POLLUX

Advanced Decentralized Blockchain Platform

Whitepaper Version: 1.0

POLLUX Protocol Version:

1.0

POLLUX DAO

February 10th, 2023, Dubai

1. Introduction

1.1 Vision

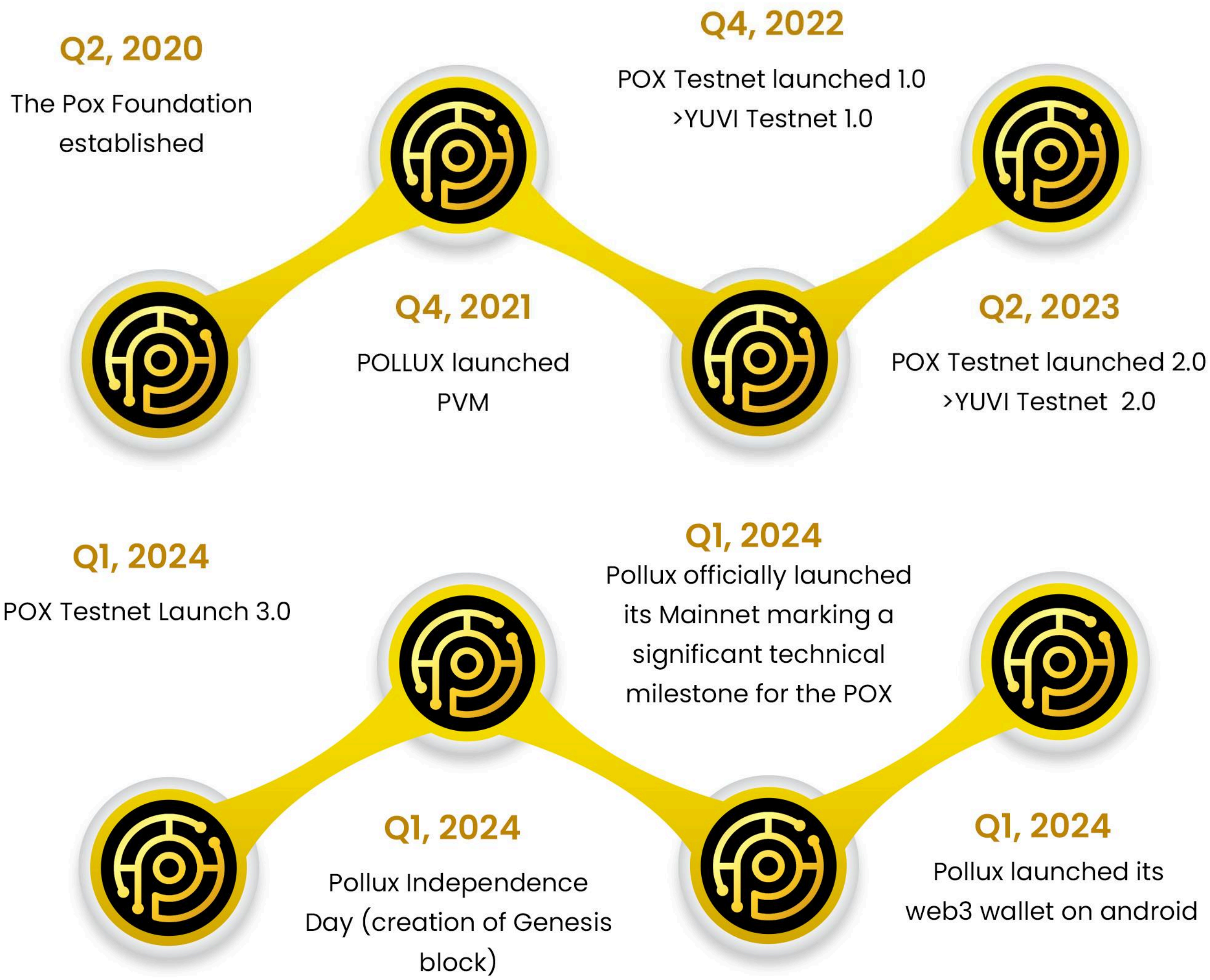
POLLUX represents a bold initiative aimed at building a fully decentralized Internet along with its infrastructure. At the heart of this endeavor lies the POLLUX Protocol, which stands as one of the most extensive blockchain-operating systems globally. It provides robust support to public blockchains, offering exceptional throughput, scalability, and availability for all Decentralized Applications (DApps) within the POLLUX ecosystem.

1.2 Background

The emergence of Bitcoin in 2009 marked a significant shift in society's perception of the traditional financial system, especially in the aftermath of the Great Recession (2007-2008). Amid the collapse of centralized hedge funds and banks due to speculative activities in opaque financial derivatives, blockchain technology introduced a transparent universal ledger accessible to anyone seeking transaction details. Transactions were safeguarded through cryptographic techniques, utilizing a Proof of Work (PoW) consensus mechanism to thwart double-spending concerns.

In late 2013, the Ethereum white paper outlined a vision for a network where smart contracts and a Turing-complete Ethereum Virtual Machine (EVM) would enable developers to engage with the network via Decentralized Applications (DApps). However, as transaction volumes surged in Bitcoin and Ethereum by 2017, it became evident that the existing state of cryptocurrencies struggled with scalability issues, manifesting in prolonged transaction times and exorbitant fees. Consequently, POLLUX emerged as an innovative response to these critical scalability challenges.

1.3 History



Q2, 2020

The Pox Foundation established

Q4, 2021

POLLUX launched PVM

Q4, 2022

POX Testnet launched 1.0
>YUVI Testnet 1.0

Q2, 2023

POX Testnet launched 2.0
>YUVI Testnet 2.0

Q1, 2024

POX Testnet Launch 3.0

Q1, 2024

Pollux Independence Day (creation of Genesis block)

Q1, 2024

Pollux officially launched its Mainnet marking a significant technical milestone for the POX

Q1, 2024

Pollux launched its web3 wallet on android

The POLLUX DAO was established in June 2020 in Dubai. In December 2021, POLLUX launched its open-source protocol. The Testnet and Blockchain Explorer Web Wallet were all launched by March 2022. POLLUX Mainnet launched shortly afterward in 2024, marking the PoxChain release as a technical milestone. In Q1 2024, POLLUX declared its independence by creating the Genesis block. In Dec 2021, POLLUX launched the POLLUX Virtual Machine (PVM), a complete developers' toolset, and 360 support system. The POLLUX roadmap involves combining 100 million users with the POLLUX network via Project Atlas and fostering the developer community to launch exciting new DApps on the POLLUX network.

1.4 Terminology

Address/Wallet
An address or wallet consisting of account credentials on the POLLUX network is generated by a key pair, which consists of a private key and a public key, the latter being derived from the former through an algorithm. The public key is usually used for session key encryption, signature verification, and encrypting data that could be decrypted by a corresponding private key.

ABI
An application binary interface (ABI) is an interface between two binary program modules; usually one of these modules is a library or an operating system facility, and the other is a user-run program.

API
An application programming interface (API) is mainly used for user-client development. With API support, token issuance platforms can also be designed by developers themselves.

Asset
In POLLUX's documents, the asset is the same as a token, which is also denoted as a PRC-10 token.

Bandwidth Points (BP)
To keep the network operating smoothly, POLLUX network transactions use BP as fuel. Each account gets 300 free daily BP and more can be obtained by freezing POX for BP. Both POX and PRC-10 token transfers are normal transactions costing BP. Smart contract deployment and execution transactions consume both BP and Energy.

Block
Blocks contain the digital records of transactions. A complete block consists of the magic number, block size, block header, transaction counter, and transaction data.

Block Reward
Block production rewards are sent to a sub-account (address/wallet). Super Representatives can claim their rewards on POLLUXscan or through the API directly.

Block Header
A block header is part of a block. POLLUX block headers contain the previous block’s hash, the Merkle root, timestamp, version, and witness address.

Cold Wallet
A Cold wallet, also known as an offline wallet, keeps the private key completely disconnected from any network. Cold wallets are usually installed on "cold" devices (e.g. computers or mobile phones staying offline) to ensure the security of the POX private key.

DApp
Decentralized Application is an App that operates without a centrally trusted party. An application that enables direct interaction/agreements/communication between end users and/or resources without a middleman.

gRPC
gRPC (gRPC Remote Procedure Calls) is an open-source remote procedure call (RPC) system initially developed at Google. It uses HTTP/2 for transport, Protocol Buffers as the interface description language, and provides features such as authentication, bidirectional streaming, flow control, blocking or nonblocking bindings, and cancellation and timeouts. It generates cross-platform client and server bindings for many languages. Most common usage scenarios include connecting services in microservices style architecture and connecting mobile devices, and browser clients to backend services.

Hot Wallet
Hot wallet, also known as an online wallet, allows a user's private key to be used online, thus it could be susceptible to potential vulnerabilities or interception by malicious actors.

JDK
Java Development Kit is the Java SDK used for Java applications. It is the core of Java development, comprising the Java application environment (JVM+Java class library) and Java tools.

KhaosDB
POLLUX has a KhaosDB in the full-node memory that can store all the newly forked chains generated within a certain period and supports witnesses to switch from their active chain swiftly into a new main chain. See 2.2.2 State Storage for more details.

LevelDB
LevelDB was initially adopted with the primary goal of meeting the requirements of fast R/W and rapid development. After launching the Mainnet, POLLUX upgraded its database to an entirely customized one catered to its very own needs. See 2.2.1 Blockchain Storage for more details.

Merkle Root
A Merkle root is the hash of all hashes of all transactions included as part of a block in a blockchain network. See 3.1 Delegated Proof of Stake (DPoS) for more details.

Public Testnet (Yuvi)
A version of the network running in a single-node configuration. Developers can connect and test features without worrying about the economic loss. Testnet tokens have no value and anyone can request more from the public faucet.

RPC
In distributed computing, a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in a different address space (commonly on another computer on a shared network), which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction.

Scalability
Scalability is a feature of the POLLUX Protocol. A system, network, or process can handle a growing amount of work or its potential to be enlarged to accommodate that growth.

RAM
RAM replaced drop as the smallest unit of POX. 1 POX = 1,000,000 RAM.

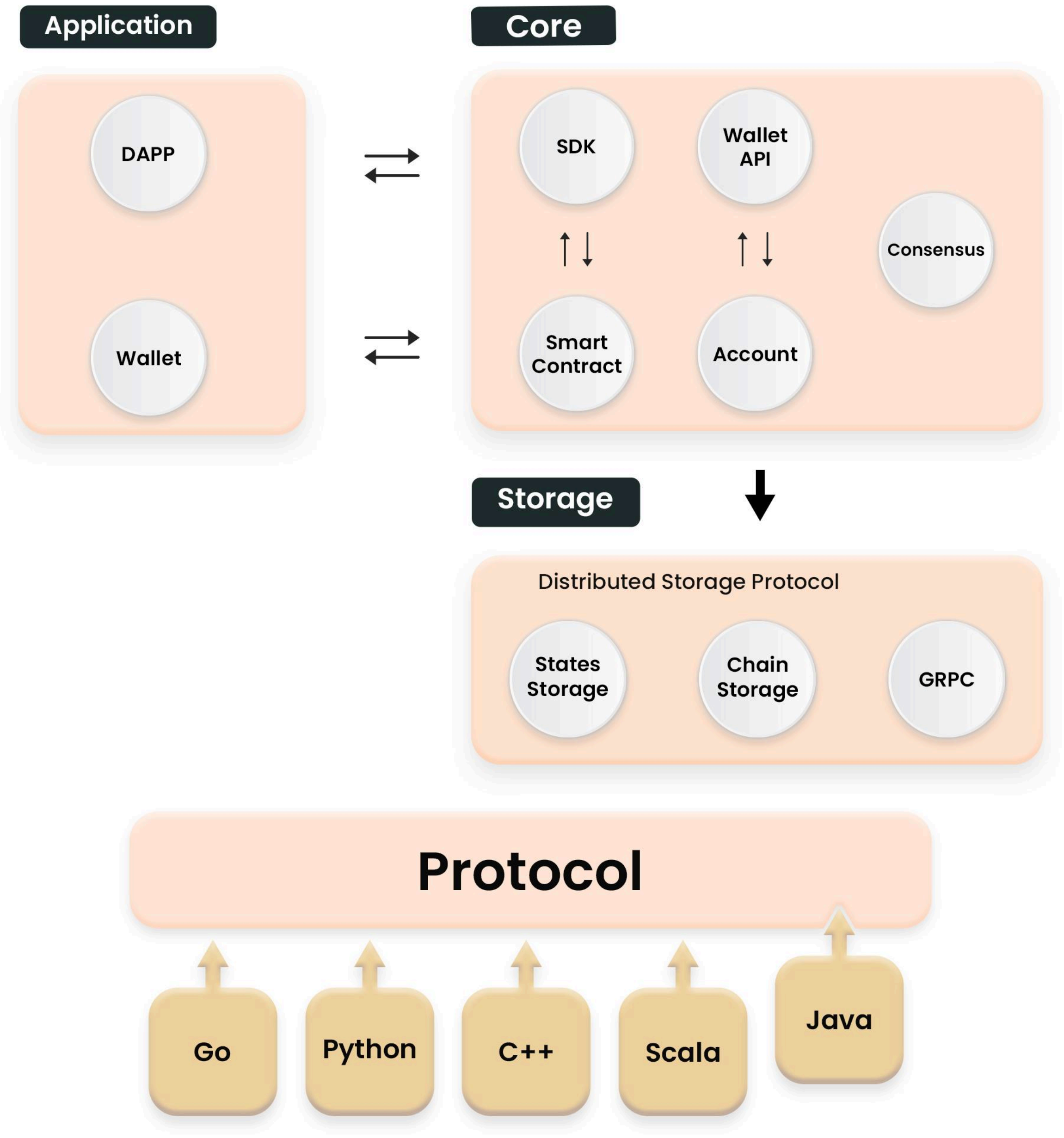
Throughput
High throughput is a feature of POLLUX Mainnet. It is measured in Transactions Per Second (TPS), namely the maximum transaction capacity in one second.

Timestamp
The approximate time of block production is recorded as Unix timestamp, which is the number of milliseconds that have elapsed since 00:00:00 01 Jan 1970 UTC.

TKC
Token configuration.

PRC-10
A standard of the crypto token on the POLLUX platform. Certain rules and interfaces are required to follow when holding an initial coin offering on the POLLUX blockchain.

POX
POX stands for POLLUX, which is the official cryptocurrency of POLLUX.



2. Architecture

POLLUX adopts a 3-layer architecture divided into Storage Layer, Core Layer, and Application Layer. The POLLUX protocol adheres to Google Protobuf, which intrinsically supports multi-language extension.

Figure 1: POLLUX 3-layer Architecture

2.1 Core

The core layer of POLLUX encompasses various modules, such as smart contracts, account management, and consensus mechanisms. Utilizing a stack-based virtual machine and an optimized instruction set, POLLUX provides a robust infrastructure. To enhance support for DApp developers, Solidity was selected as the primary smart contract language, with plans for future integration of other advanced languages. Moreover, POLLUX's consensus mechanism relies on Delegated Proof of Stake (DPoS), prompting numerous innovations tailored to meet its distinctive needs.

2.2 Storage

POLLUX designed a unique distributed storage protocol consisting of Block Storage and State Storage. The notion of a graph database was introduced into the design of the storage layer to better meet the need for diversified data storage in the real world.

2.2.1 Blockchain Storage

POLLUX has developed a distinctive distributed storage protocol comprising Block Storage and State Storage. Introducing the concept of a graph database into the storage layer design enables more versatile data storage solutions to address real-world requirements effectively.

2.2.2 State Storage

POLLUX incorporates a KhaosDB within the memory of its full nodes, capable of storing all newly forked chains produced within a defined timeframe. This feature supports witnesses in seamlessly transitioning from their active chain to a new main chain. Moreover, KhaosDB serves to enhance the stability of blockchain storage, safeguarding against abnormal terminations while in an intermediate state.

2.3 Application

Developers can create a diverse range of DApps and customized wallets on POLLUX. Since POLLUX enables smart contracts to be deployed and executed, the opportunities for utility applications are unlimited.

2.4 Protocol

The POLLUX protocol adopts Google Protocol Buffers, a language-agnostic, platform-independent, and extensible method for serializing structured data. This facilitates seamless integration into various communication protocols, data storage systems, and other applications.

2.4.1 Protocol Buffers

Protocol Buffers (Protobuf) offers a versatile, efficient, and automated approach to serializing structured data, akin to JSON or XML but notably smaller, faster, and simpler.

Protobuf definitions (in .proto format) can be leveraged to generate code for various programming languages including C++, Java, C#, Python, Ruby, Golang, and Objective-C, thanks to official code generators. Additionally, numerous third-party implementations exist for supporting additional languages. This streamlines development for clients by providing unified API definitions and optimizing data transfers. Clients can access the API definitions from POLLUX's protocol repository and seamlessly integrate them using the automatically generated code libraries.

In comparison to XML, Protocol Buffers typically yield data that is 3 to 10 times smaller and operations that are 20 to 100 times faster, boasting a syntax that is less prone to ambiguity. Protobuf generates data access classes that offer enhanced programmability and ease of use.

2.4.2 HTTP

POLLUX Protocol provides a RESTful HTTP API alternative to the Protobuf API. They share the same interface but the HTTP API can be readily used in javascript clients.

2.5 POLLUX Virtual Machine (PVM)

The PVM is a lightweight, Turing-complete virtual machine developed for POLLUX’s ecosystem. The PVM connects seamlessly with the existing development ecosystem to provide millions of global developers with a custom-built blockchain system that is efficient, convenient, stable, secure, and scalable.

2.6 Decentralized Exchange (DEX)

The POLLUX network inherently supports decentralized exchange functionalities, with each decentralized exchange comprising multiple trading pairs. A trading pair, denoted as "Exchange," represents a market for trading PRC-10 tokens, or between a PRC-10 token and POX. Any account within the network can create a trading pair between any tokens, even if an identical pair already exists.

Trading activities and price fluctuations within these trading pairs adhere to the principles outlined in the Bancor Protocol. As per the POLLUX network's guidelines, the weights of the two tokens within all trading pairs are identical. Consequently, the ratio of their balances determines the price between them. For instance, consider a trading pair involving tokens ABC and DEF. If ABC has a balance of 10 million and DEF has a balance of 1 million, their equal weights dictate that 10 ABC is equivalent to 1 DEF. In other words, the ratio of ABC to DEF in this pair is 10 ABC per DEF.

2.7 Implementation

The POLLUX blockchain code is implemented in Java and was originally a fork from Ethereum.

3. Consensus

3.1 Delegated Proof of Stake (DPoS)

The earliest consensus mechanism is the Proof of Work (PoW) consensus mechanism. This protocol is currently implemented in Bitcoin and Ethereum. In PoW systems, transactions broadcast through the network are grouped into nascent blocks for miner confirmation. The confirmation process involves hashing transactions using cryptographic hashing algorithms until a Merkle root has been reached, creating a Merkle tree:

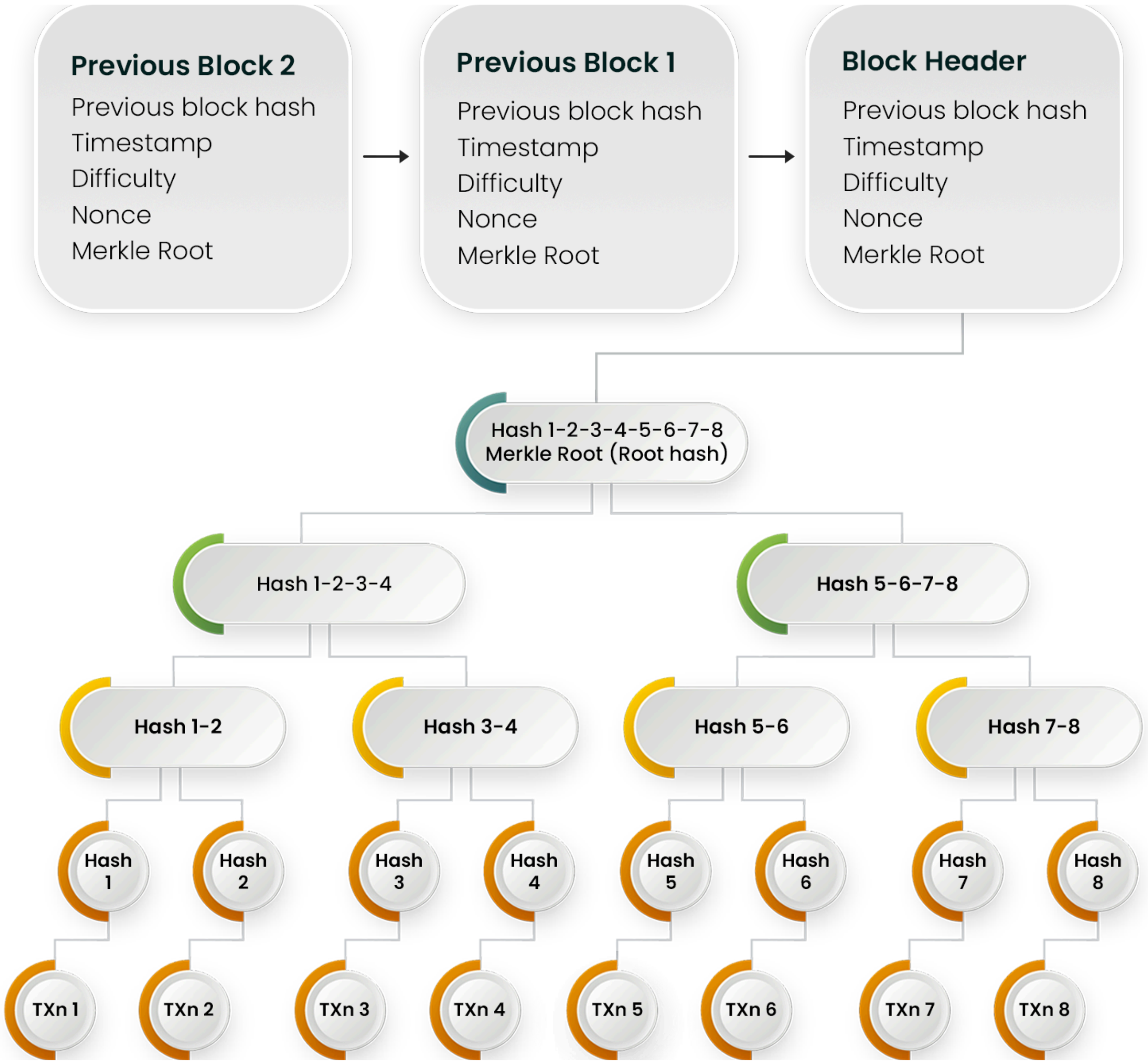


Figure 2: 8 POX transactions are hashed into the Merkle root. This Merkle root is then included in the block header, which is attached to the previously confirmed blocks to form a blockchain. This allows for easy and transparent tracking of transactions, timestamps, and other related information.

Cryptographic hashing algorithms are useful in network attack prevention because they possess several properties:

- ***Input/Output length size*** - The algorithm can pass in an input of any length in size, and outputs a fixed-length hash value.
- ***Efficiency*** - The algorithm is relatively easy and fast to compute.
- ***Preimage resistance*** - For a given output z, it is impossible to find any input x such that $h(x) = z$. In other words, the hashing algorithm $h(x)$ is a one-way function in which only the output can be found, given an input. The reverse is not possible.
- ***Collision resistance*** - It is computationally infeasible to find any pairs $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$. In other words, the probability of finding two different inputs hashing to the same output is extremely low. This property also implies *second preimage resistance*.
- ***Second preimage resistance*** - Given x_1, and thus $h(x_1)$, it is computationally infeasible to find any x_2 such that $h(x_1) = h(x_2)$. While this property is similar to *collision resistance*, the property differs in that it is saying an attacker with a given x_1 will find it computationally infeasible to find any x_2 hashing to the same output.
- ***Deterministic*** - maps each input to one and only one output.
- ***Avalanche effect*** - a small change in the input results in an entirely different output.

These characteristics imbue the cryptocurrency network with inherent value, safeguarding against attacks that could compromise its integrity. When miners validate a block, they receive tokens as a built-in incentive for their participation in the network. However, with the steady growth of the global cryptocurrency market capitalization, miners began to centralize their operations, directing their computing resources towards accumulating tokens as assets rather than actively contributing to the network. This evolution saw a transition from CPU mining to GPU mining, and eventually to the utilization of powerful ASICs.

In a notable study, the total power consumption of Bitcoin mining has been estimated to be as high as 3 GW, comparable to the power consumption of Ireland. Moreover, projections suggest that this figure could escalate to 8 GW in the near future.

To address the issue of energy wastage, many new networks have proposed the Proof of Stake (PoS) consensus mechanism. In PoS networks, token holders lock their token balances to become block validators, who then take turns proposing and voting on the next block. However, a drawback of standard PoS is that the influence of validators is directly proportional to the amount of tokens they have locked up. Consequently, entities with substantial holdings of the network's base currency wield disproportionate influence within the network ecosystem.

The POLLUX consensus mechanism employs an innovative Delegated Proof of Stake (DPoS) system, where 27 Super Representatives (SRs) are responsible for producing blocks on the network. Every 6 hours, POX account holders who freeze their accounts have the opportunity to vote for a selection of SR candidates. The top 27 candidates, based on the number of votes received, are designated as the SRs. Voters may choose SRs based on various criteria, such as projects sponsored by SRs to increase POX adoption and rewards distributed to voters. This approach fosters a more democratized and decentralized ecosystem. Although SRs' accounts function like normal accounts, their accumulation of votes grants them the ability to produce blocks.

Compared to the low throughput rates of Bitcoin and Ethereum, attributed to their Proof of Work (PoW) consensus mechanism and scalability issues, POLLUX's DPoS system offers a more efficient mechanism, resulting in 2000 transactions per second (TPS) compared to Bitcoin's 3 TPS and Ethereum's 15 TPS.

The POLLUX protocol network generates one block every three seconds, with each block awarding 0.05 POX to Super Representatives. Annually, a total of 1,555,200 POX tokens are distributed among the 27 SRs. Upon completion of block production, rewards are deposited into a sub-account within the super-ledger. SRs can review these POX tokens but cannot directly utilize them. Once every 24 hours, each SR has the option to withdraw rewards from the sub-account to their specified SR account.

Within the POLLUX network, three types of nodes exist: Witness Node, Full Node, and Solidity Node. Witness nodes, established by SRs, primarily handle block production and proposal creation/voting. Full nodes offer APIs and broadcast transactions and blocks. Solidity nodes synchronize blocks from other Full Nodes and provide indexable APIs.

4. Account

4.1 Types

The three types of accounts in the POLLUX network are regular accounts, token accounts, and contract accounts.

1. Regular accounts are used for standard transactions.
2. Token accounts are used for storing PRC-10 tokens.
3. Contract accounts are smart contract accounts created by regular accounts and can be triggered by regular accounts as well.

4.2 Creation

There are three ways to create a POLLUX account:

1. Create a new account through API
2. Transfer POX into a new account address
3. Transfer any PRC-10 token into a new account address

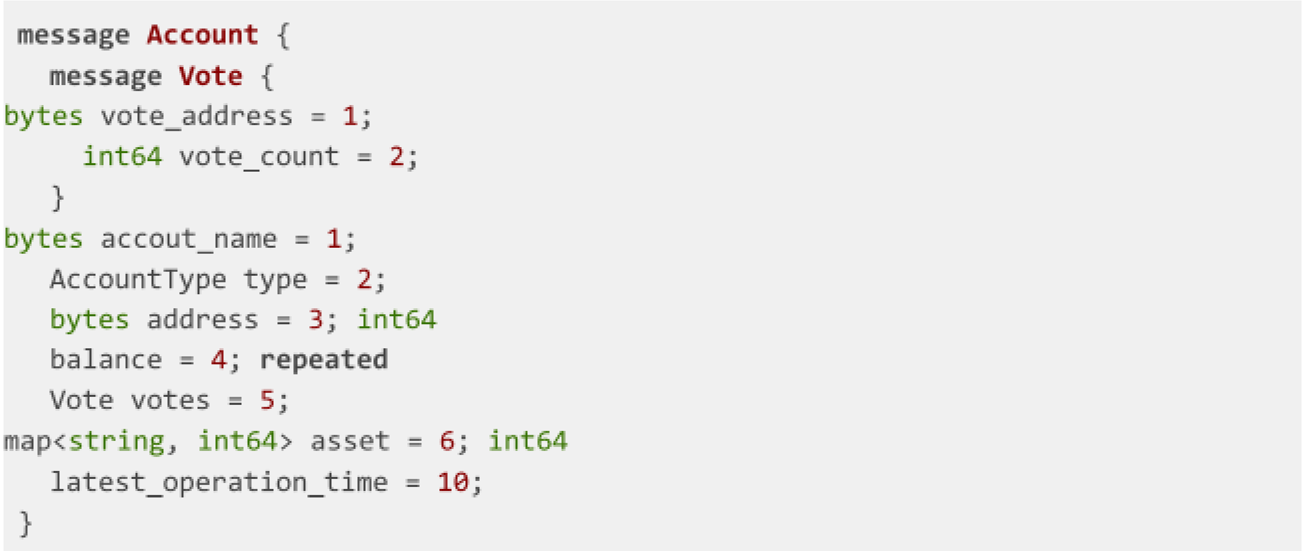
The process of generating an offline key-pair address independent of the POLLUX network involves several steps to ensure its uniqueness and security. First, a key-pair consisting of a public and private key is generated. The public key, represented as a 64-byte byte array, is then hashed using the SHA3-256 function, specifically employing the KECCAK-256 protocol. From the resulting hash, the last 20 bytes are extracted. To create the initial address, 37 is prepended to the byte array, ensuring a length of 21 bytes. Next, the initial address undergoes double hashing using the SHA3-256 function. From the second hash, the first 4 bytes are taken as the verification code. This verification code is appended to the initial address, resulting in the formation of the address in base58check format through base58 encoding. Notably, a valid Mainnet address, encoded in this manner, commences with P and spans 34 bytes in length. This rigorous process ensures the generation of a secure and compatible address for use within the POLLUX network.

4.3 Structure

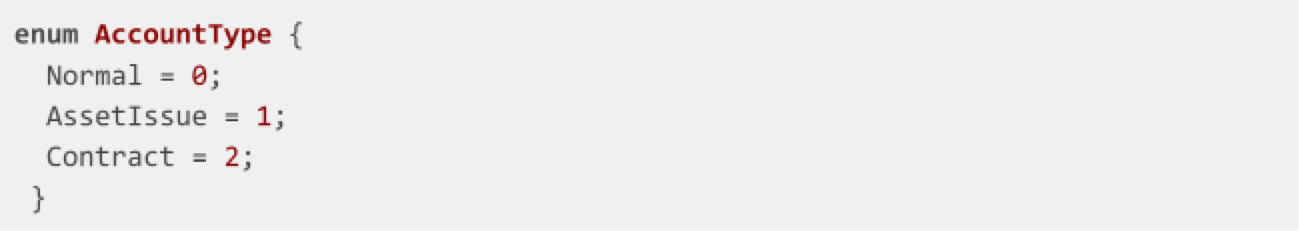
The three different account types are Normal, AssetIssue, and Contract. An Account contains 7 parameters:

1. **account_name**: the name for this account – e.g. BillsAccount.
2. **type**: what type of this account is – e.g. 0 (stands for type ‘Normal’).
3. **balance**: balance of this account – e.g. 4213312.
4. **vote**: received votes on this account – e.g. {(“0x1b7w…9xj3”,323), (“0x8djq…j12m”,88),…,(“0x82nd…mx6i”,10001)}.
5. **asset**: other assets expected POX in this account – e.g. {<“WishToken”, 66666>, <” Dogie”, 233>}.
6. **latest_operation_time**: the latest operation time of this account.

Protobuf data structure:



```
message Account {
  message Vote {
bytes vote_address = 1;
    int64 vote_count = 2;
  }
bytes accout_name = 1;
  AccountType type = 2;
  bytes address = 3; int64
  balance = 4; repeated
  Vote votes = 5;
map<string, int64> asset = 6; int64
  latest_operation_time = 10;
}
```

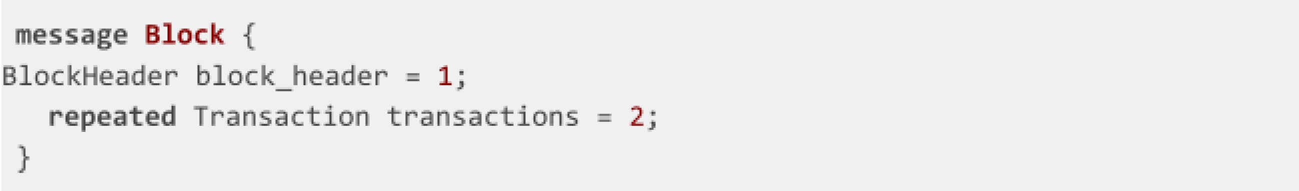


```
enum AccountType {
  Normal = 0;
  AssetIssue = 1;
  Contract = 2;
 }
```

5. Block

A block typically contains a block header and several transactions.

Protobuf data structure:

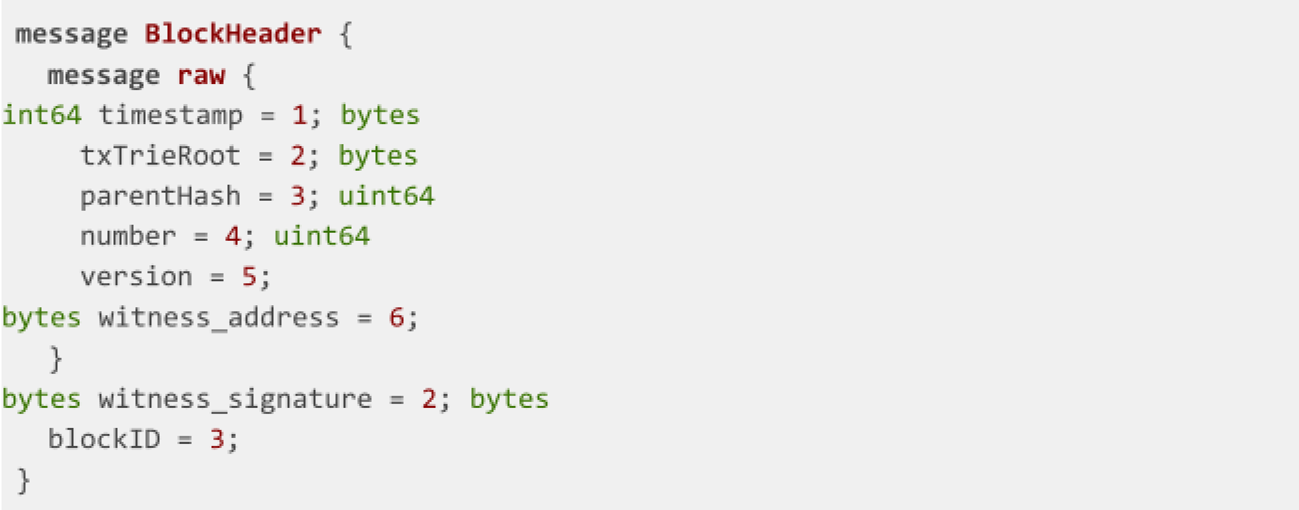


```
 message Block {
BlockHeader block_header = 1;
   repeated Transaction transactions = 2;
 }
```

5.1 Block Header

A block header contains **raw_data**, **witness_signature**, and **blockID**.

Protobuf data structure:



```
 message BlockHeader {
   message raw {
int64 timestamp = 1; bytes
     txTrieRoot = 2; bytes
     parentHash = 3; uint64
     number = 4; uint64
     version = 5;
bytes witness_address = 6;
   }
bytes witness_signature = 2; bytes
   blockID = 3;
 }
```

5.1.1 Raw Data

Raw data is denoted as **raw_data** in Protobuf. It contains the raw data of a message, containing 6 parameters:

1. **timestamp**: timestamp of this message – e.g. 1543884429000.
2. **txTrieRoot**: the Merkle Tree’s Root – e.g. 7dacsa…3ed.
3. **parentHash**: the hash of the last block – e.g. 7dacsa…3ed.
4. **number**: the block height – e.g. 4638708.
5. **version**: reserved – e.g. 5.

6. **witness_address**: the address of the witness packed in this block – e.g. 37928c...4d21.

5.1.2 Witness Signature

The Witness signature is denoted as **witness_signature** in Protobuf, which is the signature for this block header from the witness node.

5.1.3 Block ID

Block ID is denoted as **blockID** in Protobuf. It contains the atomic identification of a block. A Block ID contains 2 parameters:

1. **hash**: the hash of the block.
2. **number**: the hash and height of the block.

5.2 Transaction

5.2.1 Signing

POLLUX's transaction signing process adheres to the standard ECDSA cryptographic algorithm, employing the SECP256K1 elliptic curve. In this scheme, a private key is generated as a random number, while the corresponding public key is derived as a point on the elliptic curve. The process begins with the generation of a random number to serve as the private key, followed by multiplication of the base point of the elliptic curve by this private key to obtain the public key. When a transaction occurs, the raw data associated with the transaction is first converted into byte format. Subsequently, the raw data undergoes SHA-256 hashing. The private key corresponding to the contract address is then utilized to sign the resulting SHA256 hash. The resulting signature is subsequently appended to the transaction data, finalizing the transaction signing process.

5.2.2 Bandwidth Model

In the POLLUX network, ordinary transactions consume bandwidth points, while smart contract operations consume both energy and bandwidth points. There are two types of bandwidth points available to users: those obtained from freezing POX and a daily allocation of 300 free bandwidth points.

When a POX transaction is broadcast, it is transmitted and stored in the form of a byte array over the network. The bandwidth points consumed by a transaction are calculated by multiplying the number of transaction bytes by the bandwidth points rate. For example, if a transaction has a byte array length of 200, it consumes 200 bandwidth points. However, if a POX or token transfer results in the creation of a target account, only the bandwidth points consumed to create the account are deducted, and additional bandwidth points are not deducted.

In standard POX transfer scenarios from one POX account to another, the network first consumes the bandwidth points gained by the transaction initiator for freezing POX. If that is insufficient, it then consumes the daily allocation of 300 free bandwidth points. If neither source provides enough bandwidth points, the network consumes the POX balance of the transaction initiator.

For PRC-10 token transfers, the network verifies whether the total free bandwidth points of the issued token asset are sufficient. If not, the bandwidth points obtained from freezing POX are consumed. If there are still insufficient bandwidth points, the network consumes the POX balance of the transaction initiator. This process ensures that transactions are processed efficiently and fairly within the POLLUX network.

5.2.3 Fee

POLLUX network generally does not charge fees for most transactions, however, due to system restrictions and fairness, bandwidth usage and transactions do take in certain fees.

Fee charges are broken down into the following categories:

1. Normal transactions cost bandwidth points. Users can use the free daily bandwidth points (300) or freeze POX to obtain more. When bandwidth points are not enough, POX will be used directly from the sending account. The POX needed is the number of bytes * 10 RAM.
2. Smart contracts cost energy (Section 6) but will also need bandwidth points for the transaction to be broadcasted and confirmed. The bandwidth cost is the same as above.
3. All query transactions are free. It doesn’t cost energy or bandwidth.

POLLUX network also defines a set of fixed fees for the following transactions:

1. Creating a witness node: 9999 POX
2. Issuing a PRC-10 token: 1024 POX
3. Creating a new account: 0.1 POX
4. Creating an exchange pair: 1024 POX

5.2.4 Transaction as Proof of Stake (TaPoS)

POLLUX implements TaPoS (Transaction as Proof of Stake) as a crucial mechanism to ensure that transactions are confirmed on the main blockchain, while also deterring the forging of counterfeit chains. In TaPoS, each transaction is mandated to include a portion of the hash of a recent block header. This stipulation serves multiple purposes: it prevents transactions from being replayed on forks that do not incorporate the referenced block, and it signals to the network the presence of a particular user and their stake on a specific fork. By adhering to this consensus mechanism, the network is fortified against various forms of attacks, including Denial of Service, 51% attacks, selfish mining, and double spend attempts. TaPoS thus plays a vital role in safeguarding the integrity and security of the POLLUX network.

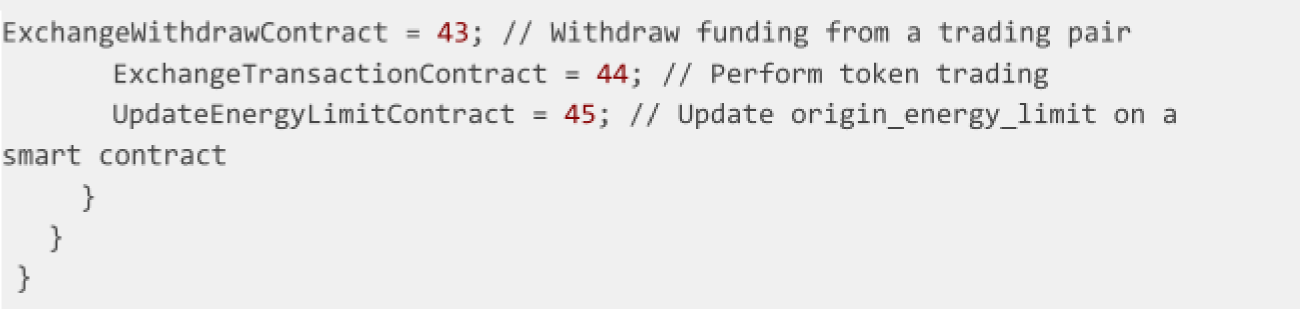
5.2.5 Transaction Confirmation

A transaction is included in a future block after being broadcast to the network. After 19 blocks are mined on POLLUX (including its own block), the transaction is confirmed. Each block is produced by one of the top 27 Super Representatives in a round-robin fashion. Each block takes ~3 seconds to be mined on the blockchain. Time may slightly vary for each Super Representative due to network conditions and machine configurations. In general, a transaction is considered fully confirmed after
~1 minute.

5.2.6 Structure

Transaction APIs consist of the following functions:

```
message Transaction {
  message Contract {
    enum ContractType {
      AccountCreateContract = 0; // Create account/wallet
      TransferContract = 1; // Transfer POX
      TransferAssetContract = 2; // Transfer PRC10 token
      VoteWitnessContract = 4; // Vote for Super Representative (SR)
      WitnessCreateContract = 5; // Create a new SR account
      AssetIssueContract = 6; // Create a new PRC10 token
      WitnessUpdateContract = 8; // Update SR information
      ParticipateAssetIssueContract = 9; // Purchase PRC10 token
      AccountUpdateContract = 10; // Update account/wallet information
      FreezeBalanceContract = 11; // Freeze POX for bandwidth or energy
      UnfreezeBalanceContract = 12; // Unfreeze POX
      WithdrawBalanceContract = 13; // Withdraw SR rewards, once per day
      UnfreezeAssetContract = 14; // Unfreeze PRC10 token
      UpdateAssetContract = 15; // Update a PRC10 token’s information
      ProposalCreateContract = 16; // Create a new network proposal by any SR
      ProposalApproveContract = 17; // SR votes yes for a network proposal
      ProposalDeleteContract = 18; // Delete a network proposal by owner
      CreateSmartContract = 30; // Deploy a new smart contract
      TriggerSmartContract = 31; // Call a function on a smart contract
      GetContract = 32; // Get an existing smart contract
      UpdateSettingContract = 33; // Update a smart contract’s parameters
      ExchangeCreateContract = 41; // Create a token trading pair on DEX
      ExchangeInjectContract = 42; // Inject funding into a trading pair
```



```
ExchangeWithdrawContract = 43; // Withdraw funding from a trading pair
       ExchangeTransactionContract = 44; // Perform token trading
       UpdateEnergyLimitContract = 45; // Update origin_energy_limit on a
smart contract
     }
   }
 }
```

6. POLLUX Virtual Machine (PVM)

6.1 Introduction

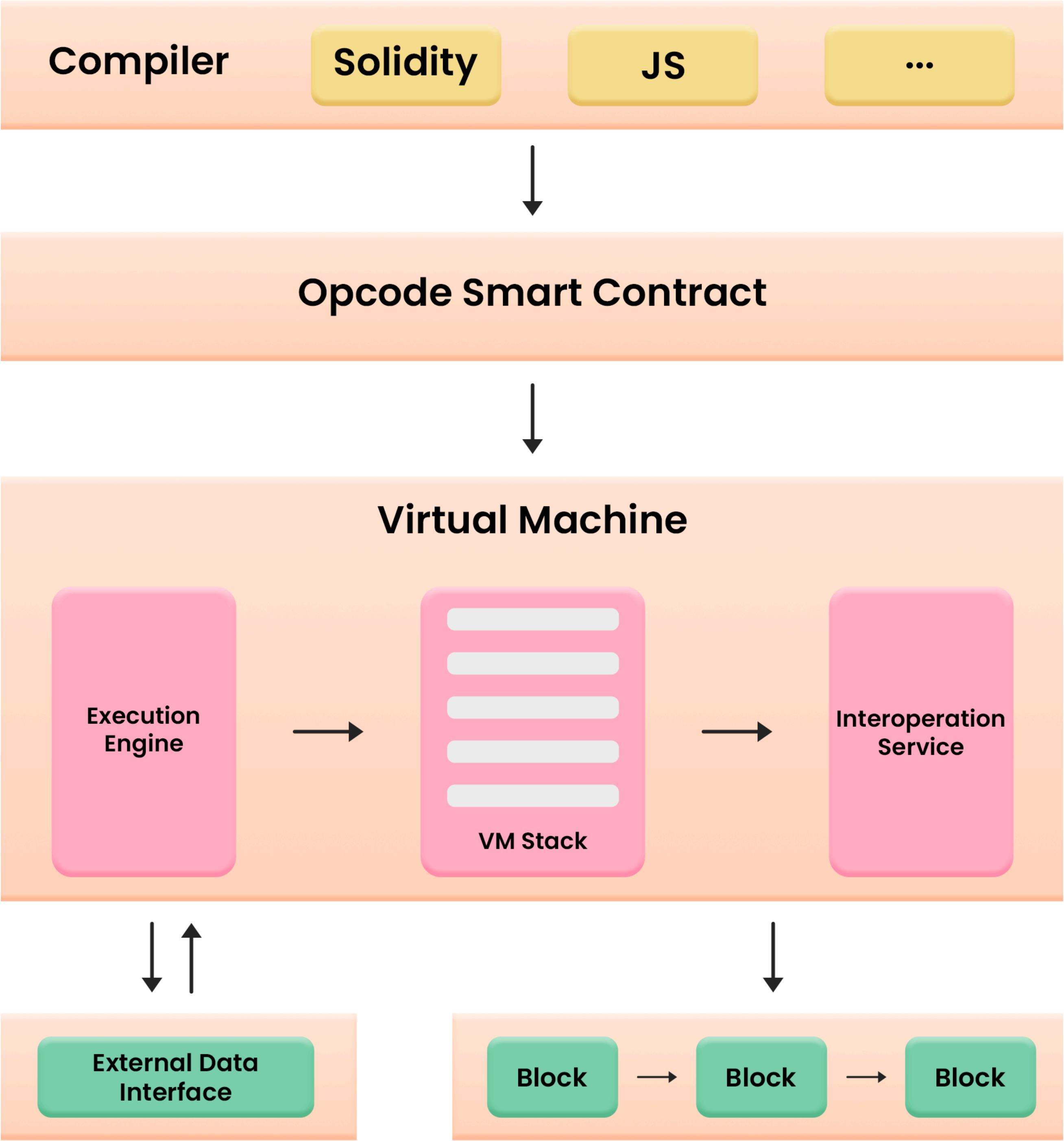
The POLLUX Virtual Machine (PVM) stands as a lightweight, Turing complete virtual machine tailored specifically for the POLLUX ecosystem. Its primary objective is to furnish a bespoke blockchain system characterized by efficiency, convenience, stability, security, and scalability.

Initially derived from the Ethereum Virtual Machine (EVM), PVM seamlessly integrates with the existing Solidity smart contract development ecosystem. Furthermore, PVM extends its support to the Delegated Proof of Stake (DPoS) consensus mechanism, enhancing its versatility and adaptability.

A distinguishing feature of PVM is its utilization of the concept of Energy. Unlike the Gas mechanism employed by EVM, transactions and smart contract operations on PVM incur no POX consumption, rendering them free. This departure from the Gas model ensures that executable computation capacity on PVM remains unrestricted by the total holding amount of tokens, fostering a more inclusive and efficient ecosystem.

6.2 Workflow

The compiler first translates the Solidity smart contract into bytecode readable and executable on the PVM. The PVM then processes data through opcode, which is equivalent to operating the logic of a stack-based finite state machine. Finally, the PVM accesses blockchain data and invokes the External Data Interface through the Interoperation layer.


Compiler
Solidity
JS
...
Opcode Smart Contract
Virtual Machine
Execution Engine
VM Stack
Interoperation Service
External Data Interface
Block
Block
Block

6.3 Performance

6.3.1 Lightweight Architecture

PVM adopts a lightweight architecture with the aim of reducing resource consumption to guarantee system performance.

6.3.2 Robust

POX transfers and smart contract execution cost bandwidth points only, instead of POX, which exempts POLLUX from being attacked. Bandwidth consumption is predictable and static since each computational step cost is fixed.

6.3.3 High Compatibility

PVM is compatible with EVM and will be compatible with more mainstream VMs in the future. Thereby, all smart contracts on EVM are executable on PVM.

6.3.4 Low Cost

Due to PVM's bandwidth setup, development costs are reduced and developers can focus on the logic development of their contract code. PVM also offers all-in-one interfaces for contract deployment, triggering, and viewing to offer convenience for developers.

7. Smart Contract

7.1 Introduction

A smart contract acts as a digital protocol that validates and executes contract negotiations autonomously. It establishes the terms, conditions, and penalties governing an agreement, and ensures their automatic enforcement. The code within a smart contract facilitates, verifies, and enforces the negotiation or execution of an agreement or transaction. In the realm of tokenization, smart contracts enable automatic funds transfers between involved parties upon meeting predefined conditions.

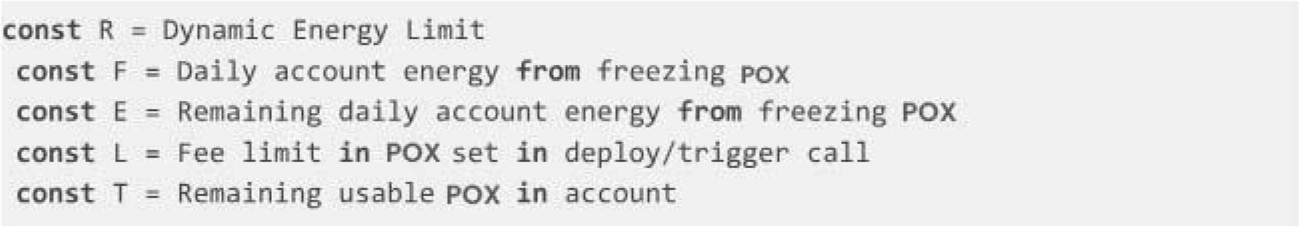
POLLUX smart contracts are authored in the Solidity programming language. After development and testing, they are compiled into bytecode and subsequently deployed onto the POLLUX network for execution by the POLLUX Virtual Machine. Once deployed, smart contracts can be interacted with using their unique contract addresses. The contract Application Binary Interface (ABI) provides a structured representation of the contract's callable functions and serves as the interface for external interaction with the network.

7.2 Energy Model

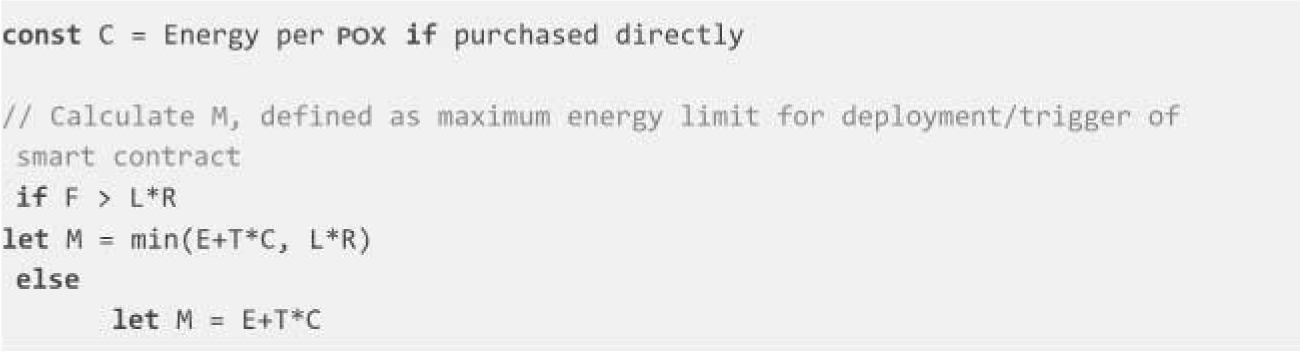
The maximum energy limit for deploying and triggering a smart contract is a function of several variables:

- Dynamic energy from freezing 1 POX is 50,000,000,000 (Total Energy Limit) / (Total Energy Weight)
- Energy limit is the daily account energy limit from freezing POX
- Remaining daily account energy from freezing POX is calculated as Energy Limit - Energy Used
- Fee limit in POX is set in smart contract deploy/trigger call
- Remaining usable POX in the account
- Energy per POX if purchased directly (10 RAM = 1 Energy) = 100,000, SRs can vote on adjustment

There are two consumption scenarios to calculate for maximum energy limit for deployment and trigger. The logic can be expressed as follows:



```
const R = Dynamic Energy Limit
 const F = Daily account energy from freezing POX
 const E = Remaining daily account energy from freezing POX
 const L = Fee limit in POX set in deploy/trigger call
 const T = Remaining usable POX in account
```



```
const C = Energy per POX if purchased directly

// Calculate M, defined as maximum energy limit for deployment/trigger of
 smart contract
 if F > L*R
let M = min(E+T*C, L*R)
 else
      let M = E+T*C
```

7.3 Deployment

When a POLLUX solidity smart contract undergoes compilation, the POLLUX Virtual Machine (PVM) interprets the resulting compiled bytecode. This bytecode is structured into three primary sections: code deployment, contract code, and Auxdata. The Auxdata acts as a cryptographic fingerprint of the source code, essential for verification purposes.

The deployment bytecode is responsible for executing the constructor function and initializing the initial storage variables of the contract. Additionally, this section of bytecode calculates the contract code and transmits it back to the PVM for further processing.

The Application Binary Interface (ABI) is a JSON file providing a comprehensive description of the functions within a POLLUX smart contract. It delineates the names of functions, their payability, return values, and state mutability. By leveraging the ABI, developers can interact with the smart contract and invoke its functions in a standardized and structured manner, ensuring seamless integration and interaction with the POLLUX network.

7.4 Trigger Function

Once smart contracts are deployed onto the POLLUX network, their functions can be activated individually through either POLLUXStudio or API calls. These functions can be categorized into two main types: state-changing functions and read-only functions.

7.5 POLLUX Solidity

POLLUX Solidity is a fork from Ethereum's Solidity language. POLLUX modifies the original project to support POX and RAM units (1 POX = 1,000,000 RAM). The rest of the language syntax is compatible with Solidity ^0.4.24. Thus the POLLUX Virtual Machine (PVM) is almost 100% compatible with EVM instructions.

8. Token

8.1 PRC-10 Token

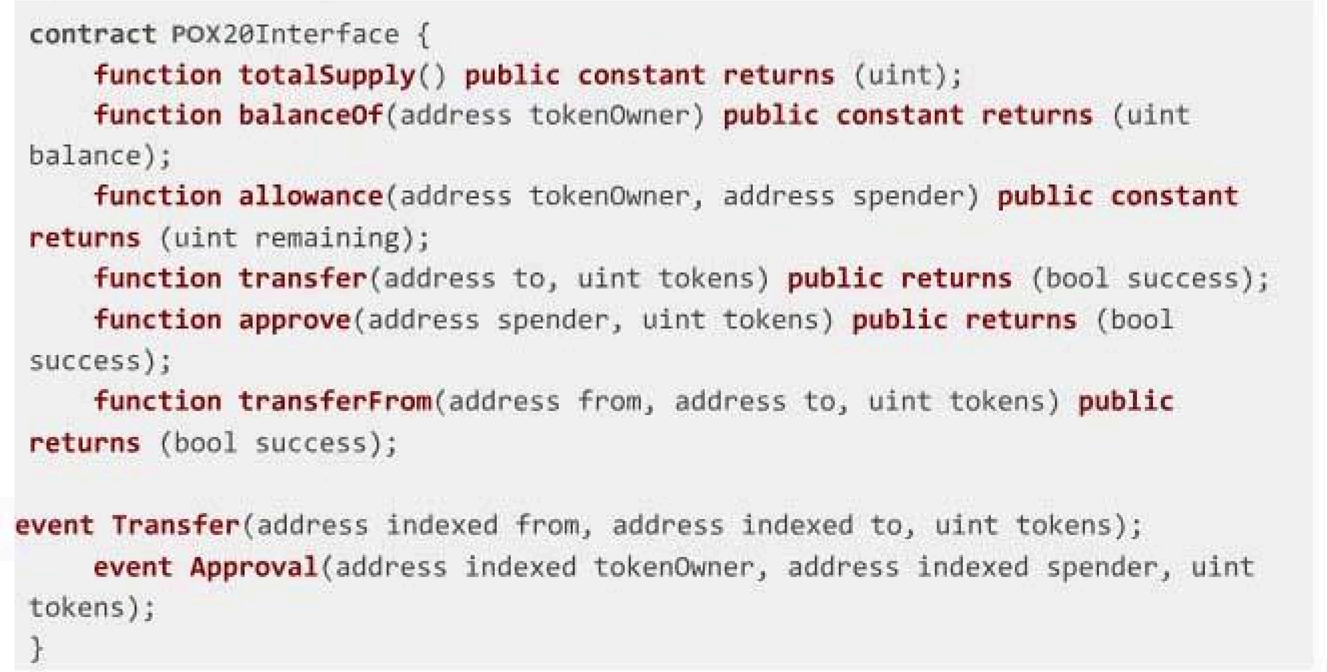
In the POLLUX network, each account has the capability to issue tokens, a process which incurs an expense of 1024 POX. When issuing tokens, the issuer is required to specify various parameters including the token name, total capitalization, exchange rate to POX, circulation duration, description, website, maximum bandwidth consumption per account, total bandwidth consumption, and the amount of token frozen.

Additionally, during token issuance, specific configurations can be set such as the maximum daily token transfer Bandwidth Points for each account, the overall network's maximum daily token transfer Bandwidth Points, total token supply, locking duration in days, and the total amount of tokens locked. These parameters help define the characteristics and behavior of the newly issued tokens, allowing for customization and control over their circulation and usage within the network.

8.2 PRC-20 Token

PRC-20 is a technical standard used for smart contracts implementing tokens supported by the POLLUX Virtual Machine. It is fully compatible with ERC-20.

The interface is as follows:



```
contract POX20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint
balance);
    function allowance(address tokenOwner, address spender) public constant
returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool
success);
    function transferFrom(address from, address to, uint tokens) public
returns (bool success);

event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint
tokens);
}
```

From a developer's standpoint, there are notable distinctions between PRC-10 and PRC-20 tokens. One significant difference lies in accessibility: PRC-10 tokens are accessible via both APIs and smart contracts, while PRC-20 tokens offer interface customization but are solely accessible within smart contracts.

Cost-wise, PRC-10 tokens present a considerable advantage in transaction fees, as they are 1000 times lower than those associated with PRC-20 tokens. However, PRC-10 tokens do entail bandwidth costs for API transfers and deposits. Conversely, transfers and deposits involving PRC-10 tokens within smart contracts incur both bandwidth and energy costs.

These differences underscore the importance of understanding the specific characteristics and functionalities of each token standard, allowing developers to make informed decisions based on their project requirements and cost considerations.

8.3 Beyond

Since POLLUX uses the same Solidity version as Ethereum, more token standards could be readily ported to POLLUX.

9. Governance

9.1 Super Representative

9.1.1 General

In the POLLUX network, every account has the opportunity to apply and potentially become a Super Representative (SR), denoted as SR. Individuals within the network can cast votes for their preferred SR candidates. The top 27 candidates receiving the most votes are designated as SRs, granting them the authority and responsibility to generate blocks. Voting occurs at intervals of every 6 hours, and the composition of SRs changes accordingly based on the latest vote counts.

To safeguard against malicious attacks and ensure the integrity of the SR election process, there is a cost associated with becoming an SR candidate. Upon applying for candidacy, 9999 POX will be deducted, effectively burned, from the applicant's account. Once this cost is paid, the account becomes eligible to participate in the SR election process and compete for a position as an SR. This mechanism serves as a deterrent against frivolous or disruptive candidacy applications, promoting a fair and secure SR election system within the POLLUX network.

9.1.2 Election

POLLUX Power (denoted as TP) is needed to vote and the amount of TP depends on the voter’s frozen assets (POX).

TP is calculated in the following way:

$$1\ TP\ = 1\ POX\ frozen\ to\ get\ bandwidth$$

Every account in the POLLUX network has the right to vote for their own SRs.

After the release (unfreeze, available after 3 days), users won't have any frozen assets and lose all TP accordingly. As a result, all votes become invalid for the ongoing and future voting round unless POX is frozen again to vote.

Note that the POLLUX network only records the most recent vote, which means that every new vote will negate all previous votes.

9.1.3 Reward

a. Vote Reward

Also known as Candidate Reward, which the top 127 candidates updated once every round (6 hours) will share 115,200 POX as mined. The reward will be split in accordance with the vote weight each candidate receives. Each year, the total reward for candidates will be 168,192,000 POX.

Total vote reward per round

Why 1,080 POX every round?

$$1{,}080\ POX = total\ vote\ reward\ per\ round\ (V\ R/round)$$

$$V\ R/round = 0.15\ POX/block \times 20\ blocks/min \times 60\ mins/hr \times 6\ hrs/round.$$

Total vote reward per year

Why 1,576,800 POX every year?

$$1{,}576{,}800\ POX = total\ vote\ reward\ per\ year\ (V\ R/year)$$

$$V\ R/year = 1{,}080\ POX/round \times 4\ rounds/day \times 365\ days/year$$

b. Block Reward

Also known as Super Representative Reward, the top 27 candidates (SRs) who are elected every round (6 hours) will share roughly 360 POX as mined. The reward will be split evenly between the 27 SRs (minus the total reward blocks missed due to network error).

Total block reward per round

Why 360 POX every round?

$$360\ POX = total\ block\ reward\ per\ round\ (BR/round)$$

$$BR/round = 0.05\ POX/bloc \times 20\ blocks/min \times 60\ mins/hr \times 6\ hrs/round$$

Notice: the unit block reward is set by WITNESS_PAY_PER_BLOCK = 32 POX. See dynamic network parameters.

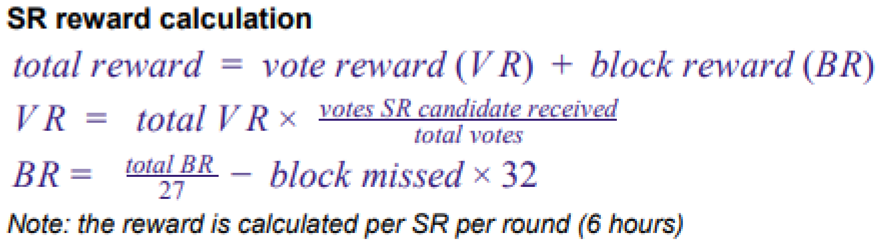
Total block reward per year

Why 525,600 POX every year?

$$525{,}600\ POX = total\ block\ reward\ per\ year\ (BR/year)$$

$$BR/year = 360\ POX/round \times 4\ rounds/day \times 365\ days/year$$

c. Reward Calculation



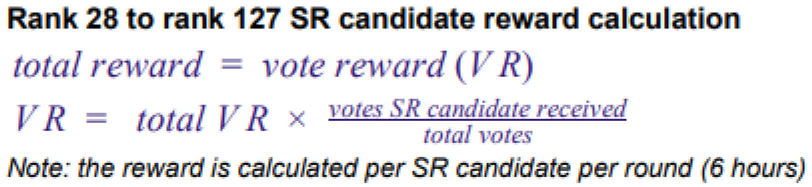
SR reward calculation

$$total\ reward = vote\ reward\ (VR) + block\ reward\ (BR)$$

$$VR = total\ VR \times \frac{votes\ SR\ candidate\ received}{total\ votes}$$

$$BR = \frac{total\ BR}{27} - block\ missed \times 32$$

Note: the reward is calculated per SR per round (6 hours)



Rank 28 to rank 127 SR candidate reward calculation

$$total\ reward = vote\ reward\ (VR)$$

$$VR = total\ VR \times \frac{votes\ SR\ candidate\ received}{total\ votes}$$

Note: the reward is calculated per SR candidate per round (6 hours)

9.2 Committee

9.2.1 General

The committee within the POLLUX network plays a crucial role in modifying dynamic network parameters, which include block generation rewards, transaction fees, and other relevant settings. Comprised of the 27 Super Representatives (SRs) currently active in the network, the committee grants each SR the authority to both propose and vote on various proposals.

When a proposal garners 19 votes or more from the SRs, it is considered approved. Subsequently, the newly agreed-upon network parameters are scheduled to be implemented in the subsequent maintenance period, which occurs every 3 days. This democratic process ensures that any changes to the network's operational parameters are thoroughly considered and endorsed by the majority of the SR committee, promoting transparency and stability within the POLLUX ecosystem.

9.2.2 Dynamic Network Parameters

0. MAINTENANCE_TIME_INTERVAL
   a. Description
      Modify the maintenance interval time in ms. Known as the SR vote interval time per round.
   b. Example
      [6 * 3600 * 1000] ms - which is 6 hours.
   c. Range
      [3 * 27* 1000, 24 * 3600 * 1000] ms

1. ACCOUNT_UPGRADE_COST
    a. Description
    Modify the cost of applying for an SR account.
    b. Example
    [9,999,000,000] RAM - which is 9,999 POX.
    c. Range
    [0,100 000 000 000 000 000] RAM
2. CREATE_ACCOUNT_FEE
    a. Description
    Modify the account creation fee.
    Example
    [10,000] RAM - which is 0.1 POX.
    b. Range
    [0,100 000 000 000 000 000] RAM
3. TRANSACTION_FEE
    a. Description
    Modify the amount of fee used to gain extra bandwidth.
    b. Example
    [10] RAM/byte.
    c. Range
    [0,100 000 000 000 000 000] RAM/byte
4. ASSET_ISSUE_FEE
    a. Description
    Modify asset issuance fee.
    b. Example
    [1024,000,000] RAM - which is 1024 POX.
    c. Range
    [0,100 000 000 000 000 000] RAM
5. WITNESS_PAY_PER_BLOCK
    a. Description
    Modify SR block generation reward. Known as a unit block reward.
    b. Example
    [50,000] RAM - which is 0.05 POX.
    c. Range
    [0,100 000 000 000 000 000] RAM
6. WITNESS_STANDBY_ALLOWANCE
    a. Description
    Modify the rewards given to the top 127 SR candidates. Known as the total vote reward per round.
    b. Example
    [150,000] RAM - which is 0.15 POX.
    c. Range
    [0,100 000 000 000 000 000] RAM

7. CREATE_NEW_ACCOUNT_FEE_IN_SYSTEM_CONTRACT
    a. Description
    Modify the cost of account creation. Combine dynamic network parameters #8 to get total account creation cost:
    $CREATE\_NEW\_ACCOUNT\_FEE\_IN\_SYSTEM\_CONTRACT \times CREATE\_NEW\_ACCOUNT\_BANDWIDTH\_RATE$
    b. Example
    [0] RAM.
    c. Range
    [0,100 000 000 000 000 000] RAM
8. CREATE_NEW_ACCOUNT_BANDWIDTH_RATE
    Description
    Modify the cost of account creation. Combine dynamic network parameters #7 to get total account creation cost:
    $CREATE\_NEW\_ACCOUNT\_FEE\_IN\_SYSTEM\_CONTRACT \times CREATE\_NEW\_ACCOUNT\_BANDWIDTH\_RATE$
    a. Example
    [1].
    b. Range
    [0,100,000,000,000,000,000]
9. ALLOW_CREATION_OF_CONTRACTS
    a. Description
    To turn on POLLUX Virtual Machine (PVM).
    b. Example
    True -
    c. Range
    True/False
10. REMOVE_THE_POWER_OF_THE_GR
    a. Description
    Remove the initial GR genesis votes
    b. Example
    True -
    c. Range
    True/False - Notice: cannot set back to False from True.
11. ENERGY_FEE
    a. Description
    Modify the fee of 1 energy.
    b. Example
    20 RAM.
    c. Range
    [0,100 000 000 000 000 000] RAM
12. EXCHANGE_CREATE_FEE
    a. Description
    Modify the cost of trading pair creation. Known as the cost of creating a trade order.
    b. Example

   [1,024,000,000] RAM - which is 1024 POX.

   c. Range

   [0,100 000 000 000 000 000] RAM

13. MAX_CPU_TIME_OF_ONE_TX

   a. Description

   Modify the maximum execution time of one transaction. Known as the timeout limit of one transaction.

   b. Example

   50 ms.

   c. Range

[0, 1000] ms

14. ALLOW_UPDATE_ACCOUNT_NAME
    a. Description
    Modify the option to let an account update their account name.
    b. Example
    False - which is available to propose from java-POLLUX PoxChain.
    c. Range
    True/False - Notice: cannot set back to False from True.
15. ALLOW_SAME_TOKEN_NAME
    a. Description
    Modify the validation of allowing different tokens to have a duplicate name.
    b. Example
    False - which is available to propose from java-POLLUX PoxChain.
    c. Range
    True/False - Notice: cannot set back to False from True.
16. ALLOW_DELEGATE_RESOURCE
    a. Description
    Modify the validation of allowing to issuing token with a duplicate name, so the **tokenID** of the token, in long integer data type, would be the only atomic identification of a token.
    b. Example
    False - which is available to propose from java-POLLUX PoxChain.
    c. Range
    True/False - Notice: cannot set back to False from True.
17. TOTAL_ENERGY_LIMIT
    a. Description
    Modify the whole network's total energy limit.
    b. Example
    [50,000,000,000,000,000] RAM - which is 50,000,000,000 POX.
    c. Range
    [0,100,000,000,000,000,000] RAM
18. ALLOW_PVM_TRANSFER_PRC10
    a. Description
    Allow PRC-10 token transfer within smart contracts.
    ALLOW_UPDATE_ACCOUNT_NAME, ALLOW_SAME_TOKEN_NAME, ALLOW_DELEGATE_RESOURCE proposals must all be approved before proposing this parameter change.
    b. Example
    False - which is available to propose from java-POLLUX PoxChain.
    c. Range
    True/False - Notice: cannot set back to False from True.

9.2.3 Create Proposal

Only the SR accounts have the right to propose a change in dynamic network parameters.

9.2.4 Vote Proposal

Only committee members (SRs) can vote for a proposal and the member who does not vote in time will be considered as a disagree. The proposal is active for 3 days after it is created. The vote can be changed or retrieved during the 3-day voting window. Once the period ends, the proposal will either succeed (19+ votes) or fail (and end).

9.2.5 Cancel Proposal

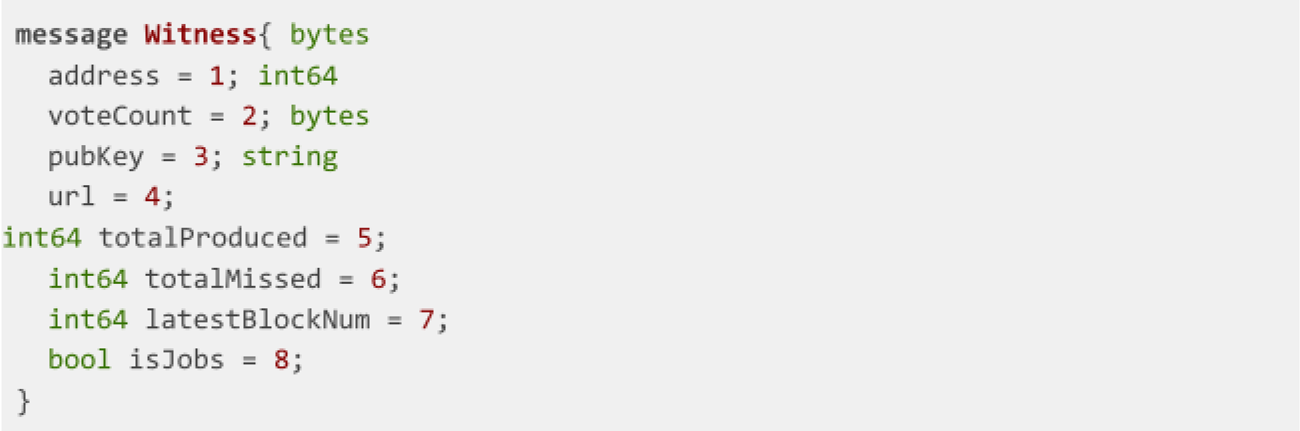
The proposer can cancel the proposal before it becomes effective.

9.3 Structure

SRs are the witnesses of newly generated blocks. A witness contains 8 parameters:

1. **address**: the address of this witness – e.g. 0xu82h…7237.
2. **voteCount**: number of received votes on this witness – e.g. 234234.
3. **pubKey**: the public key for this witness – e.g. 0xu82h…7237.
4. **url**: the url for this witness – e.g. https://www.noonetrust.com.
5. **totalProduced**: the number of blocks this witness produced – e.g. 2434.
6. **totalMissed**: the number of blocks this witness missed – e.g. 7.
7. **latestBlockNum**: the latest height of block – e.g. 4522.
8. **isjobs**: a boolean flag.

Protobuf data structure:



```
message Witness{ bytes
  address = 1; int64
  voteCount = 2; bytes
  pubKey = 3; string
  url = 4;
int64 totalProduced = 5;
  int64 totalMissed = 6;
  int64 latestBlockNum = 7;
  bool isJobs = 8;
}
```

10. DApp Development

10.1 APIs

The POLLUX network provides developers with an extensive array of options for interacting with its blockchain through over 60 HTTP API gateways. These gateways facilitate communication with the network via Full and Solidity Nodes, offering diverse functionalities and capabilities.

Moreover, developers can leverage POLLUXWeb, a comprehensive JavaScript library encompassing a wide range of API functions. This library empowers developers to perform various tasks such as deploying smart contracts, modifying the blockchain state, querying blockchain and contract data, executing trades on the decentralized exchange (DEX), and much more.

These API gateways are versatile and can be directed towards different environments, including a local privatenet, the Yuvi testnet, or the POLLUX Mainnet. This flexibility allows developers to seamlessly integrate with the POLLUX network and build decentralized applications (DApps) tailored to their specific needs and preferences.

10.2 Networks

POLLUX offers developers access to both its Yuvi testnet and Mainnet, providing environments for testing and deploying decentralized applications (DApps). Developers can connect to these networks by deploying their own nodes, interacting through the user-friendly interface of POLLUXStudio, or utilizing APIs via the POLLUXGrid service.

The POLLUXGrid service is a key component, comprising load-balanced node clusters hosted on AWS servers distributed globally. This infrastructure ensures high availability and reliability, allowing developers to access the network seamlessly from various locations worldwide. As DApp development progresses and API call volumes grow, POLLUXGrid efficiently manages the increased traffic, ensuring smooth and uninterrupted operation for developers.

10.3 Tools

POLLUX provides developers with a comprehensive suite of development tools aimed at facilitating the creation of innovative decentralized applications (DApps). At the core of this toolkit is POLLUXBox, a framework enabling the testing and deployment of smart contracts via the

POLLUXWeb API. Complementing this, POLLUXGrid offers a hosted API service, ensuring seamless access to both the Yuvi testnet and the POLLUX Mainnet without the need for individual node operation. For developers seeking an integrated development environment (IDE), POLLUXStudio offers a robust solution, featuring tools for compiling, deploying, and debugging Solidity smart contracts, alongside an internal full node for local testing. Furthermore, the POLLUXWeb API library simplifies network connectivity through a diverse range of HTTP API calls wrapped in JavaScript. Together, these tools empower developers to efficiently build and deploy DApps on the POLLUX network, fostering innovation and advancement within the blockchain ecosystem.

10.4 Resources

The POLLUX Developer Hub is a comprehensive API documentation site tailored towards developers wishing to build on the POLLUX network. The Developer Hub provides a high-level conceptual understanding of POLLUX and walks users through the details of interacting with the network. The guides walk developers through node setup, deployment and interaction with smart contracts, API interaction and implementation, building sample DApps, and using each of the developer tools. Additionally, developer community channels are available through Discord.

11. Conclusion

POLLUX is a scalable blockchain solution that has employed innovative methods for tackling challenges faced by legacy blockchain networks. Having reached over 2M transactions per day, with over 700K POX accounts, and surpassing 2000 TPS, POLLUX has enabled the community to create a decentralized and democratized network.