ONTOLOGY MULTICHAIN DESIGN
DRAFT v0.1

THE ONTOLOGY RESEARCH INSTITUTE
RESEARCH@ONT.IO

Abstract. Due to the diversity of real-world business scenarios, a single blockchain is not able to meet the needs of all business scenarios. Therefore, blockchains with different architectures and features are needed for different scenarios. Under a multichain system, value exchange and interoperability between different blockchains are important and meaningful. Ontology has launched a light-weight, low-coupling, safe and reliable multichain system and cross-chain solution. The solution uses the Ontology blockchain as the main chain, and supports both homogeneous side-chains (side-chains whose architecture is the same as that of the Ontology main chain) and heterogeneous side-chains (side-chains whose architecture is different from that of the Ontology main chain), and allows for interaction between the main chain and side-chains, and also between side-chains. The main chain manages side-chains using a multichain management contract, and the cross-chain interaction between source chain and target chain is realized by a cross-chain management contract. The proof of cross-chain interaction is realized by synchronizing key block headers and other state information.

1. Introduction

The development of blockchain technology has given rise to different types of blockchains, such as public chains, consortium chains, and private chains. However, each blockchain has its own systems and there is a lack of easy interoperability and a convenient means of value exchange. As a result, dApp developers are unable to develop cross-chain dApps based on different blockchain platforms, which greatly limits the applications of blockchain technology, and prevents blockchain technology from wider business adoption.

1.1. **Ontology multichain architecture.**

To improve the interoperability between blockchains and their value exchange, and enable wider business adoption, Ontology has proposed a lightweight, low-coupling, safe, and reliable multichain system and cross-chain solution.

Ontology's multichain design uses a two-layer structure. Ontology, as the main chain, is responsible for the registration of side-chains and asset staking. Both homogeneous and heterogeneous side-chains are supported. Side-chains need to register with the main chain, and after the application is approved by the Main Chain Governance Committee, the registered side-chains are able to interact with the main chain or other side-chains.

The block information on the main chain is trusted by side-chains, but the block information on side-chains are not inherently trusted by the main chain and other side-chains. To verify the legitimacy of the cross-chain state, side-chains need to submit the block header information about their genesis blocks, consensus epoch, and other basic information to the Multichain Management Contract of the main chain when registering. In addition, the side-chains need to initialize the key block header information of the main chain in their block header synchronization contract. Side-chains also need to stake a certain amount of ONG on the main chain to prevent malicious acts.

The cross-chain interaction between source chain and target chain requires the corresponding proof of

legitimacy for cross-chain transactions. The proof information needs to be relayed through chains when cross-chain transactions are taking place. The proof of legitimacy also includes some key block header information about the source chain or target chain. The interaction between the main chain and side-chains requires them to synchronize key block header information from each other; for the interaction between side-chains, they can obtain each other's key block header information from the main chain.

The Ontology main chain, homogeneous side-chains, and heterogeneous side-chains form the Ontology blockchain network, on which gas fees and mining fees are paid in ONG.

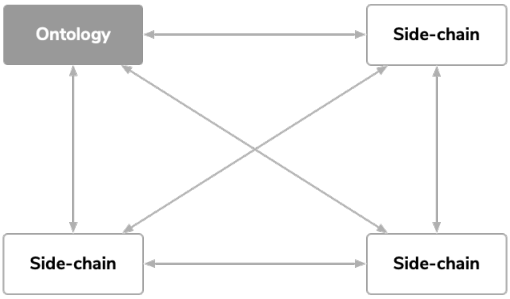

FIGURE 1. Ontology Multichain Design

The Ontology blockchain network supports cross-chain interaction between the main chain and side-chains, and between side-chains. Its cross-chain design consists of the following components:

(1) *Multichain Management Contract*. It is responsible for registering and managing side-chains, including the registration of side-chains, state management of side-chains, and management of side-chains'ONG staking, fund pool, and consensus switch epoch. The contract is deployed on the main chain;

(2) *Block Header Synchronization Contract*. During the process of interaction between the main chain and side-chains, and between side-chains, key block header information must be synced to verify the legitimacy of cross-chain transactions. When side-chains interact with each other, they obtain information about the previous key block header that requires cross-chain communication from the main chain. The block header sync contract is deployed on the main chain and each side-chain;

(3) *Cross-Chain Management Contract*. All cross-chain transactions are managed by the Cross-Chain Management Contract on the source chain. To realize cross-chain interaction, dApp developers only need to follow the instructions of the Cross-Chain Management Contract. The Cross-Chain Management Contract is deployed on the main chain and each side-chain;

(4) *ONG (x) Contract*. The asset contract allows cross-chain transfer of assets, which can be used as transaction fees and mining fees for cross-chain transactions. The ONG contract on the main chain supports locking and unlocking cross-chain assets, and ONGx contract on side-chains can increase and burn cross-chain assets. The ONG contract is deployed on the main chain and ONGx contract on each side-chain.

(5) *Relayer*. A state information synchronizer that constantly monitors requests from the cross-chain contract and some key block headers, and syncs cross-chain transactions or key block headers to earn mining fees.

2. Side-Chain Life Cycle Management

The life cycle of a side-chain is managed by the **multichain management contract**, which is mainly responsible for registering and managing side-chains, including the registration and state management of side-chains, and management of side-chains'ONG staking and fund pool. The contract is deployed on the main chain.

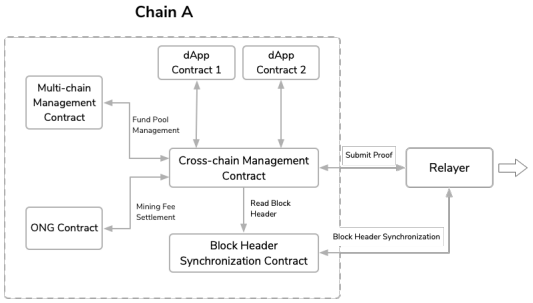

FIGURE 2. Module Design

2.1. **Side-Chain Registration.** When side-chains register with the main chain, they need to use their ONT ID to complete the KYC process and submit basic information such as genesis block information and stake size to the main chain. At the same time, the side-chain needs to stake a certain amount of ONG as insurance to the main chain, which is jointly staked by the initial validator of the side-chain. The amount of the ONG staked by the side-chain validators is agreed by themselves, but should not be lower than the promised total amount of ONG. Before submitting the registration information, the side-chain validators need to stake enough ONG into the main chain. If the staked amount is insufficient, the registration will fail. If there are malicious acts on the side-chain (for example, the side-chain validators increasing the supply of ONGx maliciously), a corresponding amount will be deducted from the margin as a penalty. The main chain governance committee reviews the basic information submitted by the side-chain, and after the approval, allocates a fund pool for the side-chain based on the amount of the staked tokens, which is used to manage the assets used by that chain to interact with other chains. At this point, side-chain registration is completed.

The total ONGx assets on the side-chain are determined by its fund pool on the main chain. In the event of any malicious acts by a side-chain, its staked tokens on the main chain will be used to compensate users. Therefore, normally, to ensure safety, we define the ratio of the capacity of the side-chain's fund pool Tcp to the staked tokens Tsd as:

$$\eta = \frac{T_{cp}}{T_{sd}} \in (0, 1]. \tag{2.1}$$

2.2. **Change of Validator Information.** In the event of a change of side-chain validator information, suppose the validator set of that chain changes from v_1 to v_2, then the new validator needs to go to the main chain before the change, first stake enough ONG, which means the stake amount $T_{sd}^{(v_2)}$cannot be lower than the total ONG of the stake when the old validator was around $T_{sd}^{(v_1)}$, which is,

$$T_{sd}^{(v_2)} \geq T_{sd}^{(v_1)}. \tag{2.2}$$

When the information state synchronization program Relayer submits the validator's block header of the new consensus cycle to the main chain, the main chain verifies whether the relation (2.2) is true. If not, the block header synchronization will be rejected; otherwise, the ONG staked by the old validator of the side-chain will be unlocked after the validator's information has been changed twice.

2.3. **Cross-Chain Asset Exchange.** After the side-chain has been registered, users can lock their ONG on the main chain into the fund pool of the side-chain registered with the main chain, to exchange for the assets on that side-chain accordingly.

After users lock their ONG on the main chain into the fund pool of the side-chain registered with the main chain, the main chain will generate the corresponding IOU information. If the fund pool on the main chain has been filled up, then the locking fails. If the locking is successful, then the Relayer or users can release the corresponding amount of ONGx based on the IOU information generated by the main chain. Reversely, if users burn ONGx on the side-chain, then they can release the corresponding amount of ONG on the main

chain based on the IOU information generated by the side-chain.

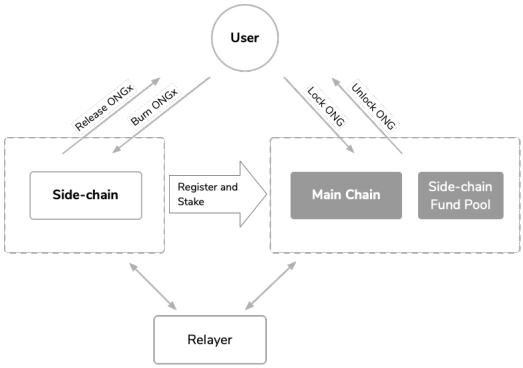

FIGURE 3. Cross-Chain Asset Exchange

2.4. **Side-Chain Asset Management.** If the side-chain wants to increase the capacity of the fund pool on the main chain, it can increase its margin staked into the main chain and submit an application to the main chain. If its application is approved, then its fund pool will be enlarged and the asset supply on the side-chain will be increased accordingly.

Similarly, if the side-chain wants to reduce the capacity of the fund pool on the main chain, it can submit an application to the child chain. If its application is approved, then its fund pool will be reduced and the asset supply on the side-chain will be decreased accordingly. Please note that the margin corresponding to the lower limit of the side-chain fund pool capacity is not allowed to be less than the total amount of ONG currently locked by all users.

2.5. **Side-Chain Logout.** If the side-chain needs to log out, it needs to submit a logout application to the main chain and there will be a challenge period, during which users can convert the assets on the side-chain back to the main chain. After the challenge period, any unconverted ONGx on the side-chain cannot be transferred back to the main chain.

3. Block Header Synchronization

For cross-chain transactions, one of the keys is how to verify the legitimacy of the cross-chain state. When the source chain initiates a cross-chain transaction, the target chain needs to verify the legitimacy of the cross-chain data from the source chain. An obvious way is to sync all the blocks of the source chain, starting from the genesis block. Since blocks contain detailed information about cross-chain transactions and block validators, the legitimacy of cross-chain data from the source chain can be verified. However, given the huge amount of data contained in these blocks, having to sync all of them seems poor design and would require too much work.

Block headers contain the state root of the state tree of cross-chain transactions and validator information. Syncing the block headers alone can also solve this problem. One way is to deploy a light client of the source chain, which can be used to validate the state of a certain block height. However, the block height of the source chain synced by different consensus nodes may be different. When cross-chain transactions are being executed, some consensus nodes may be able to obtain the block headers they need, whereas others may not, resulting in the discrepancy in the state of cross-chain transactions between different consensus nodes. To avoid this, the synchronization of block headers needs to be confirmed by both chains involved in the transaction.

The Ontology cross-chain solution is more intuitive. In the consensus governance model of the Ontology main chain, the Ontology network changes the consensus node between a certain number of blocks, that is, the validator set remains unchanged during a consensus period. Therefore, if the side-chain is a homogeneous chain, then not all the blocks need to be synced, only the key blocks (i.e. the periodic block switch that switches the validator set)

and the block in which the cross-chain transaction takes place. Such design greatly reduces the number of synced block headers. The synced block headers are stored in the **block header synchronization contract**, and any other contract on the current chain can read the synchronized block headers from the contract.

As mentioned above, when the side-chain is initialized, it needs to start a registration request to the main chain and submit its block header information, which contains the information about validators. If there is any change of the validator' s information during the consensus period, the side-chain needs to submit the block header information about the new validator set. Similarly, the cross-chain solution for other heterogeneous chains can also adopt the method of syncing certain key blocks.

In the meantime, in case there is no update on the key block header synchronization on the side-chain after a consensus switch epoch, and the validators from the previous side-chain consensus epoch create malicious block headers, Ontology includes the side-chain consensus epoch management in the **multi-chain management contract** on the main chain. The validators on the current side-chain can alter the attribute. After the side-chain consensus epoch has ended, if there is no new key block header being submitted, then the cross-chain interaction between side-chains and the main chain will be terminated. If the new side-chain validators discover that the consensus epoch on the main chain is inconsistent with the actual consensus switch epoch on the side-chain, they can alter the attribute on the main chain, and the side-chains will be punished accordingly.

The block headers in cross-chain transaction blocks contain the state hash root generated by the cross-chain management contract. When cross-chain transactions take place, the cross-chain state submitted by the source chain needs to be validated, that is, the submitted cross-chain transactions need

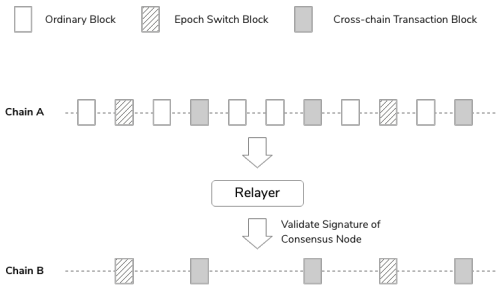

FIGURE 4. Key Block Header Synchronization

to have the Merkle proof of the cross-chain state, and validate the cross-chain state by using the state root of the synced block headers of the source chain and the Merkle proof of the cross-chain state. When the cross-chain transaction is validated, the cross-chain state is returned for the following steps.

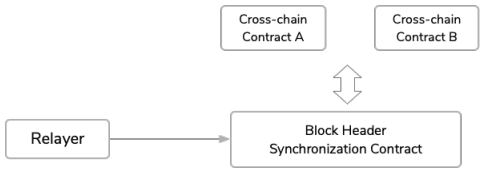

FIGURE 5. Block Header Synchronization Procedure

4. Cross-Chain Interaction

To reduce the cross-chain dApp development costs for dApp developers and help them focus on their business, all cross-chain transactions are managed by the cross-chain management contract on the source chain. All dApps need to do is to call the cross-chain management contract, which makes developing cross-chain dApp contracts much easier.

dApps only need to deploy the corresponding contracts on the source chain and target chain to realize cross-chain interaction. When the business

logic on the source chain is realized and you are moving on to the target chain, call the cross-chain method in the dApp cross-chain contract on the source chain, determine the contract address and calling method on the target chain, and the Relayer will sync the cross-chain transaction. When the cross-chain transactions are sent to the target chain, its cross-chain management contract will call the corresponding business logic according to the contract address and call parameters determined by the source chain.

4.1. **Interaction Between Main Chain and Side-Chain.** When the dApp is processing the cross-chain interaction between the main chain and the side-chain, the dApp contract first processes its logic on the source chain and needs to call the cross-chain API of the cross-chain management contract. After the Relayer syncs the validated state information to the target chain, the dApp continues to process its logic on the target chain. The procedures are as follows:

(1) The cross-chain management contract on the source chain will assign an ID for each cross-chain transaction, put the transaction into the Merkle Tree, and then the Merkle Root will be put into the block headers of the block. After this, the ID and block height will be broadcast through an event. At the same time, when initiating cross-chain transactions, users need to burn or freeze a portion of ONG on the source chain as mining fees.

(2) The Relayer is responsible for monitoring these cross-chain events. When the Relayer is monitoring a cross-chain transaction, it can obtain Merkle proof on the source chain according to the ID and block height, and then submit the Merkle proof to the target chain, during which a mining fee needs to be paid.

(3) After the target chain (its cross-chain management contract) receives the Merkle proof, it will obtain and validate the completed cross-chain transaction on the source chain, and label the transaction ID as "spent". Then, according to the cross-chain transaction parameters, it will call the dApp contract on the target chain and complete the dApp contract logic on the target chain.

(4) After the cross-chain transaction is successfully executed on the target chain, the Relayer will receive incentives accordingly. Based on transaction types, the incentives could be ONGx issued by the ONGx contract on the side-chain or ONG released by the multichain management contract on the main chain.

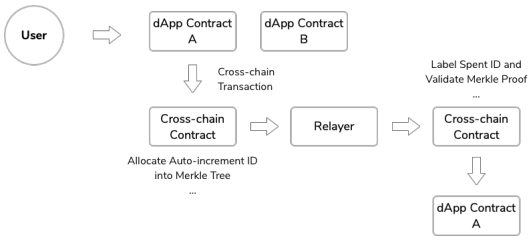

FIGURE 6. Inter-Chain Interaction Procedure

Relayer needs to pay the transaction fee for the target chain when submitting state information to the target chain, so the Relayer needs incentives to cover this cost. Users of cross-chain transactions can submit state information themselves, or they can include a miner fee in a cross-chain transaction to entrust the Relayer to motivate the transaction. Specifically, when the user conducts a cross-chain transaction on the source chain, it needs to lock a part of the ONG or burn part of the ONGx as the miner fee[1], After the Relayer monitors the cross-chain request, it will set whether the miner's cost

[1]If the mining fee is too low, say 0, then users may need to submit the proof themselves since no Relayer would have incentive to do it.

is higher than the self. A threshold is set to decide whether to submit this cross-chain transaction. When it is higher than the miner's fee threshold set by itself, the Relayer will submit the cross-chain transaction to the target chain. After the cross-chain transaction is successfully executed, the release on the target chain will correspond to the ONG or the ONGx corresponding to the additional issue as the miner's fee. This ONG or ONGx will be forwarded to the Relayer that submitted the successful cross-chain transaction to motivate the Relayer.

(1) When users initiate a cross-chain transaction on the main chain, they will lock up the mining fees paid for cross-chain data transfer in the multichain management contract on the main chain. After the transaction has been executed successfully on the side chain by the Relayer, the cross-chain management contract will then increase the amount of ONGx in the side-chain ONGx contract accordingly to pay for the mining fees of the cross-chain transactions.

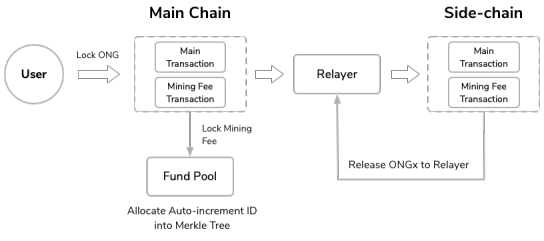

FIGURE 7. Transfer of Mining Fees from Main Chain to Side Chain

(2) When users initiate a cross-chain transaction on the side chain, they can use ONGx on the side chain to pay the mining fee, which will be burned in the side-chain's ONGx contract. When the Relayer performs the successful transaction on the main chain, the main chain cross-chain management contract will release the corresponding ONG from the multichain management contract to the Relayer as the miner's fee.

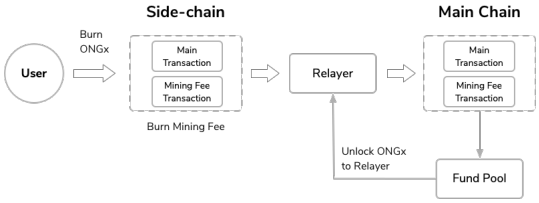

FIGURE 8. Transfer of Mining Fees from Side Chain to Main Chain

4.2. **Interaction Between Side-Chains.** Interaction between side-chains and the main chain and side-chains have two differences: the way of obtaining the information about the key block header; and the way the Relayer receives mining fees.

The information of all the side-chains is recorded on the main chain. If chain A and chain B want to establish a direct connection, they do not need to register with each other, nor do they need to sync each other's key block header information from the genesis block, since both of them can get the key block header information about each other from and verified by the main chain. They only need to obtain the information about the key block header that comes before the cross-chain transaction. The method of validating the cross-chain state is the same as that of interaction between the main chain and a side-chain.

If chain A initiates a cross-chain transaction to chain B, then users need to lock up a certain amount of ONGx as mining fees into the cross-chain management contract on chain A. After it completes the cross-chain transaction, the Relayer can access the record of the successful transaction from chain B, and then unlock the mining fees on chain A –and vice versa.

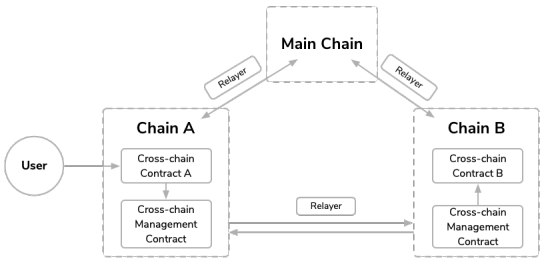

FIGURE 9. Interaction Between Side-chains

5. RELAYER

Anyone can be a Relayer, as long as they deploy an operation status information sync program, continually monitor the cross-chain request of cross-chain contract and the epoch switch block, and sync the cross-chain transactions or block headers.

For each cross-chain transaction or block header, only one Relayer can submit successfully. After the cross-chain transaction is submitted successfully, the Relayer can then receive the incentives for cross-chain interaction. The Relayer does not need to stake tokens and they can decide whether to sync transactions with low mining fees by setting a threshold for mining fees. Whether the Relayer can receive cross-chain incentives depends on the order of transaction execution. Validators who sync the transactions that are executed earlier can receive incentives whereas the later ones will fail and lose their fees.

During the cross-chain interaction process, Relayers need to sync two types of key information: key block headers and cross-chain transactions. Cross-chain transaction fees are paid by users to Relayers who sync data. Key block headers may not include cross-chain transactions, therefore there might be no one who will pay the fees for syncing the key block headers. However, since key block headers will affect cross-chain transactions during the entire epoch, if no Relayer will sync the key block headers, then cross-chain transactions after this key block header in the entire epoch will not be able to proceed. Therefore, a portion of the cross-chain transaction fees in the entire epoch will be used to incentivize Relayers who sync the key block headers. In extreme cases, if there is no cross-chain transaction taking place in the entire epoch after Relayers syncing the key block headers, they will not receive the incentives.

6. SIDE-CHAIN MALICIOUS ACTS

An important security issue involved in cross-chain interaction is how to prevent side-chain validators from acting maliciously. In Cosmos [6]], side-chains are an autonomous system, and side-chain validators are elected by the side-chain itself; whereas in Polkadot [8], the side-chain validators are managed by the Polkadot main chain. Whether it is the autonomous or unified validator election, a fundamental problem is that these side-chain validators are not necessarily reliable. If the asset value of the interacting chains is greater than the value of the validators'stake on the main chain, the validators will have enough motivation to act maliciously.

For example, a dApp developer deploys smart contracts on both the main chain and the side-chain, hoping to achieve cross-chain asset interaction. When dApp users transfer a part of their assets to the side-chain, the side-chain validators can directly transfer the assets to themselves if they find out that the asset value is greater than the value of their stake on the main chain. Then they can transfer the assets onto the main chain and sell them on the exchange. Of course, the side-chain validators'stake on the main chain will be used to compensate the users. Nonetheless, if the value of users'assets is greater than the value of the validators'stake on the main chain, then there is a high possibility the validators will make a colluded attempt to steal the assets.

6.1. **Method of Malicious Act.** Colluded malicious acts by side-chain validators can take two forms, which will affect the interaction between the main chain and side-chains and between side-chains:

(1) *Malicious Blocks.* Side-chain validators can generate two blocks at the same block height, which will cause forking. When the consensus epoch switches, the current side-chain validators can create a malicious block header to make itself or others the consensus validators in the next consensus epoch, whereas the other block header is the correct one. If the malicious block header is being synced to the MainNet before the correct one, all cross-chain transactions will be terminated on that side-chain, despite the side-chain validators not being able to make any profit out of it.
(2) *Malicious State Root.* Existing cross-chain solutions mostly adopt Merkle Tree Proof, that is, side-chains will generate in each block a State Root containing the state of all the transactions in the current block, and side-chain validators will sign that State Root. When a cross-chain transaction is taking place, the cross-chain state can be validated by validating the State Root. If the asset value of the interacting chains is greater than the value of the validators'stake on the main chain, then the side-chain validators can forge a State Root based on the current block, which means they ignore the executed results of the current block and forcefully create a State Root in their favor, so as to steal the user assets locked on the main chain.

6.2. **How to prevent malicious acts.** In response to the forking problem at the same block height, we can set a challenge period during which anyone can submit the two block headers at the same block height to the main chain to prove the malicious behavior.

In response to the inconsistency of State Root of the current block, we can set a challenge period during which anyone can do the following:

(1) Submit the block where the malicious act takes place;
(2) Submit the previous state proof right before the malicious transaction;
(3) Submit the malicious smart contract;
(4) Check whether the State Root generated in the corresponding virtual machine is consistent with the State Root of the current block.

We can see that validators do malicious acts by making a colluded attempt to forge a State Root in the current block and the transactions in the block cannot be altered as user signatures cannot be forged. We have come up with an idea to solve this problem. During the challenge period, if a malicious transaction is spotted, we can run the malicious block, transaction in the block, the previous state of the transaction in the block, and the malicious smart contract on the corresponding virtual machine, and then compare the State Root generated to the State Root of the malicious block to see if that State Root is valid or not.

In the meantime, whether there are cross-chain transactions taking place or not, the Relayer will monitor the side-chains in real time. If the Relayer discovers that there are two block headers at the same block height or the State Root of the current block header is inconsistent with the State Root that is actually in operation, it can immediately submit the proof to the main chain, prove the malicious behavior on the side-chain, and receive the incentives the side-chain validators staked on the main chain.

Note: *The method of validating the State Root in the block is quite complex, especially for heterogeneous chains. In addition, the challenge period is not user-friendly. We will continue to improve on this solution and come up with more feasible and efficient ones.*

7. Conclusion

We proposed a brand new lightweight, low-coupling, safe, and reliable multi-chain system and a cross-chain solution. The solution uses the Ontology blockchain as the main chain, and supports both homogeneous and heterogeneous side-chains. It allows the interaction between the main chain and side-chains, and between side-chains.

Future work:

(1) A more viable solution to colluded malicious act by side-chain validators.
(2) A more efficient multi-layer blockchain network architecture, which is an efficient scaling solution that support multi-layer side-chains.。
(3) In the Ontology cross-chain solution, homogeneous chains which have the same architecture as that of Ontology all have finality, but this is not necessarily true for heterogeneous chains. Heterogeneous chains which have finality can use the cross-chain solution just like homogeneous chains. However, for heterogeneous chains which have no finality, Cosomos[6] adopts PegZone solution and Polkadot[8] uses a relay chain solution. We will continue to look for more feasible solutions for heterogeneous chains without finality.

References

[1] Polkadot-lightpaper. 2017.
[2] Vitalik Buterin. Chain interoperability. 2016.
[3] Vitalik Buterin. Minimal viable plasma, 2018.
[4] Vitalik Buterin. Plasma cash: Plasma with much less per-user data checking, 2018.
[5] Poon Joseph and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. 2017.
[6] Jae Kwon and Ethan Buchman. A network of distributed ledgers. 2016.
[7] Gautier Marin. Understanding the value proposition of cosmos. 2018.
[8] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. 2016.

Appendix A. Related Work

Chain interoperability[2] was first raised by Vitalik Buterin in September 2016. It enables interaction between different blockchain platforms simply through the blockchain protocol itself and without third-party involvement (such as exchanges). Blockchain interoperability protocol can be divided into the following models: notary schemes, side-chains/relays, and hash-locking. . At the moment, both Polkadot[1] and Cosmos [7] are dedicated to creating network protocols between blockchains in an effort to realize secure and reliable interaction. New blockchains based on these protocols can send transactions and messages between them.

A.1. **Plasma.** Plasma [5] is a layer 2 scaling solution first proposed in August 2017 by Joseph Poon and Vitalik Buterin. It focuses on providing increased scalability. This is done by moving transactions onto faster and less crowded side-chains. These side-chains are secured by the underlying root chain and users can make transactions freely off-chain, which interact as sparingly as possible with the main chain. Plasma users can always exit the side-chain and recover their funds on the main chain if they detect malicious node behavior. The solution moves most of the computation off-chain and submits the results to the main chain regularly or when necessary to ensure finality. The core principle behind layer 2 scalability is that with a consensus based on the root chain, smart contracts and other means are used to achieve off-chain and on-chain interaction, and off-chain users can still return to a certain on-chain state in the event of fraudulent behavior.

The Plasma design is divided into Plasma MVP [3] and Plasma Cash [4]. Plasma MVP is achieved using Merkle proof and UTXO. When funds need to be returned from the side-chain back to the main chain, users need to provide the Merkle Path Proof containing the UTXO of the withdrawal, the

block where UTXO is located, and UTXO. It is worth noting that a malicious user can use the spent UTXO to unlock the funds on the main chain. In order to solve this problem, MVP has added a challenge period in the design. During the challenge period, the Merkle proof showing that the UTXO has been spent can be submitted to the main chain to solve the problem. At the same time, in order to punish the malicious actor, exitors need to stake a certain number of tokens on the main chain. If proved to be malicious actors, those who withdraw will be punished. If the side-chain validators are found to act maliciously, an exit queue will be forcefully executed by the smart contract in the MVP to ensure that the previous UTXO exits first. However, a few things need to be considered in the design of MVP:

(1) One issue is how validators exit orderly when they act maliciously. Also, a lot of proof and computation is needed to validate UTXO.
(2) The UTXO model has limitations as it cannot label all assets.

Plasma Cash is more mature than Plasma MVP. Since UTXO is not suitable for all assets, Plasma Cash uses NFT to label on-chain assets, instead of the UTXO transaction model used by MVP. To fully prove and protect the ownership of token holders, Plasma Cash uses Sparse Merkle Proof, which allows the efficient verification of inclusion and non-inclusion of a transaction in a block. Of course, malicious users can use NFT that has been spent to withdraw assets from the main chain, since Sparse Merkle Proof includes the complete circulation process of NFT. This problem can be solved by setting a challenge period during which the Sparse Merkle Proof of NFT is submitted. Moreover, to withdraw, users need to stake a certain amount, and they will be punished if they act maliciously. Another benefit of using NTF is that it reduces the user checking requirements to only the NFTs that they own without affecting other assets. Nevertheless, Plasma Cash also has its limitations:

(1) When an asset has been circulating on a side-chain long enough, more proof and computation are required as the NFT contains the complete circulation information of the asset.
(2) Merges and splits of NFT are still an issue yet to be addressed.
(3) Cross-chain communication is only limited to transfer of digital assets.
(4) The issue of economic incentives for validators in Zones and Relayers for cross-chain asset interaction is yet to be addressed.。
(5) Using light clients for interaction between Zones and Hubs may result in the inconsistency of cross-chain transaction state.

A.2. **Cosmos.** Cosmos [6, 7] is a heterogeneous network that supports cross-chain interaction proposed by the Tendermint team. Its ultimate goal is to create a blockchain network consisted of a large number of independent and parallel blockchains. The first blockchain in the network is Cosmos Hub, which is the Cosmos MainNet, and the other parallel chains are called Zones, which achieve cross-chain operation with Hubs through the Inter-Blockchain (IBC) Protocol. The IBC protocol is designed for the Cosmos network, and given its timely finality, is used for sending messages between Hubs and Zones. In the IBC Protocol design, before the two chains connect to each other, they need to register with each other, and save the algorithm of each other's validator set and Merkle Proof, so that the receiving chain can validate the message is correct. In the meantime, different Hubs can also interact with Zones through Hub routers. That being said, Cosmos still has some limitations:

(1) Side-chain validators interact with the main chain through staking. During cross-chain interaction, if the asset value of two interacting chains exceeds the staked value of a certain

chain, then a colluded attempt by validators to steal assets is possible.

(2) Cross-chain communication is only limited to digital assets (tokens).

(3) The issue of economic incentives for validators in Zones is yet to be addressed.

A.3. **Polkadot.** Polkadot [8, 1] is a scalable heterogeneous multichain system proposed by Ethereum' s core developer Gavin Wood. It is designed to address the issue of blockchain scalability. Polkadot plans to integrate private chains/consortium chains into a consensus network of public chains while retaining their features, such as data privacy and use by permission. It can connect multiple blockchains. Polkadot sees other blockchains as parallel chains and it can transfer tokens from original chains to multi-signature addresses for a temporary lock-up periods, which seem to be controlled by multi-signature control through relay chain technology. The result of the transaction on the relay chain will be voted by these signees to determine whether it is valid. It also introduces phishers to report and monitor transactions. Cross-chain communication can be achieved by linking Polkadot to Bitcoin, Ethereum, etc.

In the Polkadot design, validators of each shard are uniformly allocated by the main chain and need to have the complete ledger of all shards, and the stored data will be too large. In addition, if the asset value of the two interacting shards is greater than the value of the validators'stake on the main chain, then a colluded attempt by validators to steal assets is possible. Polkadot has the following issues:

(1) Validators become shard validators by staking. If the asset value of the two interacting shards is greater than the value of the validators'stake on the main chain, then a colluded attempt by validators of the child shards to steal assets is possible.

(2) Validators of each shard need to have the complete ledger of all shards, of which the stored data will be too large.