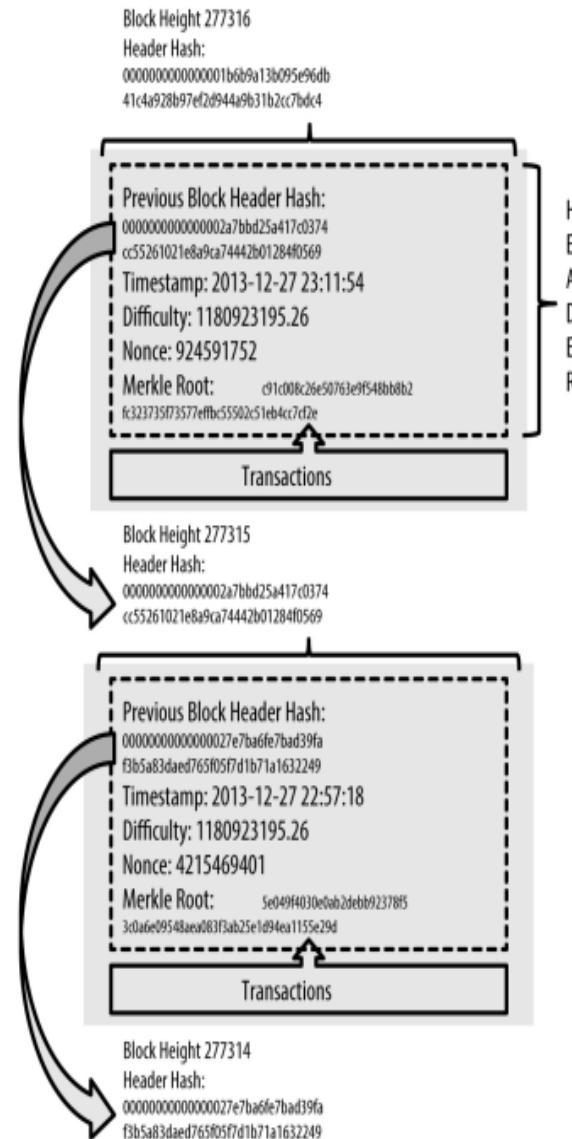
# Block and its Hash



- Now, if a malicious actor were to go back to Block A's transactions and change even the smallest detail, then Hash A would completely change, which in turn would change the subsequent Hash B because you just altered the data that Block B hashed
- Since, it is computationally very hard to add a block, hence it is not possible to alter the block structure for a malicious node

# Linking of Blocks in a Chain

- ❖ The new block is added to the blockchain
  - ❖ which is then validated by other nodes
  - ❖ (minimum 6 validations needed)
- ❖ This new block is a child of the last block on the chain and extends the existing blockchain
- ❖ The node adds this new block to the end of the chain, making the blockchain longer with a new height of 277,316





**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

**BLOCKCHAIN TECHNOLOGY**  
**BITS F452**  
**1<sup>st</sup> Sem 2022-23**  
**Lecture 3**

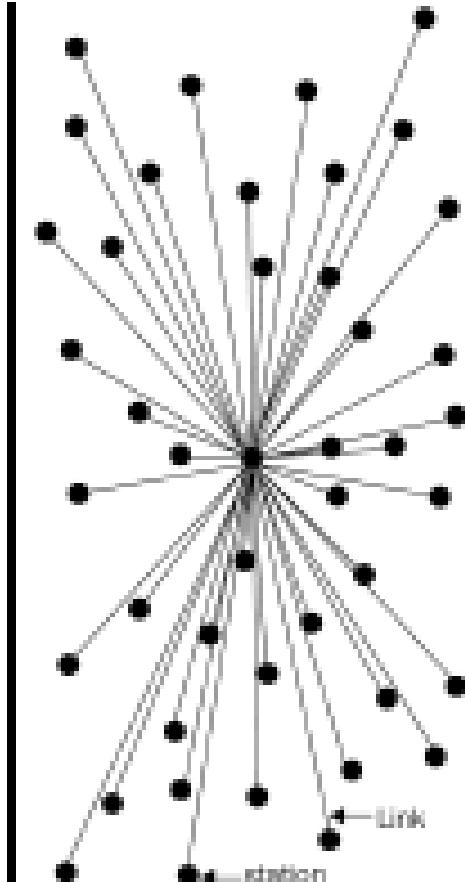
Decentralization in the context of blockchain  
P2P Systems....



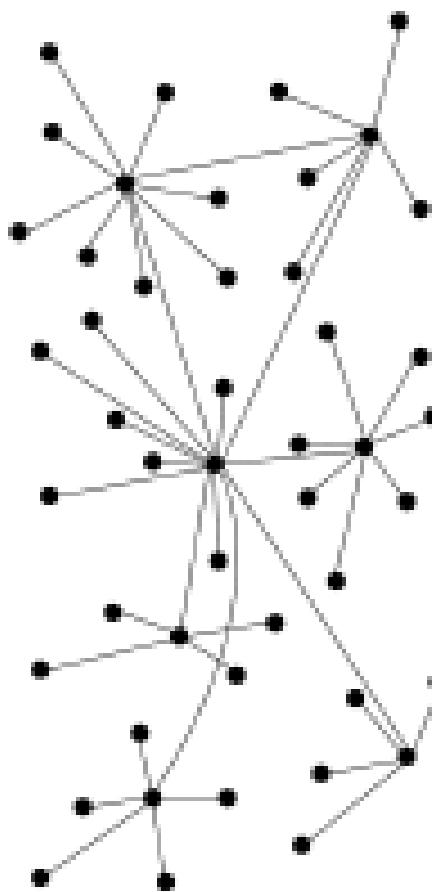
# Decentralization in the context of blockchain

- ❖ Decentralization from a blockchain perspective is a mechanism that provides a way to remodel existing applications and paradigms or build new applications in order to give full control to users
- ❖ Blockchain by design is a perfect vehicle for providing a platform that **does not need any intermediaries** and can function with many different leaders chosen via consensus mechanisms
- ❖ This model allows anyone to compete to become the decision-making authority
- ❖ This competition is governed by a **consensus** mechanism and the most commonly used method is known as **Proof of Work (PoW)**

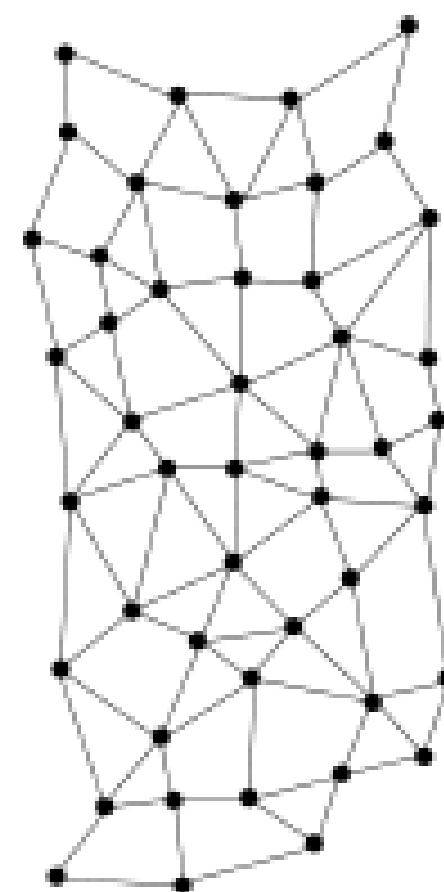
# Different types of systems



CENTRALIZED



DECENTRALIZED



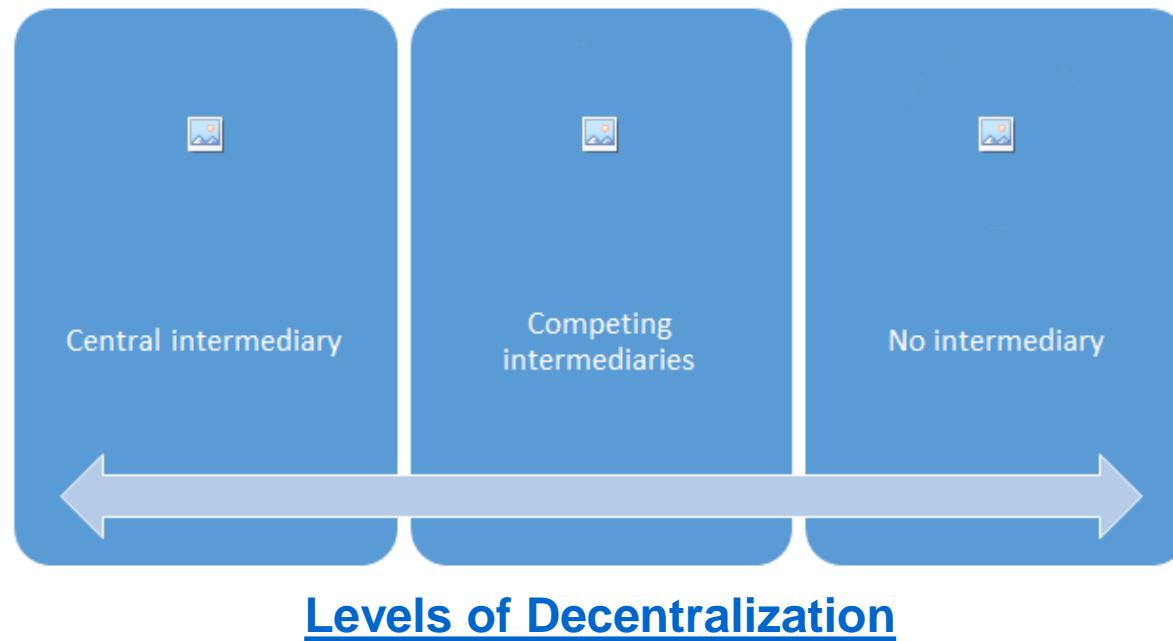
DISTRIBUTED

# Decentralized vs distributed

- ❖ In a **distributed system**, there still exists a central authority that governs the entire system
- ❖ In a **decentralized system**, no such authority exists
- ❖ A decentralized system is a type of network whereby nodes are not dependent on a single master node; instead, control is distributed among many nodes
- ❖ **Decentralized consensus:** enables a user to agree on something via a consensus algorithm without the need for a central trusted third party, intermediary, or service provider

# Methods of decentralization

- ❑ Disintermediation: Example...
- ❑ Through competition: no monopoly of SPs; use of smart contracts
- ❑ Scale of decentralization
- ❑ Challenges ..
- ❑ Do we need to decentralize every environment??
- ❑ BitTorrent/Gnutella (before bitcoin/blockchain)



# How to decentralize?

1. What is being decentralized?
2. What level of decentralization is required?
3. What blockchain is used?
4. What security mechanism is used? - atomicity

1. **Answer 1:** Money transfer system.
  2. **Answer 2:** Disintermediation.
  3. **Answer 3:** Bitcoin.
  4. **Answer 4:** Atomicity.

# Blockchain and full ecosystem Decentralization: the elements

Storage  
Communication  
Computation



Decentralized paradigms

identity  
wealth



Centralized paradigms

# Storage

- ❖ Blockchain not suitable for storing images or large blobs of data; alternative is to use **distributed hash tables (DHTs)**
- ❖ DHTs originally used in peer-to-peer file sharing software, such as BitTorrent, Napster, Kazaa, and Gnutella. DHT research was made popular by CAN, Chord, Pastry, and Tapestry projects
- ❖ BitTorrent turns out to be the most scalable and fast network; why users do not prefer to keep the files indefinitely?

## Requirements:

1. High availability
2. Link stability

EX: IPFS uses Kademlia DHT and merkle **DAG (Directed Acyclic Graph)** to provide the storage and searching functionality, respectively



# Communication

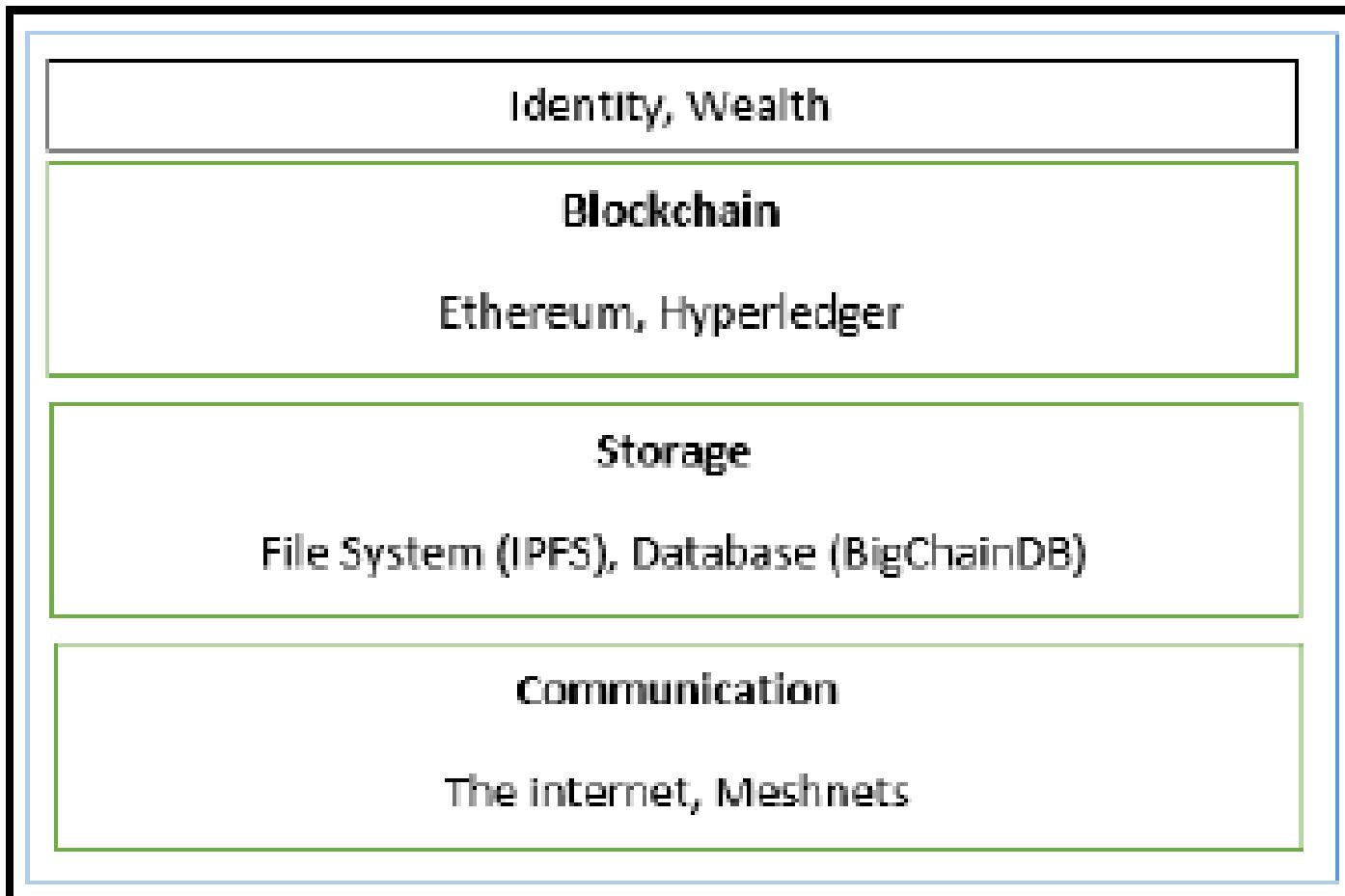
---

- ❖ Internet - (the communication layer in blockchain) is decentralized
  - ❖ ISPs can be bottlenecks
  - ❖ Mesh networks – a viable alternative
-

# Computation

- ❖ Decentralization of computing or processing is achieved by blockchain technology ex:Ethereum
- ❖ Smart contracts with embedded business logic can run on the network

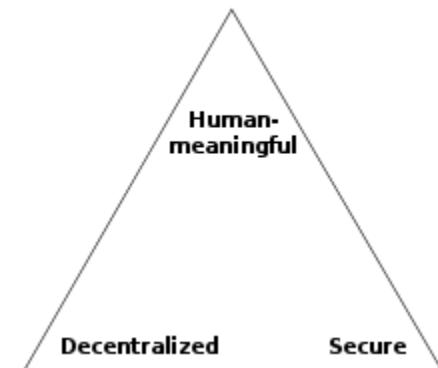
# Blockchain decentralized ecosystem overview



# The concept of Zooko's Triangle

Zooko's triangle is a [trilemma](#) of three properties that are generally considered desirable for names of participants in a [network protocol](#)

- ❖ that a naming system in a network protocol be secure, decentralized, and meaningful to humans
- ❖ It is conjectured that a system can have only two of these properties simultaneously, but with the advent of blockchain, in the form of Namecoin, this problem was resolved



# More on terminology:

---

- ❖ **Decentralized organization (DOs)** are software programs that run on a blockchain and are based on the idea of real human organizations with people and protocols
- ❖ **Decentralized autonomous organization (DAO)** is also a computer program than runs on top of a blockchain and embedded within it are governance and business logic rules: Ex: Ethereum blockchain

What is the difference between DAO and DO?

- ❖ **Decentralized autonomous corporations (DACs)** are a similar concept but are considered a smaller subset of DAOs

What is the difference between DAO and DAC?

- ❖ **Decentralized autonomous societies (DASSs):** entire societies can function on a blockchain with the help of multiple complex smart contracts and a combination of DAOs and Decentralized applications (DAPPs) running autonomously

# Requirements of a decentralized application

---

- 1) fully open source and autonomous
- 2) Data and records of operations of the application must be cryptographically secured and stored on a public, decentralized blockchain
- 3) A cryptographic token must be used by the application in order to provide access and rewards to those who contribute value to the applications
- 4) The tokens must be generated by the decentralized application according to a standard cryptographic algorithm

# Operations of a DAPP

- ❖ Consensus established by a DAPP using consensus algorithms such as Proof of Work and Proof of Stake
- ❖ only PoW has been found to be incredibly resistant to **51% attacks**
- ❖ DAPP can distribute tokens (coins) via mining, fundraising, and development

A 51% attack is when **a single cryptocurrency miner or group of miners gains control of more than 50% of a network's blockchain**. Such attacks are one of the most significant threats for people who use and buy cryptocurrencies.

- ❖ DAPP Examples:: KYC-Chain; OpenBaazar; Lazooz;

# Platforms for decentralization

---

## 1. Ethereum

Ethereum tops the list as being the first blockchain that introduced a Turing-complete language and the concept of a virtual machine; public blockchain

## 2. MaidSafe

MaidSafe provides a SAFE (Secure Access for Everyone) network that is made up of unused computing resources, such as storage, processing power, and the data connections of its users

## 3. Lisk

Lisk is a blockchain application development and cryptocurrency platform. It allows developers to use JavaScript to build decentralized applications and host them in their own respective sidechains. Lisk uses the Delegated Proof of Stake (DPOS) mechanism



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

**BLOCKCHAIN TECHNOLOGY**  
**BITS F452**  
**1<sup>st</sup> Sem 2022-23**  
**Lecture 4**

**Cryptography Essentials for Blockchain Technology**



# Cryptography and Technical Foundations

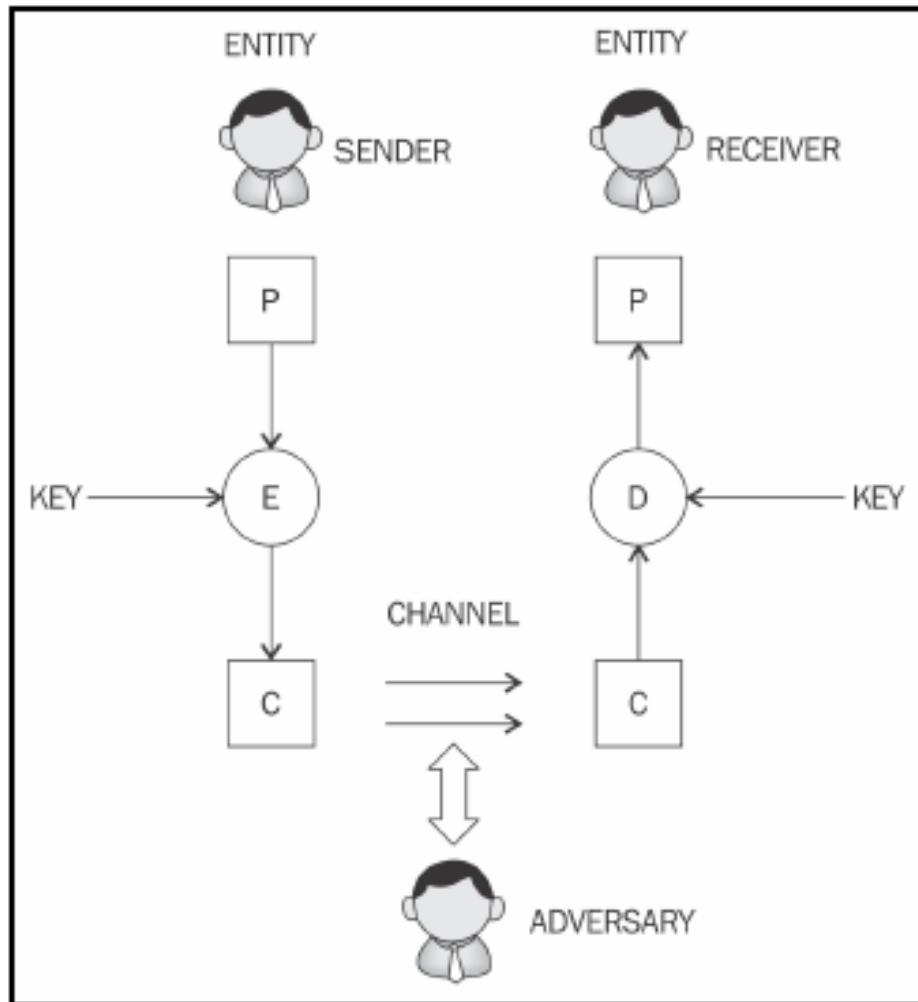
- ❖ Crypto basics essential for blockchain technology
- ❖ Hash functions and their properties
- ❖ Public Key Cryptosystems
- ❖ Digital Signatures
- ❖ Hash Puzzles
- ❖ Hash Pointers
- ❖ Merkle Data Structures

# Crypto services in blockchain

- ❖ Confidentiality
- ❖ Integrity
- ❖ Authentication (entity, data origin)
- ❖ Non-repudiation
- ❖ accountability

# Cryptographic primitives

## Symmetric model and Asymmetric model



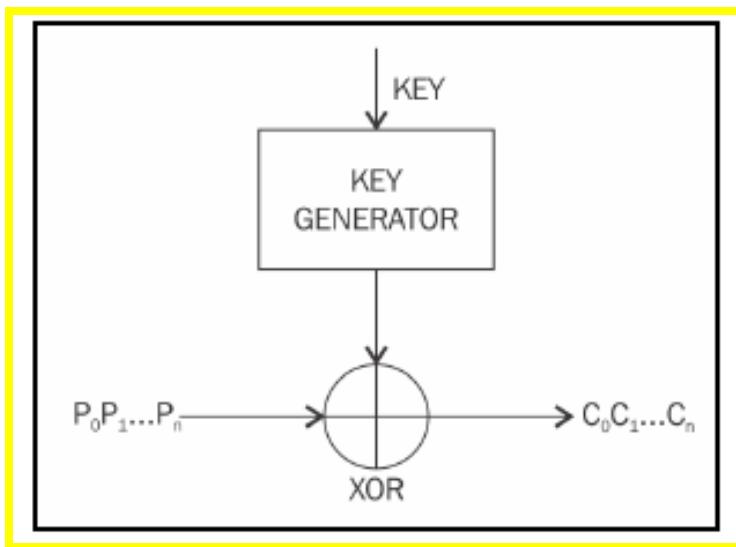
- ❖ Plaintext (or cleartext)
  - ❖ The message.
  - ❖ Encryption (encipher)
  - ❖ Encoding of message.
- ❖ Ciphertext
  - ❖ Encrypted message.
  - ❖ Decryption (decipher)
  - ❖ decoding of ciphertext

**Symmetric Case:** both keys are the same or derivable from each other.  $K_1 = K_2$ .

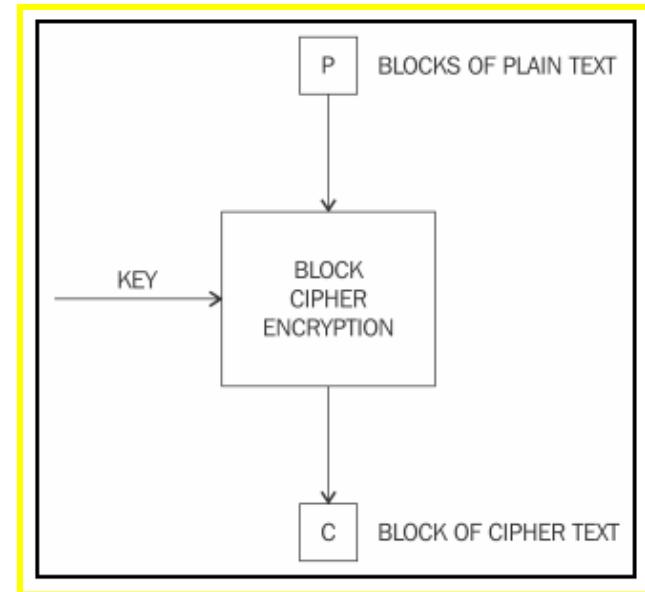
**Asymmetric Case:** keys are different and not derivable from each other.  $K_1 \neq K_2$

Generic encryption/decryption model

# Symmetric ciphers

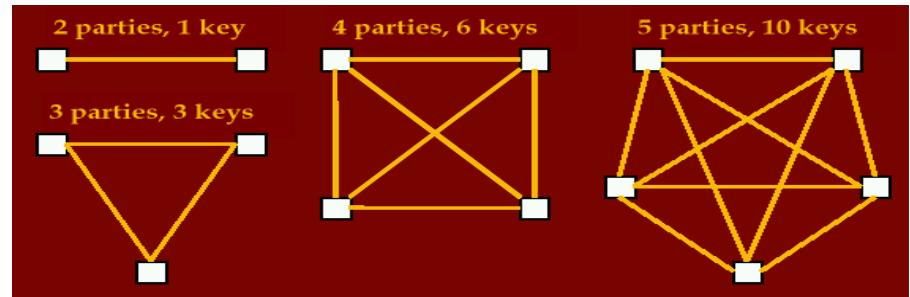
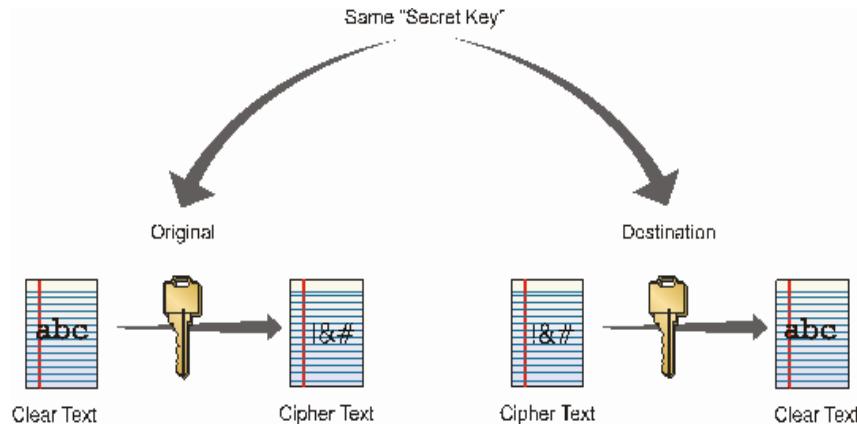


Stream cipher

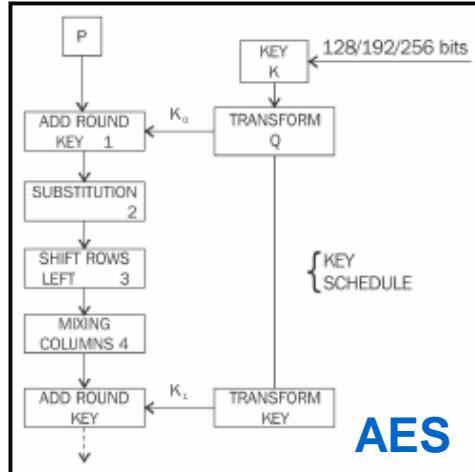


block cipher

# Symmetric system



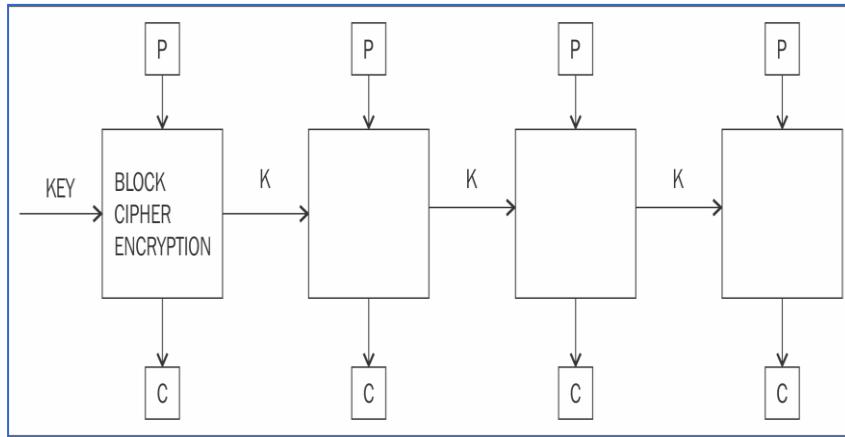
DES cipher  
AES cipher



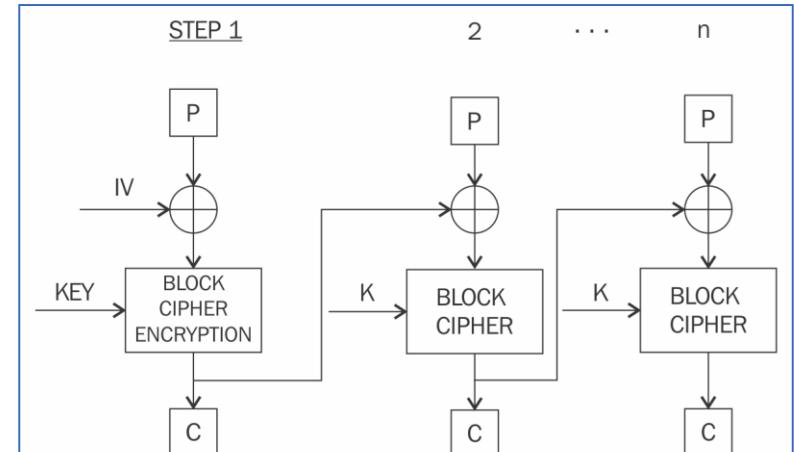
Refer to the OpenSSL example of  
how to encrypt and decrypt using  
AES (given in the text book)

# Block cipher modes of operations

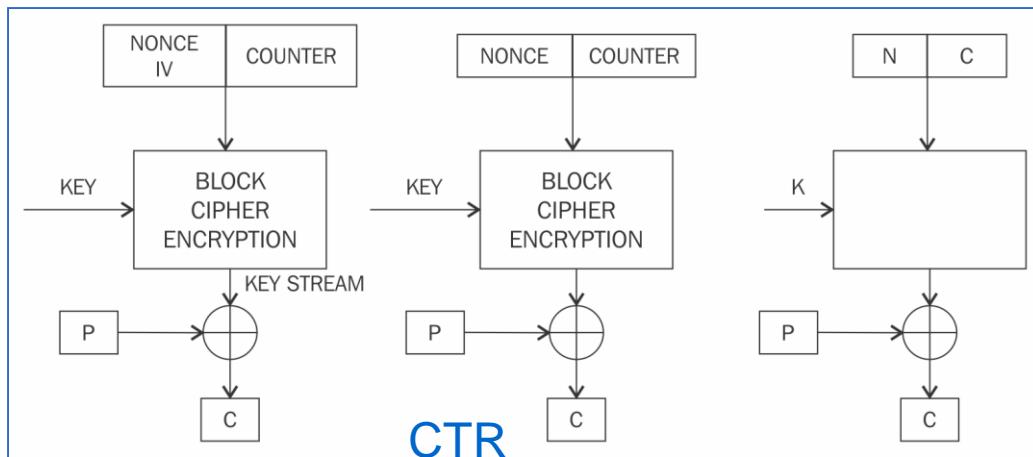
- ❖ Electronic Code Book (ECB), Cipher block chaining (CBC), Cipher Feedback Mode (CFB) Output Feedback Mode (OFB), or Counter mode (CTR)



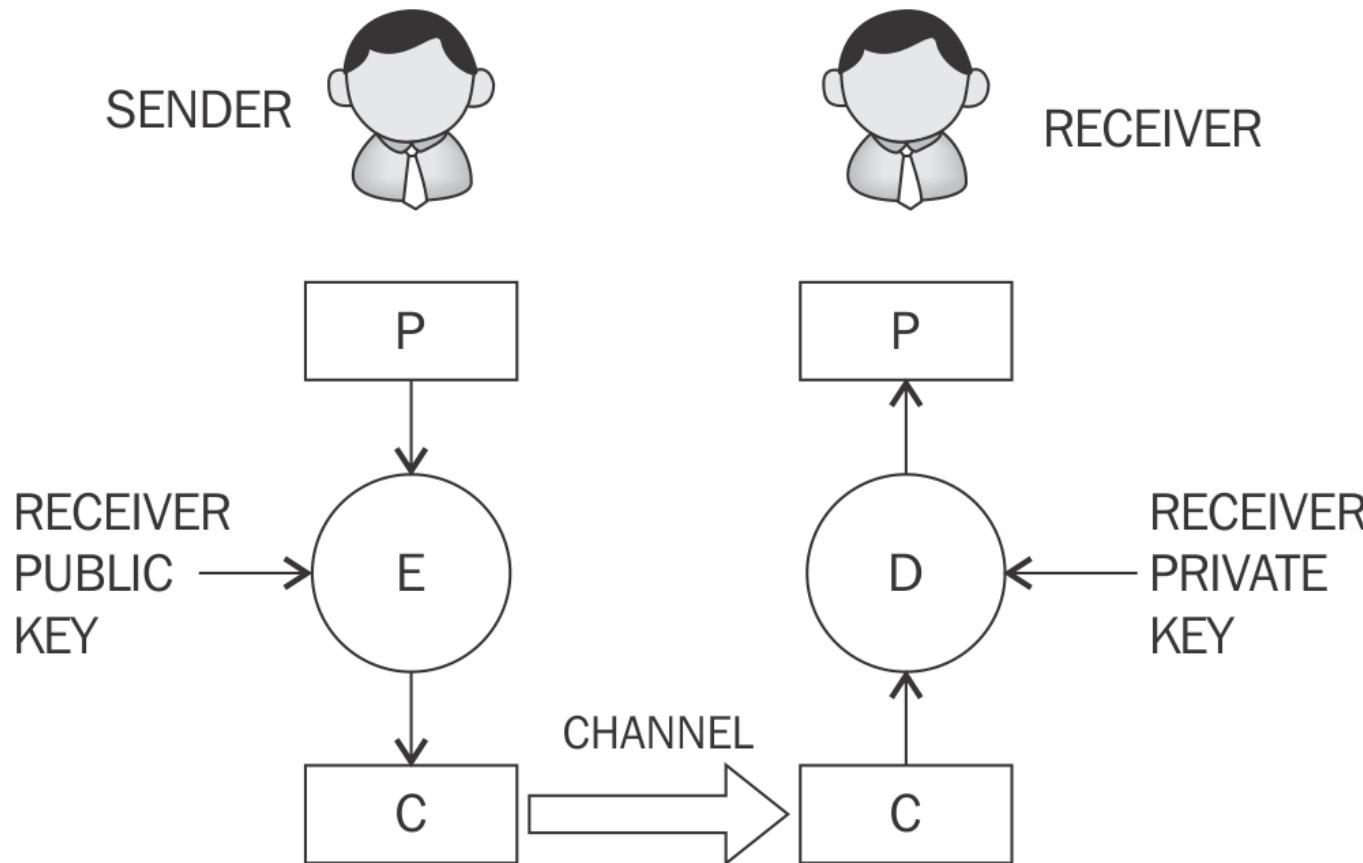
ECB



CBC



# Asymmetric cryptography



ENCRYPTION DECRYPTION USING PUBLIC / PRIVATE KEY

# Asymmetric (public key) cryptography



## The challenge:

- ❑ Suppose Alice has a channel for communicating with Bob.
- ❑ Alice and Bob wish to use this channel to establish a shared secret
- ❑ But, Eve is able to learn everything sent over the channel
- ❑ If Alice and Bob have no other channel to use, can they establish a shared secret that Eve does not know?

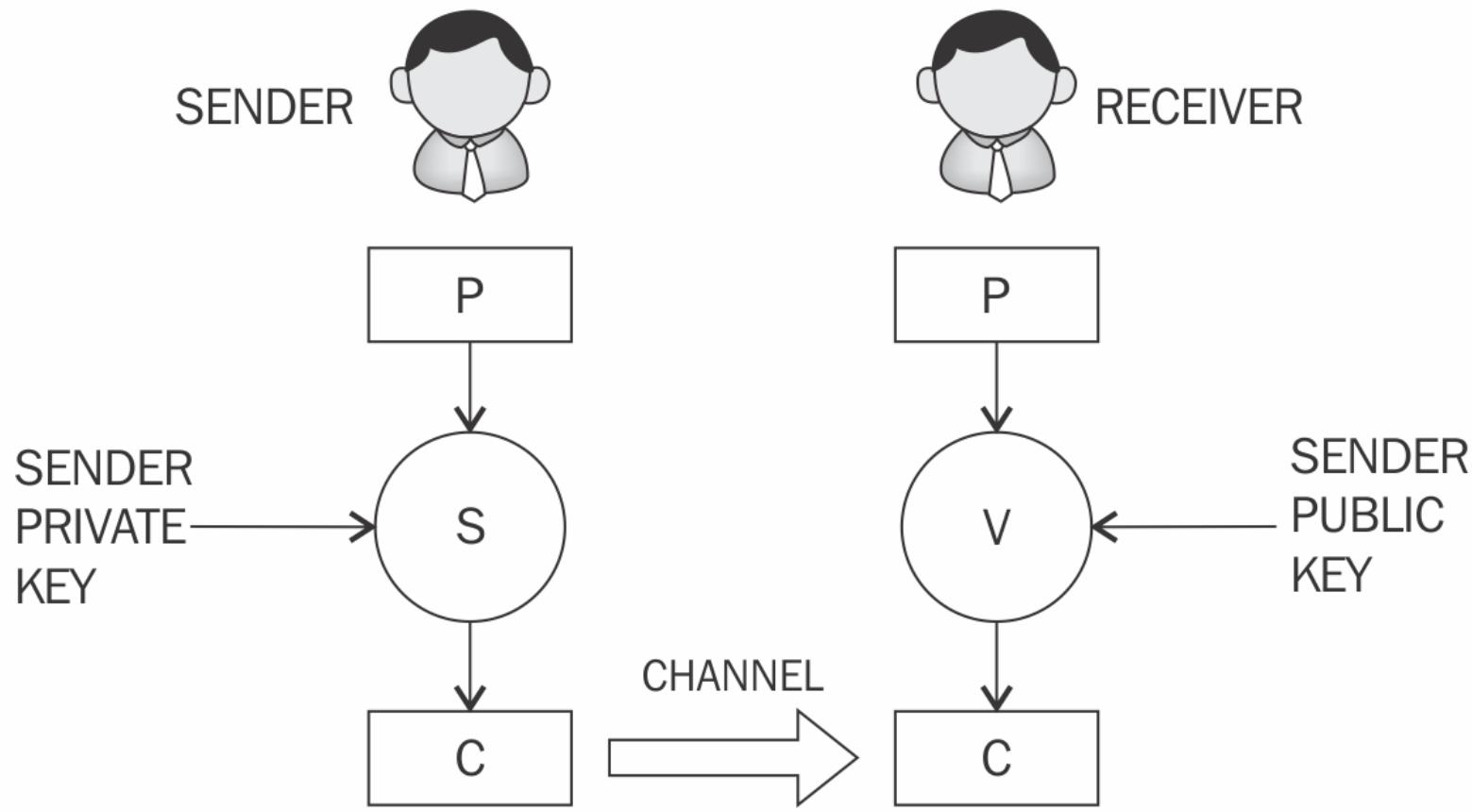
## The solution:

- ❑ Find a hard math problem, that is easy to compute in the forward direction, but is difficult to solve in the reverse direction, unless you have some special knowledge

## Asymmetric algorithms:

- ❑ Also called public-key algorithms.
- ❑ Encryption key is different from decryption key.
- ❑ Furthermore, one cannot be calculated from other.
- ❑ Encryption key is often called the **public key** and decryption key is often called the **private key**.
- ❑ Advantages: better key management.
- ❑ Disadvantages: slower, more complex.
- ❑ Both techniques are complementary.
- ❑ Examples: RSA, Diffie-Hellman, El Gamal, etc

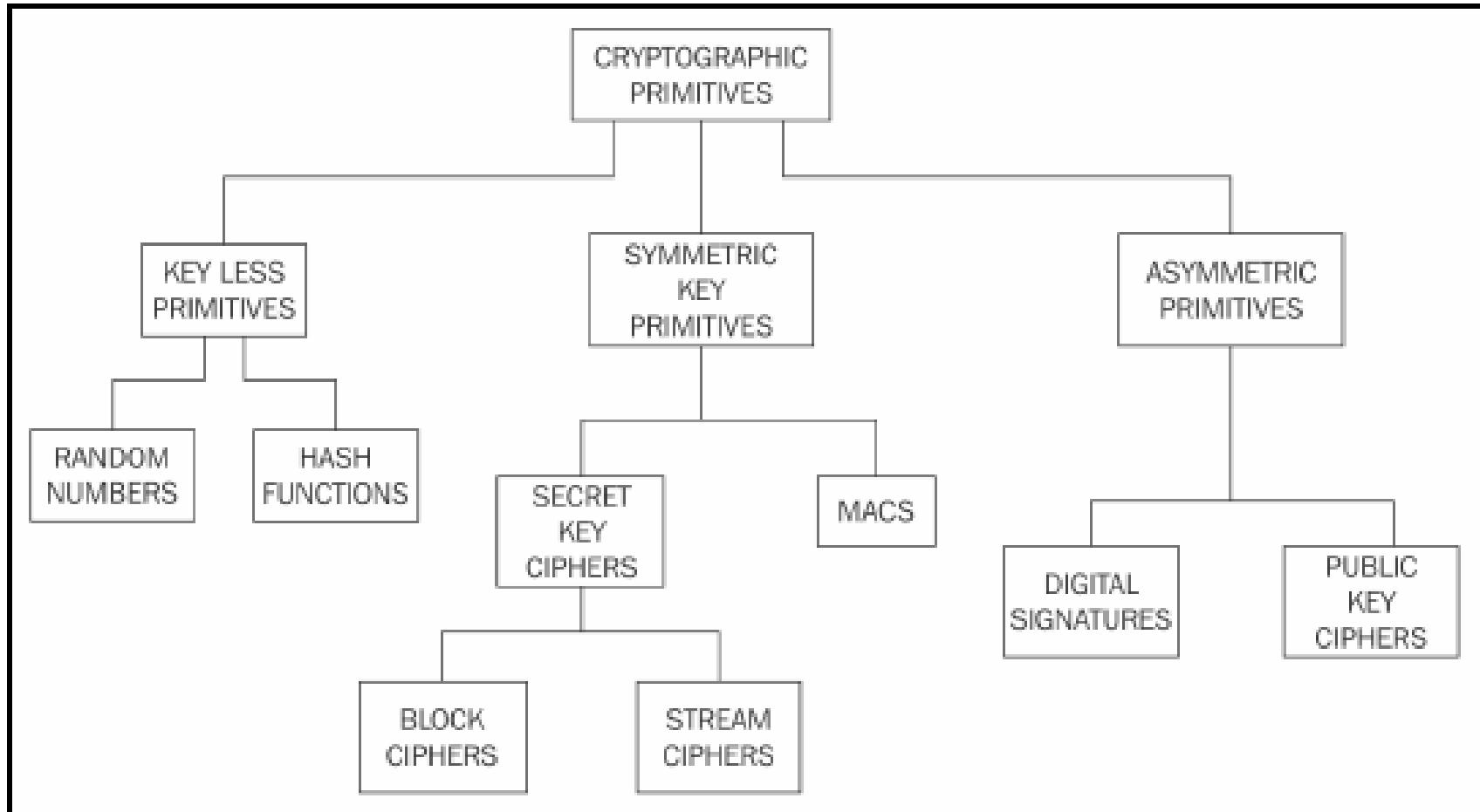
# Public key crypto signature scheme



SIGNING & VERIFICATION USING PUBLIC / PRIVATE KEY

# Taxonomy of Cryptographic primitives

## - Blockchain context



# Cryptographic Hash functions and blockchain



- ❖ Hash functions are used to create fixed length digests of arbitrarily long input strings
- ❖ Hash functions are keyless and provide the data integrity service
- ❖ Built using iterated and dedicated hash function construction techniques
- ❖ Various families of hash functions: MD, SHA1, SHA-2, SHA-3, RIPEMD, and Whirlpool
- ❖ Hash functions are commonly used in digital signatures and message authentication codes, such as HMACs

# Hash function categories

## 1. Message Digest (MD)

- ❖ Message Digest functions were very popular in early 1990s. MD4 and MD5 are members of this category. Both MD functions are found to be insecure and not recommended for use any more. MD5 is a 128-bit hash function that was commonly used for file integrity checks.

## 2. Secure Hash Algorithms (SHAs)

- ❖ SHA-0: This is a 160-bit function introduced by NIST in 1993.
- ❖ SHA-1: SHA-1 was introduced later by NIST as a replacement of SHA-0. This is also a 160-bit hash function. SHA-1 was used commonly in SSL and TLS implementations – now deprecated.
- ❖ SHA-2: This category includes four functions defined by the number of bits of the hash: SHA-224, SHA-256, SHA-384 and SHA-512.
- ❖ SHA-3: This is the latest family of SHA functions. SHA3-224, SHA3-256, SHA3-384 and SHA3-512 are members of this family. SHA3 is a NIST-standardized version of [Keccak](#).
- ❖ [Keccak](#) uses a new approach called sponge construction instead of the commonly used Merkle-Damgard transformation.

**3. RIPEMD:** RIPEMD is the acronym for [RACE Integrity Primitives Evaluation Message Digest](#). It is based on the design ideas used to build MD4. There are multiple versions of RIPEMD, including 128-bit, 160-bit, 256-bit, and 320-bit.

**4. Whirlpool:** This is based on a modified version of [Rijndael cipher](#) known as W. It uses the [Miyaguchi-Preneel compression function](#), which is a type of one-way function used for the compression of two fixed length inputs into a single fixed length output. It is a single block length compression function:

# Cryptographic Hash functions

---

- ❖ Hash function:
  - ❖ takes an arbitrary length string as input
  - ❖ produces a fixed-size output (e.g 256 bits)
    - ❖ Easy to compute
    - ❖ Almost impossible to reverse
- ❖ Security properties:
  - ❖ collision-resistant
  - ❖ Hides the original String
    - ❖ Almost impossible to get the original string from the output
  - ❖ puzzle-friendly

# Hash functions & blockchain

---

- Hash functions are typically used to provide data integrity services
- Can be used as one-way functions
- Also to construct other cryptographic primitives, such as MACs and digital signatures
- Some applications used hash functions as a means of generating pseudo random numbers (PRNGs)
- Hash functions do not require a key

# Cryptographic hash functions

- ❖ An **important component of blockchain technology** is the use of cryptographic hash functions for many operations
- ❖ **Hashing** is a method of applying a cryptographic hash function to data, which calculates a relatively unique output (called a *message digest*, or just *digest*) for an input of nearly any size (e.g., a file, text, or image)
- ❖ It allows individuals to independently take input data, hash that data, and derive the same result – proving that there was no change in the data
- ❖ Even the **smallest change to the input** (e.g., changing a single bit) will result in a completely **different output digest**

# Three security properties of hash functions

---

- ❖ pre-image resistance
  - ❖ second preimage resistance
  - ❖ and collision resistance
-

# Pre-image resistance

---

- ❖  $h(x) = y$
- ❖  $h$  is the hash function,  $x$  is the input, and  $y$  is the hash
- ❖ The first security property requires that  $y$  cannot be reverse computed to  $x$
- ❖  $x$  is considered a *pre-image* of  $y$ , hence the name pre-image resistance
- ❖ This is also called one-way property

# Pre-image resistance: contd..



- ❖ *preimage resistant* means that they are one-way
- ❖ it is computationally infeasible to compute the correct input value given some output value
- ❖ (e.g., given a digest, find  $x$  such that  $\text{hash}(x) = \text{digest}$ )

# Second pre-image resistance

---

- ❖ This property requires that given  $x$  and  $h(x)$  , it is almost impossible to find any other message  $m$  , where  $m \neq x$  and *hash of m = hash of x*
- ❖  $h(m) = h(x)$
- ❖ This property is also known as weak collision resistance

# Second pre-image resistance: contd..



- ❖ *Second preimage resistant* means one cannot find an input that hashes to a specific output
- ❖ More specifically, cryptographic hash functions are designed so that given a specific input, it is computationally infeasible to find a second input which produces the same output
- ❖ (e.g., given  $x$ , find  $y$  such that  $\text{hash}(x) = \text{hash}(y)$ )
- ❖ The only approach available is to exhaustively search the input space, but this is computationally infeasible to do with any chance of success

# Collision resistance

---

- ❖ This property requires that two different input messages should not hash to the same output
  - ❖  $h(x) \neq h(z)$
  - ❖ This property is also known as **strong collision resistance**
-

# Collision resistance: contd..



- ❖ *Collision resistant* means that one cannot find two inputs that hash to the same output
- ❖ More specifically, it is computationally infeasible to find any two inputs that produce the same digest
- ❖ (e.g., find an  $x$  and  $y$  which  $\text{hash}(x) = \text{hash}(y)$ )

# Use of SHA 256 in blockchain

---

- ❖ A specific cryptographic hash function used in many blockchain implementations is the **Secure Hash Algorithm (SHA)** with an output size of 256 bits (**SHA-256**)
  - ❖ Many computers support this algorithm in hardware, making it fast to compute
  - ❖ SHA-256 has an output of 32 bytes (1 byte = 8 bits, 32 bytes = 256 bits), generally displayed as a 64-character hexadecimal string
  - ❖ This means that there are  $2^{256} \approx 10^{77}$  possible digest values
-

# Examples of Input Text and Corresponding SHA-256 Digest Values in blockchain

---

Input Text	SHA-256 Digest Value
1	0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
2	0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35
Hello, World!	0xdffd6021bb2bd5b0af676290809ec3a53191dd81c7f70a4b28688a362182986f

# SHA 256 hash values in blockchain

- ❖ There are an infinite number of possible input values and a finite number of possible output digest values
- ❖ So, it is possible but **highly unlikely** to have a collision where  $\text{hash}(x) = \text{hash}(y)$
- ❖ i.e., the hash of two different inputs producing the same digest is unlikely
- ❖ SHA-256 is said to be collision resistant
- ❖ That is: to find a collision in SHA-256, one would have to execute the algorithm, on average, about  $2^{128}$  times (which is 340 **undecillions**, or more; roughly  $3.402 \times 10^{38}$ )

Undecillion: a number equal to 1 followed by 36 zeros

What is **hash rate**? Hashes per second in a blockchain network

# Use of crypto hash functions within a blockchain network

- ❖ Address derivation
- ❖ Creating unique identifiers
- ❖ Securing the block data
  - ❖ *a publishing node will hash the block data, creating a digest that will be stored within the block header*
- ❖ Securing the block header
  - ❖ *a publishing node will hash the block header*
  - ❖ *if the blockchain network utilizes a proof of work consensus model, the publishing node will need to hash the block header with different nonce values until the puzzle requirements have been fulfilled*
  - ❖ *the current block header's hash digest will be included within the next block's header, where it will secure the current block header data*

Because the block header includes a hash representation of the block data, the block data itself is also secured when the block header digest is stored in the next block

# Families of hash functions used in blockchain technology

---

- ❖ SHA-256
- ❖ Keccak (which was selected by NIST as the winner of a competition to create the SHA-3 hashing standard)
- ❖ RIPEMD-160

# Avalanche effect in hash functions



- ❖ Hash functions, due to their very nature, will always have some collisions (where two different messages hash to the same output)
- ❖ but they should be computationally infeasible to find
- ❖ Avalanche effect desirable in hash functions
- ❖ Avalanche effect specifies that a small change, even a single character change in the input text, will result in a totally different hash output

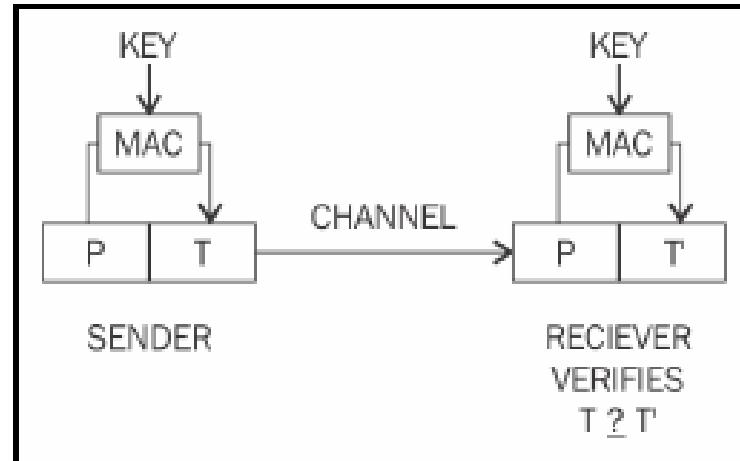
# Iterated hash functions

- ❖ the input message is compressed in multiple rounds on a block-by-block basis to produce the compressed output
- ❖ iterated hash function ex: [Merkle-Damgard construction](#)
- ❖ based on the idea of **dividing the input data into equal sizes of blocks** and then feeding them through the compression functions in an **iterative manner**
- ❖ The **collision resistance of the property of compression functions ensures that the hash output is also collision-resistant**
- ❖ Compression functions can be built using block ciphers
- ❖ other constructions of compression functions ex: [\*Miyaguchi-Preneel\*](#) and [\*Davies-Meyer\*](#)

# Keyed hash functions

Message Authentication codes (MACs)

- ❖ MACs using block ciphers and CBC
- ❖ HMACs (hash-based MACs): two methods
  1. *Secret prefix:*  $M = \text{MAC}_k(x) = h(k // x)$
  2. *Secret suffix:*  $M = \text{MAC}_k(x) = h(x // k)$



Operation of a MAC function

# Cryptographic Nonce

---

- ❖ A cryptographic nonce is an arbitrary number that is only used once
- ❖ A cryptographic nonce can be combined with data to produce different hash digests per nonce:  
**hash (data + nonce) = digest**
- ❖ Only changing the nonce value provides a mechanism for obtaining different digest values while keeping the same data
- ❖ This technique is utilized in the proof of work consensus model



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

# **BLOCKCHAIN TECHNOLOGY**

**BITS F452**

**1<sup>st</sup> Sem 2022-23**

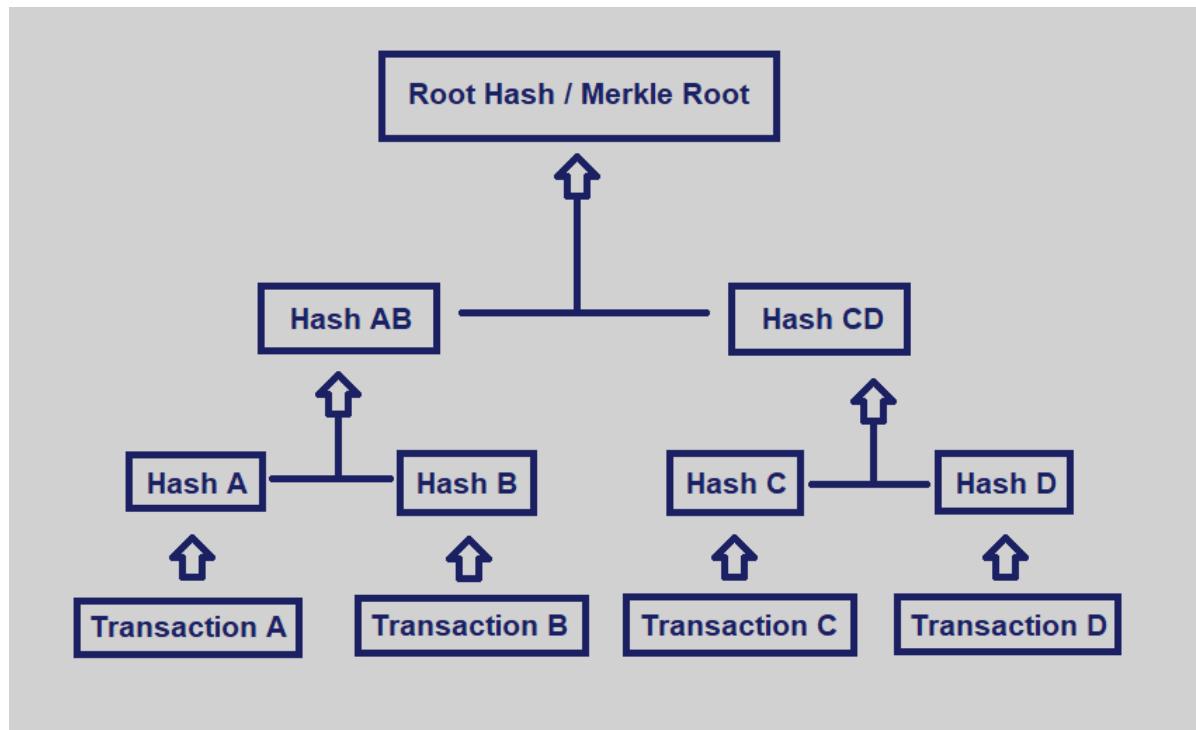
**Lecture 5**

**Merkle Trees and Blockchain Technology**



# Blockchain Merkle tree datastructure

- ❖ **Merkle tree:** A data structure where the data is hashed and combined until there is a singular root hash that represents the entire structure



# Merkle tree

---

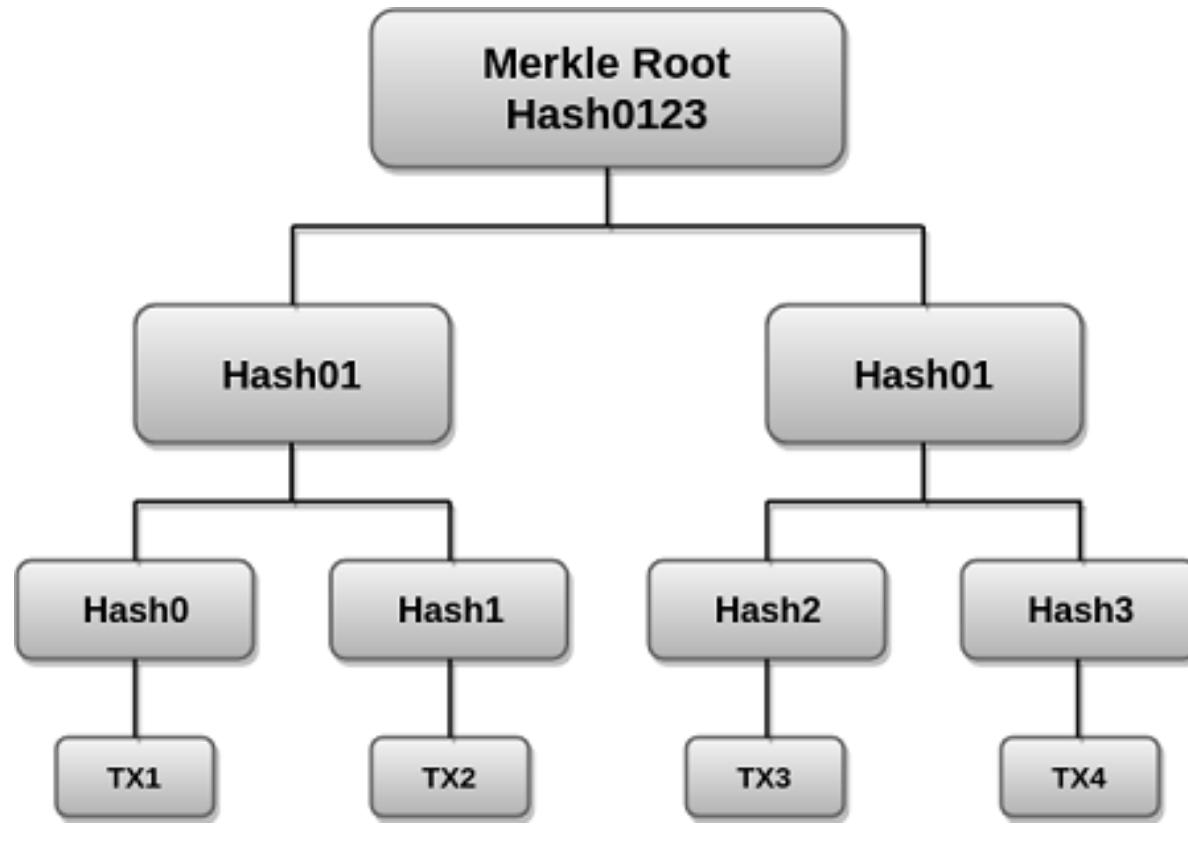
- ❖ Merkle tree is a fundamental part of blockchain technology
  - ❖ It is a mathematical **data structure** composed of hashes of different blocks of data, and which serves as a summary of all the transactions in a block
  - ❖ It also allows for efficient and secure verification of content in a large body of data
  - ❖ It also helps to verify the consistency and content of the data. Both Bitcoin and Ethereum use Merkle Trees structure. **Merkle Tree is also known as Hash Tree.**
  - ❖ The concept of Merkle Tree is named after **Ralph Merkle**, who patented the idea in **1979**
  - ❖ Fundamentally, it is a data structure tree in which every **leaf node** labelled with the hash of a data block, and the **non-leaf node** labelled with the cryptographic hash of the labels of its child nodes. The leaf nodes are the lowest node in the tree
-

# How do Merkle trees work?

---

- ❖ A Merkle tree stores all the transactions in a block by producing a **digital fingerprint** of the entire set of **transactions**
  - ❖ It allows the user to verify whether a transaction can be included in a block or not
  - ❖ Merkle trees are created by **repeatedly calculating hashing pairs of nodes until there is only one hash left**
  - ❖ This hash is called the **Merkle Root**, or the **Root Hash**
  - ❖ The Merkle Trees are constructed in a **bottom-up approach**
-

# Construction of Merkle trees



# Merkle tree....

---

- ❖ Every leaf node is a hash of transactional data, and the non-leaf node is a hash of its previous hashes
  - ❖ Merkle trees are binary trees, so it requires an even number of leaf nodes
  - ❖ If there is an odd number of transactions, the last hash will be duplicated once to create an even number of leaf nodes
-

# Merkle tree....

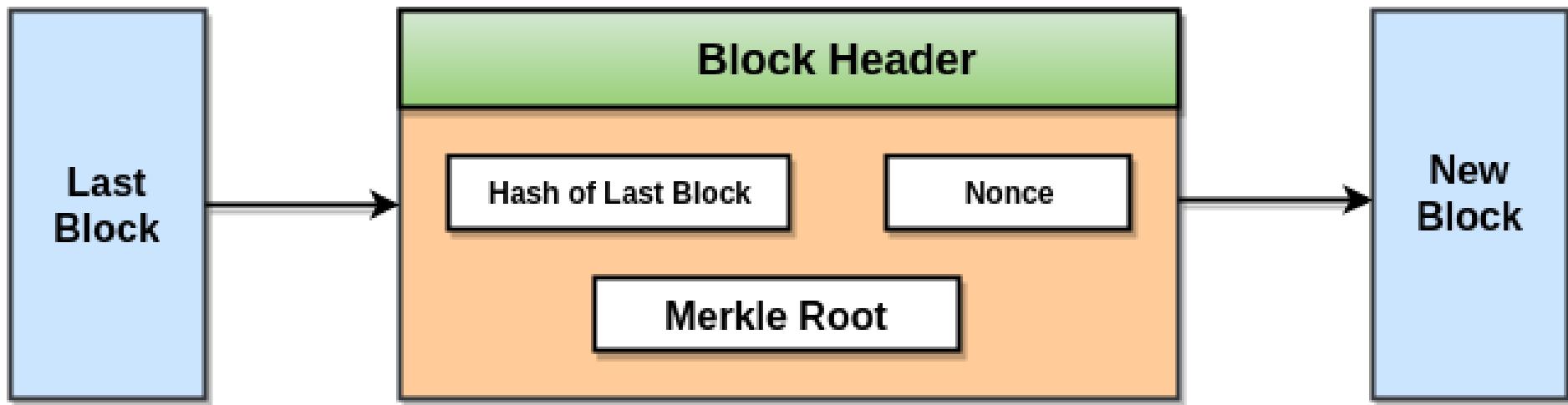
- ❖ The above example is the most common and simple form of a Merkle tree, i.e., **Binary Merkle Tree**
- ❖ There are four transactions in a block: **TX1, TX2, TX3, and TX4**
- ❖ There is a top hash which is the hash of the entire tree, known as the **Root Hash**, or the **Merkle Root**
- ❖ Each of these is **repeatedly hashed**, and **stored in each leaf node**, resulting in Hash 0, 1, 2, and 3
- ❖ Consecutive pairs of leaf nodes are then **summarized in a parent node** by hashing **Hash0** and **Hash1**, resulting in **Hash01**, and separately hashing **Hash2** and **Hash3**, resulting in **Hash23**.
- ❖ The two hashes (**Hash01** and **Hash23**) are then hashed again to produce the **Root Hash or the Merkle Root**

# Merkle root

---

- ❖ Merkle Root is stored in the **block header**
  - ❖ Ex: the block header is the part of the bitcoin block which gets hash in the process of mining
  - ❖ It contains the hash of the last block, a Nonce, and the Root Hash of all the transactions in the current block in a Merkle Tree
  - ❖ Having the Merkle root in block header makes the transaction **tamper-proof**
  - ❖ Root Hash includes the hashes of all the transactions within the block, thus, these transactions may result in saving the disk space
-

# Merkle root



# Merkle tree and integrity of blockchain data



- ❖ If any single detail of transactions or order of the transaction's changes, then these changes reflected in the hash of that transaction
- ❖ This change would cascade up the Merkle Tree to the Merkle Root, changing the value of the Merkle root and thus invalidating the block
- ❖ Merkle tree allows for a quick and simple test of whether a specific transaction is included in the set or not

# Benefits of Merkle trees

---

- ❖ provides a means to **maintain the integrity** and validity of data
  - ❖ helps in **saving the memory or disk space** as the proofs, computationally easy and fast
  - ❖ their proofs and management **require only tiny amounts of information to be transmitted** across networks
-

# Merkle tree as transaction summary

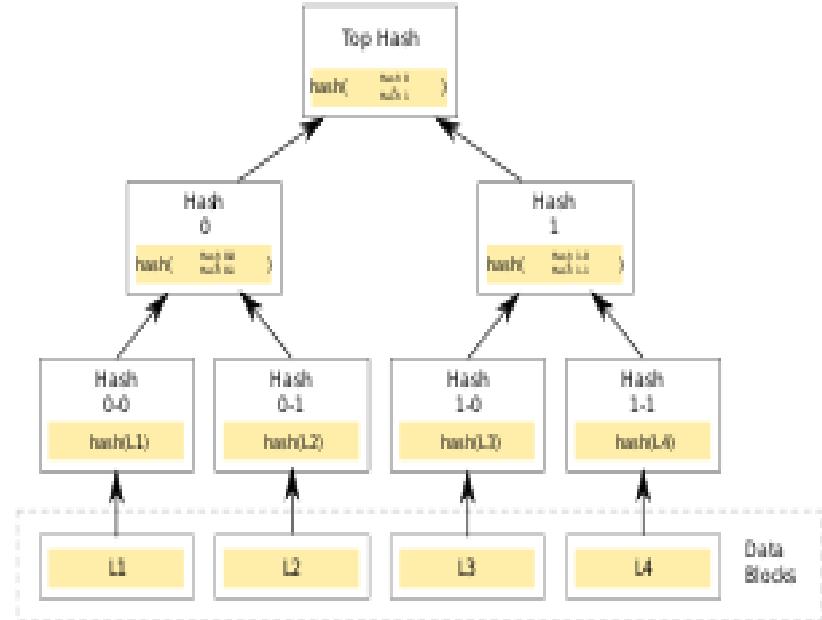
- the transaction history is **summarized** with Merkle trees
- efficiently and **securely** records a chain of events
- any attempt to edit or change a past transaction will also require a **recalculation** of all subsequent blocks of transactions

**cryptographic commitment scheme,**

- the root of the Merkle tree is seen as a commitment
- leaf nodes may be revealed and proven to be part of the original commitment

# Attacks on Merkle tree datastructure

- ❑ The Merkle hash root does not indicate the tree depth, enabling a second-preimage attack in which an attacker creates a document other than the original that has the same Merkle hash root
- ❑ For the example, an attacker can create a new document containing two data blocks, where the first is hash 0-0 + hash 0-1, and the second is hash 1-0 + hash 1-1



Solution against **second preimage attack**: Certificate Transparency: when computing leaf node hashes, a 0x00 byte is prepended to the hash data, while 0x01 is prepended when computing internal node hashes

**WHAT IS A MERKLE TREE?**  
Merkle trees are a data structure that is used for secure verification of data in a large content pool.

**HOW DO MERKLE TREES WORK?**

- MERKLE ROOT**  
The root of the upside-down tree which contains the hash code of the whole tree
- LEAF NODES**  
The leaf nodes are the nodes containing transaction data hashes
- NON-LEAF NODES**  
The non-leaf nodes contains the transactions

**HOW BITCOIN USES MERKLE TREE**

- It utilizes SHA-256 hash function in the Merkle tree
- Miners include transactions into a block by hashing and then including it in the Merkle tree
- The benefit of using Merkle trees in bitcoin include simple payment verification (SPV)
- SPV nodes are lightweight nodes where it is not necessary to download the whole ledger to verify transactions
- The SPV node stores the block headers of the longest chains

**HOW ETHEREUM USES MERKLE TREES**

Ethereum being Turing-complete utilize Merkle Trees through a complex version of it: Merkle Patricia Tree

**USE-CASES**

- InterPlanetary File System
- Distributed control system
- Used in No-SQL databases

**BENEFITS**

- Offers data integrity
- Saves disk space
- Provides efficient verification

# What problem is addressed by Merkle trees in blockchain?



**The Problem:** when it comes to the decentralized blockchain network, each data is copied among the nodes. So, it is a challenge to efficiently access data. The challenge is also to make a copy of the data and share it among nodes. Also, the shared data needs to be verified for each of the receiving nodes.

**The Solution:** Merkle Trees enable decentralized blockchains to share data, verify them, and make them trustworthy. It organizes data in such a way that not much processing power is needed to share and verify data. It also facilitates the secure transaction thanks to the use of hash functions and cryptography.

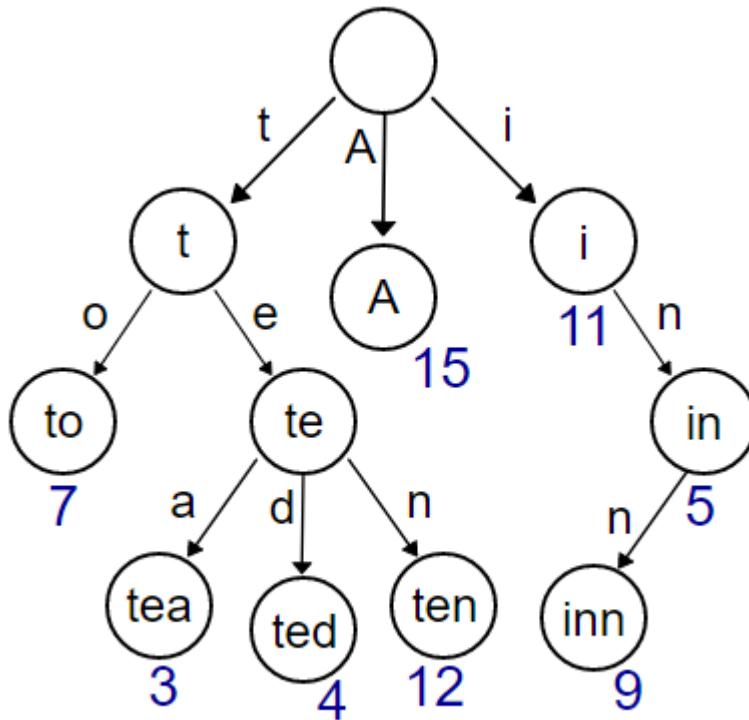
Satoshi Nakamoto was the first person who implemented Merkle trees in blockchain technology via Bitcoin. His usage opened up a new branch of computer science where there is no need for a centralized authority. He also used Merkle trees to an excessive degree and used Fast Merkle trees.

However, the concept was first introduced by Ralph Merkle, who patented it in 1979

# Patricia trees

- ❖ Merkle trees allow secure and efficient verification of large data sets
- ❖ Patricia trees are complex versions of Merkle trees
- ❖ A trie or a digital tree is an ordered tree data structure used to store a dataset
- ❖ **Practical Algorithm to Retrieve Information Coded in Alphanumeric (Patricia)**
- ❖ Also known as **Radix tree**: a compact representation of a **trie** in which a node that is the only child of a parent is merged with its parent
- ❖ **Merkle-Patricia tree**, based on the definitions of Patricia and Merkle, is a tree that has a root node that contains the hash value of the entire data structure

# Merkle-Patricia-tree - Ethereum



*A trie for keys "A", "to", "tea", "ted", "ten", "i", "in", and "inn".*

# Ethereum Modified Merkle Patricia Trie

innovate

achieve

lead

modified merkle patricia trie defines three different node types; extension, branch and leaf.

**Block Header,  $H$  or  $B_H$**   
**stateRoot,  $H_r$**   
 Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied

Hash function:  
**KECCAK256 ()**

World State Trie

ROOT: Extension Node		
prefix	shared nibble(s)	next node
0	a7	

Simplified World State, $\sigma$										Values
Keys										Values
a	7	1	1	3	5	5				45.0 ETH
a	7	7	d	3	3	7				1.00 WEI
a	7	f	9	3	6	5				1.1 ETH
a	7	7	d	3	9	7				0.12 ETH

Branch Node

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	value
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-------

Leaf Node

prefix	key-end	value
2	1355	45.0ETH

Extension Node

prefix	shared nibble(s)	next node
0	d3	

Leaf Node

prefix	key-end	value
2	9365	1.1ETH

Prefixes

- 0 - Extension Node, even number of nibbles
- 1□ - Extension Node, odd number of nibbles,
- 2 - Leaf Node, even number of nibbles
- 3□ - Leaf Node, odd number of nibbles
- = 1<sup>st</sup> nibble
- 1 nibble = 4 bits

Branch Node

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	value
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-------

Leaf Node

prefix	key-end	value
3□	7	1.00WEI

Leaf Node

prefix	key-end	value
3□	7	0.12ETH

# Properties of Merkle Trees

- ❖ data verification and synchronization; fingerprinting of data
- ❖ allows efficient mapping of huge data and small changes made to the data can be easily identified
- ❖ to know where data change has occurred, we can check if data is consistent with root hash and we will not have to traverse the whole structure but only a small part of the structure
- ❖ The root hash is used as the fingerprint for the entire data

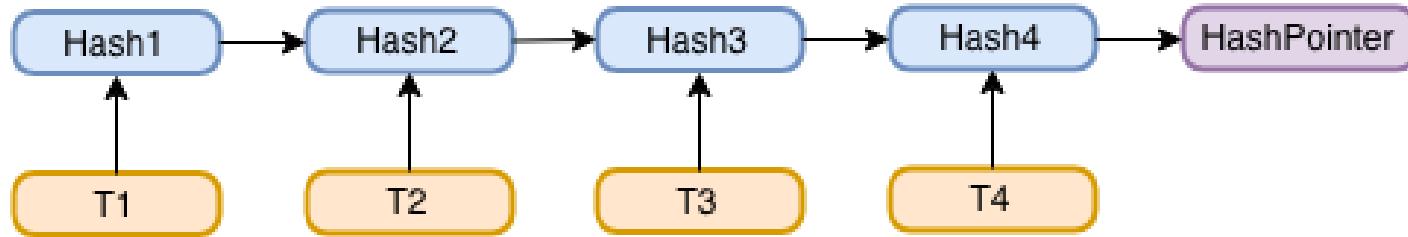
Binary Merkle tree	
Operation	Complexity
Space	$O(n)$
Searching	$O(\log n)$
Traversal	$O(n)$
Insertion	$O(\log n)$
Deletion	$O(\log n)$
Synchronization	$O(\log n)$



# Merkle tree – popular applications

- ❖ Amazon Dynamo DB
- ❖ Git
- ❖ Blockchain

# Bitcoin blocks may store upwards of 1,000 transactions



**Needed** To validate, we need all of these values

**Computed** We can compute these, given the values above

**Known** This is the information we need to store

Ex: Bitcoin blocks may store upwards of 1,000 transactions

# Merkle trees and their use in Blockchain transaction validation



## How to build Merkle trees? The steps:

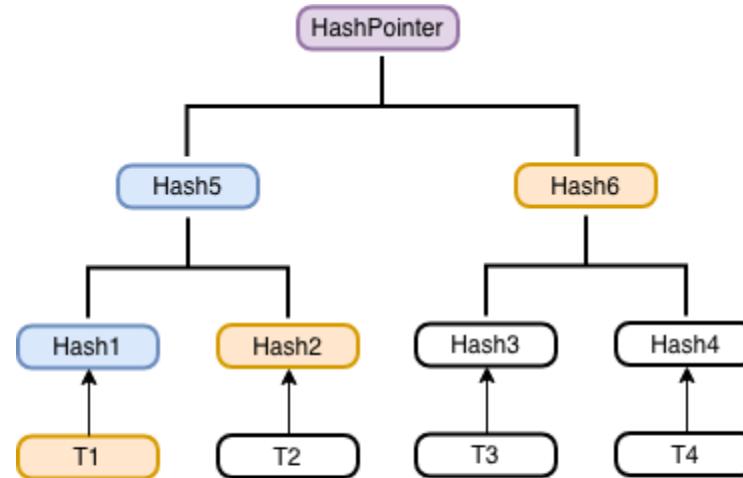
1. transactions are first paired together based on an agreed approach
2. if an odd number of transactions exists then the **last** is **duplicated**
3. a hash is created for each pair
4. hashes are then paired up and hashed, the results paired up and hashed, and so on, until eventually a single hash value remains
5. this is the **hash pointer** for the block

# Transaction validation

---

- ❖ We receive a transaction that we wish to validate
  - ❖ With a Merkle tree the only information we need is the hash of the sibling of the transaction, and the hash of each sibling up the branch to the hash pointer
  - ❖ We need only re-compute this branch to produce a hash pointer than matches the one we have stored, not the entire tree
  - ❖ This means we no longer need all of those other transactions
-

# Transaction validation Using Merkle tree



- Needed To validate, we need all of these values
- Computed We can compute these, given the values above
- Known This is the information we need to store

- ❖ massively reduces the size of the payload we receive
- ❖ results in less computation during the validation step
- ❖ an improvement of  $\log_2(n)$  vs  $n$  operations, where  $n$  is the number of transactions in the block

# Merkle trees as a smart way to hash



## A scenario

Consider two users Alice and Bob

- ❖ Bob stores a set of items for Alice
- ❖ Alice keeps a single value
- ❖ Alice can validate the Items returned to her

## Cryptographic hash

- Arbitrary Input size
- Output size is fixed
- $H(x)$  is easy to compute
- Finding any  $x, x'$  s.t.  $H(x) = H(x')$ , should be computationally hard
- The output should also appear “random”

# Approach 1:

Alice keeps the hash of the entire set  $H(\text{[red box]}, \text{[blue box]}, \text{[green box]})$

## Validation Of An Item

- ❖ Bob sends all of the items to Alice
- ❖ Alice computes the hash of the items
- ❖ Alice compares the result to the value she has saved

$$H(\text{[red box]}, \text{[blue box]}, \text{[green box]}) == H( \text{[red box]}, \text{[blue box]}, \text{[green box]} )$$

## Problem with Approach 1

- ❖ Bob must send Alice the entire set for validation
- ❖ Let  $n$  be the size of the set; we will have  $O(n)$  complexity for validating a single item

# Approach 2:

---

## Validating An Item

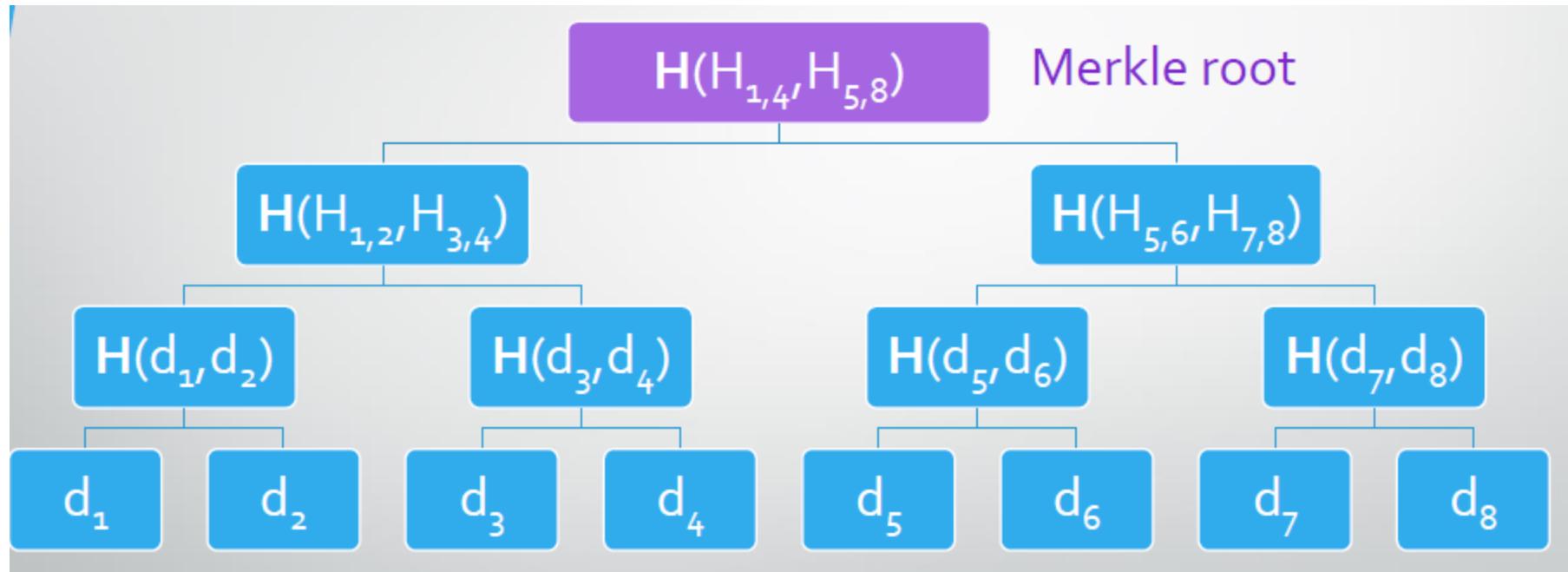
Bob sends Alice an item  $d$  and a logarithmic size proof

Alice computes a function of the item and proof

Alice compares the result to the value she has saved

**ce86b7dde40 == F( , )**

# Merkle Tree and hash - validation in blockchain



# Validating An Item using Merkle tree

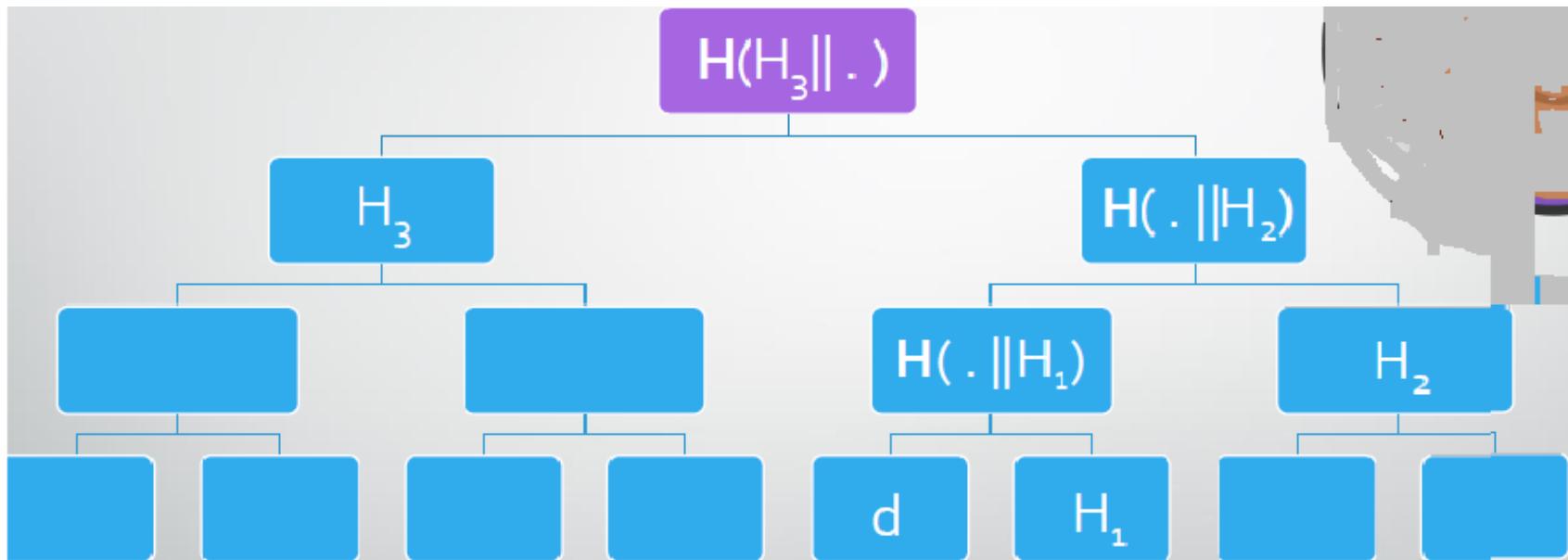
---

- ❖ Bob sends Alice an item d and a logarithmic size proof
- ❖ Alice uses the proof to compute the Merkle root
- ❖ Alice compares the result to the root she has saved

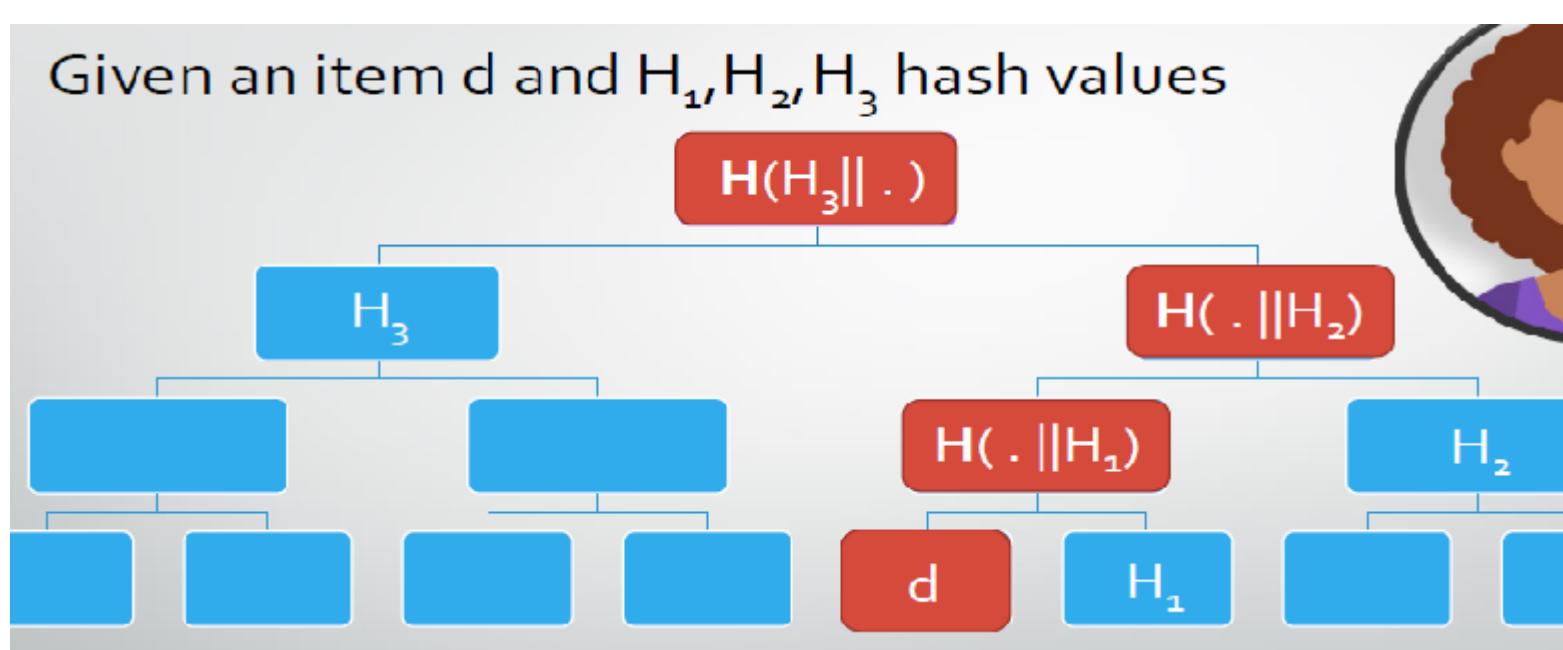
Merkle root ==  $F( \quad , \quad )$

# Alice computes the root

Given an item d and H1,H2,H3 hash values

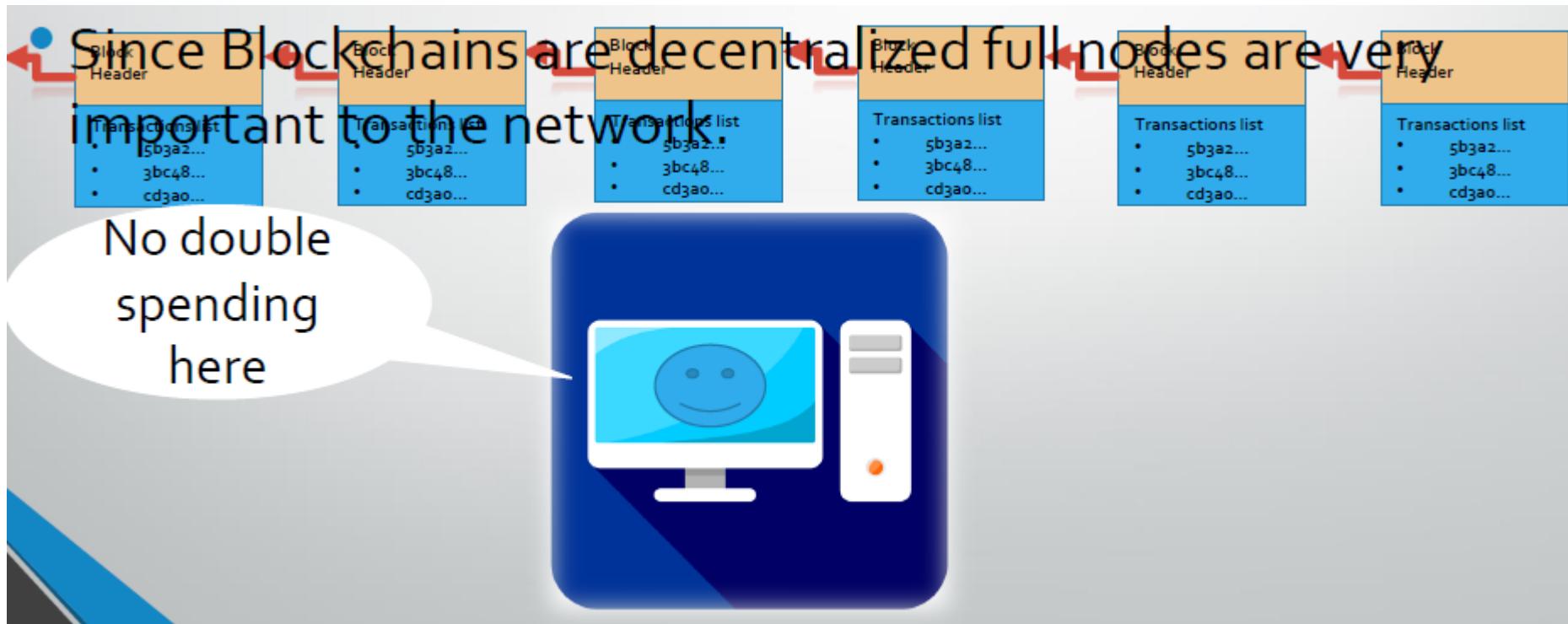


# What if d is not valid?



# Blockchain – Full nodes

- Full nodes are nodes in the Blockchain network that store the entire Blockchain in order to validate new transactions



# How Merkle tree helps in blockchain?

---

- ❖ Blockchain can take up a lot of memory
- ❖ The numbers are constantly growing
- ❖ Some devices such as mobiles cannot spare that much space

Database size	
Bitcoin	149 GB (December 2017)
Ethereum	400 GB (February 2018)

# A Block in the Blockchain

---

- ❖ Blocks consist of two main parts:

## Block Header

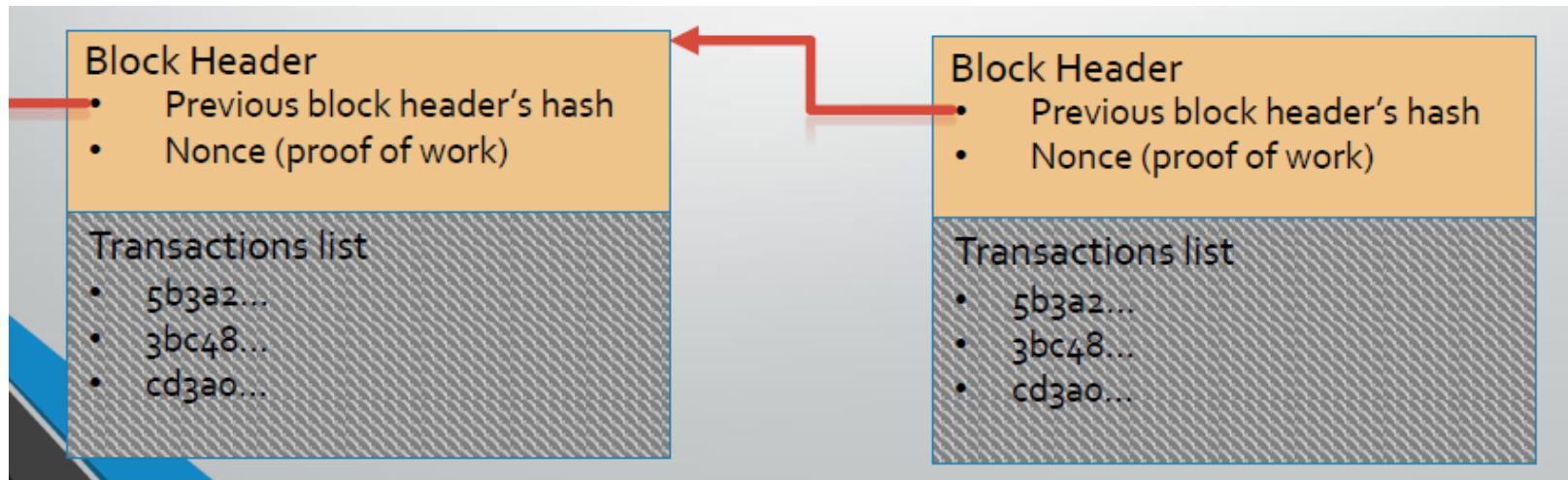
- Previous block header's hash
- Nonce (proof of work)

## Transactions list

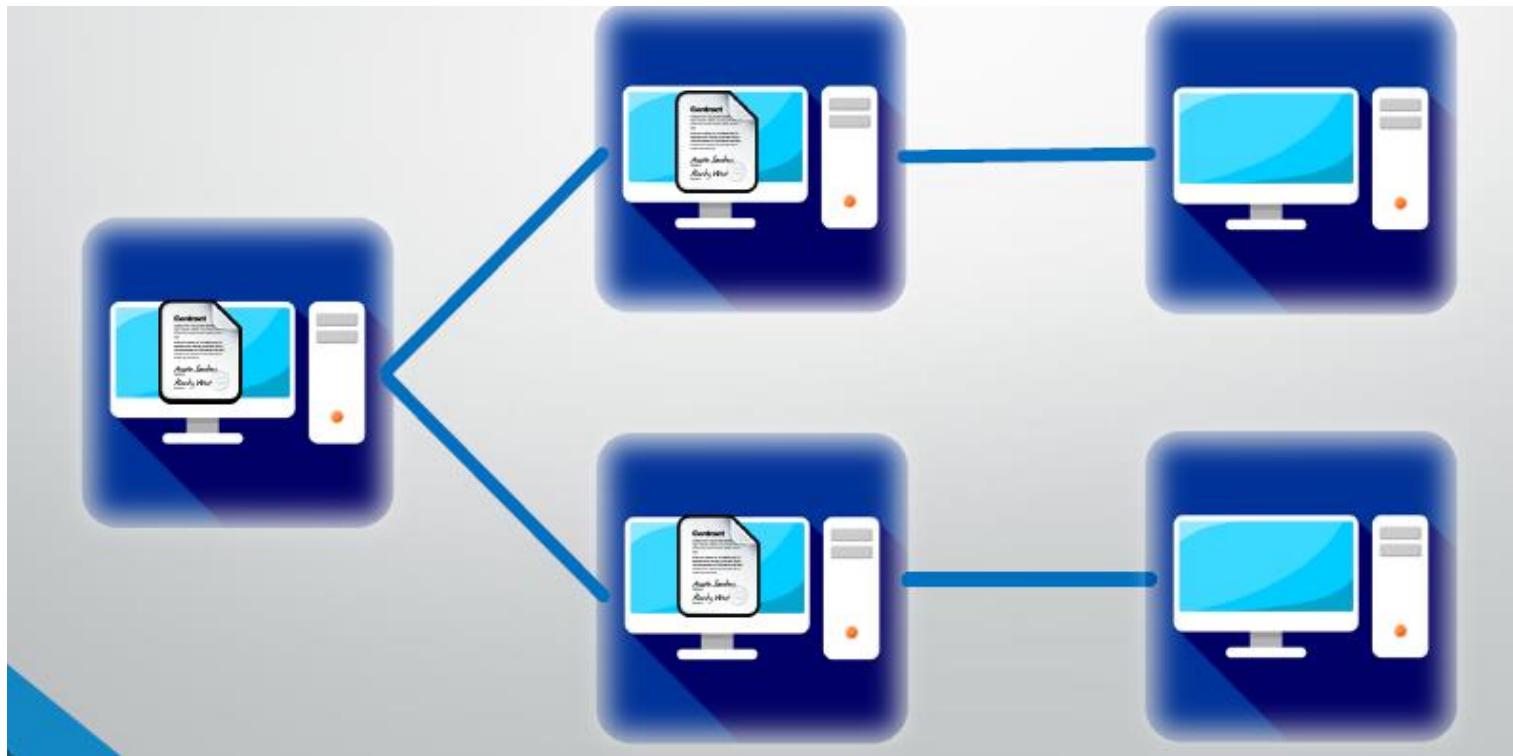
- 5b3a2...
- 3bc48...
- cd3ao...

# Light Nodes

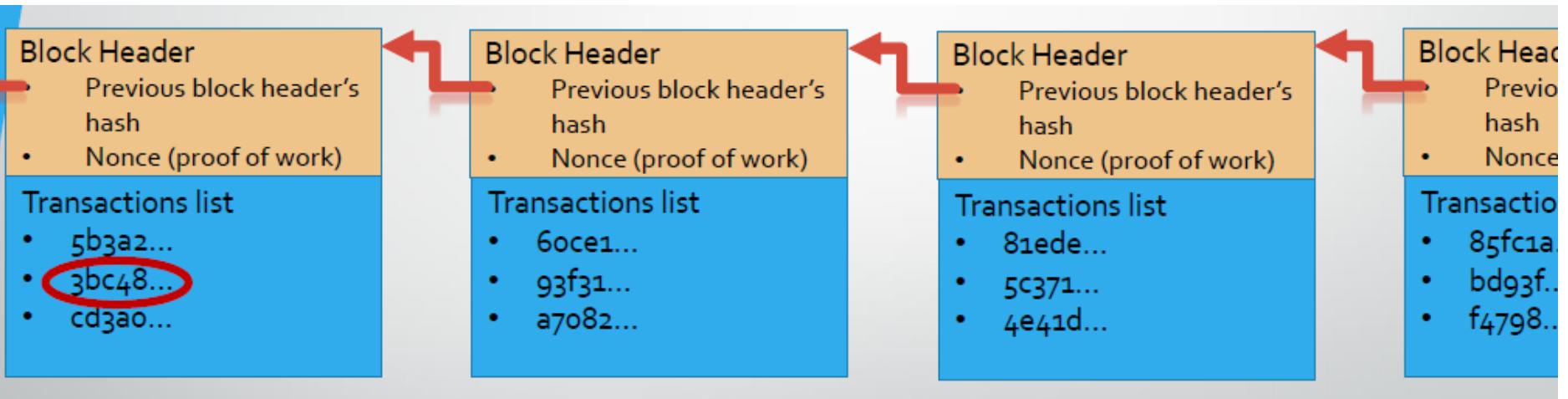
- ❖ Light nodes were created for simple clients who want to save on storage
- ❖ They only store the chain of block headers
- ❖ They follow the **longest chain rule**, without validating transactions



# Light Nodes & Full Nodes



# Accepting payment



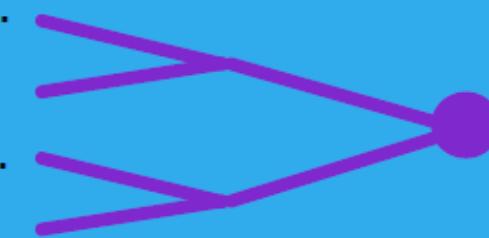
# Merkle Root In Header

## Block Header

- Previous block header's hash
- Nonce (proof of work)
- Merkle root

## Transactions list

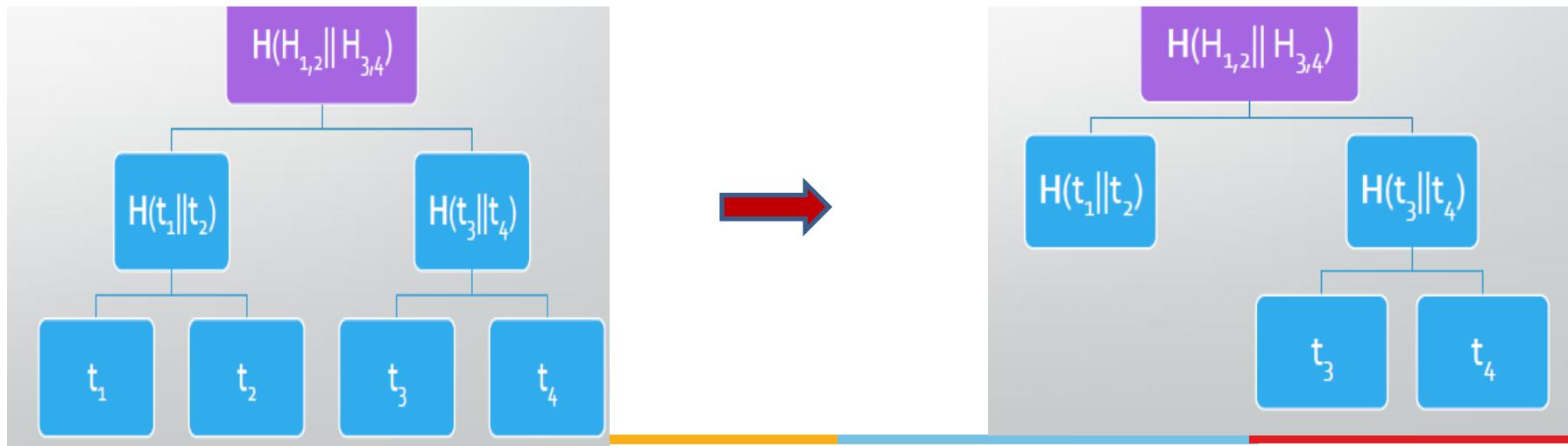
- 5b3a2...
- 3fc48...
- cd3ao...
- 73e7c...



- Put the Merkleroot of transactions in the header
- Now light nodes can request transactions from full nodes, and know that they were from a block (just like Alice did with Bob)

# Forgetting Spent Transactions

- ❖ Having the Merkle root in the header has an advantage
- ❖ It helps in freeing storage in full nodes by forgetting transactions that are already spent
- ❖ Suppose that  $t_1, t_2, t_3$  have all been spent; we can get rid of most of them
- ❖ Now we are storing  $H(t_1||t_2)$  in the block instead of  $t_1$  and  $t_2$



# Summarizing Merkle Trees

---

- ❖ Merkle trees are a smart way to hash
  - ❖ Allow for easier storage of blockchains, allowing headers to represent the entire block in a concise way
  - ❖ Also allow us to forget the transaction IDs of spent transactions
-

# What is double spending in blockchain and how to avoid it?

---

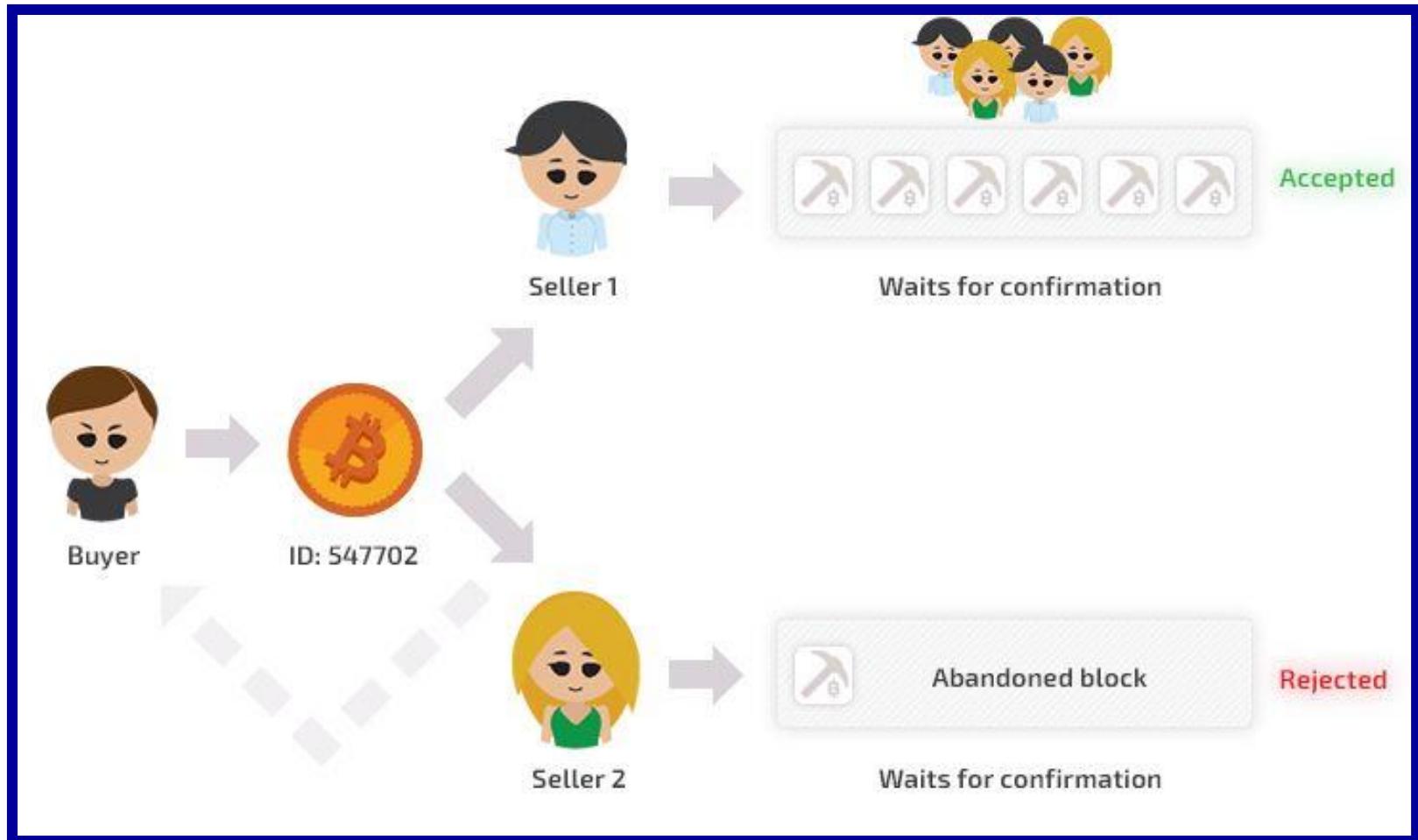
- ❖ Double-spending occurs **when a blockchain network is disrupted and cryptocurrency is essentially stolen**
  - ❖ The attacker/malicious user would send a copy of the currency transaction to make it look legitimate, or might erase the transaction altogether
  - ❖ not common, but double-spending attack does occur
- examples of double-spending attacks:

Allow bad actors to “reverse” a crypto transaction:

*These are the three types of probable attacks in the bitcoin system*

- **Finney attacks**
  - **race attacks**
  - **51% attacks**
-

# Double spending attack in blockchain



# Finney attack

---

- ❖ In this case, the attacker is the miner
  - ❖ The miner mines a block with his transaction and does not release it in the system
  - ❖ He now uses the same coins in a second transaction and then releases the pre-mined block
  - ❖ Obviously, the second transaction would be rejected eventually by other miners, but this will take some time
  - ❖ To mitigate this risk, the seller should wait for at least **six block confirmations** before releasing the goods
-

# Race Attack

---

- ❖ The attacker may send the same coin to different vendors in rapid succession, probably by using two different machines
  - ❖ If the vendors do not wait for the block confirmation before delivering the goods, they will very soon realize that the transaction was rejected during the mining process
  - ❖ The solution to this kind of attack is that the vendor must wait for at least one block confirmation before sending out the goods
-

# 51% attack

---

- ❖ This attack is based on an impractical assumption that somebody owns 51% of the computing power of the network
  - ❖ The attacker in this kind of attack mines a private blockchain where he double-spends the coins
  - ❖ As he owns the majority of computing power, he is guaranteed that his private blockchain at some point of time would be longer than the chain of “honest” network
  - ❖ He then releases his private blockchain in the system making all the transactions earlier recorded in the honest blockchain to be invalid
  - ❖ **This kind of attack is fictitious** as it is very expensive to acquire computing power which equals or exceeds 51% of the computing power of the entire network
-

# How does blockchain determine double-spending?

---

- ❖ The way that users detect tampering such as an attempt to double-spend in practice is **through hashes**, long strings of numbers that serve as proof of work (**PoW**)
  - ❖ Put a given set of data through a hash function (bitcoin uses SHA-256), and it will **only ever generate one hash**
-

# How does blockchain prevent double-spending?

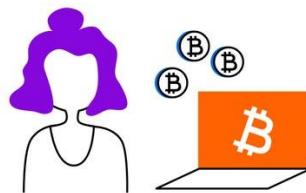


Ex: Bitcoin's network prevents double-spending by combining complementary security features of the blockchain network and its decentralized network of miners to verify transactions before they are added to the blockchain

# Double spending - illustration

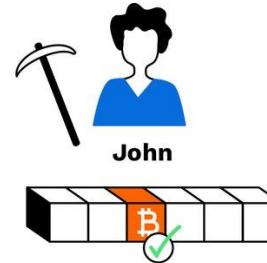
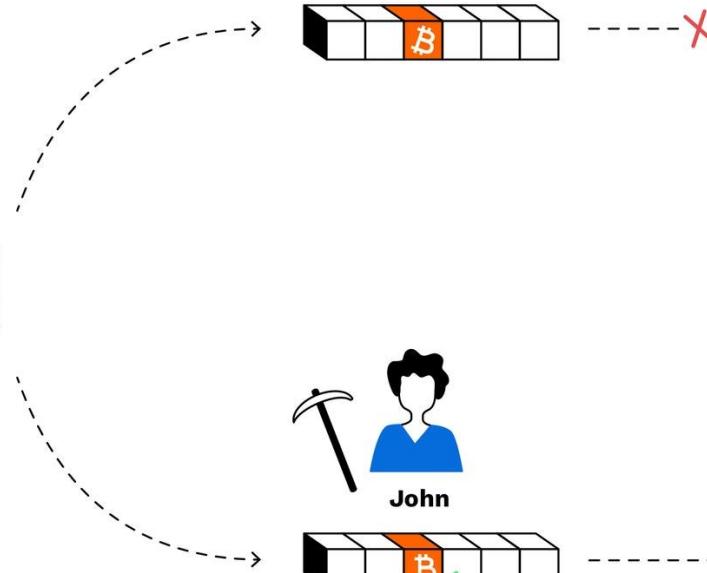
## What is Double Spending

and why is it such a problem?



**Alice**

Without exception, all Bitcoin transactions are included in a block of transactions. Each block has a timestamp with encoded information that makes it more difficult to manipulate the blockchain.



The mechanism of the blockchain ensures that the party spending the bitcoins is the real owner.



**Katy**

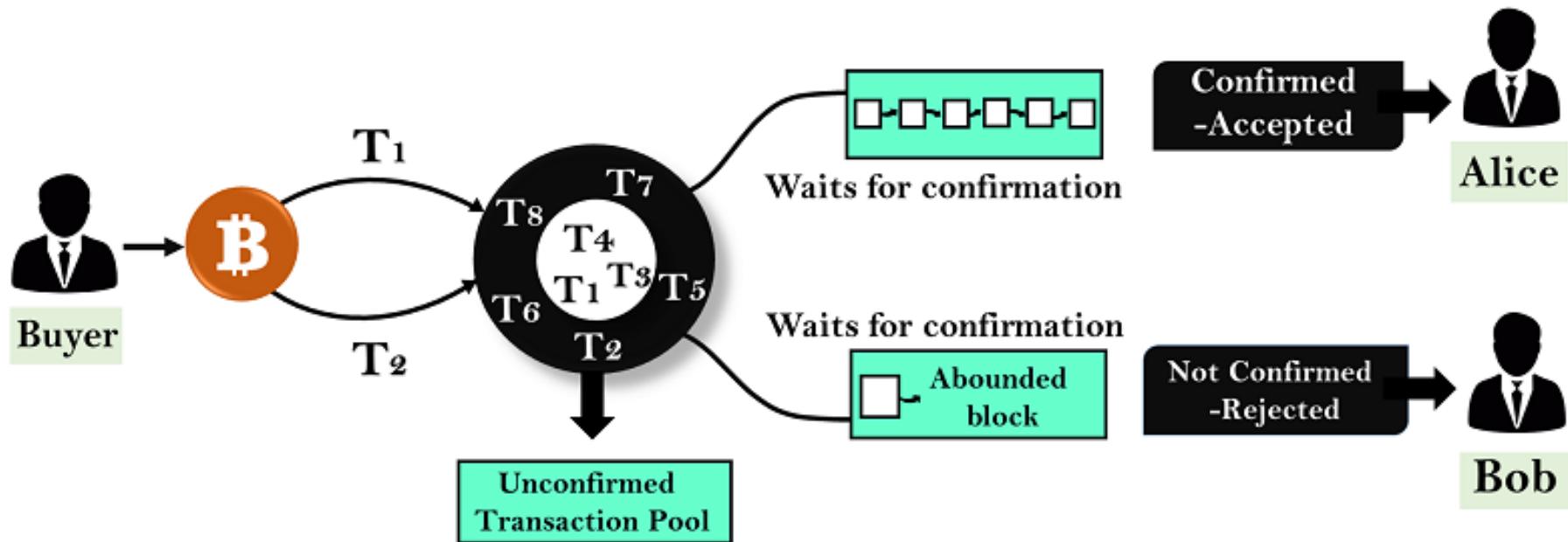
Double spending is a type of deceit where the same money is promised to two parties but only delivered to one.



**Bob**

The technology behind Bitcoin ensures that the party who spends the bitcoins is the real owner by only processing verified transactions.

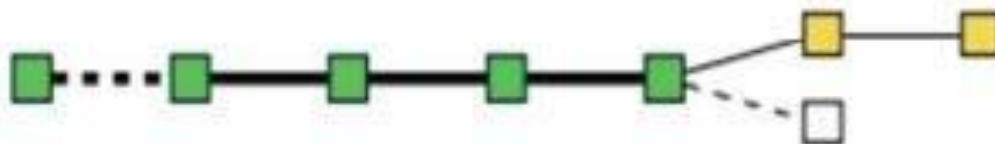
# Double spending – another illustration



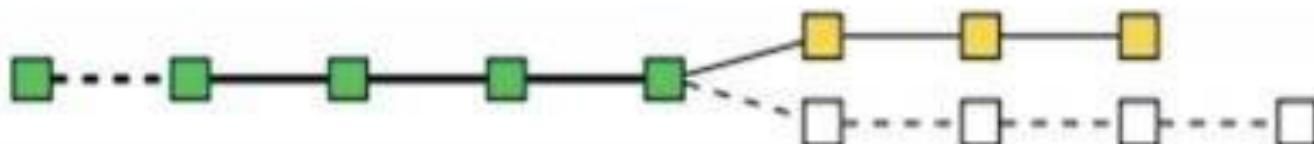
# Double spending attack



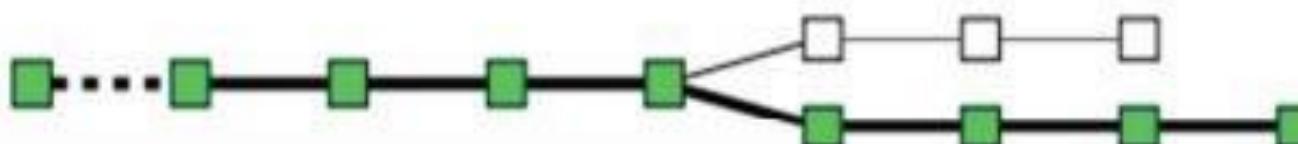
(a) Initial state of the blockchain in which all transactions are considered as valid.



(b) Honest nodes continue extending the valid chain by putting yellow blocks, while the attacker secretly starts mining a fraudulent branch.

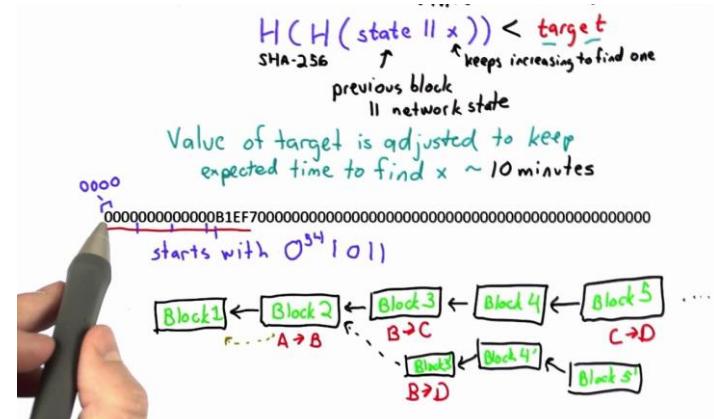
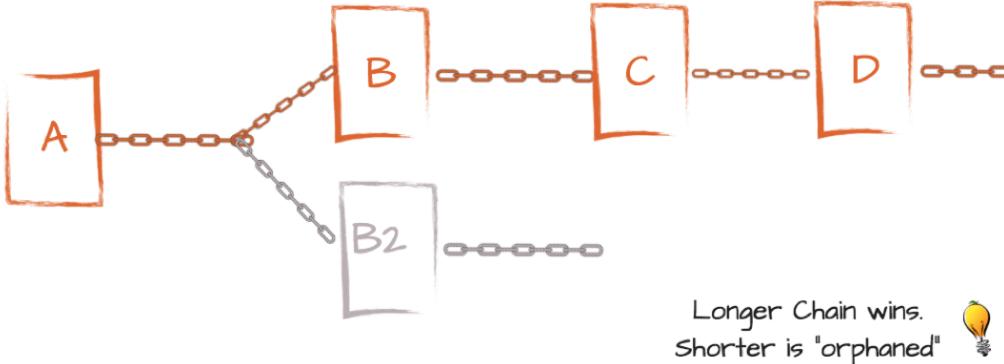


(c) The attacker succeeds in making the fraudulent branch longer than the honest one.

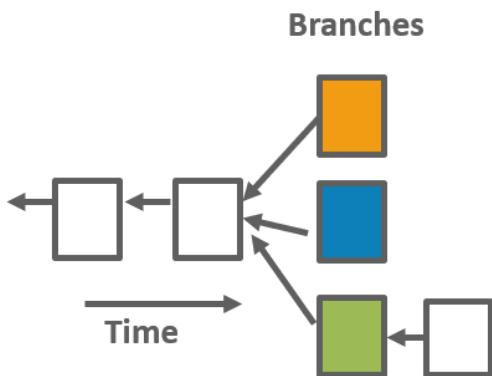
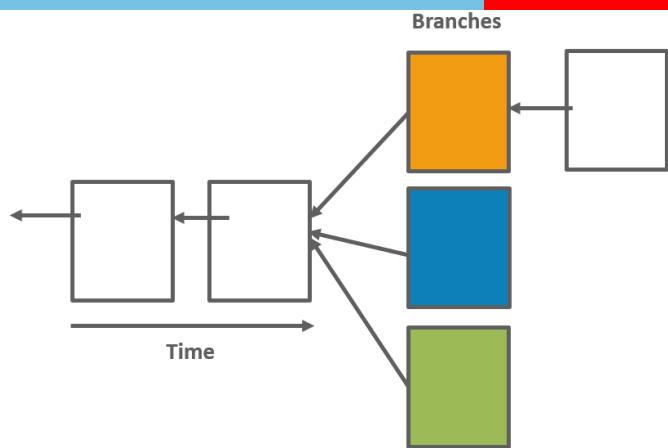


# Longest Chain Rule in Blockchain Technology

- The **longest chain** is **what individual nodes accept as the valid version of the blockchain**
  - The rule that nodes adopt the longest chain of blocks allows every node on the network to agree on what the blockchain looks like, and therefore agree on the same transaction history
  - The Longest Chain Rule ensures **that network will recognise the “chain with most work” as the main chain**
  - The chain with the most work is typically (not always) the longest of the forks

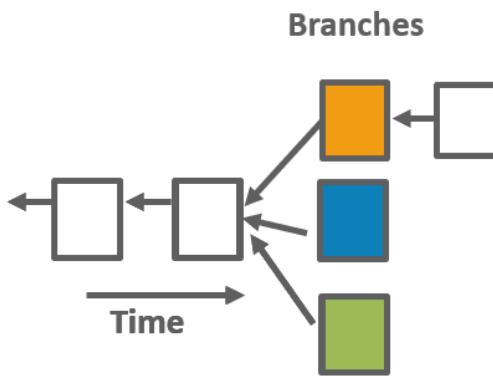


# Contd..



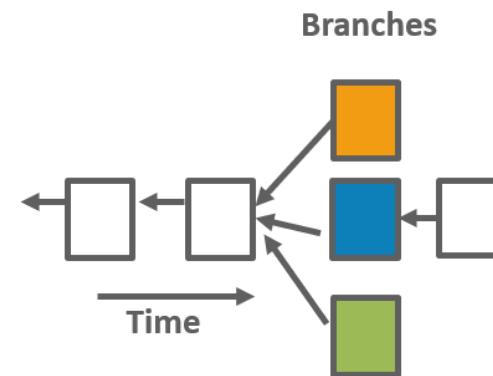
**Paul's Blockchain**

Paul received green block first.  
Hence, he builds the next block on top of green



**Robert's Blockchain**

Similarly, Robert received orange first. Hence, he builds the next block on top of orange

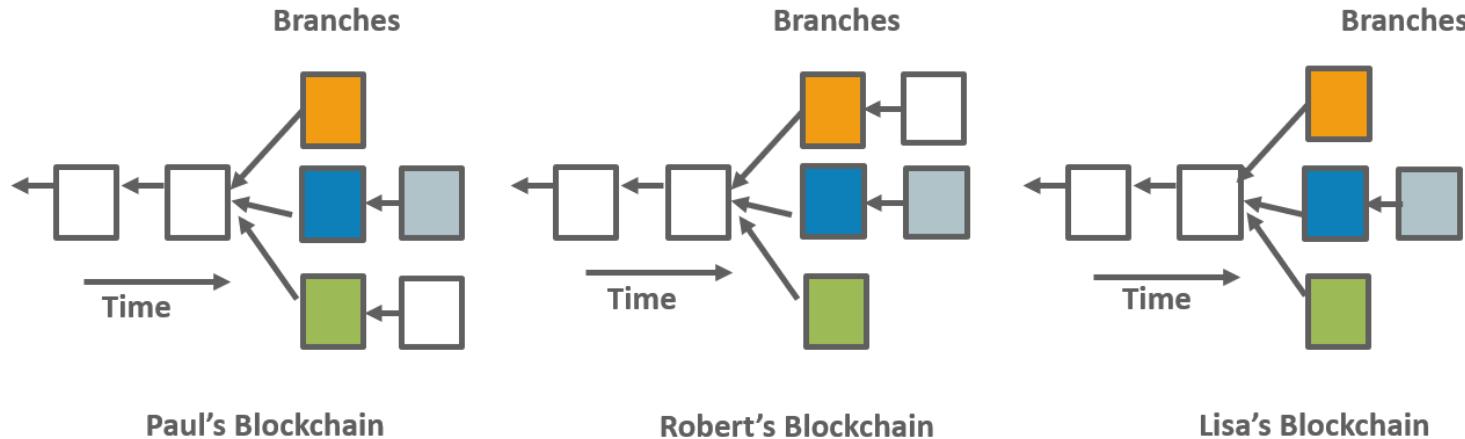


**Lisa's Blockchain**

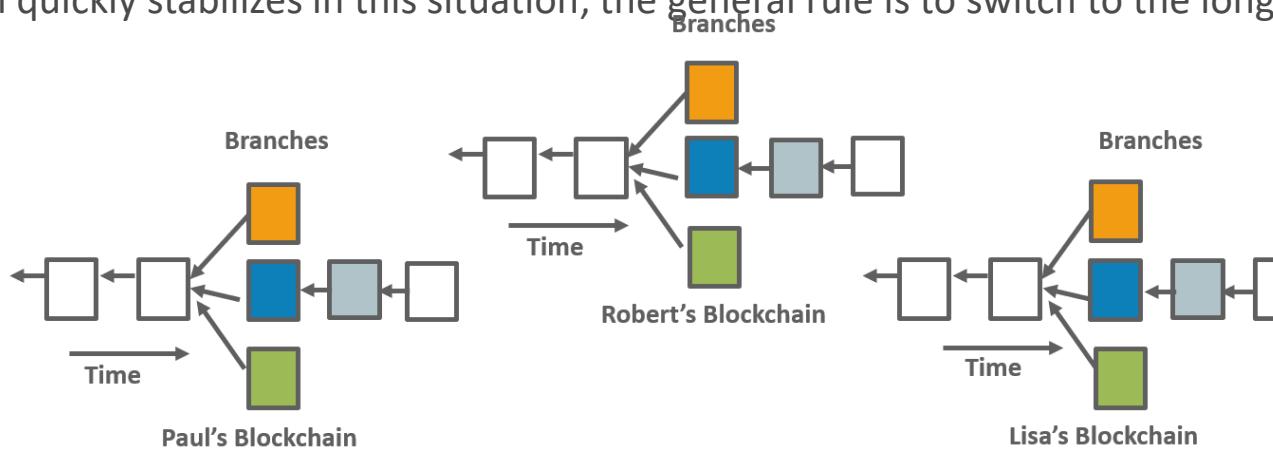
Also, Lisa received blue first.  
Hence, she builds the next block on top of blue

# Contd..

The tie gets broken when someone solves the next block because it is very rare for this situation to happen multiple times in a row



Blockchain quickly stabilizes in this situation; the general rule is to switch to the longest chain available



# Contd..

- The Blockchain quickly stabilizes
- Every node is now in agreement with the current state of the ledger

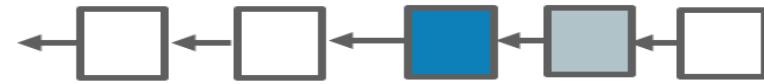
**Paul's Blockchain**



**Robert's Blockchain**



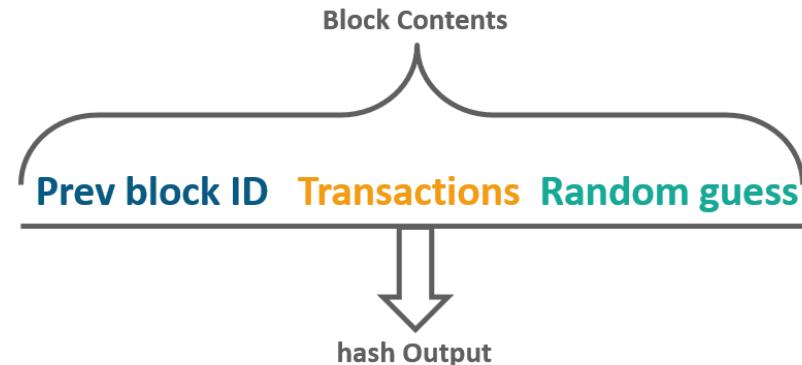
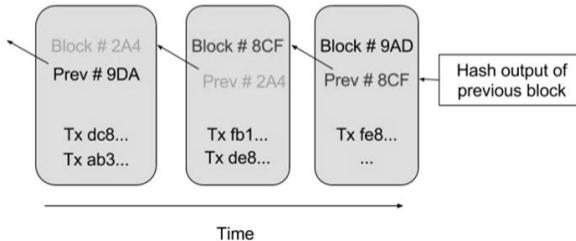
**Lisa's Blockchain**



# Can someone hack the system?

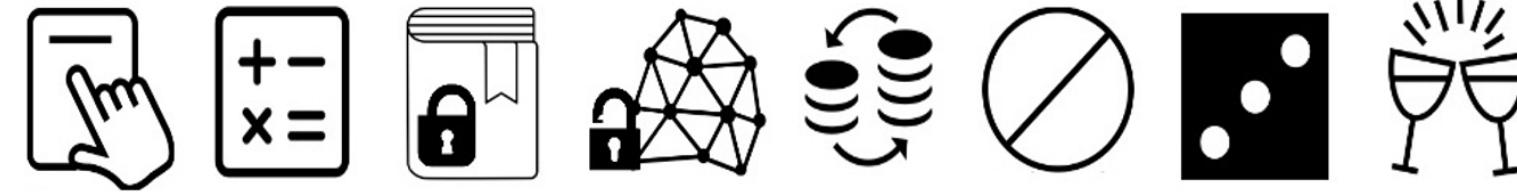
What if Someone tries to Hack the System?

Since blockchain is a back-linked distributed database of records, when a block is formed, the cryptographic hash output becomes the identifier of that block, which ties into the next block, creating a chain of blocks



- ✓ Hence, the blockchain is secured by the strong cryptographic algorithm and there is no way to alter any record
- ✓ If someone tries to alter any transaction in any of the blocks, the hash of the block changes and consequently hash of all the previous blocks will change
- ✓ The nodes will not arrive at **the consensus and hence, the fraud can easily be detected**

# Access to single source of truth



The use of Mathematics To create a secure ledger This enables transactions Without the need for third parties

Access to shared single source of truth



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

# **BLOCKCHAIN TECHNOLOGY**

**BITS F452**

**1<sup>st</sup> Sem 2022-23**

**Lecture 6**

**Decentralized identity management**

**&**

**Bitcoin as a blockchain application**



# Bitcoin – the open source P2P money

---

- ❖ Bitcoin is an innovative payment network
  - ❖ Bitcoin uses peer-to-peer technology to operate with no central authority or banks
  - ❖ managing transactions and the issuing of bitcoins is carried out collectively by the network
  - ❖ **Bitcoin is open-source; its design is public, nobody owns or controls Bitcoin and everyone can take part**
  - ❖ Through many of its unique properties, Bitcoin allows exciting uses that could not be covered by any previous payment system
  - ❖ **Bitcoin is the first application of blockchain technology**
-

# Digital currencies – the genesis

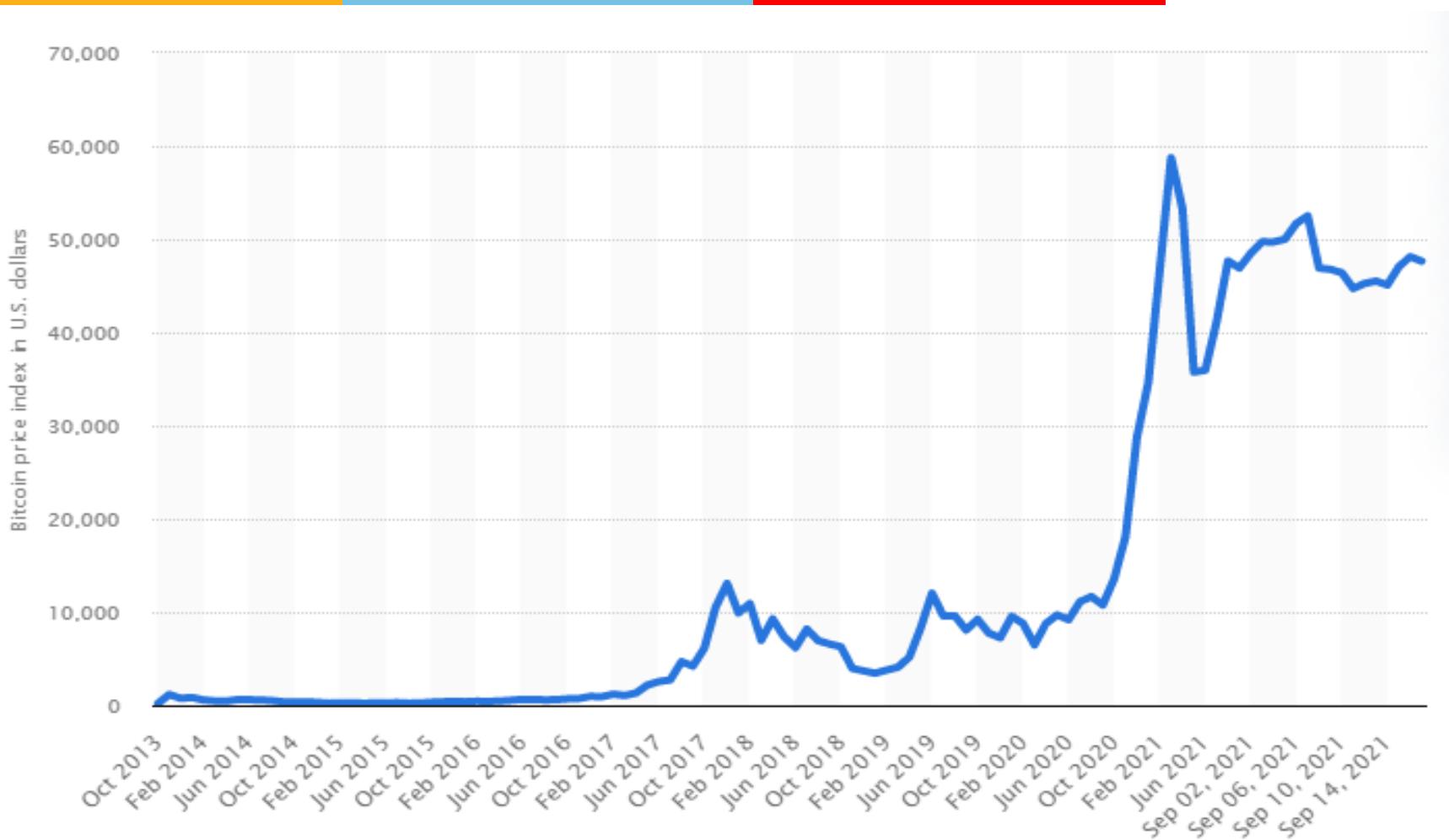
---

- ❖ In 1982, *David Chaum* proposed a scheme that used blind signatures to build untraceable digital currency
  - ❖ In 1990, *David Chaum* proposed a refined version named **e-cash** (blinded signature, private identification data, **detection of double spending**)
  - ❖ Adam Back's **hashcash**, introduced in 1997 - originally proposed to thwart e-mail spam
  - ❖ **B-money** was proposed by *Wei Dai* in 1998, which introduced the idea of using Proof of Work to create money
  - ❖ Nick Szabo introduced the concept of **BitGold**
  - ❖ Tomas Sander and Ammon TaShama introduced an **e-cash scheme** in 1999 that, for the first time, used **Merkle trees** to represent coins and **zero knowledge proofs** to prove the possession of coins
  - ❖ **RPOW (Reusable Proof of Work)** was introduced by Hal Finney in 2004 and used the hashcash scheme
-

# Bitcoin - introduction

- ❖ In 2008, *Satoshi Nakamoto* wrote a paper on bitcoin, *Bitcoin: A Peer-to-Peer Electronic Cash System*
- ❖ Bitcoin is built on decades of cryptographic research such as the research in Merkle trees, hash functions, public key cryptography, and digital signatures
- ❖ Solution to the **Byzantine Generals problem** along with a practical solution of the **double-spend problem**

# Bitcoin price from October 2013 to September 17, 2021 (in U.S. dollars)



*Bitcoin hashrate - how many Bitcoins are being mined*

# Is the world running out of bitcoin?

---

- ❖ Unlike fiat currencies, Bitcoin's supply is finite
  - ❖ BTC has a maximum supply embedded in its design, of which roughly 89 percent had been reached in April 2021
  - ❖ It is believed that Bitcoin will run out by 2040, despite more powerful mining equipment
  - ❖ This is because mining becomes exponentially more difficult and power-hungry every four years, a part of Bitcoin's original design
  - ❖ Because of this, a Bitcoin mining transaction could equal the energy consumption of a small country in 2021
  - ❖ The growth of Bitcoin is also due to so-called *Network Effect*
-

# The Bitcoin bubble

- ❖ Only few cryptocurrency holders own a large portion of available supply (they are referred to as “**whales**”)
- ❖ Said to make up of two percent of anonymous ownership accounts, whilst owning roughly 92 percent of BTC
- ❖ most people who use cryptocurrency-related services worldwide are retail clients rather than institutional investors

# Definition of Bitcoin – is there a single definition?

- a protocol, a digital currency, a platform
- a combination of peer-to-peer network, protocols, and software that facilitate the creation and usage of the digital currency named bitcoin
- Bitcoin with a capital **B** is used to refer to the **Bitcoin protocol**
- bitcoin with a lowercase **b** is used to refer to bitcoin, the currency
- Nodes in this peer-to-peer network talk to each other using the Bitcoin protocol

# What bitcoin has achieved?

---

- ❖ Decentralization of currency for the first time
- ❖ Double spending problem solved
  - ❖(a user sends coins to two different users at the same time and they are verified independently as valid transactions)

# Keys and addresses

---

- ❖ Elliptic curve cryptography is used to generate public and private key pairs in the Bitcoin network
  - ❖ The bitcoin address is created by taking the corresponding public key of a private key and hashing it twice, first with the SHA256 algorithm and then with RIPEMD160
  - ❖ The resultant 160-bit hash is then prefixed with a version number and finally encoded with a Base58Check encoding scheme
  - ❖ The bitcoin addresses are 26-35 characters long and begin with digit 1 or 3
-

# A typical bitcoin address

---

**1ANAguGG8bikEv2fYsTBnRUmxB7QUcK58wt**

Or

*commonly encoded in a QR code for easy sharing*



# Types of bitcoin addresses

- ❖ P2PKH – starting with 1
- ❖ P2SH - starting with 3

Addresses should not be used more than once; otherwise privacy and security issues can arise

## Some bitcoin security issues:

- ❖ Anonymity issue
- ❖ Transaction malleability

Bitcoin transaction malleability is **an attack wherein someone changes a TX ID before it is confirmed or validated by the network**. Once a part of that TX ID is changed, that ripple effect affects the hash—and if the hash is altered, the transaction can't be confirmed

# Generating bitcoin address

<https://www.bitaddress.org/bitaddress.org-v3.3.0-SHA256-dec17c07685e1870960903d8f58090475b25af946fe95a734f88408cef4aa194.html>



# Public keys in bitcoin

---

- ❖ Bitcoin uses ECC based on the SECP256K1 standard
  - ❖ In public key cryptography, public keys are generated from private keys
  - ❖ A private key is randomly selected and is 256-bit in length
  - ❖ Public keys can be presented in an uncompressed or compressed format
  - ❖ Public keys are basically  $x$  and  $y$  coordinates on an elliptic curve and in an uncompressed format and are presented with a prefix of 04 in a hexadecimal format
-

# Public keys in bitcoin contd..

---

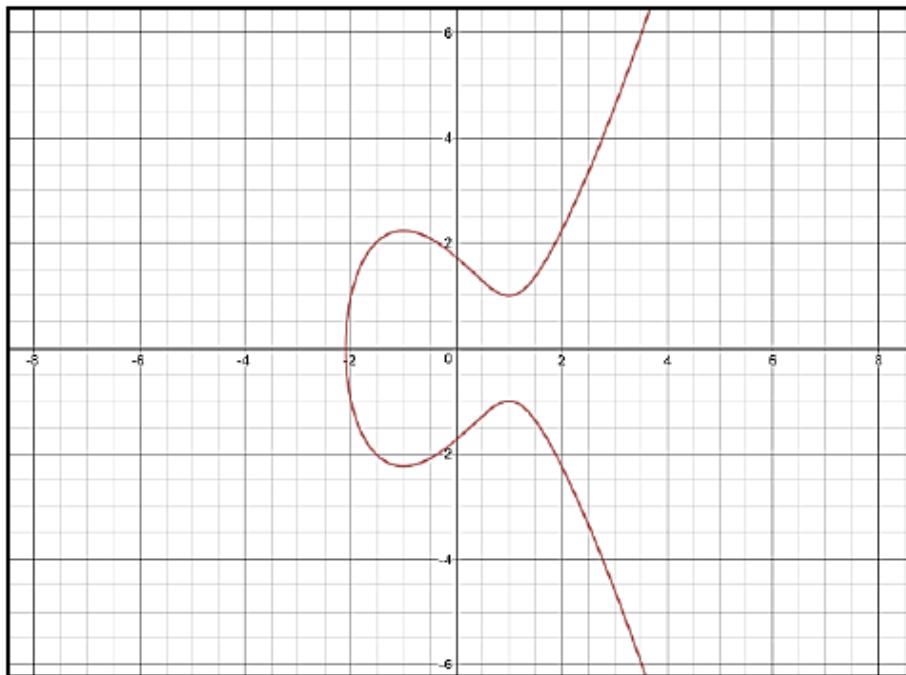
- ❖ X and Y coordinates are both 32-bit in length
  - ❖ In total, the compressed public key is 33 bytes long as compared to 65 bytes in the uncompressed format
  - ❖ The compressed version of public keys basically includes only the X part, since the Y part can be derived from it
  - ❖ The reason why the compressed version of public keys works is that the bitcoin client initially used uncompressed keys, but starting from bitcoin core client 0.6, compressed keys are used as the standard
-

# Identification of keys using various prefixes

---

- ❖ Uncompressed public keys used 0x04 as the prefix
- ❖ Compressed public key starts with 0x03 if the y 32-bit part of the public key is odd
- ❖ Compressed public key starts with 0x02 if the y 32-bit part of the public key is even

# ECC for Bitcoin



Elliptic curve over reals,  $a = -3$  and  $b = 3$

$$y^2 = x^3 + Ax + B \bmod p$$

Here A and B belong to prime finite field

If the ECC graph is visualized, it reveals that the  $y$  coordinate can be either below the  $x$  axis or above the  $x$  axis and as the curve is symmetric, only the location in the prime field is required to be stored

# Private keys in bitcoin

---

- ❖ Private keys are basically **256-bit numbers** chosen in the range specified by the SECP256K1 **ECDSA** recommendation
  - ❖ Any randomly chosen 256-bit number from 0x1 to 0xFFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFE BAAE DCE6 AF48 A03B BFD2 5E8C D036 4140 is a valid private key
  - ❖ Private keys are usually encoded using **Wallet Import Format (WIF)** in order to make them easier to copy and use
  - ❖ WIF can be converted into private key and vice versa
-

# Private keys in bitcoin

## Contd..

---

- ❖ **Mini Private Key Format** is sometimes used to encode the **key in under 30 characters** in order to allow storage where physical space is limited, for example, **etching on physical coins or damage-resistant QR codes**
- ❖ The bitcoin core client also allows the **encryption of the wallet** that contains the private keys

# Bitcoin currency units



DENOMINATION	ABBREVIATION	FAMILIAR NAME	VALUE IN BTC
Satoshi	SAT	Satoshi	0.00000001 BTC
Microbit	µBTC (uBTC)	Microbitcoin or Bit	0.000001 BTC
Millibit	mBTC	Millibitcoin	0.001 BTC
Centibit	cBTC	Centibitcoin	0.01 BTC
Decibit	dBTC	Decibitcoin	0.1 BTC
Bitcoin	BTC	Bitcoin	1 BTC
DecaBit	daBTC	Decabitcoin	10 BTC
Hectobit	hBTC	Hectobitcoin	100 BTC
Kilobit	kBTC	Kilobitcoin	1000 BTC
Megabit	MBTC	Megabitcoin	1000000 BTC

The smallest bitcoin denomination is Satoshi

# Base58Check encoding

Bitcoin addresses are encoded using the Base58check encoding

- used to limit the confusion between various characters, such as **0O11** as they can look the same in different fonts
- takes the **binary byte arrays** and converts them into human-readable strings

```
6  /**
7   * Why base-58 instead of standard base-64 encoding?
8   * - Don't want 0O11 characters that look the same in some fonts and
9   *   could be used to create visually identical looking data.
10  * - A string with non-alphanumeric characters is not as easily accepted as input.
11  * - E-mail usually won't line-break if there's no punctuation to break at.
12  * - Double-clicking selects the whole string as one word if it's all alphanumeric.
13  */
14 #ifndef BITCOIN_BASE58_H
```

base58.h source file in the bitcoin source code

# Vanity addresses

- ❖ As bitcoin addresses are based on base 58 encoding, it is possible to generate addresses that **contain human-readable messages**



Public address encoded in QR code

The vanity addresses are **addresses of cryptocurrencies, personalized and created respecting a series of parameters given by the users of said addresses**. This with the aim of making them more personal and easily identifiable, but without giving up the security they provide

# Generation of vanity addresses

- ❑ Vanity addresses are generated using a purely brute-force method
- ❑ Ex:



Vanity address generated from <https://bitcoinvanitygen.com/>

# Transactions in Bitcoin system

---

- ❖ **Transactions** are at the core of the bitcoin ecosystem
- ❖ Transactions can be as simple as:
  - ❖ just sending some bitcoins to a bitcoin address
  - ❖ or it can be quite complex depending on the requirements
- ❖ Each transaction is composed of at least one input and output
- ❖ Inputs can be thought of as coins being spent that have been created in a previous transaction and outputs as coins being created

# Transactions

---

- ❖ If a transaction is minting new coins, then there is no input and therefore no signature is needed
- ❖ If a transaction is to send coins to some other user (a bitcoin address), then it needs to be signed by the sender with their private key and a reference is also required to the previous transaction in order to show the origin of the coins
- ❖ Coins are unspent transaction outputs represented in Satoshis
  - Transactions are not encrypted and are publicly visible in the blockchain
  - Blocks are made up of transactions and these can be viewed using any online blockchain explorer

# 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

*Six is just a recommended number; the transaction can be considered final even after the first confirmation*

*The key idea behind waiting for six confirmations is that the probability of double spending is virtually eliminated after six confirmations.*

# What does a transaction contain?

- A transaction at a high level contains:
  - Metadata
  - Inputs
  - outputs
- Transactions are combined to create a block

# Parts of a transaction

Each input spends a previous output

The Main Parts Of  
Transaction 0

Version	Inputs	Outputs	Locktime
---------	--------	---------	----------

The Main Parts Of  
Transaction 1

Version	Inputs	Outputs	Locktime
---------	--------	---------	----------

Each output waits as an Unspent TX Output (UTXO) until a later input spends it

# Transaction components

## ✓ 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

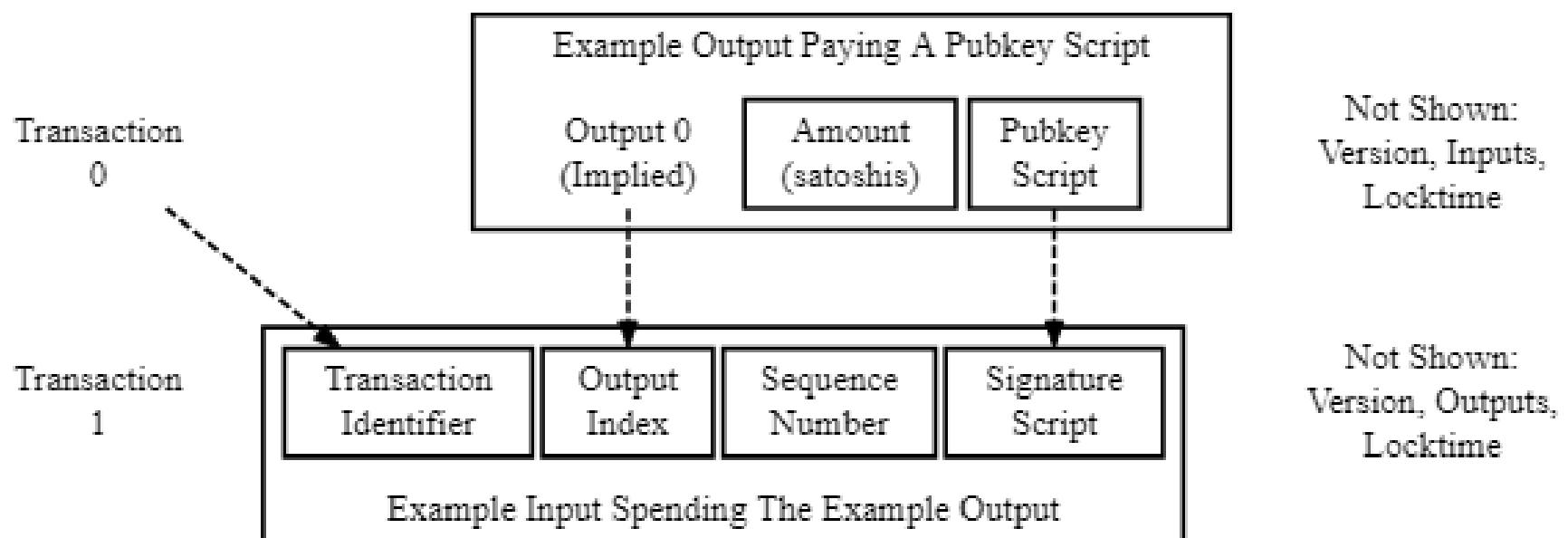
## ✓ Outputs

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

## ✓ Verification

Verification is performed using bitcoin's scripting language

# Spending an output



# 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.

# A sample decoded transaction

```
{
  "txid": "08af7960ca9255c67686296fb65452ed3f96f18831c9a3d8ea552e4cce5c4af",
  "hash": "08af7960ca9255c67686296fb65452ed3f96f18831c9a3d8ea552e4cce5c4af",
  "size": 226,
  "vsize": 226,
  "version": 1,
  "locktime": 969523,
  "vin": [
    {
      "txid": "3e553260a0a94860f7043eb6576e15e6cfecb2990aea961210ae1fde328bb08b0",
      "vout": 1,
      "scriptSig": {
        "asm":
"3045022100cfb31edabc62c82b41d12f651d2e3e013ee1a7ee2bb4526f3dda640e6d8d224502207d8d1d8e41350b9cdf36f389f942ab68c12
f113fe99014f5d6df6610407877d2[ALL] 037bc82d0078993f6943e7ff6e82e82da600f34edc8bca136331a9901c8bb60b0d",
        "hex":
"483045022100cfb31edabc62c82b41d12f651d2e3e013ee1a7ee2bb4526f3dda640e6d8d224502207d8d1d8e41350b9cdf36f389f942ab68c
12f113fe99014f5d6df6610407877d20121037bc82d0078993f6943e7ff6e82e82da600f34edc8bca136331a9901c8bb60b0d"
      },
      "sequence": 4294967294
    }
  ],
  "vout": [
    {
      "value": 2.30000000,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 07e78644a61343068fa8d4940a79976e758ac6ef OP_EQUALVERIFY OP_CHECKSIG",
        "hex": "76a91407e78644a61343068fa8d4940a79976e758ac6ef88ac",
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [
          "mgEkNzxV3qbYtDKEKTvo1VpgJ63Au619q2"
        ]
      }
    }
  ]
}
```

# Bitcoin transactions

---

<https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch05.html#:~:text=In%20essence%2C%20a%20bitcoin%20transaction,of%20thousands%20of%20bitcoin%20nodes.>

# The **script** language used in bitcoin



- ❖ Bitcoin uses a simple stack-based language called **script** to describe how bitcoins can be spent and transferred
- ❖ It is **not Turing complete** and has no loops to avoid any undesirable effects of long running/hung scripts on the bitcoin network
- ❖ This scripting language is based on a Forth-like syntax and uses a reverse polish notation in which every operand is followed by its operators
- ❖ It is evaluated from the left to the right using a **Last in First Out (LIFO)** stack

Forth combines a **compiler with an integrated command shell**, where the user interacts via subroutines called words

# More on scripts

---

- ❖ Scripts use various opcodes or instructions to define their operations
- ❖ **Opcodes** are also known as **words**, **commands**, or **functions**

Various categories of the scripting opcodes:

- Constants
  - flow control
  - Stack
  - bitwise logic
  - Splice
  - Arithmetic
  - Cryptography
  - lock time
-

# Evaluation of transaction script

---

- ❖ A transaction script is evaluated by combining **ScriptSig** and **ScriptPubKey**
  - ❖ **ScriptSig** is the **unlocking script**, whereas **ScriptPubKey** is the locking script
  - ❖ The transaction is evaluated to be spent; first, it is unlocked and then it is spent
  - ❖ **ScriptSig** is provided by the user who wishes to **unlock the transaction**
  - ❖ **ScriptPubkey** is part of the transaction output and **specifies the conditions** that need to be fulfilled in order to spend the output
  - ❖ Outputs are locked by the **ScriptPubKey (Locking script)** that contains the conditions, when met will unlock the output, and coins can then be redeemed
-

# Commonly used opcodes

---

- All Opcodes are declared in the `script.h` file in the bitcoin reference client source code

<https://github.com/bitcoin/bitcoin/blob/master/src/script/script.h>

---

# Most commonly used opcodes

Opcode	Description
OP_CHECKSIG	This takes a public key and signature and validates the signature of the hash of the transaction. If it matches, then TRUE is pushed onto the stack; otherwise, FALSE is pushed.
OP_EQUAL	This returns 1 if the inputs are exactly equal; otherwise, 0 is returned.
OP_DUP	This duplicates the top item in the stack.
OP_HASH160	The input is hashed twice, first with SHA-256 and then with RIPEMD-160.
OP_VERIFY	This marks the transaction as invalid if the top stack value is not true.
OP_EQUALVERIFY	This is the same as OP_EQUAL, but it runs OP_VERIFY afterwards.
OP_CHECKMULTISIG	This takes the first signature and compares it against each public key until a match is found and repeats this process until all signatures are checked. If all signatures turn out to be valid, then a value of 1 is returned as a result; otherwise, 0 is returned.

# Types of transactions

---

- ❖ There are various scripts available in bitcoin to handle the value transfer from the source to the destination
- ❖ These scripts range from very simple to quite complex depending upon the requirements of the transaction
- ❖ Types:
  - ❖ Pay to Public Key Hash (P2PKH)
  - ❖ Pay to Script Hash (P2SH)
  - ❖ MultiSig (Pay to MultiSig)
  - ❖ Pay to Pubkey
  - ❖ Null data/OP\_RETURN

# Transaction evaluation

- ❖ Standard transactions are evaluated using `IsStandard()` and `IsStandardTx()` tests
- ❖ Only standard transactions that pass the test are generally allowed to be mined or broadcasted on the bitcoin network
- ❖ However, nonstandard transactions are valid and allowed on the network
- ❖ Non-standard transactions—**those that fail the test—may be accepted by nodes not using the default Bitcoin Core settings**

# Pay to Public Key Hash (P2PKH)



- ❖ P2PKH is the most commonly used transaction type and is used to send transactions to the bitcoin addresses
- ❖ The format of the transaction is shown as follows:

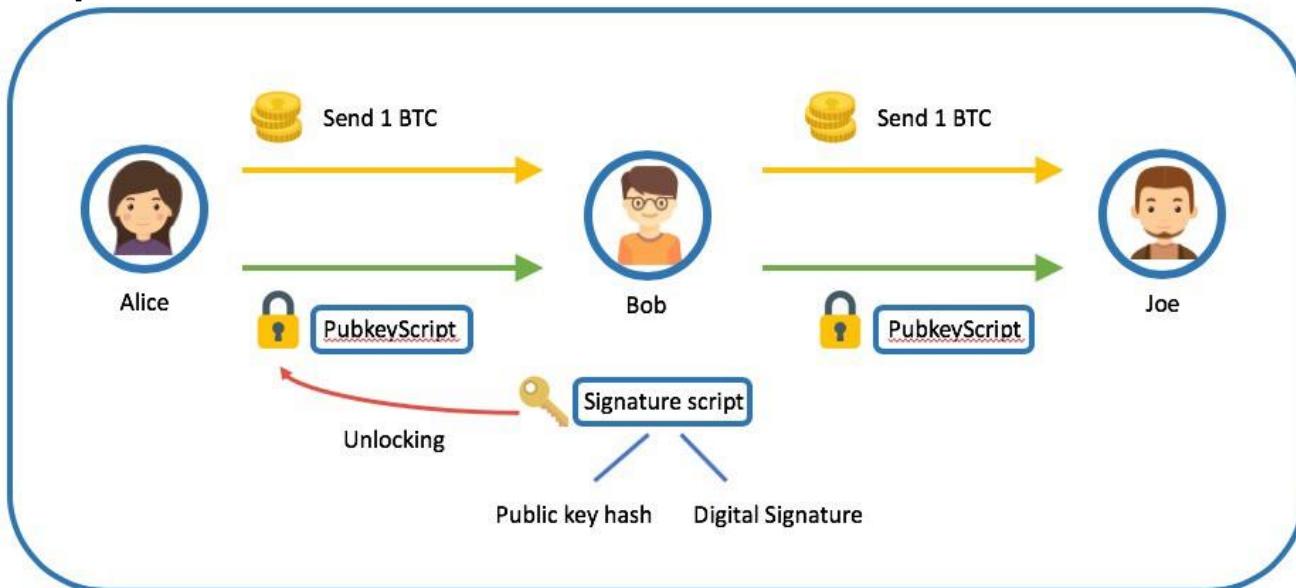
```
ScriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY  
OP_CHECKSIG

ScriptSig: <sig> <pubKey>
```

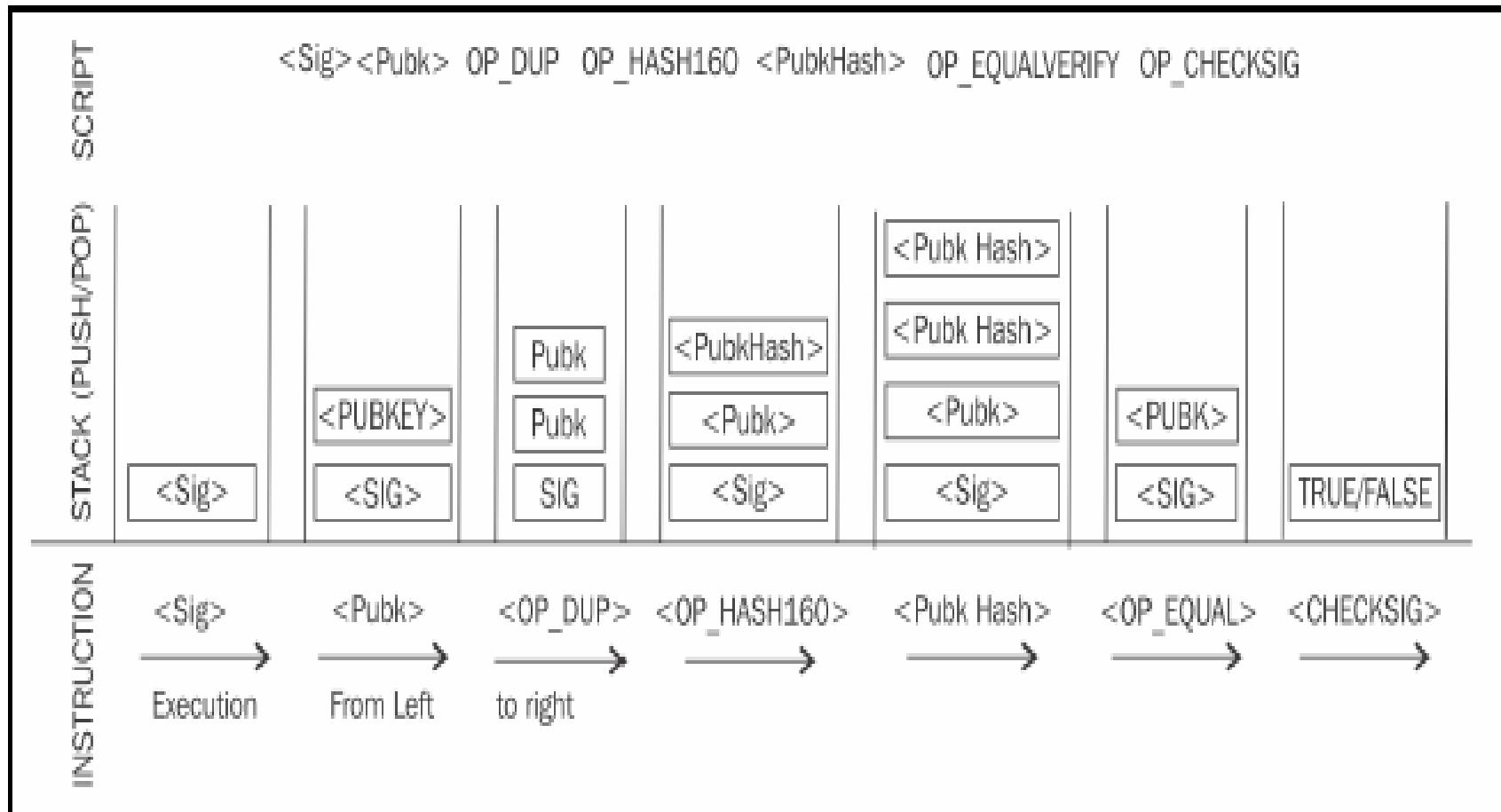
- ❖ The **ScriptPubKey** and **ScriptSig** parameters are concatenated together and executed

# Pay-to-Public-Key-Hash (P2PKH)

- ❖ It is a type of ScriptPubKey which locks bitcoin to the hash of a public key
- ❖ When Bob tries to spend the bitcoin he received, he must sign the transaction with the private key corresponding to the public key whose hash matches the hash provided in Alice's transaction



# P2PKH script execution



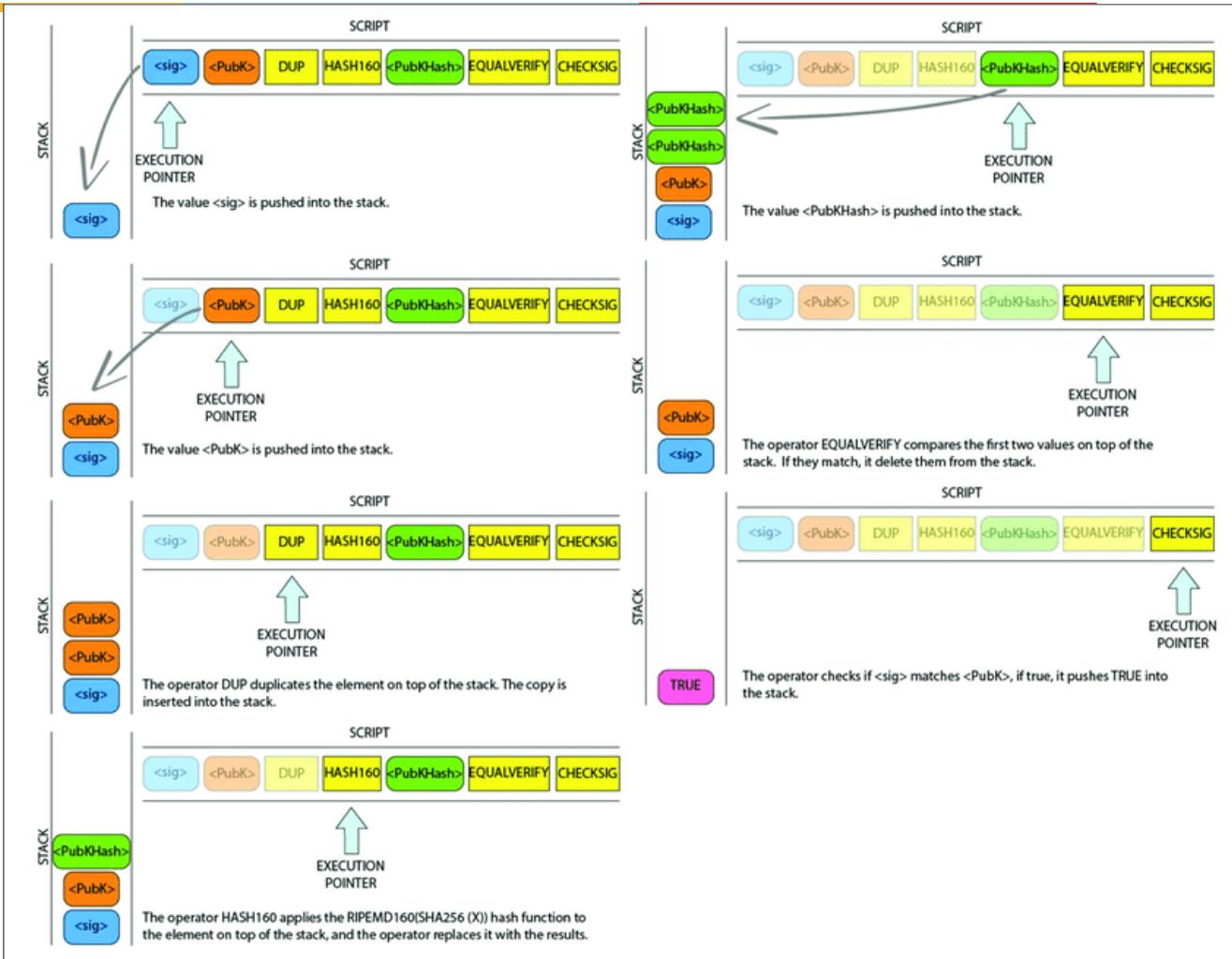
# scriptsig and scriptpubkey

---

<https://bitcoin.stackexchange.com/questions/8250/what-is-the-relation-between-scriptsig-and-scriptpubkey>

<https://genesisblockhk.com/what-is-p2pkh/>

# P2PKH script validation



# Pay-to-Script-Hash (P2SH)



- ❖ P2SH is used in order to send transactions to a script hash (that is, the addresses starting with 3) and was standardized in [BIP16](#)
- ❖ In addition to passing the script, the [redeem script](#) is also evaluated and must be valid
- ❖ The template is shown as follows:

```
ScriptPubKey: OP_HASH160 <redeemScriptHash> OP_EQUAL
```

```
ScriptSig: [<sig>...<sign>] <redeemScript>
```

What is [redeem script](#)? - A script **similar in function to a pubkey script**. One copy of it is hashed to create a P2SH address (used in an actual pubkey script) and another copy is placed in the spending signature script to enforce its conditions

BIP (Bitcoin Improvement Proposal)

# Pay-to-Script-Hash (P2SH)

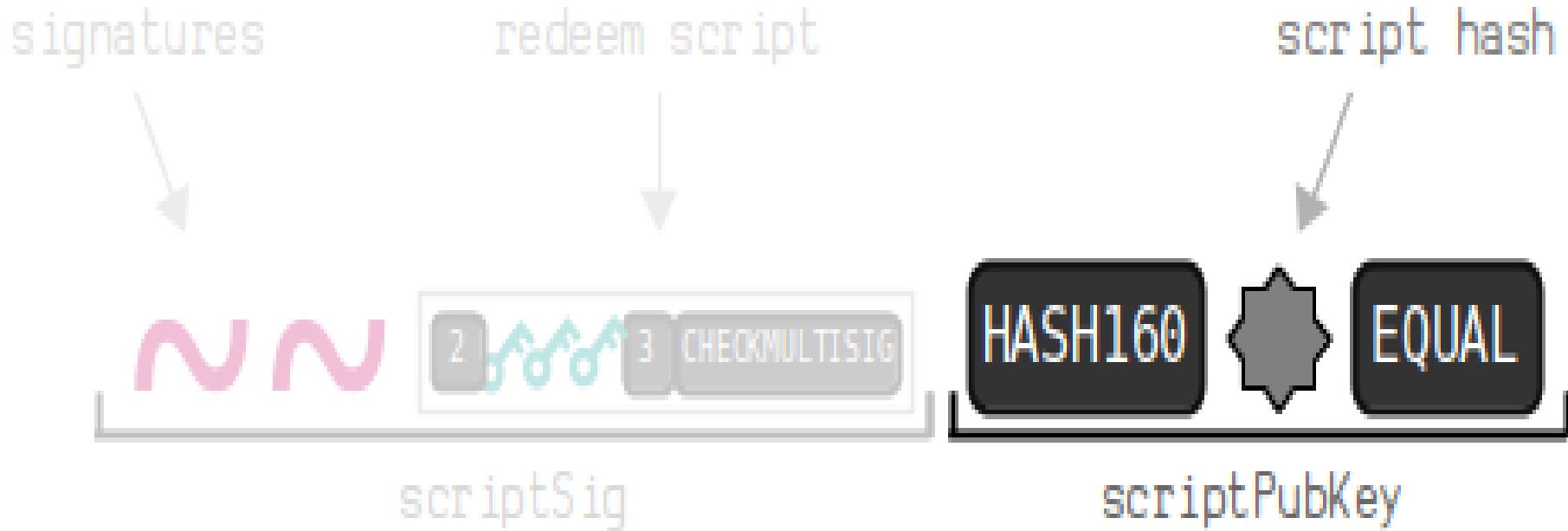
- ❖ Pay-to-Script-Hash (P2SH) is a type of ScriptPubKey which allows for the spending of bitcoin based on the satisfaction of the script whose hash is specified within the transaction
- ❖ A P2SH transaction is a transaction whose inputs were locked using a P2SH ScriptPubKey

**Scenario:** *if Alice sends Bob 1 BTC in a P2SH transaction, she includes the hash of the script required to spend the bitcoin in the transaction*

*This script can require signatures by Bob's private key and/or many other qualifications*

*When Bob wants to spend the bitcoin he has received from Alice, he reconstructs the script whose hash Alice used to send the bitcoin, and signs the transaction with any private key required by the script*

# Pay-to-Script-Hash (P2SH)



<https://bitcoin.stackexchange.com/questions/89323/how-to-build-a-p2sh-whos-scriptsig-contains-a-signature>

# How does P2SH work?



The locking script contains the hash of your custom locking script (the "script hash"), surrounded by the HASH160 and EQUAL opcodes

<b>scriptPubKey</b>	<b>OP_HASH160</b>	748284390f9e263a4b766a75d0633c50426eb875	<b>OP_EQUAL</b>	P2SH
---------------------	-------------------	--	-----------------	------

The unlocking script then contains your original custom locking script (the "redeem script"), preceded by the data/opcodes needed to unlock it:

<b>scriptSig</b>	<b>OP_0</b>	3046022100a07b2821f96658c938fa9c68950af0e69f3b2ce5f8258b3a6ad254d4bc73e11e022100e82fab8df3f
		7e7a28e91b3609f91e8ebf663af3a4dc2fd2abd954301a5da67e701
		5121022afc20bf379bc96a2f4e9e63ffceb8652b2b6a097f63fbe6ecec2a49a48010e2103a767c7221e9f15f87
		0f1ad9311f5ab937d79fcac15bb2c722bca515581b4c052ae

# P2SH advantages

---

- ❖ extremely flexible because it allows users to construct arbitrary scripts
- ❖ P2SH is used to enable backwards compatibility with new transaction types, including **SegWit**
- ❖ the sender of the transaction does not need to know what script type they are sending to
  - ❖ Ex: Bob can privately construct his desired script and only send Alice the hash of that script, preserving more privacy for Bob

# What Is SegWit (Segregated Witness)?

- ❑ SegWit is the process by which the block size limit on a blockchain is increased by removing signature data from bitcoin transactions
- ❑ When certain parts of a transaction are removed, this frees up space or capacity to add more transactions to the chain
- ❑ Segregate means to separate, and witnesses are the transaction signatures
- ❑ segregated witness, in short, means to separate transaction signatures

•SegWit is an action pertaining to Bitcoin that is designed to help increase the block size limit on a blockchain  
•SegWit helps increase the block size limit by pulling signature data from Bitcoin transactions  
•The term SegWit refers to segregate, or separate, and to witnesses, which are the transaction signatures

# MultiSig (Pay to MultiSig)

- ❖ M of n multi signature transaction script is a complex type of script where it is possible to construct a script that required multiple signatures to be valid in order to redeem a transaction
- ❖ Various complex transactions such as escrow and deposits can be built using this script
- ❖ The template is shown here:

```
ScriptPubKey: <m> <pubKey> [<pubKey> . . . ] <n> OP_CHECKMULTISIG
```

```
ScriptSig: 0 [<sig> . . . <sign>]
```

An escrow account is a third party account where funds are kept before they are transferred to the ultimate party

# Pay to Pubkey

---

- ❖ This script is a very simple script that is commonly used in coinbase transactions
- ❖ It is *now obsolete* and was used in an old version of bitcoin
- ❖ The public key is stored within the script in this case, and the unlocking script is required to sign the transaction with the private key
- ❖ The template is shown as follows:

```
<PubKey> OP_CHECKSIG
```

# Null data/OP\_RETURN

- ❖ This script is used to store arbitrary data on the blockchain for a fee
- ❖ The limit of the message is 40 bytes
- ❖ The output of this script is unredeemable because OP\_RETURN will fail the validation in any case
- ❖ ScriptSig is not required in this case
- ❖ The template is very simple and is shown as follows

```
OP_RETURN <data>
```



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

**BLOCKCHAIN TECHNOLOGY**  
**BITS F452**  
**1<sup>st</sup> Sem 2022-23**  
**Lecture 7**  
**Bitcoin Transactions...**



# Coinbase Transactions

- ❖ A **coinbase transaction** or **generation transaction** is always created by a miner and is the **first transaction** in a block
- ❖ Used to create new coins
- ❖ A special field, also called **coinbase** is included which acts as an input to the coinbase transaction
- ❖ This transaction also allows up to 100 bytes of arbitrary data that can be used to store arbitrary data

Ex:

- ❖ In the genesis block, this included the most famous comment taken from The Times newspaper

*"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks"*
- ❖ This message is proof that the genesis block was not mined earlier than January 3, 2009

# What is UTXO?

---

- ❖ **Unspent Transaction Output (UTXO)** is an unspent transaction output that can be spent as an input to a new transaction

# Transaction Fee

- ❖ Transaction fees are charged by the miners
- ❖ Fee charged is dependent upon the size of the transaction
- ❖ Transaction fees are calculated by subtracting the sum of the inputs and the sum of the outputs
- ❖ The fees are used as an incentive for miners to encourage them to include a user transaction in the block the miners are creating
- ❖ All transactions end up in the memory pool, from where miners pick up transactions based on their priority to include them in the proposed block
- ❖ There are different rules based on which fee is calculated for various types of actions:
  - ❖ sending transactions
  - ❖ inclusion in blocks
  - ❖ relaying by nodes
- ❖ Fees are not fixed by the Bitcoin protocol and are not mandatory
- ❖ Even a transaction with no fee will be processed in due course but may take a very long time.

# Contracts

<https://developer.bitcoin.org/devguide/>

<https://bloomberg.github.io/blpapi-docs/>

- ❖ contracts are basically transactions that use the bitcoin system to enforce a financial agreement and which allow users to design complex contracts that can be used in many real-world scenarios
- ❖ Contracts allow the development of a completely decentralized, independent, and reduced risk platform
- ❖ Contracts which can be built using the bitcoin scripting language include:
  - ❖ Escrow
  - ❖ Arbitration
  - ❖ Micropayment channels

Multisig  
Transaction locktime

# Transaction malleability

---

- ❖ Transaction malleability in bitcoin was introduced due to a **bug** in the bitcoin implementation
- ❖ Due to this bug, it becomes possible for an **adversary** to **change the Transaction ID of a transaction**, thus resulting in a scenario where it would appear that a certain transaction has not been executed
- ❖ Which means that this **bug allows the changing of the unique ID of a bitcoin transaction before it is confirmed**
- ❖ Allows scenarios where double deposits or withdrawals can occur

***Can you answer, why?***

---

# Transaction pools

---

- ❖ Also known as *memory pools*
  - ❖ These pools are basically created in local memory by nodes in order to maintain a temporary list of transactions that are not yet confirmed in a block
  - ❖ Transactions are included in a block after **passing verification** and based on their **priority**
-

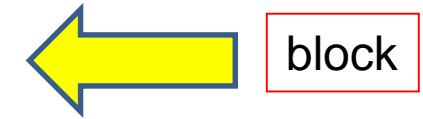
# Transaction verification process

1. Check the syntax and ensure that the syntax of the transaction is correct.
2. Verify that inputs and outputs are not empty.
3. Check whether the size in bytes is less than the maximum block size, which is 1 MB currently.
4. The output value must be in the allowed money range (0 to 21 million BTC).
5. All inputs must have a specified previous output, except for coinbase transactions, which should not be relayed.
6. Verify that nLockTime must not exceed 31-bits. For a transaction to be valid, it should not be less than 100 bytes. Also, the number of signature operands in a standard signature should be less than or not more than 2.
7. Reject *nonstandard* transactions; for example, ScriptSig is allowed to only push numbers on the stack. ScriptPubkey not passing the isStandard() checks.
8. A transaction is rejected if there is already a matching transaction in the pool or in a block in the main branch.
9. The transaction will be rejected if the referenced output for each input exists in any other transaction in the pool.

10. For each input, there must exist a referenced output transaction. This is searched in the main branch and the transaction pool to find whether the output transaction is missing for any input, and this will be considered an orphan transaction. It will be added to the orphan transactions pool if a matching transaction is not in the pool already.
11. For each input, if the referenced output transaction is the coinbase, it must have at least 100 confirmations; otherwise, the transaction will be rejected.
12. For each input, if the referenced output does not exist or has been spent already, the transaction will be rejected.
13. Using the referenced output transactions to get input values, verify that each input value, as well as the sum, is in the allowed range of 0-21 million BTC.
14. Reject the transaction if the sum of input values is less than the sum of output values.
15. Reject the transaction if the transaction fee would be too low to get into an empty block.

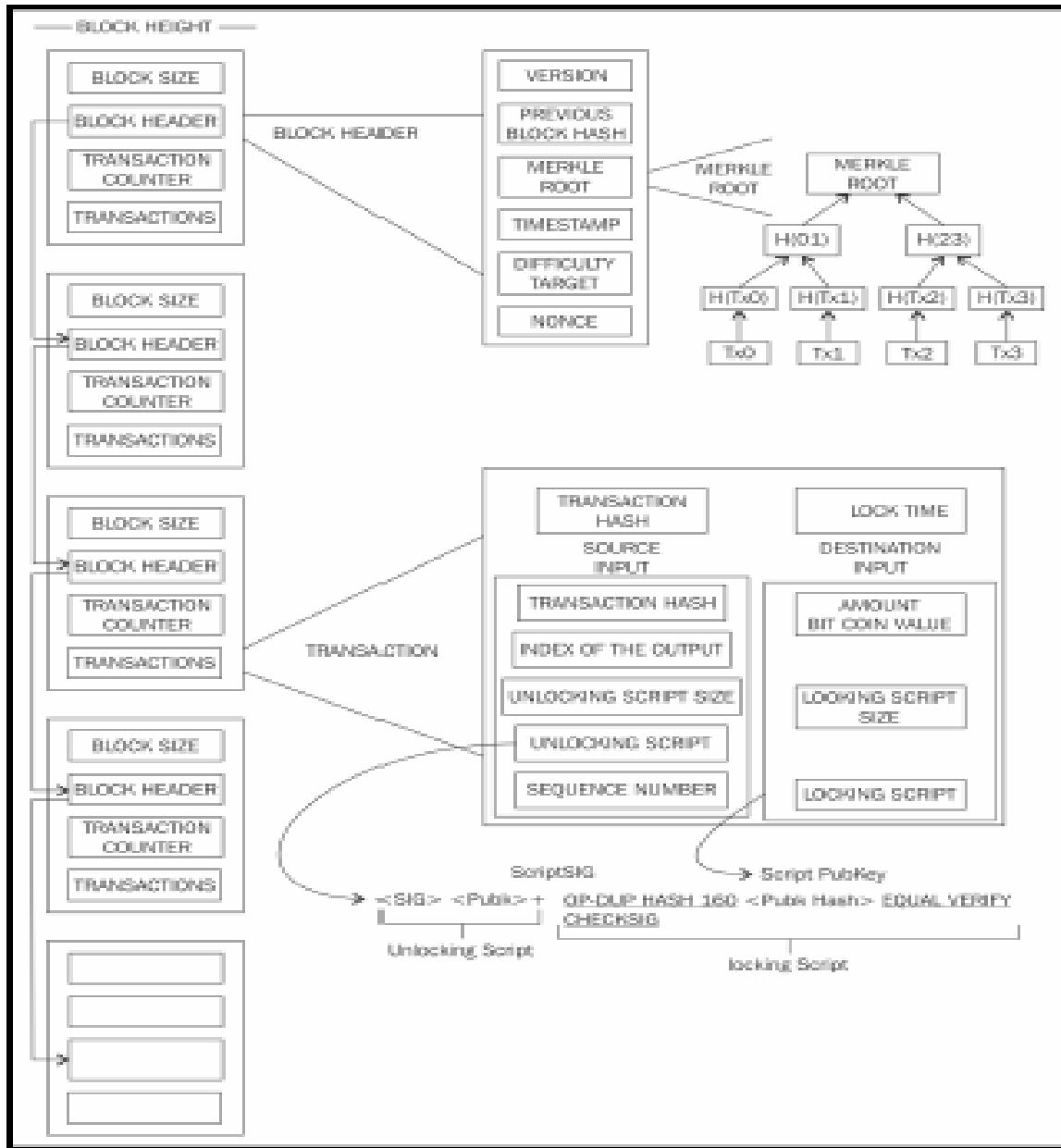
# More on the bitcoin blockchain

Bytes	Name	Description
80	Block header	This includes fields from the block header described in the next section.
variable	Transaction counter	The field contains the total number of transactions in the block, including the coinbase transaction.
variable	Transactions	All transactions in the block.



Bytes	Name	Description
4	Version	The block version number that dictates the block validation rules to follow.
32	previous block header hash	This is a double SHA256 hash of the previous block's header.
32	merkle root hash	This is a double SHA256 hash of the merkle tree of all transactions included in the block.
4	Timestamp	This field contains the approximate creation time of the block in the Unix epoch time format. More precisely, this is the time when the miner has started hashing the header (the time from the miner's point of view).
4	Difficulty target	This is the difficulty target of the block.
4	Nonce	This is an arbitrary number that miners change repeatedly in order to produce a hash that fulfills the difficulty target threshold.





- ❖ blockchain is a chain of blocks where each block is linked to its previous block by **referencing the previous block header's hash**
- ❖ This linking makes sure that no transaction can be modified unless the block that records it and all blocks that follow it are also modified
- ❖ The first block is not linked to any previous block and is known as the **genesis block**

# The genesis block

- ❖ This is the first block in the bitcoin blockchain
- ❖ The genesis block was hardcoded in the bitcoin core software

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

- ❖ Bitcoin provides protection against double spending by enforcing strict rules on transaction verification and via mining
- ❖ Blocks are added in the blockchain only after strict rule checking and successful Proof of Work solution
- ❖ Block height is the number of blocks before a particular block in the blockchain
- ❖ Proof of Work is used to secure the blockchain
- ❖ Each block contains one or more transactions, out of which the first transaction is a coinbase transaction
- ❖ There is a special condition for coinbase transactions that prevent them to be spent until at least 100 blocks in order to avoid a situation where the block may be declared stale later on

[https://developer.bitcoin.org/reference/block\\_chain.html](https://developer.bitcoin.org/reference/block_chain.html)

# Stale blocks & Orphan blocks

## ❖ Stale blocks:

- ❖ created when a block is solved and every other miner who is still working to find a solution to the hash puzzle is working on that block
- ❖ as the block is no longer required to be worked on, this is considered a stale block

## ❖ Orphan blocks:

- ❖ also called **detached blocks**, were accepted at one point in time by the network as valid blocks but were rejected when a proven longer chain was created that did not include this initially accepted block
- ❖ not part of the main chain and can occur at times when two miners manage to produce the blocks at the same time

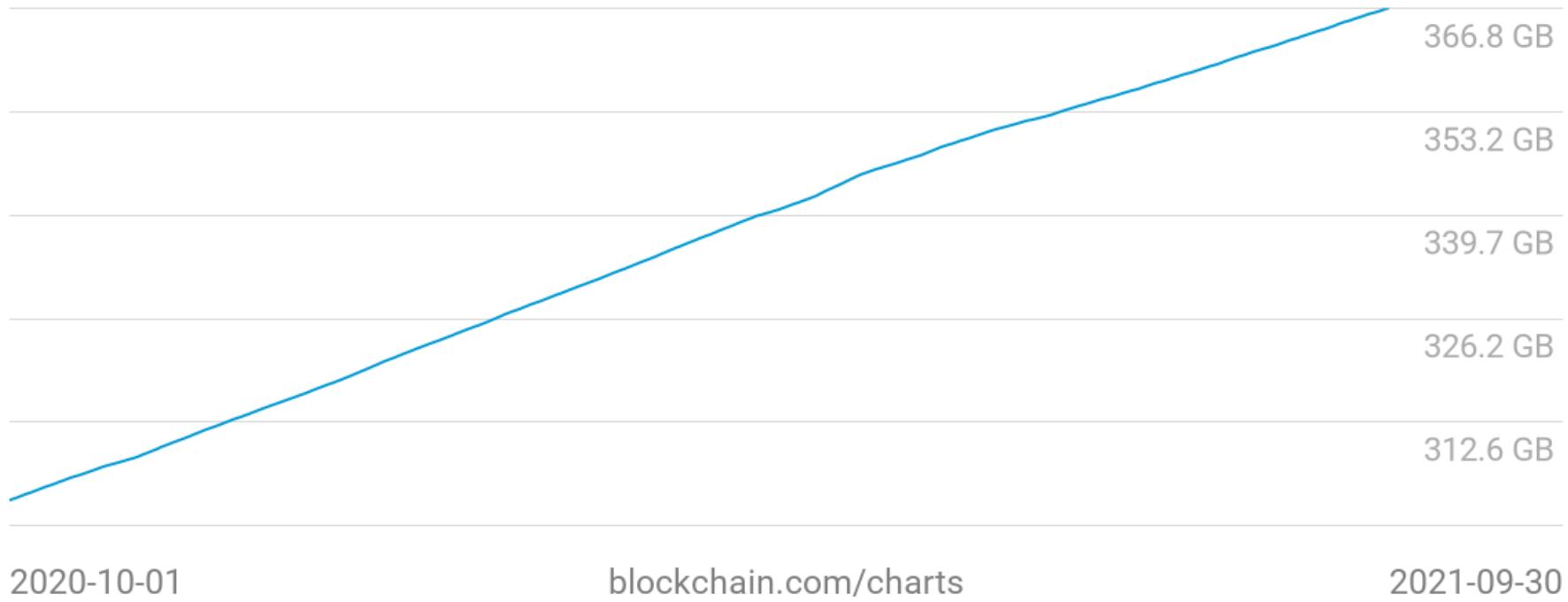
# Network forks in bitcoin

- ❖ Forks in blockchain can occur with the introduction of changes in the Bitcoin protocol (BIP)
- ❖ Soft forks
  - ❑ only previous valid blocks are no longer acceptable, thus making soft fork backward compatible
  - ❑ only miners are required to upgrade to the new client software in order to make use of the new protocol rules
  - ❑ planned upgrades do not necessarily create forks because all users should have updated already
- ❖ Hard forks
  - ❑ invalidates previously valid blocks and requires all users to upgrade

*New transaction types are sometimes added as a soft fork, and any changes such as block structure change or major protocol changes results in hard fork.*

# Current size of BTC blockchain

Blockchain Size  
366.8 GB



New blocks are added to the blockchain approximately every 10 minutes and network difficulty is adjusted dynamically every 2016 blocks in order to maintain a steady addition of new blocks to the network

# Calculating Network Difficulty

*Target = Previous target \* Time/2016 \* 10 minutes*

- ❖ Difficulty and target are interchangeable and represent the same thing
- ❖ Previous target represents the old target value, and time is the time spent to generate previous 2016 blocks
- ❖ Network difficulty basically means how hard it is for miners to find a new block
- ❖ This means how difficult the hashing puzzle is now

*Solving the hashing puzzle.....?*

# Mining

- ❖ Mining is a resource-intensive process by which new blocks are added to the blockchain
- ❖ Blocks contain transactions that are validated via the mining process by mining nodes and are added to the blockchain
- ❖ This process is resource-intensive in order *to ensure that the required resources have been spent by miners in order for a block to be accepted*
- ❖ New coins are minted by the miners by spending the required computing resources
- ❖ This also secures the system against frauds and double spending attacks while adding more virtual currency to the bitcoin ecosystem

# More on mining

---

- ❖ Roughly one new block is created (mined) every 10 minute
- ❖ Miners are rewarded with new coins if and when they create new blocks and are paid transaction fees in return of including transactions in their blocks
- ❖ New blocks are created at an approximate fixed rate
- ❖ Also, the rate of creation of new bitcoins decreases by 50%, every 210,000 blocks, roughly every 4 years
- ❖ Approximately 144 blocks, that is, 1,728 bitcoins are generated per day
- ❖ Bitcoin supply is also limited and in 2140, almost 21 million bitcoins will be finally created and no new bitcoins can be created after that

The current block reward is **6.25 BTC per block**. The next halving will occur sometime in spring of 2024, when the block reward will be reduced to 3.125

# Miners' tasks

## Synching up with the network

- Once a new node joins the bitcoin network, it downloads the blockchain by requesting historical blocks from other nodes*
- ✓ **Transaction validation:** Transactions broadcasted on the network are validated by full nodes by verifying and validating signatures and outputs
- ✓ **Block validation:** Miners and full nodes can start validating blocks received by them by evaluating them against certain rules
- ✓ **Create a new block:** Miners propose a new block by combining transactions broadcasted on the network after validating them
- ✓ **Perform Proof of Work:** This task is the core of the mining process and this is where miners find a valid block by solving a computational puzzle; the block header contains a 32-bit nonce field and miners are required to repeatedly vary the nonce until the resultant hash is less than a predetermined target
- ✓ **Fetch reward:** Once a node solves the hash puzzle, it immediately broadcasts the results, and other nodes verify it and accept the block

Is there a chance that the newly minted block will not be accepted by other miners? If so, why?

# Proof of Work

- ❖ This is a proof that enough computational resources have been spent in order to build a valid block
- ❖ **PoW** is based on the idea that a random node is selected every time to create a new block
- ❖ In this model, nodes compete with each other in order to be selected in proportion to their computing capacity
- ❖ PoW requirement in bitcoin eco system:

$$H(N \parallel P\_hash \parallel Tx \parallel Tx \parallel \dots \parallel Tx) < Target$$

- ❖ Where  $N$  is a nonce,  $P\_hash$  is a hash of the previous block,  $Tx$  represents transactions in the block, and  $Target$  is the target network difficulty value
- ❖ The hash of the previously mentioned concatenated fields should be less than the target hash value; only way to find this nonce is the brute force method
- ❖ Once a certain pattern of a certain number of zeroes is met by a miner, the block is immediately broadcasted and accepted by other miners

# Steps in mining algorithm

1. The previous hash block is retrieved from the bitcoin network
2. Assemble a set of potential transactions broadcasted on the network into a block
3. Compute the double hash of the block header with a nonce and the previous hash using the SHA256 algorithm
4. If the resultant hash is lower than the current difficulty level (target), then stop the process
5. If the resultant hash is greater than the current difficulty level (target), then repeat the process by incrementing the nonce
6. Mining difficulty increased over time and bitcoins that could be mined by single CPU laptop computers, now require dedicated mining centers to solve the hash puzzle

# Current difficulty level

---

- ❖ The current difficulty level can be queried using the bitcoin command line interface using the following command **\$ *bitcoin-cli getdifficulty***
- ❖ <https://www.blockchain.com/charts/difficulty>
- ❖ <https://www.coinwarz.com/mining/bitcoin/difficulty-chart>

# Hashing rate

---

- ❖ represents the rate of calculating hashes per second
- ❖ Mining pool, ASIC, increased difficulty
- ❖ Hashing rate now measured in **exa hashes** (1 000 000 000 000 000 000 hashes per second)
- ❖ <https://bitinfocharts.com/comparison/bitcoin-hashrate.html#3y>

# Mining Systems

- 
1. CPU mining
  2. GPU mining
  3. FPGA mining
  4. ASIC mining *(most profitable mining h/w as on today)*
-

# Mining pools

---

- A **mining pool** forms when group miners work together to mine a block
- The *Pool manager* receives the **coinbase transaction** if the block is successfully mined, which is then responsible for distributing the reward to the group of miners who invested resources to mine the block
- This is profitable as compared to **solo mining** *[where only one sole miner is trying to solve the partial hash inversion function (hash puzzle) because in mining pools, the reward is paid to each member of the pool regardless of whether they (more specifically, their individual node) solved the puzzle or not]*

# Mining pool - models

---

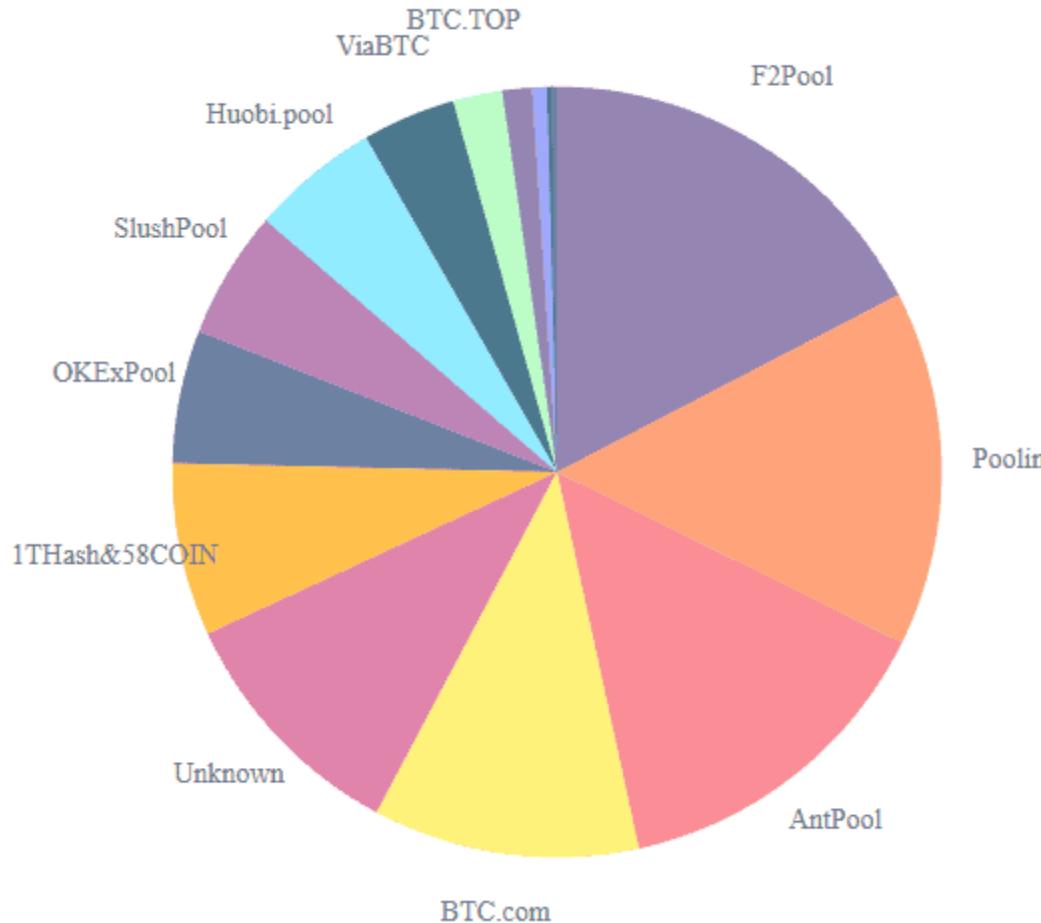
## 1. pay-per-share model

The mining pool manager pays a **flat fee** to all miners who participated in the mining exercise

## 2. proportional model

the share is calculated based on the **amount of computing resources** spent to solve the hash puzzle

# Mining Service Providers via cloud and web



*(as on 2021)*

# Mining centralization

- ❖ Mining centralization is a major concern that can occur if a pool manages to control more than 51% of the network by generating more than 51% hash rate of the bitcoin network
- ❖ *Which attack it can lead to? => this attack further leads to another one?*

# Bitcoin Network

- ❖ P2P network where nodes exchange transactions and blocks
- ❖ Two types of nodes:
  - ❖ Full nodes
  - ❖ SPV (Simplified Payment Verification ) nodes or lightweight nodes
  - ❖ full blockchain nodes (no mining etc..)
  - ❖ Solo miner nodes
  - ❖ Pool protocol servers

# Full nodes

- ❑ implementations of bitcoin core clients performing the
  - ❑ Wallet
  - ❑ Miner
  - ❑ full blockchain storage
  - ❑ network routing functions
- ❑ it is not necessary to perform all these functions

# SPV nodes

---

- ❖ SPV nodes or lightweight clients perform only wallet and network routing functionality
-

# full blockchain nodes - no mining

- 
- ❑ contain complete blockchain and perform network routing functions but do not perform mining or store private keys (the wallet function)

# Solo miner nodes

- 
- ❖ They can perform mining, store full blockchain, and act as a bitcoin network routing node

# Pool protocol servers



- ❖ heavily used nodes
- ❖ make use of alternative protocols, such as the stratum protocol
- ❖ some nodes perform only **mining functions** and are called **mining nodes**
- ❖ Nodes that only compute hashes use the stratum protocol to submit their solutions to the mining pool
- ✓ Stratum is a line-based protocol that makes use of plain TCP sockets and human-readable JSON-RPC to operate and communicate between nodes

**Stratum, the de-facto mining communication protocol used by blockchain based cryptocurrency systems**, enables miners to reliably and efficiently fetch jobs from mining pool servers

<https://briiins.com/stratum-v2>

# Magic values in Bitcoin networks



Magic values are used to indicate the message origin network

## Magic number in programming:

- Unique values with unexplained meaning or multiple occurrences which could (preferably) be replaced with named constants
- A constant numerical or text value used to identify a file format or protocol
- Distinctive unique values that are unlikely to be mistaken for other meanings

Size	Field	Details
4 bytes	Magic no.	Value is always 0xD9B4BEF9
4 bytes	Blocksize	number of bytes following up to end of block
80 bytes	Blockheader	consists of 6 items
1 - 9 bytes	Transaction counter	positive integer VI = VarInt
-	transactions	list of transactions

<https://bitcoin.stackexchange.com/questions/2337/how-was-the-magic-network-id-value-chosen>

<https://bitcoin.stackexchange.com/questions/43189/what-is-the-magic-number-used-in-the-block-structure?noredirect=1&lq=1>

# Blockchain wallets

---

- A blockchain wallet is a **digital wallet that allows users to store and manage their Bitcoin, Ether, and other cryptocurrencies**
  - Blockchain wallets follow a similar process as email system **using a public key and a private key together**
  - A public key is similar to your email address; you can give it to anyone
  - When your wallet is generated, a public key is generated, and you can share the public key with anyone in order to receive funds
-

# Best Bitcoin Wallets of 2021

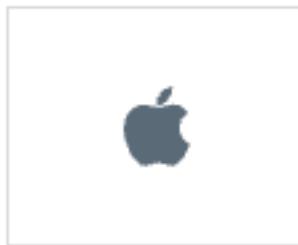
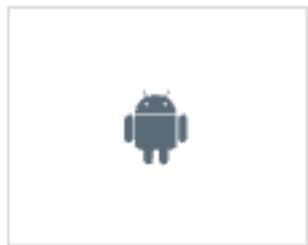
- 
- #1. **Ledger Nano X**: Overall Best Hardware Wallet.
  - #2. **Trezor Model T**: Top Bitcoin Wallet Company for Wallet Purchase.
  - #3. Ledger Nano S: Best to Buy Bitcoin.
  - #4. Exodus: Best for Managing Bitcoin & Other Cryptocurrencies.
  - #5. Mycelium: Easily Store Bitcoin in Offline Device.
-

# Wallets

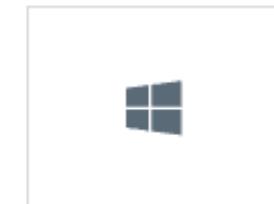
- ❖ A Bitcoin wallet is a **device or program for holding and sending Bitcoins**
- ❖ Bitcoin wallets contain the private keys needed to sign Bitcoin transactions
- ❖ Anyone who knows the private key can control the coins associated with that address
- ❖ The most secure Bitcoin wallets are hardware wallets
- ❖ The **wallet** software is used to store private or public keys and bitcoin address
- ❖ Performs various functions, such as receiving and sending bitcoins
- ❖ [https://en.bitcoin.it/wiki/Wallet#Bitcoin\\_Wallet](https://en.bitcoin.it/wiki/Wallet#Bitcoin_Wallet)

# Bitcoin wallets

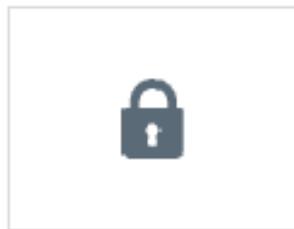
## Mobile wallets



## Desktop wallets



## Hardware wallets



# More on bitcoin wallets

---

- ❖ Private keys can be generated in different ways and are used by different types of wallets
  - ❖ **Wallets do not store any coins**, and there is no concept of wallets storing balance or coins for a user
  - ❖ In fact, in the bitcoin network, **coins do not exist**
  - ❖ instead, only transaction information is stored on the blockchain
  - ❖ UTXO - unspent outputs are then used to calculate the amount of bitcoins
-

# Types of wallets in bitcoin networks

---

**Non-deterministic wallets**

**Deterministic wallets**

**Hierarchical deterministic wallets**

**Brain wallets**

**Paper wallets**

**Hardware wallets**

**Online wallets**

**Mobile wallets**

# Non-deterministic wallets



- ❖ These wallets contain randomly generated private keys and are also called *Just a Bunch of Key wallets*
- ❖ The bitcoin core client generates some keys when first started and generates keys as and when required
- ❖ Managing a **large number of keys** is very difficult and an error-prone process can lead to theft and loss of coins
- ❖ Need to create regular **backups of the keys** and protect them appropriately in order to prevent theft or loss

# Deterministic wallets



- ❖ Keys are derived out of a **seed value** via **hash functions**
- ❖ This seed number is generated randomly and is commonly represented by human-readable ***mnemonic code words***
- ❖ **Mnemonic code words** are defined in BIP39
- ❖ used to recover all keys and make private key management comparatively easier

# Hierarchical Deterministic wallets



- ❖ HD wallets store keys in a tree structure derived from a seed
- ❖ The seed generates the parent key (master key), which is used to generate child keys, subsequently, grandchild keys
- ❖ Key generation in HD wallets does not generate keys directly
- ❖ instead, it produces some information (private key generation information) that can be used to generate a sequence of private keys
- ❖ The complete hierarchy of private keys in an HD wallet is easily recoverable if the master private key is known
- ❖ It is because of this property that HD wallets are very easy to maintain and are highly portable

# Brain wallets



- ❖ The master private key can also be derived from the hash of passwords that are memorized
- ❖ The key idea is that this passphrase is used to derive the private key and if used in HD wallets, this can result in a full HD wallet that is derived from a single memorized password
- ❖ This is known as **brain wallet**
- ❖ prone to password guessing and brute force attacks
- ❖ ***key stretching*** can be used to slow down the progress made by the attacker



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

**BLOCKCHAIN TECHNOLOGY**  
**BITS F452**  
**1<sup>st</sup> Sem 2022-23**  
**Lecture 8**  
**Smart Contracts**

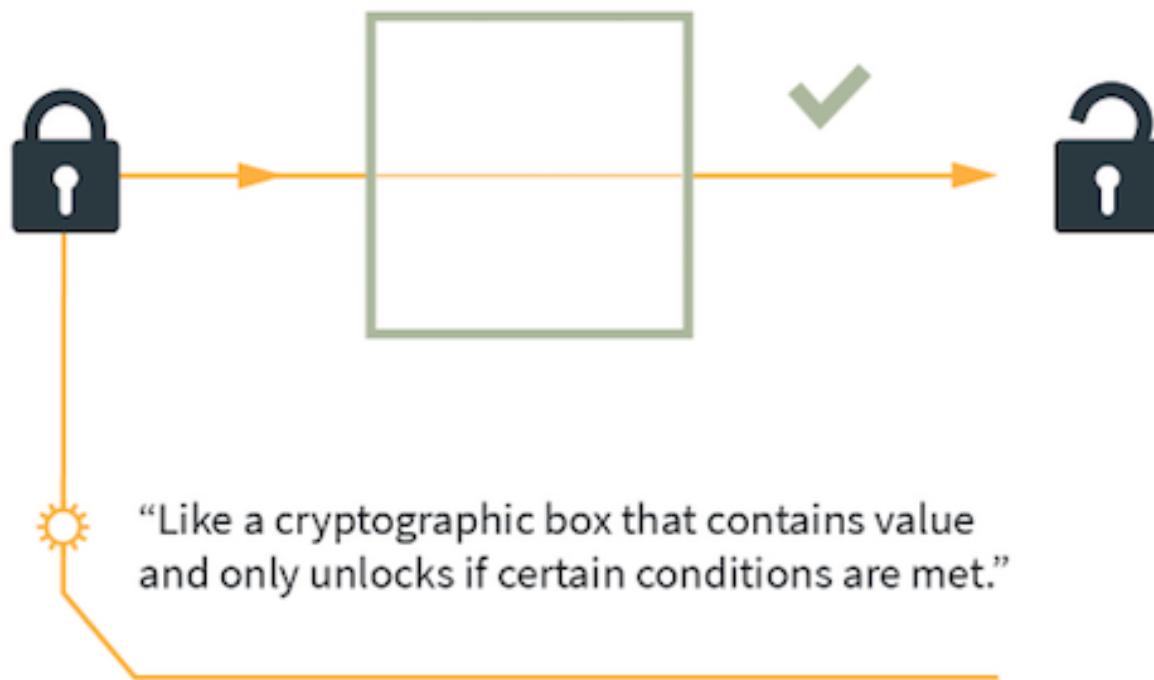


# Smart contracts blockchain

- ❖ Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met
- ❖ Used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss
- ❖ Can also automate a workflow, triggering the next action when conditions are met

Smart contracts are digital contracts stored on a blockchain that are automatically executed when predetermined terms and conditions are met

# *Smart Contracts*



From the Book “**Token Economy**” by Shermin Voshmgir, 2019  
Excerpts available on <https://blockchainhub.net>

## *Aspects of Smart Contracts*

<i>Technical Aspects</i>	<i>Legal Aspects</i>	<i>Economic Aspects</i>
Self-verifying (Auditing on the fly)	Smart contracts can map legal obligations into an automated process.	Higher transparency
Self-executing (Enforcement on the fly)	If implemented correctly, they can provide greater degree of contractual security	Less intermediaries
Tamper resistant (No cheating)		Lower transaction costs

From the Book "**Token Economy**" by Shermin Voshmgir, 2019  
Excerpts available on <https://blockchainhub.net>

# Smart Contracts

- ❑ *A smart contract is a computerized transaction protocol that executes the terms of a contract*
- ❑ *The general objectives are::*
  - ❑ *to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement),*
  - ❑ *minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries*
- ❑ *Economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs*

# How smart contracts work?



- Smart contracts work by following simple “if/when...then...” statements that are written into a code on a blockchain
- A network of computers executes the actions when predetermined conditions have been met and verified
- Ex: releasing funds to the appropriate parties, registering a vehicle, sending notifications, or issuing a ticket
- The blockchain is then updated when the transaction is completed
- That means the transaction cannot be changed, and only parties who have been granted permission can see the results

# How smart contracts work? – contd.



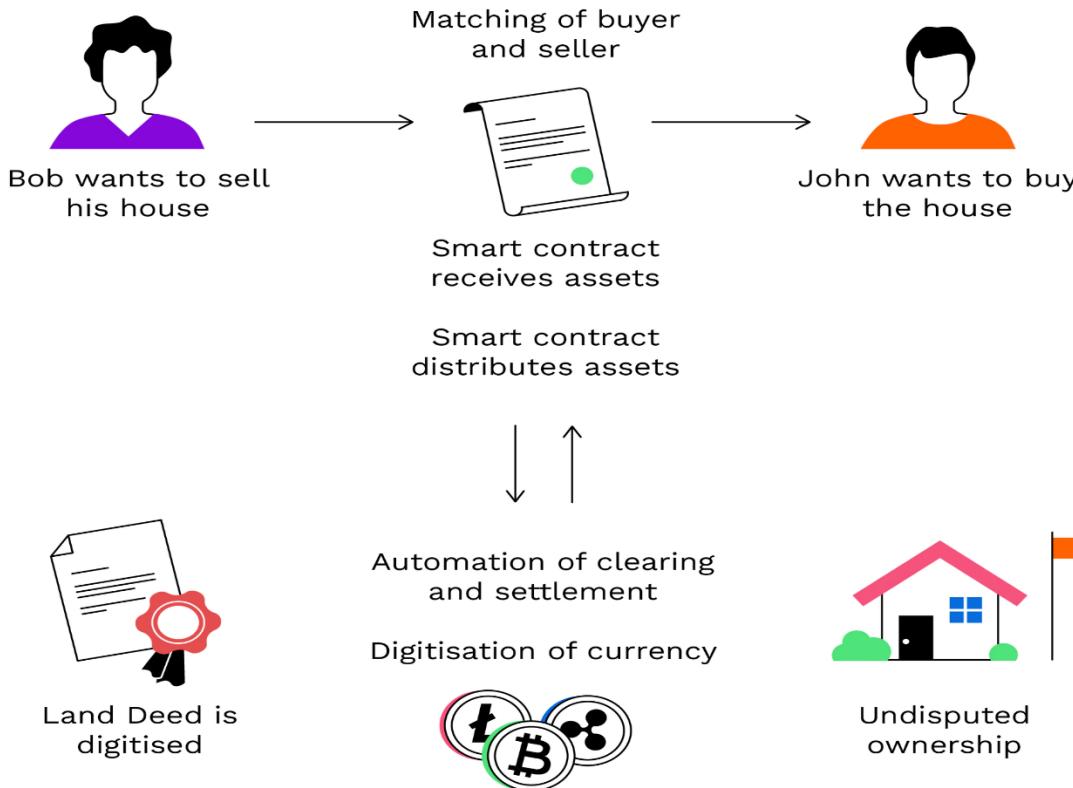
- ❖ Within a smart contract, there can be as many **stipulations** as needed to satisfy the participants that the task will be completed satisfactorily
- ❖ To establish the terms, participants must determine how transactions and their data are represented on the blockchain, agree on the “if/when...then...” rules that govern those transactions, explore all possible exceptions, and define a framework for resolving disputes

<https://blockchainhub.net/smart-contracts/>

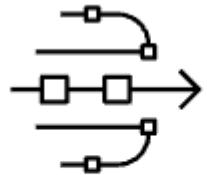
# How smart contracts work? contd..



## How a smart contract works



# Benefits of smart contracts



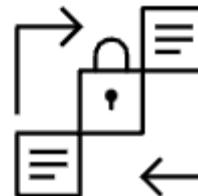
Speed, efficiency and accuracy



Trust and transparency



Security



Savings

[https://blockgeeks.com/guides/smart-contracts/#A\\_Smart\\_Contract\\_Example](https://blockgeeks.com/guides/smart-contracts/#A_Smart_Contract_Example)

A

*smart contract is a  
secure and unstoppable computer program  
representing an agreement that is  
automatically executable and  
enforceable*

# Four properties of smart contracts

- 1. Automatically executable
  - 2. Enforceable
  - 3. Semantically sound
  - 4. Secure and unstoppable
- } Minimum requirement
- } can be relaxed

# Ricardian contracts

---

- ✓ Proposed by Ian Grigg in 90's
  - ✓ used initially in a bond trading and payment system called **Ricardo**
  - ✓ The key idea is to write a document which is understandable and acceptable by both a court of law and computer software
  - ✓ It identifies the issuer and captures all the terms and clauses of the contract in a document in order to make it acceptable as a legally binding contract
-

# Ricardian contracts

## Contd..

---



- ❖ Ricardian contracts address the challenge of issuance of value over the Internet
- ❖ It identifies the issuer and captures all the terms and clauses of the contract in a document in order to make it acceptable as a **legally binding contract**

original definition by *Ian Grigg* at:

[http://iang.org/papers/ricardian\\_contract.html](http://iang.org/papers/ricardian_contract.html)

# Ricardian contracts

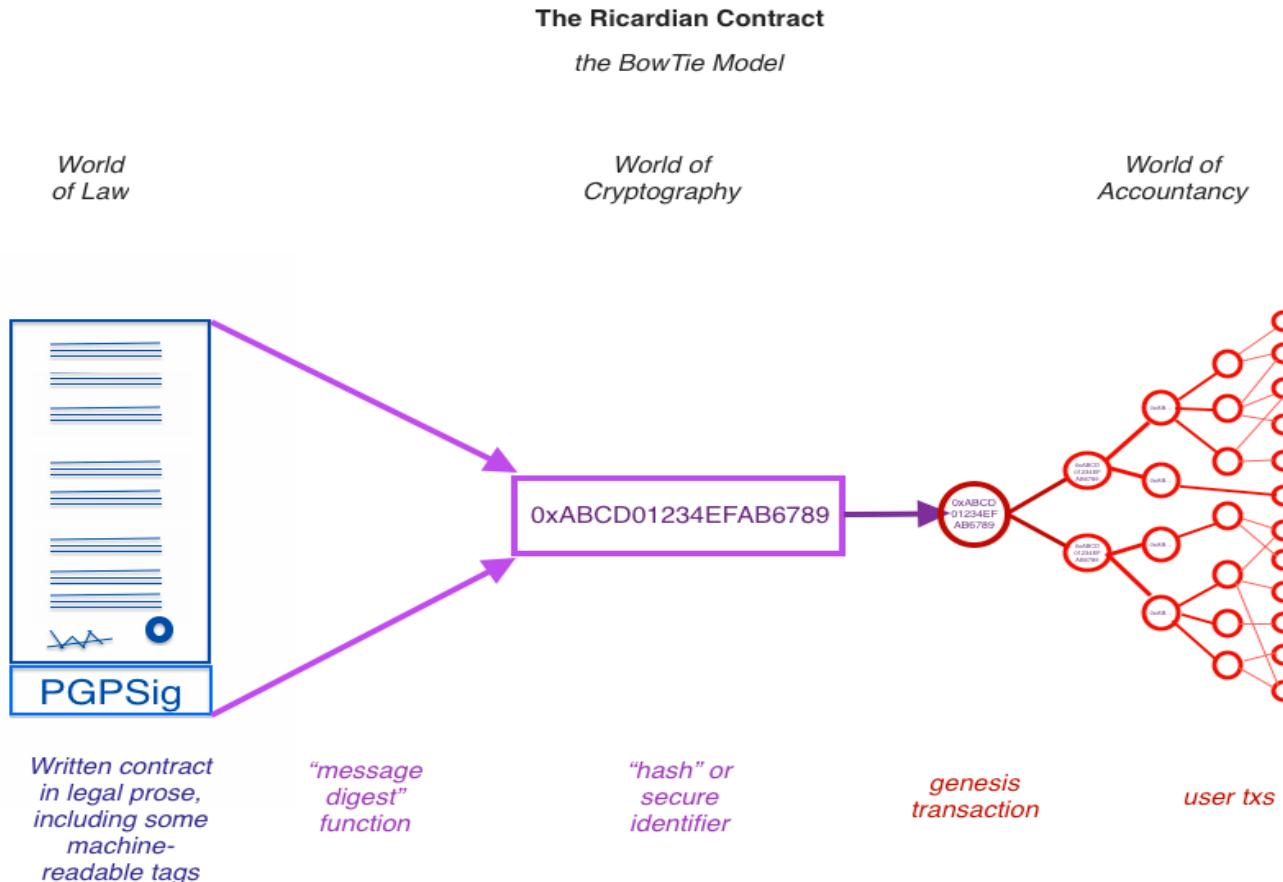


## *Properties:*

---

1. A contract offered by an issuer to holders
  2. A valuable right held by holders, and managed by the issuer
  3. Easily readable by people (like a contract on paper)
  4. Readable by programs (parseable, like a database)
  5. Digitally signed
  6. Carries the keys and server information
  7. Allied with a unique and secure identifier
-

# BowTie model



# Ricardian contracts:- implementation using bowtie model

---

- the contracts are implemented by producing a single document that contains the terms of the contract in legal prose and the required machine-readable tags
  - This document is digitally signed by the issuer using their private key
  - This document is then hashed using a message digest function to produce a hash by which the document can be identified
  - This hash is then further used and signed by parties during the performance of the contract in order to link each transaction, with the identifier hash thus serving as evidence of intent.
-

# World of Law and World of Accountancy

---

- ❖ **World of Law =>** from where the document originates
- ❖ It is then hashed and the resultant message digest is used as an identifier throughout the World of Accountancy
- ❖ **World of Accountancy =>** represents any or multiple accounting, trading and information systems that are being used in a business to perform various business operations
- ❖ message digest generated by hashing the document is first used in a so called *genesis transaction*, and then used in every transaction as an identifier throughout the operational execution of the contract
- ❖ **Secure link** is created between the **original written contract** and every transaction in the World of Accounting

# Differences between Ricardian & smart contracts



## Ricardian contract

- concerned with the **semantic richness** and production of a document that contains contractual legal prose

## Smart contract

- does not include any contractual document and is focused purely on the **execution** of the contract

### ✓ Semantics of a contract

- operational semantics
- Denotational semantics
- Operational => defines the actual execution, correctness and safety of the contract
- Denotational => concerned with the real-world meaning of the full contract

*Ex: in the world of smart contracts*

# *intersection of Ricardian and smart contracts*



# Bitcoin example:

- ❖ Smart contract is fully oriented towards the execution of the contract
- ❖ Ricardian contract is more geared towards producing a document that is understandable by humans, with some parts that a computer program can understand
- ❖ This can be viewed as legal semantics vs operational performance (semantics vs performance)
- A smart contract is made up to have both of these elements (performance and semantics) embedded together, which completes an ideal model of a smart contract

# Ricardian contracts

---

- ❑ A Ricardian contract can be represented as a tuple of three objects, namely Prose, parameters and code
  - ❑ Prose represents the legal contract in regular language
  - ❑ code represents the program that is a computer-understandable representation of legal prose
  - ❑ parameters join the appropriate parts of the legal contract to the equivalent code
- ❑ Ricardian contracts have been implemented in many systems, such as CommonAccord, OpenBazaar, OpenAssets, and Askemos

# Smart contract templates

<https://arxiv.org/abs/1608.00771>

- Automation
- Enforceability
- Semantics

<https://github.com/topics/smart-contract-template>

standard templates that provide a framework to support legal agreements for financial instruments

domain-specific language DSLs are already in use in the financial industry to provide specific language for a specific domain

# Oracles

- ❖ An Oracle is an **interface** that delivers data from an external source to smart contracts; important component of the smart contract ecosystem
- ❖ **Smart contracts cannot access external data** which might be required to control the execution of the business logic
  - ❖ the stock price of a security that is required by the contract to release the dividend payments
- ❖ Oracles can be used **to provide external data to smart contracts**
- ❖ Oracles can deliver **different types of data** ranging from weather reports, real-world news, and corporate actions to data coming from Internet of Things (IoT) devices
- ❖ Oracles are trusted entities that use a **secure channel** to transfer data to a smart contract

# Oracles contd..

---

- ❖ Oracles are also capable of **digitally signing** the data proving that the source of the data is **authentic**
  - ❖ Smart contracts can then subscribe to the Oracles, and the smart contracts can either **pull** the data, or Oracles can **push** the data to the smart contracts
  - ❖ Oracles should **not** be able to **manipulate** the data they provide and must be able to provide authentic data
  - ❖ Even though Oracles are trusted, it may still be possible in some cases that the data is incorrect due to manipulation
-

# Types of Oracles

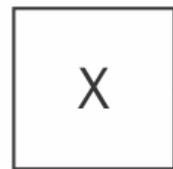
---

- ❖ Standard (simple) oracles – trusted third party – centralized
  - ❖ Decentralized oracles – distributed – blockchain – distributed consensus
  - ❖ Hardware oracles – telemetry – IoT
-

# Oracle interacting with smart contract on blockchain



DATA FEED 1



MARKET DATA



WEATHER DATA



ORACLE



BLOCK  
CHAIN

SECURE CHANNEL

How to write data? in bitcoin blockchain, an oracle can write data to a specific transaction via an OP\_RETURN Opcode, and a smart contract can monitor that transaction and read the data

# Smart oracles

---

- ❖ Smart Oracle proposed and implemented in *Codius*
- ❖ Smart Oracles are basically entities just like Oracles, but with the added capability of contract code execution
- ❖ Smart Oracles proposed by Codius run using Google Native Client which is a sandboxed environment for running untrusted x86 native code

<https://codius.org>

# Deploying smart contracts on a blockchain



- ❖ Ethereum is an example of a blockchain that natively supports the development and deployment of smart contracts
- ❖ Smart contracts on Ethereum blockchain are usually part of a larger application such as **Decentralized Autonomous organization (DAOs)**
- ❖ bitcoin blockchain: the lock\_time field in the bitcoin transaction can be seen as an enabler of a basic version of a smart contract
  - ❖ *The lock\_time field enables a transaction to be locked until a specified time or after a number of blocks, thus enforcing a basic contract that a certain transaction can only be unlocked if certain conditions (elapsed time or number of blocks) is met*
- ❖ The DAO is one of the highest crowdfunded projects



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

**BLOCKCHAIN TECHNOLOGY**  
**BITS F452**  
**1<sup>st</sup> Sem 2022-23**  
**Lecture 9**  
**Alternate Coins**  
**&**  
**Alternatives to PoW**



# Alternative Coins

- ❖ **Altcoins** must be able to attract new users, trades, and miners otherwise the currency will have no value
- ❖ Currency gains its value, especially in the virtual currency space, due to the network effect and its acceptability by the community
- ❖ Users can be attracted by providing an initial amount of coins and can be achieved by using various methods

# Methods of providing an initial number of altcoins

---

- ❖ **Create a new blockchain**
  - ❖ **Proof of burn**
  - ❖ **Proof of Ownership**
  - ❖ **Pegged sidechains**
-

# Create a new blockchain



- ❑ Altcoins can create a new blockchain and allocate coins to initial miners
- ❑ now unpopular due to many scam schemes (*pump and dump*) schemes where initial miners made a profit with the launch of a new currency and then disappeared

# Proof of burn



one-way peg or price ceiling

---

- ❖ Users permanently destroy a certain quantity of bitcoins in proportion to the quantity of altcoins to be claimed
- ❖ For example if 10 bitcoins were destroyed then altcoins can have a value no greater than the amount of bitcoins destroyed
- ❖ bitcoins are being *converted* into altcoins by burning them

# Proof of Ownership



- ❖ Instead of permanently destroying bitcoins, an alternative method is to prove that **users own** a certain number of bitcoins
- ❖ This **proof of ownership** can be used to claim altcoins by tethering altcoin blocks to bitcoin blocks
- ❖ For example, this can be achieved by merged mining in which effectively bitcoin miners can mine altcoin blocks while mining for bitcoin without any extra work

# Pegged sidechains



- ❖ **Sidechains**, are blockchains separate from the bitcoin network but bitcoins can be transferred to them
- ❖ Altcoins can also be transferred back to the bitcoin network
- ❖ This concept is called a **two-way peg**

# Alternatives to PoW

---

- ❖ PoW provides decentralization, security and stability for the blockchain
  - ❖ main vehicle in bitcoin for providing decentralized distributed consensus
  - ❖ PoW should have **progress freeness**, which basically means that the reward for consuming computational resources should be random and proportional to the contribution made by the miners
  - ❖ some chance of winning the block reward is given to even those miners who have comparatively less computational power
  - ❖ **Adjustable difficulty** ensures that the difficulty target for mining on the blockchain is regulated in response to increased hashing power and the number of users
  - ❖ **Quick verification** is a property which means that mining computational puzzles should be easy and quick to verify
-

# Major challenge of PoW

---

## Bitcoin (Double SHA-256) PoW:

- ❖ since the introduction of ASICs the power is shifting towards miners or mining pools who can afford to operate large-scale **ASIC farms**
- ❖ This challenges the core philosophy of the decentralization of bitcoin
- ❖ Self mutating puzzles; memory hard puzzles etc..

# Proof of Storage

---

- ❖ Also known as **proof of irretreviability**, this is another type of **proof of useful work** that requires storage of large number of data
- ❖ Introduced by Microsoft Research
- ❖ provides a useful benefit of distributed storage of archival data
- ❖ Miners are required to store a pseudo, randomly-selected subset of large data in order to perform mining

# Proof of Stake

- ❑ This proof is also called **virtual mining**
- ❑ another type of mining puzzle that has been proposed as an alternative to traditional PoW schemes ; first proposed in **PeerCoin** in August, 2012
- ❑ users are required to **demonstrate possession of a certain amount of currency (coins)** thus proving that they have a stake in the coin
- ❑ The simplest form of stake is where mining is made comparatively easier for those users who demonstrably own larger amounts of digital currency
- ❑ **Benefits:**
  - ❑ acquiring large amounts of digital currency is relatively difficult as compared to buying high-end ASIC devices
  - ❑ results in saving computational resources

# Proof of Stake - forms

---

- Proof of coinage
  - Proof of deposit
  - Proof of burn
  - Proof of activity (hybrid of PoW and PoS)
-

# Proof of coinage



- ❖ The age of a coin is the time since the coins were last used or held
- ❖ Different approach from the usual form of Proof of Stake where mining is made easier for users who have the highest stake in the altcoin
- ❖ In the coin-age-based approach the age of the coin (coinage) is reset every time a block is mined
- ❖ The miner is rewarded for holding and not spending coins for a time period
- ❖ This mechanism has been implemented in Peercoin combined with PoW in a creative way
- ❖ The difficulty of mining puzzles (PoW) is inversely proportional to the coin-age, meaning that if miners consume some coin-age using *coin-stake* transactions then the PoW requirements are relieved

# Proof of deposit



- ❖ The core idea behind this scheme is that newly minted blocks by miners are made unspendable for a certain period of time
- ❖ the coins get locked for a set number of blocks during the mining operation
- ❖ The scheme works by allowing miners to perform mining at the cost of freezing a certain number of coins for some time
- ❖ This is a type of Proof of Stake

# Proof of burn



- As an alternate expenditure to computing power, proof of burn in fact destroys a certain amount of bitcoins in order to get equivalent altcoins
- This is commonly used when starting up new coin projects as a means to provide a fair initial distribution
- an alternative mining scheme where the value of the new coins comes from the fact that previously a certain number of coins have been destroyed

# Proof of activity (hybrid of PoW and PoS)



- ❖ This scheme is a hybrid of PoW and Proof of Stake
- ❖ blocks are initially produced by using PoW - but then each block randomly assigns three stakeholders that are required to digitally sign it
- ❖ The validity of subsequent blocks is dependent on the successful signing of previously randomly chosen blocks



**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

**BLOCKCHAIN TECHNOLOGY**  
**BITS F452**  
**1<sup>st</sup> Sem 2022-23**  
**Lecture 10a**  
**Ethereum Blockchain**



# Introduction to Ethereum Blockchain

- ❖ Ethereum was conceptualized by *Vitalik Buterin* in November 2013
- ❖ key idea proposed was the development of a Turing-complete language that allows the development of arbitrary programs (smart contracts) for blockchain and decentralized applications
- ❖ In contrast to bitcoin, where the scripting language is very limited and allows basic and necessary operations only
- ❖ <https://ethereum.org/en/>
- ❖ Ethereum is the community-run technology powering the cryptocurrency ether (ETH) and thousands of decentralized applications

# More on Ethereum

---

- ❑ Various Ethereum clients have been developed using different languages
  - ❑ Currently most popular are **go-Ethereum** and **parity**
  - ❑ **go-Ethereum** was developed using **Golang**, whereas **parity** was built using **Rust**
  - ❑ **go-Ethereum** client known as ***geth*** is sufficient for all purposes
  - ❑ **Mist** is a user-friendly **Graphical User Interface (GUI) wallet** that runs **geth** in the background to sync with the network
-



- ❖ Ethereum is a decentralized, open-source blockchain with smart contract functionality
- ❖ Ether (ETH or  $\Sigma$ ) is the native cryptocurrency of the platform
- ❖ Amongst cryptocurrencies, Ether is second only to Bitcoin in market capitalization
- ❖ Ethereum has started implementing a series of upgrades called Ethereum 2.0, which includes a transition to proof of stake and aims to increase transaction throughput using sharding

A **database shard**, or simply a **shard**, is a horizontal partition of data in a database or search engine. Each shard is held on a separate database server instance, to spread load. Some data within a database remains present in all shards, but some appears only in a single shard. Each shard (or server) acts as the *single* source for this subset of data.

# Ethereum Protocol Upgrades

Code name	Release date	Release block
Frontier	30 July 2015 <sup>[29][4]</sup>	0
Ice Age	8 September 2015	200,000
Homestead	15 March 2016	1,150,000
DAO Fork	20 July 2016	1,920,000
Tangerine Whistle	18 October 2016	2,463,000
Spurious Dragon	23 November 2016	2,675,000
Byzantium	16 October 2017	4,370,000
Constantinople	28 February 2019 <sup>[30]</sup>	7,280,000
St. Petersburg	28 February 2019 <sup>[31]</sup>	7,280,000
Istanbul	8 December 2019	9,069,000
Muir Glacier	2 January 2020 <sup>[32]</sup>	9,200,000
Berlin	15 April 2021 <sup>[33]</sup>	12,244,000
London	5 August 2021 <sup>[34]</sup>	12,965,000

# Ethereum 2.0

---

- ✓ Open-source development is currently underway for a major upgrade to Ethereum known as **Ethereum 2.0** or **Eh2**
  - ✓ The main purpose of the upgrade is to increase transaction throughput for the network from the current of about **15 transactions per second** to up to **tens of thousands of transactions per second**
  - ✓ The goal is to increase throughput by splitting up the workload into many blockchains running in parallel (referred to as **sharding**) and then having them all share a common consensus proof-of-stake blockchain
  - ✓ To maliciously tamper with any singular chain would require one to tamper with the common consensus, which would cost the attacker far more than they could ever gain from an attack
-

# Ethereum 2.0 : The three phases

Ethereum 2.0 is also known as **Serenity**

The three phases are:

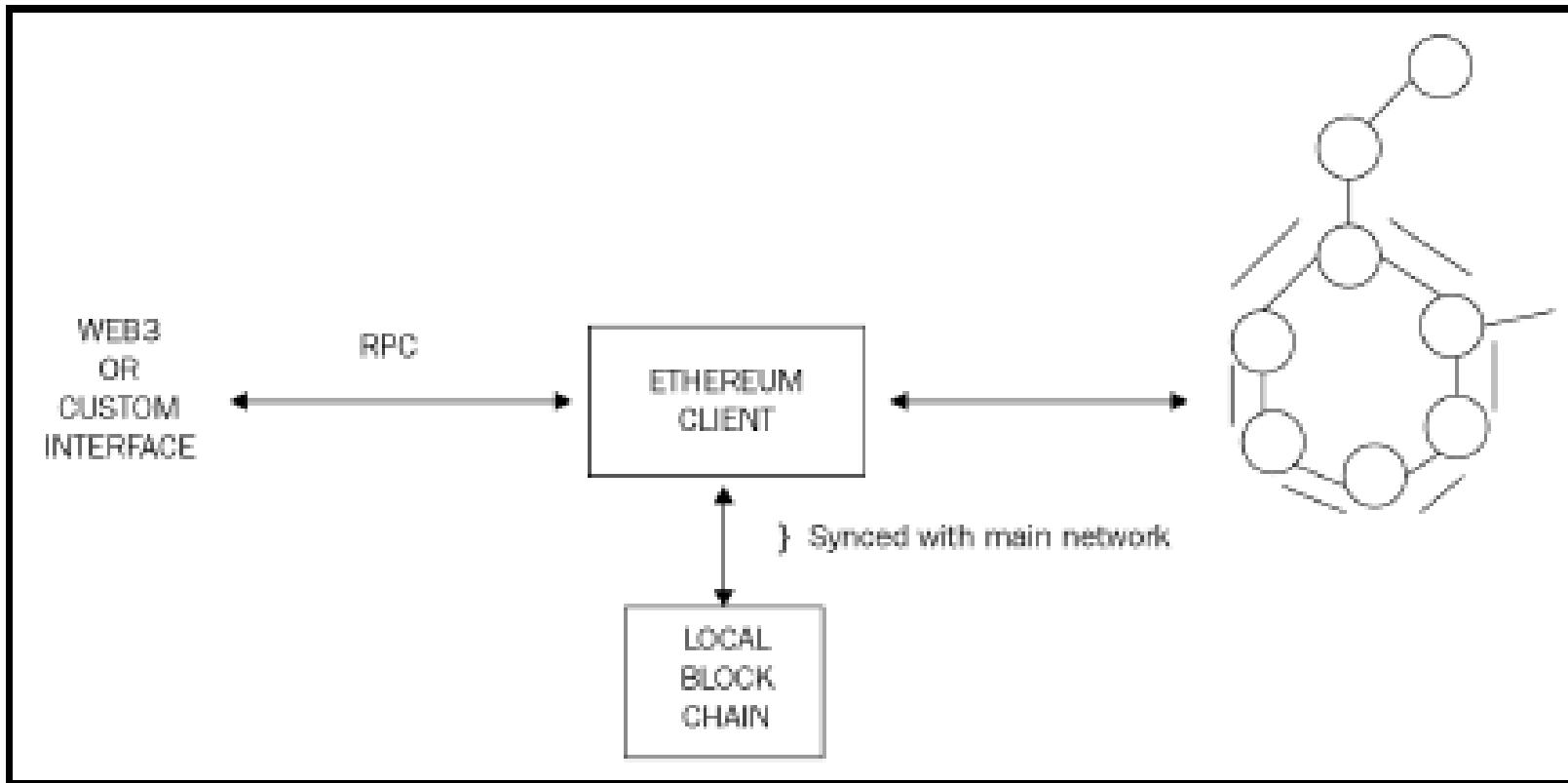
1. "Phase 0" also known as "**The Beacon Chain**" was launched in December 2020 and created the **Beacon Chain**, a proof-of-stake (PoS) blockchain that will act as the central coordination and consensus hub of Ethereum 2.0
2. "Phase 1" also known as "**The Merge**" will merge the Beacon Chain with the current Ethereum network, transitioning its consensus mechanism from proof-of-work to proof-of-stake. (Sept 2022)
3. "Phase 2" also known as "**Shard chains**" will implement state execution in the shard chains with the current Ethereum 1.0 chain expected to become one of the shards of Ethereum 2.0. Shard chains will spread the network's load across 64 new chains (2023).

<https://www.gemini.com/cryptopedia/ethereum-2-0-blockchain-roadmap-proof-of-stake-pos>

# The Ethereum stack

- ❖ At the core, there is the Ethereum blockchain running on the P2P Ethereum network
- ❖ Secondly, there's an Ethereum client (usually `geth`) that runs on the nodes and connects to the peer-to-peer Ethereum network from where blockchain is downloaded and stored locally
  - ❖ It provides various functions, such as mining and account management
  - ❖ The local copy of the blockchain is synchronized regularly with the network
- ❖ Another component is the `web3.js` library that allows interaction with `geth` via the **Remote Procedure Call (RPC) interface**

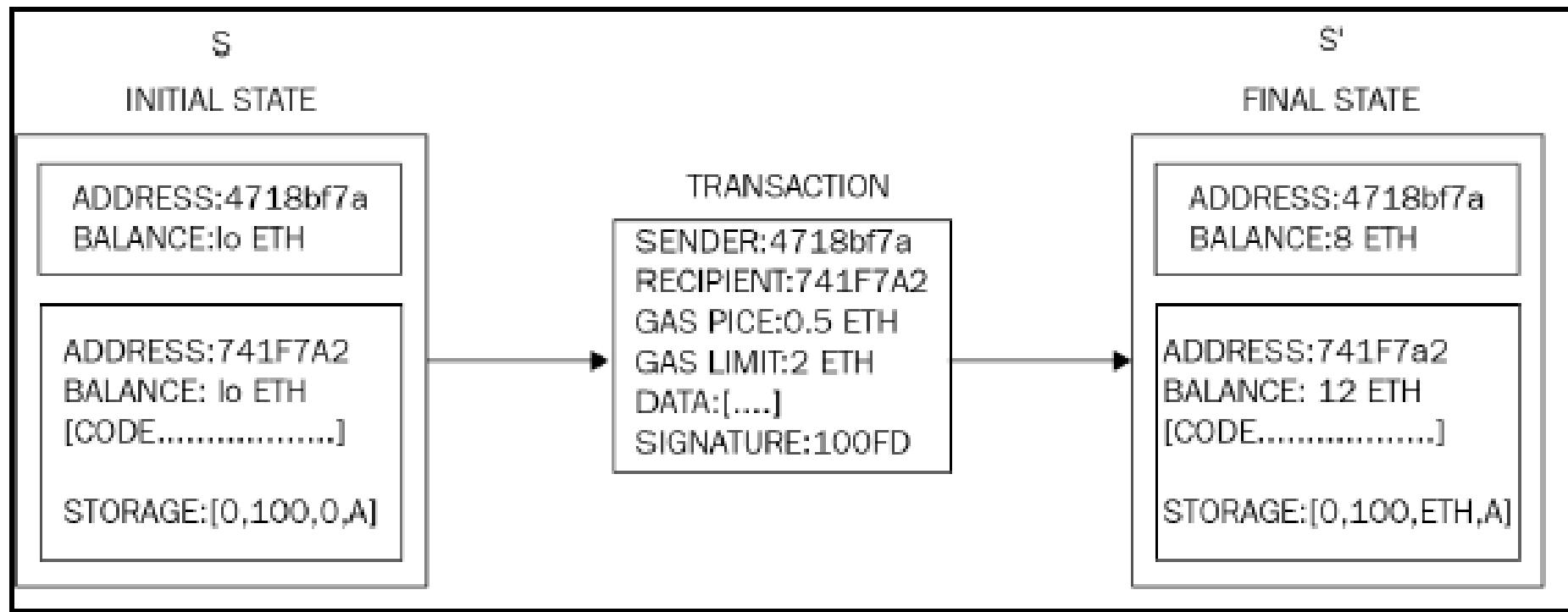
# Ethereum stack with various components



# Ethereum blockchain

- ✓ can be visualized as a **transaction-based state machine**
- ✓ The idea is that a **genesis state** is transformed into a **final state** by executing transactions incrementally
- ✓ The final transformation is then accepted as the **absolute undisputed version of the state**
- ✓ Ethereum state transition function => transaction execution results in a state transition

# Ethereum state transaction function



- ❑ Here, a transfer of 2 Ether from **Address 4718bf7a** to **Address 741f7a2** is initiated
- ❑ The initial state represents the state before the transaction execution and the final state is what the morphed state looks like

# Ethereum design

---

- ❖ Ethereum is a permissionless, non-hierarchical [network of computers](#) (nodes) which build and come to consensus on an ever-growing series of "blocks", or batches of transactions, known as the [blockchain](#)
- ❖ Each block contains an identifier of the block that it must immediately follow in the chain if it is to be considered valid
- ❖ Whenever a node adds a block to its chain, it executes the transactions therein in their order, thereby altering the ETH balances and other storage values of Ethereum accounts
- ❖ These balances and values, collectively known as the state, are maintained on the node's [computer](#) separately from the [blockchain](#), in a [Merkle](#) tree

# Ethereum design characteristics

---

- Each node communicates with a relatively small subset of the network, known as its peers
- Whenever a node wishes to include a new transaction in the blockchain, it sends the transaction to its peers, who then send it to their peers, and so on, it propagates throughout the network
- Certain nodes, called miners, maintain a list of all of these new transactions and use them to create new blocks, which they then send to the rest of the network
- Whenever a node receives a block, it checks the validity of the block and of all of the transactions therein and, if valid, adds it to its blockchain and executes all of said transactions
- As the network is non-hierarchical, a node may receive competing blocks, which may form competing chains
- The network comes to consensus on the blockchain by following the "longest-chain rule", which states that the chain with the most blocks at any given time is the canonical chain
- This rule achieves consensus because miners do not want to expend their computational work trying to add blocks to a chain that will be abandoned by the network.

# Ethererum currency

- ❖ Ether (ETH) is the cryptocurrency generated by the Ethereum protocol as a reward to miners in a proof-of-work system for adding blocks to the blockchain
- ❖ It is the only currency accepted in the payment of transaction fees, which also go to miners
- ❖ The block reward together with the transaction fees provide the incentive to miners to keep the blockchain growing (i.e. to keep processing new transactions)
- ❖ Therefore, ETH is fundamental to the operation of the network
- ❖ Each Ethereum account has an ETH balance and may send ETH to any other account
- ❖ The smallest subunit of ETH is known as a **Weí** and is equal to  **$10^{-18}$  ETH**
- ❖ Ether is often erroneously referred to as "Ethereum"
- ❖ Ether is listed on exchanges under the currency code ETH. The Greek uppercase Xi character **Ξ** is sometimes used for its **currency symbol**