MAP Protocol White Paper

MAP Protocol is a Bitcoin layer-2 and peer-to-peer omnichain infrastructure built upon light clients and ZK technology, focusing on cross-chain interoperability. During the cross-chain process, it does not rely on any privileged third parties, but is purely peer-to-peer and only puts trust in code. It provides an omnichain smart contract development platform for dApps and interoperability for the Bitcoin ecosystem.

** Note: This white paper was first published in 2019 and has been updated as it has developed. Like other community-driven open-source software projects, the MAP Protocol white paper has continued to evolve since its release.*

The white paper explains the basic principles and technical framework of MAP Protocol, which is necessary for a systematic study of MAP technology and understanding its vision. If you want to know the latest developments and updates of MAP Protocol, you can read other related content on [MAP Protocol website](#).

A Bitcoin Layer-2 Focused on Cross-chain

Bitcoin Development History

In 2008, Satoshi Nakamoto published the [Bitcoin Whitepaper](#), outlining the blueprint for a decentralized ledger known as blockchain within just nine pages, and introduced the concept of Bitcoin as a currency.

Since 2008, Bitcoin – a token not backed by any physical object or intrinsic value, not controlled by any centralized institution – has not only survived but also flourished, and has even become a reserve currency for some countries as a means of storing value. Influenced by Bitcoin, the cryptocurrency field has also created waves of innovation, attracting and inspiring individuals like Vitalik Buterin to attempt to develop more programmable protocols.

However, the Bitcoin network mechanism did not remain unchanged. Over the past decade, apart from Bitcoin mining, its network mechanism has also gone through some changes:

- In 2012, the Bitcoin network introduced Pay to Script Hash (P2SH) through BIP 16, simplifying multi-signature transactions. Before the advent of P2SH, multi-signature transactions were cumbersome and risky, requiring the entire redeem script (defining spending conditions) to be disclosed in advance. With P2SH, users send funds to a standardized Bitcoin address representing the hash of the redeem script, thus hiding its complexity. The full script is only disclosed and its conditions are met when spending the tokens, aiming to simplify transactions, enhance user-friendliness, and improve scalability.
- Another very important Bitcoin Improvement Proposal (BIP) — Segregated Witness (SegWit), also known as SegWit, became effective in 2017. It addressed the scalability of transactions and effectively increased the block size limit from the original 1MB to 4MB.
- SegWit paved the way for a proposal called Taproot in 2021. Taproot makes transactions more efficient and private and also allows users to engage in more complex transaction types.

The Future of the Bitcoin Ecosystem

The Taproot upgrade in 2021 allows for faster verification of multi-signature transactions and opens the door for inscribing text, images, SVG, and HTML on the smallest denomination of Bitcoin (called "satoshis"). In May 2023, with the proposal and explosion of BRC-20, an experimental, alternative token standard for Bitcoin, people are paying more attention to the future of Bitcoin ecosystem interoperability.

MAP Protocol supports the cross-chain transfer of BRC-20 assets from the Bitcoin network to the MAP Protocol network in a peer-to-peer manner. This enables other cryptocurrencies on different blockchains to be traded with BRC-20 assets more conveniently and cost-effectively, enhancing the transferability of BRC-20 assets. This layer of interoperability will help expand the use cases of Bitcoin and integrate the Bitcoin ecosystem into a broader crypto-based financial ecosystem, thus bringing new contributions to the Bitcoin community.

How assets and users on other public chains can move to the Bitcoin network through peer-to-peer cross-chain interoperability is also a technical issue that the industry has been exploring. As a Bitcoin-level peer-to-peer cross-chain infrastructure, MAP Protocol is committed to achieving this goal as a Bitcoin Layer 2, enriching the interoperability of the Bitcoin ecosystem.

A Peer-to-Peer Cross-Chain Smart Contracts and Web3 DApp Development Platform

Satoshi Nakamoto's development of Bitcoin in 2019 was a revolutionary innovation for assets and currencies. The reason is that Bitcoin is a peer-to-peer, trustless third-party electronic currency (asset) — initiated by one party, and accepted by another, without the need for a privileged third-party institution.

Prior to Bitcoin, the digital signature in cryptographic technology solved the problem of digital currency, but it still required a trusted third party to prevent double-spending. With the major innovations in peer-to-peer technology, Bitcoin has also achieved great success, and people are paying more attention to the blockchain industry and the application of other blockchain technologies.

By 2020, the rise of decentralized finance led to the birth of many blockchain mainnets, and users' cross-chain needs began to explode, leading to a large number of cross-chain solutions on the market. Although many cross-chain solutions have emerged, most of them basically rely on off-chain third-party roles to prevent double-spending. That is to say, whether a cross-chain event is valid or not depends not on whether the event occurs on the origin chain but on a group of off-chain consensus nodes or roles with trust assumptions. Although they have incorporated various security assumptions, they are completely contrary to the decentralized spirit of peer-to-peer and trustless third-party roles defined by Satoshi Nakamoto.

However, the majority of those cross-chain solutions essentially rely on off-chain third-party intermediaries to prevent double-spending, i.e., the validity of a cross-chain event is determined not by whether or not it has occurred on the source chain, but rather by a set of validators or roles with off-chain consensus and trust assumptions. Despite their inclusion of various security assumptions, they are fundamentally at odds with Satoshi Nakamoto's definition of peer-to-peer and no third-party decentralization ethos.

It is worth noting that although Plokadot and Cosmos have achieved peer-to-peer cross-chain interoperability, their cross-chain interoperability ability is only confined to chains within their ecosystem. This is not enough to meet users' needs. The industry is in urgent need of a cross-chain smart contract development platform that covers all types of chains and is completely peer-to-peer.

MAP Protocol adopts the Simplified Payment Verification (SPV) light client technology defined by Satoshi Nakamoto, innovatively turning the light client verification mechanism into smart contracts, and realizes peer-to-peer cross-chain verification by trusting the code instead of any third parties. Whether a transaction is valid or not

depends only on the occurrence of the source chain, rather than any form of off-chain third-party role or organization; and through zero-knowledge proof technology, it further enhances cross-chain verification efficiency. At the same time, the MAP Relay Chain can include all kinds of Layer1 signature, hashing, and mining algorithms through the built-in pre-compiled contracts, achieving cross-chain interoperability with heterogeneous chains, and ushering in the era of peer-to-peer omnichain dApp development.

Background

Up till now, there are three stages of a decentralized financial network.

1. The birth of the Bitcoin peer-to-peer e-cash payment system: created the crypto industry and stimulated the development of centralized exchanges (CEXs) since 2009.

2. The emergence of the programmable smart contract of Ethereum: boosted the development of public chains and decentralized applications (dApps) since 2015.

3. The emergence of omnichain network infrastructure: making omnichain dApps possible and significantly boosts development.

Before MAP Protocol, the three primary cross-chain solutions are:

1. CEXs: Characterized by centralized institutions that ensure the safety of user asset exchanges through methods like KYC, cold wallets, and regulatory compliance.

2. Off-Chain Consensus Verification Mechanism: This includes MPC, oracle-based verification, cross-chain verification done by validators, Optimistic Rollups, and others. Among those solutions. MPC cross-chain bridges cannot eliminate the existence of privileged roles and are susceptible to hacking or inside jobs. Oracle may verify based on block headers but could falsify those headers, leading to false verification. Cross-chain verification done by validators, i.e., verifications done off-chain, entails collusion risk. Optimistic Rollups, while secure, have long verification waiting times.

3. Polkadot and Cosmos: Both are Bitcoin-level peer-to-peer cross-chain solutions but are limited to internal ecosystem chain cross-chain communications; they can't interoperate peer-to-peer with EVM chains and other heterogeneous chains.

MAP Protocol Peer-to-Peer Cross-Chain Solution

To achieve peer-to-peer cross-chain verification, two major obstacles must be resolved:

1. How can data communicate when each chain has different signatures, hashing algorithms, and consensuses, being heterogeneous with one another?
2. How to ensure that the cross-chain request sent by interchain messaging components truly comes from the source chain?

Cross-Chain Solutions Relying on Third-Party Trust

Cross-chain solutions that rely on third-party trust typically adopt a group of third-party off-chain witnesses as validators, who either stake assets or use the authority of a reputable traditional brand to decide if cross-chain requests are valid, thus preventing double-spending.

In some solutions the role for request and validation is the same, while others are different. Trust in these third-party consensus solutions is based on code but based on the trustworthiness of those off-chain third parties, which contradicts Satoshi's vision of peer-to-peer, trustless decentralization. Polkadot and Cosmos, although being peer-to-peer, can't solve the interoperability issues among heterogeneous and EVM chains.

A Peer-to-Peer, Trustless Cross-chain Network Able to Cover All Types of Chains

1. Set up a relay chain that algorithmically harmonizes any data-heterogeneous blockchain mainnet.

   MAP Protocol has three layers: peer-to-peer layer, consensus layer, and smart contract development layer. Different chains have different signature algorithms and hash algorithms. For example, Ethereum uses the **secp256k1** signature algorithm and **keccak-256** hash algorithm. NEAR uses the **ed25519** signature algorithm and **SHA-256** hash algorithm. In simple terms, blockchain transactions are initiated by one party signing the transaction, and another party accepts the transaction by cracking the hash, with no third-party trust involved. To build a peer-to-peer network that covers different chains, a relay chain needs to exist that makes these different algorithms homogenous.

   MAP Protocol's solution is built on the MAP Relay Chain's Virtual Machine development layer, with pre-compiled contracts embedding various chain

signature algorithms, hash algorithms, mining, and Merkle Tree proofs. MAP Relay Chain aims to make mainstream algorithms homogenous, and the algorithms already integrated into this pre-compiled contract are as follows.

- SHA3
- SHA-256
- ed25519
- secp256k1
- keccak-256
- sr25519
- blake2b
- Merkle Tree Proof

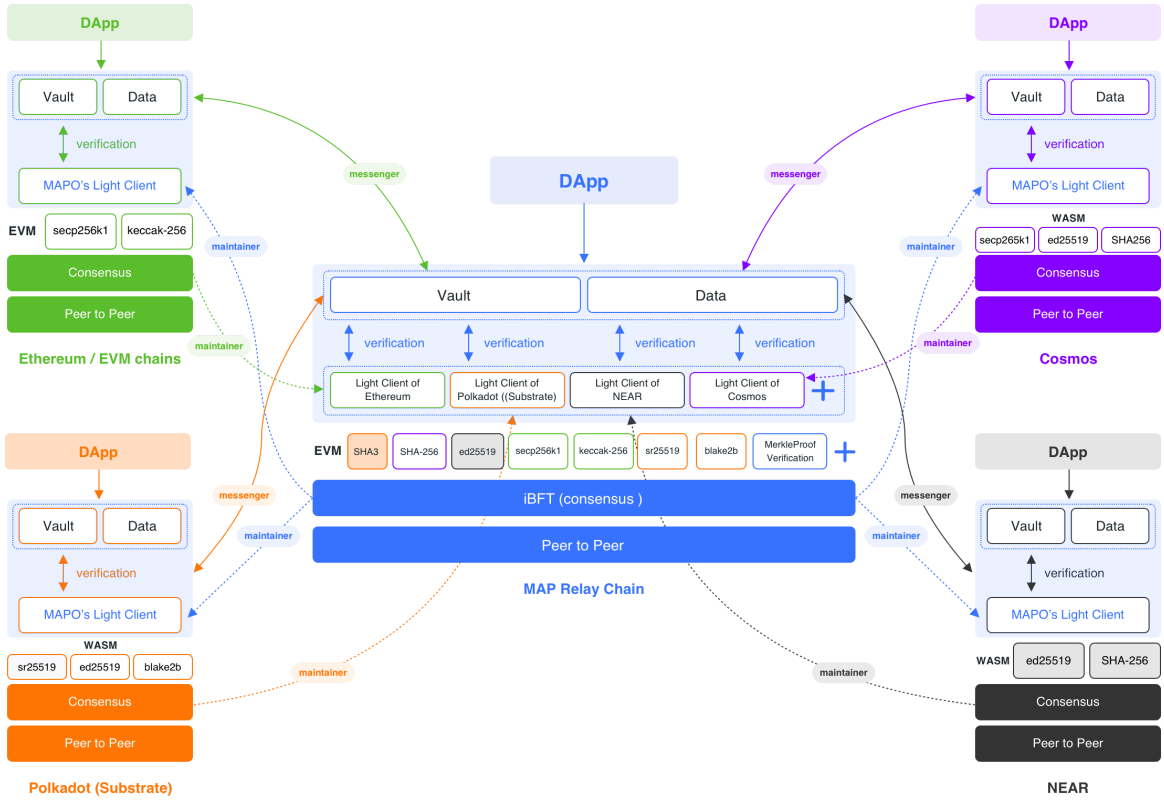
Although this approach requires a lot of work, it allows the target chain with different algorithms to read and verify cross-chain transaction data from the origin chain with different algorithms. It should be noted that EVM and non-EVM only refer to differences in the Virtual Machine development layer, while heterogenous chains in the general sense differ in their signature algorithms, hash algorithms, and mining algorithms. After achieving data homogeneity through the Relay Chain, transaction data can be read on different chains. The next problem to solve is the issue of double-spending in peer-to-peer transactions without a third party.

2. The Light Client for Simple Payment Verification (SPV) from the origin chain is deployed to the target chain as a smart contract. For example, if a peer-to-peer cross-chain transaction is initiated from Ethereum to NEAR, then Ethereum's Light Client must first be deployed as a smart contract to the MAP Relay Chain, followed by deploying MAP Relay Chain's Light Client to NEAR as a smart contract.

3. Cross-chain message forwarding is managed by a group of off-chain Messengers who forward cross-chain requests from the origin chain to the target chain. The target chain's origin chain Light Client smart contract then validates the requests.

4. Updating the Light Client is done by a group of off-chain Maintainers responsible for updating the latest block headers and Merkle Tree proofs from the origin chain to the smart contract deployed on the target chain. Messengers and Maintainers only serve as message carriers and have no authority for cross-chain validation. If they initiate attacks toward the self-verifying Light Client smart contract, such attacks would be deemed invalid.

5. The Light Client smart contract deployed on the target chain carries out peer-to-peer verification of the validity of cross-chain requests forwarded by the off-chain roles and broadcasts them for transaction execution.



Light Clients

Satoshi Nakamoto first defined Simple Payment Verification (SPV) in the Bitcoin white paper, "It is possible to verify payments without running a full network node. Users only need to possess a copy of the block headers of the longest chain with the most proof of work. They can verify they have the longest chain by querying other network nodes and obtain a Merkle branch linking the transaction to the timestamped block."

Although one cannot independently verify a transaction, if it is already linked to a position in the chain, it means a network node has accepted the transaction, and subsequent blocks appended to the chain further confirm its acceptance. Thus, it is evident that to verify the authenticity of a transaction on the origin chain, one only needs to see if it has entered into the block headers of the longest chain and can be found in detail on the Merkle tree. SPV defined by Nakamoto later evolved into the concept of light client technology in academia.

The Trustless and Self-verifying Nature of Light Clients

Although Oracle solutions also verify cross-chain validity through block headers, oracles are not smart-contract-level code trust. Instead, they are organized by a group of off-chain third-party roles who have the power to tamper with the submitted block header information. MAP Protocol's solution deploys the light client of the origin chain in the form of a smart contract to the target chain. The cross-chain verification of whether a transaction is genuine and valid is entirely through the light client smart contract of the origin chain on the target chain. The light client smart contract follows the longest chain principle and is maintained by a group of cross-chain message role maintainers responsible for synchronizing the latest light client status of the origin chain to the corresponding smart contract on the target chain. This smart contract stores the block headers of the origin chain and Merkle tree-proof data sufficient to satisfy the longest chain verification of a transaction.

Although maintainers are off-chain programs, once the light client smart contract is correctly and honestly initialized, maintainers have no opportunity to tamper with the subsequently appended light client status, that is, block headers and Merkle trees. This is because a blockchain is a data structure that links blocks containing transaction information in reverse order. Each block header will link the hash value of the previous block header. False block headers cannot match the hash value of the real block header already stored in the smart contract, and the smart contract is code trust, so no one can modify it again. Therefore, any malicious attacks by maintainers on the light client smart contract would be invalid. On the other hand, any attempt to change any part of the Merkle tree will ultimately lead to inconsistency somewhere on the chain, making the maintainers' attack on the light client smart contract invalid.

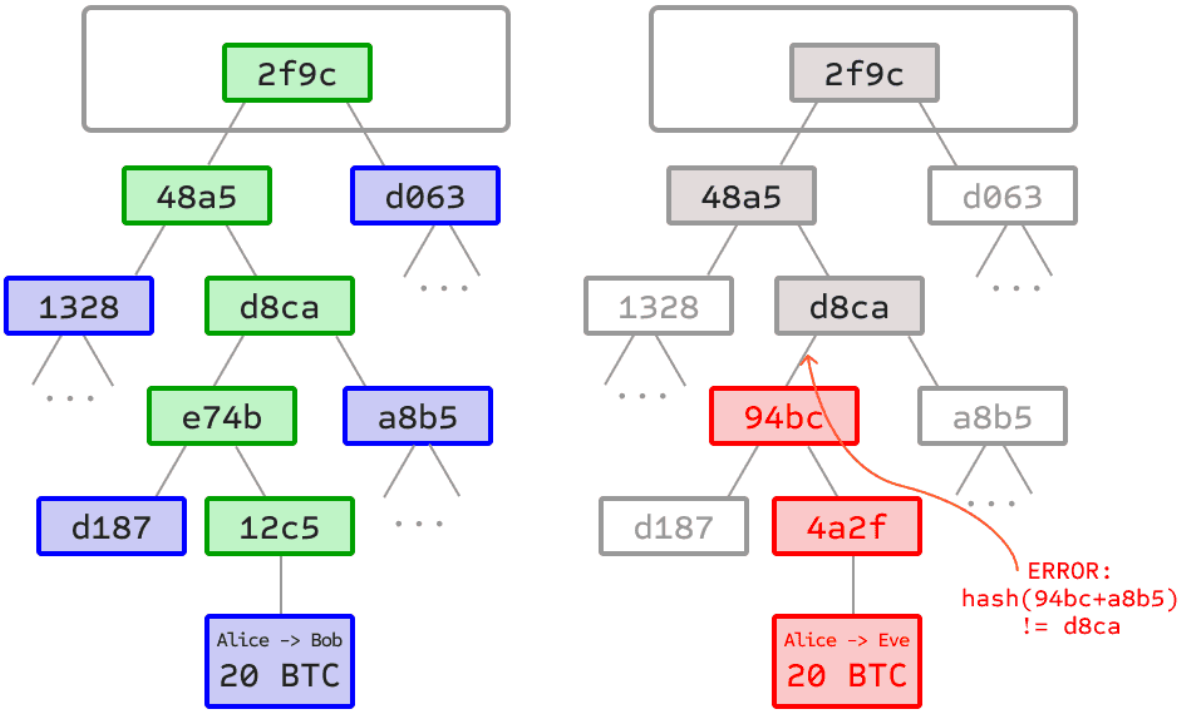
Block headers and Merkle trees

In a blockchain network, blocks are stored in a multi-level data structure. The hash of a block is just the hash of the block header, which is a fixed byte size of data, containing a timestamp, nonce, hash of the previous block, and hash of the root of the Merkle tree. The Merkle tree is a data structure that stores all transactions of that block. The Merkle tree is a binary tree, consisting of a set of leaf nodes, a set of intermediate nodes, and a root node. At the bottom are numerous leaf nodes containing basic data, each intermediate node is the hash of its two child nodes and the top root node is also the hash of its two child nodes.

The purpose of the Merkle tree is to allow block data to be transmitted in fragments: nodes can download block headers from one source, download a small part of the relevant tree from another source, and still be able to confirm that all data is correct.

This is because hashes propagate upwards: if a malicious user tries to replace a fake transaction at the bottom of the tree, this change will cause a change in the nodes in the upper layer of the tree, and further change in the upper nodes, eventually causing the root node to change and the block hash to change, so the protocol will record it as a completely different block (almost certainly with an invalid proof of work).

The Merkle tree protocol is the basis of the long-term continuity of the blockchain. A full node in a blockchain network - a node that stores and processes all block data, the data scale is too large, leading to a large workload to verify a transaction. Simplified Payment Verification (SPV) allows another type of node to exist, such nodes are called "light nodes" (light clients), they download block headers, use block headers to confirm proof of work, and then only download the Merkle tree branches related to their transactions. This allows light nodes to securely determine the status of any transaction and the current balance of an account by downloading only a small part of the entire blockchain.



* ***Left:*** Providing only a small number of nodes on the Merkle tree is sufficient to give legitimate proof of the branch.

* ***Right:*** Any attempt to change any part of the Merkle tree will ultimately lead to inconsistency somewhere in the chain.

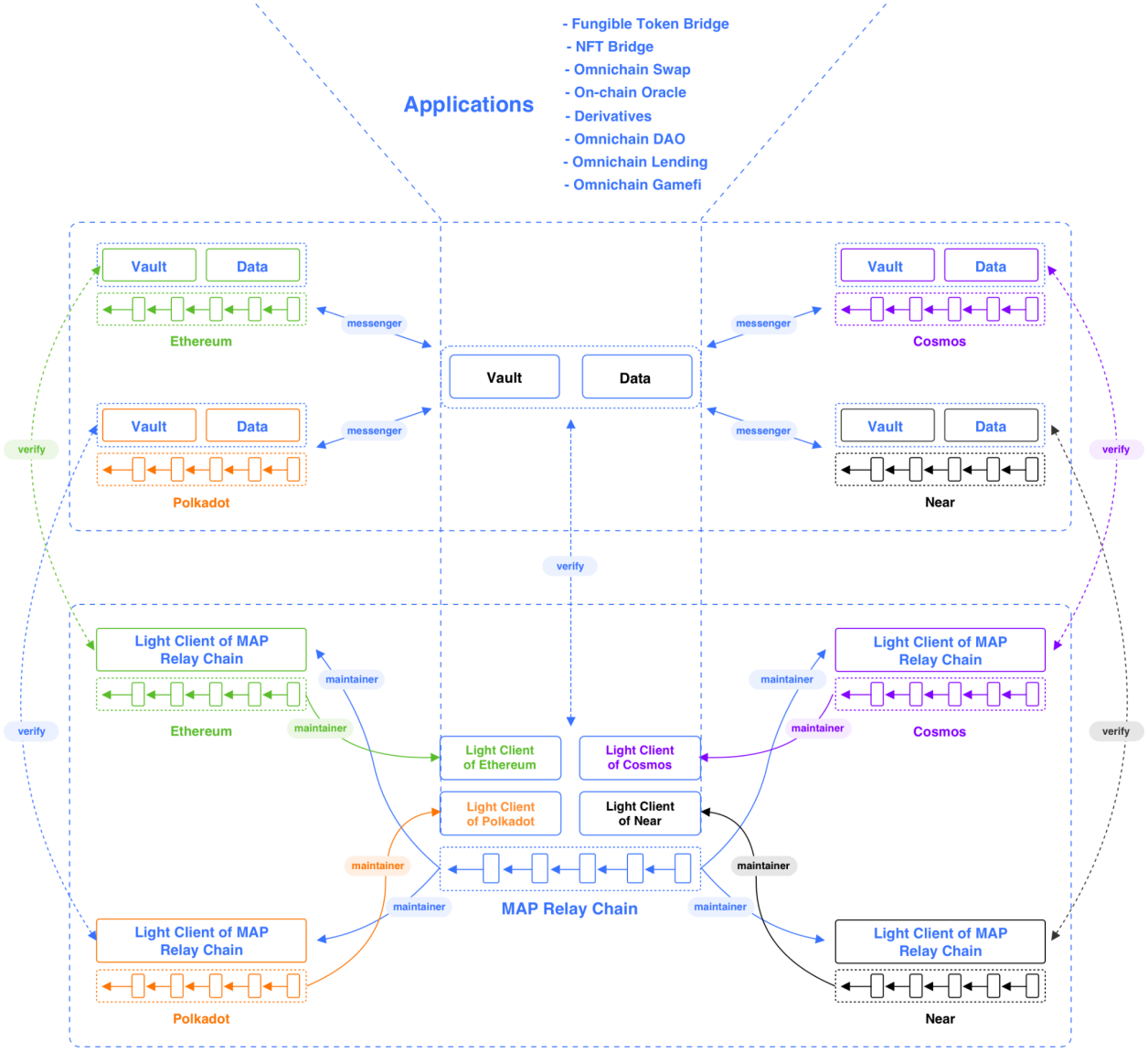
Zero-Knowledge Proof

Zero-knowledge proofs refer to a method of verifying the validity of a statement without disclosing the statement itself. The prover is the party attempting to prove the statement, while the verifier is responsible for verifying it. Through the efforts of the Ethereum community, the effectiveness of zero-knowledge-proof technology has been fully recognized. In cross-chain verification, ZK technology can shorten the cross-chain verification time and reduce the gas cost of light client smart contract verification, without affecting the verification of peer-to-peer cross-chain validity.

Peer-to-Peer Cross-Chain Verification Solution Combining Light Client with ZK

In the MAP Protocol's technical mechanism, ZK proof is responsible for verifying the BLS aggregate signature hash value of the origin chain, followed by the light client smart contract verifying the Merkle proof and re-verifying the ZK proof. By combining the MAP Relay Chain compatible with various blockchain algorithms, MAP Protocol has achieved code-level trust without privileged third parties as defined by Satoshi Nakamoto; furthermore, it can cover different types of blockchains, thus truly realizing a peer-to-peer cross-chain blockchain network.

MAP Protocol Three-Layer Architecture



To ensure the omnichain infrastructure's overall flexibility and robustness, MAP Protocol's technical architecture is divided into three layers: the protocol layer, the MOS layer, and the application layer.

The middle layer is MOS (MAP Omnichain Service), which provides a series of components to help dApps quickly deploy cross-chain smart contract applications. This includes the cross-chain transaction forwarding role *Messenger*, cross-chain lock-up smart contracts, cross-chain message components, etc. The third layer is the application layer, which is the MAP Protocol whole chain smart contract ecological application.

MAP Protocol Protocol Layer

MAP Protocol Protocol layer is the bottom layer of MAP Protocol, responsible for cross-chain traction verification and ensuring peer-to-peer cross-chain. It is the core of MAP Protocol, including MAP Relay Chain, Miantianer, and ZK-optimized light clients.

MAP Relay Chain

The MAP Relay Chain is a blockchain, based on the Proof of Stake (POS) mechanism. Compared to Proof of Work (POW) systems like Bitcoin, it is more environmentally friendly. Under the POS mechanism, users can conduct cheaper, faster transactions with irreversible results once completed. The MAP Relay Chain implements the Istanbul Byzantine Fault Tolerance (IBFT) consensus algorithm, which allows a well-defined group of validator nodes to reach an agreement by broadcasting signed messages among them in a series of steps, even if up to one-third of the nodes are offline, faulty, or malicious. When a quorum of validators reaches a consensus, that decision is final. One of the unique features of the MAP Relay Chain is the precompiled contracts that incorporate the core algorithms of almost all chains.

Maintainer

Maintainers are a group of off-chain roles responsible for updating the latest block headers and Merkle tree proofs from the origin chain consensus layer to the light client smart contract deployed on the target chain. Although maintainers are off-chain programs, once the light client smart contract is correctly and honestly initialized, maintainers have no opportunity to tamper with the subsequent appended light client state, i.e., the block headers and Merkle trees. Therefore, any malicious attacks by maintainers on the light client smart contract are ineffective.

ZK-Improved Light Clients

Using light clients to perform cross-chain verification is peer-to-peer and decentralized, but the verification cost is higher compared to other cross-chain solutions that rely on third parties and the efficiency is also relatively low. Therefore, MAP Protocol innovatively introduces ZK technology, optimizing the peer-to-peer cross-chain verification with light clients using ZK to optimize the verification rate.

Light Client Cross-Chain Peer-to-Peer Verification Mechanism

PoS mechanism L1 and PoW mechanism L1 have different categories of block headers. Under the PoS mechanism, the core of block header verification is to verify the Validator's signature information. The specific implementation logic is as follows:

When the Origin Chain is a PoS Mechanism Blockchain

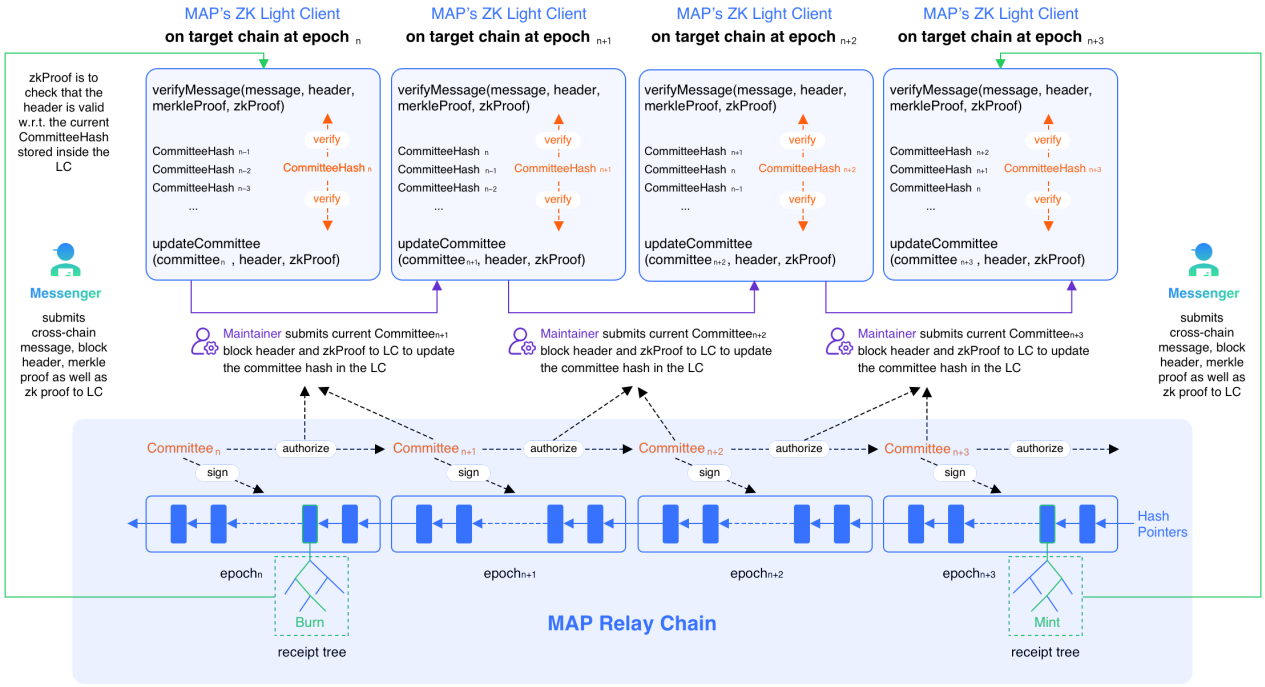
After the origin chain's light client is deployed on the target chain's smart contract, in the event of a changeover in the validator set of the origin chain, maintainers will write the BLS aggregate signature and voting weight of the origin chain validator set into the light client smart contract deployed on the target chain. The light client smart contract deployed on the target chain stores the validator public keys and voting weights of multiple terms of the origin chain (Satoshi's longest chain principle) validator committee.

Since each term of the validator group of the PoS mechanism chain is produced by the signature authorization of the previous term's committee, maintainers, as off-chain programs, if they try to write false origin chain validator set information into the target chain light node, the signature validator information of the previous term stored in the light node (light client) smart contract will not pass its written request because the false signature validator set does not have the signature authorization of the previous committee; to make it pass, it would require attacking the entire origin chain or rewriting the smart contract. Thus, peer-to-peer and independent self-verification is ensured.

When the Origin Chain is a PoW Mechanism Blockchain

In the case where the origin chain's light client is deployed on the target chain's smart contract, maintainers are responsible for synchronizing the latest block header information of the origin chain to the light client contract on the target chain; the light client contract deployed on the target chain stores the latest Nth block header of the origin chain; if maintainers try to write a false block header into the origin chain, the false information will not have a hash value that matches the previous block header, so the false information will not be accepted by the light client smart contract stored on the target chain.

ZK-Improved Light Clients for MAP Relay Chain



The cross-chain verification of MAP Protocol is mainly performed by the light client smart contract of the origin chain deployed on the target chain to perform the following two verifications:

1. Correctness verification of the block header: Verify the legality of the block header requested by the maintainer to be written in. Depending on the different chain consensus mechanisms, this verification scheme will vary. For chains using PoS and BFT mechanisms, it is usually to verify that the legal signature represented in the block header has more than 2/3 of the voting rights.

2. Verification of Merkle proof: Verify that a specific event is emitted at a specific block height, the correct Merkle root value is required in the block header, and the first step ensures correctness. In a group of blockchains similar to Ethereum's structure, this Merkle proof is usually proof of the existence of the Merkle Patricia Trie, that is, the receipt tree MPT does indeed contain a specific event.

Improving the implementation of the MAP Relay Chain light client using zkSNARK technology aims to solve two problems:

1. Reduce the amount of MAP Relay Chain metadata that needs to be stored in the light client contract, lowering the gas cost for updating the state of the light client itself.

2. Move the signature validity and signature weight check part of the block header legality verification process into the zero-knowledge proof circuit and use the Groth16 scheme for verification to reduce gas costs.

In the MAP Relay Chain light client smart contract, it is necessary to store the public keys and staking weight information of all validators in the current epoch. When verifying the validity of a new block header, based on the information in the block header and the current validator information stored in the light client contract, the light client contract can calculate the aggregated public key required for the aggregate signature verification of the block header.

If the aggregate signature verification passes and the sum of the voting weights represented by the aggregate public key exceeds 2/3, the block header will pass verification. In the MAP Relay Chain light client built on zkSNARK, only the commit value of the current validator set metadata needs to be stored, i.e., SHA256((pk_0, wt_0), (pk_1, wt_1), ..., (pk_n, wt_n)). This means the light client contract is simplified from needing to store n public keys and weight information to only needing to store a 256-bit commit value.

Therefore, the block header verification logic of the MAP Relay Chain light client built on zkSNARK can be simplified as whether the input block header is the correct block within the validator set information represented by the current commit value. The success or failure of this judgment depends on the zkSNARK proof input along with the block header.

Cross-chain mechanism after introducing zkSNARK

For clarity, here are the newly introduced roles of the prover in the cross-chain process after introducing zkSNARK — the updated logic of messenger/maintainer, and the inputs received by each party:

```
    function hashToG1(bytes memory message) public view returns (G1 memory h)
{
        uint256 t0 = hashToBase(message, 0x00, 0x01);
        uint256 t1 = hashToBase(message, 0x02, 0x03);

        G1 memory h0 = baseToG1(t0);
        G1 memory h1 = baseToG1(t1);

        // Each BaseToG1 call involves a check that we have a valid curve
point.
        // Here, we check that we have a valid curve point after the addition.
        // Again, this is to ensure that even if something strange happens, we
        // will not return an invalid curvepoint.
        h = bn256G1Add([h0.x, h0.y, h1.x, h1.y]);
        require(bn256G1IsOnCurve(h), "Invalid hash point: not on elliptic
curve");
        require(safeSigningPoint(h), "Dangerous hash point: not safe for
signing");
    }
```

Prover's input: block header, public key of each validator in the current validator set, and their voting weights.

1. Circuit's input: t0 and t1, the aggregate signature to be verified, public keys and voting weights of each validator in the current validator set, and the bitmap indicating the aggregate validators.
2. Input for light client event legitimacy verification: block header, zk-proof, and mpt-proof.
3. Input for light client sync commit legitimacy verification: block header, zk-proof.

Where t0 and t1 are the intermediate values of block header hashToG1, referring to [the current light client implementation code](). The reason why the circuit chooses t0 and t1 as inputs instead of the entire block header is that expressing the hashToBase calculation inside hashToG1 with a circuit is costly and not worth it. When the prover server gets the block header information submitted by the maintainer or messenger, it continues to execute related operations.

The prover can be understood as a server that generates proofs, exposing an interface for proof generation requests, with input parameters: block header, public key of each validator in the current validator set, and their voting weights. The output is zk-proof.

For the aforementioned reasons, before the prover starts calculating the zk-proof, it first extracts and calculates the parameters required to generate the zk-proof from the block header in the request: t0 and t1, and the bitmap indicating the validators participating in the aggregate signature, then performs the specific calculation process of zk-proof.

Circuit's encoded logic

1. According to the bitmap in the block header and the information of the full set of validators, calculate the aggregate public key and verify that the signature weight represented by the aggregate public key exceeds 2/3.
2. Calculate the final result of hashToG1 based on t0 and t1, use this value, the aggregate public key to be verified, and the aggregate BLS verification to check the legitimacy of the signature.

The final zk-proof is an assertion of the legality of the above statement under specific public input. In the process of verifying the block header by the light client, public input is constructed based on the commit information stored by the light client and the block header information to be verified, and the legality of the zk-proof is verified. If it can be verified, it means that the zk-proof is valid, thereby proving the legitimacy of the corresponding block header.

Example Walkthrough

Taking the light client's verification of event legality as an example, its inputs include the block header, zk-proof, and merkle-proof. The entire proof verification process is as follows:

1. Calculate t0 and t1 based on the input block header and hashToBase.
2. Take t0, t1, and the commit of the current validator set stored by itself as public inputs, and verify the legality of zk-proof according to the groth-16 scheme.
3. If the second step is verified, it means the block header is legal. Then extract the root node (MPT root) of the Merkle Patricia Tree (MPT) from it and continue to verify the legality of the mpt-proof.

Step one is an additional calculation introduced to accommodate the construction of the circuit. The appropriate principle of using the appropriate scheme to verify the appropriate thing is reflected: the calculation involved in hashToBase is not costly with

solidity, but it is not worth it with a circuit. The key step of block header verification is the second step, which actually verifies one thing: the input block header is a legitimate block header under the validator set corresponding to the commit stored by the light client. This inference can be easily deduced from the internal logic of the circuit. The third step has no difference from the current implementation of the light client.

The process of the light client verifying the legitimacy of the synchronization submission is basically consistent with the above process, with the only difference being that the operation of verifying the legality of the MPT root in the third step is replaced with recalculating the commit according to the new validator set information and updating its own storage.

Conclusion

According to the foregoing discussion, after the MAP Relay Chain light client is upgraded based on zkSNARK, the corresponding adjustments that the maintainer/messenger needs to make are as follows:

1. Before the maintainer wants to update the light client state, they need to obtain zk-proof from the prover and construct a tx with the block header and zk-proof.
2. Before the messenger wants to initiate a cross-chain request, they also need to obtain zk-proof from the prover and construct a tx with the block header, zk-proof, and mpt-proof.

Users who interact cross-chain through the front end usually feel the changes happening behind the scenes. To highlight the characteristics of zero-knowledge proof, some information related to zero-knowledge proof can be displayed on the frontend:

1. When the messenger submits a zk-proof request, the frontend can simply display a notification prompt, such as: "A cross-chain zk-proof request has been submitted to the prover."
2. When the messenger obtains the zk-proof request, the front end can simply display a notification prompt, such as: "zk-proof has been obtained."
3. When the messenger submits a transaction containing the block header, zk-proof, and mpt-proof, the frontend prompts: "zk-proof and cross-chain request have been submitted to the target chain."

After the third step's transaction is packaged, the front end can prompt: "The zk-proof and cross-chain request have been processed by the target chain.

Further Ensure MAP Relay Chain Security through the Bitcoin Network

Long-range attacks are a type of assault specifically aimed at Proof of Stake (PoS) systems. Attackers attempt to create a forked chain starting from the early history of the blockchain. If successful, this could redefine the authoritative historical record of the chain, leading to double-spending issues or undermining network trust. In the relay chain architectures, this issue is particularly critical because they often coordinate interactions between multiple independent blockchains within a network. If a relay chain suffers a long-range attack, it could impact all chains that rely on it as a basis for trust and communication.

Bitcoin, with its immense computational power, can be considered a natural source of trust and serves as a timestamp server supported by proof of work. It provides an irreversible time order for events. In its native application, events involve various transactions executed on the Bitcoin ledger. In current applications aimed at enhancing the security of other blockchains, Bitcoin can also be used to timestamp events occurring in other blockchains. Each such event triggers a transaction sent to miners, who subsequently insert it into the Bitcoin ledger, thus timestamping the event. Transactions that timestamp events are referred to as checkpoints.

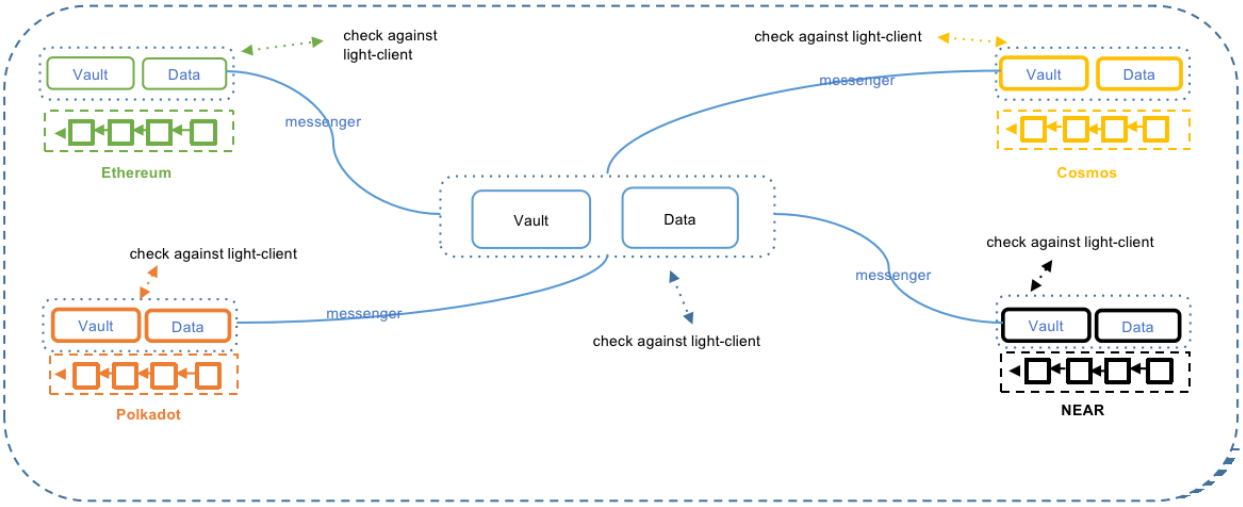
Checkpoints can be implemented using the` OP_RETURN` opcode of Bitcoin, which allows the publication of arbitrary 80-byte data in unspendable Bitcoin transactions. Each checkpoint must contain at least the hash of the PoS block to be checked (32 bytes) and a signature finalizing that block (32 bytes each). Here, the hash is used to identify the PoS block being checkpointed, and the signature is required to prevent adversaries from sending arbitrary hashes and pretending to checkpoint PoS blocks on Bitcoin.

A PoS chain can enhance its security and address the long-range attack problem by utilizing the Bitcoin timestamp service's features. The MAPO platform regularly (every epoch) submits the hash and signature of the last block of each epoch as a checkpoint to the Bitcoin network. These checkpoints consist of the hash of the block and a single aggregated BLS signature, corresponding to the signature of the 2/3 set of validators who signed the block for finality, as well as the epoch number and bitmap number.

As a result, MAPO clients can determine the final canonical chain of the MAP Protocol‘s PoS chain by retrieving checkpoints from the Bitcoin network, thus protecting against long-range attacks by malicious validators on the MAP Protocol network.

MAP Omnichain Service (MOS) Layer

The MOS layer includes Messenger, Vault & Data, and other cross-chain message components.



Messenger

Messenger is an independent inter-chain program. It listens to relevant events as preset in the program and builds a proof on the ledger of the source chain; then transmits the message of the event and proof to Vault or Data on the target chain.

Messenger prepays the gas fee in MAP Relay Chain and the target chain for users; but since the gas fee for target chains cannot be estimated, rewards for Messenger will be provided by dApps. The flexibility of the application provides Messenger with numerous possibilities. Applications can collect customizable transaction fees from omnichain users and consequently reward Messenger. As a main component of MOS, Messenger SDK is fully open to dApp developers.

Messenger is a high-concurrency inter-chain program. Theoretically, as long as one honest Messenger is working in between chains, all cross-chain transaction messages of the dApp can be transferred. Malicious attacks from Messengers will not result in asset loss, but the failure of cross-chain verification of MAP Protocol at the protocol layer.

Vault & Data

On the source chain, Vault & Data contracts are responsible for receiving assets or data and triggering events for Messenger to listen to. On the relay chain or target chain, Vault and Data contracts are responsible for receiving cross-chain messages sent by

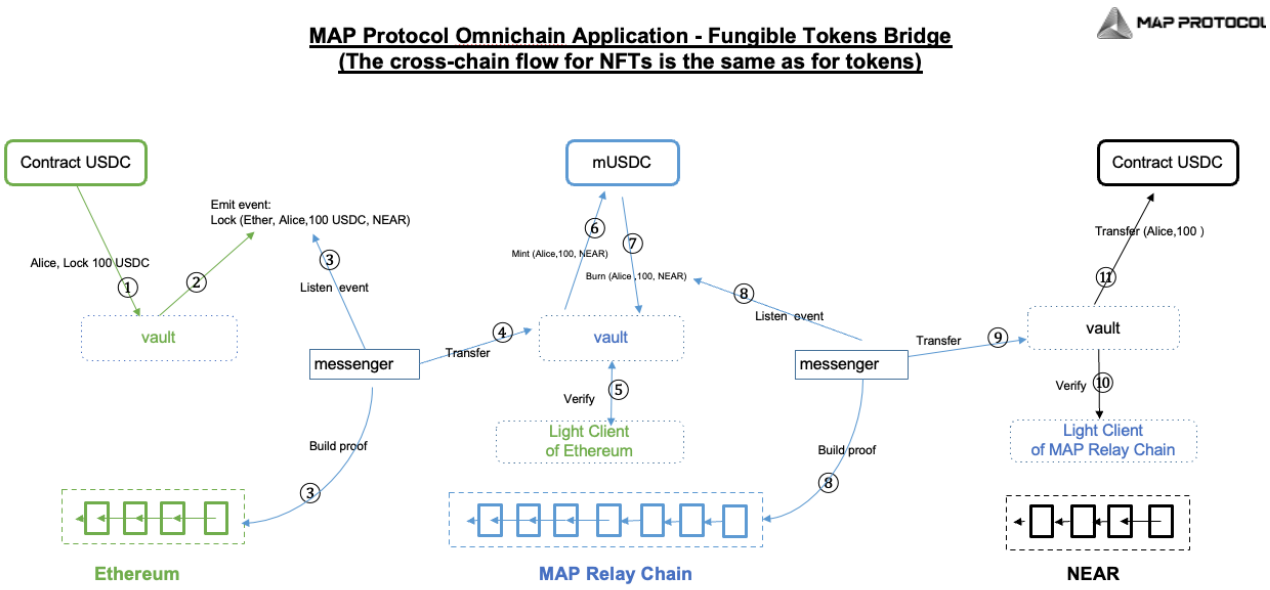
Messenger and forwarding the request to the light client of the source chain deployed on the relay chain/target chain. Once the verification is complete, Vault and Data will execute the corresponding instructions.

MAP Protocol Application Layer

When building an omnichain dApp, it is necessary to know first how assets cross-chain and cross-chian messaging is done. Below are the details.

Asset Cross-chain

The peer-to-peer asset cross-chain flow is as follows:

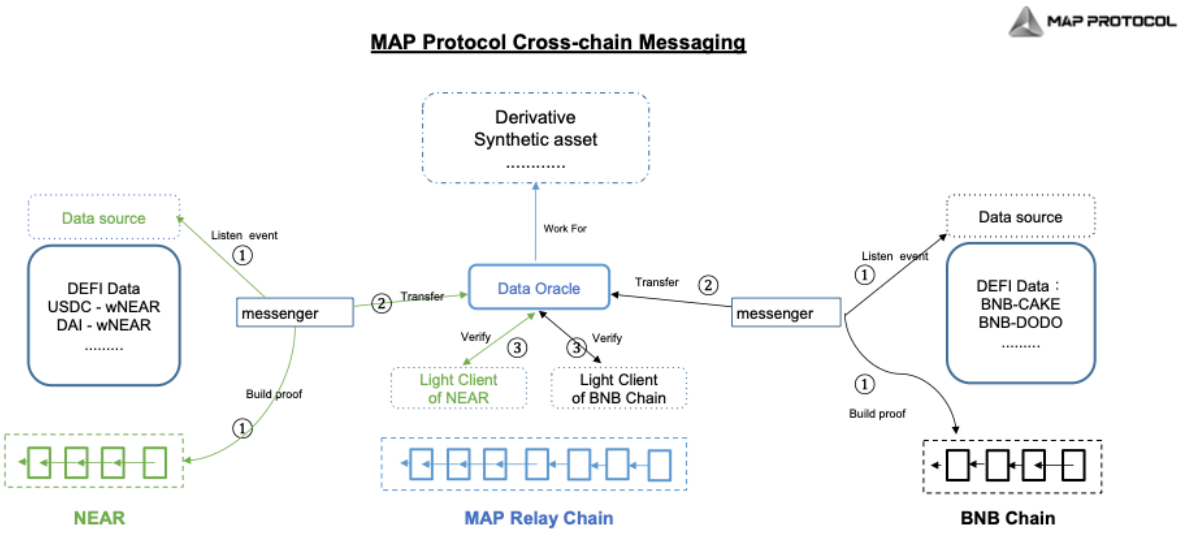


1. The Vault cross-chain smart contract (MOS component) receives assets.
2. The Vault contract emits information containing the origin chain address, target chain address, token type, and amount to the Messenger.
3. Upon detecting this event, and in order to verify its authenticity, the Messenger needs to construct proof of this event being recorded on the origin chain. The Messenger forwards this request to the relay chain's Vault contract.
4. The Vault contract hands over the verification of this request's authenticity to the light client smart contract deployed on the relay chain from the origin chain.

5. The light client smart contract stores the longest chain block header information of the origin chain and can verify whether the transaction has indeed occurred on the origin chain.
6. The Vault contract mints an equivalent amount of the relay chain version of the cross-chain request assets on the relay chain.
7. The Vault contract destroys the minted assets. Steps six and seven are for bookkeeping on the relay chain, so the assets are minted and then destroyed.
8. The Messenger listens for the event on the relay chain and constructs a proof of this transaction being recorded on the relay chain.
9. The Messenger then forwards the message to the target chain's Vault contract.
10. The Vault contract submits the cross-chain request to the relay chain's light client smart contract deployed on the target chain for verification.
11. Upon receiving the light client's verification pass instruction, the Vault contract releases the tokens to the target chain address specified by the origin chain smart contract.

Cross-chainn Messaging

Cross-chain messaging is the interoperability of smart contracts. The steps are similar to asset cross-chain.



The composability of smart contracts on different chains can further enhance DeFi efficiency, allowing a user or a smart contract to interact with DeFi products on other chains and return their original assets on their original chain in a peer-to-peer manner, regardless of which token they use or which chain they are on.

MAP Protocol Tokenomics

As an omnichain infrastructure, MAP Protocol is dedicated to building an open economy powering everyone equally to participate and its ecosystem operations, growth initiatives, and strategic investments. Many public blockchain projects have monetary systems that solely incentivize their node operators. MAP Protocol's tokenomics is designed to compensate more diverse forms of contributions from a wider range of participants and has a built-in incentive structure to procure sustained resources to fuel future growth initiatives and strategically sourced investment projects in addition to maintaining its blockchain nodes.

Allocation and Distribution

MAP PROTOCOL

Tokenomics

Initial token supply: **10,000,000,000**

Group	Ratio	2023				2024					2025	
		March	April	May	June	June	July	Aug	Sept	Dec	Jan	June
Team	15%	50,000,000	50,000,000	50,000,000	50,000,000	100,000,000	100,000,000			200,000,000	200,000,000	50,000,000
Foundation	12%	100,000,000	100,000,000	100,000,000	50,000,000	200,000,000	100,000,000	100,000,000	100,000,000			
Ecosystem DAO	21%	This Ecosystem removes the previous released 3.11% of 21%. The remaining 17.89% will form a DAO Treasure, used for developer incentives, cross-chain vault incentives, marketing and community incentives, etc. It will be released through DAO governance, and community members can participate in voting through community proposals. Participate and earn part of the tokens and share the benefits.										
Institutions and Partners	22%	Used for project development.										
Mining	30%	Released by 1% p.a. for the first 2 years, then 2% for the next 14 years.										

The total supply of MAPO is 10 billion. MAP Protocol's token allocation is in line with incentivizing MAP Protocol's network's block generation, network maintenance, ecosystem development, and community growth. Detailed distribution and predetermined ratio is as follows:

- 15% are for developers of MAP Protocol, with a vesting period of 2019 to 2025.
- 21% are for Ecosystem DAO, not locked, and fully decided by the MAPO Community on how the token should be used. For MAPDAO governance, all major decisions that may impact community members need to be fully discussed on the MAP forum and subsequently be voted on-chain.
- 12% are for MAP Foundation for building the initial state of the MAP Protocol ecosystem and the web3 omnichain ecosystem before it becomes fully decentralized.
- 22% are for its investors and early supporters.
- 30% are mining rewards for validators on MAP Relay Chain, and Maintainers.

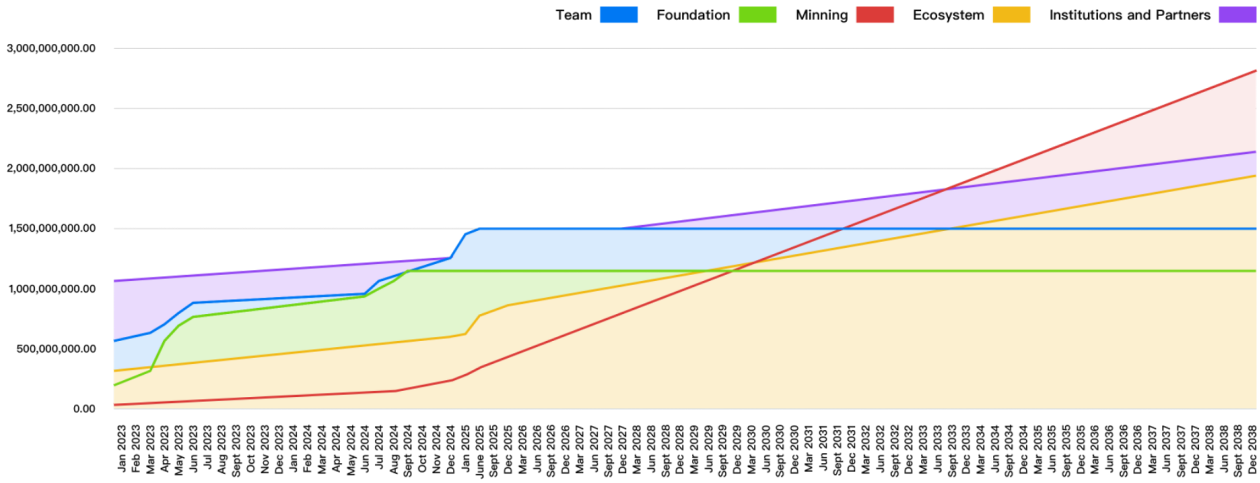
Fee Model

MAP Protocol will only charge the gas fee generated on MAP Relay Chain for each cross-chain transaction. Maintainer can get extra rewards from MAP Protocol for updating and maintaining Light-Clients.

As an inter-chain program, Messenger is an essential part of MOS. Messenger needs to prepay the gas fees of MAP Relay Chain and the target chain for omnichain users. Fees on the target chain cannot be estimated, so MAP Protocol provides the Messenger SDK to developers. Application Layer provides the flexibility for dApp developers to determine the cross-chain transaction fee standard, rewards to Messengers, and their entry requirements.

Vault & Data deployed on each chain are also essential parts of the MAP Omnichain Service (MOS) Layer, responsible for managing assets (fungible tokens and NFTs) and data on each chain. For developers of Vault & Data, MOS will not charge any fees. Applications can determine the fee structure for sharing the liquidity in vaults and data pools at their discretion.

Anticipated unlocking Schedule in the next 15 years



Both unlocked (i.e. tokens that are available for distribution) and locked (i.e. not available for distribution) tokens can be staked. It is important to note that anticipated total circulating token supply for the next 15 years is subject to network performance and staking assumptions.

How Community Members Can Participate and Earn

Each layer of MAP Protocol offers opportunities for community members to participate.

On Protocol Layer

- Run a node and become a validator: Validators are the foundation of the MAP relay chain. Community members can stake a certain amount of $MAPO and run a node with required computational power. Community members can also delegate their $MAPO to validation node operators.
- Become a Maintainer: The value of Maintainers lies in updating the status of light clients deployed on the target chain, ensuring the smooth operation of the verification network. When a Maintainer updates a light client, a transaction must be written to the chain, incurring gas fees. Therefore, the MAP Protocol's economic model includes a separate component designed to incentivize and compensate Maintainers. Running as a Maintainer requires computational capacity, sufficient cash flow to prepay gas fees on the target chain, and staking $MAPO.

On MAP Omnichain Service (MOS) Layer

- Become a Liquidity Provider: Community members can provide liquidity to Vaults on each chain through dApps deployed within the MAP Protocol ecosystem. Each dApp designs its own incentive model.
- Run Messenger: Running Messenger requires providing sufficient gas fees for the MAP relay chain ($MAPO) and the target chain (native tokens). Each dApp establishes its incentivization scheme.

On Application Layer

- Be a dApp Developer: With the complete set of SDKs provided by MAP Protocol, dApp developers can build various cross-chain applications.

MAP Protocol Use Case Examples

Omnichain Token Issuance

When a project issues tokens, it often involves fragmented issuance across multiple chains, requiring users to navigate complex cross-chain exchanges through external bridges. However, with MAP Protocol, projects can achieve omnichain coverage at the beginning of token issuance, and the ledgers of different chains will be aligned automatically.

Through cross-chain messaging, intrinsic cross-chain exchange interoperability is realized. Even though users may be unaware of the presence of multiple chains, the chain coverage is indeed multiple. This user experience is analogous to the usage of overseas and domestic currencies.

Omnichain Loan

If a user has funds on Chain A but wants to mine on Chain B, they typically need to go through 9 steps, each of which involves friction costs. Collateralize on Chain A → borrow → bridge (fee) → swap (fee) → farm on the destination chain → swap back (fee) → bridge back (fee) → repay the loan → un-collateralize.

Through MAP Protocol, users can directly collateralize on Chain A, complete borrowing, mining, repayment, and collateral unlocking on the target chain, effectively skipping four instances of cross-chain bridge and exchange fees.

Omnichain Swap

Omnichain Swap allows users to swap coins with substantially lower fees than traditional DeFi exchanges by connecting users to the best cross-chain DeFi protocols. By using MAP Protocol, developers can build a truly decentralized peer-to-peer omnichain exchange that enables users to swap any token on any chain.

Omnichain swap achieves omnichain swap aggregation by connecting every major DEX liquidity. Existing AMMs can be wrapped to achieve omnichain swaps from one asset to another without the need to modify any existing code. Users can swap from $ETH on Ethereum to $NEAR on NEAR Protocol in one single transaction on the source chain.

In an omnichain swap built with MAP Protocol, users can add multichain coin liquidity in one pool, which means providing liquidity to token pairs from different chains becomes possible. Users can swap one token directly for another token on a different chain

without using any intermediate token, such as stablecoins, thus enjoying the shortest route for omnichain swap.

** ButterSwap, the first truly decentralized cross-chain exchange that enables users to swap any token on any chain, includes all the above functionalities and officially launched on March 30, 2023.*

Omnichain and Fully On-chain Games

With the rise of the cryptocurrency industry and the concept of the metaverse, many innovative ideas have been brought into the realm of traditional gaming. GameFi refers to the gamification of financial systems, where profits are generated through participation in cryptocurrency games that offer earning opportunities. Unlike traditional games, play-to-earn games incentivize players to earn rewards. Players can create in-game assets and have full ownership over them.

Omnichain and fully on-chain games extends beyond cross-chain but means that every aspect of the game other than the front end (the part of the game the player sees on their screen) is run solely using blockchain technology. Even the narrative and governance for the development of the game will also be decentralized based on DAOs.

MAP protocol supports different types of blockchains to cross-chain and interoperable. Through MAP Protocol, not only GameFi projects can connect to EVMs and non-EVMs, but also have their storage and computing on-chain. Additionally, since MAP Relay Chain actively connects with all types of chains, GameFi projects can fully focus on enhancing user experiences without concerning about any scalability and security issues.

On-Chain Data: On-chain Oracle and Derivatives

Decentralized derivative and synthetic assets are usually constrained by the accuracy and timeliness of asset prices and quantities originating from other chains. Such issue can be solved by multi-chain deployment, but this is extremely complex.

By building a reliable omnichain network, MAP Protocol has enabled Data cross-chain and is nurturing a completely new 预言机 (Oracle) market - On-Chain Oracle. By deploying on MAP Relay Chain, derivative and synthetic asset applications can easily acquire reliable multi-chain data from On-Chain Oracle.

Omnichain Governance — Aave

When Aave executes a proposal (which is established on the Ethereum network), the proposal is forwarded to the Polygon (MATIC) FxPortal. Subsequently, the mechanism reads data from Ethereum and passes it to the Polygon network for verification. Later on, the bridging contract for Aave‘s cross-chain governance receives this data, parses it, and queues it for the next step, which is pending a timelock for finalization.

The Aave cross-chain governance bridge is built in a generic way to be easily adapted to operate with any chain that supports the EVM and cross-chain messaging. Currently, the repository supports contracts bridging to Polygon and Arbitrum. On Aave, users can submit Aave Improvement Protocols, or AIPs, to target various features on the DeFi platform.

Although this solution seems fairly comprehensive, with the help of MAP Protocol's omnichain interoperability, governance on Aave could achieve omnichain management across all EVM and heterogeneous chains.

Fungible Token and NFT Bridge

Cross-chain bridges and cross-chain NFT bridges no longer have to build their infrastructure or use MPC. Using MAP Protocol’s peer-to-peer cross-chain verification network and the MOS application developer service kit, bridge developers can easily build their NFT or fungible token bridge application.

AIGC + Web3

AI data generated by users when they interact with AIGC services can be stored on decentralized storage Layer 1 (L1). However, for those data to be tradable as assets, a decentralized L1 that has a thriving DeFi market will be needed. Through peer-to-peer cross-chain messaging, trading user-generated AI data as assets will be possible.

BRC-20 Asset Transfer

BRC-20 is an experimental standard for fungible tokens on the Bitcoin blockchain and inscribed on the Bitcoin network. Through MAPO BRC201 Protocol— an extension protocol to the BRC-20 standard, MAP Protocol supports the cross-chain transfer of BRC-20 tokens from the Bitcoin network to the MAP Protocol network in a peer-to-peer manner. This also enables other cryptocurrencies on different blockchains to be traded with BRC-20 assets more conveniently and cost-effectively, enhancing the liquidity of inscription assets.

This layer of interoperability helps expand the use cases of Bitcoin and integrates the Bitcoin ecosystem into a broader crypto-based financial ecosystem, bringing new contributions to the Bitcoin community.

Conclusion

MAP Protocol is a Bitcoin layer-2 as well as a peer-to-peer omnichain network built upon light clients and ZK technology, focusing on cross-chain interoperability. It provides the essential omnichain infrastructure for achieving interoperability among blockchain-based assets, storage, and computing across EVM and non-EVM chains.

As an omnichain infrastructure, MAP Protocol does not rely on any trusted third party for cross-chain communication. The only trust is put in the code, which leverages the self-verifying nature of light clients to ensure cross-chain interaction in a completely peer-to-peer manner. When a cross-chain request occurs on the source chain, it is transmitted to the target chain via an off-chain role. Then, the light client deployed on the source chain will verify the validity of the cross-chain request sent by the off-chain role in a peer-to-peer manner.

The security of MAP Protocol is further enhanced through the Bitcoin network. All relevant information on MAP Protocol, such as epochs and the PoS consensus of the MAP relay chain, is written into the Bitcoin blockchain as transactions. This prevents long-range attacks on the relay chain and further ensures the security of the relay chain by utilizing the security mechanisms of the Bitcoin network.

MAP Protocol is a flexible and innovative omnichain infrastructure as well as a platform for peer-to-peer cross-chain interoperability and Web3 dApp development, rather than a closed, single-use protocol specifically for a particular use case. It is open in terms of technical design and community development, and we look forward to serving as the foundation for many new financial and non-financial innovations.