Blockchain Engineering Playbook

Rohas Nagpal

This book is part of the Official Courseware of the free **Blockchain Engineering Program** conducted by Rohas Nagpal.

To register, join one of these WhatsApp Groups:
Group 1 or Group 2 or Group 3

The Official LinkedIn Group for this course is:
https://www.linkedin.com/groups/9082777

You can subscribe to my Blockchain Newsletter:
https://blockchainblog.substack.com

Contents

This book is part of the courseware of the free Blockchain Engineering Program conducted by Rohas Nagpal.

01

Blockchain Basics

A blockchain:

- is a linear, chronological structure
- consists of blocks of data (transactions) that are chained together.

Blockchains are **Internets of Value.**

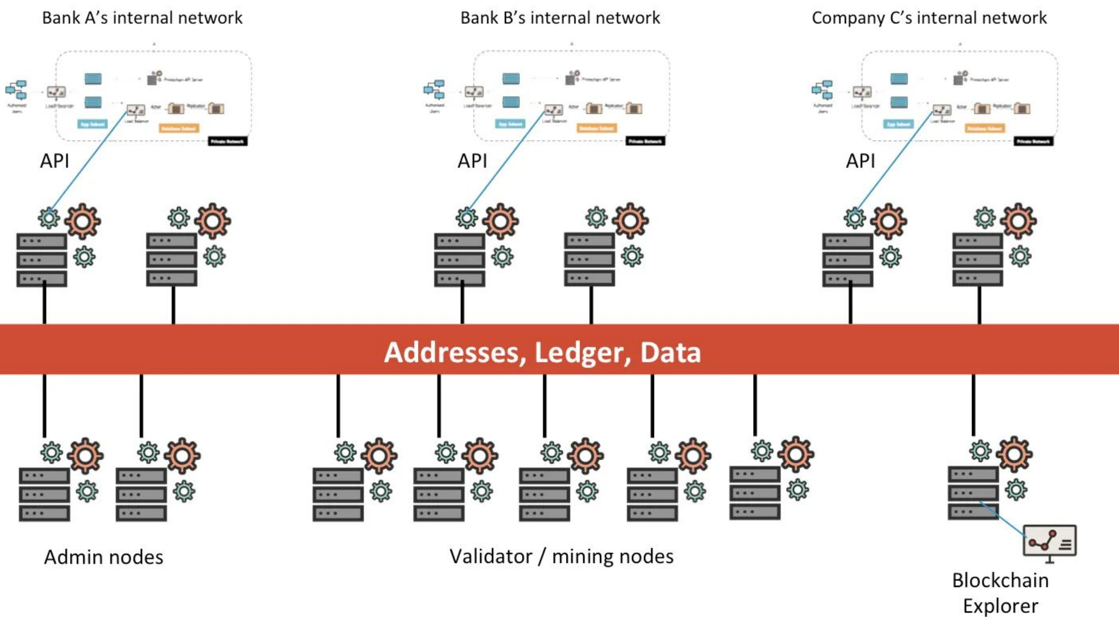
The conventional internet uses protocols like TCP/IP and SMTP to move data across the globe in seconds (email video pdf, text, etc).

Blockchains enable the **movement of value** across the world in seconds.

This value can be crypto assets like Bitcoin & Monero or tokenized versions of real-world assets like airplanes, carbon credits, real estate, etc.

Tokenization of real-world assets is the most important Blockchain use case. It is explained in depth later in this book.

Blockchains increase transparency and reduce the need for intermediaries in financial use cases.



Conceptual Image of a Blockchain Network

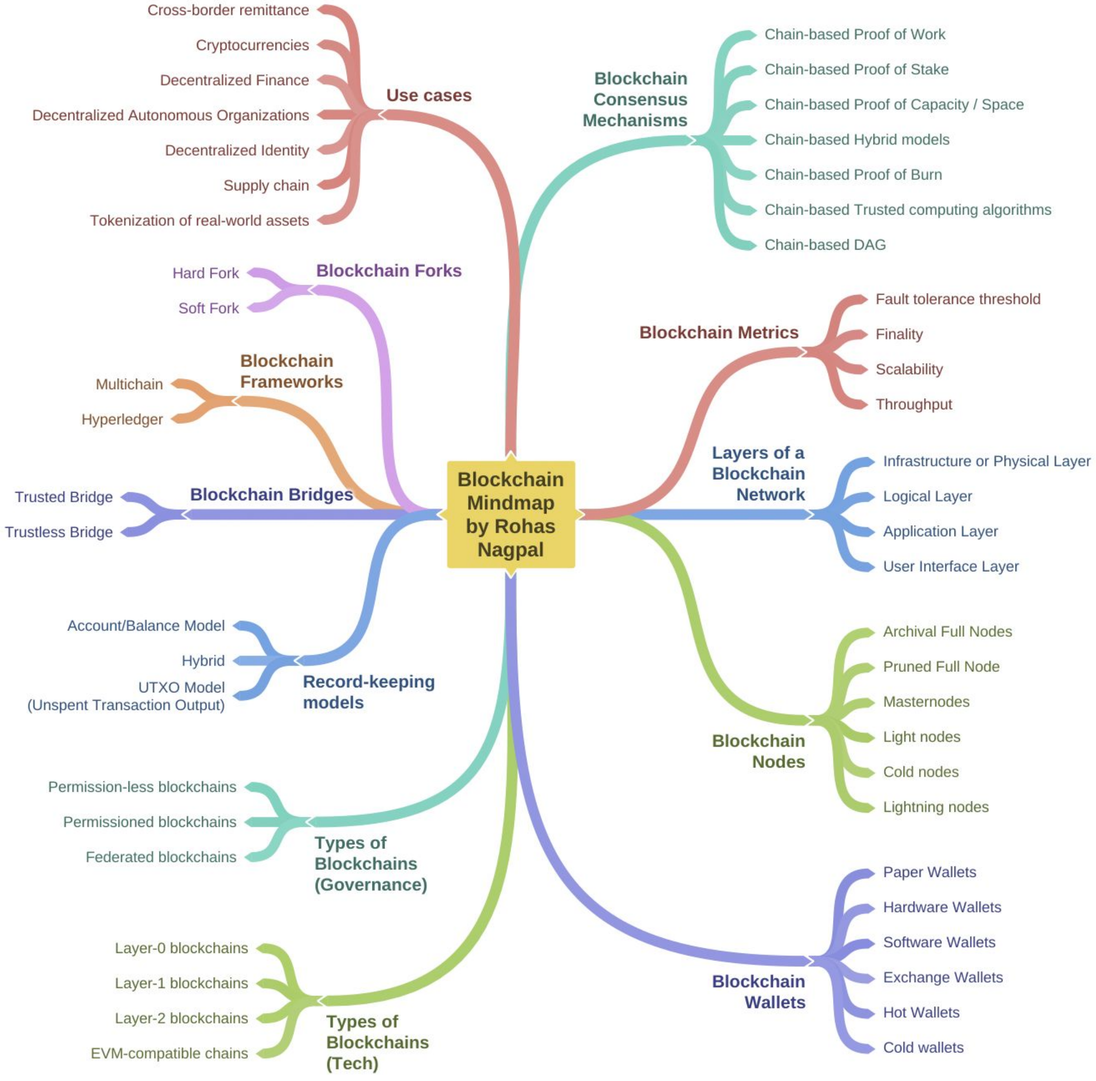
While Blockchains have a linear chronological structure, Distributed Ledger Technologies (DLTs) can have different structures, such as:

- Graph e.g. IOTA
- Mesh e.g. Holochain
- Tree e.g. Corda

DLTs are suited for multiple use cases such as:

- Supply Chain Management
- Identity and Verification
- Healthcare
- Government and Public services

All Blockchains are DLTs,
but all DLTs are not Blockchains.


Blockchain Mindmap by Rohas Nagpal
Use cases
Cross-border remittance
Cryptocurrencies
Decentralized Finance
Decentralized Autonomous Organizations
Decentralized Identity
Supply chain
Tokenization of real-world assets
Blockchain Forks
Hard Fork
Soft Fork
Blockchain Frameworks
Multichain
Hyperledger
Blockchain Bridges
Trusted Bridge
Trustless Bridge
Record-keeping models
Account/Balance Model
Hybrid
UTXO Model (Unspent Transaction Output)
Types of Blockchains (Governance)
Permission-less blockchains
Permissioned blockchains
Federated blockchains
Types of Blockchains (Tech)
Layer-0 blockchains
Layer-1 blockchains
Layer-2 blockchains
EVM-compatible chains
Blockchain Consensus Mechanisms
Chain-based Proof of Work
Chain-based Proof of Stake
Chain-based Proof of Capacity / Space
Chain-based Hybrid models
Chain-based Proof of Burn
Chain-based Trusted computing algorithms
Chain-based DAG
Blockchain Metrics
Fault tolerance threshold
Finality
Scalability
Throughput
Layers of a Blockchain Network
Infrastructure or Physical Layer
Logical Layer
Application Layer
User Interface Layer
Blockchain Nodes
Archival Full Nodes
Pruned Full Node
Masternodes
Light nodes
Cold nodes
Lightning nodes
Blockchain Wallets
Paper Wallets
Hardware Wallets
Software Wallets
Exchange Wallets
Hot Wallets
Cold wallets

Blockchain Nodes

A blockchain is a **transaction database** shared by all nodes participating in a network.

A node is a computer that runs the blockchain's software to validate & store the complete history of transactions.

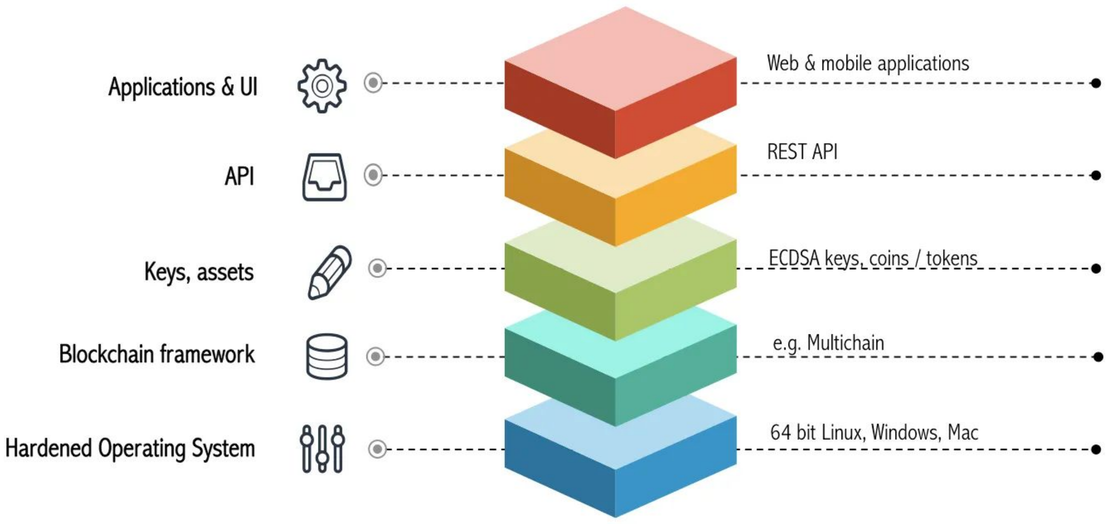
A full copy of a blockchain contains every transaction ever executed in the native coin (e.g. ETH) and tokens (e.g. BAT) created on the blockchain.

Transactions are bundled in **blocks**.

Each block contains a record of some / all recent transactions. Each block also contains a reference to the block that came immediately before it.

A blockchain node is a **computer** that runs the blockchain's software to validate & store the complete history of transactions on the network.

Blockchain nodes **constantly exchange information** on new transactions & blocks.


Applications & UI
API
Keys, assets
Blockchain framework
Hardened Operating System
Web & mobile applications
REST API
ECDSA keys, coins / tokens
e.g. Multichain
64 bit Linux, Windows, Mac

A conceptual overview of a Blockchain Node

Archival Full Nodes

They are the most important. They host the entire blockchain, validate blocks & maintain consensus.

Pruned Full Node

A pruned full node first downloads the entire blockchain and then deletes blocks beginning with the oldest block till it holds only the most recent transactions up to the size limit set by the operator.

Master nodes

Users run masternodes to earn network rewards. Some amount of native tokens have to be "locked" by masternode operators.

Light nodes

A light node does not hold the full copy of the blockchain. It saves download time & storage space by only downloading block headers.

Cold nodes

They are used for signing transactions offline and storing private keys away from the network.

Lightning nodes

They reduce the load on the network by enabling off-chain transactions. These nodes enable faster and cheaper transactions.

Mining

In a Proof of Work blockchain like Bitcoin, each block also contains an answer to a difficult-to-solve **mathematical puzzle**.

Mining is the process of competing to be the next to find the answer that "solves" the current block.

New blocks cannot be submitted to the network without the correct answer.

The mathematical problem in each block is very difficult to solve, but once a valid solution is found, it is very easy to verify.

Every block contains a **hash of the previous block**.

This creates a chain of blocks from the first (genesis) block to the current block.

It is extremely difficult to modify a block once it has been in the blockchain for some time because every block after it would also have to be generated again.

Layers of a Blockchain Network

Layer 0: Infrastructure or Physical Layer
This consists of the hardware, software, and networks that enable the functioning of the blockchain.

Layer 1: Logical Layer
This consists of the protocols, rules, and standards that define the functioning of the blockchain.

Layer 2: Application Layer
This consists of the protocols, applications, and services that build on top of the basic building blocks provided by layer 1.

Layer 3: User Interface Layer
This consists of the interfaces, applications, and services that enable users to interact with the blockchain and access its underlying functionality.

The infographic below depicts the 4 layers of a Blockchain Network.

Layers of a Blockchain Network

Something missing? Spotted an error? Email me: rohasnagpal@gmail.com

User interface

Wallets

Applications that enable users to manage and transact tokens within the blockchain e.g. balance tracking, transaction history, sending & receiving.

External Integrations

Integration of the blockchain with external systems e.g. databases, APIs etc. to enable the exchange of data and functionality.

Application Layer

Scaling Solutions

Protocols & technologies designed to improve the scalability and performance of the blockchain e.g. state channels, sidechains & payment channels.

Decentralized Applications

Applications built on top of the blockchain and using its underlying infrastructure to function e.g. DEX, prediction markets.

Protocols & Standards

Additional protocols & standards enabling specific functions or capabilities within the blockchain e.g. identity management protocols.

Services & Tools

Services & tools built on top of the blockchain to enable specific functions or capabilities e.g. asset management, data analytics, governance.

Logical Layer

Consensus Protocol

Rules and procedures used to reach consensus on the state of the blockchain e.g. proof-of-work or proof-of-stake.

Network Protocol

Rules and standards that govern the communication & transmission of data within the blockchain network e.g. gossip protocol.

Cryptographic Primitives

Basic cryptographic functions e.g. hash functions & digital signatures, that are used to secure and validate transactions within the blockchain.

Token & Transaction Model

Rules and standards that govern the creation and transfer of tokens or other assets within the blockchain.

Infrastructure or Physical Layer

Hardware

Servers, computers, and other physical devices that are used to store, process, and transmit data within the blockchain network.

Network infrastructure

Cables, routers, switches, and other devices that enable the communication and transmission of data within the blockchain network.

Connect with me @rohasnagpal on LinkedIn, Instagram, YouTube, Twitter, or Gmail.

Types of Blockchains

Layer-0 Blockchains

Layer-0 blockchains enable developers to create custom blockchains.

Example: Polkadot enables the creation of sovereign blockchains with their own tokens.

These parachains plug into a single base platform called the Relay Chain which is responsible for shared security, consensus & cross-chain interoperability.

Other examples of L0 are Cosmos and Horizen.

L0 blockchains are different from Blockchain Frameworks like Multichain and Hyperledger Fabric.

Frameworks are open-source software that enable developers to create custom blockchains.

Remember: Layer-0 blockchains are networks while Frameworks are software programs.

Layer-1 Blockchains

Layer-1 blockchains validate & execute transactions without the need for any external network. *Examples:* Bitcoin, and Ethereum Mainnet.

Layer-2 Blockchains

Layer-2 blockchains are "sidechains" built on top of Layer-1 blockchains. The underlying Layer-1 (e.g. Ethereum Mainnet) provides decentralization & security, while the Layer-2 (e.g. Polygon PoS) provides scalability.

Permission-less blockchains

Anyone can participate on public / permission-less blockchains without restrictions. *Examples:* Bitcoin, Litecoin, Ethereum.

Permissioned blockchains

Various controls can be set in a private / permissioned blockchain.

Example: Hybrid Finance Blockchain (HYFI) where permissions such as connect, send, receive, issue, create, mine, activate, and admin can be set.

Federated blockchains

In a federated / consortium blockchain network, the validation process is controlled by a select set of nodes.

EVM-compatible chains

Ethereum Virtual Machine (EVM) is "the environment in which all 'Ethereum' accounts and smart contracts live".

Smart contracts are programs that run automatically when some pre-defined conditions are met.

EVM compatibility is the ability to write & deploy smart contract code that are:

1. compatible with EVM, and
2. can be recognized by Ethereum nodes.

Examples of EVM-compatible blockchains are Avalanche, Binance Smart Chain, Fantom, and Polygon.

Blockchain Consensus Mechanisms

Consensus algorithms are the **heart of blockchains.**

They enable network participants to agree on the contents of a blockchain in a distributed and trustless manner.

Today there are over 75 consensus algorithms including:

Proof-of-Work (PoW)
This was the world’s first consensus algorithm. Miners "solve" mathematical puzzles by investing in electricity and computational power e.g. Bitcoin.

Proof of Stake (PoS)
Holders "lock" a number of coins as a "stake" and are randomly assigned validation rights for a new block e.g. Algorand.

Hybrid PoW / PoS
This brings together the security of Proof of work and the governance & energy efficiency of Proof-of-Stake e.g. Decred.

Proof-of-Work-Time (PoWT)

This features a variable blocktime that scales with mining power. The blockchain speeds up with power increases. This mechanism scales the blockchain well and increases transaction speed with power.

Proof of Meaningful Work (PoMW)

The energy invested by miners' is used for calculations that benefit public scientific research projects (e.g. medical research for cures, chemical research, and astrophysical simulations).

Proof of Elapsed Time (PoET)

Each node goes to sleep for a random "wait time" generated by the network. The node with the shortest wait time wakes up first and commits a new block e.g. Hyperledger Sawtooth.

Proof-of-replication (PoRep)

Storage miners prove 2 things:

- that they are using space to store replicas of data,
- that the data can easily be accessed.

They get rewards in exchange for their storage space e.g. Filecoin.

Delegated Proof of Stake (DPoS)

Holders "lock" a number of coins as a "stake" but outsource validation to "delegates" selected based on reputation and trustworthiness e.g. Bitshares.

Proof-of-Spacetime (PoSt)

Randomly selected miners prove that they have been physically storing data for a certain period of time e.g. Filecoin.

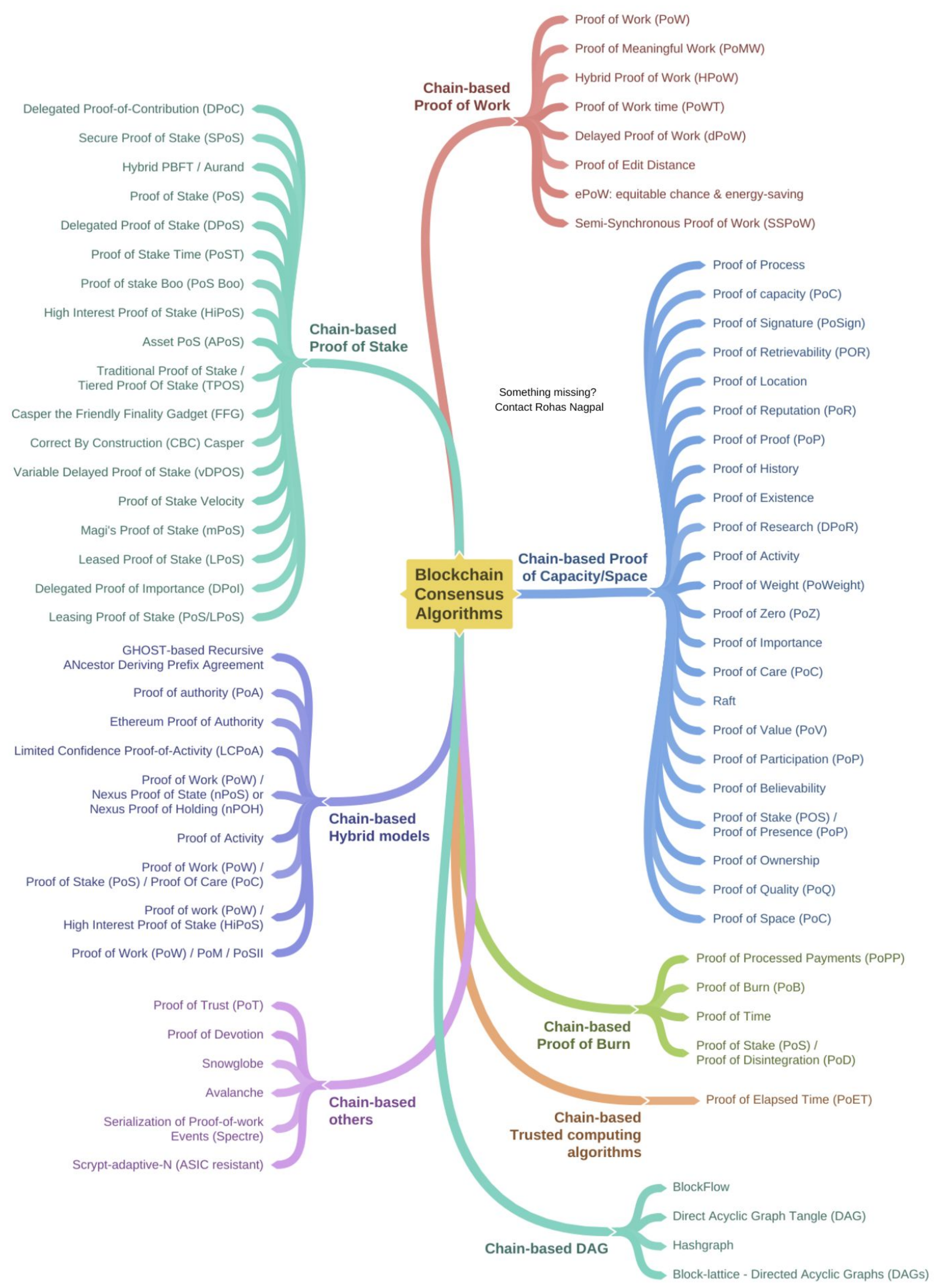
Proof-of-burn (PoB)

Miners reach a consensus by sending coins to an "eater" or "burn" address. This permanently eliminates coins from circulation, reduces inflation, and validates transactions e.g. Slimcoin.

Proof-of-Authority (PoA)

Identified, known, and credible validators produce blocks in this system. It is meant for private & enterprise blockchains.

This mindmap shows 82 blockchain consensus mechanisms divided into 9 categories.


Blockchain Consensus Algorithms
Chain-based Proof of Work
Proof of Work (PoW)
Proof of Meaningful Work (PoMW)
Hybrid Proof of Work (HPoW)
Proof of Work time (PoWT)
Delayed Proof of Work (dPoW)
Proof of Edit Distance
ePoW: equitable chance & energy-saving
Semi-Synchronous Proof of Work (SSPoW)
Chain-based Proof of Stake
Delegated Proof-of-Contribution (DPoC)
Secure Proof of Stake (SPoS)
Hybrid PBFT / Aurand
Proof of Stake (PoS)
Delegated Proof of Stake (DPoS)
Proof of Stake Time (PoST)
Proof of stake Boo (PoS Boo)
High Interest Proof of Stake (HiPoS)
Asset PoS (APoS)
Traditional Proof of Stake / Tiered Proof Of Stake (TPOS)
Casper the Friendly Finality Gadget (FFG)
Correct By Construction (CBC) Casper
Variable Delayed Proof of Stake (vDPOS)
Proof of Stake Velocity
Magi's Proof of Stake (mPoS)
Leased Proof of Stake (LPoS)
Delegated Proof of Importance (DPoI)
Leasing Proof of Stake (PoS/LPoS)
Something missing? Contact Rohas Nagpal
Chain-based Proof of Capacity/Space
Proof of Process
Proof of capacity (PoC)
Proof of Signature (PoSign)
Proof of Retrievability (POR)
Proof of Location
Proof of Reputation (PoR)
Proof of Proof (PoP)
Proof of History
Proof of Existence
Proof of Research (DPoR)
Proof of Activity
Proof of Weight (PoWeight)
Proof of Zero (PoZ)
Proof of Importance
Proof of Care (PoC)
Raft
Proof of Value (PoV)
Proof of Participation (PoP)
Proof of Believability
Proof of Stake (POS) / Proof of Presence (PoP)
Proof of Ownership
Proof of Quality (PoQ)
Proof of Space (PoC)
Chain-based Hybrid models
GHOST-based Recursive ANcestor Deriving Prefix Agreement
Proof of authority (PoA)
Ethereum Proof of Authority
Limited Confidence Proof-of-Activity (LCPoA)
Proof of Work (PoW) / Nexus Proof of State (nPoS) or Nexus Proof of Holding (nPOH)
Proof of Activity
Proof of Work (PoW) / Proof of Stake (PoS) / Proof Of Care (PoC)
Proof of work (PoW) / High Interest Proof of Stake (HiPoS)
Proof of Work (PoW) / PoM / PoSII
Chain-based Proof of Burn
Proof of Processed Payments (PoPP)
Proof of Burn (PoB)
Proof of Time
Proof of Stake (PoS) / Proof of Disintegration (PoD)
Chain-based others
Proof of Trust (PoT)
Proof of Devotion
Snowglobe
Avalanche
Serialization of Proof-of-work Events (Spectre)
Scrypt-adaptive-N (ASIC resistant)
Chain-based Trusted computing algorithms
Proof of Elapsed Time (PoET)
Chain-based DAG
BlockFlow
Direct Acyclic Graph Tangle (DAG)
Hashgraph
Block-lattice - Directed Acyclic Graphs (DAGs)

Blockchain Bridges

Let's say you live in India and are traveling to Singapore. In India, we use Indian Rupees (INR) while in Singapore, they use Singapore Dollars (SGD).

So you could convert some money from INR to SGD using a bank. Or you could use an international card that enables you to pay in SGD when you are in Singapore.

This is simple in the world of banks. But a lot more difficult in the world of Blockchains.

Ethereum Mainnet and Terra are two independent Blockchains. They have different rules and consensus mechanisms. The native token of Ethereum Mainnet is ETH while that of Terra is LUNA.

Now suppose that you have a bunch of LUNA on Terra, but want to use it on the Ethereum Mainnet. How do you do that?

That's where a Blockchain Bridge comes into the picture. You could use the Terra Bridge to buy Wrapped Luna (WLUNA) which is an ERC-20 token native to the Ethereum Mainnet.

In simple terms, here's how a typical bridge works:

1. It receives one type of crypto e.g. LUNA.

2. It locks this LUNA as a deposit.

3. It "mints" an equal amount of another crypto e.g. Wrapped LUNA and releases it on another blockchain e.g. Ethereum Mainnet.

Hacking Blockchain Bridges

Because bridges hold a ton of crypto, they are juicy targets for hackers.

And because writing the code for bridges is insanely complex, hackers are having a field day plundering them.

When a bridge gets attacked, the hacker withdraws crypto from one side of the bridge, e.g. Wrapped LUNA, without depositing anything on the other side e.g. LUNA.

One of the biggest crypto attacks took place in March 2022 when the Ronin bridge, built for Axie Infinity, was hacked for $600+ million in ETH and USDC.

Ronin worked "off-chain" - it interfaced with the blockchain but existed on external servers that were not a part of the blockchain.

Ronin relied on 9 validator nodes, of which 5 nodes were needed to validate transactions.

The hackers exploited code vulnerabilities and also used social engineering.

Another major Bridge hack targeted Wormhole which supports 6 blockchains - Terra, Solana, Ethereum, Binance Smart Chain, Avalanche, and Polygon.

The cost of the hack was $325 million.

Bridges can be **Trusted or Trustless.**

A **Trusted Bridge** depends upon a central entity and you lose control of your cryptos when you use it.

A **Trustless Bridge** operates using smart contracts and you remain in control of your cryptos.

Blockchain Metrics

Throughput

Throughput is the number of transactions per second that a Blockchain consensus algorithm can process.

Fault tolerance threshold

Fault tolerance threshold is the upper limit of faulty nodes that directly impacts the performance of a blockchain consensus algorithm.

Finality

Finality (also called Latency) represents the time it takes for a transaction to be settled in the "ledger" of a blockchain.

Scalability

Scalability is the ability of a blockchain to expand without degrading performance.

Bitcoin

- Throughput: 7 tps
- Threshold: 51%
- Finality: 60 min

Ethereum Mainnet

- Throughput: 14 tps
- Threshold: 51%
- Finality: 6 min

Polkadot

- Throughput: 1500 tps
- Threshold: 33%
- Finality: 60 secs

Merged Mining

Merged mining is the process of **mining two or more blockchains at the same time.**

Essentially the same proof of work can be used on multiple chains.

Merged mining is technically called **Auxiliary Proof-of-Work** (AuxPoW).

The concept is that the work done on one blockchain can be leveraged as valid work on another blockchain.

The blockchain that provides the proof of work is called the parent blockchain.

The blockchain that accepts the proof of work as valid is the auxiliary blockchain.

To perform merged mining, all the involved blockchains should be using the same algorithm.

Since Bitcoin uses SHA-256, any other blockchain that uses SHA-256 can be mined along with Bitcoin subject to some technical implementations.

Merged Mining increases the profitability and performance of mining and is hugely beneficial for miners.

Some examples of merge mining:

- Dogecoin and Litecoin
- Bitcoin and RSK

Blockchain Forks

A "**fork**" is when a single blockchain splits into two.

Forks can be categorized as **hard forks** or **soft forks**.

Hard Fork

A hard fork is the result of an extensive network change. Each node has to upgrade its software to be compatible with the new processes.

If a node does not upgrade its software, it will be on a different blockchain.

Soft Fork

In a soft fork, nodes running the old software are still able to validate transactions. Software forks are backward-compatible.

Ethereum Fork

In 2016, the world's first decentralized autonomous organization (DAO) raised about US$ 150 million worth of ETH through a token sale.

But a hacker exploited a bug in the "smart contract" and siphoned out all the money!

A hard fork rolled back Ethereum's history to before the hack.

This reallocated the hacked ether to a different "smart contract" and allowed investors to withdraw their funds.

The purists hated this, which led to Ethereum splitting into 2 blockchains: Ethereum and Ethereum Classic.

Bitcoin Cash Fork

In July 2017, Bitcoin (BTC) miners representing more than 80% of the Bitcoin computing power voted to incorporate the SegWit2x (segregated witness) technology to improve Bitcoin.

Many miners and developers, who did not want SegWit2x to be introduced, initiated a hard fork and created a new blockchain and cryptocurrency - Bitcoin Cash (BCH).

Blockchain Addresses

When you buy gold or real estate, you actually get physical possession.

Blockchain assets (e.g. cryptocurrencies) are very different.

On a blockchain, we all need a key pair - public key & private key.

This key pair is generated using an algorithm such as **ECDSA** (Elliptic Curve Digital Signature Algorithm).

Sample Private Key

b5ba96aae89dc703c27ec5b3d478a8b176b874f248c8 3e533d9edc18e6356d44

The private key is what you would need to digitally "sign" transactions i.e. to send crypto to someone else.

If someone gets hold of your private key, they can transfer all your crypto to another address.

This is what happens in most crypto ‘hacks’.

That's why we say that "**Not your keys, not your coins**".

Sample Public Key
02b90ecf13d0ef14cd777f3a427b712ddd46703375f458092714c1c72106803ca2

The public key is used to verify the digital signature. It proves ownership of the private key.

Sample WIF
L3Jy6k2KCJ6uDNEj1hirw49sPWghT7Cg77rSDvfpAkAC7F63PhGe

A "Wallet Import Format" (wif) is a shorter version of a private key.

Sample Address
1AAE1EDcCAUmyBfi46G3vpik8oCaKVoabT

The address is similar to your bank account or UPI ID. Anyone can send crypto to your address. The address is derived from the public key.

If you send crypto to the "wrong" address, it’s gone forever!

Blockchain Wallets

A blockchain wallet is designed to:

- store your public and private keys,
- send and receive cryptos,
- monitor 'balances', and
- interact with supported blockchains.

There are **4 main types of wallets**:

1. Paper Wallets
2. Hardware Wallets
3. Software Wallets
4. Exchange Wallets

A hot wallet is one that is connected to the Internet and is considered the most vulnerable to hacking.

Examples: Software Wallets, Exchange Wallets.

A cold wallet is not connected to the Internet and is considered more secure.

Examples: Paper Wallets, Hardware Wallets,

Paper wallets are inconvenient to use but are the safest option. Consider using them if you have a large number of crypto assets to keep for a long period of time.

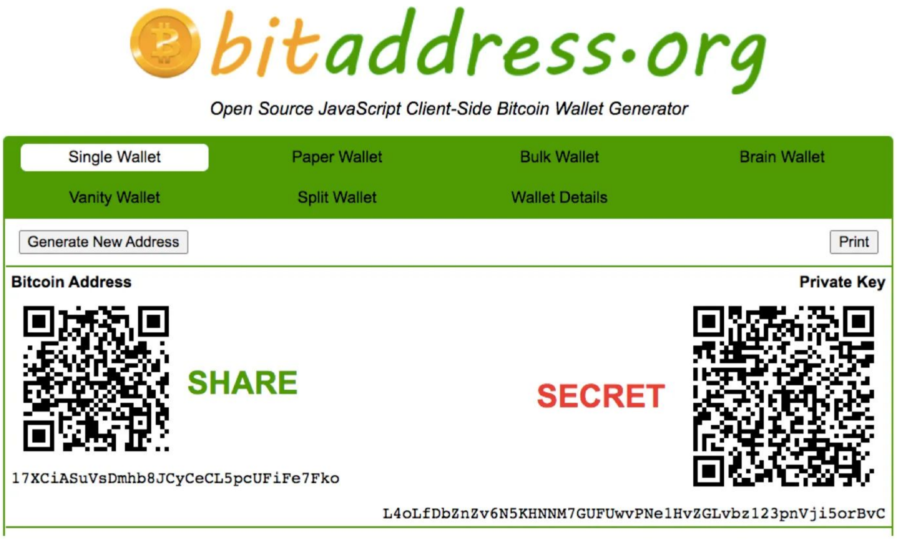
BitAddress is a JavaScript Client-Side Bitcoin Wallet Generator.

You can use it to generate Bitcoin addresses and their corresponding private keys in a web browser.

Site: https://www.bitaddress.org

To generate a Bitcoin address and its corresponding private key, simply move your mouse randomly on the bitaddress page.

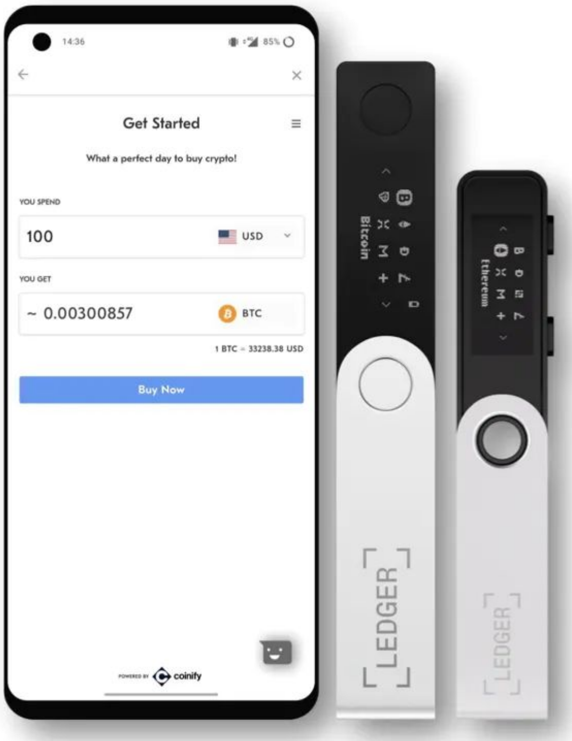
A wallet will be generated in your web browser. It will look something like this:



17XCiASuVsDmhb8JCyCeCL5pcUFiFe7Fko

L4oLfDbZnZv6N5KHNNM7GUFUwvPNe1HvZGLvbz123pnVji5orBvC

Hardware wallets are a little pricey and there's always the risk of losing or breaking them.



Software wallets are free & easy to use. But if you accidentally delete them, your crypto is gone forever.

Examples: Trust Wallet, Metamask

So remember to back up the **seed phrase** - a bunch of words that you can write down. Example:

history
lumber
quote
board
young
dove
robust
kit
invite
plastic
regular
skull

Wallets provided by Crypto Exchanges are called "custodial" because the private keys are under the control of the Exchange.

If they go bankrupt, you lose all your crypto.

Remember: Not your keys, not your coins.

Blockchain Record-keeping Models

Accounting or **record-keeping** methods are used by blockchains to determine the provenance and ownership of coins in the network.

3 types of record-keeping models are used in blockchain networks:

1. UTXO (Unspent Transaction Output) Model e.g. Bitcoin

2. Account/Balance Model e.g. Ethereum

3. Hybrid e.g. Cardano

Each Bitcoin transaction spends **output** from prior transactions.

Each Bitcoin transaction also generates new outputs that can be spent by transactions in the future.

A Bitcoin user's wallet keeps track of all unspent transactions associated with all addresses owned by the user.

The balance of the wallet is the sum of those unspent transactions.

UTXO can be compared to spending banknotes and getting back change.

The account / balance model can be compared to spending using a debit or credit card.

The benefits of the UTXO Model are scalability and privacy (if the user uses new addresses for each transaction).

The benefits of the account / balance model are simplicity and efficiency.

Bitcoin derivatives such as Bitcoin Cash, Zcash, Litecoin, Dogecoin, Dash, etc. are UTXO chains.

Ethereum, EOS, Tron, and Ethereum Classic are account-based chains.

Cardano, Komodo, Tron, and Qtum use a hybrid model.

Tokenization

Imagine that you are relaxing on your favorite beach. You whip out your smartphone and within minutes you've...

...bought shares in an innovative startup halfway across the world

...traded a fraction of a Picasso painting for a fantastic pair of collectible sneakers

...invested in the copyright license of your favorite movie

...swapped your gold & platinum holdings for fractional ownership of a private plane and a cruise liner...

...invested in fractional ownership of an office building in an upcoming high-rent location...

That's the **power of tokenization.**

Tokenization
= Asset
+ Blockchain
+ Dematerialization
+ Fractionalization
+ Legal compliance

What is Tokenization?

Tokenization is the **creation of Blockchain Tokens** that represent real-world assets - from airplanes & carbon credits to ships & real estate.

Each token represents a **fraction of ownership** in the underlying asset, enabling investors to buy, sell, or trade these tokens on digital asset marketplaces.

Tokenization bridges the gap between traditional finance and the digital world, leveraging the benefits of blockchain technology.

Tokenization of real-world assets is a multi-trillion-dollar opportunity.

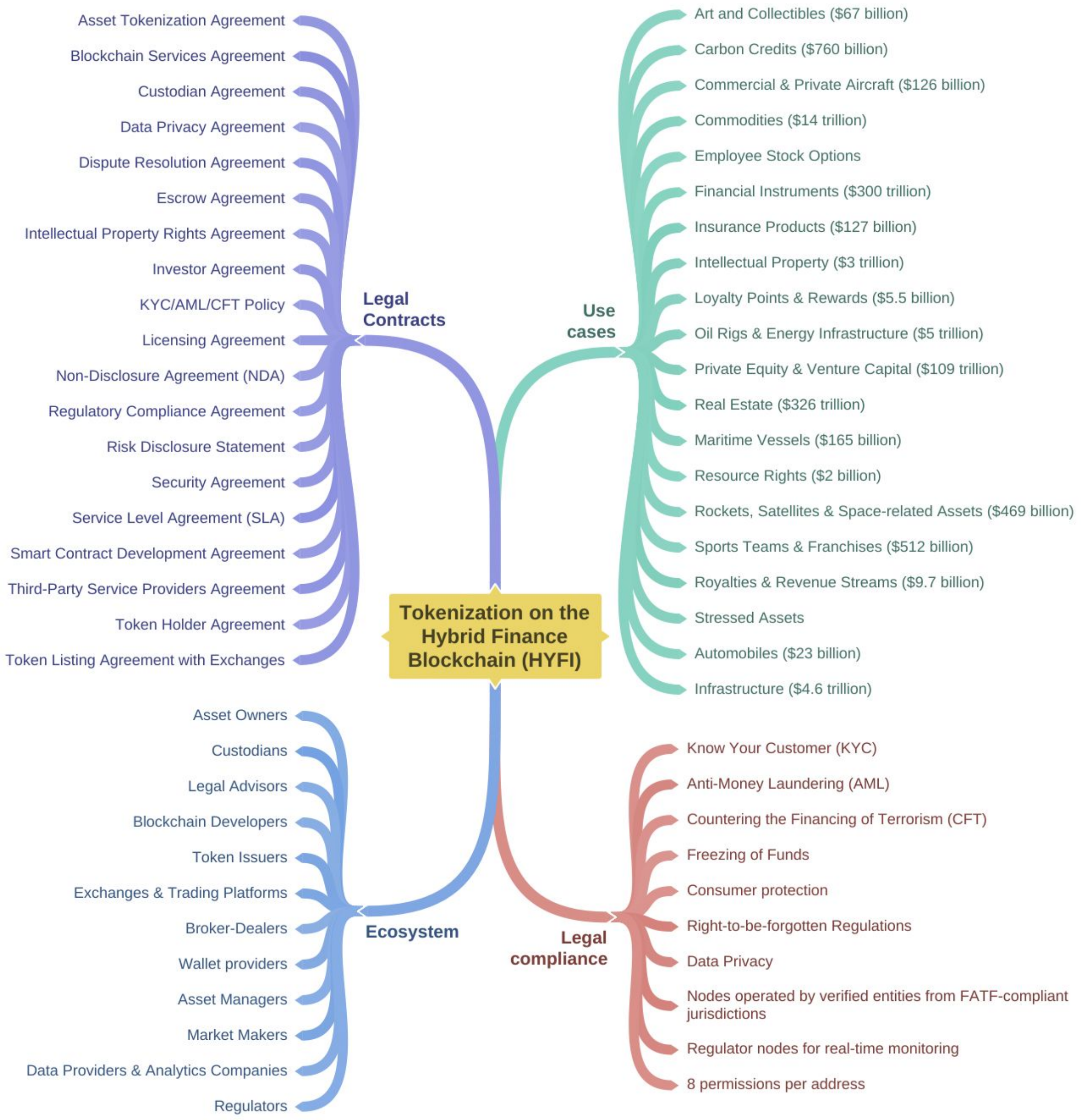
For **asset owners**, Tokenization offers benefits like:

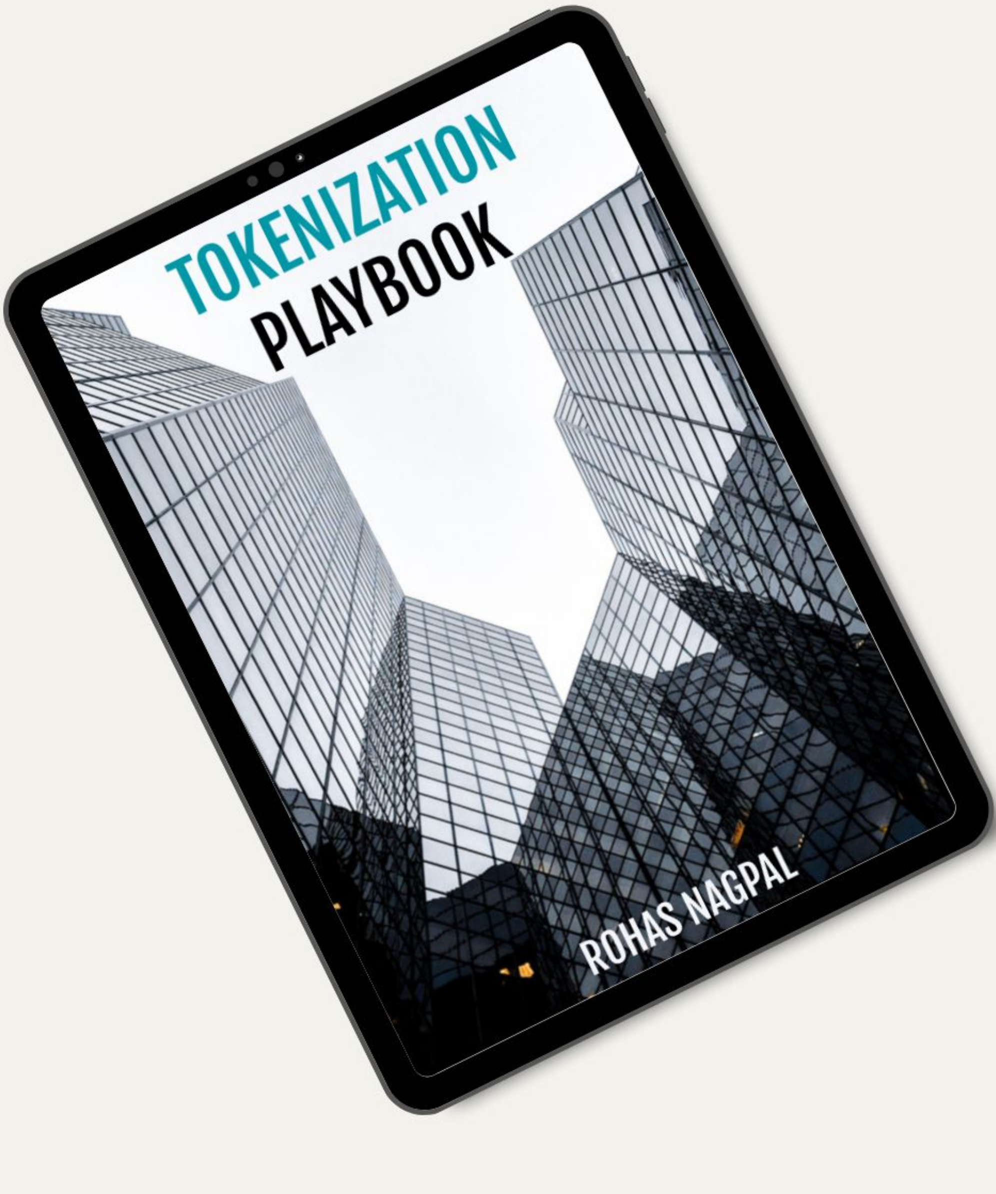
- Access to a Larger Investor Base
- Increased Liquidity
- Efficient Capital Raising
- Flexibility in Asset Management

For **investors**, Tokenization offers benefits like:

- Lower Entry Threshold
- Portfolio Diversification
- Global Access to Assets
- Transparency and Security
- Improved Liquidity

Here's a mind map of how it works on the Hybrid Finance Blockchain (HYFI)


Tokenization on the Hybrid Finance Blockchain (HYFI)
Legal Contracts
Asset Tokenization Agreement
Blockchain Services Agreement
Custodian Agreement
Data Privacy Agreement
Dispute Resolution Agreement
Escrow Agreement
Intellectual Property Rights Agreement
Investor Agreement
KYC/AML/CFT Policy
Licensing Agreement
Non-Disclosure Agreement (NDA)
Regulatory Compliance Agreement
Risk Disclosure Statement
Security Agreement
Service Level Agreement (SLA)
Smart Contract Development Agreement
Third-Party Service Providers Agreement
Token Holder Agreement
Token Listing Agreement with Exchanges
Use cases
Art and Collectibles ($67 billion)
Carbon Credits ($760 billion)
Commercial & Private Aircraft ($126 billion)
Commodities ($14 trillion)
Employee Stock Options
Financial Instruments ($300 trillion)
Insurance Products ($127 billion)
Intellectual Property ($3 trillion)
Loyalty Points & Rewards ($5.5 billion)
Oil Rigs & Energy Infrastructure ($5 trillion)
Private Equity & Venture Capital ($109 trillion)
Real Estate ($326 trillion)
Maritime Vessels ($165 billion)
Resource Rights ($2 billion)
Rockets, Satellites & Space-related Assets ($469 billion)
Sports Teams & Franchises ($512 billion)
Royalties & Revenue Streams ($9.7 billion)
Stressed Assets
Automobiles ($23 billion)
Infrastructure ($4.6 trillion)
Ecosystem
Asset Owners
Custodians
Legal Advisors
Blockchain Developers
Token Issuers
Exchanges & Trading Platforms
Broker-Dealers
Wallet providers
Asset Managers
Market Makers
Data Providers & Analytics Companies
Regulators
Legal compliance
Know Your Customer (KYC)
Anti-Money Laundering (AML)
Countering the Financing of Terrorism (CFT)
Freezing of Funds
Consumer protection
Right-to-be-forgotten Regulations
Data Privacy
Nodes operated by verified entities from FATF-compliant jurisdictions
Regulator nodes for real-time monitoring
8 permissions per address



Download the Tokenization Playbook by Rohas Nagpal
https://www.rohasnagpal.com/docs/Tokenization_Playbook.pdf

Smart Contracts

A smart contract is a **computer program** that automatically executes tasks when specified conditions are met.

Smart contracts:

- enable the automation of digital assets,
- enable trustless, transparent & verifiable transactions,
- streamline processes,
- reduce the need for intermediaries and their associated costs.

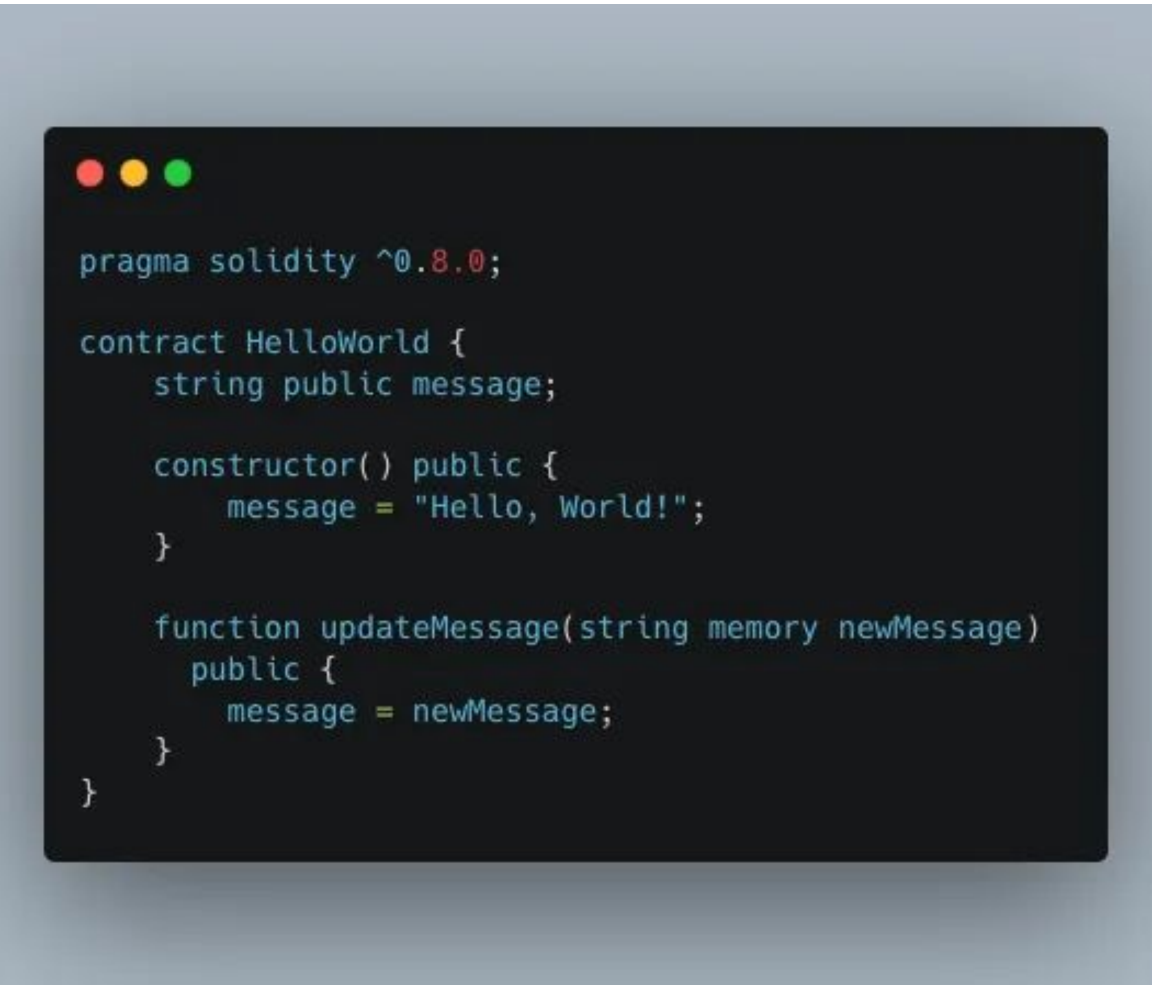
Smart contracts can be used for:

- financial transactions,
- supply chain management,
- digital identity,
- legal agreements.

The most important smart contract languages are Solidity, Vyper, Cairo, and Rust.

Ethereum is the most widely used smart contract blockchain.

Here's the code for a very basic smart contract called "**HelloWorld**".



```
pragma solidity ^0.8.0;

contract HelloWorld {
    string public message;

    constructor() public {
        message = "Hello, World!";
    }

    function updateMessage(string memory newMessage)
      public {
        message = newMessage;
    }
}
```

- It has one string variable called "message" which is public and can be accessed by anyone.

- The contract has a constructor and an "updateMessage" function.

- The constructor function executes automatically when the contract is first deployed. In this case, it sets the initial value of the "message" variable to "Hello, World!".

- The "updateMessage" function allows anyone to change the value of the "message" variable. It takes in a single input, "newMessage", which is the new value that the message variable should be set to.

- This contract can be deployed on Ethereum. Other contracts & external accounts can interact with it via function calls.

To learn more, see these:

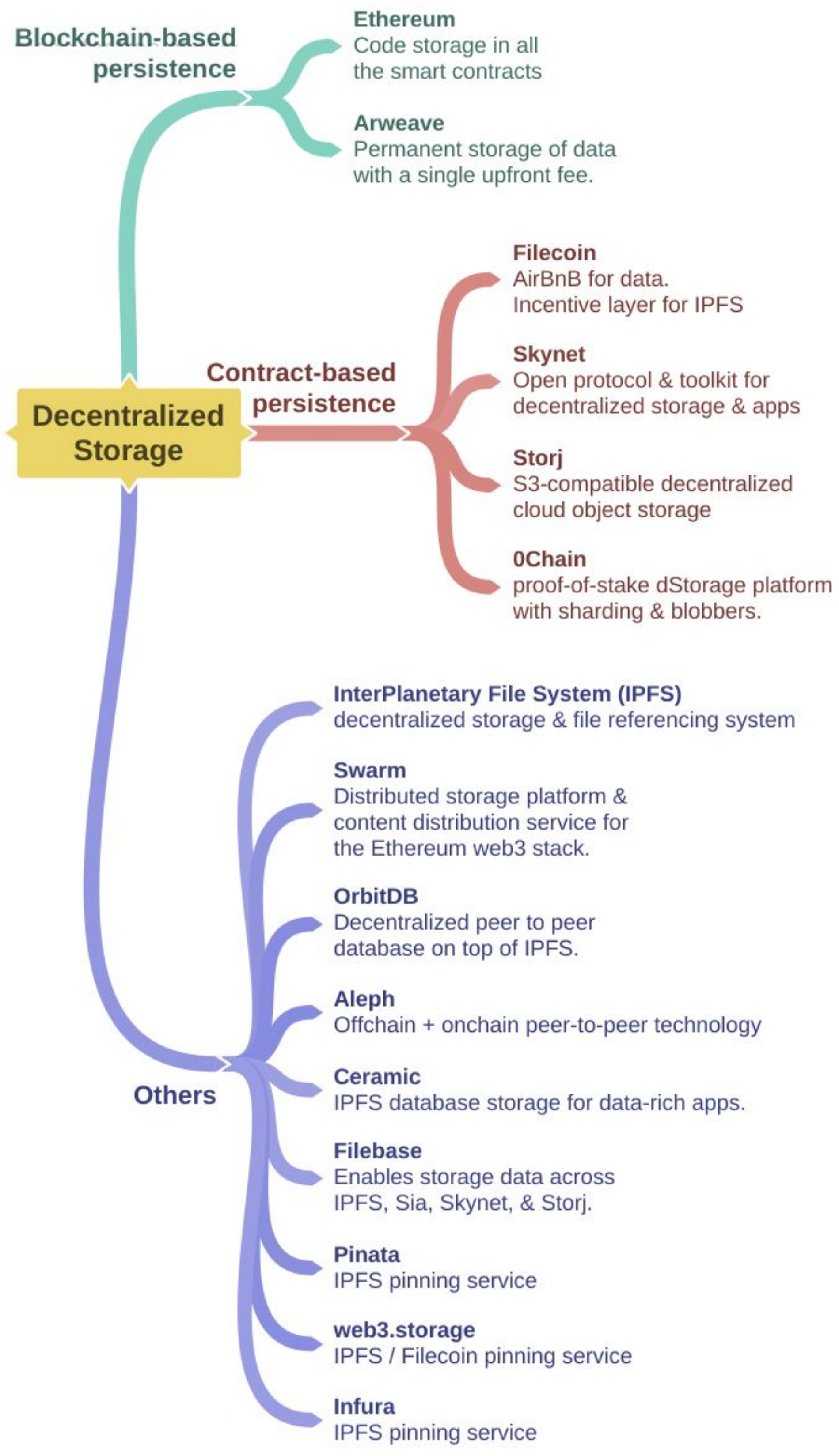
- Getting started with Smart Contracts
- How to write, compile & deploy a simple Solidity smart contract
- How to create an ERC 20 token
- How to create an ERC721 token
- dApp Development frameworks
- How to build smart contracts with Marlowe

Blockchain use-cases

- Transfer of land records / property
- Digital certificates management
- Pharmaceutical supply chain
- e-Notary service
- Blockchain-enabled e-Sign solution
- Farm Insurance
- Identity management
- Duty payments
- Automated customs enforcement and compliance
- Agriculture / farm produce supply chains
- E-Voting
- Smart Grid applications include energy transmission, distribution, trading, and marketing of energy
- Authorized access to relaying in the substation protection system
- Government crypto wallet platform for selling, buying, and trading
- Multiple layer and multiple level access Blockchain-based cloud storage of health test records
- Validation of Bill of Lading in cross-border transport
- Ease of validation of documents at customs at the ports of entry and exit

- Electronic health record management
- Digital evidence management system
- Public service delivery
- Blockchain for social good use cases (charity, donations)
- Metering and settlement
- Payment security mechanism
- Authentication and authorization services
- Automated control of decentralized power
- Smart grid application and grid management
- Microfinance for Self-Help Groups (SHG)
- Customs and trade finance
- Cross border trade
- Renewable energy trading and management
- Insurance underwriting and claims management
- Aggrotech environment
- Micro-financing, financing small businesses or individuals
- Secured logistics document exchange (SLDE)
- Cold chain for supply chain
- National and state highways, toll collection, tracking of public infrastructure
- Blockchain for urban development tracking through Public Private Partnership
- Tracking the progress on climate agreement through Blockchain

- Asset transfer across different government departments
- Digital identities, and verifiable credentials to secure privacy and enable new use cases
- Safe and secure vaccine distribution and administration
- IoT device management and security
- Vehicle lifecycle management
- Chit fund operation and administration


Decentralized Storage
Blockchain-based persistence
Ethereum
Code storage in all the smart contracts
Arweave
Permanent storage of data with a single upfront fee.
Contract-based persistence
Filecoin
AirBnB for data. Incentive layer for IPFS
Skynet
Open protocol & toolkit for decentralized storage & apps
Storj
S3-compatible decentralized cloud object storage
0Chain
proof-of-stake dStorage platform with sharding & blobbers.
Others
InterPlanetary File System (IPFS)
decentralized storage & file referencing system
Swarm
Distributed storage platform & content distribution service for the Ethereum web3 stack.
OrbitDB
Decentralized peer to peer database on top of IPFS.
Aleph
Offchain + onchain peer-to-peer technology
Ceramic
IPFS database storage for data-rich apps.
Filebase
Enables storage data across IPFS, Sia, Skynet, & Storj.
Pinata
IPFS pinning service
web3.storage
IPFS / Filecoin pinning service
Infura
IPFS pinning service

02

Network Security & Privacy

2.1 Blockchain Network Attacks & Vulnerabilities

51% Attack

If a single miner or mining pool controls more than 50% of the network's mining hashrate, they can manipulate the blockchain's consensus mechanism, allowing them to reverse transactions and double-spend coins. This attack is mostly a concern for smaller, less well-established blockchains like Bitcoin SV.

Sybil Attack

In a Sybil attack, an attacker subverts the reputation system of a peer-to-peer network by creating a large number of pseudonymous identities, using them to gain a disproportionately massive influence.

Eclipse Attack

An Eclipse attack is when an attacker controls all of the victim's connections to the network, isolating the victim from the rest of the blockchain network, and feeding them false information.

Routing Attack

By manipulating the routing of information over the internet, an attacker can partition the blockchain network or delay block propagation, increasing the risk of double-spending attacks.

Selfish Mining Attack

In a selfish mining attack, a miner mines a new block and keeps it secret from other miners. If the selfish miner can find a second block before the others find any, they can publish their chain and invalidate the work of the other miners.

Double Spending Attack

This type of attack involves a situation where a user manages to spend their cryptocurrencies more than once by creating multiple copies of the transaction.

Replay Attack

A replay attack involves re-transmitting or delaying a valid data transmission that has been fraudulently repeated or delayed. This might occur during a hard fork if proper security measures are not in place.

Timejacking Attack

This attack involves manipulating the timestamp of a node, which can affect the blockchain's functionality, as block generation relies on accurate timekeeping.

Smart Contract Vulnerabilities

Smart contracts are prone to various types of bugs and vulnerabilities, including re-entrancy attacks, where external contract calls can be hijacked. They are also vulnerable to issues with underflows and overflows.

Phishing Attacks

Similar to traditional phishing attacks, blockchain phishing attempts involve tricking users into giving away their private keys, often through fake websites or social engineering tactics.

Note: Each blockchain protocol will have its own specific potential vulnerabilities depending on its design and architecture. It's important for blockchain engineers to be aware of these possible attacks.

2.2 Node Security Best Practices

Securing a blockchain node is essential to maintain the integrity of the overall network and to protect the information stored within the node.

Here is a comprehensive checklist for best practices in node security:

- ❑ **Regular Software Updates:** Always ensure that the node software and all of its dependencies are up-to-date. This includes the operating system, blockchain software, and any other installed software.

- ❑ **Firewall Configuration:** Use a firewall to limit and control the traffic that your node accepts. The firewall should be configured to allow necessary P2P traffic while blocking unnecessary or potentially harmful traffic.

- ❑ **DDoS Protection:** Protect your node against Distributed Denial-of-Service (DDoS) attacks. This could include rate limiting, IP whitelisting, or using a third-party DDoS protection service.

- ❑ **Secure Communication:** Use secure communication protocols to protect data in transit. For instance, Secure Socket Layer (SSL) / Transport Layer Security (TLS) can be used to encrypt communication.

- ❑ **Role-Based Access Control:** Use Role-Based Access Control (RBAC) to limit who can access what resources. It should be ensured that users and applications only have the permissions that they need and nothing more.

- ❑ **Secure Key Management:** Private keys should be securely stored and never exposed. They should also be backed up in a secure location. Hardware Security Modules (HSMs) or secure key management services can be used for enhanced security.

- ❑ **Regular Auditing:** Regularly audit your node for any signs of irregular or suspicious activity. This includes monitoring log files, checking for unauthorized access, and inspecting outgoing traffic.

- ❑ **Physical Security:** Physical security is as important as digital security. The hardware running the node should be secured against physical tampering.

- ❑ **Redundancy and Backups:** Ensure data redundancy and regular backups. If the node data gets corrupted or lost, having a backup allows for a quick recovery.

- ❑ **Secure APIs:** If your node provides an API, ensure that it's secure. This might include requiring API keys, rate limiting, and regularly auditing API usage.

- ❏ **Patch Management:** Regularly apply security patches to all software. Unpatched software can be a security vulnerability.

- ❏ **Network Segmentation:** Use network segmentation to limit the potential impact of a compromise. Sensitive systems, like a node, should be isolated from less secure parts of the network.

- ❏ **Incident Response Plan:** Have a clear incident response plan in place in case of a security breach. The quicker you can respond to a security incident, the less damage it's likely to cause.

2.3 Network Monitoring Tools

Network monitoring is an essential part of maintaining a healthy and secure blockchain node.

Here are some popular tools used for network monitoring:

Wireshark

This is an open-source protocol analyzer that network professionals use for troubleshooting, analysis, software and protocol development, and education. It allows you to examine data from a live network or from a capture file on disk.

Nagios

Nagios is a powerful, enterprise-grade open-source monitoring system that allows organizations to identify and resolve IT infrastructure problems before they affect critical business processes.

Zabbix

Zabbix is an open-source monitoring tool for networks and applications. It is designed to monitor and track the status of network services, servers, and other network hardware.

Prometheus

This open-source systems monitoring and alerting toolkit can handle multi-dimensional data collection as well as querying and alerting. It's often used in combination with a visualization tool, like Grafana.

Netdata

Netdata is a free, open-source and real-time performance and health monitoring tool with a web front-end. It's useful for tracking performance anomalies.

Ethereum Network Intelligence API

This is a backend-only part of a tool that provides a complete overview of the current state of the Ethereum network, including information about blocks, transactions, and other network activity.

Bitcoin CLI (Command-Line Interface)

For Bitcoin nodes, the built-in command-line interface can provide a wealth of information about the node's operation and the network's status.

IFTOP

It listens to network traffic on named interfaces and shows a table of current bandwidth usage by pairs of hosts.

TCPdump

This command-line utility is a powerful tool for analyzing network behavior and troubleshooting problems.

Elastic Stack (ELK - Elasticsearch, Logstash, Kibana):

This is a set of open-source tools that work together to help you monitor your system by collecting logs from various sources, storing, indexing and visualizing them in a centralized location.

2.4 Privacy Enhancing Technologies

Privacy Enhancing Technologies (PETs) are designed to uphold data minimization principles and allow individuals to retain control and ownership of their personal data.

In the context of blockchain, several technologies can enhance privacy:

Zero-Knowledge Proofs (ZKPs)

ZKPs are cryptographic methods that allow one party (the prover) to prove to another party (the verifier) that they know a value of a specific piece of information without conveying any other information apart from the fact that they know this value.

Zcash, a cryptocurrency, uses ZKPs to maintain the privacy of transactions.

Ring Signatures

A type of digital signature that can be performed by any member of a group of users that each have keys. It's impossible to determine which of the group members' keys was used to produce the signature.

Monero, a privacy-focused cryptocurrency, uses ring signatures to provide transaction privacy.

Coin Mixing or CoinJoin

Services like CoinJoin allow multiple users to combine their transactions into a single transaction, which makes it difficult to determine which inputs (coins sent) correspond to which output (coins received).

Mimblewimble

This blockchain protocol uses confidential transactions and a unique transaction structure to dramatically improve both scalability and privacy. It's used by cryptocurrencies like Grin and Beam.

Homomorphic Encryption

This form of encryption allows computations to be done on encrypted data without decrypting it first. The result of the computation is encrypted and can be decrypted to get the actual result.

Private Information Retrieval (PIR)

PIR protocols allow a user to retrieve an item from a server in possession of a database without revealing which item is retrieved. This is useful for light clients in blockchain networks, which need to retrieve certain parts of the blockchain without revealing their interests.

Secure Multi-Party Computation (sMPC)

sMPC allows a set of parties to compute a function over their inputs while keeping those inputs private. This could be used in a blockchain setting to enable private transactions and smart contracts.

Tor/I2P Networks

While not specific to blockchain, using privacy networks like Tor or I2P can help hide a user's IP address when making transactions or operating a node, thus enhancing privacy.

These technologies can be used individually or combined to provide multiple layers of privacy protection.

03

Node Maintenance & Performance Optimization

3.1 Blockchain Data Storage and Management

In a blockchain network, data storage and management is a crucial aspect of maintaining efficient operations and reliability.

Here's a breakdown of key concepts and strategies involved in blockchain data storage and management:

Blockchain Structure

A blockchain is essentially a linked list where each block contains a group of validated transactions and a reference to the previous block via a hash.

Each transaction in a block is made up of metadata and inputs and outputs.

On-Chain Storage

Storing data directly on the blockchain is known as on-chain storage. This means that the data is stored in the transactions themselves and becomes an immutable part of the blockchain.

This is generally the most secure form of storage but is also the most expensive and least scalable due to the size limitations of a block and the cost of transactions.

Off-Chain Storage

Large scale data storage in blockchain is not feasible due to cost and size restrictions. Hence, only the hash of the data or a reference to the data is stored on-chain, while the actual data is stored off-chain, i.e., outside of the blockchain.

Off-chain data storage could be centralized servers, distributed file systems (like IPFS), or data clouds.

Decentralized Storage Systems

Decentralized storage systems like the InterPlanetary File System (IPFS), Filecoin, or Swarm are used for more efficient and distributed data storage and retrieval.

This type of system is often used in conjunction with a blockchain to create decentralized applications that require file storage.

Sharding

Sharding is a scalability technique where the state of the blockchain is divided into partitions or "shards", each capable of processing its own transactions and smart contracts.

Sharding can potentially allow a blockchain to process many transactions in parallel, improving scalability.

Pruning

Blockchain pruning is a method of limiting a node's storage requirements by discarding older blocks from storage after they've been verified and their transactions have been accounted for.

Pruned nodes keep only the set of transactions that are necessary to maintain a fully validating node (i.e., the blockchain's UTXO set).

State Channels

These are two-way pathways opened between two users that want to make multiple transactions without committing all transactions to the blockchain.

This reduces the strain on the network and reduces fees for the two participants.

Database Management

Depending on the specific implementation, blockchains may utilize traditional database systems (like LevelDB or RocksDB used in Bitcoin and Ethereum) to store and retrieve data efficiently.

Backups

Regular backups of the blockchain data (i.e., the entire ledger or just the latest state) are essential to ensure data availability in case of any faults or data corruption.

A combination of these techniques can be used depending on the specific use case and requirements.

Each technique comes with its own trade-offs in terms of security, cost, scalability, and decentralization.

3.2 Node Performance Metrics

Node performance metrics help in understanding the overall health and efficiency of a blockchain node.

They enable you to identify potential issues and optimize the node for better performance.

Here are some key performance metrics for a blockchain node:

Block Propagation Time

The amount of time it takes for a block to propagate through the network. Faster propagation times generally indicate a healthier network.

Memory Usage

The amount of RAM being used by the node. Excessive memory usage could indicate a memory leak or inefficient memory management.

CPU Usage

The amount of CPU resources being used by the node. Similar to memory usage, high CPU usage could indicate inefficiencies or bugs in the node's software.

Disk Usage

The amount of disk space being used by the node. Blockchain data can take up a significant amount of storage space, especially for full nodes.

Network Latency

The time it takes for a packet of data to get from one designated point to another in the network. Lower latency is better for overall network performance.

Number of Connected Peers

The number of other nodes that your node is connected to. A healthy number of peers is crucial for the robustness and redundancy of the network.

Number of Transactions in the Mempool

The mempool (or memory pool) is a holding area for transactions waiting to be confirmed by the network. A growing mempool could indicate network congestion.

Transaction Verification Time

The time it takes to verify a transaction. Faster verification times improve the overall user experience.

Block Verification Time

The time it takes to verify a block of transactions. Faster block verification times improve the overall throughput of the blockchain network.

Uptime

The amount of time the node has been running without interruption. High uptime is generally desirable, as interruptions can disrupt the functioning of the network and the applications running on it.

Each of these metrics can be monitored using various tools and techniques, including command-line tools, APIs, log analysis, and dedicated monitoring software.

Regular monitoring and optimization based on these metrics can help ensure the smooth operation of your blockchain node.

3.3 Performance Tuning & Optimization

Performance tuning and optimization for a blockchain node involves identifying and resolving issues that hinder its efficiency and reliability.

Here are some strategies:

Hardware Upgrade

Enhancing the node's hardware can significantly improve performance. This could mean adding more memory (RAM), using faster storage (SSD instead of HDD), or upgrading the CPU.

A fast and stable internet connection is also essential.

OS Optimization

The operating system hosting the node can be optimized to better handle the node's demands. This can include tuning network settings, optimizing disk I/O, adjusting CPU scheduling, and ensuring adequate system resources are available.

Database Optimization

The underlying database that the node uses to store blockchain data can often be optimized. This could involve tuning database parameters, indexing, or optimizing database queries.

Node Configuration

Most blockchain nodes have several configuration options that can be tuned for better performance.

These options can control things like memory usage, the number of peer connections, or the amount of data to keep in cache.

Sharding

Sharding is a technique where the state of the blockchain is divided into partitions or "shards", each capable of processing its own transactions and smart contracts.

Sharding can potentially allow a blockchain to process many transactions in parallel, improving scalability.

Off-Chain Transactions

Implementing off-chain transactions or state channels can help scale the blockchain by handling transactions off the blockchain and settling the net result on-chain.

Pruning

Pruning is a method of reducing storage requirements by discarding older blocks from storage after they've been verified and their transactions have been accounted for.

Network Optimization

This involves optimizing the node's connections to its peers to ensure efficient communication. This could involve managing the number of connections, optimizing data transfer protocols, or selecting peers based on their network latency or reliability.

Threading and Parallel Processing

If the node software and the hardware support it, enabling multi-threading or parallel processing can significantly improve the performance of a node.

Regular Monitoring and Updates

Regularly monitor performance metrics to identify potential bottlenecks or issues. Keep the node software up-to-date, as updates often include performance improvements and bug fixes.

Note: Optimization strategies can vary depending on the specific blockchain technology, the role of the node (full node, miner, validator, etc.), and the underlying hardware and network conditions.

Always test changes in a controlled environment before applying them to a production node to understand their impact.

3.4 Backup & Disaster Recovery

Backup and disaster recovery are crucial aspects of maintaining a reliable and resilient blockchain node.

Here's how you might approach this:

Regular Backups

Regularly back up your node's data. This includes the blockchain data, configuration files, and any private keys or wallets associated with the node.

The frequency of the backups will depend on your node's activity and your tolerance for data loss.

Offsite Storage

Store backups in a different location from your primary node. This protects against physical disasters like fires, floods, or hardware failures.

Secure Storage

Protect your backups from unauthorized access. This might involve encryption, access controls, or physically secure storage locations.

Redundancy

Consider setting up redundant nodes that can take over if your primary node fails. This can be part of a load balancing setup, which can also improve performance.

Testing

Regularly test your backup and recovery procedures to make sure they work as expected. This might involve setting up a test node and trying to restore it from a backup.

Disaster Recovery Plan

Have a disaster recovery plan that outlines what to do in various disaster scenarios. This plan should be reviewed and updated regularly.

Failover Mechanisms

Consider setting up automatic failover mechanisms that can detect when your primary node is down and switch over to a backup node.

Backup Software

Use reliable backup software that can automate the backup process, handle encryption, and ensure data integrity.

Documentation

Document your backup and recovery procedures, and make sure that the relevant people are trained and familiar with these procedures.

Node Software Backups

Besides the blockchain data, keep copies of the node software or docker images. In case of software corruption, having these backups will be useful.

Note: Remember, the goal of backup and disaster recovery is not just to protect against data loss, but also to minimize downtime and ensure the smooth operation of your node and the services that depend on it.

The specific strategies you choose will depend on the specific requirements of your node, your network, and your organization.

04

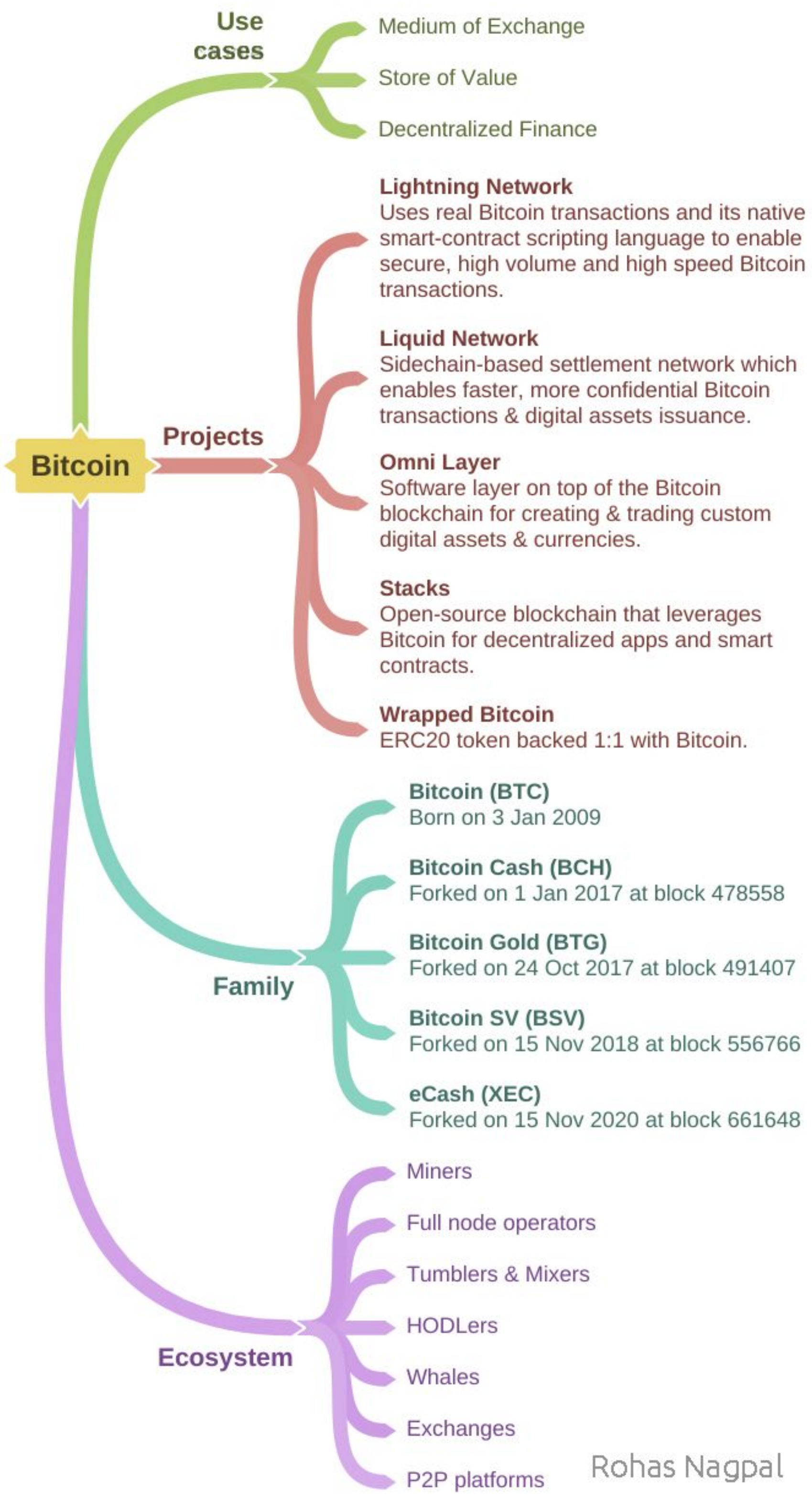
Bitcoin

If blockchain technology were a religion, the Bitcoin whitepaper would be its Bible.

The Bitcoin whitepaper is the **foundational document** that started the Blockchain revolution. As a Blockchain Engineer, you MUST have a deep understanding of it.

The video below covers Digital Signatures, Hash-based Proof-of-Work, Double-spending, Transactions, Simplified Payment Verification, Combining & Splitting Value, and Privacy.

https://www.youtube.com/watch?v=3BBRF1XgjTE&t=75s


Bitcoin
Use cases
Medium of Exchange
Store of Value
Decentralized Finance
Projects
Lightning Network
Uses real Bitcoin transactions and its native smart-contract scripting language to enable secure, high volume and high speed Bitcoin transactions.
Liquid Network
Sidechain-based settlement network which enables faster, more confidential Bitcoin transactions & digital assets issuance.
Omni Layer
Software layer on top of the Bitcoin blockchain for creating & trading custom digital assets & currencies.
Stacks
Open-source blockchain that leverages Bitcoin for decentralized apps and smart contracts.
Wrapped Bitcoin
ERC20 token backed 1:1 with Bitcoin.
Family
Bitcoin (BTC)
Born on 3 Jan 2009
Bitcoin Cash (BCH)
Forked on 1 Jan 2017 at block 478558
Bitcoin Gold (BTG)
Forked on 24 Oct 2017 at block 491407
Bitcoin SV (BSV)
Forked on 15 Nov 2018 at block 556766
eCash (XEC)
Forked on 15 Nov 2020 at block 661648
Ecosystem
Miners
Full node operators
Tumblers & Mixers
HODLers
Whales
Exchanges
P2P platforms
Rohas Nagpal

05

Ethereum

Ethereum is NOT a blockchain. It's NOT a cryptocurrency either! It's actually a protocol (a set of rules or procedures) like "HTTP" or "HTTPS".

Multiple independent blockchains run on the Ethereum protocol:

- **Ethereum Mainnet**: This is the production network.

- **Görli**: This is a Proof of Authority testnet.

- **Private networks**: Development networks, and Consortium networks.

- **Layer 2 testnets**: Arbitrum Rinkeby, and Optimistic Kovan.

When most people talk about Ethereum, they are talking about **Mainnet** - the primary public Ethereum production blockchain.

This is where actual-value transactions occur on the blockchain. The native crypto of this Ethereum is **Ether** (ETH).

Ethereum Virtual Machine (EVM)

One of the greatest blockchain innovations is the Ethereum Virtual Machine (EVM).

EVM is "the environment in which all 'Ethereum' accounts and smart contracts live". Smart contracts are programs that run automatically when some predefined conditions are met.

The sole purpose of the Ethereum protocol is to keep "the continuous, uninterrupted, and immutable operation" of the EVM. At any given block, Ethereum has only one "canonical" or unique state.

EVM defines the rules for computing new valid states from one block to another.

EVM exists as a single entity maintained by a large number of connected computers (nodes) running an Ethereum client e.g. Geth or OpenEthereum.

A client is a software that enables nodes to read blocks on the blockchain and smart contracts.

For details, see:
https://ethereum.org/en/developers/docs/evm

Ethereum standards

The most popular Ethereum standards are:

ERC-20

This is a standard API for tokens within smart contracts. It provides basic functionality to transfer tokens and allow tokens to be approved and be spent by another on-chain third party.

For details, see:
https://ethereum.org/en/developers/docs/standards/tokens/erc-20

ERC-721

This is a standard for non-fungible tokens (NFT). ERC-721 tokens are unique and can have a different value as compared to another token from the same smart contract - due to age, rarity, image, etc.

For details, see:
https://ethereum.org/en/developers/docs/standards/tokens/erc-721

ERC-777

This is a fungible token standard that improves ERC-20.

For details, see:
https://ethereum.org/en/developers/docs/standards/tokens/erc-777

ERC-1155

This is a multi-token standard. A single contract can include a combination of tokens - fungible, non-fungible, semi-fungible, etc. An ERC-1155 token can perform the functions of ERC-20 and ERC-721 tokens with more efficiency.

For details, see:
https://ethereum.org/en/developers/docs/standards/tokens/erc-1155

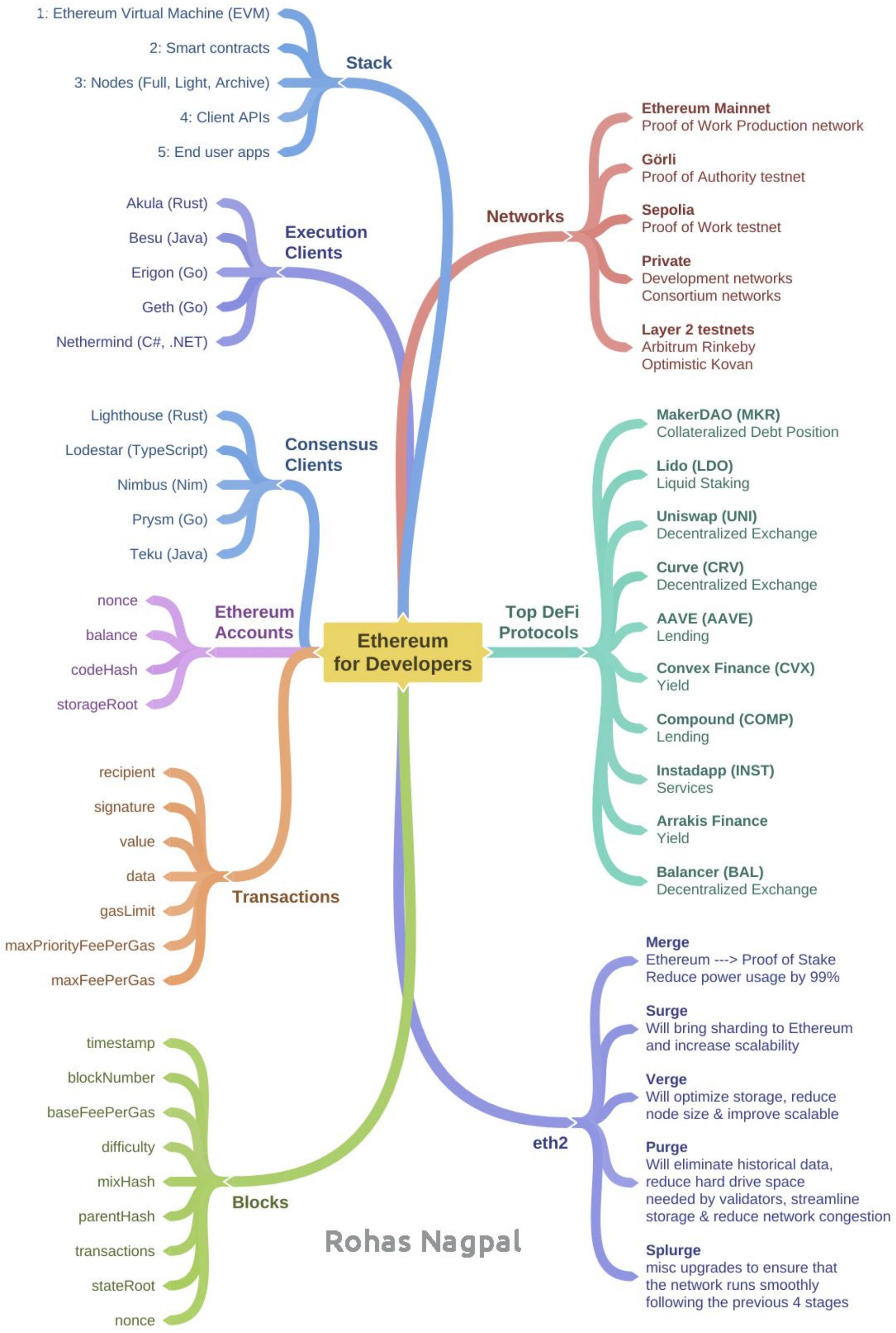
ERC-4626

This is a standard API for tokenized yield-bearing vaults which represent shares of a single underlying ERC-20 token.

For details, see:
https://ethereum.org/en/developers/docs/standards/tokens/erc-4626

Others
For a detailed list of Ethereum Token Standards, see:
https://github.com/PhABC/ethereum-token-standards-list

For a deep dive into Ethereum, see:
https://blockchainblog.substack.com/p/ethereum-for-developers


Ethereum for Developers
Stack
1: Ethereum Virtual Machine (EVM)
2: Smart contracts
3: Nodes (Full, Light, Archive)
4: Client APIs
5: End user apps
Execution Clients
Akula (Rust)
Besu (Java)
Erigon (Go)
Geth (Go)
Nethermind (C#, .NET)
Consensus Clients
Lighthouse (Rust)
Lodestar (TypeScript)
Nimbus (Nim)
Prysm (Go)
Teku (Java)
Ethereum Accounts
nonce
balance
codeHash
storageRoot
Transactions
recipient
signature
value
data
gasLimit
maxPriorityFeePerGas
maxFeePerGas
Blocks
timestamp
blockNumber
baseFeePerGas
difficulty
mixHash
parentHash
transactions
stateRoot
nonce
Networks
Ethereum Mainnet
Proof of Work Production network
Görli
Proof of Authority testnet
Sepolia
Proof of Work testnet
Private
Development networks
Consortium networks
Layer 2 testnets
Arbitrum Rinkeby
Optimistic Kovan
Top DeFi Protocols
MakerDAO (MKR)
Collateralized Debt Position
Lido (LDO)
Liquid Staking
Uniswap (UNI)
Decentralized Exchange
Curve (CRV)
Decentralized Exchange
AAVE (AAVE)
Lending
Convex Finance (CVX)
Yield
Compound (COMP)
Lending
Instadapp (INST)
Services
Arrakis Finance
Yield
Balancer (BAL)
Decentralized Exchange
eth2
Merge
Ethereum ---> Proof of Stake
Reduce power usage by 99%
Surge
Will bring sharding to Ethereum
and increase scalability
Verge
Will optimize storage, reduce
node size & improve scalable
Purge
Will eliminate historical data,
reduce hard drive space
needed by validators, streamline
storage & reduce network congestion
Splurge
misc upgrades to ensure that
the network runs smoothly
following the previous 4 stages
Rohas Nagpal

06

Multichain

Multichain Practical Activities

1. Setting up a Linux server
2. Creating a permissioned blockchain
3. Creating custodial addresses
4. Creating non-custodial addresses
5. Creating a consensual governance model
6. Creating and verifying digital signatures
7. Issuing and re-issuing assets
8. Atomic Exchange Transactions

Links

Use these instructions to set up the blockchain:

- https://www.multichain.com/download-community
- https://www.multichain.com/developers/creating-connecting

Multichain documentation
https://www.multichain.com/developers/json-rpc-api

Atomic Exchange Transactions
https://www.multichain.com/developers/atomic-exchange-transactions

07

Hyperledger

Hyperledger is an open-source initiative for growing enterprise use of blockchain technologies.

Hyperledger projects include:

- 5 Distributed Ledgers
- 4 Libraries
- 6 Tools

Distributed Ledgers

Distributed Ledgers are **multiparty databases** with no central trusted authorities.

Hyperledger has **5 Distributed Ledgers**:

1. Besu
2. Fabric
3. Indy
4. Iroha
5. Sawtooth.

Libraries

Libraries are code bases that add functionality to enterprise blockchain use cases e.g. digital credentials, smart contracts & cryptographic code.

Hyperledger has **4 Libraries**:

1. AnonCreds
2. Aries
3. Transact
4. Ursa

Tools

Tools improve interoperability, performance & security in enterprise blockchains.

Hyperledger has **6 Tools** -

1. Bevel
2. Cacti
3. Caliper
4. Cello
5. Firefly
6. Solang

HYPERLEDGER BESU

Hyperledger Besu

Hyperledger Besu is an Ethereum client that is optimized for use in enterprise environments and can be used for both public and private permissioned networks.

- It includes several consensus algorithms & permissioning schemes.
- It can run on various test networks.
- It is flexible & modular, with an extractable EVM implementation.
- It is suitable for consortium environments.

Hyperledger Fabric

Hyperledger Fabric is a modular blockchain platform with plug-and-play components such as consensus & membership services.

It can be used to develop applications for a broad range of enterprise use cases.

It enables high performance & preserves privacy.

Hyperledger Indy

Hyperledger Indy is a set of tools, libraries & reusable components for creating interoperable digital identities.

Indy enables the decentralization of identity.

Indy can be used standalone or integrated with other blockchain systems.

HYPERLEDGER
IROHA

Hyperledger Iroha

Hyperledger Iroha is a blockchain platform designed for simplicity & ease of integration into infrastructural or IoT projects.

It features a simple construction, modular, domain-driven C++ design.

Its focus is on client application development.

It has a crash fault-tolerant consensus algorithm called YAC.

HYPERLEDGER SAWTOOTH

Hyperledger Sawtooth

Hyperledger Sawtooth is a flexible & modular blockchain platform that separates the core system from the application domain.

It allows smart contracts to define business rules for applications without needing knowledge of the underlying core system design.

It supports multiple consensus algorithms such as Practical Byzantine Fault Tolerance (PBFT) and Proof of Elapsed Time (PoET).

HYPERLEDGER ANONCREDS

AnonCreds

Hyperledger AnonCreds is a type of verifiable credential that utilizes zero-knowledge proof cryptography to support advanced privacy-protecting capabilities.

HYPERLEDGER
ARIES

Aries

Hyperledger Aries is a toolkit for creating, transmitting & storing verifiable digital credentials in a p2p blockchain-rooted environment.

HYPERLEDGER TRANSACT

Transact

Hyperledger Transact provides a standard interface for executing smart contracts, separate from the underlying ledger implementation. It supports multiple smart contract engines.

HYPERLEDGER
URSA

Ursa

Hyperledger Ursa is a shared cryptographic library that provides a centralized repository for cryptographic code & interfaces.

HYPERLEDGER
BEVEL

Bevel

Hyperledger Bevel is an accelerator for deploying production-ready distributed networks across public & private cloud providers.

Cacti

Hyperledger Cacti is a blockchain integration tool enabling the integration of different blockchains.

HYPERLEDGER
CALIPER

Caliper

Hyperledger Caliper is a blockchain benchmark tool for measuring the performance of a blockchain implementation with a set of predefined use cases.

HYPERLEDGER
CELLO

Cello

Hyperledger Cello is an operational console for managing blockchains running on bare-metal, virtual machine & container platforms. It can also be used for creating Blockchain as a Service.

HYPERLEDGER
FIREFLY

Firefly

Hyperledger FireFly is a complete stack for building & scaling secure Web3 applications.

HYPERLEDGER
SOLANG

Solang

Hyperledger Solang compiles Solidity for Solana & Substrate.

08

ChatGPT Super Prompt Templates

ChatGPT Super Prompt Templates

Here are some ChatGPT super prompt templates that you can customize and use:

1. Troubleshooting smart contract code

I am currently developing a smart contract and require your expertise in troubleshooting some issues I have encountered. The purpose of the smart contract is [briefly describe the purpose or functionality], and it is built on the [specify the platform, e.g., Ethereum, Binance Smart Chain] platform. Here is the relevant code snippet where I am facing difficulties: [provide the code snippet(s)]. My concerns encompass functionality, where I have observed [describe problems or unexpected behavior]; security, with suspected vulnerabilities such as [mention security concerns]; gas consumption, particularly in relation to [explain concerns about excessive gas usage or optimization needs]; and code readability, specifically in areas like [highlight areas where the code is difficult to understand or maintain]. Given this information, identify the root cause of these issues and propose suitable solutions or optimizations. Additionally, provide any general best practices or recommendations for enhancing my smart contract.

2. Building scalable decentralized applications (dApps)

In need of guidance for building scalable decentralized applications (dApps), the following information is provided: the dApp's purpose [briefly describe the purpose or functionality], the target platform [specify the platform, e.g., Ethereum, Binance Smart Chain], and the intended user base [describe the target users or market]. Additionally, the dApp must address challenges such as [list specific scalability concerns or constraints]. Considering this context, please offer expert insights into the architecture, technologies, and best practices for developing a scalable dApp, addressing aspects like efficient smart contract design, off-chain data management, layer 2 scaling solutions, and other pertinent factors that contribute to the dApp's performance, security, and user experience.

3. Advising on token design & implementation

Deliver strategic and well-structured advice on token design and implementation for a [specific blockchain platform or project], tailored to [target audience, e.g., developers, project managers, or stakeholders], incorporating the following key elements: clear token utility and purpose, [specific token design principles or best practices], relevant and context-appropriate token distribution and governance models, and effective evaluation methods to assess the impact and benefits of the token design on the overall ecosystem and user experience.

4. Suggesting improvements to existing blockchain systems

Provide a comprehensive and innovative improvement proposal for an existing blockchain system [specify the blockchain], designed for [target audience, e.g., developers, stakeholders, or users], incorporating the following key elements: clear problem identification, [specific improvement technique or strategy], relevant and context-appropriate use cases, and effective evaluation methods to assess the impact and benefits of the proposed improvements.

5. Providing recommendations for tooling and development frameworks

Offer insightful and practical recommendations for tooling and development frameworks in the context of [specific technology or programming language], designed for [target audience, e.g., developers, project managers, or stakeholders], incorporating the following key elements: clear requirements analysis, [specific evaluation criteria or comparison methodology], relevant and industry-appropriate tools and frameworks, and effective implementation strategies to ensure seamless integration and adoption within the development process.

6. Choosing blockchain consensus mechanisms

Present a comprehensive and well-reasoned guide for choosing the most suitable blockchain consensus mechanism for a [specific blockchain project or use case], designed for [target audience, e.g., developers, project managers, or stakeholders], incorporating the following key elements: clear understanding of different consensus mechanisms and their trade-offs, [specific evaluation criteria or decision-making framework], relevant and context-appropriate comparisons of existing consensus mechanisms, and effective implementation strategies to ensure seamless integration and optimal performance of the chosen consensus mechanism within the blockchain system.

7. Best practices for building secure smart contracts

Deliver an exhaustive and actionable guide on best practices for building secure smart contracts on [specific blockchain platform or programming language], tailored for [target audience, e.g., developers, project managers, or stakeholders], incorporating the following key elements: clear understanding of common security vulnerabilities and risks, [specific secure coding techniques or principles], relevant and context-appropriate tools and frameworks for smart contract security analysis, and effective testing and auditing methodologies to ensure the robustness and reliability of the smart contract code against potential attacks and exploits.

8. Implementation of blockchain-based payment systems

Provide a detailed and practical blueprint for implementing a blockchain-based payment system within [specific business or industry context], designed for [target audience, e.g., developers, project managers, or stakeholders], incorporating the following key elements: clear understanding of payment system requirements and regulations, [specific blockchain platform or technology], relevant and context-appropriate transaction processing and settlement mechanisms, and effective security and scalability measures to ensure a robust, efficient, and user-friendly blockchain-based payment system.

9. Building decentralized finance (DeFi) protocols

Provide an in-depth and comprehensive guide for building decentralized finance (DeFi) protocols on [specific blockchain platform], designed for [target audience, e.g., developers, project managers, or stakeholders], incorporating the following key elements: clear understanding of DeFi concepts and financial primitives, [specific DeFi protocol design principles or best practices], relevant and context-appropriate smart contract templates and governance models, and effective risk management and security measures to ensure the stability, interoperability, and user trust in the developed DeFi protocol.

10. Design & implementation of blockchain-based gaming systems

Present a detailed and engaging blueprint for the design and implementation of blockchain-based gaming systems within [specific gaming genre or platform], tailored for [target audience, e.g., developers, project managers, or stakeholders], incorporating the following key elements: clear understanding of gaming mechanics and player incentives, [specific blockchain technology or tokenization strategy], relevant and context-appropriate integration of decentralized assets and game economy, and effective performance optimization and security measures to ensure an enjoyable, fair, and immersive gaming experience for users.

11. Securing an Ethereum Node

Prepare a detailed checklist with relevant commands and code for securing an Ethereum node. Include physical security measures, operating system security, application security, system administrator security, account security, monitoring procedures, and encryption methods. The checklist must be comprehensive and based on the best practices, tools, and techniques to secure each area to protect the server from unauthorized access, data breaches, and other security threats.

12. Simulating a Bitcoin node

I want you to act as a Bitcoin full node using Bitcoin Core. I will type commands and you will reply with what the node should show. I want you to only reply with the terminal output inside one unique code block, and nothing else. Do not write explanations. Do not type commands unless I instruct you to do so. When I need to tell you something in English I will do so by putting text inside curly brackets {like this}. My first command is bitcoin-cli getnewaddress.

09

Interview Questions

- Can you explain the basic structure and operation of a blockchain node?
- What are some common consensus mechanisms used in blockchain systems? Can you discuss the pros and cons of each?
- What is a blockchain bridge and what is its role in interoperability between different blockchains?
- Could you explain the concept of merged mining?
- Can you differentiate between hard and soft forks in a blockchain?
- What are the different types of blockchain (public, private, consortium, etc.) and in what scenarios might each type be used?
- What are blockchain wallets and addresses? How do they work?
- Can you discuss some common use cases for blockchain technology?
- Can you discuss some common vulnerabilities in blockchain networks and how to protect against them?
- What are some best practices for maintaining the security of a blockchain node?

- What tools would you use to monitor the performance and security of a blockchain network?
- Can you name and describe some privacy-enhancing technologies used in blockchain?
- What are some key considerations for blockchain data storage and management?
- What metrics would you use to assess the performance of a blockchain node?
- How would you approach performance tuning and optimization for a blockchain node?
- What are some best practices for backing up blockchain data and ensuring disaster recovery?
- What are the steps involved in setting up and managing a Bitcoin node?
- How does setting up and managing an Ethereum node differ from managing a Bitcoin node?
- What is the process of setting up a private blockchain using Multichain?

10

Quiz Questions

BEP-1. Which of these Blockchain performance indicators represents the transactions per second that a consensus algorithm can process?

A. Fault tolerance threshold
B. Latency / Finality
C. Scalability
D. Throughput

BEP-2. Which of these represents an upper bound of faulty nodes that directly impacts the performance of the consensus algorithm?

A. Fault tolerance threshold
B. Latency / Finality
C. Scalability
D. Throughput

BEP-3. Which of these Blockchain performance indicators represents the time it takes for a transaction to become settled in the ledger?

A. Fault tolerance threshold
B. Latency / Finality
C. Scalability
D. Throughput

BEP-4. Which of these Blockchain performance indicators represents the ability for a network to expand without degrading performance?

A. Fault tolerance threshold
B. Latency / Finality
C. Scalability
D. Throughput

BEP-5. Which of these enables developers to create custom blockchains?

A. Layer-0 Blockchains
B. Layer-1 Blockchains
C. Layer-2 Blockchains

BEP-6. Which of these are open-source software that enable developers to create custom blockchains?

A. Blockchain Frameworks
B. Cosmos & Horizen

BEP-7. Which of these validate & execute transactions without the need for any external network?

A. Layer-0 Blockchains
B. Layer-1 Blockchains
C. Layer-2 Blockchains

BEP-8. Which type of blockchain nodes enable faster and cheaper transactions?

A. Cold nodes
B. Light nodes
C. Lightning nodes

BEP-9. Smart contracts are the environment in which the Ethereum Virtual Machine (EVM) lives.

A. True
B. False

BEP-10. Which of these runs the blockchain's software to validate & store the complete history of transactions on the network?

A. Nodes
B. Smart contracts

BEP-11. Which of these nodes hosts the entire blockchain, validates blocks & maintains consensus?

A. Archival Full Nodes
B. Pruned Full Node

BEP-12. In which type of Blockchain fork does each node need to upgrade its software to be compatible with the new processes?

A. Hard Fork
B. Soft Fork

BEP-13. Which of these blockchain nodes saves download time & storage space by only downloading block headers?

A. Light nodes
B. Lightning nodes

BEP-14. Which of these blockchain nodes are used for signing transactions offline and storing private keys away from the network?

A. Cold nodes
B. Lightning nodes

BEP-15. Which of these are "sidechains" built on top of Layer-1 blockchains?

A. Blockchain Frameworks
B. Layer-2 Blockchains

BEP-16. Which of these consists of the hardware, software, and networks that enable the functioning of a blockchain?

A. Infrastructure Layer
B. User Interface Layer

BEP-17. Which of these consists of the interfaces, apps & services that enable users to interact with the blockchain and access its functionality?

A. Application Layer
B. User Interface Layer

BEP-18. In which type of Blockchain Bridge do you remain in control of your cryptos?

A. Trusted
B. Trustless

BEP-19. Which of these Blockchain nodes holds only the most recent transactions up to the size limit set by the operator?

A. Archival Full Nodes
B. Pruned Full Node

BEP-20. Which of these enables fungible Ethereum tokens to be re-used by other applications such as wallets and decentralized exchanges?

A. ERC-20
B. ERC-721

BEP-21. Which of these describes a method for initially locking tokens within a token contract and slowly dispensing them using a Proof of Work algorithm?

A. ERC-918
B. ERC-1178

BEP-22. Ether (ETH) is NOT an ERC20 token.

A. True
B. False

BEP-23. Which standard improves on ERC-20 and enables operators to send tokens on behalf of other address—contracts?

A. ERC-10
B. ERC-777

BEP-24. Which standard outlines a smart contract interface that can represent any number of fungible and non-fungible token types?

A. ERC-1203
B. ERC-1155

BEP-25. Which standard enables multi-class fungible tokens within smart contracts?

A. ERC-1178
B. ERC-1203

11

Quiz Answers

BEP-1: D	BEP-2: A	BEP-3: B
BEP-4: C	BEP-5: A	BEP-6: A
BEP-7: B	BEP-8: C	BEP-9: B
BEP-10: A	BEP-11: A	BEP-12: A
BEP-13: A	BEP-14: A	BEP-15: B
BEP-16: A	BEP-17: B	BEP-18: B
BEP-19: B	BEP-20: A	BEP-21: A
BEP-22: A	BEP-23: B	BEP-24: B
BEP-25: A		

About the Author

Rohas Nagpal

Chief Blockchain Architect, Hybrid Finance Blockchain (HYFI)

Rohas Nagpal, began his career as a hacker in the early 1990s,

In 1999, he co-founded the Asian School of Cyber Laws, dedicating 16 years to cyber investigation and cyber law.

During this time, he also assisted the **Government of India** in framing draft rules under the *Information Technology Act*.

His work spanned **18 countries**, investigating cybercrimes & data breaches for hundreds of organizations across sectors like aerospace, banking, law enforcement, pharma & shipping.

He also authored the **Cyber Crime Investigation Manual***,* hailed as a "bible for cybercrime investigators" by the *Times of India*.

In 2016, he co-founded BankChain, a **community of 37 banks** + IBM, Microsoft, and Intel. The core product was a self-building blockchain ecosystem with a web app, PWA & Blockchain REST API service.

He has also been a consultant for the **Reserve Bank Innovation Hub** for preparing a Whitepaper on Non-Fungible Tokens (NFT) and Central Bank Digital Currency (CBDC).

He writes a weekly newsletter for **Mint**, advises several Blockchain startups, and has developed the **Blockchain Token Valuator.**

He also conducts the **free Blockchain Engineering Program.**

He can be contacted at *rohas@hyfiblockchain.com*

Credits

- Cover image: https://www.freepik.com
- https://blockchainblog.substack.com
- https://ethereum.org
- https://bitcoin.org
- https://blog.coinbase.com/a-simple-guide-to-the-web3-developer-stack-8364b612d69c
- https://alchemy.com/blog/web3-stack
- https://en.bitcoin.it
- https://www.multichain.com