# BitYuan Whitepaper

## (ver 5.0)

A Simple、Stable and Scalable Blockchain Network

2024-4-6

# Directory

# Foreword

At present, the timestamp system, with features of distributed bookkeeping and token incentives, and represented by Bitcoin (BTC), is widely considered to be the pillar of future finance. For an emerging technology, only by continuously upgrading and optimizing functions and performance can it be recognized by the market and then extensively applied, leading the transformation in an era. BitYuan (BTY) is designed to achieve decentralized governance, allowing coin holders to formulate relevant rules. With sufficient development funds, it can mobilize and inspire the entire society to promote its development.

The core of BTY is just as stable as BTC's, coupled with flexible and efficient scalability. Developers can build powerful decentralized applications (Dapps) and multi-chain ecosystems on BTY to jointly maintain the development of BTY system. The underlying system of BTY, supported by FUZAMEI Chain33, is the very first public-blockchain project in the world to empower and implement a multi-chain (parallel public blockchain) architecture. A plurality of parallel public blockchains can be extended on BTY, while each of them can independently develop DApps, construct diversified application ecologies, and enable cross-chain swap.

Parallel public blockchains may be applied to a variety of scenarios, including stable coins, social networking, e-commerce, digital asset, on-chain creditor's rights, data storage certification, blockchain games and more.

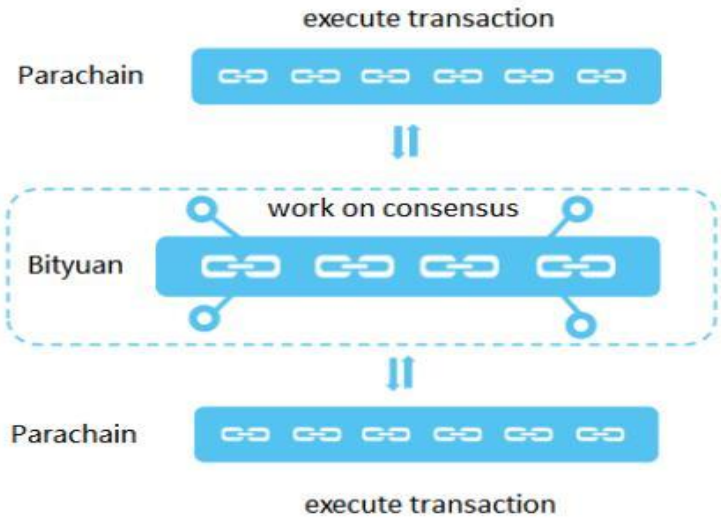
BTY aims to establish a public blockchain alliance that provides basic services for each parallel public blockchain, and BTY itself is the fuel necessary for all transaction.

# Chapter I Underlying Architecture

## 1.1 System Profile

The underlying system of BTY adopts a blockchain architecture that supports modules such as consensus, database, and actuator, and is pluggable and easy to upgrade. Such a layered architecture is creative, where the main chain is responsible for transaction clearing, while smart contracts and virtual machines are separated from the main chain and run on parallel blockchains. Multiple parallel blockchains coexist to improve computational efficiency, and are interconnected by the main chain.

The modular design of BTY is based on the analysis over blockchain′ s underlying architecture and the different functions and requirements raised by application development. A series of functional modules are thus divided and designed, offering diversified products through the selection and combination of modules so as to meet varied market needs.




BTY follows the "chaos-order development mode", which makes it easy for developers to adjust and expand at any time. From the perspectives of iteration, reconstruction, and system scalability, BTY takes into account the underlying architecture of blockchain, and the different functions and requirements raised by application development, and modularizes the system. All modules at the core of BTY are open for customization, including

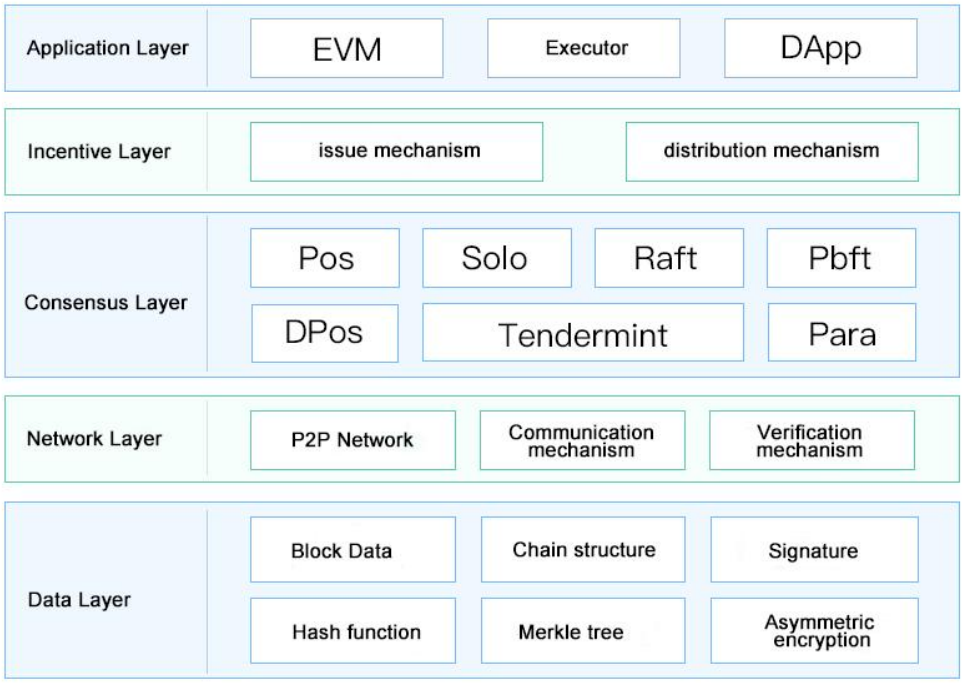
Mempool queuing model, cryptographic signature methods, consensus modes, RPC functions, commands in command line, wallet's internal logic, database storage modes, etc.

## 1.2 Overall Architecture

Chain33 is a blockchain underlying development platform, supporting consensus, database, executor and other modules pluggable and easily upgradeable blockchain architecture.

The module plug-in design has many benefits, it can strip the extended functionality from the system framework, reduce the complexity of the framework, and make the framework easier to implement. Chain33 provides several functional modules that are collected to form a module plug-in repository.

The design architecture of module plug-in is to be able to build blockchain more easily. A single module does not show its advantages. If there are different modules, it is possible to build chains with different attributes and provide customers with personalized choices for building chains. Such an approach allows customers to devote more energy to their business rather than to the development of the underlying blockchain technology.



### 1.2.1 Module introduction

Chain33 adopts a layered module plug-in design, the modules are pluggable, adapt to a variety of scenarios, and help customers to quickly change the chain. The core modules includes:

(1) **DApplication layer**：Used to implement business logic on the blockchain.

(2) **Smart contracts layer**： Intelligent execution of pre-agreed contract terms, along with support for EVM, WASM and go language for smart contract writing, enables Turing-complete computational operations. Among them, EVM is responsible for parsing and executing contract operations, while RPC provides external access capabilities.

(3) **Core layer**： It is divided into blockchain protocol, consensus algorithm, mining management, and distributed network core components, of which the consensus algorithm uses Chain33's self-developed SPOS.

### 1.2.2 Core layer introduction

**Blockchain protocol**：Used for read and write support to the state database and local database during block execution (including data such as blocks, transactions and state), as well as responding to query operations on the data.

**Consensus algorithm**：Running the SPOS consensus protocol.

**Mining management**：Manage the life cycle of tickets, support agent mining, and improve mining security.

**Distributed network management**： Enables data interaction with other blockchain nodes, including broadcasting and receiving transaction, block and shatding data, as well as proactively requesting data from other nodes.

**Underlying**： Including p2p, encryption, database and other basic modules.

(1) P2P module: Gossip protocol and dht protocol are both supported.

(2) Encryption module: Compatible with international standards and national secret standards (SM2、SM3、SM4、SECP256K1、ED25519).

(3) Data storage module: The data storage method supports scalability and storage sharding, and currently supports MPT, MAVL, KVDB, MVCCKVDB, among which MVCCKVDB can avoid the problem of performance degradation due to state tree expansion, and supports LevelDB, ES and other back-end storage.

### 1.2.3 Performance Enhancement

To improve the overall performance of the system, Chain33 has been optimized in two areas:

**Calculation fragmentation：** Chain33 adopts the architecture of "main chain + parallel public chain", the main chain is responsible for transaction clearing and deposition, smart contracts and virtual machines are separated from the main chain and put on parallel chains for independent execution, multiple parallel chains co-exist and execute transactions belonging to their respective parallel chains independently to achieve parallel execution of transactions, thus improving the transaction processing capacity of the system and increasing tps. The parallel chains are interconnected by the main chain.

**Storage fragmentation：** Storage sharding is mainly based on kad network. The sharding algorithm can ensure that the data is relatively evenly distributed on each node of the blockchain, and each node only needs to keep a part of the data, which is ideal for the scenario of massive data storage, and the machine can be added at any time to realize dynamic expansion. Distributed storage after data packaging reduces the amount of data and avoids too much data fragmentation to increase the network load.

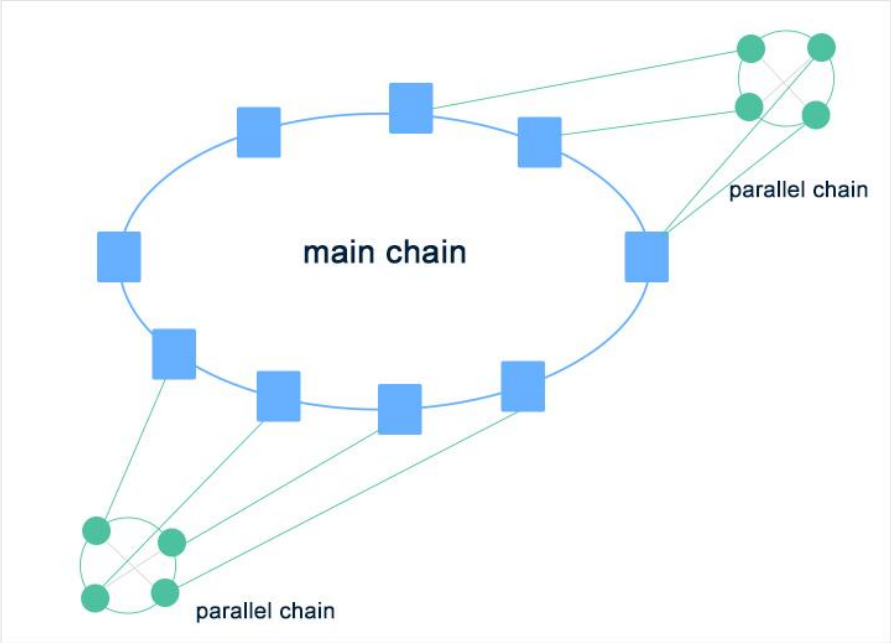
## 1.3 "Main chain + Parallel public chain" Architecture

Parallel chain, known as parallel public chain, is an independent public chain that share the BitYuan consensus network and are connected to the main chain node through a grpc interface. Theoretically, there are countless parallel public chains that can be mounted on the BitYuan main chain.

The main chain is very stable and is responsible for consensus and storage, as well as running the core blockchain contract; the data between parallel public chains is isolated and does not interfere with each other, and each parallel public chain only runs its own data. Multiple parallel public chains coexist to achieve parallel execution of transactions.

Transactions on the parallel chain are sent to the main chain to be packaged by consensus, then synchronized to the parallel chain to be executed, and finally the execution results are written back to the main chain for deposition, so as to achieve

shared network security while separating consensus and transaction execution, realizing parallel execution of transactions and improving TPS.




### 1.3.1 Parallel public chain advantages

(1) **Low cost:** The minimum requirement for a parallel chain node server is one and the specifications are not high.

(2) **Fast development:** Creating a parallel chain requires only modifying the configuration file and executing a few CLI commands, which is not technically demanding and efficient to build.

(3) **High compatibility:** Supports deployment of smart contracts in multiple programming languages, including Solidity, Java, C++ and Golang, and other mature languages.

(4) **Stable and secure:** Even if the parallel chain is damaged or under attack, the data can be quickly synchronized from the main chain to ensure data security.

(5) **Parallel execution of transactions:** data isolation between parallel public chains, no interference with each other, multiple parallel public chains coexist and transactions are executed in parallel.

### 1.3.2 Parallel public chain application areas

Parallel Chain has its own blockchain ecology with a wide range of application areas, including different scenarios such as stable coins, social networks, e-commerce, asset digitization, debt on the chain, data deposition, contract games and so on.

Developers can freely build parallel public chains and support R&D testing work based on parallel chains such as issuing digital assets, customizing smart contracts, creating super nodes, and docking external blockchain applications.

### 1.3.2 Parallel public chain cross-chain transactions

The "Main chain + Parallel public chain" layered architecture of BitYuan naturally supports cross-chain transactions between the main chain and parallel chains, and between parallel chains and parallel chains; through a variety of cross-chain solutions combined with parallel chains, the value transfer between BitYuan's main chain and external heterogeneous public chains is realized.

Parallel chains and main chains belong to isomorphic chains, only the titles and forks are different. Parallel chains can only complete the transfer of assets across chains by creating consensus account groups, i.e. transferring assets from main chains to parallel chains or from parallel chains to main chains.

## 1.4 System Module

- **P2P Module**

P2P network is a peer-to-peer network in which each node in the network participates and shares the storage capacity, broadcast capacity, and network connectivity that it possesses. Nodes in the network can be accessed directly by peer nodes, and each node is both a consumer and a provider of resources and services.

P2P network is a decentralized network where there is no central node for the whole network, unlike traditional central networks where all nodes depend on a central node.

P2P module is the component that handles nodes' communication with each other, including nodes discovering and establishing connections, broadcasting blocks and transactions, and synchronizing blocks.

- **Storage Module**

BitYuan underlying technology, Chain33, uses high-performance and highly reliable KV database for blockchain data storage, supporting goleveldb, gobadgerdb, gomemdb, gossdb and other KV database types.

The storage format of the state data in the blockchain is configurable and pluggable, and can support the storage formats of mavl, mpt, and kvmvcc.

- **Blockchain Module**

The Blockchain module is a component that handles blocks, including storing block information, adding blocks to the main chain or parallel public chain, and synchronizing blocks; it also provides an interface for querying block and blockchain status information to the outside world.

- **Mempool Module**

Mempool module, i.e. transaction cache pool, is mainly intended to solve the problem that the consensus module may be slower than the RPC module.

The Mempool module is responsible for doing initial legitimacy checks on incoming transactions and filtering out some illegal transactions; at the same time, it does traffic restrictions on the sender of transactions to prevent the same address from sending too often.

- **Consensus Module**

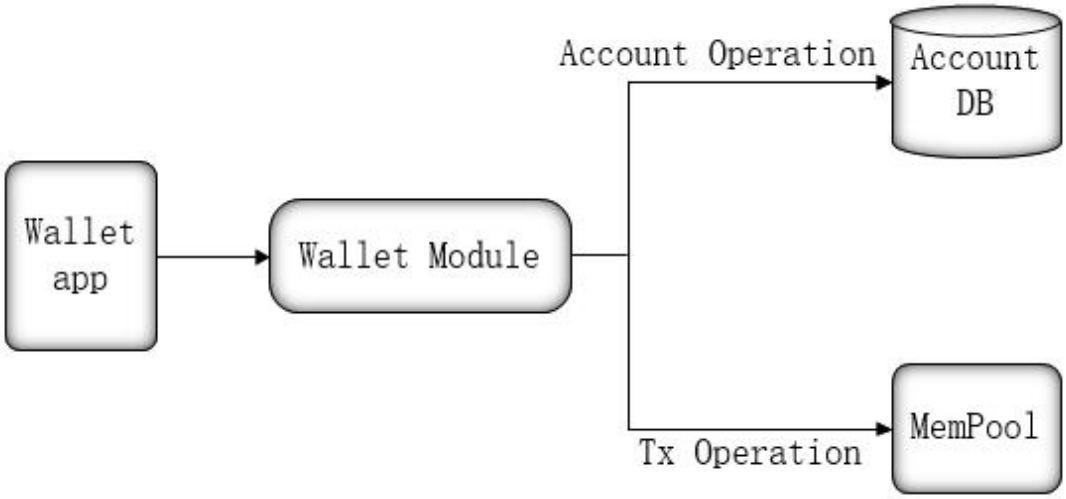
The module that implements the blockchain consensus mechanism is the most important component of the blockchain technology. The purpose of the blockchain consensus mechanism is to make all nodes in the blockchain network agree on which node generates a new block through a consensus algorithm, thus ensuring a consistent block state for the whole system.

The consensus algorithm of the BitYuan mainnet is SPOS, a secure POS algorithm that supports tens of thousands of people mining together for consensus. Parallel public chains in the BitYuan ecosystem can also have their own independent antecedent consensus, using the strongly consistent Byzantine consensus algorithm, and introducing the concept of DPOS voting rights, where super nodes must pack transaction information into blocks and then broadcast the block information to other nodes, storing the transaction information on the blocks to function as a shared governance community.

- **Wallet Module**

The Wallet module provides two main services for the BitYuan Wallet App.

- One is blockchain account management, including account creation, management of public and private keys and address information. The wallet client can create and manage accounts, export and import private keys, etc. through these service interfaces.
- The other is transaction signing and sending, for bty transfer and smart contract transaction sending.




- **Actuator Module**

The actuator module is the logical processing center of the blockchain and is used to execute specific business logic.

An executor is usually understood as a container that supports the operation of various contracts (including system contracts as well as various extension contracts); receives transactions and assigns them to the corresponding contracts for specific execution; and responds to various events on the blockchain (transaction checks, query requests, executed transactions, added blocks, rolled back blocks) and dispatches the events to specific contract logic.

Take a smart contract, all the logic contained in the contract is the logic to be executed by the executor; only, the contract is just a specific logic, while the executor is a container that can execute several contracts without needing to know the specific content of the contract.

- **Queue Module**

Queue module, i.e. message queue module, the main purpose of this module is to reduce the coupling of various modules in the Chain33 system of the blockchain.

With the introduction of Queue module, communication between modules is not through interface calls, but through messages, each module is relatively independent, reducing the coupling of each module in the chain33 system, which also facilitates the splitting of chain33 by module and paves the way for the microservicing of each module later.

- **RPC Module**

RPC, i.e. Remote Procedure Call, the RPC module mainly provides external programs with transaction construction and sending, block, transaction and other information query interfaces.

The RPC module supports both grpc and jsonrpc interfaces, other processes can call both grpc and jrpc interfaces, in general, if a process can access the internal data structure of chain33 to construct data, the grpc interface is preferred, and vice versa, the jrpc interface is recommended.

# Chapter II Technical Features

## 2.1 SPOS Consensus

The consensus module is the module that implements the blockchain consensus mechanism, which is an important component of blockchain technology. The goal of the blockchain consensus mechanism is to enable all honest nodes to preserve a consistent view of the blockchain. In layman's terms, the consensus mechanism plays a role in deciding who is responsible for generating new blocks and maintaining the unity of the blockchain within the blockchain network. BitYuan uses an innovative SPOS consensus, the Safe POS consensus mechanism.

The innovation and application of SPOS are mainly in two aspects.

- POS consensus is achieved through the Ticket ballot mechanism.
- Generate fair random numbers during block generation.

### 2.1.1 Achieving consensus through ballot mechanisms

BitYuan implements the secure mining logic of POS through Ticket. Users use the balance of BTY in their wallet accounts to purchase tickets and obtain mining rights (currently 3000 BTY corresponds to 1 ticket), one ticket corresponds to a unique Ticket ID; each block will only be mined by one ticket, and the actual mining odds are equally divided among the tickets (if there are N tickets in the whole network, the odds of each ticket being mined are 1/N).

The ballot mining process is shown below:

(1) Periodically check the BTY balance in your wallet to purchase tickets, and when the purchase conditions are met, construct a ticket purchase transaction and send it to the blockchain.

(2) SPOS consensus will always try to use locally held tickets to pack blocks. Once the packing is successful, it means that the corresponding Ticket holder is successful in mining and gets the corresponding block reward.

(3) The smart contract will write the Ticket information corresponding to the address to the blockchain database, and each Ticket corresponds to a unique Ticket ID, and there will also be a data record in the database.

**Note:** Malicious nodes that attempt to fork BitYuan, or any malicious behavior that the system can detect, may be penalized with a 20% loss of assets per penalty.

Mining must be done with standard wallets published by the BitYuan Foundation, and tampering with mining behavior that is automatically determined to be malicious by the system can result in significant losses to miners.

SPOS consensus features:

(1) Compared with POW, POC and other consensus mechanisms eliminate unnecessary resource waste in the consensus process, such as storage waste, arithmetic waste, etc.
(2) The smart contract performs 1/N ballot logic to ensure a fair and equitable block out.
(3) The penalty mechanism is designed to ensure the positive operation of miners and the safety of the main network operation.

### 2.1.2 Generate fair random numbers

In application scenarios such as games, a random number that cannot be predicted is needed on the blockchain to ensure fairness and equity in order to reflect fairness. Similarly, BitYuan is able to generate a fair random number during block generation to guarantee the security of random number calls for on-chain applications.

The current blockchain has the following solutions.

(1) Invocation of an external centralized random number generator in the contract to obtain random numbers.
(2) Use some value in the block hash as a random number.

But both options have very obvious drawbacks.

(1) If the contract reads data from outside, it is very likely to get different results (for example, the network causes some nodes to read properly and some to return errors), which will lead to forking.
(2) The hash of a block can be controlled, leading to the random number being controlled. Since some blockchain systems do not provide good random number algorithms, many DApp developers will package their own random number algorithms that they think are perfect, resulting in random numbers being predicted.

To address the above two drawbacks, BitYuan implements the optimization of random number generation: BitYuan introduces VRF (Verifiable Random Functions) verifiable random function logic on top of the original random numbers, which makes the randomness further enhanced.

### 2.1.3 Implementation logic of random number generation

(1) Users purchase tickets by using the BTY in their wallet account. 3000 BTY corresponds to one ticket.

The wallet will generate a randNum, hashed and then combined with the private key of the wallet mining address, the ticket corresponding to the index (you can buy more than one ticket at a time) and other elements to do two more hashes to get a public hash parameter pubHash: pubHash=hash(hash(privateKey:index:hash (randNum))).

(2) Newly purchased tickets contain pubHash and randNum and are deposited into the blockchain. Tickets require a 12-hour maturation period, after which they can participate in mining.

(3) The consensus algorithm finds already mature Tickets from the blockchain and starts packaging them.

Since the consensus packing block operation is only performed locally at the node, it can read the locally stored private key, compute a private hash (privHash) and put it into the mining transaction: privHash = hash(privateKey:index:hash(randNum))

(4) The smart contract receives the mining transaction and compares the values of hash(privHash) and pubHash, and the two agree that the mining transaction is successful and the corresponding node gets the mining reward, otherwise the execution of the mining transaction fails.

**Key Terms**

VRF: Verifiable Random Functions

SK, PK: The public-private key pair used in the VRF, SK is the private key, PK is the public key

M: Input Data

R: VRF Output Hash

P: VRF Certification

Prover: The certifier, who has the public and private keys of the VRF (PK and SK)

Verifier: The authenticator, who has the public key (PK) in the VRF

**Function Groups**

Four functions are included in generating random numbers, which are mainly divided into two categories: generating functions and validating functions.

Generating Functions:

R=VRF_Hash(SK,M)

P=VRF_Proof(SK,M)

Validating Functions:

R=VRF_P2H§

VRF_Verify(PK,M,P)

**VRF Usage Process**

(1) The provers generate a pair of keys (PK and SK).

(2) The provers calculate R and P (R = VRF_Hash(SK, M), P = VRF_Proof(SK, M)).

(3) The prover submits R, P, PK, M to the verifier.

(4) Verifier calculates that satisfies VRF_P2H(P) = R and VRF_Verify(PK, M, P) = True

If the conditions are met, the verification is passed, otherwise the verification is not passed.

### 2.1.4 BTY process for implementing randomness using VRF

(1) SK corresponds to the private key of the mining node account, PK corresponds to the public key of the mining node account.

(2) At that height of the BitYuan upgrade fork in 2019, the mining node first reads the privHash stored in the previous block as the input M value and calculates the values of R and P, respectively, through the VRF generation function.

(3) The mining node writes R and P to the mining transaction, packs them into blocks and broadcasts them.

(4) All nodes receive the block and verify the correctness by the VRF verification function in the smart contract. If VRF_P2H(P) = R and VRF_Verify(PK, M, P) = True, the verification passes, otherwise the verification fails.

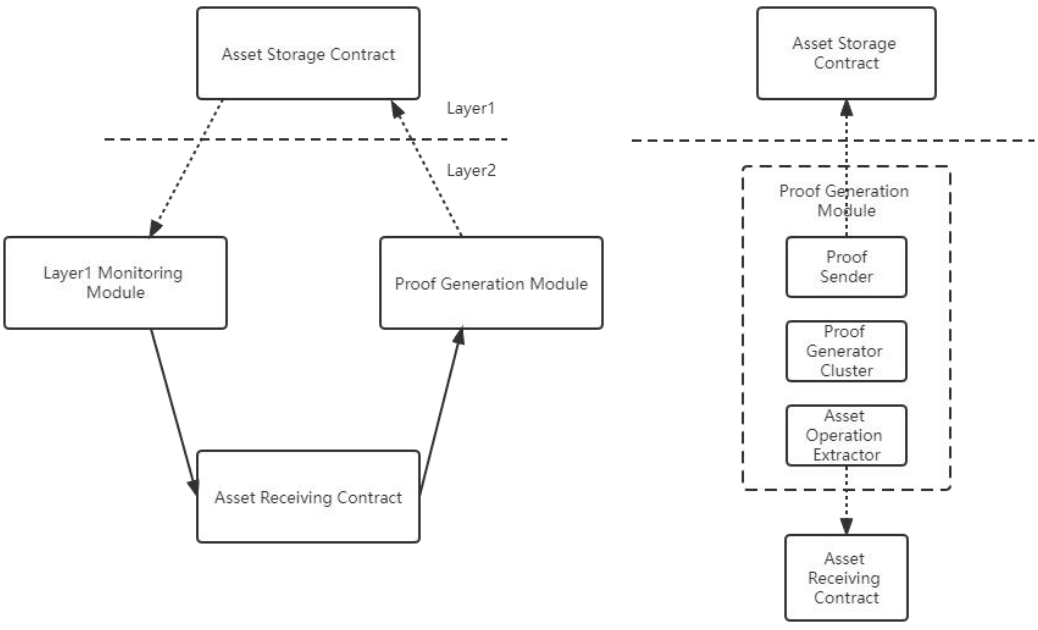
(5) For further blocks, the R stored in the previous block is read as input M to calculate R and P.

It can be seen that the random numbers introduced in the current block all need to depend on the random value R of the previous block, the random value is less easily controlled and the randomness is greatly enhanced.Thus, the implementation of SPOS consensus combines random number. Since it is generally impossible to predict the consensus information of other nodes, it is also impossible to obtain its consensus random number. And the system sets the private hash (privHash) cannot be leaked in advance, even if a malicious miner exposes himself in advance, its corresponding ticket will be invalidated, while the principal will be frozen for a longer period of time (more than 2 days). In addition, the system sets a 12-hour maturity period before the ticket can be involved in mining, and the introduction of VRF verifiable random function implementation in the consensus logic, the random value of each block depends on the random number R of the previous block, and the randomness is further enhanced.

The combination of these conditions makes the system's random number almost impossible to manipulate. Thus, when a developer implements a DApp that requires fair randomness, it can directly use this safe random number provided by the system.

## 2.2 Zero-Knowledge Proof

It adopts the zkSync solution to realize cross-chain and asset transfer. The overall structure is shown in the figure below, including the four modules: asset storage contract, asset receiving contract, Layer 1 monitoring module, and proof generation module. In the scheme, the chain where the asset is located is called Layer 1, and the asset receiving chain and the cross-chain module are called Layer 2.



### 2.2.1 Module Function Introduction

- **Asset storage contract**

Deploying smart contract on Layer 1 to record assets transferred to Layer 2.

- **Asset receiving contract**

Deploying the smart contract on the Layer 2 blockchain to receive the assets transferred out of Layer 1, as well as operation records, such as asset transfer on Layer 2.

- **Cross-chain module**

(1) Layer 1 monitoring module, monitoring the transaction of Layer 1's assets deposited into the asset storage contract, and initiating the transaction of assets deposited into the asset receiving contract on Layer 2.

(2) Proof generation module, monitoring all operational transactions of assets in the asset receiving contract on Layer 2, using the zero-knowledge proof algorithm to generate compressed proof data, and submitting the compressed proof data to Layer 1's asset storage contract. That is, to synchronize Layer 2's asset status to Layer 1's asset storage contract.

- **Asset Storage Contract**

(1) Contract events, realizing the deposit and transfer of assets.

(2) Contract events, verifying the proof submitted by the cross-chain module, and updating the asset status.

(3) Contract events, realizing the forced retrieval of Layer 2 assets in the event of Layer 2 exceptions.

(4) Contract docking, realizing docking with the asset contracts.

- **Asset Receiving Contract**

(1) Contract events, realizing the deposit of Layer 1 assets.

(2) Contract events, realizing the return of assets to Layer 1.

(3) Contract docking, authorizing business contracts to process assets.

- **Layer 1 Monitoring Module**

(1) Integrating the Layer 1 blockchain and managing the asset storage contracts docked on Layer 1 through configuration.

(2) Event processing, monitoring the assets operation of the Layer 1 asset storage contract, producing the transaction of the Layer 2 asset receiving contract, and monitoring the correct execution of the transaction.

- **Proof Generation Module**

(1) The calculation of the zero-knowledge proof is large, and the proof generation module will be divided into three sub-modules, which include the asset operation extractor, proof generator, and proof sender.

(2) The asset operation extractor extracts asset-related operations from the asset receiving contract extracted on the Layer 2 blockchain, processes them into the data format received by the proof generator, and sends them to the proof generator cluster.

(3) The proof generator is designed as a cluster due to a large amount of calculation. It receives the data sent by the asset operation extractor, performs zero-knowledge proof verification off the chain, and produces proof. It produces data suitable for asset storage contracts to verify on the chain and sends the data to the proof sender.

(4) The proof sender receives the verification data generated by the proof generator cluster and submits the data to Layer 1's asset storage contract in the form of transactions.

## 2.3 Compatible with Ethereum EVM

Chain33 is fully compatible with Ethereum, supports addresses in the format of Ethereum, and uses the signature method of Ethereum. It can be traded with Chain33 nodes through the web3.js libraries. Developers in the Chain33 ecosystem can seamlessly cooperate with those in the Ethereum ecosystem.

The blockchain has opened up a new direction for Web3.0 applications. Web3.0 eliminates the concept of middlemen, does not require a database to store the state of the App, and does not require a centralized network server to store the core back-end logic. The following is a Web3 dApp architecture diagram based on Chain33.

(1) The front-end defines the user interface logic, and the front-end will interact with the application logic defined in the smart contract. The communication between the front-end and the blockchain is via JSON-RPC.

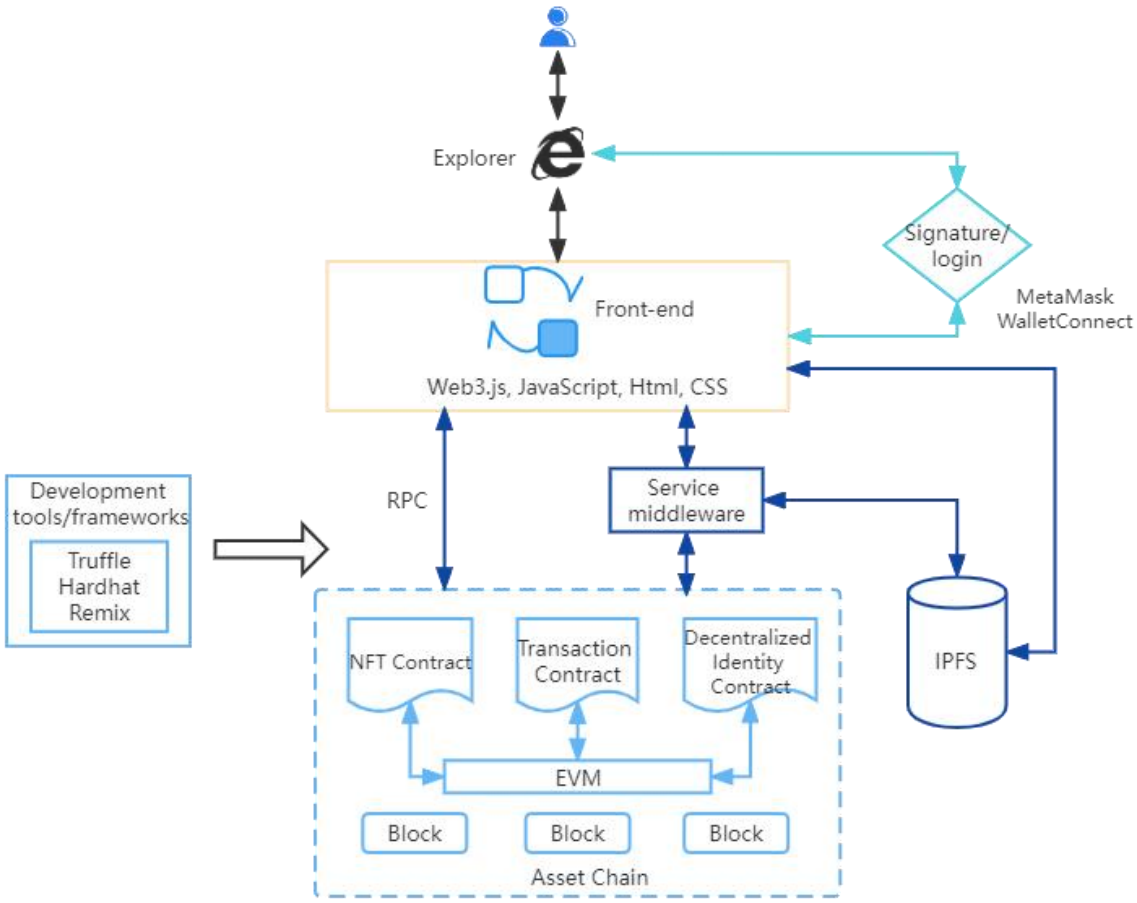
(2) Users on Chain33 can manage their own private key and transaction signatures through MetaMask or WalletConnect. When a user needs to sign a transaction to log in to the platform or send a transaction, MetaMask will be called or WallectConnect will be used to sign.

(3) To save the storage cost of the blockchain, a decentralized off-chain storage solution, such as IPFS, is used for large files (picture, audio, video).

(4) Use the web3.js libraries to query and monitor smart contract events. Specific events can be monitored and a callback will be specified each time the event is triggered to realize the event interaction between the front-end and the blockchain smart contract.

(5) The smart contract is fully compatible with the EVM and supports the seamless migration of contracts on the Ethereum (or Ethereum ecosystem) chain.

(6) Support tools of the Ethereum ecosystem to develop on Chain33. For example, it is easier to build, deploy and test smart contracts through development frameworks or tools, such as HardHat, Truffle, and Remix.



## 2.4 Storage Sharding

### 2.4.1 Technical Background

Currently, every node on most blockchain networks needs to store all the historical data, which can result in significant resource waste when the number of nodes reaches a certain scale. In the context of digital transformation, allowing more data to flow by having each node store only a portion of historical data and obtaining the required data through network interaction is ideal. The storage of our blockchain platform is based on the Kad network, and the distributed algorithm ensures that data is relatively evenly distributed among the nodes of the blockchain while automatically adjusting local data distribution to achieve a new balance when new nodes join or old nodes crash, ensuring data security.

### 2.4.2 Technical Advantages

The p2p module of each node in our blockchain platform holds a public-private key pair, which can be imported by users or generated automatically, and the public key generates a node ID as the unique identifier in the network. The latest 10,000 blocks in the network are not stored in a distributed manner, which increases query efficiency and handles block data rollback scenarios. After surpassing 10,000 blocks, every 1,000 blocks are packed and stored on the 100 closest nodes in the network (The parameters above can be set freely).

**Our storage sharding technology has several advantages as follows.**

(1) Each node only needs to store a portion of the data, making it perfect for mass storage scenarios, and dynamic expansion can be achieved by adding machines at any time.

(2) Packaging data before distributed storage reduces the amount of data and avoids an increase in network load due to excessive data fragmentation.

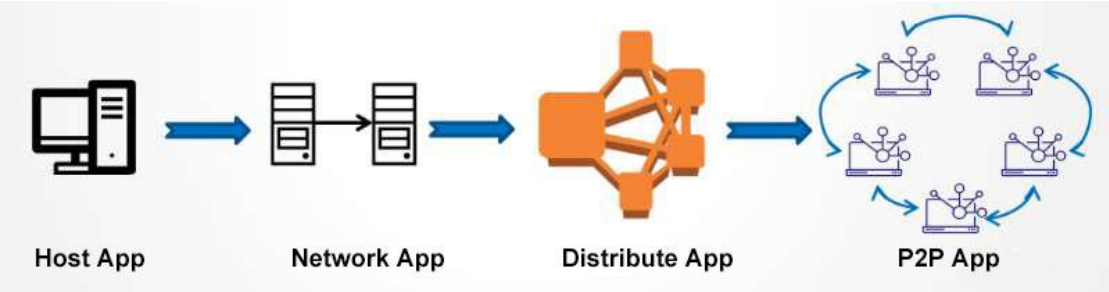
(3) With data stored on 100 nodes, in the extreme case where half of them crash, the probability of data loss is only 1/2^100 ≈ 10^-30, ensuring safe storage of data.

## 2.5 Decentralized Application

DAPP is an application built on a decentralized network, relatived to a traditional centralized application. Traditional centralized applications are deployed on centralized servers and the data is owned by one company or organization; decentralized applications run on decentralized blockchain networks and the data is not controlled by anyone and cannot be deleted or tampered with.

### 2.5.1 Generation of DAPP

Application, in the limited decades since the birth of computers, has also undergone several generations of evolution, which can be briefly summarized as follows.




- **Stand-alone APP**

All the elements that make up an App are stored in a separate node and can be used without a network, which is the earliest form of App existence, such as DOS systems, single-player mini-games, etc.

- **Network APP**

Can be simply understood as the classic Client/Server model, APP consists of these two parts, can exist on top of more than 1 different nodes, the absence of one; such as CS, Red Alert these traditional online games.

- **Distributed APP**

The constituent elements of the App are scattered to N nodes, and the normal use of the App is not affected even if a few nodes hang; such as WeChat, Taobao, etc.

- **P2P APP**

P2P network-based applications, the composition or content of the APP is dispersed to N nodes, and the nodes communicate directly with each other; currently, there are common models such as BitYuan downloads and blockchain applications.

Comparative analysis of the background of each stage of APP generation and its advantages and disadvantages.

| | Advantage | Disadvantage | Background |
|---|---|---|---|
| Stand-alone APP | Easy to use<br>No networking required | Restricted use and limited media distribution<br>Information isolation and inability to share | No Internet<br>Limited network<br>Simple logic |
| Network APP | Low threshold, easy to get started<br>Easy to spread with the help of network<br>Information sharing, attract users | Single network with low reliability<br>A single point of bottleneck<br>High expansion cost | Low-speed network<br>High data volume<br>Low reliability |
| Distributed APP | Fast dissemination<br>Full connectivity of data<br>Low expansion costs<br>High reliability | Data monopoly<br>Mechanism is not transparent<br>Credibility relies heavily on third parties | High-speed network<br>Massive data<br>High reliability<br>Corporate or industry monopoly |
| P2P APP | Network-wide connectivity<br>Transparent processing mechanisms<br>Open and trustworthy data<br>Self-governing, not dependent on third parties | Early development, immature<br>Low processing performance<br>Small amount of processed data | High-speed network<br>Massive data<br>High reliability<br>Open and trustworthy data |

**Definition of DAPP**

DAPP（Decentralized Application），in essence, is a "smart contract", but with a layer of interface wrapped around it to make it easy for end users to use.

Traditional APP composition can be simply understood as: APP = Frontend + Server

Then the DAPP can be simply understood as: DAPP = Frontend + Contracts

The Server of traditional APP can be a single node/distributed/service node in the cloud, which is characterized by centralized control, owned by one or more companies/organizations/individuals, and belongs to centralized mode, and its owner can control the whole logic of the application, so it is called "centralized application".

Contracts (smart contracts) in DAPP are program logic deployed in the blockchain network, which runs in all nodes in the blockchain network, the logic of each node is identical, and the node owner can be anyone, once the contract is deployed, its behavior is not controlled by any particular person, so it is called "decentralized application".

### 2.5.2 Features of DAPP

Compared with traditional centralized APP, DAPP have the following characteristics.

- No registration required: Decentralized applications do not require user registration and can directly access all DAPPs in all public chain networks using the wallet address (which can also be understood as the account address).
- Operation requires authorization or signature: Each step in a decentralized application requires a signature from the user's address and requires a certain on -chain transaction fee, and if the fee is insufficient the transaction will fail.
- Smart Contract Technology: The DAPP may contain one or more smart contracts, which will be executed automatically when the conditions are met.
- Zero falsification of data disclosure: Every transaction on the chain is recorded and transparent, cannot be deleted or tampered with, and any user can access the transaction records without any restrictions.

## 2.6 Smart Contract

Smart Contract, a program deployed on the blockchain network, automates the processing of contracts in the form of computer instructions. Smart contract developers can specify the contract content in advance, and when the contract trigger conditions are met, the program will automatically execute the contract content.

### 2.6.1 Smart Contract Language

Smart contracts actually have their own ecology, and most of them are currently written in solidity with Etherpad. If a new blockchain system is not compatible with this ecology, then it may encounter resistance when promoting its application.

BitYuan's smart contract module is pluggable and also supports a diverse range of smart contract languages, including established development languages such as Solidity, Java, C++ and Golang.

### 2.6.2 Smart Contract Library

Customers do not need to develop every smart contract in their project from scratch, BitYuan already provides many smart contracts for users to use directly.

- Cross-Chain Bridge
- Lending Contracts
- Summary Contracts

- C2C de-neutralized transaction platform contracts
- Privacy Transaction
- Smart Lock Contracts
- Digital Guessing Game Contracts
- Wallet Retrieval Contracts
- Multi-signature Contracts

## 2.7 Account Model

BitYuan R&D is based on the underlying Chain33 blockchain architecture, and internal accounts are often called Chain33 accounts.

This article introduces some basics of Chanin33 accounts and contract accounts to help developers get started more easily subsequently.

### 2.7.1 Chain33 Account Basic Model

There are many addresses stored in the wallet, and these addresses we can understand as accounts. Each account stores many assets: it can be bityuan; it can be some token; it can also be parallel chain or cross-chain assets.

Assets can be moved and used between contracts, and if there is one element that describes the contract in which an asset is located, that element is "the contract in which the asset is currently located".

Assets have a clear attribution, and the address of this attribution is the address (account) to which the asset belongs.

Assets have a number and a nature. For example, if there are 100 active bityuan, "100" denotes the number of such bityuan assets and "active" denotes the nature of such bityuan (and the nature of "frozen " nature).

**View wallet details**

1. View the status of all accounts for all wallets under account.

./chain33-cli account list

In All account information, view the account status of minerAddr.

```
//bityuan
{
    "wallets": [
        {
            "acc": {
                "balance": "9977.6850",
                "frozen": "0.0000",
                "addr": "12qyocayNF7Lv6C9qW4avxs2E7U41fKSfv"
            },
            "label": "minerAddr"
        }
    ]
}
```

2. View the assets of the minerAddr account.

```
./chain33-cli account balance -a 12cjnN5D4DPdBQSwh6vjwJbtsW4EJALTMv
```

Example values are returned as shown below.

```
//token assets
{
    "token": "TC",
    "balance": "50.0000",
    "frozen": "0.0000",
    "addr": "12cjnN5D4DPdBQSwh6vjwJbtsW4EJALTMv"
}
//Parallel Chain Assets
{
    "exec": "user.p.para.token",
    "symbol": "GD",
    "amount": 1100000000,
    "addr": "12cjnN5D4DPdBQSwh6vjwJbtsW4EJALTMv"
}
//Cross-chain assets
{
"exec": "paracross",
    "symbol": "token.TC",
    "balance": "50.0000",
    "frozen": "0.0000",
```

```
    "addr": "12cjnN5D4DPdBQSwh6vjwJbtsW4EJALTMv"
}
```

### 2.7.2 Chain33 Account Introduction

From the basic Chain33 account model, we can find that each account corresponds to an address, and each address is held by the corresponding private key. So how is the existence of multiple assets at one address represented within Chain33?

First of all, from the storage point of view, an asset corresponds to a state, which is stored in the state tree, and the KEY of the node contains the address and asset information, which is expressed as follows.
mavl-{exec}-{symbol}-{addr}

Create an address for a general account, using bityuan as an example, whose status tree KEY is represented as.
exec=coins
symbol=bityuan
addr=12cjnN5D4DPdBQSwh6vjwJbtsW4EJALTMv
key=mavl-coins-bty-12cjnN5D4DPdBQSwh6vjwJbtsW4EJALTMv

The exec and symbol status will be different for different assets at the same address. For example, in the chain33 source code, the NewCoinsAccount function in acccount is to facilitate access to this state. The generic function for accessing assets is NewAccountDB, with the following code.

```
func NewCoinsAccount() *DB {
    prefix := "mavl-coins-bty-"
    return newAccountDB(prefix)
}
func NewAccountDB(execer string, symbol string, db dbm.KV) (*DB, error) {
    ...
    accDB := newAccountDB(symbolPrefix(execer, symbol))
    ...
    return accDB, nil
}
```

Using NewAccountDB we can easily export the asset information we want, for example if we want to export token assets we can execute the following command (the address of token in AccountDB is key=mavl-token-TC-

12cjnN5D4DPdBQSwh6vjwJbtsW4EJALTMv).
NewAccountDB("token", "TC", db)

Now there is linj's account address 12qyocayNF7Lv6C9qW4avxs2E7U41fKSfv, and under his address there are two assets bityuan and TC, and we can see the model under chain33 as follows.

```
└── label:linj-addr:12qyocayNF7Lv6C9qW4avxs2E7U41fKSfv
    ├── mavl-coins-bty-12qyocayNF7Lv6C9qW4avxs2E7U41fKSfv
    └── mavl-token-TC-12qyocayNF7Lv6C9qW4avxs2E7U41fKSfv
```

### 2.7.3 Contract Accounts / Contract Sub-Accounts

A contract account corresponds to the address of a contract, but no one holds its private key. The main function of the contract account is to help us complete the business specified in the contract.

Here we give an example of a transaction contract TRADE, the main function of this contract is to help us complete the exchange of assets.

- **Address Information**

Let's say user linj wants to trade 2 bityuan for 10 TC tokens, and it just so happens that user alice is willing to make this trade. If we want to make this trade, we first need to know their addresses.

The address is as follows.

trade contract address：1BXvgjmBw1aBgmGn1hjfGyRkmN3krWpFP4

linj address： 12qyocayNF7Lv6C9qW4avxs2E7U41fKSfv

alice address： 1GaMeW34xAMo27iPEkVLeZenu1MfLAVn8d

Then check their KEY stored in the database, you can find that their addresses correspond to the same naming rules of KEY and ordinary addresses.

```
//trade bty
mavl-coins-bty-1BXvgjmBw1aBgmGn1hjfGyRkmN3krWpFP4
//trade TC token
mavl-token-TC-1BXvgjmBw1aBgmGn1hjfGyRkmN3krWpFP4
```

For security reasons, chain33 restricts the handling of assets, so the contract needs to handle the assets in its name by itself. So user linj and user alice need to transfer assets to the TRADE contract first if they want to make a transaction.

The contract sub-account format:
mavl-exec-symbol-exec_addr:user_addr

The following are the changes to the contract sub-accounts.
The sub-accounts of alice in the TRADE contract for bityuan and TC are as follows.
label:alice-addr:1GaMeW34xAMo27iPEkVLeZenu1MfLAVn8d
mavl-coins-bty1BXvgjmBw1aBgmGn1hjfGyRkmN3krWpFP4:1GaMeW34xAMo27iPEkVLeZenu1MfLAVn8d
mavl-token-TC-1BXvgjmBw1aBgmGn1hjfGyRkmN3krWpFP4:1GaMeW34xAMo27iPEkVLeZenu1MfLAVn8d

- **Restricted asset disposal rules**

Three basic rules for asset disposal.
(1) Contracts can handle their own accounts.
(2) Contracts can handle sub-accounts in their own account.
(3) Contracts can own the records of the contract (including asset accounts, other business-related records).

Specific extensions to the above three rules (using trade as an example).
(1) The coins contract can handle the state of mavl-coins-start.
(2) The token contract can handle the state of mavl-token-start.
(3) The trade contract can handle the state of mavl-trade-start, which is the order record associated with the transaction.
(4) trade contract can handle its own accounts and sub-accounts in other contracts, satisfying the format mavl-exec-symbol-trade_addr/mavl-exec-symbol-trade_addr: user_addr.

## 2.8 Network Type

BitYuan network is a peer-to-peer, decentralized network called P2P networks. P2P network is a decentralized network where there is no central node for the entire network and all nodes in a traditional central network depend on some central node.

Each node in a P2P network participates and shares the storage capacity, broadcast capacity, and network connectivity that it possesses. Nodes in the network can be accessed directly by peer nodes, and each node is both a consumer and a provider of resources and services.

For customers or developers, the types of bityuan network is roughly divided into main, test and private networks.

- **Main Network**

The main network refers to BitYuan's ecological blockchain network, and all transactions are recorded in the chain's distributed ledger.

The main network is at the same time a public network that anyone can access through the Internet network, supporting anyone to query, generate, and verify transactions on that network.

- **Test Network**

BitYuan offers a public test network.

The test network is a network that simulates the BitYuan mainnet ecosystem, supporting developers to test protocols or smart contracts on the test network that have not yet been deployed to the mainnet environment.

- **Private Network**

Users and developers can build their own bityuan nodes, but not connect to the bityuan network (main and test network), which can be called a private network or can become a development network. It is mainly used to develop bityuan applications on this network and to deploy and run them on this network.

# Chapter Ⅲ Case Introduction

## 3.1 Case Food Traceability

We will explain how to design, implement, deploy, and invoke smart contracts on the Bityuan parachain with a food traceability smart contract based on Solidity language.

### 3.1.1 Parallel Chain Environment Deployment

- **Create wallet**

You don't need to do this step if you have already created a wallet.

*//Generate random number seed. We suggest that users should generate the seed in person and keep it manually since it can be used to recover wallet.*

*./chain33-cli --rpc_laddr="http://localhost:8901"  seed generate -l 0*

*//Save seeds, and create password for wallet.*

*//The password can be customerized. (More than 8 digits, combination of letters and numbers, such as fzm12345) Remember your password, which is used to unlock your wallet.*

*./chain33-cli --rpc_laddr="http://localhost:8901"  seed save -p fzm12345 -s "The seed generated in the previous step"*

*//Unlock wallet. -p is password for wallet, -t is time for automatic wallet lock, -o means never lock unless restart.*

*./chain33-cli --rpc_laddr="http://localhost:8901"  wallet unlock -p fzm12345 -t 0*

*# (Optional) Check the wallet status.*

*./chain33-cli --rpc_laddr="http://localhost:8901"  wallet status*

*//Create an account address and private key*

*./chain33-cli --rpc_laddr="http://localhost:8901"  account create -l testEvm*

*//Query the address and private key of the above created account.*

*./chain33-cli --rpc_laddr="http://localhost:8901"  account dump_key -a "The address generated in the previous step"*

*//View account*

*./chain33-cli --rpc_laddr="http://localhost:8901" account list*

caution: --parameter[rpc_laddr] means the rpc address of the chain that users remote call. (it can be modified based on the real monitoring jrpc address of users, marked as "jrpcBindAddr" in configuration file.)

- **View the status of block synchronization on parachain**

*./chain33-cli --rpc_laddr="http://localhost:8901"  block last_header*


*{*

*"version": 0,*

*"parentHash":*

*"0x905d3c3ab62718381436720382e436a52976b6798896c77c97cb4e751e3a67c9",*

*"txHash":*

*"0x765a8babc9b63f7a5c608afb0943001741f3591f676026cd67dd99f6b3ad5122",*

*"stateHash":*

*"0xeb240fe1248028e9c7271ae2838ea3970bb880031764c8154c8bce2d16262cb7",*

*"height": 57,*

*"blockTime": 1546501778,*

*"txCount": 1,*

*"hash":*

*"0xac2be112305b231b9851a34f6db7bde1745a2b7c9c3a684c736dc59baf3e6e51",*

*"difficulty": 0*

*}*


## 3.1.2 Contract Design

To realize a food traceability process is to achieve the origin, production, circulation and other links of traceability.

The actual traceability is a very complicated process, which is reduced to the following steps: farm, food factory, supermarket, food quality inspection department, and user.

Properties of the product, take live pigs as an example:

(1) Add Pork Information

(2) Add Quality Control Information

(3) Add Supermarket Shelf Information

(4) User Rating Update

### 3.1.3 Contract deployment and invocation

- **Contract Deployment**

*//Constructing contracts to deploy transactions. --user.p.mbaas. is the title of the parachain, this parameter must be filled in correctly, otherwise the transaction cannot be received.*

*./chain33-cli --rpc_laddr="http://localhost:8901" --paraName="user.p.mbaas." evm create -s foodTest -f 1000000 -c "btyecode value"*

*//Signature Transactions -k：Contract deployer's private key.*

*./chain33-cli --rpc_laddr="http://localhost:8901" wallet sign -k 0xdb9415dfbd54ed84dccde80a0f9f2497c7b967116da92d8682900b324ea33d68 -d "data from last step"*

*//Transaction sending*

*./chain33-cli --rpc_laddr="http://localhost:8901" wallet send -d "data after signature from last step"*

*./chain33-cli --rpc_laddr="http://localhost:8901" tx query_hash -s*

*./chain33-cli --rpc_laddr="http://localhost:8901" --paraName="user.p.mbaas." evm calc -a 1LQJbjsNxh6ve6RCXBuDnUuMQsqXsc4cWK -s*

- **Information Input**

(1) Add Pork Information

Cattle ranches selling

| Batch Number | Name | Weight | Date | Origin |
|---|---|---|---|---|
| 00001 | pig001 | 500 | 20190221 | NanJing |

(2) Add Food Information

Food factory production (for example, ham)

| Food number(such as QR code) | Name | Weight | Production Date | Packing Date | Expiration Date | Pork Batch |
|---|---|---|---|---|---|---|
| 001 | food001 | 500 | 20190215 | 20190220 | 20210215 | 00001 |

(3) Add Quality Control Information

Food quality inspection department sampling

| Food number(such as QR code) | Test Time | Test Results | Test Instructions |
|---|---|---|---|
| food001 | 20190226 | Qualified | The food is good. |

(4) Add Supermarket Shelf Information

| Food number(such as QR code) | Last Date |
| --- | --- |
| food001 | 20190304 |

(5) User Rating Update

| Food number | User Rating |
| --- | --- |
| food001 | 90 |

## 3.2 Decentralized Domain Name System on BTY Chain

The Decentralized Name System (DNS) is an identity system based on the BTY public chain, with the purpose of decentralized management of user identities, making blockchain identities more secure, reliable, transparent, and easy to use.

Set up different second-level domain names for various scenarios to replace complex address information. Different second-level domain names can be established through the DNS resolution system and corresponding resolution content can be set, through which it is more flexible to meet application needs. (Register a domain name with name and set a second-level domain name: BTY.yourName.yuan to resolve it to your address on the BTY public chain. In other operations such as transferring funds, this domain name can be entered as replacement.)

Provide multiple suffixes for more flexible use. Currently, the. yuan suffix is provided, and more suffixes will be provided in the future to meet different application scenarios.

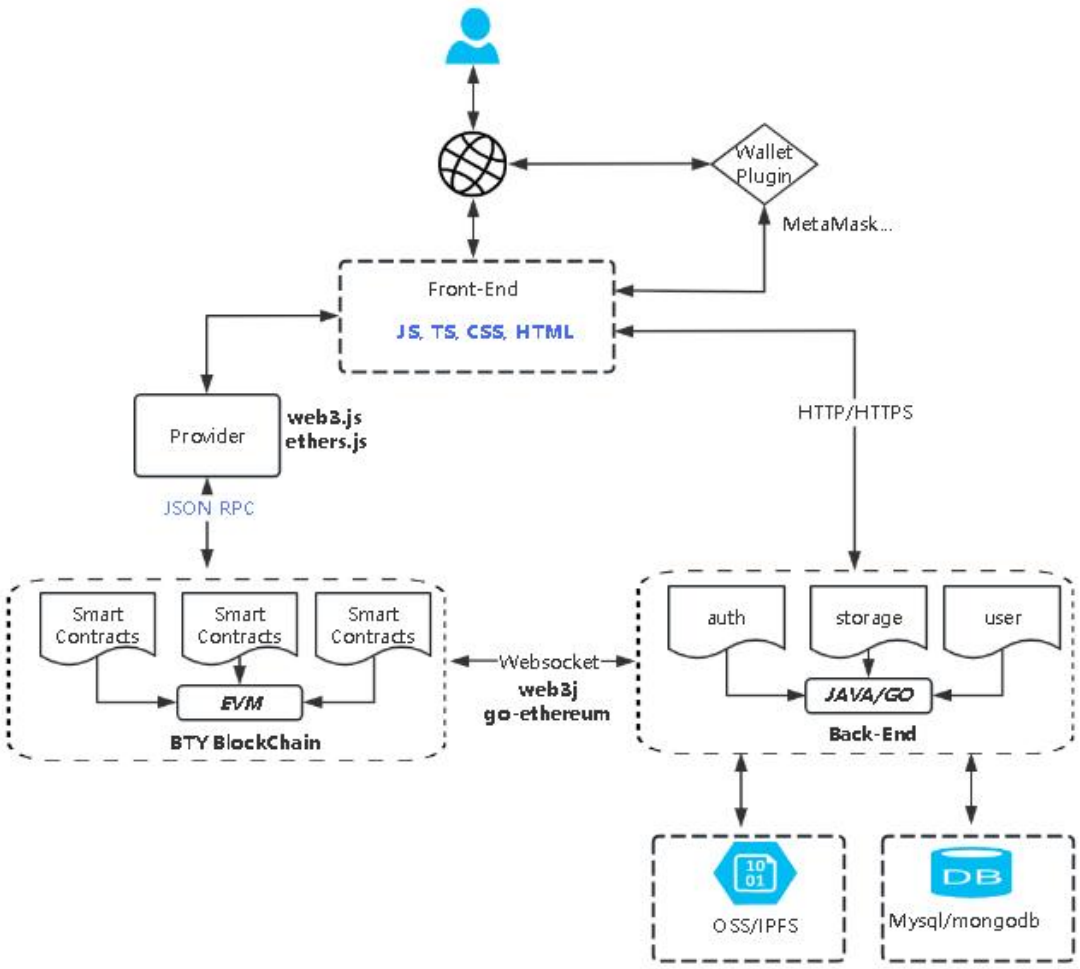
### 3.2.1 Technical Architecture

(1) User login Users log in to DAPP through wallet plugins (MetaMask, etc.), and authenticate their identity through front-end signature and server verification. Save the user's address information in the server database to confirm whether the user is logging in for the first time (equivalent to registering) or has ever logged in.

(2) Contract Interaction Data is signed through the wallet plugin and interacted with the blockchain through JS libraries such as web3. js or ethers. js, calling the contract method to obtain the data.

(3) Data monitoring, order processing, data storage, and other DAPP business logic Servers use SDKs such as web3j or go ethereum to monitor data on the blockchain to update order status, store data, and implement other DAPP business logic. And display the updated data to users through the front-end.

- The following figure shows the general architecture for implementing DAPP on the BTY public chain




## 3.2.2 DNS Contract Introduction

- **DNSRegistry**

Basic Domain Name Registration Management Contract:

(1) Define the event triggering when sub domain name allocation, transfer, parser changes, and domain name TTL survival time changes.

(2) Set domain name records, sub domain name records, owner of the sub domain name, contract address of the domain name parser, and TTL lifetime of the domain name.

(3) Obtain the domain owner, domain manager, domain parser address, whether the domain name exists, and so on

- **Root**

Top level domain name management contract:

(1) Define& lock top-level domain name event.

(2) Assign top-level domain names, view top-level domain's related information, recycle top-level domain names, etc

- **PublicResolver**

Common universal parser contract:

(1) Remove the second-level domain name information, set reverse parsing records, and obtain parsing data based on the second-level domain name node.

(2) Obtain the reverse resolution nodeList and obtain the second-level domain name based on the node.

(3) Obtain a list of all second-level domain names based on the first-level domain name, and query the list in pages

- **ERC721Registrar**

Register domain name contracts with NFT attributes:

(1) Check repeatability and validity of token ID .

(2) Domain name registration, metadata binding, renewal, recycling, etc. for ERC721

(3) Domain name ownership's transfer

## 3.3 Decentralized NFT trading system on the BTY chain

The decentralized trading system is a system that implements NFT asset trading based on the BTY public chain, supporting users to create NFT assets (ERC1155) on the platform and sell them; It also supports importing external NFT assets (ERC1155 and ERC721 (such as domain names on BTY) ) to the platform for sale.

### 3.3.1 Introduction to Trading Contracts

- **Consideration**

Consideration contract, responsible for executing off chain orders on the chain:

(1) Support full transaction and partial transaction functions for orders.

(2) Support order verification and cancellation functions.

(3) Support functions such as obtaining order hash, status, and consideration contract information

- **Asset Contract**

Asset contract, achieving a complete set of NFT/FT creating platforms:

Support functions such as asset creating, batch creating, transfer, destruction, metadata setting, query, etc

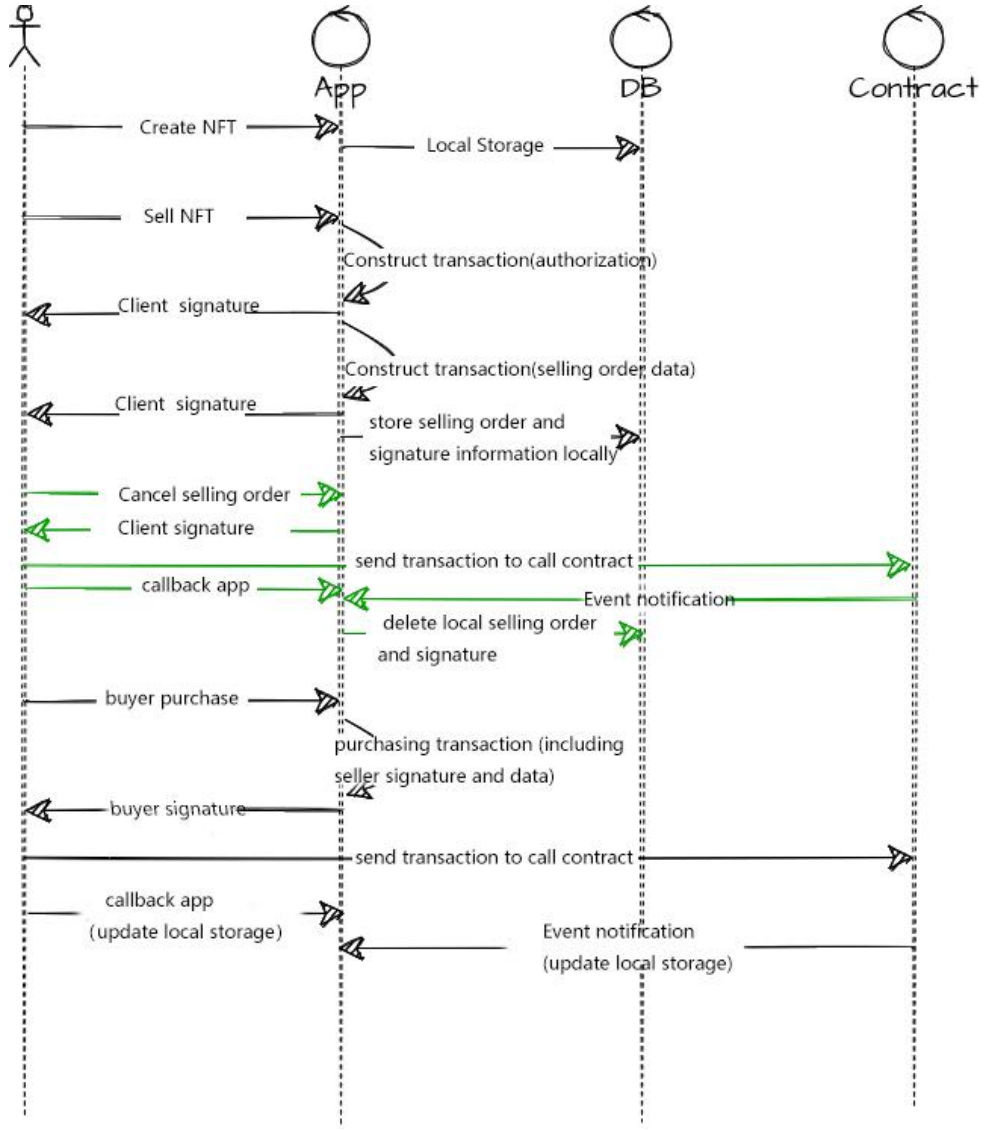
## 3.3.2 Introduction to transaction process

- **Create and sell**

(1) In consideration of saving users' fee, the two steps of creating NFTs and selling them were constructed offline, which are signed by the wallet plugin, and stored in the database (not on the chain).

(2) When a buyer makes a purchase, the purchasing transaction and the previously signed transaction from the seller stored in the database are signed together and then sent on chain.

(3) The server monitors on-chain events and updates the local storage once the transaction is successfully executed.

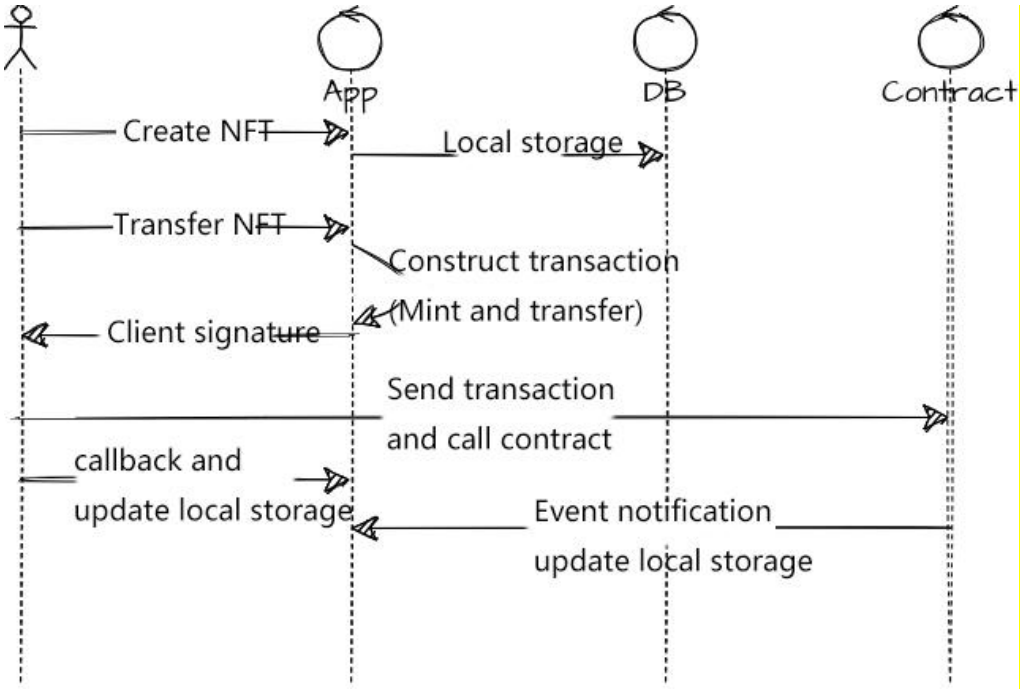


- **Create and transfer**

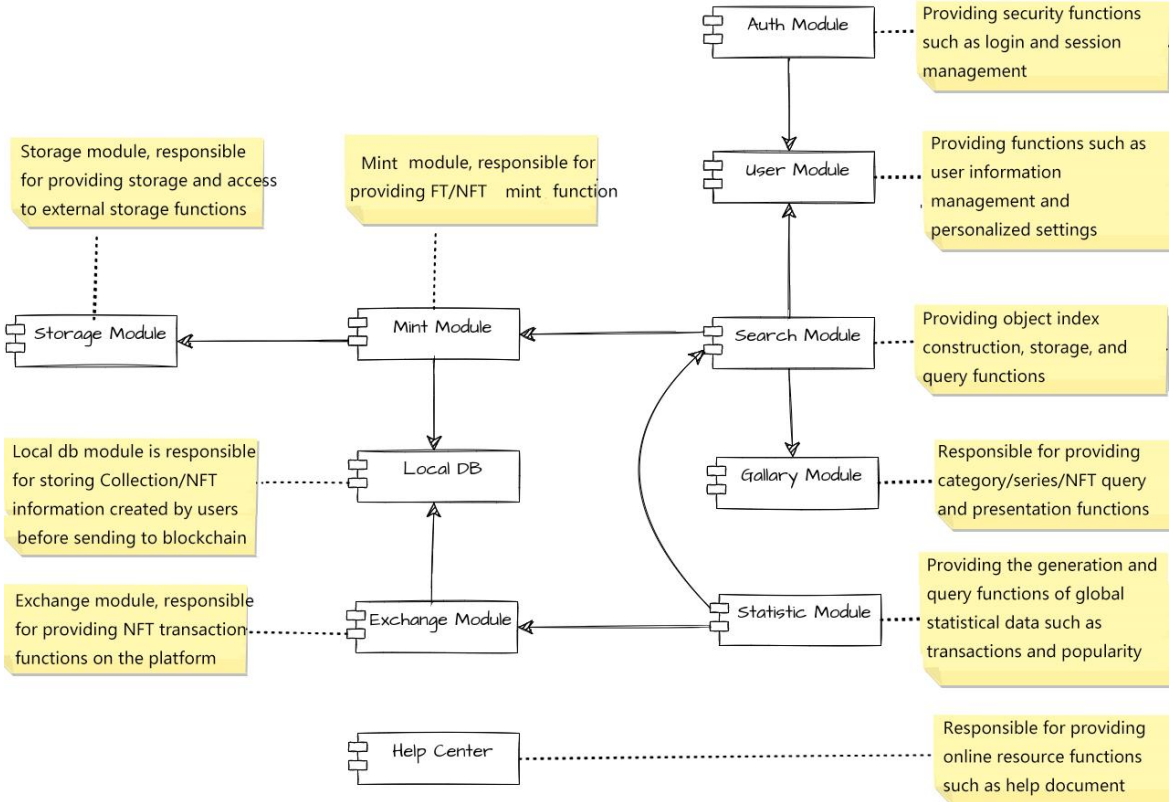
(1) Create NFT that are constructed offline and stored in the database (not on the chain)

(2) Sign the transaction when the transfer occurs, and then upload it to the chain.

(3) The server monitors on-chain events and updates the local storage once the transaction is successfully executed.




## 3.3.3 Platform functional module design



# Chapter Ⅳ BTY Profile

## 4.1 Official Information

Foundation: bityuan foundation limited (established in Singapore)

Address of Development Foundation:

1JmFaA6unrCFYEWPGRi7uuXY1KthTJxJEP

Official Website: www.bityuan.com

Explorer: https://mainnet.bityuan.com/index

Open-source Address: https://github.com/bityuan/bityuan

**INTRODUCTION OF Ishant Singh，CEO OF BITYUAN FOUNDATION**

As an AI expert, Ishant has ever held positions at Amazon and OpenAI, and his achievements are successively recognized by the industry. He was ever featured on Business Insider among 18 other ex-OpenAI employees who successfully started their venture. Ishant developed a ChatGPT-like AI assistant platform for enterprises, capable of integrating with all common business apps like Slack, Google Drive, GitHub, JIRA, Notion, etc. providing an AI infused with contextual business understanding.

As the CEO of BITYUAN FOUNDATION, Ishant is committed to integrating artificial intelligence with blockchain technology to develop more innovative applications and build a smarter, more efficient, and safer technology and application ecosystem.

During working in OpenAI in 2021-2013, Ishant assisted in developing & implementing automated safety solutions for OpenAI's GPT-3, ChatGPT & GPT-4 language models, incorporating reputation management systems, content classifiers, and enforcement mechanisms to ensure secure and responsible usage as AI Safety Specialist – Trust & Safety.

During working in Amazon in 2019, Ishant recommended a new machine learning (ML) model building process to personalize third-party Alexa skills to allow third-party Alexa developers to provide personalized responses to their users, based on user's past usage and preferences as product manager .

## 4.2 Mechanism Issuance

In December 2013, the Bityuan blockchain was launched, and version 1.0 adopted the POW mining mechanism. In May 2014, Bityuan was upgraded to version 2.0 and transformed to adopt the POS consensus mechanism. In order to adapt to future blockchain applications and ecosystem expansion, after four years of development and testing, on May 17, 2018, the blockchain version 3.0 mainnet was launched, and the token migration of BTY was completed. The initial token in circulation is 300 million, and the smallest unit is 10-8.

Before June 20, 2019, the Bityuan mainnet generated a block every 15 seconds, and each block produced 30 BTY, of which 18 BTY were obtained by miners, and the other 12 BTY entered the development fund.

After June 20, 2019, the Bityuan mainnet was upgraded, generating a block every 5 seconds, and each block produced 8 BTY, of which 5 BTY were obtained by miners, and the other 3 BTY entered the development fund.

## 4.3 Development Foundation

Bityuan has always been committed to solving blockchain governance issues in an autonomous manner. Community participants jointly formulate the community operating rules, create the Bityuan ecosystem, and build Bityuan into an autonomous and decentralized digital currency. All participants include but are not limited to managers, technicians, promotion personnel, etc., all may receive corresponding BTY rewards for their efforts.

Currently, the basic allocation ratio of BTY Development Fund is as follows.

- Fund operation and management expenses: 10%
- Ecosystem technology development funds: 30%
- Community promotion incentive funds: 50%
- Special public welfare investment funds: 10%

## 4.4 Community Autonomy DAO

Bityuan will create a set of on-chain voting contracts, aiming to provide a fair, just and open community development autonomous system. The main roles involved in voting governance include community decision-making organization and all ticket holders (deployment of wallet mining, 3000BTY= 1 ticket), proposal voting requires transferring a certain amount of BTY to the development fund. The main functions of the on-chain voting system include the election of community decision-making

organization members, the approval of special fund for development fund, and the adjustment of voting system parameters.

All ticket holders are mainly responsible for:

(1) Regularly elect members of the development fund decision-making organization. Every expenditure in the future BTY development fund will first be voted by members of the decision-making organization.
(2) For large expenditure projects that have passed the decision-making organization's votes and enter the public notice period, they have the right to vote against it.
(3) Once the vote to adjust the DAO system parameters is passed, the voting judgment will be based on the new parameters.

The daily expenditures of the development fund are determined by the development fund decision-making organization's votes. For a certain proposal, the organization must adhere to the following rules.

(1) The daily expenditures required by the development fund will first be voted by the members of the development fund decision-making organization. After approval, it will be decided whether a public notice period is required based on the amount of expenditures (system parameters) required for the project.
(2) If it is less than or equal to the system amount, there is no need to enter the public notice period. It can be implemented after the decision-making organization votes and passes, and the development fund automatically allocates BTY to the designated address.
(3) If it is greater than the system amount, after the decision-making organization votes and passes, it will enter the public notice period. During the public notice period, all ticket holders can vote against the project. When the dissenting votes accumulate to the system parameter indicator, the proposal will be rejected, the development fund will not automatically allocate BTY.

## 4.5 Fuel Consumption

The Bityuan blockchain system aims to create a multi-chain ecosystem that meets the connection of data and assets in different application scenarios. The massive amount of data on the chain should be valuable and meaningful, so BTY is required as fuel, which can filter invalid information to a certain extent and provide a fuel consumption mechanism for BTY under the SPOS consensus.

### 4.5.1 Asset issuance/transfer

In various application scenarios such as stable currency, asset digitization, NFT, and other asset issuance on chain that we are familiar with, it is necessary to generate fuel consumption of BTY as the main chain’s token. For example, issuing NFT assets such as ERC721/ERC1155 on the Bityuan chain need consume BTY which costs little. When we need to transfer digital assets through wallet addresses, we also need to consume BTY to complete the transaction. For example, on e-commerce platforms which based on Bityuan blockchain, consumers need to burn BTY not only to complete the purchase of digital goods, but also to complete digital product transfer and pickup operations.

### 4.5.2 Data storage certificate

Application scenarios such as text, images, audio and video data uploading, as well as non instant messaging, can all be attributed to data storage. The process of storage certificate on blockchain can be divided into three steps: generating digital summaries, writing digital summaries into the blockchain, and verifying digital summaries. Data storage on chain is essentially consistent with the transaction of assets on the chain, except that the former's purpose is to retain digital summaries, while the latter's purpose is to achieve asset transfer. Both require the consumption of BTY token to complete. For example, , in order to display user information that cannot be tampered with and traceable, producers can store the product information (raw materials, processes, packaging, logistics, and after-sales) on the chain at each phase , and perform hash connections to form a complete anti-counterfeiting traceability information chain, on an anti-counterfeiting traceability system based on the Bityuan blockchain. By this way producers can enhance consumer recognition and trust, and thereby increase product sales. During this process, producers only need to consume a small amount of BTY to complete this scenario application.

### 4.5.3 Smart contract

With the rapid iteration of blockchain technology, the application of smart contract is becoming increasingly widespread, including Defi, Gamefi, and others that are currently widely used. On Bityuan blockchain, developers can develop and deploy native smart contracts and EVM smart contracts, and users also need to consume BTY to complete contract interactions when using these contracts. At present, Bityuan supports DAPP development in multiple mainstream programming languages such as go, Java, solidity, and JavaScript. It has also completed the integration of Ethereum web3. js and supports seamless migration of EVM smart contracts. In the future, the application ecology of Btiyuan will have rapid development, and the demand for BTY fuel is bound to significantly increase.

### 4.5.4 Parallel chain ecology

Based on the multi-chain architecture of Bityuan chain, more development teams and application scenarios are willing to choose Bityuan parallel public chain to build their own projects, because this is an excellent solution that not only saves construction costs, but also balances security and performance. On these parallel public chains, there are also application requirements such as digital asset issuance/transfer, data storage, and smart contracts. The Bityuan main blockchain provides a public consensus network that provides verification services for operations on all parallel public chains. All parallel public chains obtain consensus verification from the main Bityuan chain through BTY consumption. Of course, considering the operational independence of parallel public chains, Bityuan blockchain provides fuel withholding solutions for parallel public chains. Users on the parallel public chain do not need to hold BTY and can complete online operations such as token transfer and NFT issuance on the parallel public chain. The operator can configure an exclusive BTY fuel withholding address to uniformly pay users for the BTY fuel required for operation, and can also charge users with coins/tokens on the parallel public chain as service fees on the application terminals.

## 4.6 Risk Warning

Dear BTY community members, there are a number of risks while you are holding and using BTY, please be informed and acknowledge.

**(1) Risk of jurisdiction and law enforcement**

In many jurisdictions, the legal policies associated with BTY and other blockchain technology organizations are not clarified or implemented. It is impossible to predict how, when, or whether a regulatory agency will adopt existing or introduce new regulatory policies for technologies such as BTY and its applications. Such regulations may impose negative effects on BTY and/or BTY systems. If a regulatory action or change to law or regulation makes it illegal to operate within such a jurisdiction, or difficult to do business under necessary regulatory permissions, the Foundation (or its affiliates) may cease to operate in the jurisdiction. .

The Foundation is cautious about BTY sales, with reference to consultations and discussions with a large number of professional legal advisors and ongoing analysis over the development of digital currencies and the legal framework. Therefore, for mass sales, the Foundation needs to constantly adjust its sales strategy to avoid legal risks; besides, it has established legal partnership with Tzedek LAW LLC, a leading legal advisory firm in Singapore with strong reputation in the blockchain sector.

**(2) Risk of market competition**

There is a possibility that an alternative network technology emerges which uses codes and protocols identical to or similar with those of BTY and/or BTY systems to build similar facilities. BTY systems may need to compete with such alternative technologies, leading to negative impacts on BTY and/or BTY systems.

**(3) Risk of team member quit**

The development of BTY systems relies on the continued cooperation between existing technical teams and expert consultants, who are knowledgeable and experienced in their respective fields. The quit of any member may affect the platform of BTY systems or its future development.

**(4) Risk of development failure**

Due to a variety of reasons, the development of BTY systems comes with the risk of failing to proceed as planned, including but not limited to the price decline of certain digital assets, virtual currencies or BTY, unforesee technical difficulties, and the shortage of funds required by system development.

**(5) Risk of security**

Hackers or other malicious groups or organizations may attempt to interfere with BTY and/or BTY systems in a variety of ways, including but not limited to malicious attacks, denial-of-service attacks, consensus-based attacks, Sybil attacks, money laundering, and frauds. In addition, there is a risk that a third party, Foundation member or Foundation branch may intentionally or unintentionally introduce a vulnerability that poses threats to the core infrastructure of BTY and/or BTY systems and has negative impacts on BTY and/or BTY systems.

**(6) Other risks**

In addition to the above risks, there are other risks (such as the special compilation of token purchase agreements) related to your purchase, holding and use of BTY, including those that the Foundation cannot predict. Such risks may also evolve into a variety of unforeseen circumstances or a combination of risks.

You should make adequate due diligence on the Foundation and its affiliates, and understand the overall framework and vision of BTY systems before purchasing BTY.