# Darwinia Optimistic Bridge: Sublinear Relay for Interoperable Blockchains

Xiaodong Qian [1] and Xiaoyin Wang[2]

[12]Itering Tech Pte. Ltd
{[1]*alex.chien,* [2]*aki.wu*}*@itering.io*

***Abstract*—To verify transactions, cryptocurrencies such as Bitcoin [1] and Ethereum [2] require nodes to verify that the blockchain is valid. This requirement means downloading and verifying all blocks, which takes hours and requires gigabytes of bandwidth and storage space. Therefore, clients with limited resources cannot independently verify transactions without trusting the full node. Bitcoin and Ethereum provide lightweight clients called simplified payment verification (SPV) clients that can verify the chain by downloading only the block header. Unfortunately, the storage and bandwidth requirements of SPV clients still grow linearly with chain length. Recently, NIPoPoW [3] and FlyClient [4] have proposed a type of solution called super-light client. It is expected that light clients only need to download and store the logarithmic number of block headers, but this type of solution cannot be directly used for light clients on the chain, that is, cross-chain Relay. FlyClient requires making a certain degree of a hard fork to the corresponding chain and supports the Merkle Mountain Range (MMR) commitment before it can be used for relay on the chain, they all have a certain degree of poor generality, Non-Interactive Proof of Work (NIPoPoW) is only applicable to chains with fixed block difficulty, and FlyClient needs to modify the best probability block sampling protocol and variable-difficulty verification model.**

**We introduce Darwinia ChainRelay, which is a novel cross-chain verification relay, using technologies such as Merkle mountain range (MMR) commitment, optimistic verification game, super-light client framework, relay incentive model, etc. Darwinia ChainRelay is able to achieve sub-linear performance by only submitting logarithmic numbers of data including the block header and its derived data. Darwinia ChainRelay overcomes the limitations of FlyClient. The protocols do not need to fork the target chain to use MMR. It has strong generality and can be applied not only to the POW chain but also to other consensus algorithm chains such as POS. Also, by only storing a single block header between two verification executions, we can implement the protocol using the design of on-demand verification, that is, before each verification, we only need to submit one block header and make sure it is confirmed and finalized.**

## I. Introduction

There are many efforts on building decentralized bridge between different public chains for supporting swapping, transfer, and more general verification.

Various technologies have been innovated to trying achieve these goals. Hash Time-Locked Contracts(aka. HTLC) are invented to support atomic swap assets in two different public chains, but cannot support asset transfer and verification. To support transfer and verification, several bridges solutions are introduced, including semi-decentralized custodian model and on-chain light client(aka. chain relay) solution.

Chain relay is a light client running on chain with verified headers relayed from another chain's full nodes. Cross-chain verification transactions must be atomic and unstoppable, we need chain relay to confirm the final state of another chain before using it for cross-chain verification. Chain relay is to build and maintain such an on-chain light client with the help of off-chain relayers and on-chan light client verification.

The earliest on-chain light client we know is BTCRelay [5] built by Consensys, later, Kyber Network also use on-chain light client technologies to build a bi-direction bridge between Ethereum and EOS, which is called WaterLoo [6].

This classic chain relay solutions have challenges of economic infeasible due to it is linear relay which means the maintain cost of the relay is growing linear with the block height.

Here we propose a new solution which can resolve this challenges by eliminate the need of relaying each block header from target chain, and achieve a sublinear relay, we call it Darwinia Chain Relay.

To verify events on target chain, the relay need commitment such as merkle roots of events which are commonly recorded in each block header. In darwinia relay, eliminating the need of relaying each block header will result in lacking of these merkle roots, thus we introduce merkle mountain range(MMR) [7] as a new commitment representing the header's previous blockchain history. Besides, because the relayed block headers are not continuous chained, this will make light client unable to validate header, darwinia relay resolve header and MMR validation by introducing optimistic verification game [8] as sub protocol in relay.

By eliminating the need of relaying each block, darwinia relay can achieve sublinear performance, comparing verifiable blocks on target chain to the fee cost to maintain the relay.

*Our contribution*

- Through the introduction of an economic incentive mechanism, verification users collect fees from the Relay chain, and incentives are provided to honest block header submitters. For fraudulent block header submissions to Relayer, pledge guarantees are based on Slash-like penalties.
- By introducing the Optimistic Verification Game method, the problem of not including MMR in the blockchain head can be solved, and most blockchain protocols can be supported.
- Optimistic Verification Game can also solve the problem of malicious attacks under rational economic market

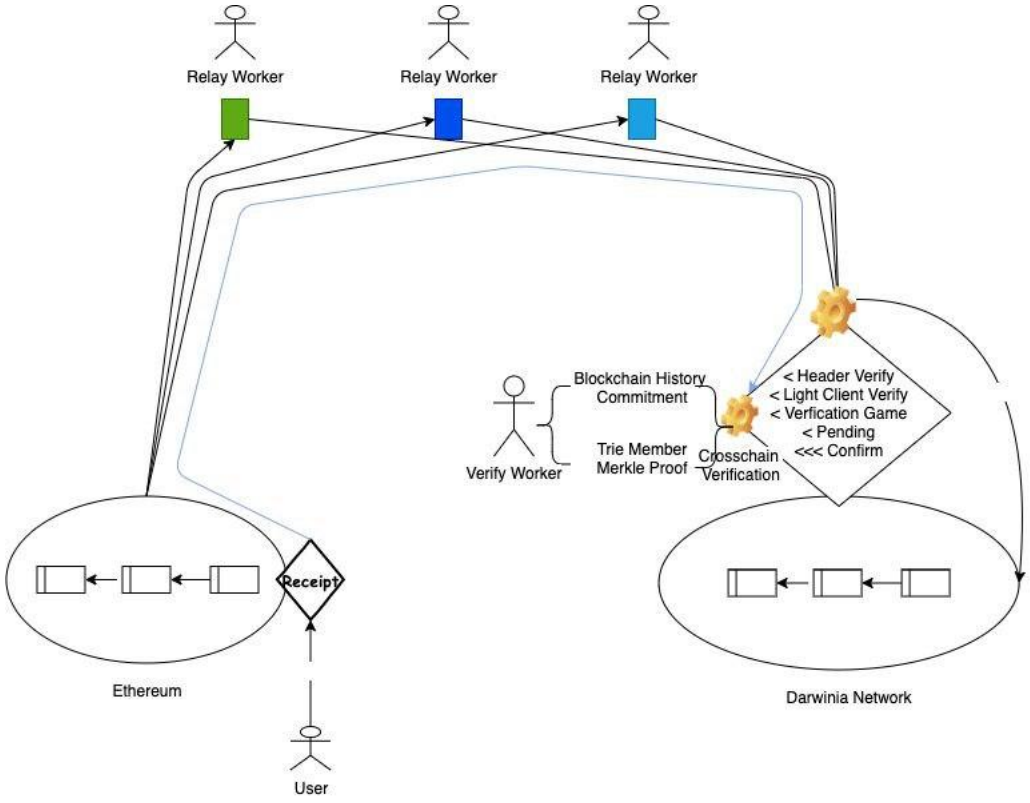



Fig. 1. Overview

games, so it can replace the complex probability-based abstract proof design and heuristic non-interactive design in FlyClient [4].

- Retain the design of MMR and variable difficulty MMR check-in FlyClient design, and realize sub-linear Relay under the premise that the block header is correct

The rest of the paper is organized as follows. Section II provides an overview of our entire system. Section III introduces under what circumstances our system is running, and also introduces the definition of some technical indicators of our system. Then, we introduced each component of our system in section IV with technical details. In the end, we introduced the practical application prospects of this paper.

## II. OVERVIEW

For light clients, they need to be able to connect to a complete set of nodes (called certifiers), at least one of which is honest (holding a copy of a valid chain). The light client does not know which one is honest, so Before the light client updates the status, its verification logic needs to be able to identify and filter out the data submitted by those honest nodes. For the off-chain light client, the problem of having at least one honest node can be solved by connecting to the full node, but for the on-chain light client CR, the status update is not made through a network interaction. It is achieved by submitting data to the chain through blockchain transactions through the prover. The cost will be higher, and that's why we need to design reasonable economic incentives to make this process practical.

The outer layer of Darwinia ChainRelay uses verification games as subroutines. The roles in the verification game include a solver who provides a solution for a specific task and a challenger who disagrees with the solver's solution. The final decision role, that is, the referee, can always perform calculations correctly and get results but has extremely limited resources such as computing bandwidth or storage. Darwinia's referee is the entire group of validators, who reach a verdict through systematic consensus.

### *Relay Worker*

This worker is responsible for verifying, calculating, and submitting the correct block header and its MMR on the chain. Anyone can be a relay worker. The block header and its MMR submitted to the relay on the chain are the solutions. The answer is given by the author. When the solver submits the block header and MMR, he also needs to attach a certain amount of collateral. If a challenger later challenges his results and is found and proved to be cheating by the outer layer, then the system Punishment will be made and its pledge will be confiscated. On the contrary, if there is no challenger to challenge or win in a subsequent verification game, it will prove to be correct and honest, and its pledge will be retrieved and motivated by Relayer Models give rewards.

### *Challenger*

The challenger is an off-chain role. Anyone can take it. It will monitor each block header, and MMR answers submitted to the Relay on the chain and compare it with the value calculated by itself because the off-chain calculation cost is low

( (No on-chain fees are required), so the challenger can easily calculate the correct value. By comparing the correct value with the submitted value of the solver, if there is a problem, the challenger can determine that he can finally win through the verification game. And make money by getting winning rewards. Qualified challengers are rational and keen monitors. They will seize every opportunity to challenge the solver and win the verification game. If the final verification game finds them false alarms, then the challenger will need to pay the extra cost due to false alarms. Resources, and confiscate the pledge fee attached to its challenge. On the contrary, once it proves that the challenge is successful, it will be rewarded systematically. These rewards are likely to come from pledged items confiscated from the other party's solver.

Base on our assumption, there should be at least one honest Relay Worker. As a result, if this honest Relay Worker finds that the block submitted by other people is different from what he thinks is correct, then he will initiate a verification game as a challenger, and reach a consensus in the end.

*Judge*

In our system, Judge is a credible character with fair but expensive computing power. He is a program running on the blockchain. Fig. 2. FLowChart in Section IV shows Judge's behavioral logic.

*Verification game process*

The verification game is advanced by a round of fixed duration. The range of each round must be narrowed relative to the last time to allow the verification game to converge to a specific chain. Otherwise, the verification game will continue forever, and it may fail. In Darwinia ChainRelay, the solver and challenger in the verification game need to follow the proof-or-punishment process, because the challenger may not be able to prove that the submitted value of the solver is wrong in only one round, so If you cannot submit a proof, you can narrow the verification calculation by submitting an earlier block header and its MMR. Once entering the verification game, the challenger and the solvers are zero and the opponents of the game. Before the final end, there will be a countdown clock for each round time. If the countdown zero is reached, the opponent does not continue the verification process (proof -or-punishment or result-or-punishment), then the other party wins and the game ends. On the contrary, if the verification game continues, it will converge or partially converge to a range that can be verified on the external chain.

*Challenge waiting period*

Waiting period for the opponent's challenge. After the waiting period, the opponent will give up and the other will win.

## III. System Model and Problem Formulation

If the blockchain network wants to complete interoperability by supporting and using Darwinia ChainRelay, it needs to have some conditions, which mainly include two parts, support for light clients and can implement Darwinia ChainRelay protocol through smart contracts, runtime or fork.

We assume that there are two blockchain networks, called the target chain (T) and the verification chain (V), and they meet an assumption that $T$ and $V$ are each safe. Among them, $T$ may be based on POW or POS, but in any case, based on a consensus principle, it can determine a consistent, valid chain, and continuously generate new legal blocks, There are ways to verify the finality of its blocks There is a $T$ on-chain light client (CR) on $V$. $V$'s verifier client can verify whether $R$ (T) already exists on T by broadcasting and executing a transaction containing the receipt $R$ (T) on $T$.

Standard cross-chain verification refers to verifying what happens on another chain, including transactions, events, states, etc., but there are premise assumptions for cross-chain verification decentralized security, that is, the two chains and their consensus are all safe:

*Assumption 1:* (SPV assumption). The chain with the most PoW workload follows the network consensus and eventually is accepted by most miners.

In addition, we also need to ensure that the two chains where our relay is located can work normally:

*Assumption 2:* (Cross-chain Assumption). Both the cross-chain target chain and the chain where verification occurs are safe.

The $Assumption2$ is intuitive, because if there's a $51\%$ attack on one chain, then that means existing consensus could be violated. In that case, of course, some on-chain activities confirmed by our relay will be invalid. Previous research has shown that this assumption holds if the attacker has only a small portion of all computing power. Under the SPV assumption, light clients can verify whether a particular transaction is included in the ledger. This ability is achieved by using the Merkle tree root of the block transaction of the blocks stored in the block header. A full node provides a chain of SPV certificates for light clients, including the accompanying node path of the transaction to the Merkle root node in the transaction tree.

*Verifier Model*

The actor's model can be simplified to only connect with two provers and one verifier. We first assume that at least one prover can be honest through the economic incentive model. At the same time, to illustrate the problem, we can assume another prover is a malicious attacker. The two provers update the state of CR by continuously submitting the block header data of T, and the verifier uses the state of CR to verify the existence of the information (state, transaction, receipt, etc.) on T.

*Assumption 3:* (one honest prover) At least there is one honest prover.

As we said before, our system can continue to run properly in the presence of a malicious relay. Then, of course, we need to define the "malicious"in the environment where this system is running.

*Assumption 4:* (economic rationality) Any relay included in our system, if it is malicious, then it must have economic rationality. It will not launch an attack that has no economic value.

Detailed explanation about technical indicator.

*Sub-linear Relay*

To achieve sub-linear Relay, we need to change the original process of sequentially submitting block headers to on-demand block headers. On-demand submitter blocks also mean that new blocks can be submitted at intervals. The challenges are:

- The latest submitted block header The previous block header and many previous block headers have not been submitted, but both the pre_hash verification and the verification of the block difficulty adjustment require the previous block header.
- Although MMR in FlyClient is used as a Chain Commitment, it can solve the block existence and difficulty adjustment before verification. However, because of the limitation of the need to fork the protocol, it cannot be used here. We can only assume that the verification block header does not contain MMR. If MMR is required, we can only maintain and calculate the MMR value at all nodes outside the chain and submit it to Relay, but the correctness of the MMR value is also calculated by the MMR value of the previous block and the Hash increment of the previous block.
- Variable difficulty MMR verification faces the same problem as Challenge 2.

In the absence of the previous block header, a correct and legal block header of a specific height H and its MMR value are given. With the latest and correct block header and its MMR value, it can be used to verify all previous blocks and transactions or events in the block. We use a verification game to verify the latest block header submitted to the on-chain Relay and in section IV we will explain its details.

*Finality*

Using Darwinia ChainRelay for cross-chain verification, its effectiveness depends on the validity of the block's existence proof and the transaction/receipt existence proof (Merkle proof). Among them, the validity of the block's existence proof is related to the correctness and completeness of the on-chain verification logic. Also, the confirmation of the block header and its MMR used in the verification process is related to the validity and termination of the block.

The characteristics related to block termination are called the finality of the chain. The finality of different blockchain networks is different. We mainly discuss two significant categories. One is consensus-based on POW, such as Bitcoin and Ethereum 1.0. The probabilistic finality of the mechanism can ensure that after a certain number of blocks are confirmed, the probability of termination is almost close to 1. Although the probabilistic finality cannot guarantee the absolute termination, because such a network will have the phenomenon of local chain reorganization (Reorg), its mathematical probabilistic meaning of almost the end can already guarantee sufficient security and practicality and resistance to attacks. The other is a BFT-like consensus algorithm or a hybrid algorithm that combines BFT, such as DPOS, etc. The finality of the block is to achieve a certain confirmation when the block (usually exceeds the current 2/3 validator node) After confirming), and it can reach 100% finality, which can guarantee that this block must be on-chain.

When we use Darwinia ChainRelay to verify transactions or receipts, we need to consider whether the corresponding block header (and its MMR value) used for verification has been finalized. Checking finality is also an indispensable step in verification protocol.

*Finality of Verification game*

Another very important property is the Finality of the verification game. If this nature cannot be satisfied, it means that in the event of a dispute, the judge cannot guarantee that it will be able to give the final judgment. This important property gives our system security in the presence of a malicious relay.

## IV. System composition and design details

Darwinia ChainRelay has three key design schemes. The first is to use MMR to solve the scheme without uploading each block header. The second is to use MMR to verify the validity of the block without forking the target chain. Under the assumption of economic rationality, the system is optimized and optimized by using the Verification Game and the pledge incentive mechanism.

*Chain Commitment*

In order to achieve the sub-linear property, we have to make sure that those provers commit their chain before further interaction, because the relay in our system won't download every node's head to achieve economic efficiency. If we don't ask for a commitment in advance, it will be difficult for a judge with limited computing power to efficiently locate the fork point of the two chains involved in potential disputes, because the malicious node can deceive the judge along with the erification game. Finding a invalid node will be much easier with a chain commitment.

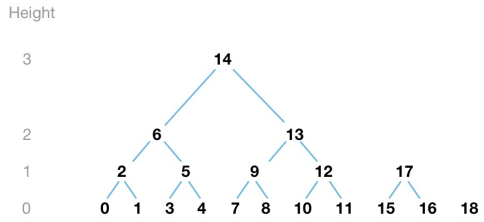
The most basic and vital blockchain commitment data structure is the Merkle Tree. Bitcoin and its SPV clients have widely used Merkle Tree. In some other blockchain networks in the future, some changes in Merkle Tree have also occurred. Such as Merkle Patricia Tree and Merkle Mountain Range(MMR), etc. These variants have added some other useful features while maintaining the characteristics of the Merkle Tree. Here we will only introduce MMR, and the introduction about Merkle Tree and Merkle Patricia Tree is put in the appendix.

*Merkle Mountain Range*

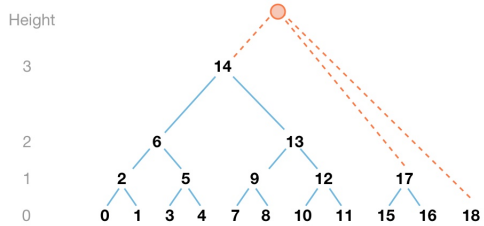
To submit the entire blockchain, Darwinia ChainRelay requires the prover to maintain a Merkle tree structure called Merkle Maintain Range (MMR) on all blocks added to the blockchain so far. In addition to Merkle trees, MMR also allows valid additions on the prover and valid block inclusion verification on the verifier. Also, it enables a valid subtree proof, that is, a proof that two MMRs are consistent on the first k leaves. At each block height i, the prover appends the hash of the previous block Bi-1 to the latest MMR and records the new MMR root Mi in the header of Bi (see Figure 1). As a result,

each MMR root stored at each block height can be viewed as a commitment to the entire blockchain that reaches that height.

Merkle Mountain Range is a variant of the Merkle Tree data structure. Unlike Merkle Tree, MMR can quickly build new MMRs by adding leaf nodes to the existing tree structure incrementally. Below is an example of an MMR structure:




Here is a picture showing how to calculate root:




**definition**

A Merkle Mountain Range, M, is defined as a tree with n leaves, root r, and the following properties:

- M is a binary hash tree.
- M has depth log2 n.
- `If n>1, let n=2i +j such that i=log2(n1):`
  - `r.left is an MMR with 2i leaves.`
  - `r.right is an MMR with j leaves.`

**Note**: M is a balanced binary hash tree, i.e., M is a Merkle tree. Therefore, for all nodes $k \in M$ , $\exists \Pi k \in M$.

*Theorem 1:* (Incremental) Given an MMR, M, with root r and n leaves, AppendLeaf(r,x) will return an MMR, M', with $n+1$ leaves (the n leaves of M plus x added as the right-most leaf).

*Theorem 2:* (SubTree Root Generation) For $k \leq n$, given $\Pi xk \in Mn$ , i.e., the Merkle proof that leaf xk is in Mn , a verifier can regenerate rk, the root of Mk.

*Corollary 1:* (Chain Prefix Commit) If $x1, ..., xn$ are the hashes of blocks 1 through n of chain Cn, rn commits the first n blocks to xn, and $\Pi k \in Mn$ for any k commits $x1, ..., xk$ as the blocks of the chain Ck, where chain Ck is a prefix of chain Cn.

*Corollary 2:* (Block Commit) If an adversary changes any block i in the chain in any way, then its hash xi also changes, so any MMR Mk for $k \leq i$ with root rk' that contains the new block xi' has that $rk'/ = rk$.

Darwinia ChainRelay is a sub-linear Relay, it cannot store all the blocks on the blockchain, and the verifier can trick the Relay by submitting a non-existing block certificate. To solve this problem, we need to append a new MMR root value to the latest block. At each block height i, the prover appends the hash of the previous block Bi-1 to the latest MMR and adds the new The MMR root Mi is recorded in the header of Bi. This MMR root value can be understood as a commitment to integrate the history of the blockchain. The prover of Darwinia ChainRelay can submit the block header and the MMR value together by maintaining the block header and its MMR value. (How to ensure the validity of the block header and MMR value is another issue, which we will solve by verifying the game protocol).

If MMR is applied to all block headers of the blockchain, that is, all block headers are used as leaf nodes of the MMR, the prover will be able to efficiently attach the MMR to the block header and push it to the light client or relay at the same time. With MMR, the verifier will be able to use the latest block MMR and verification block for efficient block inclusion proof, without requiring Relay to store all historical blocks. In addition, MMR can also be used for subtree proof, that is, the proof that two MMRs are consistent on the first k leaves.

The Merkle Mountain Range (MMR) model is an effectively updateable commitment mechanism that allows the prover to commit to the current blockchain with a small (constant size) commitment, and the inclusion proof of the block is logarithmic.

### *Chain Relay MMR Implementation*

To implement MMR in an ordinary light client, you need to make some modifications to the protocol, and add the MMR value as a part of the block header. But for Relay, which is a light client on the chain, there is some additional complexity in the implementation.

First, because the protocol of the target chain cannot be modified, the MMR value needs to be submitted to Relay as an addition to the block header, not as part of the block header. Second, suppose that we already know the Header and MMR values of the previous block on the chain (in more complicated cases, we hand over the verification game protocol). When the Relay on the chain verifies the block header and its MMR, because we only have the MMR root Value, does not store the leaf nodes of all MMR blocks, so it is not possible to directly calculate the next block corresponding to the MMR tree from the previous MMR tree plus the next block header hash. In order to solve this problem, we divide it into two steps. First, calculate the MMR and its root value off-chain and submit the MMR root value to the chain. At the same time, the corresponding Merkle certificate of the previous block needs to be submitted. Second, after receiving the MMR root value on the chain, it is verified and passed. The Merkle proof of the block and the MMR root value is calculated to calculate the MMR root value of the previous block, and compared with the MMR value of the previous block submitted and verified before if they are consistent, the verification passes. In this way, it is not necessary to store the entire MMR tree corresponding to each block on the chain, but only the corresponding MMR root.

### *Verification Game*

The goal of the interactive verification game is to provide a dispute resolution mechanism between the problem solver and the questioner, where the problem solver provides a solution

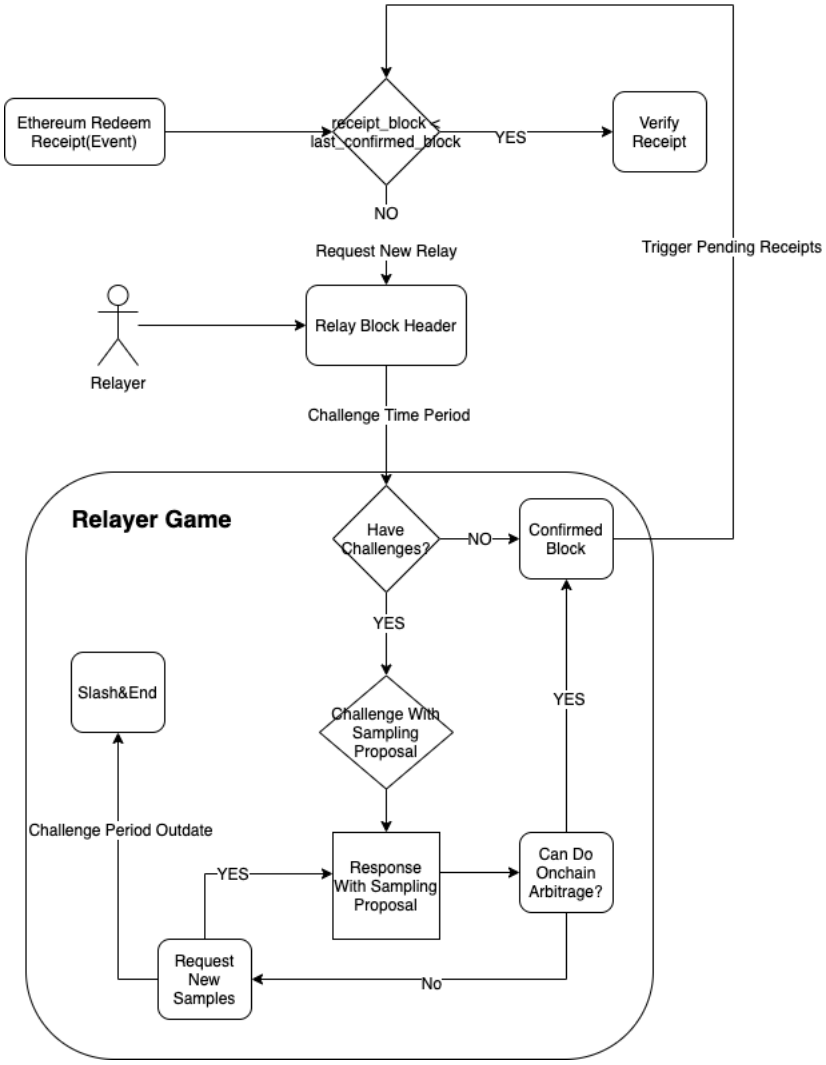



Fig. 2. FLowChart

to a computing problem, and the questioner does not agree with the solution, so Wake the verification game to achieve the purpose of the arbitration. This arbitration process is generally iterative and convergent.

The verification game protocol is used in Darwinia Chain-Relay to solve the problem of attackers submitting illegal chains. The solution is to introduce economic games to make honest nodes motivated and have methods to find the wrong submissions and ensure the correctness of the state of light nodes And legitimacy. In order to introduce a verification game, two conditions are required. One is to have an incentive system and guarantee openness to attract enough honest participants to participate in it. The second is to prove honestly when an attacker submits wrong block data. The author can have a way to prove its error. Since the performance and economic feasibility of the verification and disapproval process must be considered, this process can be either one-time fast, iterative, and convergent. The process of iteration and convergence is acceptable because this affects the participants of the system to an optimal balance, that is, under the premise of economic rationality, most malicious people will not launch attacks.

The verification game will go through a series of rounds, each round reducing the scope of the controversial calculation. In the first round, the challenger forces the solver to perform a determined and timed pair of calculation steps. In the next round, the challenger repeatedly challenges a subset calculated by the solver during this interval, and then continues to challenge in a subset of the subset, and so on, until, in the final round, the final challenge becomes minor enough that the judges can make a final decision on whether the challenge is justified. The referee also requires the solver and challenger to follow the rules of the game. At the end of the verification game, either the cheating solver is found and punished in the outer layer of Darwinia ChainRelay, or the challenger pays the resources consumed by the false alarm.

### *Verification Protocol*

With our actor's model, which is mentioned at section III, we can describe a protocol process. If both prover nodes have submitted the same block header and block, and their chain heights are the same, the client can directly accept this submission, and this part of the agreement ends. Otherwise, if the data submitted by the two provers are inconsistent, it means that one of the provers holds an invalid chain. In this

case, the system will use a sub-protocol called a verification game to determine which of the two provers submitted is the chain of honesty.

- Verifier has access to r = root of some Merkle tree, MT.
- The Prover has access to MT and generates a Merkle-Proof path of some $x \in MT = \Pi k \in MT$ using protocol 3 and sends it to the verifier.
- Verifier uses the proof and x to build up the path to r' using protocol 4, and checks that r' = r.
- If the checks pass, the Verifier accepts the proof, otherwise, it rejects the proof.

*On-demand Verification Design*

According to the commonly used collaboration process in Chain Relay, Darwinia ChainRelay steps can be summarized as follows:

- Step A: A prover (also known as Relayer or Worker) submits the block header and its MMR value to the on-chain relay.
- Step B: The on-chain relay performs the block header and MMR validity verification process. The central part includes a verification game protocol, and some other provers may participate in it.
- Step C: Relay on the chain determines the finality of the block header
- Step D: The verifier (also known as the verification service user) submits the transaction or receipt that needs to be verified to the chain, uses the MMR to prove the existence of the block and uses the corresponding Merkle tree root of the block header to conduct the existence proof of the transaction (receipt).
- Step E: Other on-chain operations performed after the cross-chain verification passes (or fails).

Even further, we can make some overall optimizations to this process, which can save unnecessary on-chain submission operations. Because the requirements generally come from steps D and E, we can improve and design a relay that the prover submits on demand, that is, the three operations A, B, and C are not scheduled or required for each block, but when the user executes step D and finds that the relevant block header and MMR certificate are missing, then A, B, and C operations are performed. The optimized steps are as follows:

$$Verifier \longrightarrow Pre\_D \longrightarrow A \longrightarrow B \longrightarrow C \longrightarrow D \longrightarrow E$$

*Economic Incentive Model*

To ensure the decentralization and sustainability of the agreement, we need to design the agreement so that it does not require a license and introduces economic incentives so that the entire system can continue to run autonomously.

In Darwinia ChainRelay, the economic incentive model is mainly divided into the Relay framework layer and the verification game sub-protocol layer. The former is a general economic model of Relayer. It is used to motivate the prover (also called Relayer or Worker) to submit block headers and MMRs to the system to maintain the Relay status and provide verification services to the verifier. In general, it is used to encourage proof The income of the verifier comes from the service fee paid by the verifier. For a specific Relayer, the corresponding Relayer status check must be completed. Therefore, the workload and cost of the prover to submit the block header and its MMR are relatively fixed, and the verification service The demand comes from the market and application needs. If the service revenue can be kept greater than the cost, we call the design of the Relayer economically feasible. For the traditional linear relayer, it is necessary to submit the block header to maintain the relayer state continuously to continuously submit the block header to maintain the relayer state, so the fuel cost is very high, and it is not economically feasible in many scenarios, such as BTCRelay. Darwinia ChainRelayer, as a sub-linear relay optimized by on-demand verification, can achieve good economic feasibility, and the verifier only needs to pay a small fee to complete cross-chain verification.

At the same time, Darwinia ChainRelay introduced a sub-protocol of the verification game to achieve a sub-linear relay. Therefore, in addition to the economic incentive design related to the relay, we also need to design a corresponding economic model for this verification game protocol. Because economic rationality is an essential premise for verifying game protocols, it is through mechanisms such as the Prove-or-punish introduced in the verification game that this hypothesis of economic rationality can work.

*Relay Economic Incentive Model*

In the general relay, the prover is responsible for submitting the block header and its MMR to maintain the relay state, and the verifier uses the verification service provided by the relay. The prover can be anyone, but the submission of the block header and its MMR is not free, and network fees need to be paid. To ensure that enough certifiers are willing to participate in it, maintain a competitive market, and be sustainable, verify the pricing of services There is a need to provide a premium above the average cost level. For the traditional linear relay, a simplified way is to calculate and accumulate all the block submission costs and accumulate them into a cost contribution pool. When providing verification services, use the average level of the cost contribution pool in the past as a reference. Under this model, because the use of future verification services is difficult to predict, the price of verification services will continue to fluctuate, and the contribution of the workload provided by the prover will be at risk of making ends meet.

Price of verification service = (cumulative fuel fee submitted by the block header within T / number of times verification service is used) + premium

Because Darwinia ChainRelay has the optimization of on-demand verification, the cost of its verification service is also simplified, so this model can be simplified to:

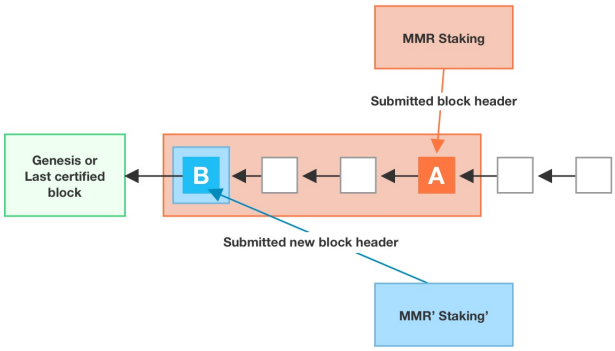
Verification service price = block header submission fuel fee + premium

The optimization of on-demand verification has two benefits. The first benefit is that before using the verification service, you can determine the price of the verification service and decide whether to continue using the verification service. If you do not continue, you do not need to submit the block header, and

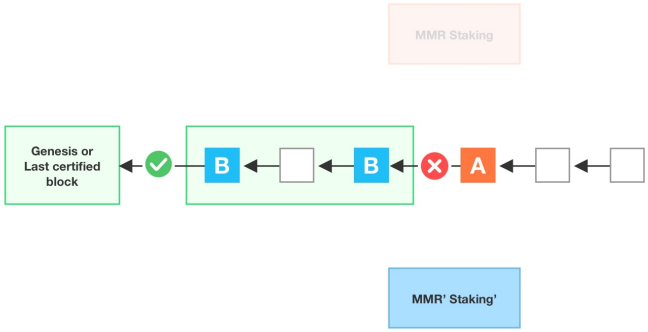
the prover does not need to submit the block header, so you do not need to pay for fuel costs, which reduces waste. It can be seen that in this case, because the prover has no risk of making ends meet, anyone is willing to participate in it if they have the ability. The second benefit is that because the cost of submitting the fuel fee for the block header will not change significantly, and under the condition of sufficient competition, the premium will also be very meager low, so it can continue to provide more stable verification service prices.

*Verification Game Economic Model*

In Darwinia ChainRelay, after the prover submits a block header and MMR value because its previous block header status is generally not submitted or unknown, it cannot be immediately determined whether it is valid or not. Observer A in the figure below might be the first submitter, but after entering the verification game protocol, it is impossible to determine whether each observer is honest until it has wholly converged, until a block header record that has been submitted is the most recent The block header to be verified is the previous one. In this case, we use the Pre_Hash and MMR subtree theorem to determine the validity of the submitted block header.




In the whole process, the design goal of the pledge punishment system is to make the verification game agreement end as soon as possible, so as to achieve the effect and equilibrium of the optimistic verification game. Therefore, when each challenge observer further advances the verification game process by submitting a new block header, a fixed value pledge is required to be locked in the protocol. After the convergence is judged, the honestly submitted pledge will be returned but the collateral submitted by the fraud challenger will be confiscated by the system. These confiscated materials can be used to reward the winner of the final verification game process, that is, the honest prover. We have previously proven that honest provers must win the verification game process.




In the above verification game harvesting process, the process of submitting the pledged items for confiscation and winning back is gradually progressive (recursive). In the simplified case, there may only be two observers and opponents playing against each other, but the more complex multi-party game situation is also possible, so during the submission and challenge, the opponent relationship between the players will be recorded. The last submitter of the challenge is the opponent of the challenger, and the pledged items confiscated after the failure of the opponent will be returned to another party. The figure above shows the process of the pledge of the fraud prover being gradually harvested by the honest prover.

It can be seen that once entering the verification game protocol, the attack cost of the fraud prover is very high, and it is proportional to the time of the attack, and the benefit of the honest challenger is very considerable, and multiple honest challengers can cooperate and work together to fight against the fraud prover, the pledge of the fraud prover will be confiscated and rewarded to the honest challenger.

*Optimistic Behavior Analysis*

The above verification game seems to have a disadvantage. Once it enters the verification game, although it is convergent, its time-consuming may be very long. However, after further analysis, we found that if the assumption of economic rationality is introduced, due to the punishment and incentive mechanism, the attacker is not willing to enter the verification game process because the game participation is sufficiently open. Eyeing enters the verification game, and the perpetrators will eventually fail with certainty, and be punished for bearing the attack losses. Because the process of verifying the game can be participated by anyone, it is only necessary to have at least one honest challenger (monitor) to ensure that the final block confirmed by Darwinia ChainRelay is legal. For economically rational participants, it is impossible to attack, because the attack has no possibility of reward or success, but it may bear the penalty of forfeiture. Therefore, it can be concluded that the vast majority of the solutions provided by the solver are correct, and no subsequent verification process will be performed. The block header and the MMR submitted by the solver will be deleted after a short challenge waiting period. confirm. Only a small number of unintentional procedural errors may enter the short-term verification game process. Participating in an unintentional error cannot be discerned whether it is intentional and affects system reliability, so it will also be punished.

**DOS** If we assume that the attacker is non-economically rational and willing to pay a certain cost to attack the system,

then Delay Confirmation attacks will be a more common type, that is, the attacker pays the price of the pledged fee for the penalty to extend the verification game as a result, a block header in Darwinia ChainRelay, namely its MMR, cannot be confirmed for a long time. However, it is worth noting that its attack cost is linearly related to its Delay Confirmation attack effect (time duration), so it can be suppressed by adjusting the penalty parameter, and this theoretical possibility exists in many networks, which is not a severe issue.

*Put Things Together*

With the help of MMR and verification game protocols, we can assemble a sub-linear Relay on the chain to provide cross-chain verification services. The Relay on the chain stores the discontinuous block header and its derived MMR value. This value is submitted by the prover and verified and verified by the verification game protocol. There is a premise that at least one honest prover participates in the Agreement process.

When a block header and its MMR value are stored and confirmed by the relay on the chain, then the validator can use this valid information of the relay for cross-chain verification, and the validator can verify any transaction or receipt before the block is changed.

## V. Darwinia ChainRelay Applications

Here we need to briefly introduce the characteristics of our products

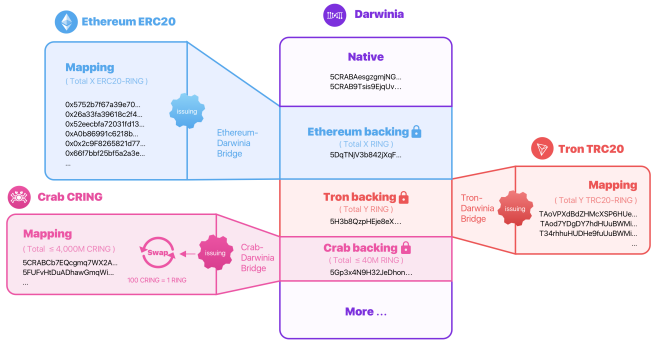
*Token Bridge*

The cross-chain token bridge refers to the asset transfer channel between two heterogeneous chains. Through the token bridge, assets can be safely and reliably transferred on different heterogeneous chains. The subsequent token bridge solution in this article will be backed by the concept of cryptocurrency backed asset(CBA) model and decentralized backing technology.

*Cryptocurrency Backed Asset Model(CBA Model)*

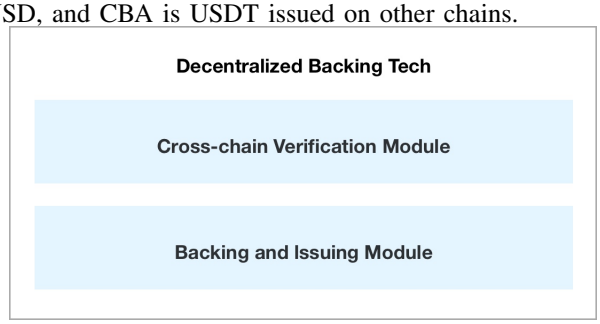
To describe the solution more accurately and clearly, we use the cryptocurrency backed asset (CBA) model to describe it. In the cross-chain CBA model, two heterogeneous chains are called backing blockchain and issuing blockchain. Cross-chain assets do not exist natively on both chains, but through backing technology, by locking as backing assets on the backing chain, while issuing assets on the issuing chain for cross-chain circulation. The assets issued by backing native assets on the backing blockchain are called backed assets, or CBA for short. The security and redeemability of backed assets are determined by the security and reliability of decentralized backing technology.

Darwinia cross-chain token bridge is a cross-chain bridge solution between the backing chain and the issuing chain, and it is also a decentralized asset backing technology.




*Decentralized Backing Technology*

Decentralized backing technology is a more accurate description of cross-chain asset gateway technology, which mainly includes cross-chain verification modules and backed issuing modules. In the case of not strictly distinguishing the degree of centralization, the backing method also includes the traditional centralized backing method, for example, a method operated by an external institution, such as USDT, the backing asset is USD, and CBA is USDT issued on other chains.




Common decentralized backing techniques include:

- There are trust nodes or custodian mechanisms, where the trusted node or custodian uses multi-signature / threshold signature and other multi-centralized technologies to avoid single points of failure, such as Parity Bridge, ChainX, etc.
- Some solutions use collateralized bridges to ensure the redeemability of cross-chain assets, partially combined with Chain Relay. XClaim has the disadvantage that it is only suitable for homogeneous assets with good liquidity, and its economic feasibility is relatively weak.
- Fully use the Chain Relay for chain verification, such as BTCRelay, WaterLoo, Darwinia ChainRelay, etc. The prerequisite of using this solution is that the issuing chain needs to support the implementation of smart contracts, on-chain runtimes, hard forks, and other methods. Such blockchains like Chain Relay and Bitcoin network cannot be used as an issuing chain. The advantage is that it can support a variety of native assets, including illiquid assets and NFT assets. Economic feasibility depends on the type of Chain Relay. Sub-linear Chain Relay can achieve better economic feasibility.

*Darwinia Cross-chain Token Bridge*

Darwinia's cross-chain token bridge solution is a two-way cross-chain token bridge based on chain relay. The chain relay

we are going to develop adopts the design of the Darwinia Sublinear Relay, which has better performance and economic feasibility.

## REFERÊNCIAS


[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[2] "Ethereum whitepaper," https://ethereum.org/en/whitepaper/, accessed: 2020-07-09.

[3] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 505–522.

[4] B. Bünz, L. Kiffer, L. Luu, and M. Zamani, "Flyclient: Super-light clients for cryptocurrencies," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 928–946.

[5] "Btc relay," http://btcrelay.org/.

[6] "Waterloo — a decentralized practical bridge between eos and ethereum," https://blog.kyber.network/waterloo-a-decentralized-practical-bridge-between-eos-and-ethereum-1c230ac65524, accessed: 2019-02-26.

[7] "Merkle mountain ranges," https://github.com/mimblewimble/grin/blob/master/doc/mmr.md, accessed: 2019-11-26.

[8] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," *arXiv preprint arXiv:1908.04756*, 2019.

[9] "Collision resistance," https://en.wikipedia.org/wiki/Collision _resistance, accessed: 2019-11-26.

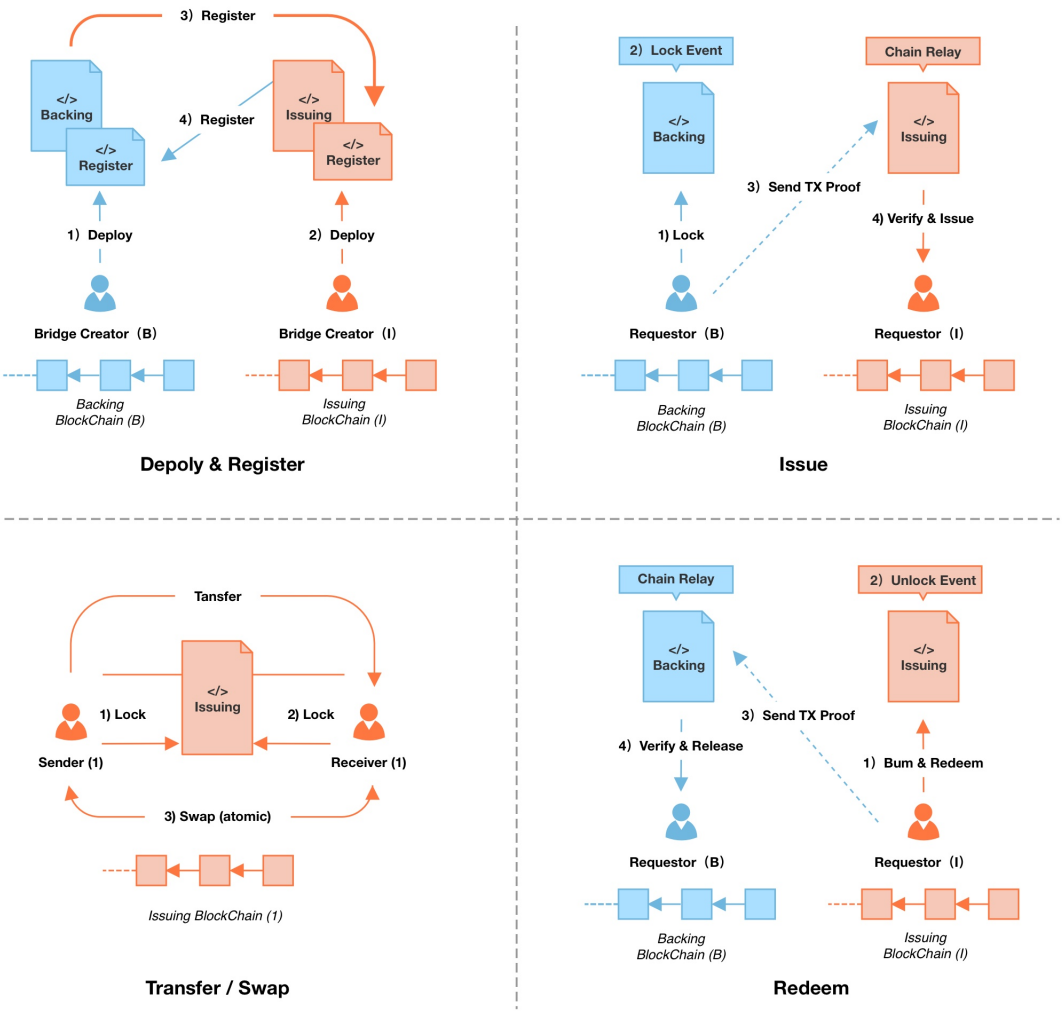



Fig. 3. High-Level Protocol Overview

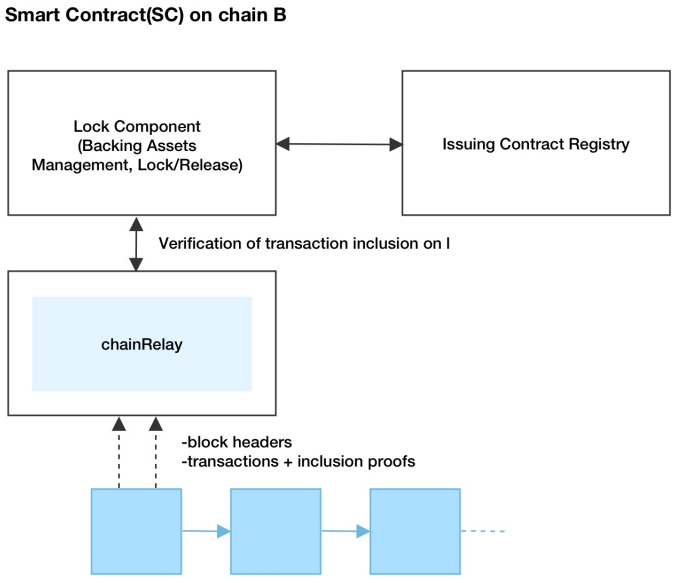



Fig. 4. Composition of cross-chain token bridge module on backing chain

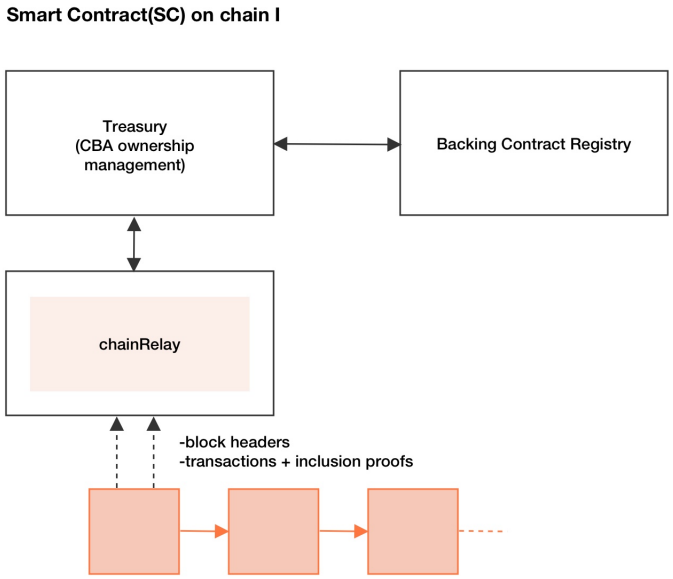


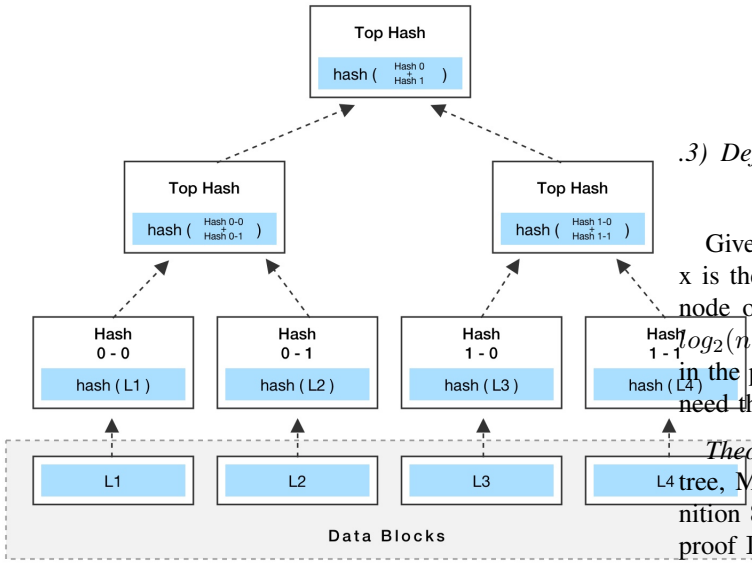

Fig. 5. Composition of cross-chain transfer bridge modules on the issuing chain

## VI. Appendix

*Merkle Tree*

Merkle Tree is a tree-like data structure in cryptography and computer science. Each leaf node uses the hash of the data block as a label, and nodes other than the leaf node use the cryptographic hash of its child node's label as the label. Hash trees can efficiently and securely verify the contents of large data structures.

Hash trees can be used to verify any type of data that is stored, processed, and transmitted from a computer to another. They can help ensure that data blocks received from other peers in the peer-to-peer network are not corrupted and altered, and even check if other peers are lying and sending fake data blocks.




*.1) Definition (Collision resistant hash function)*

Collision resistance is a property of cryptographic hash functions: a hash function H is collision resistant if it is hard to find two inputs that hash to the same output; that is, two inputs a and b such that H(a) = H(b), and a $\neq$b [9].

*.2) Definition (Merkle Tree)*

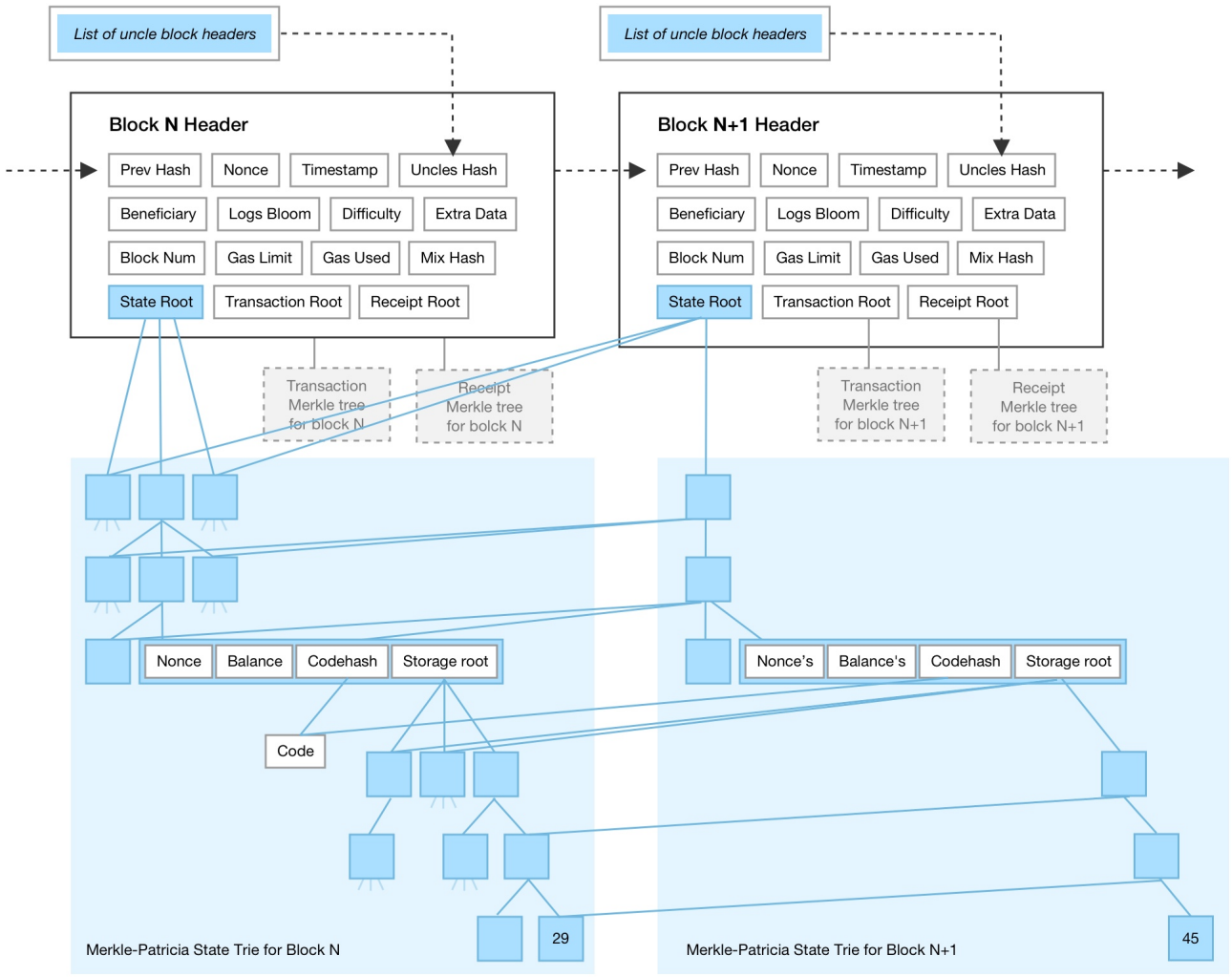
A Merkle tree is a balanced binary tree, where each leaf node stores some value, and each non-leaf node holds the value $H(LeftChild||RightChild)$, where H is a collision-resistant hash function. The Balanced binary tree here means a tree with n leaves that has a depth less than or equal to $log_2(n)$.

*.3) Definition (Merkle Proof)*

Given a Merkle tree, MT, with root r, a Merkle proof that x is the kth node in MT, $\Pi k \in MT$, are the siblings of each node on the path from x to r. Since MT has depth at most $log_2(n)$, the proof length is at most$log_2(n) + 1$ as each node in the path can be calculated from its two children so we only need the siblings and the 2 leaf nodes.

*Theorem 3:* (Soundness of Merkle-proofs). Given a Merkle tree, MT built using a collision-resistant hash function (Definition 8), a polynomial-time adversary cannot produce a valid proof $\Pi k \in MT$, for a k not in MT.

*Theorem 4:* (Completeness of Merkle proofs) Given a Merkle tree built using a collision-resistant hash function, MT, and a node $k \in MT$, a polynomial-time adversary cannot generate a proof $\Pi k \in MT$ that is not a true path in MT.




*Merkle Patricia Tree*

Merkle Patricia Tree (MPT), or Merkle Patricia Trie, is a variant structure of Merkle Tree common in Ethereum. It provides persistence and can map data (byte arrays) between binary files of arbitrary length. It is defined according to a variable data structure, mapping 256-bit binary fragments into binary data of arbitrary length. Binary data is usually stored in a database. The core of MPT and its protocol specification requirements is to provide an identifier for a given key value, which can be a 32-byte sequence or a null byte sequence. As for how to construct and implement the Trie structure efficiently, it is an implementation-level detail.

Compared with Merkle Tree, the most significant advantage of MPT is that it can quickly retrieve leaf nodes. This is very useful in the state tree. When the state needs to be continuously transferred, it can maximize the reuse of existing state storage data without a frequent copy. As shown in the figure below, when a leaf node changes, most of the stored data can remain unchanged. You only need to calculate the other branch nodes and root nodes above the leaf node.