Darwinia Relay: Sublinear Relay for Interoperable Blockchains

Preview Version

Abstract

To verify transactions, cryptocurrencies such as Bitcoin and Ethereum 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 and FlyClient 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 certain degree of 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 Relay, 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 Relay is able to achieve sub-linear performance by only submitting logarithmic numbers of datas including the block header and its

derived data. Darwinia Relay 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.

Introduction

Cross-chain Assumption

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

Interoperability

Interoperability refers to the characteristics of the interaction between two independent blockchain networks. The state change or event on one blockchain can deterministically and non-stop affect the other blockchain. Among them, the most critical is the ability to cross-chain verification. Cross-chain verification allows one chain to verify the existence of an event on another chain and then completes the cross-chain actions through the three main steps of generating an event receipt, cross-chain verification, and operation execution Interoperability.

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

Common Cross-chain Verification Solutions

Standard solutions fall into two main categories:

Trusted Nodes Solutions

Trust node solutions have a similar mechanism to the oracle. It introduces multiple nodes with multiple signatures and combines pledge and punishment mechanisms to ensure the accuracy and security of cross-chain information.

Onchain Light Client

Chain Relay is another name for the light client on the chain. The function is mainly to implement cross-chain verification and is an essential component of the cross-chain transfer bridge.

Light Client

The original Bitcoin design described a faster synchronization mechanism called Simplified Payment Verification(SPV), which allows for lightweight verification of the transaction on the blockchain through what is commonly referred to as an SPV client (also known as a light client).

Instead of downloading all blocks from the full node, the SPV client only downloads the block headers of each block, and the synchronization overhead of these block headers is much smaller than the full block (80 bytes compared to 1 MB per block in Bitcoin) . Block headers are linked together by hash and contain PoW proof of work. In this way, the SPV client can verify which chain has the most PoW proof of work. It should be noted that light clients do not know whether all transactions are valid or whether they follow all consensus rules. Light clients are based on the assumption that the chain with the most PoW workload follows the consensus rules of the network. It is also stated that all transactions in this chain are legal and valid, and most of the computing power will be supported by the same effective chain.

Assumption 2 (SPV assumption). The chain with the most PoW workload follows the network consensus and eventually is accepted by most miners.

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.

Using light clients to implement transaction cross-chain verification is a common type of cross-chain solution.

Sublinear Light Client

Someone may wonder if it is possible for a light client to verify any event on the blockchain by downloading or storing the amount of information that is only sub-linear (relative to the chain length). This performance improvement brings an vital security challenge: Since such a client cannot verify every PoW workload in the received blockchain, a malicious prover can trick it into accepting an invalid chain. A prover can use its limited computing power to pre-compute a sufficiently long chain.

As early as 2012, the concept of superblocks was first discussed in the Bitcoin forum and mailing list. Superblocks are more capable of solving PoW problems than current target problems. Since they appear randomly at a certain rate on the honest chain, if the miners act honestly, the number of superblocks in the chain can well indicate the total number of valid blocks. Kiayias et al. Introduced and formalized an interactive proof mechanism called Super Block-based Proof of Work (PoPoW). PoPoW allows the prover to convince the verifier with extremely high probability in log time and communication: a chain contains enough work. In a later work [32], Kiayaas et al. Provided an attack against the PoPoW protocol and proposed a non-interactive and still logarithmic alternative solution called Non-Interactive PoPoW (NIPoPoW).

A solution is described in FlyClient. It is reasonable to assume that an attacker can only fake some of the fake blocks (there is a maximum upper limit ratio). By introducing probability-based sampling, MMR, variable difficulty MMR checking, and Fiat-Shamir heuristic implementation Non-interactive proof and other technologies can achieve sub-linear light clients on the premise that certain conditions are met.

However, if you want to apply FlyClient to Cross-chain Relay, there are some additional challenges for light client implementation on the chain. Because the light client in the FlyClient solution does not run on the chain and the cost of connection is low, it can easily guarantee its assumptions, that is, at least one honest full node is connected to the light client. FlyClient is implemented on the chain. The block header is submitted to the chain by the Relayer through the transaction. Generally, transaction fees or fuel costs are required to send transactions, so in order to ensure that an honest node always submits a real block to the chain. In general, it is necessary to design and provide financial incentives while punishing perpetrators.

Another problem of FlyClient is that its implementation must rely on forking of the protocol to support the MMR data structure, which involves the modification of the blockchain protocol layer. It is very difficult for Cross-chain Relay, so it must find other methods which can ensure that the MMR is calculated and submitted off-chain, keeps correct when the protocol layer does not support MMRand achieves the purpose of chain commitment.

Our Contribution

1. 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.
2. 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.
3. Optimistic Verification Game can also solve the problem of malicious attacks under rational economic market 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

Overview of Darwinia Relay

We assume that there are two blockchain networks, called the target chain (T) and the verification chain (V), and they meet the cross-chain 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.

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), and 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 connection. It is achieved by submitting data to the chain through blockchain transactions through the prover.
The cost is also higher, so we need to design reasonable economic incentives to ensure that at least one honest prover is involved.

Chain Commitments

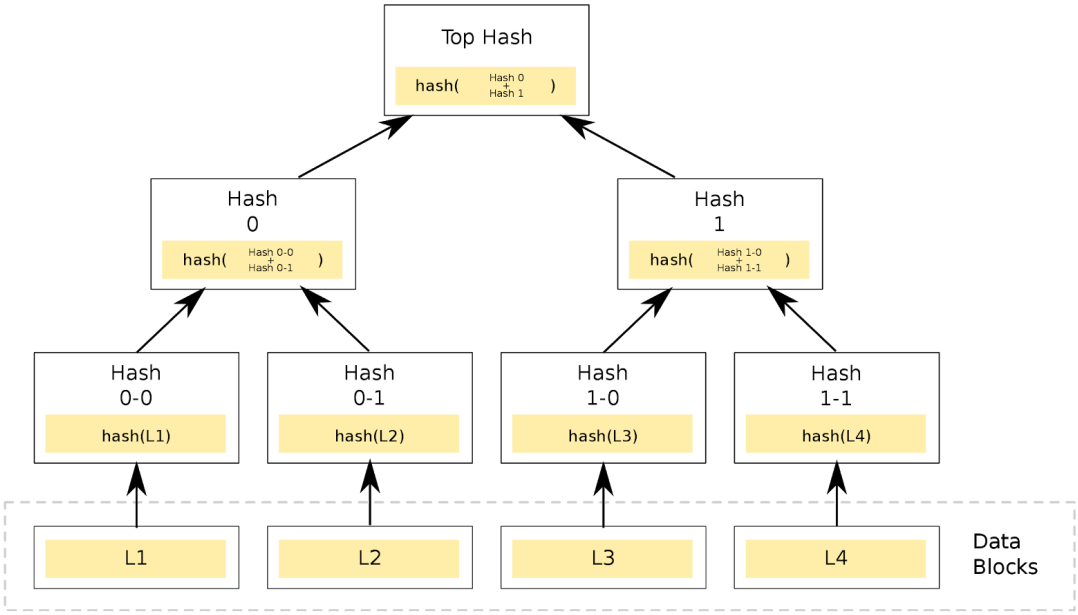
So far, we still allow malicious provers to deceive validators with an invalid chain. That is because the verifier only downloads a small number of block headers that cannot be verified consecutively, the malicious prover can choose to return the correct block from any position on the honest chain and submit it to the verifier. Because these block headers only contain the hash value of the previous block, but the verifier cannot verify the hash value of each block to achieve the validity verification of the complete history chain, which significantly reduces the probability of success of the protocol. One way to prevent this strategy is to have the prover "commit" the entire chain before the protocol begins, thus ensuring that it returns to the block at the expected location on the chain.

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, etc. These variants have added some other useful features while maintaining the characteristics of the Merkle Tree.

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 computer to computer. 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.



Definition (Collision resistant hash function).

A family of hash functions Hλ : {0,1}∗ → {0,1}λ is collision-resistant if for all efficient adversaries A the probability that x,y ←? A(1λ) and H(x) = H(y)∧x≠= y is negligible in λ.

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 ⌈log2 n⌉.

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 ⌈log2(n)⌉, the proof length is at most log2(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 (Soundness of Merkle- proofs).

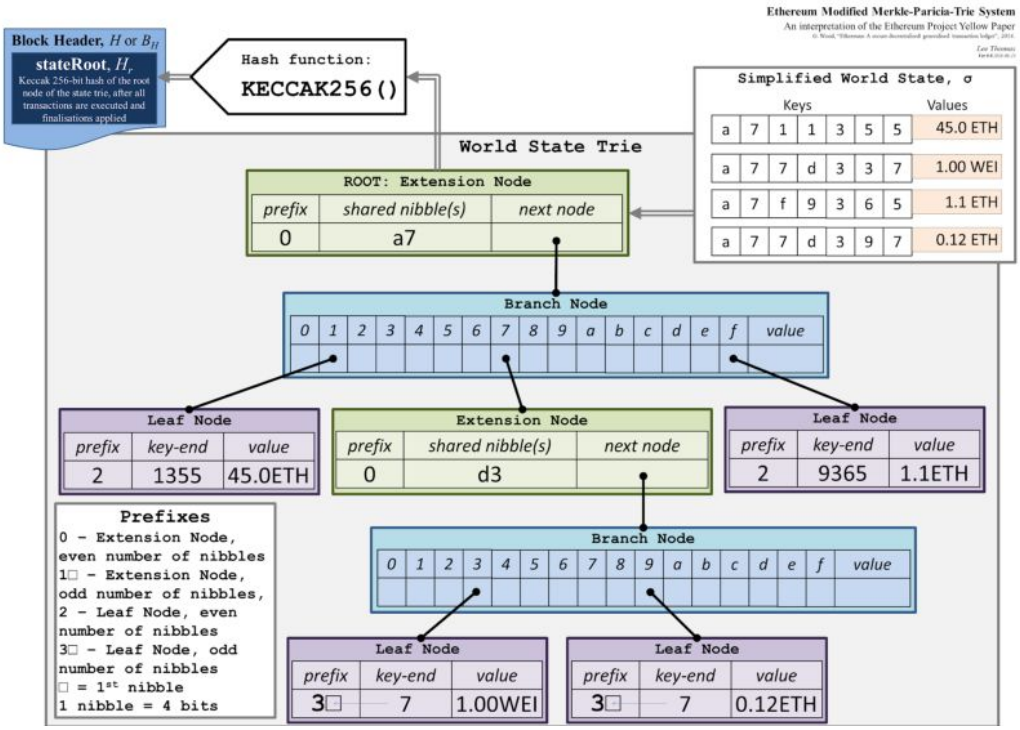
Given a Merkle tree, MT built using a collision-resistant hash function (Definition 8) , a polynomial-time adversary cannot produce a valid proof $\Pi_{k \in MT}$, for a k not in MT.

Theorem (Completeness of Merkle proofs).

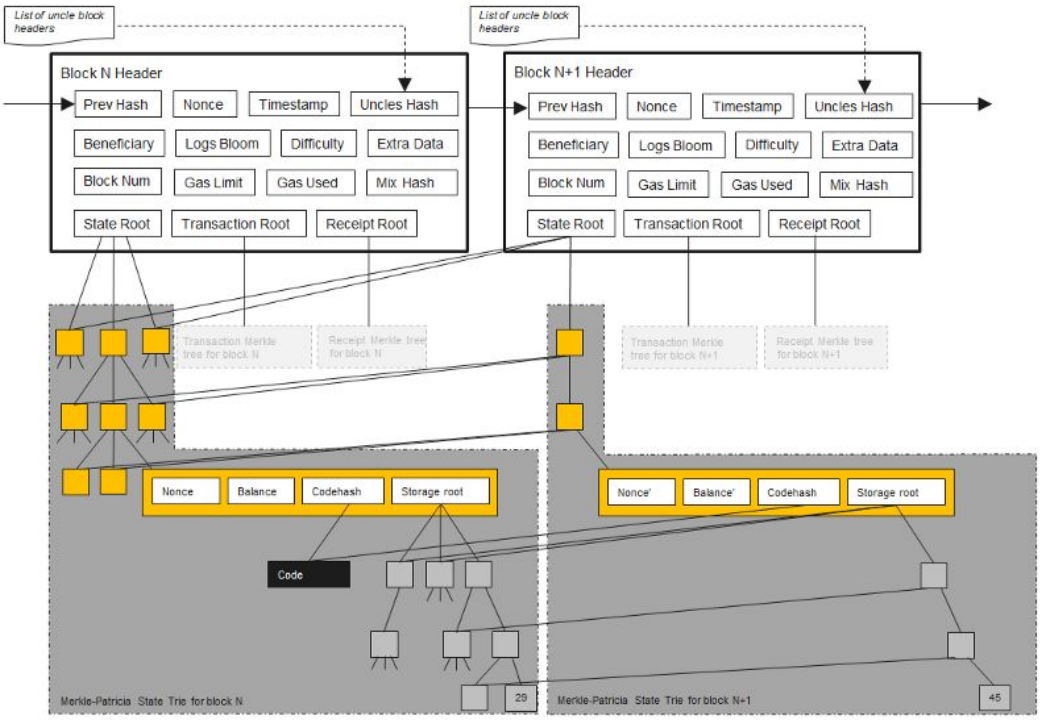
Given a Merkle tree built using a collision resistant hash function, MT, and a node $k \in MT$, a polynomial-time adversary cannot generate a proof $\Pi_{k \in MT}$ that is not a true path in MT.

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.

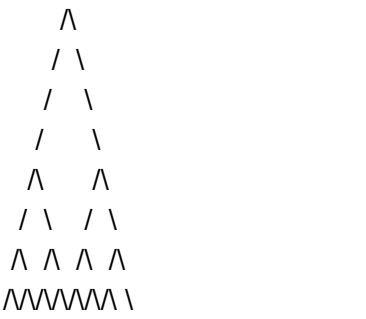


Merkle Mountain Range

To submit the entire blockchain, Darwinia Relay 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:

If a leaf node is added incrementally, the new MMR structure is:



Definition (MMR).

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

1. M is a binary hash tree.

2. M has depth ⌈log2 n⌉.

3. If n>1, let n=2i +j such that i=⌊log2(n−1)⌋:
    - 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 (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 (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 (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 (Block Commit).

If an adversary changes any block i in the chain in any way, then it's hash xi will also change, so any MMR Mk for k ≥ i with root rk′ that contains the new block x′i will have that rk′/= rk.

Because Darwinia Relay 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 Relay 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.

Prover - Verifier Model

The actors 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 proveres 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.

With this actors model, 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.

Verification Protocol

1. Verifier has access to r = root of some Merkle tree, MT.
2. 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.
3. Verifier uses the proof and x to build up the path to r′ using protocol 4, and checks that r′ = r.
4. If the checks pass, the Verifier accepts the proof, otherwise it rejects the proof.

Verification Game Sub Protocol

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

The verification game protocol is used in Darwinia 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 disproval 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 optimistic balance, that is, under the premise of economic rationality, most malicious people will not launch attacks.

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.

Model and Problem Definition

Darwinia Relay 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 target chain. Under the assumption of economic rationality, the system is optimized and optimized by using the Verification Game and the pledge incentive mechanism.

Because Darwinia Relay is based on the improvement of FlyClient and is applied to cross-chain verification scenarios, it may be necessary to introduce FlyClient, and the different challenges of Cross-chain Relay and Light Client (the target chain fork cannot be required to support MMR).

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 are 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.

Optimistic Verification Game

General Interactive Verification Game

The outer layer of Darwinia Relay 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.

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 Relay, or the challenger pays the resources consumed by the false alarm.

Block Header Verification

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:

1. 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.
2. 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.

3. Variable difficulty MMR verification faces the same problem as Challenge 2.

Before we introduce how to apply the verification game to solve the challenges in Darwinia Relay, we first define the specific problems and roles in the application according to the abstract concepts in verification.

Calculation problem to be solved: 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.

Solver: The off-chain is responsible for verifying, calculating, and submitting the correct block header and its MMR role on the chain. Anyone can play it, most likely it is a full node. 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 answer 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.

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

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 Relay, 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 solver 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.

Validation range on the external chain: For Darwinia Relay, the validation range on the external chain is when the block header submitted by the opponent is adjacent to the confirmed block height, the on-chain logic (network node or validator) will You can easily calculate whether the block header and its MMR value are valid according to the pre_hash and the MMR of the previous block header, and then verify the authenticity of this block header. Although the new block header is confirmed, the unverified block range will be Reduction, which can prove that the verification game process is convergent. Genesis is the earliest confirmed block, because it was written into the block-long before the blockchain network was launched, and was recognized by consensus.

End of verification game and settlement: As the verification game converges until the block of a specific height H is also confirmed, the game ends, and the correct behavior of the important role in the verification game participation process is rewarded, and the wrong behavior is punished.

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 Relay 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.

DDOS Attack

If we assume that the attacker is non-economically rational, and willing to pay a certain cost to attack the system, then DDOS 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 Relay, namely its MMR, cannot be confirmed for a long time. However, it is worth noting that its attack cost is linearly related to its DDOS 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.

Finality

Using Darwinia Relay for cross-chain verification, its effectiveness depends on the validity of the block existence proof and the transaction / receipt existence proof (Merkle proof). Among them, the validity of the block 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 Relay 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.

On-demand Verification Design

According to the commonly used collaboration process in Chain Relay, Darwinia Relay 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:

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

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 Relay, 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 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 Relayer, 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 Relay 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

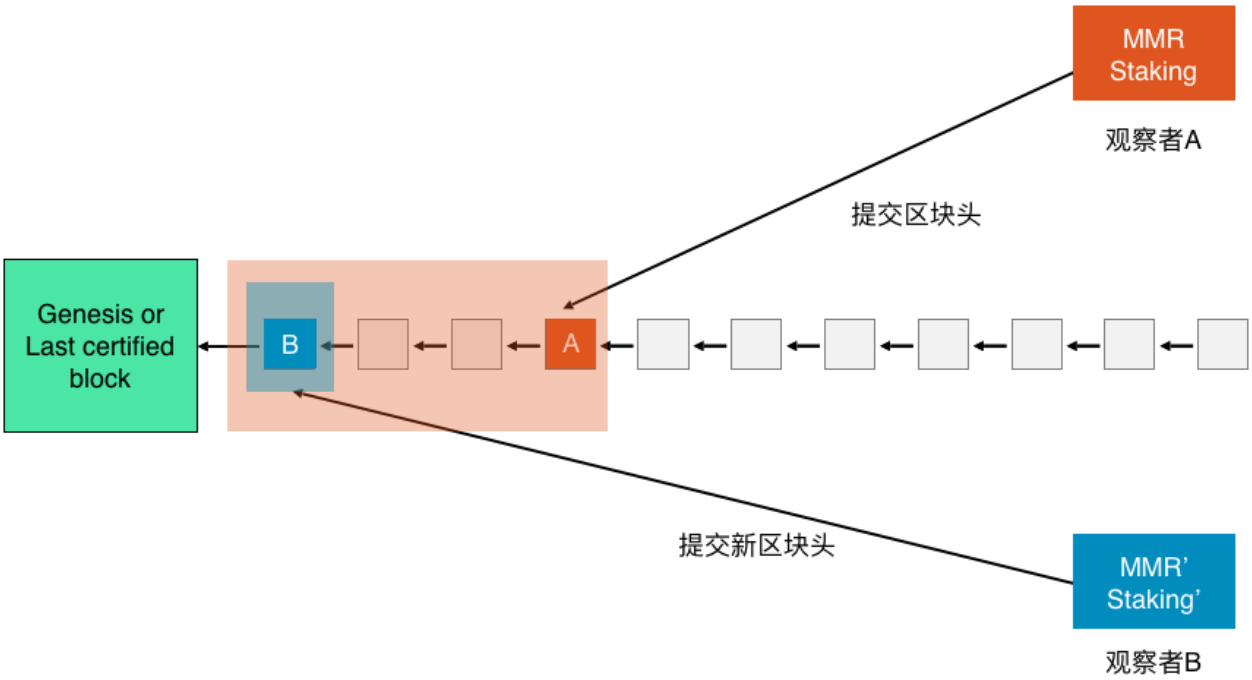
Because Darwinia Relay has the optimization of on-demand verification, the cost of its verification service is also simplified, so this model can be simplified to:

Verification service price = block header submission fuel fee + premium

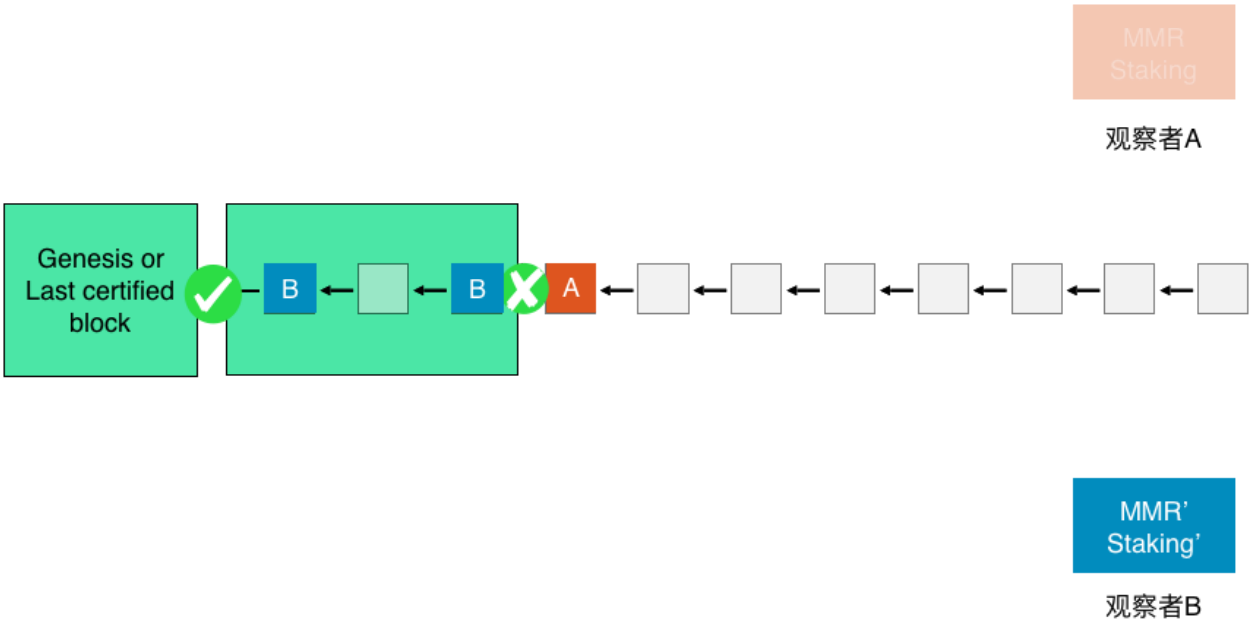
The optimization of on-demand verification has two benefits. The first benefit is that before using the verification service, you can determine the price of the verification service and decide whether to continue using the verification service. If you do not continue, you do not need to submit the block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

Verification Game Economic Model

In Darwinia Relay, 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.

Darwinia Relay Applications

Token Bridge

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.

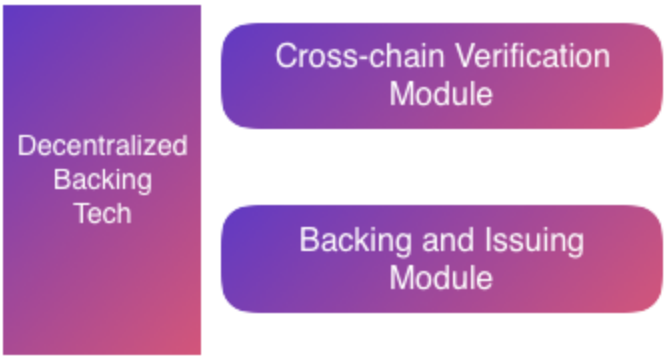
Cryptocurrency Backed Asset Model(CBA Model)

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 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 is 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.



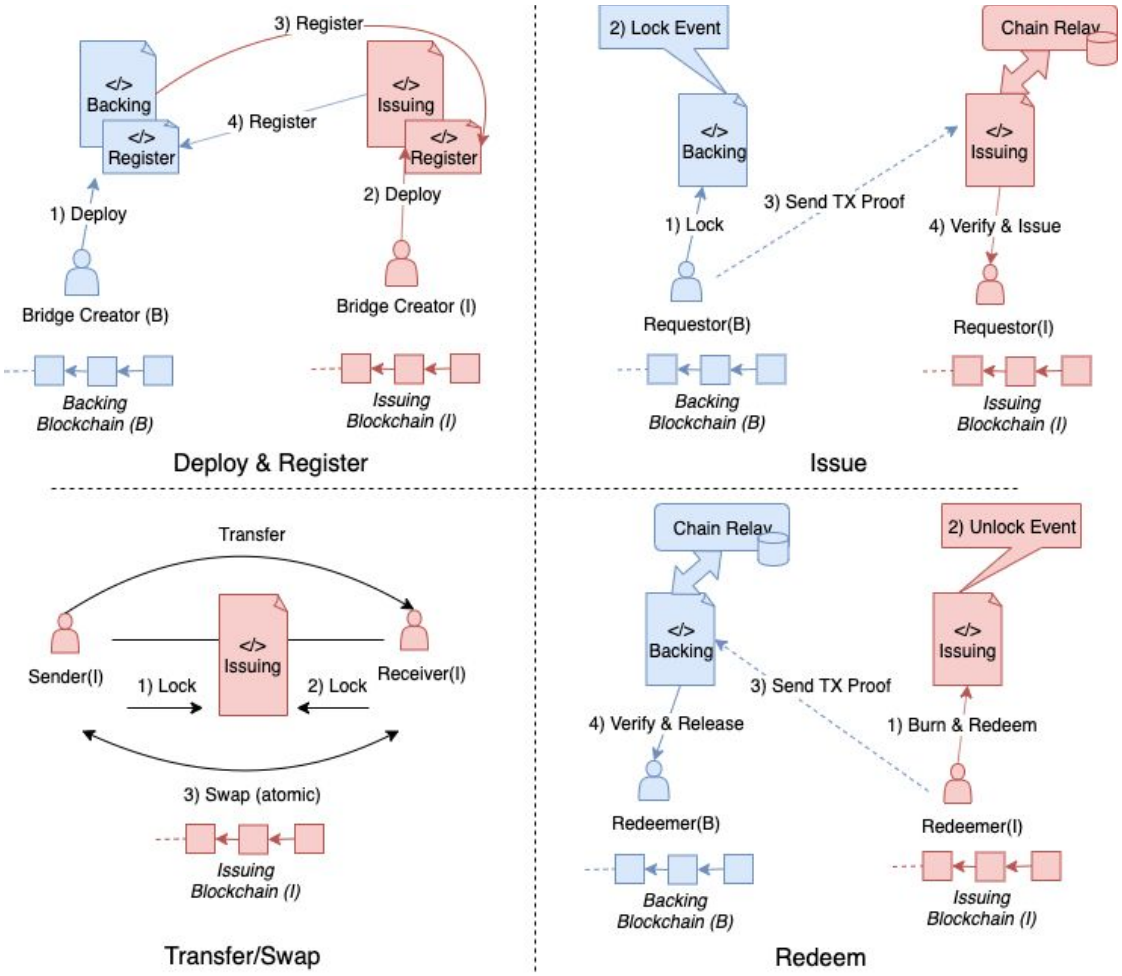
Common decentralized backing techniques include:

1. There are trust nodes or custodian mechanisms, where the trust node or custodian use multi-signature / threshold signature and other multi-centralized technologies to avoid single points of failure, such as Parity Bridge, ChainX, etc.
2. 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.
3. Fully use the Chain Relay for chain verification, such as BTCRelay, WaterLoo, Darwinia Relay, etc. The prerequisite of using this solution is that the issuing chain needs to support the implementation of through 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

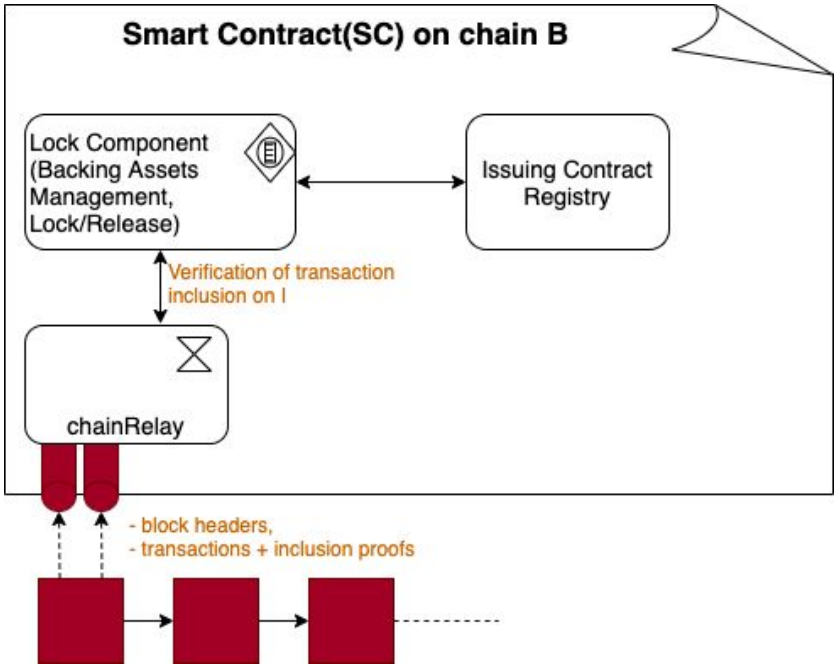
Darwinia's cross-chain token bridge solution is a two-way cross-chain token bridge based on chain relay. The chain relay we are going to develop adopts the design of the Darwinia Sublinear Relay, which has better performance and economic feasibility.

High Level Protocol Overview

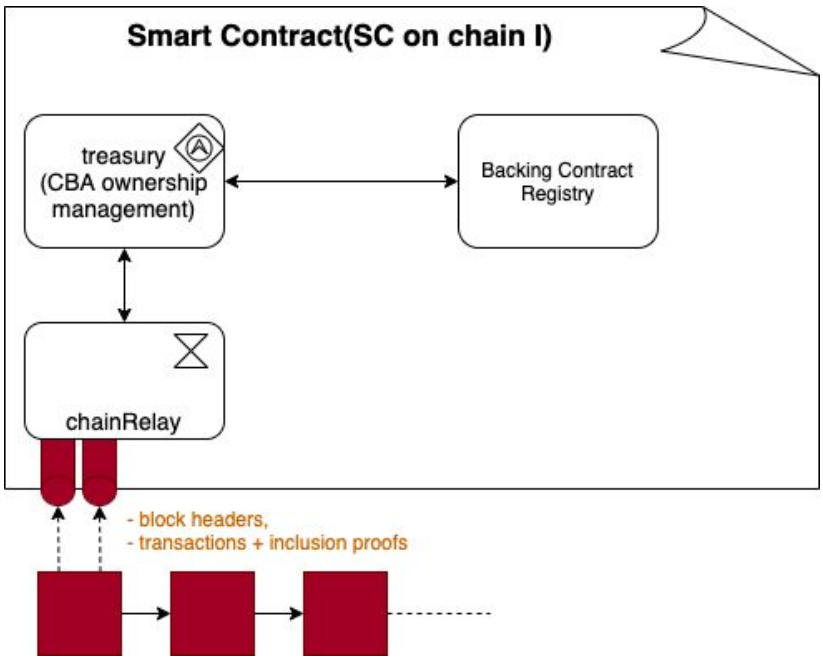


High-level overview of the Register, Issue, Swap and Redeem Protocol

Composition of cross-chain token bridge module on backing chain



Composition of cross-chain transfer bridge modules on the issuing chain



References

1. Bitcoin Whitepaper: https://bitcoin.org/bitcoin.pdf
2. XClaim: https://eprint.iacr.org/2018/643.pdf
3. Flyclient: https://eprint.iacr.org/2019/226
4. Truebit: https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf
5. Ethereum Yellow Paper: https://ethereum.github.io/yellowpaper/paper.pdf
6. MMR: https://github.com/opentimestamps/opentimestamps-server/blob/master/doc/merkle-mountain-range.md
7. https://www.youtube.com/watch?v=njGSFAOz7F8

Appendix

Simplified Token Bridge between Ethereum and Substrate

Demo: https://mp.weixin.qq.com/s/ZsS8Z8FgLifet2t-8X84NQ
Code: https://github.com/darwinia-network/darwinia/tree/icefrog/srml/eth-relay