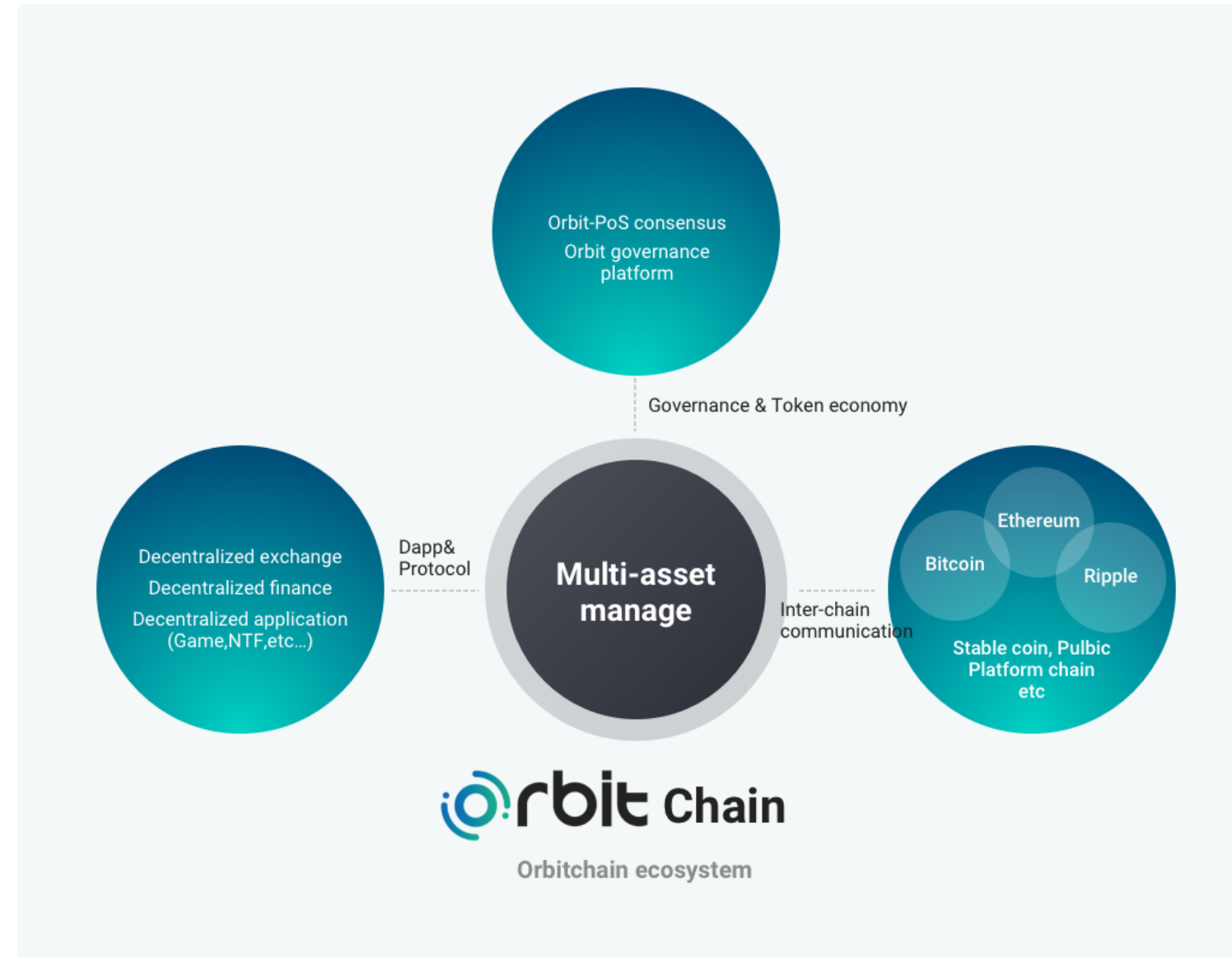


# 0. Overview

⋮



(Fig. 1 Orbit Chain Ecosystem)

Orbit Chain is a multi-asset supporting platform. Through multi-asset support, users can utilize various crypto assets of different blockchains on a singular chain. This allows for easy management and transfer of a user's asset through decentralized communication channels of different chains and Orbit Chain. To DApp and other decentralized protocol developers, Orbit Chain provides a flexible environment to develop on. In addition to providing a high level of liquidity between different assets, by using Orbit Chain, the need to develop for each individual chain is essentially eliminated. The Orbit Chain project infrastructure and road map is available for review in this document. Orbit Chain's support for a token economy and governance as a decentralized platform is also explained. Anyone can create a decentralized service on Orbit Chain with the guides listed below for development, synchronization, and interface.

# 1. Introduction

⋮

## Decentralized Economy and Blockchain

Blockchain was introduced by Nakamoto Satoshi as a decentralized replacement technology for previous, centralized economic systems. Existing central economy systems are largely based on trust of a country or an institution, and are heavily regulated. As they are based on trust, most of the operations of said systems are not transparent, and decisions within such systems are made by central institutions. This process led to questions on whether decisions were made fairly. The evidence of fraudulent transactions in exchanges and insider trading led to distrust on a societal level. To remedy this, Nakamoto Satoshi introduced Bitcoin, a currency system for a trustless, decentralized economy. Bitcoin is a currency that requires no intermediary in possession and trade, and all transactions are recorded and shown transparently. For this, a public ledger system called Blockchain was introduced. Bitcoin caused a large stir within global society, and was followed by other various decentralized systems, spearheaded by Ethereum.

Historically, the current economic system of mankind began as an exchange of services and goods such as animals and plants. Then, as different tribes began to trade, currency was introduced and was used to help with bartering. As countries formed, different currencies were made and used by each country. This led to the system that is being used today. As of now, blockchain is used only between small tribes that use the same language and the same currency - rendering it primitive by today's standards. For the decentralized economy system to become a full-fledged, widespread economic system, a method of transfer between different blockchain systems and centralized systems is a vital necessity -- a technological lingua franca.

At time of writing, the cryptocurrency market cap amounts \$220,615,997,859, with more than 3,000 different coins. Compared to a few years ago, the current amount of money vested in cryptocurrency and the rate of growth is truly astounding. However, considering that Amazon, a single global company, has a market cap of \$884,188,000,000, the cryptocurrency market still only accounts for a small percentage of the overall currency market. Whilst new coins are appearing on a daily basis, the overall market cap of the cryptocurrency market is not increasing at a high rate, and it can be deduced that cryptocurrency overall is in its infancy. All of these blockchain projects have in common the following fact: they are based on trustless block producers that continue the block, recording their transactions on a single chain. Therein lies the first limitation of blockchain, which interferes with the growth of the industry. The foundation of a blockchain system relies on a singular protocol to exchange and prove information within its blockchain. However, that also means it is unable to prove and use information on other chains or offline. This is a huge problem in terms of operation of the economic system.

A word that frequently appears while discussing the economy system and related occurrences is "liquidity". By the definition of the word itself, it is how volatile a subject is. In economic terminology, it describes how easily an asset can be converted into money. Within the current blockchain system, without a middleman, it is impossible to trustlessly trade one blockchain asset to another. The recent development in atomic swap and other IBC protocols have been proposed as a solution to increase liquidity of blockchain assets, but they are not yet implementable for use. We believe that for the cryptocurrency market to grow, the speed and the number of transactions must grow significantly first.

## A Blockchain for Decentralized Protocol / Applications

Orbit Chain is a solution to blockchain's scalability problem. Through Inter-Blockchain protocol, it supports moving decentralized asset across different chains and aims to be specialized as a multi-asset platform.

Bitcoin has a block time of approximately 10 minutes, with 7 TPS, while Ethereum has 15 TPS. Compared with VISA with a 2,000 TPS, these numbers show that they are still far from becoming a viable method of economic transaction. To increase scalability, we developed a chain system that has a higher expandability compared to previous blockchains, based on a PoS consensus algorithm. Through a Staking-based block-producing consensus, Orbit Chain resolves the limitations of previous PoW systems and remains secure against 51% attacks.

Orbit Chain has its own standardized IBC protocol, developed for asset management across different chains. Through this chain, several public blockchain systems with different protocols, such as Bitcoin, Ethereum, Ripple, Klaytn, Terra can communicate with each other. Assets can be moved across this PoS-based decentralized system, and all steps can be verified on-chain, providing a trustless secure protocol.

Orbit Chain aims to become a Hub Blockchain that utilizes multiple assets on its own chain, connecting them all. In doing so, Orbit Chain would be effective in supporting DeFi projects that use multiple types of assets. For many DEX and DeFi financial protocols, liquidity has always been a problem. Orbit Chain solves this problem with its IBC protocol. In addition to providing an environment in which multi-assets can be utilized with ease, Orbit Chain also supplies an SDK for various developers of games and payment systems that want to make the switch over to the blockchain world.

# 1.1 Issues of the Blockchain World ⋮

## **Interoperability**

Data and assets on a blockchain usually are limited to their own chain, resulting in limited access to data and assets on other blockchains. If some blockchains build a centralized approach to handle those data and assets, security and verification issues may arise.

However, Orbit Chain has already built a fully working decentralized IBC protocol which runs with a Staking based Validator governance, which is reliable and safe. All assets and data are verified and stored on the blockchain, and all those processes are transparently open to everyone.

## **Scalability**

Proof of Work (PoW) based Blockchains usually have issues of scalability, speed and expensive fees.

However, Orbit Chain is working with a Staking- based BFT consensus by Validator groups. This will provide much more scalable and reliable IBC. All the information, including assets and data from public chains, is rapidly moved to Orbit Chain. Orbit will support parallel connectivity between Orbit Chain and Public Chains to provide faster transaction speeds.

## **Usability**

Existing blockchains support only one native coin. Additionally, developing and commercializing DApp services on several blockchains are difficult and tedious processes. The developer of the DApp services has to develop separately for each blockchain, which consumes a large number of resources and leads to poor and unreliable management.

Orbit, as a multi-asset blockchain, enables DApp developers to connect with public chains through the Orbit SDK and develop using existing and familiar REST, and WebSocket system APIs, which reduces unnecessary resource consumption and helps developers focus on developing high quality DApp services.

## 1.2 Multi-Asset Blockchain

:

Token economies of blockchain projects are mostly comprised of Ethereum-based cryptocurrencies. These cryptocurrencies serve as a transaction fee on the chain, and also as a vital part of Governance. Other assets exist on ERC20 chains in different formats, and their usage varies as well. Orbit Chain allows all assets to exist on its chain in the same format, even the ones that exist in different forms outside of Orbit Chain.

Orbit Chain aims to work as a hub for various public blockchains, making possible fluid asset movement and interaction within a single blockchain network. Every blockchain has a different set of constituents and protocols, such as UTXO and Balance-state. Connecting these blockchains together is no easy feat, but through Orbit Chain, all of these chains can be linked and utilized in the same fashion, with the same amount of liquidity. To this purpose, Orbit Chain provides a system that will use a singular method of transaction to build DApps, which will then be able to use various assets in the same manner. In doing so, the previous liquidity problem of traditional blockchain systems is eliminated with the standardization of multi-asset usage.

Orbit will also include, in addition to cryptocurrencies, more traditional forms of assets within Orbit Chain. This connection will be made in a different format than cryptocurrencies, yet will still be connected to all in a similar fashion.

# 1.3 Value Proposition

:

## **Cryptocurrency Personal Users**

Previous users of cryptocurrencies had to use different wallets for every type of cryptocurrency they owned. To trade between different types of cryptocurrencies, they had to be sent to a centralized exchange, and then after trading, had to be sent to another wallet. This is a hassle that is eliminated in Orbit Chain, as only Orbit Chain is necessary to utilize all the cryptocurrencies it supports.

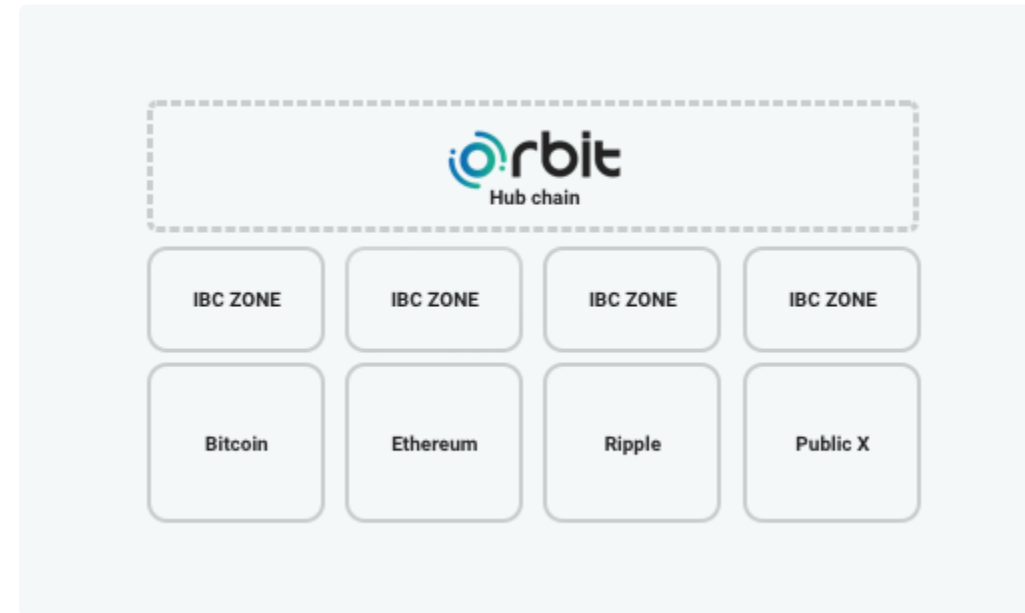
## **Dapp & Protocol Developers**

For many DApp and Protocol developers, choosing a blockchain platform to develop on can be difficult. They foremost want their service to be accessible to many users. In order to achieve that, a blockchain platform where various assets can easily be accessed is ideal. Orbit Chain achieves precisely that, and allows DApp and Protocol developers to utilize Smart Contract with all assets supported by the chain through transactions.

## **Public Blockchain Platform Developers**

Most, if not all blockchain developers want the assets of their ecosystem to be connected and expanded with as many blockchains as possible. However, to achieve this, a large amount of monetary and development resources are required. Also, to maintain this connectivity with the latest versions of Mainnets, continual investment of resources is required. Should these developers connect to Orbit Chain, they can maintain their own Mainnet identity, yet gain access to other blockchains' ecosystems and utilize DEX and DeFi as their side transaction channels.

## 2.1 Overview



(Fig. 2 Orbit Chain Structure)

### Orbit Chain (Hub chain)

- All public blockchains will be connected to Orbit Chain through the IBC zone to share information, data, and asset using a standard format. Orbit Chain will work as the hub chain to gather all the information, data, and assets from all different public blockchains. DApps can easily utilize smart contracts for their utility.
- Orbit Chain will also support simultaneous communication for multiple public blockchains.

### Orbit Inter Blockchain Communication zone (IBC zone)

- Orbit Chain and Public Chain will communicate through a decentralized Inter-Blockchain Communication (IBC) Zone. The IBC Zone will verify information and data from each public blockchain. These verifications will be processed on the blockchain making it safer and more reliable. It will operate with decentralized communication, not requiring a centralized operating and validating process.

## 2.2 Orbit Chain's Design Goal

⋮

**Decentralization:** Anybody can participate in the blockchain with  $O(c)$  amount of resources, including but not limited to verification and sending.

**Resilience:** The chain is to maintain connectivity even if majority of the nodes on the network become disconnected, or go offline.

**Simplicity:** Complexity is lowered, even if it means less efficiency.

**Longevity:** The chain is to be built with modular designs regarding functions to maintain operation continuously even while security/governance flaws are being rectified.

**Scalability:** Transactions and other functionalities are to be processed in a parallel way to increase scalability.

**Connectivity:** The chain is to connect to different systems regardless of their format and design.

**Blockchain Security:** Safeguards and protection are put in place against the risk of a 51% Attack.

**Finality reversion:** Should a certain block A be finalized by a Validator, it must be prevented from being finalized again by another individual.

**Invalid chain finalization:** A validator cannot finalize an invalid block.

**An invalid block must be rejected, following the laws regarding blocks and signature information.**

**Liveness denial:** The chain is to continue functioning in the event that a small number of validators stop producing blocks

**Because Orbit Chain is based on a PoS consensus algorithm, validators that do not validate are pushed out of validator group by penalization through the algorithm.**

**Censorship:** Validator must verify all transactions made on the blockchain fairly.

Should the majority of validators choose to take advantage of their majority voting power, a small number of honest nodes may run the chain independently from the main chain through a soft fork process. In this case, the market is left to choose between the original chain, and the new chain.



## 2.3 Block Consensus

Proof-of-Stake (PoS) is a consensus algorithm that gives voting power equivalent to the amount staked on a Public blockchain. Unlike Proof-of-Work (PoW)-based blockchains that reward participants that decrypt complex cryptography, Proof-of-Stake gives block-producing rights based on the amount of tokens held. PoS-based systems have, as of recent, become a contender for replacing PoW-based systems, as PoS-based blockchains offer high security, high levels of decentralization, and better energy efficiency.

The blockchain, by default, tracks all validators on-chain at all times. By creating a transaction that stakes a certain amount of assets onto the chain, anyone can become a validator. Production of a block is generated through the consensus algorithm, and all validators are eligible to participate in this process.

Previous PoW consensus algorithms used by BTC and Ethereum were regarded as an excessive use of energy to ad infinitum, and this problem is detailed in "EIP1011". Mentioned within this article are the various problems of PoW, such as waste of energy, monopolies introduced by mining hardware, and centralization caused by large mining pools. All of these problems can be, and must be, solved through a switch over to PoS, this article argues. Ethereum is currently being converted into a full-fledged PoS system after a transition stage called Casper FFG (Casper the Friendly Finality Gadget), which is a hybrid system employing both PoS and PoW.

Vitalic explains the philosophy behind the blockchain consensus algorithm in his paper titled "A Proof of Stake Design Philosophy." PoW, he argues, is a method employed by the miners in building the chain through injection of funds, and is representative of a Maximalist philosophy. Vitalic calls the philosophy behind Casper PoS "Cypherpunk" - a philosophy embodying a desire to respect individualism and resist against central systems. This is in line with the message contained within the Genesis Block of Bitcoin, a note left by Satoshi titled "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks." This is the very philosophy behind decentralized public ledgers, and the building block of all blockchain systems.

### PoW (Proof-of-work)

Currently, this blockchain consensus algorithm is the one used by most major (by market cap) blockchains. It is the algorithm suggested by Nakamoto Satoshi, the creator of Bitcoin, and the one detailed in the paper "Bitcoin: A Peer-to-Peer Electronic Cash System." It is used in Bitcoin, Ethereum and other various public blockchains.

The word "Proof-of-Work" signifies the similarity of block producing to actual mining. Allow us to explain PoW through the most commonly known blockchain, Bitcoin. Individual blocks in a blockchain contain a unique hash value called BlockHash. BlockHash is a value generated from version, Merklehash, bits, previousblockhash, time and nonce values. In order to "mine" the block, a miner must find a specific hash value at the current block height, and this process is done by continuously changing the nonce value to find the right hash value. Because there is no direct correlation between the nonce value and the hash value, mining computers enter random values into nonce continuously until the right hash value has been found. In the process of these computations, a large amount of computing power and electricity is consumed.

In case of PoW, the blockchain is protected from attackers by the sheer amount of computing power and electricity required to perform an attack. However, there is still a possibility of an attack succeeding, and when it does, it is not easy to revert. The ASIC chip was developed for the purpose of mining, but the possibility of utilizing a large number of these chips to engineer an attack has been raised as an issue. To combat this problem, a blockchain may change its hash algorithm to protect itself from danger, but this too may change by modifying ASIC mining equipment. A change in blockchain's hash algorithm requires a hard fork of the blockchain, and the cost of this is incurred against the cost of mining countermeasures of ASIC attackers, which does not maintain system integrity through asymmetry. It means there is a need for another consensus algorithm, and the PoS system is receiving much support as an alternative.

### PoS (Proof-of-Stake)

As observed above, PoW is a method that involves producing blocks through using a tremendous amount of computing power, whether from a CPU or GPU. However, with PoS, a method known as "staking" is used to produce blocks. Staking is a method of distributing new blocks based on the amount of monetary units a participant owns on the chain. The PoS block-producing consensus system was introduced to overcome the limitations of previous PoW systems.

As of now, there are quite a few PoS consensus algorithms have been introduced. EOS DPoS, Tezos LPoS, Cosmos Tendermint are but a few of those algorithms. PoS is still in its infancy, and is continually being updated through constructive criticism from the community.

The concept of miners of PoW is replaced with validators in PoS. Two of the most well-known designs of PoS are Chain-based PoS and BFT PoS. For Chain-based PoS, validators are chosen by a periodic random validator selection to choose a block producer. For BFT PoS, block producers are chosen at random. The block that is to be produced is then put through a Finalization process in which the Finality of the block is verified through a voting process by the validator group.

For PoW, if two blocks are to be produced at equal heights to be incorporated into the chain, the chain with higher heights is chosen to be the more legitimate one. For PoS, consensus among the validators is used to determine the legitimate block, and to prevent a party with higher hash power from altering the chain on purpose.

For EOS, a system known as DPoS is employed. It is criticized for its lack of decentralization, as only 21 BPs selected through a voting process are eligible to produce blocks. These 21 BP positions can be claimed by wealth, and they can maintain their power in the same fashion. They can also conspire for their own profit, by producing blocks that are in their favor, and actively declining the ones that go against their will. The fact that the voting against them, and the shift in power and funds occurs through the blockchain is a subject for much debate, as the block will be produced by the already-chosen 21 members. PoS can be designed in a variety of ways, and can be improved through such discussions.

The core safeguard of blockchain decentralization is a design that causes the destruction of the system to cost more than the cost to maintain it. In this regard, PoW is limited in many ways. PoS, however, is maintained by the validators that participate in the system. They become validators through staking, and are monetarily rewarded for growing and maintaining the chain. They are also punished if they were to harm the chain, in monetary terms and other ways. The PoS system is the sum of various functions within it, such as what constitutes efforts to improve the ecosystem, and the compensation for such efforts, and the definition of harming the ecosystem, and the punishment that follows. The blockchain is maintained by designing the security in asymmetrical fashion, as to make the potential punishment for harm far harsher than the potential rewards.

### Byzantine Fault Tolerance

The idea of implementing a Byzantine Fault Tolerance (BFT) consensus came from the challenges faced while building blockchain solutions for banks. Ethereum was chosen as the baseline protocol mostly because of its smart contract capability. However, the built-in consensus, proof of work or Ethash, is not the ideal choice when settlement finality and minimum latency is required.

Orbit's consensus algorithm was designed based on the Practical Byzantine Fault Tolerance (PBFT) algorithm. The logic behind our high performance and instant finality is to support the connection and usability of many assets and DApp services utilizing the Orbit chain. We operate Ethereum-based blockchains that support Ethereum smart contracts. Ethereum has been used as a platform for a long time to build decentralized applications (DApps) with the stability of the Ethereum smart contract. Additionally, Ethereum 2.0 technologies such as Plasma, Casper, and Sharding are constantly being updated to create a fully decentralized and permissionless platform. Orbit Chain is operated as a PoS consensus blockchain based on the Orbit BFT (OBFT) consensus algorithm, which improves upon the existing Ethereum PoW consensus issue.

The Orbit BFT algorithm is a 3-phase consensus algorithm that consists of PRE-PREPARE, PREPARE, and COMMIT. It confirms immediately after block generation by eliminating the issues caused by the generation of multiple blocks at the same height occurring existing in Ethereum, Bitcoin, and EOS. As such, instant finality is expected to be very supportable in a blockchain where many financial transactions occur.

Block generation consensus by a selected few representatives of staking based PoS improves transaction performance, which was limited to 10-30 per second in previous Ethereum, to enable fast block generation and high transaction throughput.

### Block Proposer Selection

For a fair selection of the next block producer, a completely unpredictable random number is used for the process. The random variable is generated through RANDAO and VDF, and each individual block producer is chosen to produce the block, and the validators validate the production.

### RANDAO

RANDAO is performed through the function "commit reveal scheme". Each validators submit a random hash value generated locally on the blockchain. When all the hash values have been collected, the REVEAL function occurs, where all the secret hash values are then calculated through XOR calculations.

For RANDAO, there is a possibility of a Last Revealer Attack. If the last person to submit the hash value is compromised, there is a possibility of engineering the random number, and RANDAO may become a predictable process.

### VDF ( Verifiable Delay Function )

To prevent a Last Revealer Attack from occurring, we have implemented a Verifiable Delay Function, or VDF. VDF is a function that solves a computational problem that takes a very long time to decode, but very short time to verify should the input and the output values exist. Through VDF, we can intentionally design the output function so that the last revealer hash value is one that cannot be predictable.



## 2.3 Orbit Chain IBC

### Basic Design

Lack of communication made it hard to exchange and transfer the value of assets between chains, leaving their economies isolated. It has been a common practice to mint assets such as USD and BTC in a chain through fixed value in a centralized method using the FIAT Gateway method, and utilize them in a chain. This method is based on the trust of the centralized subject. There are questions about uncertainty due to lack of information and the security of the assets. To resolve these issues, many protocols have emerged for communication between unconnected blockchains through decentralization of the communication.

The Orbit IBC protocol enables a decentralized 2-way pegging based on the existing lightning network (micropayment for Bitcoin) and Plasma (scalability solution for Ethereum 2.0) for public chain scalability and communication.

The method presented as Plasma Cash is a single plasma operator method but there are two problems with this approach.

- The Plasma operator can withhold a single block, forcing all users to exit the chain before they can spend their coins
- The Plasma operator can censor and order transactions

Due to these two problems, Plasma operators are in need of higher dependence of trust. This leads to show most of the problems present in a centralized exchange.

The Orbit IBC protocol solves the problems of concentration on one operator by implementing IBC with the PoS-based validation.

### Inter Blockchain Communication Design

STEP 1: Propose Orbit Chain block

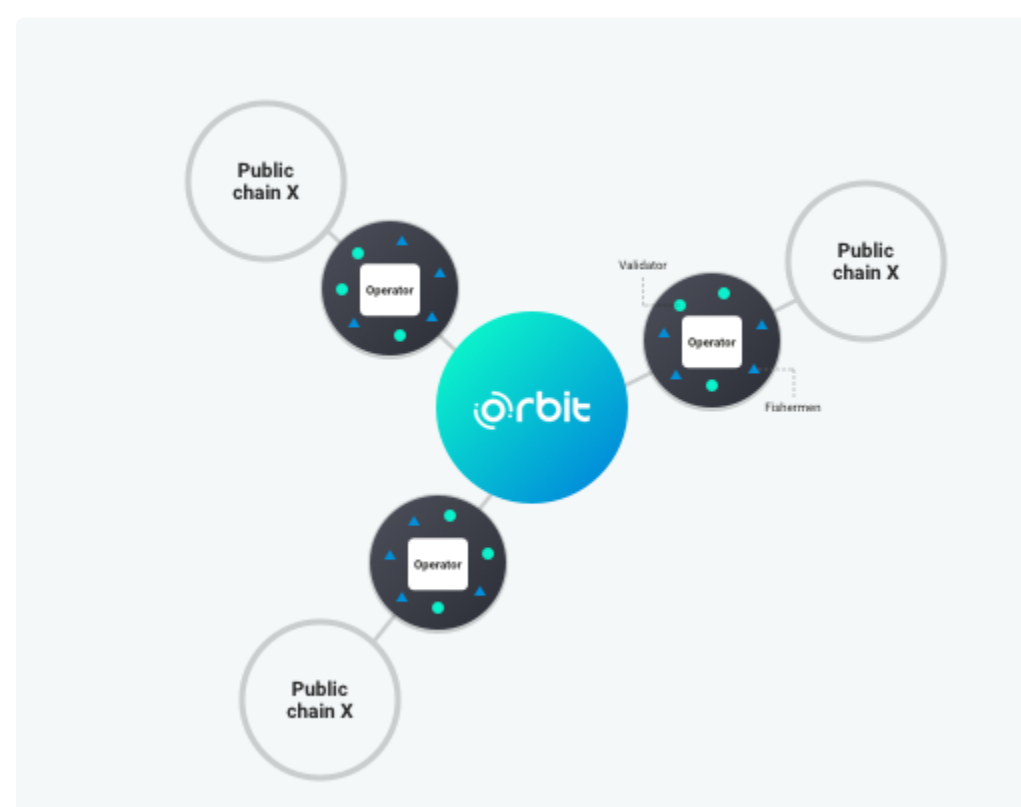
STEP 2: Make merkle object block

STEP 3: Validate object block

STEP 4: Synchronize object block to public chain

### IBC Zone

- **Validator**
  - Verifies and audits both of Orbit Chain and Public Chain. They are also responsible for verification of Inter-block communication process
- **Operator**
  - Operates Inter-block Communication to relay messages between public chains. Everyone can participate as an operator.
- **Fishermen**
  - Monitors the operation of validators and operators through a periodical check or bad signal.



(Fig. 3 Orbit IBC Zone)

### Implementation

Implementation and process of IBC vary by structure of Public blockchain but are commonly designed with the following rules:

- Fixed assets on mainnet cannot be transferred or changed by a single validator.
- Asset deposit will be verified in parallel by validators under consensus of Orbit Chain.
- Asset withdrawal will be managed by smart contracts on Orbit Chain, which will be verified by Validators simultaneously.
- All communication and settlement between Validators will be on smart contract. There is no off-chain communication and consent.

Blockchain system will be classified by following system/structure, and be built with these common conditions/ specifications.

- UTXO-based blockchain (BTC, BCH ...)
  - Implement via multiple-signature wallet
- Balance-based blockchain (ETH, EOS ...)
  - Implement via smart contract
- Ledger-based blockchain (XRP)
  - Implement via multiple-signature wallet

### IBC Zone component

#### Operator

- Listen to orbit chain and public chain
- Manage Relay Deposit/ Relay Withdraw
- Send deposit information to Orbit Chain
- Send withdrawal transaction information to public chain
- Anyone who is connected to network can be operator

#### Validator

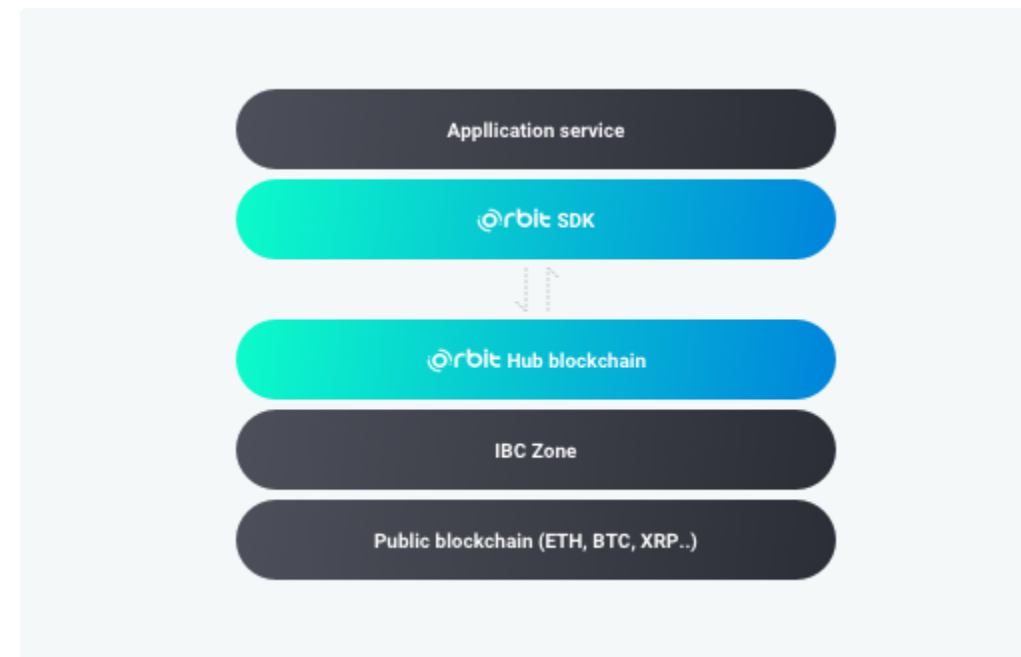
- Listen to Orbit Chain and public chain
- Verify information from public and Orbit Chain
- Signing deposit and withdrawal information
- $N = 3F + 1$  (N: total validator number, F: Bad validator number)
- Constantly increase validators to enhance network security
- Staked asset will determine the scale of verification
- Rewards are given for diligent and successful verification while on duty
- Malicious actions will cause penalty on staked token by governance agreement

#### Fishermen

- Checks and monitors the operation of validators and operators through periodic checks or bad signals.
- Get rewards by disclosing validator's misconduct/ malicious actions.

## 2.4 Orbit Chain SDK

⋮



(Fig. 4 Blockchain based service layer)

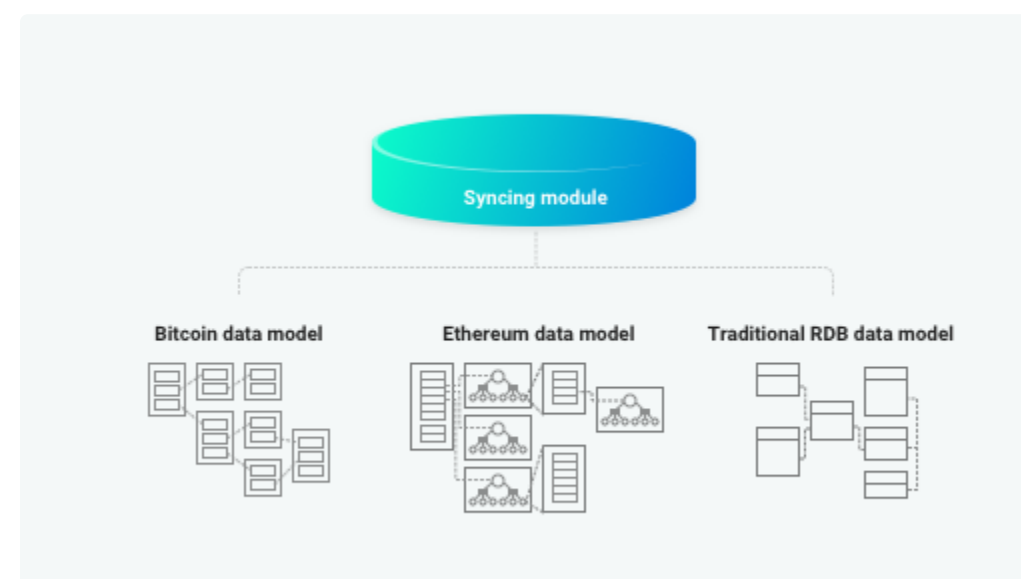
Definition of service system layer for blockchain (Fig.4 Blockchain based service layer) shows that there are some limitations in providing commercialized service on blockchain.

### Limitations of Single Layer Block Chain Services

- Synchronizing data on legacy centralized database with blockchain database
- Slow and limited throughput of blockchain client
- Limited blockchain query
- Limited interoperability between various blockchains and centralized services

Orbit Chain will provide the Orbit SDK to commercialize blockchain based services through the following system layers after many trials and bug fix by ourselves.

- Service layers are considered for direct service and user experience
- Provide REST, SOCKET which will support easy, stable and reliable connectivity between centralized legacy services and decentralized blockchain based services.
- Unified API to connect all blockchains by Orbit SDK
- System that can provide a centralized experience without harm to decentralized governance/system
  - JSON RPC wrapper layer
    - Provide lower level standard communication protocol to communicate with various public chains such as Bitcoin, Ethereum, and other chains and support extension and interoperability.
  - Sync module: distributed ledger to RDB
    - Instance syncing multi-database system



(Fig. 5 Orbit sync module)

# 2.5 Token Economy

:

ORC is used as the basis for all transaction, and as an equity for blockchain consensus. Through ORC, the values contained within Orbit Chain are preserved and operated with.

## ORC Utility

- A cost for resource usage through transactions
- A cost for transfer of funds through transactions
- A method of Staking (Participation of block producing and validation)
- Exercise of voting power within the governance
- A cost for utilizing the IBC

## Transaction Fee

Orbit Chain converts assets from other chains into a standardized token within Orbit Chain. All DApps and transactions on Orbit Chain can utilize transaction services with this standardized token.

**Within Orbit Chain, there are two major types of fees.**

Transaction Fee = Computing Usage Fee + Asset Transfer Fee

1. Computing Usage Fee is a fee incurred from the previous Ethereum chain, for the computing resource that is required for occurring transactions.
2. Asset Transfer Fee is a value that is proportionally calculated to the amount of funds being transferred, that exists for the preservation of the network.

Asset Transfer Fee = MIN (Transfer amount \* Asset transfer fee rate, Asset fee cap)

ex) Transaction transfer amount : BTC = 1 BTC/ ETH = 10ETH, Asset transfer fee rate : 0.0001

Then, Asset Transfer Fee ( BTC ) = 1 \* 0.0001 = 0.0001BTC / ( ETH ) = 10 \* 0.0001 = 0.001 ETH

Transaction fees are the backbones of the Orbit Chain token economy, as they support the chain and provide security to the value imbued to the chain. Orbit Chain, naturally, wishes to raise the value of its chain.

## Validator Economic Incentive

A validator participating in blockchain validation will be rewarded with the following:

Total Incentive to Stake = Staking Rewards + Transaction Fees + IBC Fee - Cost to run a Validator

All validators that stake their assets on the chain are subject to a monetary award defined by the inflation rate. They can also receive the transaction fee from the generated block. In addition to all this, should they be responsible in operating the IBC, they will receive additional rewards as compensation.

## Limit of Block Validator Number

Anyone can become a validator, provided they meet the minimum requirements. However, to become a block-producing validator, there may be difficulties if there are many validators, as too many block-producing validators may throttle the network with an increased number of computations. But increased number of block validators raise the stability of a network, therefore, an upgrade is planned to raise the number of block producers in the future through an upgrade in the network. Initial validators will be 21 people, but this number is designed to be increased through governance consensus in the future.

## Staking Rewards

Validators are rewarded every time a block is generated. This ratio of rewards received is proportional to the amount of funds staked in the entire network. When the staked amount is low, the ratio of reward goes up, whereas if the staked amount is high, the reward is lessened.

## Slashing & Penalty

This is a mechanism encouraging the validators to continually participate in producing blocks, through slashing and penalties. If a validator was to go offline, they would be penalized for it.

If a block is finalized and the node goes offline, the validator would lose x% of the funds staked. The value x is equal to the current interest rate.

For example, if the current yearly interest rate is at 5%, the validator could lose up to 0.0137% of the funds staked. This value is determined on daily basis.

If over 33% of the validators become offline and the block producing is stopped, then the offline validators could lose up to 60% of their funds in an 18-day period. This mechanism is a safeguard to protect the nodes that are online, so that the network can return to its normal state in a short amount of time.

If the amount staked falls below the minimum required stake, the validator would automatically become exempted from the list of validators.

If double signing occurs, where two different blocks of the same height are verified, the participants lose 60% of their staked funds, thereby maintaining the security of the network.

## 2.6 Governance

Orbit Chain uses a set of laws and governance consensus system akin to that of a basic constitution. Bitcoin and Ethereum have had problems with consensus that led to several forks of chain being created because they lack a basic governance system with consensus. As such, we believe having governance is paramount to maintaining a good, sustainable system.

Governance refers to a set of operational policies centered around a decentralized voting system. It allows Orbit ecosystem participants to cast their votes to decide the direction and policies Orbit assumes. However, Orbit governance is not a pure democracy, but a representative democracy. What that means is: regular holders can delegate their ORC to a Candidate and will not be able to directly vote on a subject matter. Instead, Candidates will have voting power vested by holders through ORC, and will exercise their influence on the outcome of a voting process.

### How voting occurs

All resolutions are to be voted in agreement or disagreement only, and candidates can participate and exercise their right by choosing favor, oppose, or abstain.

As such, all resolutions are to be suggested with specific numerical values and avoid vague, abstract forms.

Resolution Example 1) Let's lower the yearly inflation rate from 10% to 8%

Resolution Example 2) Let's use 20,000,000 ORC for execution of IEO

### Members of the Governance

#### (1) Candidate

Top 7 Validators gain additional block-producing rights (# of Validators to increase in future)

The rest of the candidates will be on a waitlist to become block producers

Candidates are required to participate in voting on Governance resolutions

#### (2) Voter

Voters are allowed to delegate their tokens by voting on a Validator. This is their only responsibility and authority.

\* **Exercising Voting Power on the Governance** : Every Candidate has Voting Power proportional to the number of ORC delegated

### Governance Voting Procedure

#### (1) Resolution Initiative

Required fields: Title, Subject Matter, Numerical Specificities and Reasoning

Requirements: Candidate must have over 3 million in Voting Power

Cost: 100,000 ORC (As soon as the resolution is initiated, this amount is locked up, and is excluded from staking)

In the case that the resolution fails, the initiative cost is moved to the reserve, whereas if it is passed, the cost is returned back to the candidate.

#### (2) Voting Start

Time: Voting starts on D+2 00:00 (UTC) from the Resolution Initiative

Once a Resolution is registered, a relevant notice is sent out through the Allbit website and other communities

Rule 1. For every resolution, Voting Power of a Candidate is determined as per the snapshot taken at the start of Voting

Rule 2. During the process of Voting, a change in the number of ORC due to staking/unstaking is not reflected on Voting Power

Reason: Due to the current structure of Orbit Chain, Voting Power of Candidates is subject to fluctuations.

#### (3) Voting Procession

Voting Period: 7 Days (168 Hours)

Voting Options: One in favor, opposition, or abstention

Voting is compulsory, and all candidates are required to participate in the voting process (Responsibility imbued by authority)

Voting may end prematurely should the valid votes in favor or in opposition pass 50% of the total votes, as determined by the snapshot.

#### (4) Grace Period

Voting Grace Period: From the end of voting, until 14 Days (336 Hours) after the start of Voting

Ex) If the voting is completed in 3 days, the grace period may increase to 11 days in total.

If votes are cast during the Grace Period, the votes are not reflected on the voting results.

#### (5) Voting Finalization

Should the quorum not be reached by the end of voting, the resolution is deemed invalid.

Quorum: The quorum is the minimum number of votes required for a resolution to pass. This is based on the number of votes determined by the snapshot at the time of voting start, the total number of votes excluding the abstention is considered valid. Quorum is defined as over 40% of these valid votes.

Only if over half of the valid votes are in favor of the resolution, the resolution is considered to be successful.

Once the result of a resolution is determined, the Grace Period starts for the said resolution.

#### (6) Penalty for failing to participate in voting

Penalty: 1% of Candidate's Voting Power (Slashing)

A candidate who fails to participate in the voting within the Grace Period loses 1% of the allocated Voting Power, and the deducted (slashed) ORC is allocated to the Reserve.

#### (7) Resolution progression

There are two major ways that a Resolution may be carried out.

Immediate Progression: Resolutions that can be carried out immediately within the system, such as inflation or block reward rate.

Subsequential Progression: Some Resolutions cannot be carried out immediately, such as development, concept, and direction. These resolutions are to be reviewed by the Committee, prior to a relevant agenda and roadmap being shared to the community.



# 3. Use Cases

⋮

Orbit Chain is a multi-asset chain based on Inter-Blockchain Protocol. The fact that various assets of different chains can be exchanged and used on a singular chain solves the biggest problem that previous blockchain-based protocols have in common, liquidity. Many DApps and protocols currently face problems overcoming the limitations provided by the platform they are based on, and as a result, have difficulties growing. The expansion of ecosystems are often hindered by the lack of connectivity to other ecosystems. Protocols, DApps, and even newer platform chains struggle not being connected to the larger ecosystems of Bitcoin, Ethereum and Ripple.

## Sidechain for Public Chains

Orbit Chain is a hub chain of every public chain which is not dependent for a specific public chain. This means that with Orbit Chain, public chains can be connected to each other to solve the isolation issue and get flexibility and interoperability. One of the key benefits of using Orbit Chain is that it is able to provide borderless multi-chain asset cross transactions without developing a direct communication channel.

Orbit Chain is already supporting cross transactions between BTC, ETH, XRP and will connect Stable Coin based blockchains soon. This will give DApps more possibilities, utility and even a larger user pool from the various environments.

## Benefits of Interoperability

Most public chains have their own DEX protocol, or run a DEX in which their own token is playing a trading pair role, but they face difficulties with the following:

1. Exchanging DApp tokens with other major coins/tokens other than their native token (DApp Token -> Main Chain Coin -> Centralization Exchange Transfers -> Other Coins/Tokens Exchange),
2. Securing liquidity and users, which is the most important feature of cryptocurrency exchanges, and
3. Developing 2-way pegging system between different chains. There always has been demand for an intercommunication system, but it is difficult to satisfy the demand because commercialized cases are rare and it requires high development resources to build the system.

Those problems are easily, quickly, and efficiently solved by Orbit Chain, which has demonstrated the high-level technology by operating and commercializing a pegging/interworking system for a year.

In the future, Orbit chain will become more diverse by adding more liquidity supply, enabling stable exchange of tokens and allowing cross-flow of cross-chain users through linking and interoperability.

## Use Case 1. Decentralized Exchange

The most basic function that can be performed with currencies, of course, is exchange. Orbit Chain solves the problem that previous chains have had with liquidity and scalability. O-DEX is the decentralized exchange protocol based on Orbit Chain that allows exchange of currencies, on-chain.

O-DEX protocol (allbit.com)

## Use Case 2. Decentralized Finance

As of recent, the need for decentralization of deposit/loan services of blockchain assets have been growing. As is with decentralized trading, liquidity and scalability are both important traits a chain must possess in operating decentralized protocols. Orbit Chain is the chain most optimized for such traits, and provides an excellent environment for DeFi protocols to develop on. Currently, the pool-based DeFi protocol, Divine, operates on Orbit Chain.

Divine protocol (trinito.io)

## Use Case 3. Chain-to-Chain Bridge Gateway

Orbit Chain is a service that appeals not only to DApp and Decentralized Protocol developers, but to projects that want their blockchain to become connected to other chains. Simply by connecting their current chain to Orbit, they can utilize the assets of other chains within their own chain, thus gaining more liquidity for their chain.

## Use Case 4. Multi-Asset Dapp

Many DApps get trapped within the confines of their platform once they start developing. This causes problems with liquidity, as once the development is somewhat complete, their assets may become limited in transferring to and from other chains.



# 4. Roadmap

⋮

## 2Q, 2019

- Official introduction of Orbit Chain
- Introduction of Orbit Chain technology, and its structure
- Introduction of major Validator partners, and staking begins
- XRP Mainnet connected

## 3Q, 2019

- Lightpaper released
- Decentralized Governance tool launched
- Developer-friendly Testnet 'Avocado' launched
- Terra & Klaytn Mainnets connected

## 4Q, 2019

- Whitepaper released
- Decentralized token swap service utilizing IBC launched

## 1Q, 2020

- Orbit Chain Mainnet launched
- Instant Swap-based DEX protocol launched
- Another Mainnet connected to Orbit Chain

## 2Q, 2020

- Additional protocol for DeFi developed and instated
- 8-10 Dapp development projects secured
- Another Mainnet connected to Orbit Chain

**\*Roadmap schedules are subject to change, depending on the development process/company direction.**

# 5.1 Asset-Balance

⋮

## Contract Interface

```
event GenesisBalance(address indexed user, bytes32 indexed tokenId, uint balance);
event BalanceChange(address indexed user, bytes32 indexed tokenId, uint balance);
event Transfer(address indexed fromAddr, address indexed toAddr, bytes32 indexed tokenId, uint amount);
event CancelTransfer(address indexed fromAddr, address indexed toAddr, bytes32 indexed tokenId, uint amount);
event Deposit(address peggingContract, bytes32 tokenId, address toAddr, uint amount);
event Withdraw(address peggingContract, bytes32 tokenId, address user, address extraUser, uint amount);
event GiveBack(address peggingContract, bytes32 tokenId, address user, address extraUser, uint amount);

mapping (bytes32 => address) public peggingContracts;
mapping (bytes32 => bytes32) public tokenIdToSummary;
mapping (bytes32 => uint) public decimals;
mapping (bytes32 => bool) public isValidToken;
mapping (bytes32 => mapping (address => uint)) public balanceOf;
mapping (address => bool) public isValidPeggingContract;
mapping (bytes32 => bool) public isUsedTransfer;
mapping (bytes32 => bool) public isUsedWithdraw;
mapping (address => bool) public userTransferable;
mapping (bytes32 => uint) public withdrawFee;

bool public defaultTransferable;
uint public withdrawCount = 0;
address public feeGovernance;

function deposit(bytes32 tokenId, address toAddr, uint amount) public;
function deposit(bytes32 tokenId, address toAddr, uint amount, address extraToAddr) public;
function giveBack(bytes32 tokenId, address toAddr, uint amount, uint networkFee) public;
function giveBack(bytes32 tokenId, address toAddr, uint amount, uint networkFee, address extraToAddr) public;
function withdraw(bytes32 tokenId, bytes memory destination, uint amount, bytes memory extraData) public;
function withdraw(bytes32 tokenId, address extraUser, bytes memory destination, uint amount) public;
function withdrawBySignature(bytes32[] memory bytes32s, uint[] memory uints, address[] memory addrs) public;
function cancelWithdrawSignature(address fromAddr, bytes32 tokenId, bytes memory destination) public;
function transfer(bytes32 tokenId, address toAddr, uint amount, bool needLog) public;
function transfer(bytes32 tokenId, address toAddr, uint amount, bool needLog, address extraToAddr) public;
function transfer(bytes32 tokenId, address toAddr, uint amount, bool needLog, bytes memory extraData) public;
function transferBySignature(bytes32[] memory bytes32s, address[] memory addrs, uint[] memory uints) public;
function transferBySignatureWithExtra(bytes32[] memory bytes32s, address[] memory addrs, uint[] memory uints, bytes memory extraData) public;
function cancelTransferSignature(bytes32[] memory bytes32s, address[] memory addrs, uint[] memory uints) public;
function cancelTransferSignatureWithExtra(bytes32[] memory bytes32s, address[] memory addrs, uint[] memory uints, bytes memory extraData) public;

function getBalance(bytes32 tokenId, address addr) public view returns(uint);
function logBalance(bytes32 tokenId, address addr) public;
```

## 5.2 IBC-Bitcoin

### Contract Interface

```
struct DepositItem {
    bytes32 txid;
    uint vout;
    uint amount;
    bytes scriptPubKey;
    address toAddr;
    address extraToAddr;
    bool deposited;
    bytes32 depositPubKeyX;
    bytes32 depositPubKeyY;
}

struct WithdrawItem {
    uint wtype; // 0: default, 1: send to cold
    uint withdrawId;
    bytes destination;
    uint amount;
}

struct Utxo {
    bytes32 txid;
    uint vout;
    uint amount;
    bytes scriptPubKey;
    bool used;
}

struct WithdrawTx {
    bytes32 suggestHash;
    bytes32[] keyHashs;
    bytes32[] inputHashs;
    uint outputStart;
    uint outputSize;
    uint fee;
    uint change;
}

struct Destination {
    address toAddr;
    address extraToAddr;
}

mapping(bytes32 => DepositItem) public deposits;
mapping(uint => WithdrawItem) public withdrawals;
mapping(bytes32 => Utxo) public utxos;
mapping(uint => WithdrawTx) public suggestions;
mapping(uint => WithdrawTx) public selections;
mapping(uint => bytes32) public utxoKeys;
mapping(bytes32 => mapping(bytes32 => Destination)) public depositDestinations;
mapping(bytes32 => bool) public isUsedMappingHash;

bool public isActivated = true;
uint public utxoCount = 0;
uint public validUtxoCount = 0;
uint public withdrawalCount = 0;
uint public withdrewCount = 0;
uint public suggestCount = 0;
uint public selectionCount = 0;
uint public btcBalance = 0;
uint public holdingLimit = 0;

address tokenAddr = 0x0000000000000000000000000000000000000000000000000000000000000001;
bytes coldAddr;
bytes public redeemScript;

event DepositDestinationsMapping(bytes32 btcPubKeyX, bytes32 btcPubKeyY, address toAddr, address extraToAddr);
event DepositRelay(bytes32 txid, uint vout, uint amount, bytes scriptPubKey, bytes32 depositPubKeyX, bytes32 depositPubKeyY);
event DepositValidated(bytes32 txid, uint vout, uint amount, bytes scriptPubKey, address toAddr, address extraToAddr);
event WithdrawAdded(address balanceAddr, uint wtype, uint withdrawId, address user, bytes32 tokenSummary, bytes memory scriptPubKey);
event TxSuggestAdded(bytes32 suggestHash, uint suggestIndex, bytes32[] keyHashs, bytes32[] inputHashs);
event TxSuggestRelay(bytes32 suggestHash, uint suggestIndex, bytes32[] keyHashs, bytes32[] inputHashs);
event TxSelected(uint selectionIndex, bytes32[] keyHashs, bytes32[] inputHashs, uint amount);
event TxSelectedRelay(uint selectionIndex, bytes32[] keyHashs, bytes32[] inputHashs, uint amount);
event WithdrawValidated(uint selectionIndex, bytes32[] keyHashs, bytes32[] inputHashs, uint amount);
event UsableUtxoRelay(bytes32 txid, uint vout, uint amount, bytes scriptPubKey);
event UtxoAdded(uint utxoIndex, bytes32 txid, uint vout, uint amount, bytes scriptPubKey);

function getSuggestKeyHashs(uint suggestIndex) public view returns(bytes32[] memory);
function getSuggestInputHashs(uint suggestIndex) public view returns(bytes32[] memory);
function getSelectionKeyHashs(uint selectionIndex) public view returns(bytes32[] memory);
function getSelectionInputHashs(uint selectionIndex) public view returns(bytes32[] memory);
function bytes32ToAddress(bytes32 data) public pure returns (address);

function setDepositDestinations(bytes32[] memory btcPubKeyX, bytes32[] memory btcPubKeyY, address toAddr, address extraToAddr);
function relayDepositUtxo(bytes32 txid, uint vout, uint amount, bytes memory scriptPubKey, bytes32 depositPubKeyX, bytes32 depositPubKeyY);
function relayUsableUtxo(bytes32 txid, uint vout, uint amount, bytes memory scriptPubKey);
function validateDepositUtxo(bytes32 depositKeyHash, address validator, uint8 v, bytes32 txid, uint vout, uint amount, bytes memory scriptPubKey);
function validateUsableUtxo(bytes32 txid, uint vout, uint amount, bytes memory scriptPubKey);
function withdraw(uint withdrawId, address user, bytes32 tokenSummary, bytes memory scriptPubKey);
function withdrawColdScript() public onlyActivated;
function suggestWithdrawTx(bytes32[] memory keyHashs, bytes32[] memory inputHashs, uint amount);
function relayTxSuggest(uint suggestIndex) public;
function validateSuggest(bytes32 suggestHash, uint suggestIndex, address validator);
function relayTxSelected(uint selectionIndex) public;
function validateWithdraw(uint selectionIndex, address validator, uint8[] memory v,
```

## 5.3 IBC-Ethereum

⋮

### Contract Interface

```
event AddToken(address tokenAddr, bytes32 tokenSummary, bytes32 tokenId);
event RemoveToken(address tokenAddr, bytes32 tokenSummary, bytes32 tokenId);

event DepositRelay(address mainAddr, address tokenAddr, address addr, address toAddr);
event DepositValidated(address mainAddr, address tokenAddr, address addr, address toAddr);

event WithdrawRelay(uint withdrawId, address balanceContractAddr, bytes32 tokenSummary);
event WithdrawValidated(uint withdrawId, address balanceContractAddr, bytes32 tokenSummary);

mapping (bytes32=>bool) public deposits;
mapping (bytes32=>address) public tokenAddr;
mapping (address=>bytes32) public tokenSummaries;
mapping (bytes32=>bool) public isListing;

struct Withdrawal{
    address user;
    bytes32 tokenSummary;
    bytes destination;
    uint amount;
    bytes comment;
}

mapping (uint=>Withdrawal) public withdrawals;

function relayDepositToken(address mainAddr, address fromAddr, address toAddr, address tokenAddr, bytes32 tokenSummary) public;
function validateDepositToken(address mainAddr, address fromAddr, address toAddr, address tokenAddr, bytes32 tokenSummary) public;
function checkValidateDepositToken(address mainAddr, address fromAddr, address toAddr, address tokenAddr, bytes32 tokenSummary) public;
function withdraw(uint withdrawId, address user, bytes32 tokenSummary, bytes memory destination) public;
function withdraw(uint withdrawId, address user, address extraUser, bytes32 tokenSummary, bytes memory destination) public;
function relayWithdraw(uint withdrawId) public;
function validateWithdraw(uint withdrawId, address validator, uint8 v, bytes32 r, bytes32 s) public;
function checkValidateWithdraw(uint withdrawId) public;
```

## 5.4 IBC-Ripple

:

### Contract Interface

```
struct Withdrawal {
    uint wtype; // 0: default, 1: send to cold
    address user;
    address extraUser;
    bytes from;
    bytes dest;
    bytes tag;
    uint amount;
    bool validated;
    bool isCanceled;
    uint fee;
    uint seq;
    mapping(address => bytes32) signatureHashs;
}

struct Suggestion {
    address[] validators;
    bytes32[] signatureHashs;
    uint fee;
    uint seq;
}

mapping(bytes32 => bool) public isDeposited;
mapping(uint => Withdrawal) public withdrawals;
mapping(uint => mapping(uint => Suggestion)) public suggestions;

bool public isActivated = true;
uint public withdrawalCount;
uint public suggestCount;
uint public lastWithdraw;
uint public lastSelected;
uint public lastSequence;
uint public xrpBalance;
uint public holdingLimit;
bytes public xrpWallet;

bytes coldAddr;
bytes32 public tSummary = sha256(abi.encodePacked('XRP', address(1)));

event XrpDepositRelay(bytes32 txid, bytes xrpWallet, address toAddr, uint amount, address extraToAddr);
event XrpDepositValidated(bytes32 txid, bytes xrpWallet, address toAddr, uint amount, address extraToAddr);
event XrpDepositColdRelay(bytes32 txid, bytes xrpWallet, uint amount);
event XrpDepositColdValidated(bytes32 txid, bytes xrpWallet, uint amount);
event XrpWithdrawAdded(uint withdrawId, address balanceAddr, uint withdrawIndex, bytes memory balance);
event XrpWithdrawSuggested(uint suggestIndex, uint withdrawIndex, address[] validators, bytes32[] signatureHashs);
event XrpSuggestionValidated(uint suggestIndex, uint withdrawIndex, uint fee, uint seq);
event XrpWithdrawValidated(uint withdrawIndex);
event XrpTransactionFailed(uint withdrawIndex);
event XrpWithdrawCanceled(uint withdrawIndex);

function relayDeposit(bytes32 txid, address toAddr, uint amount, address extraToAddr) public;
function relayDepositCold(bytes32 txid, uint amount) public onlyActivated;
function validateDeposit(bytes32 txid, address toAddr, uint amount, address extraToAddr) public;
function validateDepositCold(bytes32 txid, uint amount, address validator, uint8 vSig) public;
function withdraw(uint withdrawId, address user, bytes32 tokenSummary, bytes memory balance) public;
function withdraw(uint withdrawId, address user, address extraUser, bytes32 tokenSummary, bytes memory balance) public;
function transferToCold() public;
function suggestWithdraw(uint withdrawIndex, address[] memory validators, bytes32[] memory signatureHashs) public;
function relaySuggestion(uint withdrawIndex, uint suggestIndex) public;
function validateSuggestion(uint withdrawIndex, uint suggestIndex, address validator, uint8 vSig) public;
function relayWithdraw(uint suggestIndex, uint withdrawIndex) public;
function validateWithdraw(uint withdrawIndex, address validator, uint8[] memory vSig) public;
function relayTransactionFailed(uint withdrawIndex) public;
function cancelWithdraw(uint withdrawIndex, address validator, uint8 vSig, bytes32 tokenSummary) public;

function withdrawIndexToSuggest() public view returns(uint);
function withdrawIndexToSign() public view returns(uint);
function getSuggestionValidators(uint withdrawIndex, uint suggestIndex) public view returns(address[]);
function getSuggestionSigHashs(uint withdrawIndex, uint suggestIndex) public view returns(bytes32[]);
function getSignatureHash(uint withdrawIndex, address validator) public view returns(bytes32);
```



# 6.1 Bitcoin IBC Protocol

## Deposit

Before you send Bitcoin to our multi-sig wallet, you have to do mapping of your BTC address and Orbit address to the Orbit network

For mapping your address, you have to send a mapping transaction to BtcPeggingContract


```
function setDepositDestinations(bytes32[] memory btcPubKeyX, bytes32[] memory btcPubKeyY,
```

If mapping succeeds,

```
event DepositDestinationsMapping(bytes32 btcPubKeyX, bytes32 btcPubKeyY, address toAddr,
```

DepositDestinationsMapping event occurs in BtcPeggingContract

After deposit destination is mapped, you can send Bitcoin to our multi-sig wallet

 You have to use a p2sh wallet to send Bitcoin to our multi-sig wallet.  
V[in] UTXO in your transaction have to solve by public key

After your transaction is confirmed, Orbit BTC Operator, validator will begin to proceed your deposit.

```
//BtcPeggingContract  
event DepositValidated(bytes32 txid, uint vout, uint amount, bytes scriptPubKey, address  
//BalanceContract  
event BalanceChange(address indexed user, bytes32 indexed tokenId, uint balance);
```

## Withdrawal

Send a withdrawal transaction to Orbit Balancecontract

```
function withdrawBySignature(bytes32[] memory bytes32s, uint[] memory uints, address from  
function withdraw(bytes32 tokenId, bytes memory destination, uint amount, bytes memory c
```

Then, Bitcoin IBC operator and Validator begin to proceed this withdrawal.

## 6.2 Ethereum IBC Protocol

### Deposit

If you want to send ERC20 tokens to the Orbit IBC contract, you must check that the token is registered

```
decimal(address tokenAddr)
```

Send a deposit transaction to the Orbit IBC contract on the Ethereum MAINNET/ROPSTEN

```
// deposit ETHfunction deposit(address toAddr, address extraToAddr) payable public// dep
```

If your transaction succeeds,

```
event Deposit(address tokenAddr, address addr, address toAddr, uint amount, uint deposit
```

Deposit event occurs in your transaction.

Then, Ethereum IBC operator and Validator begin to proceed this deposit.

When deposit is completed in the Orbit chain, DepositValidated and BalanceChange events occur.

```
// EthpeggingContractevent DepositValidated(address mainAddr, address tokenAddr, address
```

### Withdrawal

Send a withdrawal transaction to Orbit Balancecontract.

```
function withdrawBySignature(bytes32[] memory bytes32s, uint[] memory uints, address fro
```

Then, Ethereum IBC operator and Validator begin to proceed this withdrawal

When withdrawal is completed in Ethereum, Withdrawal event occurs.

```
event Withdraw(address tokenAddr, address addr, address toAddr, uint amount, bytes32 wha
```

## 6.3 Ripple IBC Protocol

⋮

### Deposit

Send an XRP to our multi-sig wallet

Send Transaction Requires:

- memo field

```
memos = [{
  data : <to_addr>,
  type : 'toAddr'
},
{
  data : <extra_addr>,
  type : 'extraToAddr'
}]
```

- Transaction Type

Payment

### Withdrawal

Send a withdrawal transaction to Orbit BalanceContract

```
function withdrawBySignature(bytes32[] memory bytes32s, uint[] memory uints, address from) public {
function withdraw(bytes32 tokenId, bytes memory destination, uint amount, bytes memory c)
```

Then, XRP IBC operator and validator begin to proceed this withdrawal.