rubix

Technical Whitepaper Version 1.11

July 23, 2025

https://github.com/rubixchain/rubixgoplatform

Rubix Whitepaper

Abstract

The Rubix Tokenchain protocol is a deterministic state-machine that is designed to address the scale, cost, and privacy shortcomings of blockchain protocols that rely on one sequentially organized chain (monolithic) of all global transactions. The protocol divides the global state machine into a large, but finite number of state-machines called Tokenchains. While each Tokenchain maintains one state, together all Tokenchains represent a globally accessible singleton state that is immutable. This paper explains various components that make up the protocol. Blockchain protocols in general achieve a globally accessible singleton state by organizing all global transactions sequentially as blocks (each block having a finite number of transactions) and organizing the blocks as a hashchain. Such blockchain protocols require exhaustive mining-based Proof-of-Work (PoW) consensus algorithms to secure the state, which results in high latency, low throughput, and high transaction costs. While the Proof-of-Stake (PoS) consensus protocols may alleviate the energy & throughput issues associated with the PoW consensus, PoS protocols suffer from concentration (nodes with higher existing stakes may continue to gain larger voting power), security ("nothing-at-stake" & impersonation risks). Further, both PoW & PoS protocols require every node to store the entire global state, which results in significant storage & computational inefficiencies. In contrast to the sequential transaction architecture of blockchains, Rubix Tokenchain processes transactions in an asynchronous parallel manner. Each transaction achieves finality on its own without waiting to be pooled with unrelated transactions.

Tokenchain

The Rubix global state will be made of at least 51.4 million Tokenchains. Each Tokenchain is bound by one unique utility token (alternatively described as a digital utility tool). An Tokenchain OT is made of all transactions that use token Tn to confirm. All transactions within an Tokenchain OT are validated individually and sequentially. However, transactions of different Tokenchains are validated asynchronously and parallelly

Token Architecture in Rubix

Every transaction on the Rubix Network is intrinsically linked to one or more **native utility tokens**, forming the economic and security backbone of the protocol. Rubix supports two types of tokens:

1. **Utility Tokens**
2. **Asset Tokens**

1. Utility Tokens

Utility tokens in Rubix are **fungible** and serve as the backbone for transaction validation, token binding and network incentives. There are two types of utility tokens:

- **Primary Utility Token (RBT):**
  The core token of the Rubix network, capped at approximately **51.4 million RBT**. These tokens are used to:
  - Native token in Rubix network
  - Secure and validate transactions
  - Bind asset tokens to Tokenchains
  - Participate in mining via proof credits
  - Enable decentralized services and governance

Of the total RBT supply, **56 tokens were pre-minted** to facilitate bootstrapping. The rest are **mined by validators** through a proof credit mechanism; no energy intensive mining or large-scale staking is required. Even basic computing nodes can act as validators.

- **Secondary Utility Tokens:**
  These are **project or domain specific fungible tokens** issued by entities operating within the Rubix ecosystem. They can be used to:
  - Represent internal economies (e.g., credits, loyalty points)
  - Power microchains or subnet-specific applications
  - Facilitate governance or metering for specific services

All secondary utility tokens are interoperable with the Rubix protocol and must still anchor their transactions via RBT to participate in core consensus.

2. Asset Tokens

Asset tokens are **non-fungible tokens (NFTs)** that represent **unique digital or real-world assets**. They are indivisible, non-interchangeable and serve as verifiable representations of ownership or rights.

Asset tokens may represent:

- Digital-native items (e.g., coupons, certificates, licenses, collectibles)
- Tokenized real-world objects (e.g., land parcels, equity shares, carbon credits)

Key characteristics:

- **Has own Tokenchain mapping state transitions**
- **Carry dynamic value**, based on the utility tokens used in their last transaction
- **Do not have inherent value** until involved in a transaction with attached utility tokens

RBTs: Digital Utility Tools for the Decentralized Economy

The Rubix Network is designed to **revolutionize how identity, data, and business processes** are initiated, settled and recorded across industries. At the heart of this transformation lies the **Rubix utility token (RBT)** a **digital utility tool** that powers productivity across decentralized applications and ecosystems.

RBTs as Business Tools

Unlike traditional crypto tokens or software licenses, **RBTs function as reusable, transferable, and tradeable digital utility units** that businesses can:

- **Purchase** for long-term use (CapEx)
- **Borrow or lease** for short-term tasks (OpEx)
- **Earn** by participating as validators and securing the network (Shared Utility via Security)

This flexible model aligns RBTs with traditional factors of production such as **hardware, land, or energy** rather than legacy software tools, which are typically:

- Non-transferable
- Non-resaleable
- Locked behind licenses
- Lacking secondary market value

Rubix disrupts this status quo. **RBTs are perpetual, programmable, and carry residual value**, making them **true digital capital assets** within a decentralized productivity stack

RBTs Are Not Securities or Currencies

Importantly, **RBTs are not securities, tokens of speculation or cryptocurrencies.** They are **digital utilities**, akin to:

- **Land or infrastructure:** usable, rentable, or tradable
- **Tools and machines:** productive instruments with measurable output
- **Software primitives:** composable resources that can be owned, shared, and built upon

Businesses can hold RBTs in their balance sheet as functional assets, transfer them peer-to-peer or exchange them on compliant marketplaces

Verifiability via IPFS Integration

Rubix ensures **perpetual traceability and tamper-resistance** of all RBT tokens using a **customized version of the Interplanetary File System (IPFS)**:

- Each RBT token is stored using **multi-hash format**
- The hashes are pushed to IPFS and **committed immutably**
- Since IPFS uses **content-based addressing**, even a **single-bit alteration** changes the hash immediately flagging any invalid or forged token state

Nodes and Decentralized Identity (DID) in Rubix

Rubix nodes are physical **participants** in the decentralized infrastructure. They distribute responsibilities across:

- **Performing transactions of tokens**
- **Storage of proofs and Tokenchains**
- **Validation and computation for consensus**
- **Pledging of RBT tokens for security and mining eligibility**

Decentralized Identity (DID)

Each Rubix node is identified by a **Decentralized Identity (DID) a self-generated, cryptographically verifiable public key** that anchors the node's presence and authority in the network.

Key characteristics of Rubix DID:

- Generated using the **ECDSA P-256 elliptic curve**
- **Self-issued** and does not rely on any central certificate authority

- Public key acts as the node's **permanent, verifiable identifier**
- **Shared with peers** across the network to enable secure validation and communication

Rubix Transactions

Rubix enables **peer-to-peer transactions** between nodes in a decentralized and parallelized architecture. All nodes that join the Rubix network participate in the exchange of value or services via **cryptographically signed digital contracts**, forming the backbone of Rubix's programmable economy.

Types of Transactions

A Rubix transaction can represent:

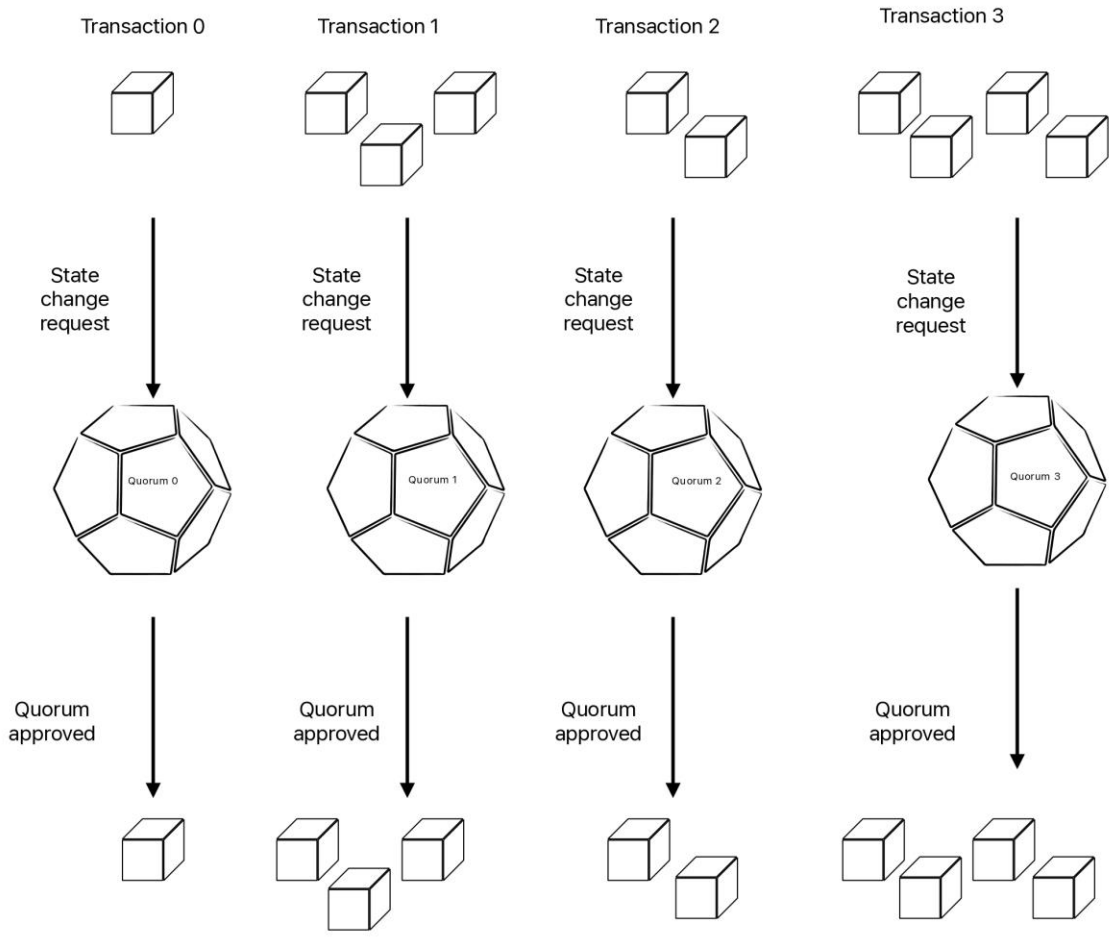
- **(a)** A **barter-style contract**: an exchange of goods or services for other goods or services
- **(b)** A **monetary contract**: an exchange of goods or services for a **medium of value** (e.g., RBT tokens, stablecoins or other tokens)
- **(c)** A **simple token transfer**: native RBT tokens transferred from one node to another

All transactions are encoded, signed, and broadcast via the Rubix network layer, leveraging Rubix's **Tokenchain architecture** for parallel, traceable processing.

Parallel Processing with Tokenchains

At any given time, multiple transactions may be initiated across the network. The Rubix protocol is designed for **parallel processing** of these transactions, with **no global mempool or block queue**, unless:

- Transactions involve a **shared peer (sender or receiver)**, in which case dependencies must be resolved in order
- The **same token (RBT)** is used in multiple transactions, requiring consensus within its associated Tokenchain



Rubix introduces a **token-centric approach** to transaction validation and ledger storage, wherein **every transaction must be anchored by at least one RBT utility token**. This architecture enables scalable, parallelized execution without centralized coordination.

Routing Logic

- If a **single RBT token** is used, the transaction is added to the corresponding **tokenchain**
- If **multiple RBT tokens** are used, the transaction is added to **multiple Tokenchains:** one for each RBT token involved

This routing logic ensures that **Tokenchains operate independently**, except in cases of shared token or participant involvement enabling Rubix to process **thousands of transactions in parallel**.

In the case of **asset tokens** (non-fungible representations of real-world or digital assets):

- An **RBT token is indirectly locked** at the time of **asset token deployment**
- This **locked RBT acts as the anchor** for the asset token's provenance and transactional traceability
- Even if the asset token is transferred multiple times, its value and verifiability remain bound to the **RBT token(s)**

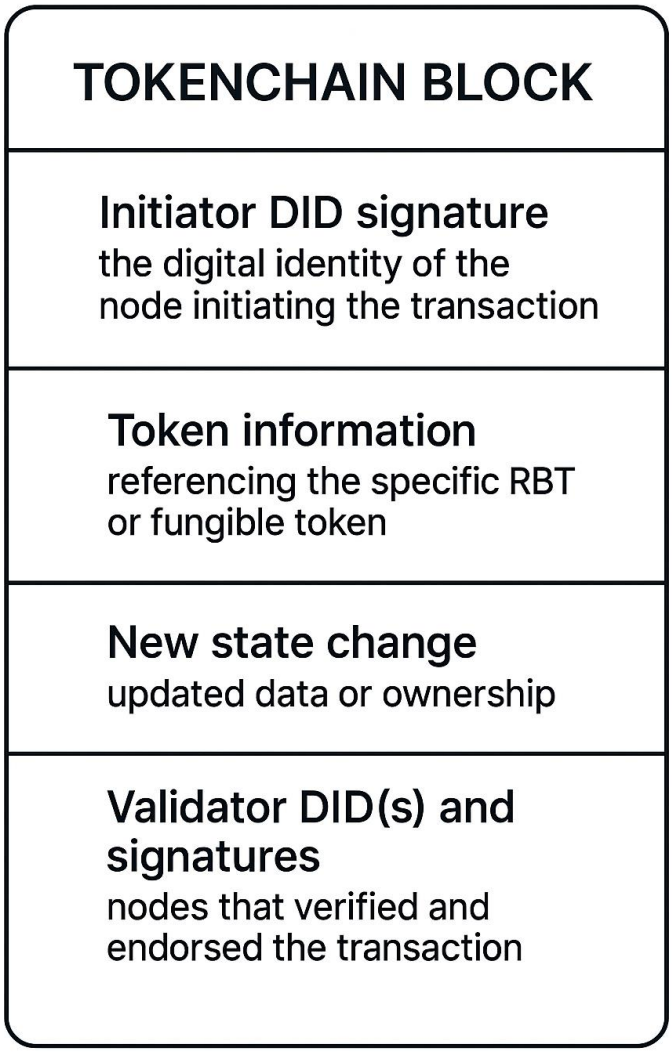
Tokenchain: Token-Bound State Chains in Rubix

An **Tokenchain** in Rubix is a **dedicated transaction history**, bound to a single token. Each Tokenchain acts as a **chronological record of state transitions** for the token from its point of origin to its current state.

Structure of an Tokenchain

Each Tokenchain entry (i.e., transaction or state change) captures:

- **Initiator DID signature**: the digital identity of the node initiating the transaction
- **Token information**: referencing the specific RBT or fungible token
- **New state change:** updated data or ownership
- **Validator DID(s) and signatures**: nodes that verified and endorsed the transaction
- **Timestamp**: capturing the precise event time



This structure ensures **non-repudiation, traceability, and finality** for each token's history.

Tokenchains for Fungible Tokens

Rubix supports **secondary fungible tokens** anchored to the RBT system.

1. RBT-Based Fungible Tokens

- A **fixed RBT token is locked** at genesis to anchor the fungible token class
- The genesis block stores the **reference to the locked RBT**, which becomes the backing utility unit

- These fungible tokens behave like **"fractionalized RBTs"**, inheriting their traceability and verifiability

2. Non-RBT Fungible Tokens (e.g., Stablecoins, STOs)

- At deployment, the **genesis block must include the RBT-equivalent value**
- This RBT value is **used as a reference** to maintain one-to-one pledging integrity

This integration makes every token in the network **individually verifiable**, **immutable**, and **tamper-evident ensuring** Rubix's integrity and trustless operation without centralized oversight.

Zero Gas Fee Architecture in Rubix

Rubix is built on a fundamentally different architecture from traditional blockchains, enabling **zero gas fees for transactions**. Instead of relying on a global chain with miners or validators competing for block space, Rubix uses a **token-centric, parallelized Tokenchain model** that removes the need for per-transaction fees entirely

Key Design Principles Behind Zero Gas

1. **Token-Bound Execution (Tokenchain Model)**
    a. Every transaction in Rubix is bound to one or more **RBT utility tokens**
    b. The transaction is recorded directly on the **Tokenchain associated with that token**
    c. There is **no global block space** or queue to compete for, removing the economic incentive for gas-based bidding
2. **Validator Pledging Instead of Fee-Based Rewards**
    a. Validators **do not earn transaction fees**
    b. Instead, they **pledge RBT tokens** to validate transactions and secure Tokenchains
    c. They earn **proof credits**, which convert into newly mined RBT tokens over time a **mining model based on contribution, not congestion**

Economic Incentive Shift: From Pay-to-Use to Earn-by-Securing

Rubix flips the blockchain economic model:

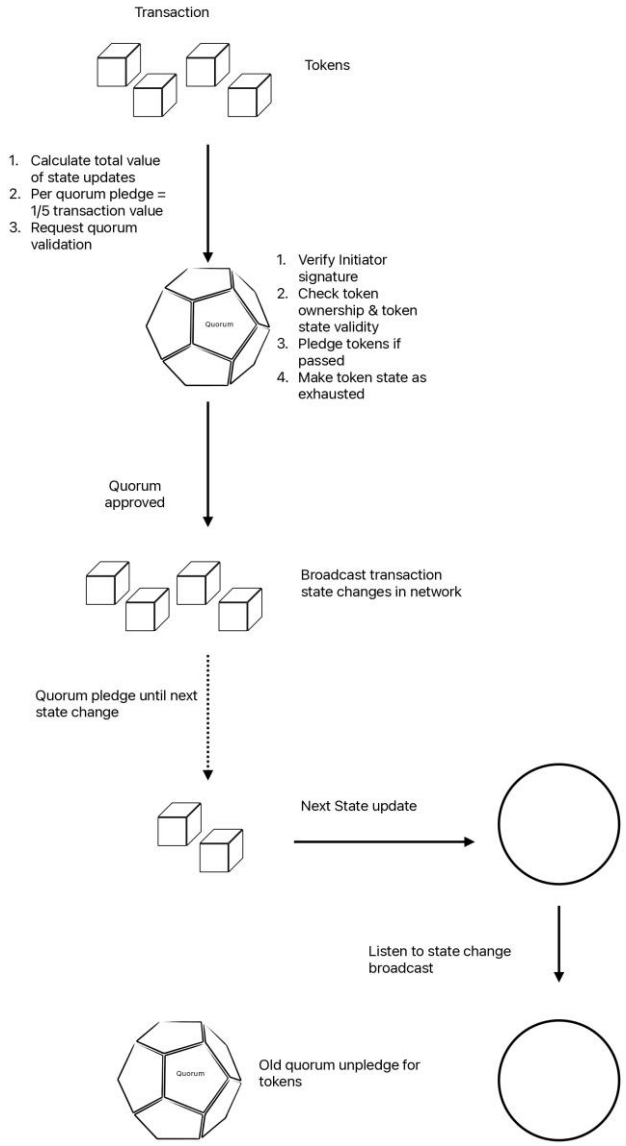
Traditional Chains	Rubix Network

Users **pay gas** to execute transactions	Users **pay nothing**; validators secure transactions via pledged tokens
Validators **compete** for fees	Validators **earn credits** based on contribution
Gas fees scale with network load	**No congestion pricing**; scalable by design

Proof of Pledge (PoP): A Lightweight and Distributed Validator Consensus Model

Rubix introduces **Proof of Pledge (PoP)** , a novel, eco-friendly consensus mechanism where **every Rubix node is eligible to become a validator (miner)** and **any node may be selected** as a validator through a decentralized selection mechanism.

Unlike traditional Proof of Work (PoW) or Proof of Stake (PoS) that depends on concentrated hash power and excessive energy consumption or large amount of stake to validate, PoP ensures:



- **Decentralized and egalitarian mining**
- **Low carbon footprint and energy efficiency**
- **Token issuance driven by real participation, not power centralization**

Feature	Proof of Stake (PoS)	Proof of Pledge (PoP)
Staking Scope	Global	Local to each Tokenchain
Reward Model	Continuous rewards based on stake size	Credit-based rewards tied to validator participation

Decentralization	Tends to concentrate over time	Inherently far more distributed
Security Risk	Higher ; central validator keys are attack vectors	Lower ; no single point of failure; validators are distributed
Stake Lock-in	Long-term staking often required	Pledge can be revoked if replaced by another validator

Formal Security Proof Considerations for Proof of Pledge (PoP)

While the Proof of Pledge (PoP) model introduces a resource-efficient and egalitarian consensus mechanism, it is essential to formalize its resilience against adversarial conditions to strengthen adoption in high-assurance environments. This section outlines the formal characteristics of the PoP model with respect to common distributed systems security assumptions.

1. Byzantine Fault Tolerance (BFT) Analysis

Each transaction in Rubix is validated by a quorum of **7 validators**, which ensures finality at the Tokenchain level. This structure enables localized fault domains, and validation proceeds independently across tokens.

According to the Byzantine fault tolerance model, a system can tolerate up to f faulty nodes in a group of n validators, where $n \geq 3f + 1$. For Rubix:

- n = 7 validators
- Tolerates up to f = 2 byzantine (malicious or faulty) validators
- Ensures both **liveness** (transactions continue processing) and **safety** (no invalid state commitment)

Validator groups are selected via a **Pseudo-Random Function (PRF)**, making group prediction infeasible in advance. This adds another layer of probabilistic resistance to collusion or targeted validator bribery.

2. Sybil Resistance

Rubix defends against Sybil attacks, where a single entity spins up many fake identities to gain influence using cryptoeconomic constraints:

- **Token-based validator pledging**: Becoming a validator requires pledging Rubix Blockchain Tokens (RBTs). Creating multiple validator identities incurs

proportional economic cost, making large-scale Sybil attacks expensive and impractical.

- **Credit-based mining eligibility**: Even after pledging, a node must **accumulate proof credits over time** to mine tokens. These credits are earned through consistent, honest validation work. Sybil identities that do not participate meaningfully accrue no credits and receive no economic gain.

Together, these mechanisms ensure that any Sybil attack would:

- Require substantial RBT capital upfront
- Involve long-term behaviour without immediate rewards
- Risk pledge slashing if dishonest activity is detected

The architecture's localized consensus, probabilistic validator selection, and economic deterrents provide a multilayered defense model against both **collusion** and **mass identity generation**, positioning Rubix as a high-integrity, low-trust-assumption protocol suitable for mission-critical and enterprise-grade deployments.

3. State Machine Modeling

Each Tokenchain in Rubix can be modeled as an independent **deterministic state machine** that handles its own state transitions securely. This abstraction allows for formal reasoning about correctness, safety and liveness.

Tokenchain State Machine:

- **States**: Idle → Transaction_Pending → Validated → Finalized
- **Events**: Transaction initiation, validator pledge, signature collection, PBFT consensus, State change of Token finalised
- **Transitions**:
    - Idle → Transaction_Pending: When a transaction is initiated by a node with valid DID
    - Transaction_Pending → Validated: When 5 validator signatures are collected
    - Validated → Finalized: When the transaction is immutably written into the Tokenchain and broadcast

Properties Satisfied:

- **Safety**: A transaction, once finalized, cannot be reverted or modified
- **Liveness**: Every initiated transaction eventually reaches a final state

- **Determinism**: Given a token's state and a valid transaction, the resulting state is unique and predictable

4. Validator Incentive Integrity (Game Theoretic Analysis)

To ensure that Rubix remains secure under rational validator behavior, it is important to demonstrate that **honest validation** is a **Nash equilibrium;** that is, the dominant strategy for any validator assuming others behave honestly.

We define a basic utility function for a validator as:

$U(v) = f(C) - g(P)$

Where:

- U(v) is the utility function of validator v
- f(C) is the value of accumulated **proof credits** over time (reward)
- g(P) is the **penalty function** applied to the pledged RBTs in case of dishonest or inactive behavior (risk)

This incentive structure means:

- Honest validators **earn credits** at a predictable rate, which can be redeemed for RBTs.
- Malicious validators **risk slashing** of pledged RBTs and lose access to future credits.
- Inactive validators earn no rewards and may be replaced in subsequent PRF validator cycles.

Hence, under standard assumptions:

- If all other validators follow protocol, a rational validator maximizes long-term utility by being honest.
- Any deviation from protocol incurs potential slashing (g(P) increases) and forfeits future credit accumulation (f(C) drops).

This establishes a **Nash equilibrium** around protocol-compliant behaviour.

Future formalizations may use **mechanism design theory** to model long-term validator participation across diverse network conditions, including fluctuating token prices, credit inflation/decay, and subnetwork-specific incentive structures.

This game-theoretic robustness ensures that Rubix incentivizes not just decentralized participation but **economically rational honesty**, enabling long-term trust without centralized enforcement.

Validator Role and Network Integrity

Rubix is purpose-built to support **decentralized applications (dApps)** that drive **real-world commercial transactions** at global scale. In this framework

- **Securing the network by pledging RBTs and participating in consensus**
- **Storing proof objects and transaction data across the Tokenchains**
- **Enhancing fault tolerance and recovery**
- **Preventing network forks**

Validators earn utility tokens (RBTs) based on the **proofs they store** and the **pledge they commit,** collectively referred to as the **Proof of Pledge**

Conclusion: Scalable, Secure, and Democratic

Rubix's consensus model offers the best of both worlds:

- **Scalability** through parallelized Tokenchains
- **Security** through token-backed pledges
- **Decentralization** through localized validator assignments
- **Sustainability** via energy-efficient operation with no heavy compute requirements

Proof of Pledge positions Rubix as a **next-generation consensus layer**—designed not just for chain security, but for equitable, verifiable, and decentralized real-world application enablement.

Validator Selection in Rubix

The Rubix network supports **two modes of validator selection**, depending on whether nodes operate in the **open public Rubix network** or within a **private subnet (microchain)**. This dual-mode structure ensures flexibility, decentralization, and domain-level autonomy.

Type 1: Open Rubix Network (Global Validators)

In the open Rubix network, validator selection is **probabilistic and decentralized**, using a **Pseudo-Random Function (PRF)** to ensure fairness and unpredictability.

- A PRF is executed to **select a candidate set of validators**
- Validators must **possess sufficient pledgeable RBT tokens** to qualify
- If a selected candidate lacks enough pledge, **a new PRF round is triggered**
- The process continues until a **group of 7 validators** is finalized who meet the **minimum pledge requirements**

Type 2: Subnet/Microchain (Private Validator Domains)

Within Rubix subnets (also known as **microchains**), validator selection can be **deterministically defined** by the subnet participants. These rules may:

- **Bypass the PRF process**
- Use **pre-approved validator lists**, **round-robin logic** or **domain-specific reputation models**
- Be **hard-coded or dynamically governed** by smart contracts or DAO-like mechanisms

Credit Accrual per Transaction

Rubix incentivizes validators through a **credit-based reward system**, which forms the foundation for mining new RBT tokens. The number of credits earned depends on the **type of network (Type 1 or Type 2)**, the **nature of the token pledged**, and the **duration of the pledge**

Type 1 Validators (Open Rubix Network)

In the public Rubix network:

- Validators earn **1 credit per week** for every **1 RBT (or equivalent)** pledged and used to secure a transaction
- Credits are accumulated as long as the validator's pledge remains active and verifiable
- These credits contribute toward reaching the threshold required to **mine a new RBT token**

Example:
If a validator pledge 5 RBT tokens and maintains them for 2 weeks, they will earn **10 credits**.

Type 2 Validators (Subnets / Microchains)

In subnets or private microchains, where validators are selected deterministically:

- Validators earn reduced credits compared to Type 1
- The rate depends on the **type of token** used for pledging:

Token Type	Weekly Credit Earned
RBT or RBT-based tokens	**1/15 credit per token per week**
Non-RBT tokens (e.g. stablecoins, STOs)	**1/30 credit per token per week**

This reduced rate reflects the **lower systemic risk** and **domain-specific scope** of subnet validators; while still ensuring they can accumulate credits for mining over time.

Security Against Common Blockchain Attacks

1. Fault Tolerance and Token Recovery

Rubix ensures fault tolerance and token recovery through a layered mechanism that balances scalability, economic incentives and verifiable accountability. This system operates in two tiers—**optimistic** and **pessimistic**—to guarantee token availability and resilience against validator failure.

Level 1: Optimistic Fault Tolerance

In the default operational mode, Rubix adopts an **optimistic fault tolerance** approach. Each transaction is validated by a group of seven validator nodes. These validators are responsible for maintaining the state of the token and its associated tokenchain in the network until the next state transition occurs. As a result, token ownership and history remain accessible to the token holder, even if they lose access to their local node or storage.

This design inherently supports **token recovery**, enabling any legitimate token owner to resynchronize their token state from the network at any time. Validators are rewarded

with **credits** for serving token data when requested, aligning availability with economic incentives.

Rubix's sharded approach—where each validator only stores and maintains tokens for transactions it has directly validated—leads to **massive reductions in storage, computational load and required pledge amounts**. This structure facilitates high validator participation and promotes a broader, decentralized validator base.

Level 2: Pessimistic Enforcement and Economic Accountability

To guard against validator inaction or data loss, Rubix introduces a **pessimistic enforcement mechanism**. Validators are economically bonded, with pledge tokens staked as collateral to ensure honest behaviour and persistent data availability.

If a validator fails to maintain access to token state or attempts to go offline prematurely, **challenger nodes** in the network can submit cryptographic proofs to dispute the validator's behavior. A **valid challenge** results in:

- **Slashing of the validator's pledge tokens**, redistributed in part to the challenger as a reward for upholding network integrity.
- **Loss of credit accumulation** for the affected transaction, disincentivizing downtime or incomplete servicing.

This dual-layer structure ensures both proactive data availability (via incentives) and reactive enforcement (via penalties), creating a self-correcting, decentralized system. The combination of lightweight sharded validation and cryptoeconomic accountability makes Rubix highly fault-tolerant without sacrificing decentralization or efficiency.

2.Distributed Denial-of-Service (DDoS)

Regarding DDoS (Distributed Denial-of-Service) attacks, Rubix is architected with **extreme decentralization** and **horizontal scalability**. The network supports **billions of fully validating nodes**, at least 1 of which is required to get network back to full state. As a result, a **coordinated global DoS attack is nearly impossible**, due to:

1. **Lack of economic incentive:** attacking the network yields no viable reward;
2. **Massive resource requirements:** launching an attack would firstly require an impractically large number of independently operating and staked nodes. Followed by an even larger DDoS on all challenger nodes.

3. 51% / Majority Attack

In traditional blockchains, control over 51% of mining power or stake can enable rewriting transaction history. Since Rubix has no global ledger and each token has its own independent Tokenchain, an attacker would need to simultaneously compromise a massive number of chains and validator sets—rendering the attack computationally and economically impractical.

4. Front-Running & MEV (Miner Extractable Value)

Front-running is a form of MEV (Maximal Extractable Value) where a malicious validator or observer sees a pending transaction and inserts their own transaction just before it; typically to gain profit in trading, minting or auction scenarios. Front-running **incentivizes manipulation**, penalizes regular users and turns open networks into **rigged games**.

Rubix's architecture **eliminates front running by design**, not just by deterrence or monitoring. Here's how:

- **No mempool:** Transactions are sent directly to validators; not publicly visible before confirmation.
- **No transaction fees:** There's no way to pay more to jump the queue.
- **Token-bound Tokenchains :** Each transaction is processed independently, no global ordering.
- **Private IPFS communication:** All transaction data is end-to-end encrypted no leaks, no sniffing.
- **No block builders:** There's no single party assembling a block who can manipulate order.

Mining: Conversion of Proof Credits into RBT Tokens

In the Rubix Network, mining refers to the conversion of accumulated **proof credits** into new **RBT tokens**. This process reinforces the network's security by incentivizing validators to actively participate in transaction verification and storage of proofs.

Each validator node, upon verifying a transaction, **stores the corresponding proofs** to maintain the integrity of the **Tokenchain**. Validators contribute to the network by:

- **Pledging RBT tokens** to signal honest participation and safeguard against malicious behaviour.
- **Committing memory resources** to persistently store proof objects,

As a reward for this contribution, validators **earn proof credits**. Once a validator accumulates sufficient proof credits beyond a defined **threshold (τ)**, they become eligible to **mine a new RBT token**. This mining event is itself **recorded as a new transaction** on the Rubix Network and undergoes **consensus** like any other network activity.

To ensure long-term sustainability and prevent centralization, the mining difficulty increases over time:

- **Initial thresholds (τ_0)** are deliberately set low to bootstrap early validators.
- **Subsequent thresholds** rise gradually, making future RBT mining incrementally more challenging.
- The current set of mining difficulty levels and required proof credits are provided in **Annexure 1**.

Mine NFT Tokenchain

Mining is tracked using a **native NFT object** called the **Mine NFT**, which acts as a verifiable ledger of all mined RBT tokens. This Tokenchain maintains:

- A **chronological log** of every mining event,
- The **proof credits consumed** during each conversion,
- The **validator node identity** (via Rubix DID) that performed the mining.

This structure ensures transparency, traceability, and fairness across the Rubix ecosystem. The **Mine NFT** chain is publicly available and cryptographically secure, enabling any participant to verify the mining history and circulating supply of RBT tokens.

To reinforce network stability and discourage speculative behaviour, newly minted RBT tokens are **non-transferable for 4 weeks** from the time of mining

Pledging for Mining: Securing the Conversion of Proof Credits

The **conversion of proof credits into RBT tokens** in the Rubix network is treated as a standard transaction. However, to enhance security and ensure responsible minting, **additional pledging requirements** are imposed during the mining process.

Genesis Pledge for New Token Minting

When a node initiates the minting of a new RBT token using its accumulated proof credits:

- The transaction must be signed by **validator quorums**
- This quorum must **collectively pledge 1 RBT token**
- The pledged token is **locked for 4 weeks, during this time the mined token also cannot be transferred**

This **longer pledge mechanism** introduces a high-integrity security layer that:

- Prevents misuse or rapid flipping of newly minted tokens
- Distributes accountability across multiple validators

Tokenomics of the Rubix Network

The Rubix Network introduces a novel, utility-based token architecture centered around **RBT (Rubix Blockchain Token)**, designed to power decentralized applications, secure the network, and align stakeholder incentives through a sustainable and transparent economic model. This section outlines the supply, distribution, use cases, and incentive mechanisms of RBT and its associated token classes.

1. Token Overview

Rubix supports three main token categories, each serving distinct functional roles within the ecosystem:

Token Type	Symbol	Supply Cap	Fungibility	Role
Primary Utility Token	RBT	51.4 million	Fungible	Transaction validation, pledge, governance, staking
Secondary Utility Tokens	Custom	Fixed/Variable	Fungible	Domain-specific credits, loyalty, subnet metering
Asset Tokens	Custom	Infinite	Non-Fungible	Digital/physical asset representation

2. RBT Distribution and Minting

- **Initial Supply**: 56 pre-mined RBT (for bootstrapping and genesis validators)
- **Remaining Supply**: ~51.4 million minted over time via Proof of Pledge (PoP)
- **Mining Mechanism**:
    - Validators pledge RBT to secure transactions
    - Earn **proof credits** based on active participation
    - Accumulate credits to mint new RBTs
- **Mining Controls**:
    - **Difficulty Increases Over Time**
    - Early mining = lower credit thresholds
    - Each mined token requires an **additional pledge lock** for 4 weeks

3. Token Velocity and Lockups

- **4-week lock** on newly minted RBTs (non-transferable period)
- Pledged RBTs are **locked for validation duration**
- Tokens used in long-term subnet validations may have **custom lock periods**
- Low token velocity = reduced speculative churn, increased stability

5. Deflationary or Scarcity Mechanisms

- **Capped supply**: No more than 51.4 million RBTs
- **Slashing**: Validators engaging in dishonest behaviour lose pledged RBTs
- **Locking**: Temporary lock of RBTs in smart contracts and secondary tokens

6. Validator Incentives and Credit Mining

- **Credit Earning**:
    - Type 1 Validators (open Rubix network): 1 credit per 1 RBT pledged per week
    - Type 2 Validators (private subnets): 1/15 or 1/30 credit per pledged token per week depending on token type
- **Mining Eligibility**:
    - Upon crossing the credit threshold τ, validators may mint 1 RBT
    - A pledge of 1 RBT must be locked for each mint, aligning reward with responsibility
- **Mining Transparency**:
    - All RBT minting events are recorded on a **Mine NFT Tokenchain**

- Each entry includes validator DID, proof credit consumed, and timestamp

Smart Contracts

Smart contracts represent sophisticated business logic encapsulated in machine-readable formats, typically articulated through programming languages. These contracts are executed within a network's nodes, operating in a deterministic, sandboxed environment. Rubix smart contracts are treated as a specialized form of Non-Fungible Tokens (NFTs) possessing a dynamic state. Every invocation of a contract function leads to an update in this state, which is meticulously recorded and preserved on the Contract tokenChain. This dedicated chain furnishes an immutable ledger, ensuring transparent and tamper-proof documentation of each contract execution. To ensure versatility and adaptability, Rubix smart contracts are crafted in prevalent web2 languages, including Rust, JavaScript, and GoLang. These contracts are subsequently executed within a WebAssembly (WASM) environment.

Smart contract Life Cycle

Rubix protocol is committed to improve adaptability of blockchain technology. Along with its revolutionary proof of pledge protocol aided by zero gas fee transactions, Rubix focuses on making dApp deployment and execution easier for our ecosystem. With WebAssembly(WASM) based smart contracts, existing web2 codebases and developers can migrate their codebase and knowledge into Rubix with ease. WebAssembly (WASM) is a binary instruction format that allows code to be executed at near-native speed in a safe, sandboxed and deterministic manner across different platforms. Smart contracts can be written in languages that compile to WebAssembly, such as Rust and C/C++, and then executed on a blockchain platform that supports WASM. Here are the steps in executing WASM in Rubix

1. Write: Smart contracts are written in high-level programming languages like Rust, Golang or C/C++. These languages offer the flexibility and expressive power of high-level languages while compiling down to WebAssembly bytecode.
2. Compile: Once the smart contract code is written, it is compiled to WebAssembly bytecode. Compilers like rustc for Rust or Emscripten for C/C++ can be used to generate WebAssembly binaries from the source code.

3. Deploy: The compiled WebAssembly code is then deployed onto a blockchain platform. The contract code, along with any necessary metadata, is stored on the blockchain.
4. Execute: When a user or another contract interacts with the deployed smart contract, the contract's functions are called via transactions. These transactions contain input data that specifies which function of the contract to execute and with what parameters.
5. Validate: The transaction is validated by the blockchain nodes to ensure it follows the rules of the blockchain protocol. Once validated, the transaction and the associated smart contract function call are processed by the nodes 6. State Change and Output: Smart contracts can read data from the blockchain's state and modify it as per the logic defined in their functions. Smart contracts can also produce output data, which is typically returned to the caller after the contract function execution is complete. The smart contract is executed on the DApp side. The DApp should have an api endpoint which must be passed as a parameter to register-callbackurl api. This api registers the api endpoint in the node. Once the endpoint is registered, each time an execution happens on the object chain, as per the logic in the smart contract deployed, the states in each of the subscribed nodes get updated.

Once deployed, the logic of a smart contract, represented by its WebAssembly bytecode, is immutable. This means it cannot be changed. If you need to update the contract's logic, a new version of the contract needs to be deployed. To learn more about smart contracts and APIs to interact with Rubixchain ,visit https://learn.rubix.net/smartcontract/

Governance Lifecycle & DAO Design

The Rubix Network is engineered for long-term decentralization, adaptability and equitable participation. Its governance architecture enables stakeholders to propose, vote and enact protocol-level and subnet-specific changes in a transparent, tamper-resistant, and verifiable manner. Governance on Rubix evolves in two tiers: **global governance** (for protocol-wide upgrades) and **subnet/local governance** (for microchain or domain-specific network decision-making).

Rubix governance follows a structured lifecycle to ensure community-driven upgrades and transparent decision-making.

Stage	Description

Proposal	Any RBT holder (meeting the threshold) can submit a governance proposal. This may include protocol upgrades, economic parameter changes, validator rules, or DAO transitions.
Review	Validators and stakeholders review the proposal. Optional third-party audits or DAO discussions can be initiated.
Voting	Proposal enters a fixed duration voting window (e.g. 14 days). Eligible RBT holders cast votes using their wallet or smart contract interfaces.
Quorum & Threshold	A proposal must meet minimum participation and pass threshold (majority or supermajority, depending on impact) to be ratified. Majority: 51% SuperMajority : 67%
Execution	Approved proposals are executed by automated code or governance executor group
Audit & Record	All governance actions are immutably recorded on the Rubix and broadcasted for public verifiability.

Rubix supports **subnet/microchain governance**, allowing domain-specific ecosystems to define and enforce their own rules.

Environmental Impact & Sustainability

Rubix Network introduces a new path: a **low-energy, high-efficiency consensus framework** called **Proof of Pledge (PoP)**, purpose-built to minimize environmental impact while maximizing network participation and economic utility.

Key sustainability features include:

- **Zero energy-based mining:** no nonce guessing or block races
- **Token bound chain:** avoid global state replication
- **Validator sharding:** reduces per-node storage and power usage
- **Mining via proof credits:** aligned with participation, not energy burn

IPFS in the Rubix Network

The **InterPlanetary File System (IPFS)** is a foundational component in Rubix network, providing a **decentralized, content-addressed, and tamper-proof storage layer**. It is tightly integrated into Rubix's architecture to support verifiability, privacy and efficient data propagation.

Key Roles of IPFS in Rubix

Rubix leverages several core properties of IPFS to enable secure and scalable decentralized operations:

1. **Immutable Object Storage**
   Every file or transaction object added to IPFS is stored as an **immutable data structure**, ensuring historical integrity and non-repudiation.
2. **Content-Based Addressing via Cryptographic Hashing**
   Files and transaction objects are addressed by their **cryptographic hash**, not by location.
   a. Any **change in content**, even a single character, results in a **completely different hash**
   b. This property enables **verifiability and integrity checks** across the Rubix network
3. **Distributed Hash Table (DHT)**
   IPFS uses a **DHT-based lookup mechanism** to:
   a. **Locate content** across peers
   b. **Announce new content** availability
   c. Enable **efficient peer discovery and routing**
   This is critical for Rubix's **validator selection**, **proof propagation**, and **Tokenchain synchronization**
4. **Redundancy Elimination & Version Control**
   IPFS naturally **removes redundant data** and supports **versioned objects**, which:
   a. Reduces storage bloat across nodes
   b. Allows **auditable history** of asset token or Tokenchain changes
5. **Private IPFS Swarm for Rubix Network**
   Rubix nodes operate within a **private IPFS network (swarm)**. This ensures:
   a. Controlled participation
   b. Encrypted, peer-to-peer data exchange
   c. **Isolation from public IPFS** to maintain network-level privacy and consensus integrity
6. **Commitment for Double-Spend Prevention**
   All transactions and tokens in Rubix are **committed to IPFS** by their content hash.
   a. Since each hash is **globally unique**, it is impossible to **duplicate or forge** transaction history
   b. This acts as a **built-in mechanism to prevent double-spending**, ensuring asset uniqueness and ledger consistency

LibP2P Protocol in the Rubix Network

The Rubix Network utilizes **LibP2P**, a modular and extensible networking stack, to enable secure, peer-to-peer communication between nodes without reliance on centralized servers.

What is LibP2P?

LibP2P is a suite of protocols, libraries, and specifications that enables the creation of **decentralized network applications**. It forms the backbone of many modern peer-to-peer systems by replacing the traditional client-server model with **direct peer-to-peer communication**.

In Rubix, LibP2P is used as the **core transport layer** for all peer communications, including:

- **Consensus communication** between initiators, validators (notaries), and participants
- **Token transfer** messages from sender to receiver
- **Proof and metadata exchange** among validator quorums

Rubix achieves this by:

- **Tunneling all Rubix network traffic through LibP2P streams**
- **Adding all Rubix nodes to a single, private IPFS swarm**, isolating network communication for privacy and security
- **Using IPFS's listen and forward commands**, part of the LibP2P library, to establish reliable multi-hop connections over the internet

This architecture ensures that all Rubix communications are:

- **Peer-authenticated**
- **Content-addressed**
- **End-to-end encrypted**
- **Independent of centralized relay infrastructure**

Summary

LibP2P serves as the **network backbone** for the Rubix protocol enabling secure, decentralized, and resilient communication among all participating nodes. Through peer routing, DHT discovery, and encrypted streams, Rubix maintains a **fully decentralized P2P messaging layer**, essential for its scalable, trustless consensus and asset transfer mechanisms.

Glossary of Terms

Asset Token (NFT)
A non-fungible token (NFT) on the Rubix Network that represents a unique digital or physical asset, such as a certificate, land parcel or carbon credit. Asset tokens are non-divisible and maintain provenance through own tokenchains.

Credit Threshold (τ)
The minimum number of proof credits a validator must accumulate to be eligible for minting a new RBT. This threshold increases as the network matures (see Annexure 1).

Decentralized Identity (DID)
A self-sovereign, cryptographically verifiable identity generated by each Rubix node. Based on ECDSA-P256, DIDs are used to sign transactions and establish trust without central authorities.

Fungible Token
A token that is interchangeable and divisible, such as RBT or secondary utility tokens. Used for transactions, validation, governance or internal economies within subnets.

LibP2P
A modular networking library enabling decentralized peer-to-peer communication across Rubix nodes. It supports encrypted messaging, routing, and proof transmission in a trustless environment.

Mine NFT
A native NFT in the Rubix ecosystem that records all RBT mining events. It logs validator DIDs, proof credits consumed, and the timestamp of each new RBT issuance for transparency.

Private IPFS Swarm
A controlled and encrypted instance of IPFS exclusive to Rubix nodes. This ensures isolated data propagation and validator coordination.

Proof Credit
A unit of validator contribution in Rubix earned by pledging RBT tokens and validating transactions. Accumulated proof credits can be converted into new RBT tokens via mining.

Proof of Pledge (PoP)
The Rubix consensus model where validators pledge RBT tokens instead of burning compute (PoW) or locking massive stake (PoS). Validators earn proof credits based on their active contribution.

RBT (Rubix Token)
The primary utility token in the Rubix Network, capped at ~51.4 million. Used for transaction validation, validator pledging, smart contract deployment and governance

Secondary Utility Tokens
Custom fungible tokens issued by Rubix ecosystem projects or subnets. While they operate independently, they are still anchored to RBT and maintain interoperability with the core network.

Smart Contract (WASM)
A self-executing program deployed on the Rubix Network in WebAssembly (WASM) format. Rubix supports smart contracts written in Rust, Go, and C++, executed securely across validator nodes.

Subnet / Microchain
A private or domain-specific network built within Rubix that operates under its own governance and validation logic. Subnets use their own validators but remain anchored to RBT for consensus integration.

Tokenchain

A dedicated transaction history bound to a single token (fungible or non-fungible). Each chain records state changes.

Validator
A node that participates in transaction validation by pledging RBT tokens. Validators earn proof credits and help maintain the integrity, fault tolerance, and consensus of the Rubix network.

References

1. Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf
2. Benet, Juan. "Ipfs-content addressed, versioned, p2p file system." arXiv preprint arXiv:1407.3561 (2014).
3. Castro, Miguel, and Barbara Liskov. "Practical Byzantine fault tolerance." OSDI. Vol. 99. No. 1999. 1999.
4. Wood, Gavin. "Ethereum: A secure decentralized generalized transaction ledger." Ethereum project yellow paper 151.2014 (2014): 1-32.
5. Buterin, V. (2013). *Ethereum white paper: A next-generation smart contract and decentralized application platform*. https://ethereum.org/en/whitepaper/
6. Garay, J., Kiayias, A., & Leonardos, N. (2015). *The bitcoin backbone protocol: Analysis and applications*. Annual International Conference on the Theory and Applications of Cryptographic Techniques.
7. Pass, R., Seeman, L., & Shelat, A. (2017). *Analysis of the blockchain protocol in asynchronous networks*. Annual International Conference on the Theory and Applications of Cryptographic Techniques.
8. King, S., & Nadal, S. (2012). *PPCoin: Peer-to-peer crypto-currency with proof-of-stake*. https://peercoin.net/assets/paper/peercoin-paper.pdf
9. Park, S., Kwon, A., & Ren, L. (2018). *SPoRE: Proofs of Space for Outsourcing Resource-intensive Computation*. arXiv preprint arXiv:1806.03818.
10. Protocol Labs. (2020). *Filecoin: A Decentralized Storage Network*. https://filecoin.io/filecoin.pdf
11. W3C. (2021). *Decentralized Identifiers (DIDs) v1.0 – Core architecture, data model, and representations*. https://www.w3.org/TR/did-core/
12. Sovrin Foundation. (2018). *The Sovrin Identity Metasystem*. https://sovrin.org/wp-content/uploads/2018/03/TheSovrinIdentityMetasystem.pdf
13. Parity Technologies. (2019). *Substrate – Blockchain Framework with WASM Smart Contracts*. https://substrate.dev/
14. Haas, Andreas et al. (2017). *Bringing the Web up to Speed with WebAssembly*. ACM SIGPLAN Notices. https://webassembly.org/
15. Ethereum Foundation. *ERC-20 Token Standard*. https://ethereum.org/en/developers/docs/standards/tokens/erc-20/
16. Ethereum Foundation. *ERC-721 Non-Fungible Token Standard*. https://eips.ethereum.org/EIPS/eip-721
17. DFINITY Foundation. (2021). *Internet Computer & Canisters (Smart Contracts using WASM)*. https://internetcomputer.org/

18. Zyskind, G., Nathan, O., & Pentland, A. (2015). *Decentralizing Privacy: Using Blockchain to Protect Personal Data*. IEEE Security and Privacy Workshops.
19. Li, J., Tople, S., & Saxena, P. (2017). *Protego: A General Privacy-Preserving Data Marketplace Architecture*. arXiv:1710.08829
20. Benet, J., Grewal, N., & Protocol Labs. *IPLD: InterPlanetary Linked Data*. https://ipld.io/
21. Buterin, V. (2021). *The Most Important Scarce Resource is Legitimacy*. https://vitalik.eth.limo/general/2021/08/16/legitimacy.html
22. Weyl, E. G., Posner, E. A., & Glen Weyl, E. (2018). *Radical Markets: Uprooting Capitalism and Democracy for a Just Society*. Princeton University Press.
23. Carbonneau, M., & Messier, P. (2023). *On-Chain Governance Frameworks: A Comparative Analysis of DAOs*. Journal of Decentralized Finance.
24. Ostrom, E. (1990). *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press.

Annexure 1: Difficulty Threshold Table (Proof Credits)

Level	Cumulative tokens ('000)	Tokens awarded ('000)	PCs/token
0	0.056	0.056	
1	4,300,000	4,300,000	0.125
2	6,725,000	2,425,000	16
3	9,028,750	2,303,750	32
4	11,217,313	2,188,563	64
5	13,296,447	2,079,134	128
6	15,271,625	1,975,178	256
7	17,148,043	1,876,419	512
8	18,930,641	1,782,598	1,024

9	20,624,109	1,693,468	2,048
10	22,232,904	1,608,795	4,096
11	23,761,259	1,528,355	8,192
12	25,213,196	1,451,937	12,288
13	26,592,536	1,379,340	18,432
14	27,902,909	1,310,373	27,648
15	29,147,764	1,244,855	41,472
16	30,330,375	1,182,612	62,208
17	31,453,857	1,123,481	93,312
18	32,521,164	1,067,307	139,968
19	33,535,106	1,013,942	209,952
20	34,498,350	963,245	314,928
21	35,413,433	915,082	472,392
22	36,282,761	869,328	590,490
23	37,108,623	825,862	738,113
24	37,893,192	784,569	922,641
25	38,638,532	745,340	1,153,301
26	39,346,606	708,073	1,441,626
27	40,019,275	672,670	1,802,032

28	40,658,312	639,036	2,252,541
29	41,265,396	607,084	2,815,676
30	41,842,126	576,730	3,519,595
31	42,390,020	547,894	4,399,493
32	42,910,519	520,499	4,949,430
33	43,404,993	494,474	5,568,109
34	43,874,743	469,750	6,264,122
35	44,321,006	446,263	7,047,138
36	44,744,956	423,950	7,928,030
37	45,147,708	402,752	8,919,034
38	45,530,323	382,615	10,033,913
39	45,893,807	363,484	11,288,152
40	46,239,116	345,310	12,699,171
41	46,567,160	328,044	14,286,567
42	46,878,802	311,642	15,179,478
43	47,174,862	296,060	16,128,195
44	47,456,119	281,257	17,136,207
45	47,723,313	267,194	18,207,220
46	47,977,148	253,834	19,345,171

47	48,218,290	241,143	20,554,245
48	48,447,376	229,085	21,838,885
49	48,665,007	217,631	23,203,815
50	48,871,757	206,750	24,654,054
51	49,068,169	196,412	26,194,932
52	49,254,760	186,592	26,587,856
53	49,432,022	177,262	26,986,674
54	49,600,421	168,399	27,391,474
55	49,760,400	159,979	27,802,346
56	49,912,380	151,980	28,219,381
57	50,056,761	144,381	28,642,672
58	50,193,923	137,162	29,072,312
59	50,324,227	130,304	29,508,397
60	50,441,500	117,273	29,951,023
61	50,547,047	105,546	30,400,288
62	50,642,038	94,992	30,856,292
63	50,727,530	85,492	31,319,137
64	50,804,473	76,943	31,788,924
65	50,873,722	69,249	32,265,758

66	50,936,046	62,324	32,749,744
67	50,992,138	56,092	33,240,990
68	51,042,620	50,482	33,739,605
69	51,088,054	45,434	34,245,699
70	51,128,945	40,891	34,759,385
71	51,165,747	36,802	35,280,775
72	51,198,868	33,121	35,809,987
73	51,228,677	29,809	36,347,137
74	51,255,506	26,828	36,892,344
75	51,279,651	24,146	37,445,729
76	51,301,382	21,731	38,007,415
77	51,320,940	19,558	38,577,526
78	51,338,543	17,602	39,156,189
Every subsequent level, token supply declines by 10% and threshold level increases by 1.5%			