MAP Protocol Whitepaper 4.0

January 19, 2021

Contents

1 UpdateList

This version is MAP protocol White Paper V4.0. It updates the section such as Introduction, MAP design philosophy, MAP Feeder sub modules, The Economic Model of the MAP Token. And it adds section such as Chain Gateway ID Specifications and Cross-Chain Data Stream Fee section, MAP bridge in Polkadot. And it also move some some technical details to Appendices.

2 Overall

MAP protocol is an open, fully decentralized chain-to-chain interoperation protocol that enables the interoperability of multiple independently verifiable consensus blockchains. MAP protocol expects to construct interoperable blockchain network, linking different blockchains together, integrating the resources and capabilities of each chain, provide a complete and available blockchain underlying infrastructure for finance, AI, Internet of things, traceability, governance and other applications.

If we treat each blockchain as a decentralized computer (which can automatically execute computation and store data), Polkadot and Cosmos are committed to building a cloud-like computing system and chain interoperability could be achieved through a specified relay chain or a hub chain. MAP protocol is a TCP/IP-like protocol, which define the interoperation regulations, to link different blockchains together.

Also we will provide one blockchain (MAP Standard Chain) which fully implement MAP protocol. It would play important role in the early ecosystem governance. It worth mentioning that MAP Standard Chain is not the only ecosystem management chain, other blockchain can server the ecosystem as management chain as well in the future if community promote.

3 Background Introduction

Since the birth of the Bitcoin in 2009, thousands of public chains have been produced by using blockchain technology and those are widely used in multiple application fields such as IoT, finance, governance, identity management, and traceability. In the early 2010s, there are only few blockchain. And the terminology blockchain interoperation mostly refer to the cross-chain technology of transferring coins between these blockchains. Sidechain are one example of cross-chain solution. It can lock certain amount of digit assets in one blockchain and unlock them in another blockchain. Later in a paper from 2016, Buterin lists ways of reaching interoperability. This paper formally analyzes the chain interoperability and discusses trusted inter-chains exchanges.

If we treat blockchain system as a distributed database, any user in this blockchain system has two basic operation for this database: read and write. Read means users could retrieve the data from the blockchain(e.g., state of certain accounts or transaction in certain blocks). It can be achieve through fully synchronization(i.e., downloading the entire blockchain and update them in real time). Besides, users could read from the blockchain using SPV technology. SPV is much cheaper in network cost than fully synchronization. And for write, users can not write date to the database directly. They have to broadcast their write requests(pending transactions) to the P2P network and wait the consensus engine to commit them(normally we call this "mining a block").

Analogously, chain interoperability means one blockchain(active chain) has the ability to read and write to another blockchain(passive chain). For cross-chain read, it means active chain can retrieve the data from passive chain. Most current chain interoperability solution suggest to use SPV technology for cross-chain read. SPV need a trusted third party to provide the block header of passive chain, and any node from passive chain could request and verify Merkle proof to retrieve any information contained in this block-

header. The trusted third party are normally called relayer or relaychain. They have to synchronize the entire header chain of active blockchain for any node who need SPV service. And for cross-chain write, it means active chain could change the state of passive chain. There are normally two ways to achieve cross-chain write: active write and passive write. Active write means active chain can construct a transaction which satisfy the rule of passive chain and broadcast it to the P2P network of passive chain. Once this transaction is committed to a confirmed block of passive chain, the cross-chain write is complete. Passive write means passive chain could change its state based on some trigger conditions. The trigger conditions should be events from active chain.

Now let us examine some chain interoperability solution:

Two-way pegged sidechain: Sidechain could transfer some digit asset from parent chain to side chain. User could freeze some coins first using a locked transaction in parent chain. Any node from side chain check if the locked transaction from parent chain is confrimed through SPV. And side chain would mint new coins once the locked transaction passed the contest period. It is a passive write example, the trigger condition is confirmed locked transaction passed the contest period.

Lightning network: Lightning network could lock some BTC to an off-chain payment channel created by two owners. They could negotiate about the new distribution plan and update it any time. Any owner could destroy the payment channel any time and claim their BTC according to the most recent distribution plan. The way lightning network change the Bitcoin network can be categorized as active write. Lightning network would construct two claim transactions which satisfy the Bitcoin rules. Once these two claim transactions is committed to a confirmed block, the Lightning network succeed change the state of Bitcoin Network.

BTC-Relay: BTC-Relay could construct a one-way bridge from Bitcoin to Ethereum. Some relayer could deliver the block headers of Bitcoin to the Relay contract in Ethereum. Through the Relay contract, any smart contract in Ethereum could retrieve certain transaction of Bitcoin using SPV, and change the state of Ethereum based on it. It can be categorized as passive write, the trigger condition is transaction in Bitcoin network.

These early solutions of chain interoperability are mostly designed for specific blockchain and thus not systematic. This means these solutions is hard to extend for constructing an interoperation network for multiple chains. Two systematic interoperation protocol Polkadot and Cosmos was proposed.

Polkadot: Polkadot defines a complete cross-chain interoperability underlying protocol. It has a complete cross-chain read and write specification, and through this protocol, it has built a complete cross-chain interoperability ecosystem. The characteristic of Polkadot is to communicate and coordinate the cross-chain interoperability of all para chains through a relay chain. The validator on the relay chain will be allocated to each para chain to work with its collator, and the para chain block header provided by collator will be synced to the relay chain. After that, the XMCP protocol is used to transfer cross-chain messages. Of course, the cross-chain status needs to be obtained through the SPV solution after obtaining the block header from the relay chain, and the cross-chain write operation needs to be customized through the para chain. The smart contract system parses the specification information defined by XMCP and executes it. We must note that the soul of Polkadot is the relay chain. The relay chain not only needs to coordinate cross-chain information interaction, but also take the responsibility of the shared security in the entire system.

Cosmos: Cosmos is also a complete cross-chain interoperability operating system. It defines a set of IBC protocols for cross-chain communication, which can guarantee asset transfer or data transmission between different chains, and communication between different HUB chains requires cross-chain communication through the IBC protocol, but reading between different HUB chains requires a Relayed cluster to provide blockheads. The design of the IBC protocol is like the two-way pegging. It consists of four parts: first, Tracking, where the Relayed cluster collects block headers for each HUB chain; then Bonding, which locks a portion of an asset on a chain; and then Proof Relay, which gets the block head and corresponding SPV proof from the Relayed cluster; and finally, The Validation, the proof obtained in the next step is validated and can be followed if the validation passes. Cosmos's cross-chain read operations rely on SPV proof provided by the Relayed cluster, while cross-chain writes required subsequent operations through the proof of validation through smart contracts.

MAP can guarantee chain interoperability for POX consensus based blockchains without relayer or relaychain, meanwhile for DPOX based blockchains, MAP will provide a pegged relay solution so that they can achieve chain interoperability.

4 MAP Protocol

MAP protocol is an open, fully decentralized chain-chain interoperation protocol. The chain itself can achieve lightweight arbitrary interaction without the need to relay through a relay chain under the independent self-verified consensus mechanism. And for the non-independent self-verified consensus mechanism, the chain can achieve lightweight arbitrary interaction through the relay anchoring mechanism. The reason for the difference here is that blockchains of independently self-verified consensuses (e.g. POW, POS, etc.) can independently verify the legitimacy of individual blocks. However other consensus mechanisms such as DPOW, DPOS, which need to be anchored to other relay chains because additional electoral representation information is required to verify the legitimacy of blocks. Any blockchain system that supports the MAP protocol can interoperate with other blockchain that also supports the MAP protocol. In this way, an open blockchain interoperability ecosystem is formed. There are no specified relay chains or other types of central chains in the MAP ecosystem, and each blockchain system maintains sufficient independence to ensure the validity and consistency.

4.1 MAP Protocol Design Philosophy

Unlike other chain interoperation solutions, MAP protocol is trying to achieve the chain interoperability in a trustless setting. This means there is no trust third party. The security of MAP protocol should rely on the cryptographic evidence. Besides, the protocol need to make sure the size of the cryptographic evidence is reasonable small. Furthermore, the verification procedure should be non-interactive. This three rules is the reason why we design MAP protocol. We can summarized them as succinctness, security and non-interactivity.

.

4.2 MAP Protocol core Sub Modules

MAP protocol includes two core sub-module:

(1) Ultra-Light Verification Protocol (ULVP)ULVP is a crosschain reading module based on flyclient[4] technology. It is a relayer-free sub-linear verification protocol for POX-based blockchain, it can capture state and transaction from POX-based blockchain with limit cost.

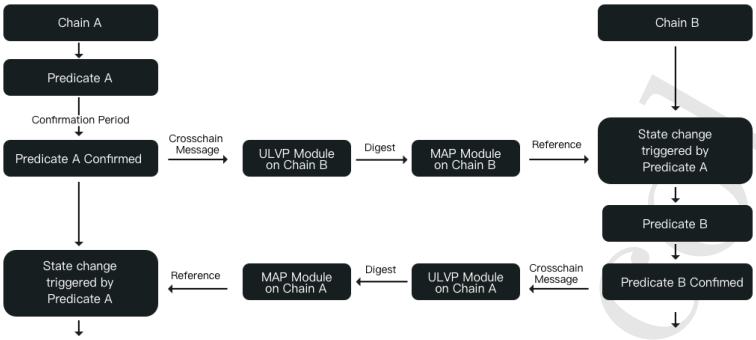

Figure 1: MAP Protocol Architecture

(2) MAP Feeder MAP-Feeder module is a crosschain writing module. MAP-Feeder module will provide a reference contract which can digest the data of other chain from ULVP module and provide it to target chain. Any contract with trigger condition can be triggered by calling the reference contract if the trigger condition from other chain happened.

4.2.1 Ultra-Light Verification Protocol (ULVP)

The MAP-ULVP module is responsible for cross-chain information reading. According to introduction, cross-chain reading in most implemented based on SPV[6] technology. The premise of this technique requires that the verifier needs to have all block headers ahead. The verifier can download all the headers of blockchain locally and update it in real time or they can retrieve it from a trusted third party(Relayer). If a transaction in a particular block needs to be verified, any full node can be asked to provide the Merkel path proof of the transaction in the Merkle tree. If the Merkel paht proof can pass the validation, it can be granted that this transaction is indeed pegged in this block. SPV-based solution require a large portion of local storage or additional trust on third party. The cross-chain information reading of the MAP-ULVP sub-module is based on the flyclient technology, which requires less data compared to the SPV system and meanwhile provide cryptographic security.

ULVP is a sub-linear verification method based on cryptographic evidence with logarithmic space complexity. It can capture data(normally states or transactions) from blockchain from any Proof-of-X(POX) public chain with

limited cost. Research in this area began with Kiayias et al [1]'s work on NIPOPOW. Their solution can be applied to the POW type of public chain in the invariable difficulty setting. Most POW-based blockchain can not use this technology directly because they are under a variable difficulty setting. Later Bunz et al proposed Flyclient to extend NIPOPOW to any POX public chain (POW, POS, POET, etc.). MAP protocol uses Flyclient technology to construct ULVP. More technical details could be found in Appendix B.

ULVP has the following three characteristics:

(1) **Security:** MAP on-chain oracle can ensure that the result of the predicate query is consistent with the honest majority's choice. An adversary can cheat the verifier with a negligible probability(less than $2^{-\lambda}$). It should be noted here that the security we are talking about is the security of the verification protocol itself, not the security of the consensus of the blockchain.

(2) **Succinctness:** MAP on-chain oracle can ensure that the network resources and local storage resources the verifier node need when acquiring the result of a predicate are sub-linear growth, that is, they do not increase significantly with the height of the chain. For example, the length of Ethereum blockchain is approximate $n = 2^{22}$, and suppose one attacker can hold around one third hash power of the total network, the proof size is below 400 KB with a failure probability of 2^{-50}, which is significant small compare to 5GB which SPV requires.

(3) **Non-interactive:** MAP on-chain oracle can ensure that the verification process is non-interactive, that is, the final result would not be obtained through multiple rounds of question and answer communication.

4.2.2 MAP Feeder

MAP Feeder is used to provide cross-chain writing functionality. It will provides a reference system contract on the passive chain, which could obtain the information on the active chain through the interface provided by ULVP module. In this way, any smart contract on the passive chain can obtain the information on the active chain, simply by calling the reference system contract and carry out the subsequent operations accordingly, thus achieving the data writing. The details can be seen in Fig 2.

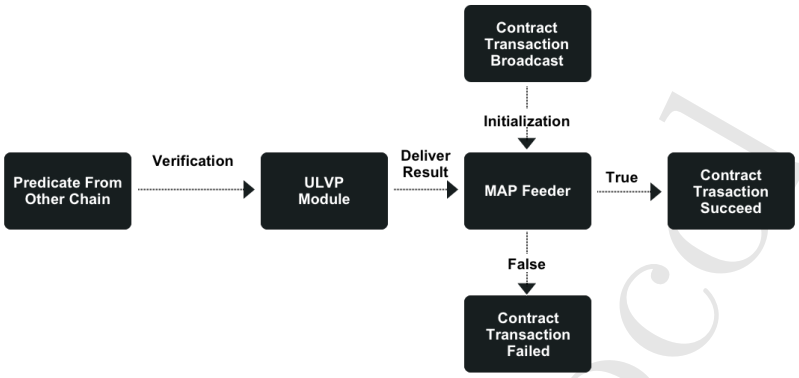

Figure 2: MAP Feeder

Any operation on the chain is triggered by certain pre-events. Once these specified pre-events are triggered, subsequent events can be executed in sequence. In chain interoperation setting, the triggering mechanism of the operations on the chain is complicated. We can summarize them into three categories:

(1) **The Status of the Chain:** for example, the length of the chain, whether there is a transaction in a block, the balance in an account, the tokens in a smart contract, etc. This information can be obtained directly from all nodes of the chain through the chain verification protocol.

(2) **Cryptographic Verifiable Information:** such as pre-image mapping of hash locks, transaction signatures, zero-knowledge proof functions, etc. This type of information can be independently verified locally by any node, generally some cryptographic evidence. This information can generally be obtained in webcasts.

(3) **On-chain Status of Another Chain:** for example, the length of another chain, the account status of another chain, etc. We need to obtain this kind of information through the cross-chain verification protocol mentioned above.

MAP feeder is build upon one smart contract system which can absorb different sources of trigger conditions. MAP feeder can provide such functionality: it can take one or more trigger conditions as inputs and output

some sequential transactions to a certain blockchain. Once the trigger conditions are met (multiple trigger conditions can be combined according to the logic arithmetic such as AND and OR), the smart contract can authorize and publish the sequential transaction to target blockchain. The trigger condition here can be one of the three categories or a combination of them, and the sequential transactions issued by the smart contract can be on local chains or on other designated chains. For example, the two-way pegged sidechain would use the third trigger condition. The side chain needs to verify whether the balance of a specified account on the main chain is locked for a sufficient time(on-chain status of main chain). Once the trigger is successful, the smart contract will allow specific account to generate a corresponding number of main chain tokens on the side chain. And for the atomic-swap transaction scheme, the trigger condition is a combination of the first one and second one. The transaction trigger on the chain requires that the account balance of the chain is sufficient(the status of the chain), and it has a specific signature and a preset map or time of a certain hash lock(cryptographic verifiable information).

4.2.3 MAP Application Tool Environment(MATE)

MATE is an open-source blockchain development tool-kit to efficiently build different blockchains. MATE provide most functionalities like accounts, balances, governance, and smart contracts as easy as plugging in a library. There are three levels of MATE: the core modules for blockchain, the runtime module and the map interoperation module.

The core modules in MATE are the necessary modules for blockchain. For example, consensus module, libp2p module, database, memory pool, cryptography module,governance module, etc. Most of these modules are similar to BTC, ETH systems. Need to mention that consensus module would support POW, MAP-POS and DPOS. MATE would support more kind of consensus in the future.

The map interoperation module would implement the core sub-modules of map protocol and provide the chain interoperability for blockchain.

The runtime module of MATE is called SMART (Sustainable MAP RunTime). SMART includes the following components:

(1) **MAP-VM:** a trusty-worthy WebAssembly virtual machine

(2) **Delta language:** Smart contract language suitable for SMART development;

(3) **Runtime:** Runtime environment based on MAP VM.

More technical details regarding SMART could be found in Appendix C.

4.3 MAP Protocol add-on modules

Besides three core modules, MAP would introduce two more add-on modules:

(1) MAP Standardized P2P Stack

(2) MAP General Transport Channel Based on IBLT(Invertible Bloom Lookup Table)

4.3.1 MAP Standardized P2P Stack

MAP Standardized P2P Stack module is a P2P node discovery and data transfer protocol implemented based on Libp2p library. By linking the underlying P2P networks of different blockchain systems, the nodes from different blockchain systems can be connected together directly. This would increase the security and efficiency, to achieve real-time interoperation.

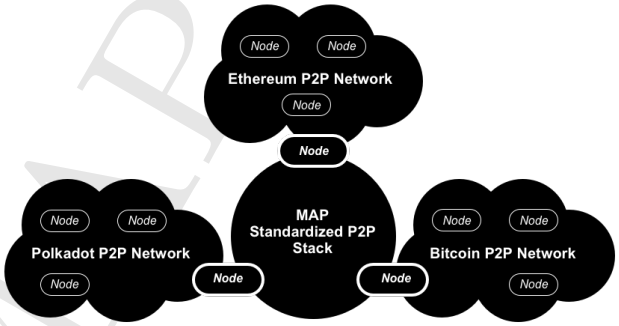

Figure 3: MAP Standardized P2P Stack

The existing blockchain underlying node discovery and data transmission algorithms are not interoperable. Bitcoin nodes cannot directly find Ethereum nodes through the P2P protocol and they have to establish a TCP

connection to realize it. However, MAP protocol requires P2P nodes from different blockchain network can communicate with each other. Therefore, we need a unified and standardized P2P communication protocol. The Protocol Lab's LibP2P protocol just meets our needs. LibP2P is a basic module built for P2P networks. It highly abstracts the mainstream transmission protocol, so that the application layer does not need to worry about the specific bottom layer implementation, in which it has achieved the cross-environment and cross-protocol P2P node communication. Currently, Ethereum 2.0, Polkadot, and other projects have announced that they will use LibP2P as their underlying node communication algorithm. MAP protocol also chose the LibP2P algorithm as our node communication algorithm.

In the past, when developing internet applications, it was only necessary to focus on the upper layer logic of the application without reimplementing the underlying communication protocol (TCP/IP). The original intention of the LibP2P design is to support the future decentralized network protocol. The purpose of it is to allow developers to develop decentralized applications without having to pay attention to the specific implementation of the underlying layer. Finally, cross-environment and cross-protocol node communication are realized.

In a distributed P2P network, the relationship between nodes is no longer the traditional server-client model, which requires that each node can perform the role of the server to process responses but also act the role of the client to send requests. In this complex situation, we need a common communication protocol that can support multiple communication protocols to support the inter-communication between arbitrary nodes. The communication protocol needs to support traditional unencrypted TCP/IP communication as well as encrypted communication protocols such as TLS. The protocol needs to include both node discovery and the establishment of long and short connections, as well as a series of functions such as encrypted data transmission. LibP2P is a general protocol that meets all the above requirements.

In an ecosystem of multi-chain interoperability based on the MAP protocol, node discovery and communication in different chains will be involved. Therefore, a common communication protocol must be supported between all chain nodes, in which all nodes are in a large P2P network. At the same time, the particular nodes for different chains require to be under different subnet structures. Therefore, the network structure should be a multi-level, structured network topology. And LibP2P supports structured, unstructured, mixed, and centralized network topology.

4.3.2 MAP General Transport Channel Based on IBLT

MAP protocol's block transmission protocol based on IBLT and the purpose of it is to solve the secondary transmission of transactions during the blockchain transmissions problem and improve the communication transmission efficiency of the entire system.

In the current blockchain design, transaction collection and block packaging transactions are asynchronous, and all transactions are first collected independently by each node and stored in the memory pool. Later, if a node successfully packages the transaction into the block, the block needs to be broadcast, and the transaction in the block will also be broadcast together. However, most of the transactions in this block already exist in the transaction pool of most nodes. This is the transaction secondary transmission problem in the blockchain transmission. In order to solve this problem, Bitcoin and Ethereum communities have proposed the Compact Block solution, that is, when transmitting blocks, only the block header and the hash summary list of transactions are transmitted. The receiving node compares it's own based on this hash summary list. If any transaction is missing in the transaction pool, it will request the sending node to send the complete content of the missing transaction again. Because the hash digest is much smaller than the original data of the transaction, it can save a lot of bandwidth. However, this summary list grows linearly with block size, so it still consumes network resources when facing large blocks.

The new data structure Invertible Bloom Lookup Tables (IBLT) proposed by Goodrich[3] can further optimize the transmission protocol of the block. This data structure is generally used for set reconciliation. This data structure can ensure that the amount of data required by the two communication parties to obtain each other's set is only related to the difference between the two sets and not to the total amount of the two sets. For example, suppose the block packaged by node A contains 10,000 transactions, and the transaction pool of node B also has 10,000 transactions, and only 100 transactions in the block are not in the transaction pool of node B. Under the transmission protocol or the Compact Block protocol, node A needs to send data with a size that is positively related to 10,000 transactions (whether it is 10,000 transactions or 10,000 transaction summaries). However, if we use MAP's transfer protocol, we can compress the 10,000 transactions of node A into an IBLT equivalent to about 100 transactions of data volume and send it to node B. Node B can use the IBLT and its own transaction pool extract

the entire transaction list and obtain the complete information of the entire block. Such a scheme can greatly improve the transmission efficiency of the block and enhance the scalability of the entire system.

5 Ecosystem based on MAP Protocol

5.1 Ways to Participate in MAP Ecosystem

MAP protocol is an open and completely decentralized chain-to-chain interoperation protocol that enables the interoperability of multiple independently verifiable consensus blockchains. Thus those chains who support MAP protocol would form an ecosystem, we call it MAP ecosystem. Blockchains in MAP ecosystem could freely interoperate with each other.

There will be many blockchains with different characteristics in MAP ecosystem, and we will support blockchains with various consensus algorithms to coexist in such an ecosystem. However, there will be some differences according to the specifications of our interoperability agreement.

For blockchain based on POX consensus(including POW and POS), they can participate in MAP ecosystem by three methods: implementing MAP core modules directly, forking from original chain and realizing MAP modules in layer 2. And for blockchain based on DPOX consensus, they can participate in MAP ecosystem through a relay chain(recommend using MAP standard chain).

(1) Implementing MAP core modules directlyFor those blockchains construct using MATE, they can directly interoperate with other blockchains in MAP ecosystem easily. Also developer could implement MAP protocol by their own under the MAP protocol regulation.

(2) Forking from original chain: For some blockchains using POX consensus, community could create a fork chain which support MAP protocol. Thus they can achieve chain interoperability with other blockchains in MAP ecosystem directly.

(3) Realizing MAP modules in layer 2For some blockchains which support layer 2 design, they can just deploy a layer 2 which support MAP protocol, and through this layer 2, the parent chain could achieve chain interoperability with other blockchains in MAP ecosystem.

(4) Through a relay chainFor some blockchains using DPOX consensus, they can interoperate with other blockchains in MAP ecosystem through a relay chain. the relay chain would record and update the delegation information and other necessary verification data so that other blockchain in MAP ecosystem could use. Through this relay chain, DPOX blockchains could achieve chain interoperability with other blockchains in MAP ecosystem indirectly.

5.2 MAP Standard Chain

As we mentioned above, we will launch a MAP protocol supported blockchain called MAP standard chain. MAP standard chain will use MAP-POS consensus based on verifiable random function(VRF)[5]. The MAP token is issued in MAP standard chain as the native token. The key usage of MAP token is to ensure that the chain interoperation through MAP protocol is not abused. We will elaborate it in later section.

5.2.1 MAP-POS Consensus Based on VRF

Blockchain consensus algorithms are generally divided into two categories. The first category can be independently verified, such as POW and POS. The characteristic of this type of algorithm is that the rules for selecting blocks are based on cryptography. Other nodes in the cluster can independently verify blocks' validity without additional information. On the contrary, the other type of consensus cannot be independently verified, such as DPoS, DPoW (Delegated Proof of Work). This type of consensus algorithm needs to select a small set of nodes from a large set of nodes as delegates, and then the small set of nodes will utilize Byzantine Fault-tolerant Algorithms for blocks to reach consensus. All other nodes must additionally obtain the consensus information from the small set of nodes to verify the validity of the block. The advantages of the first type of independently verifiable consensus algorithms are the high degree of decentralization and security, but the performance of the first type of algorithms is relatively poor. The second type of non-independently verifiable consensus algorithm has the advantage of high performance, but the security and decentralization are relatively weak due to the selection process of the small set and the vulnerability of the small set itself.

Our MAP standard chain needs to decouple consensus and state transition, and provide some ecologically shared data for other chains to use. Therefore, the demand for the security and decentralization of the consensus algorithm will be higher than the demand for scalability. Therefore, our consensus algorithm will be a consensus algorithm that can be independently verified. We have two initial choices, PoW and PoS. However, after comparison and consideration, we decided to choose the PoS instead of the PoW as the consensus engine of the MAP protocol. The main reasons are as follows:

(1) PoW requires the support of computing power to maintain its security. However, at present, a large amount of computing power is concentrated on Bitcoin. If the new PoW public chain cannot obtain enough computing power, it cannot resist 51% attacks.

(2) PoW-type consensus will perform a large number of hash calculations to obtain the final consensus, which will waste a massive amount of energy (except for ensuring the safety of the public chain, there are no other benefits of those energy consumption)

(3) The PoW consensus also heavily depends on hardware equipment. The replacement of mining machines will lead to unfair competition among nodes, which will cause more centralization. This has already been proved by some monopoly of mining pools.

In the early design of the PoS consensus algorithm, it was simply to calculate mining probability based on currency holding time and currency holding volume as the weight. However, the blockchains based on this PoS consensus would be vulnerable under "long-range attack" or "nothing at stake attack". MAP standard chain would use an original POS consensus which is robust with these two attacks. We call it MAP-POS. which is designed based on Verifiable Random Function (VRF)[5].

We first introduce VRF. VRF is a pseudo-random function that provides publicly verifiable proofs of its outputs' correctness. VRF has a triple of algorithms Keygen, Evaluate, and Verify. Keygen would take a random number as input and generate a pair of keys (private key SK and public key PK). Evaluate will take private key and message(M) as input and output a pseudo-random output number(Y) and a zero-knowledge proof ρ. Verify would take PK, M, Y, and ρ as input and return true if all information matches. Thus

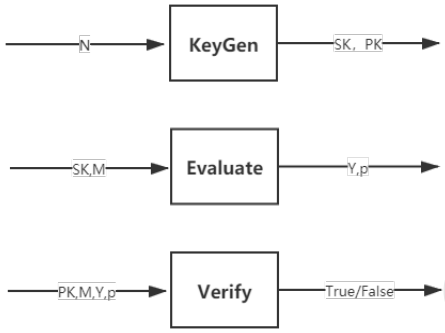

Figure 4: VRF's triple of algorithms

we can use Verify to check that the pseudo-random number is indeed created by the owner of the public key.

In MAP-POS consensus, the block is generated in epoch-based. Each epoch would be divided into several slots. Each slot would be assigned to some candidate nodes for block generation(the number of candidates is not deterministic).

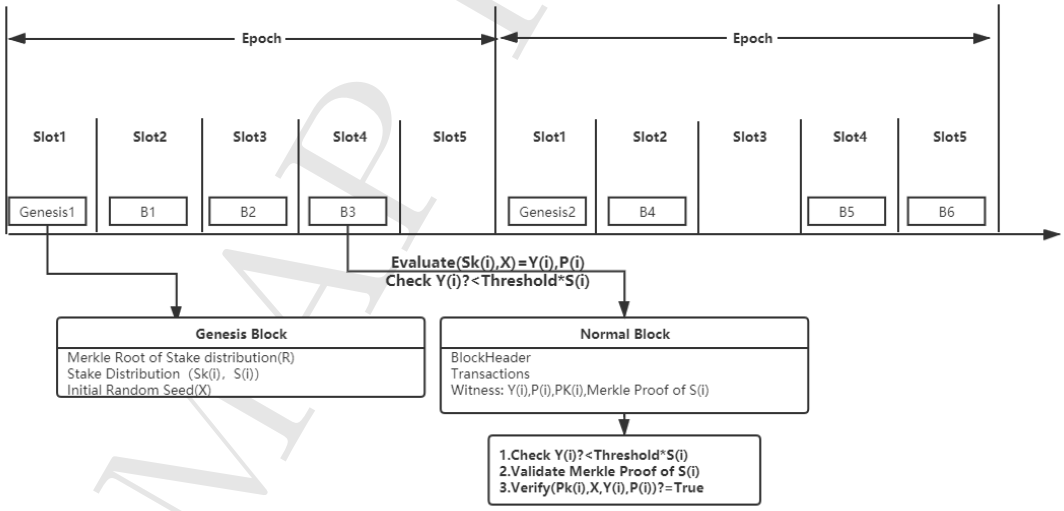

Figure 5: MAP-POS consensus design

For each epoch, the first block, called genesis block, was derived by each

node locally based on the information from the last epoch. It has two essential parts: the stake distribution and an initial random seed(X). The stake distribution, which records the public keys of all candidates and its stake portion, would be stored in a Merkle tree. The initial random seed(X) is derived from the pseudo-random numbers of blocks from the last epoch. We will elaborate on this process later.

And the rest blocks of the epoch is a normal block. These blocks are generated by some candidates by the following rules: Each candidate uses the initial random seed(X) and slot number generates a message(X(i)). And then he would use this message(X(i)) and his private key(Sk(i)) as an input of the evaluate function, and it would output a pseudo-random number Y(i) and a zero-knowledge proof $\rho(i)$. The candidate thus can check the qualification for the block generation of this slot: the candidate is qualified if the pseudo-random number Y(i) is smaller than the production of his stake portion S(i) and a predefined threshold T. The candidate only broadcast the block in the slot which he is qualified for. The block is constructed by three parts: the block header, transaction list, and witness. The block header would contain some core information of the block, such as parent hash, slot number, block height, root of the transaction tree, root of the state tree, etc. The transaction list contained all the transaction which be included in this block. The witness part would include the pseudo-random number Y(i), the zero-knowledge proof $\rho(i)$, the public key(Pk(i)) of the block creator and a Merkle proof of the stake portion(S(i)) of the block creator. Any nodes can verify the validity of this block by the following process: first, he need check the pseudo-random number Y(i) in withness is indeed below the production of the stake portion S(i) and the predefined threshold T. Then he checks that if the Merkle proof of the stake portion(S(i)) is valid. Finally, he would use the verify function of VRF to check that public key Pk(i), message X(i), pseudo-random number Y(i), and zero-knowledge proof $\rho(i)$ are consistent. All the information he needed for checking is included in the genesis block and this normal block, thus this algorithm is individually verifiable.

MAP-PoS consensus protocol has the following characteristics:

(1) There is no miner monopoly. To return the basis of the original intention of Satoshi Nakamoto's PoW consensus design, in which to ensure that all nodes participating in the consensus have a fair go.

(2) Regardless of the number of nodes participating in the consensus throughout the network, the amount of calculation required would not be too

high as to prevent waste of resources like PoW.

(3) The protocol is self-verifiable. It means that each participant can check the validity of any block individually to ensure that the block miner has spent a certain amount of stake uniquely for this block. This protocol is compatible with MAP protocol.

5.3 Chain Gateway ID Specifications and Cross-Chain Data Stream Fee

In MAP ecosystem, a unified Chain Gateway ID Specifications must be formulated to identify different blockchains under the regulation of MAP protocol. Each chain will be assigned a unique Chain Gateway ID. The role of this Chain Gateway ID is similar to IP address, to locate and identify a chain in the MAP ecosystem. When transmitting information that requires signatures, Chain Gateway ID needs to be included in the signature data to prevent repeated transmission attacks.

The Chain Gateway ID is in this form: X-Y-Z. X represent the level 1 gateway, Y represent the level 2 gateway and Z represent the level 3 gateway. Those IDs with same prefix are in the same sub-network. For example, 1-2-0 and 1-3-0 are in the same subnetwork which managed by 1-0-0. Also 1-2-2 and 1-2-5 are in the same subnetwork which managed by 1-2-0.

In the early stage of MAP ecosystem, there will be a elected committee to manage the distribution of level 1 Chain Gateway ID. The elected committee need to stake certain amount of MAP token to prevent cheating. The committee needs to review the public chain projects that apply to join this ecosystem, and the public chain that passes the review will be assigned a standardized Chain Gateway ID and updated into the contract. The contract only has the function of assigning standardized Chain Gateway ID, and does not have other centralized management functions. The committee would conduct the distribution of level 1 Chain Gateway ID. And low level Chain Gateway ID is distributed by the corresponding level 1 Chain Gateway ID owner. If any low level Chain Gateway ID owner misbehave, the corresponding high level Chain Gateway ID owner would be slashed.

Also we need to mention here, those public chains that are not assigned standardized Chain Gateway ID can also communicate with other public chains using the MAP protocol, however, their security cannot be guaranteed.

In this case, we do not recommend interoperating with public chains that do not have standardized Chain Gateway ID.

Besides committee staking, MAP token also would be used to pay the Cross-Chain Data Stream Fee among different level 1 subnetwork. This means if a crosschain message is sent from chain 1-2-4 to chain 2-3-5, they need to pay fees using MAP token as it's a cross chain request from 1-0-0 network to 2-0-0 network. The low level cross chain fee mechanism can be design by high level Chain Gateway ID holder. The reason for the existence of Cross-Chain Data Stream Fee is to avoid DOS attack. If there is no cross chain stream fee, one chain can flood another chain by keep sending cross-chain request without cost.

5.4 MAP protocol Economic Model

MAP is the native token issued by the Map Standard Chain. The initial total amount is 10 billion. With the following application scenarios to help to improve the intrinsic value of MAP protocol as well as promote the development of the MAP protocol.

(1) The Map standard chain uses a POS mechanism, and nodes need to stake MAP Token to obtain mining rights. Also the POS miners will reward by transaction fee in form of MAP token.

(2) The public chain or consortium chain using the MAP protocol requires the MAP protocol to allocate the Chain Gatewat ID. Each Chain Gatewat ID is unique in the MAP Ecosystem. ID adopts the form of pledged MAP Token to obtain an ID. As more and more public chains use the MAP protocol for chain interoperation, the demand for MAP token would be increased as well.

(3) The entity needs to stake MAP token to be eligible for governance committee election.

(4) MAP token is used for paying the Cross-Chain Data Stream Fee among different level 1 subnetwork.

(5) MAP tokens would be rewarded as incentives for Ecosystem contributor who improve MAP protocol, build Dapps and expanding technical or user communities.

Under the premise of developing and expanding the MAP protocol, as the project develops and evolves, the above scenarios will be increased or reduced.

6 Application Scenario

6.1 MAP Bridge

MAP bridge is a bridge chain in Polkadot ecology that connects Polkadot and Ethereum based on MAP protocol. It has received the Polkadot Web3 foundation grant support.

Unlike other bridge solution in Polakdot, MAP bridge is a relayer-free brigde in Polkadot ecology. It can achieve real-time chain interoperability of logarithmic space complexity based on cryptographic verification instead of trust on relayer group. There are three advantage of MAP bridge:

(1) MAP bridge can verifies the legitimacy of cross-chain data streams through ULVP protocol. The verification data is cryptography evidence, can be verified locally by any node.

(2) The data stream exchanged during cross chain communication growth with a logarithmic speed compare to the length of the chain. In other words, for the chain with length N, the length of cross-chain data stream is only logN.

(3) MAP bridge will implement the node discovery and data transfer for P2P networks of different blockchain,to achieve real-time chain interoperation

6.2 Cross-chain DEX

MAP protocol can provide automated token exchange among different blockchain systems.

First,the state from DEXs(like Uniswap) in different blockchain systems can be captured by MAP Oracle. Next M-DEX deployed in MAP Standard Chain can digest the data from MAP oracle and deal with the cross-chain token swap. Last, MAP feeder would send the cross-chain token swap transaction to target DEXs in different blockchains. The procdure can be viewed in Figure 6. Some more details of cross-chain token swap can be found in Appendix D.

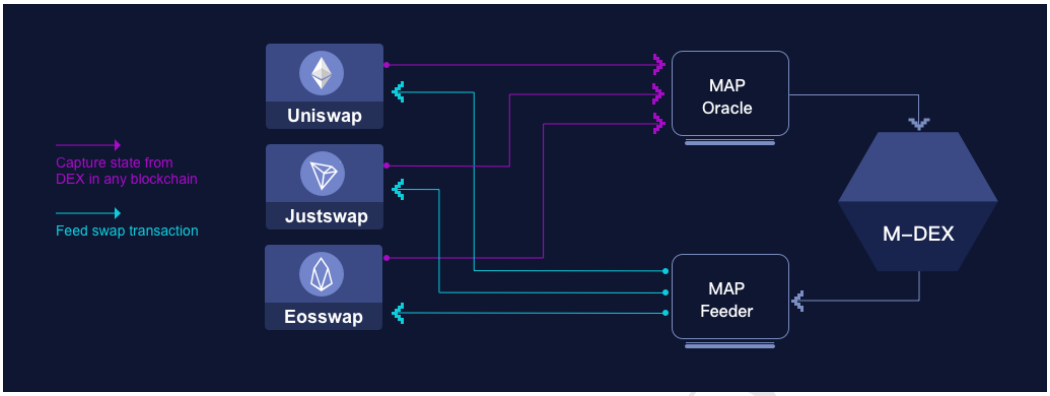

Figure 6: Cross-chain DEX build upon MAP protocol

6.3 Defi Data Oracle

MAP protocol can provide onchain defi data(for example, token value locked in Uniswap) directly to smart scripts on any blockchain or off-chain data analyzer.

First,the raw defi data from all centralized and decentralized exchanges in different blockchain system can be captured by MAP Defi Oracle. Next, these raw defi data can be relay to reference smart script in blockchain systems and off-chain databases. Last,any smart script can refer these defi data from reference smart script. Also, user can analyze the defi data from the off-chain databases for decentralized application.

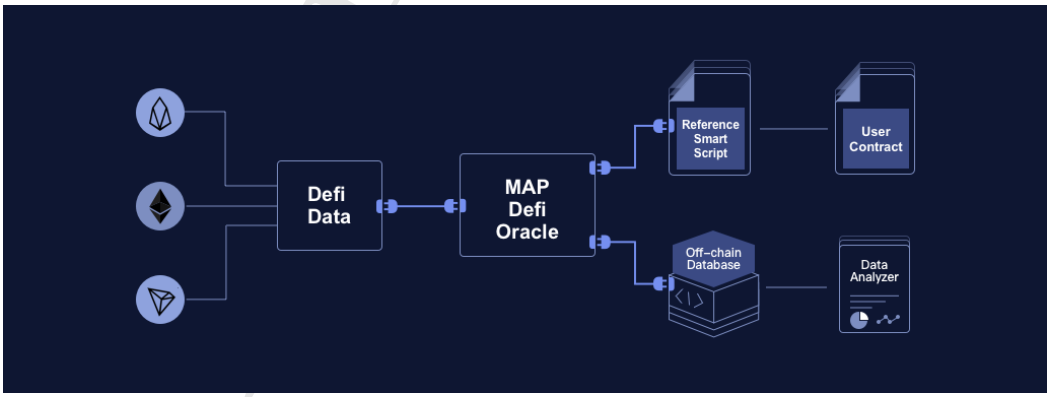

Figure 7: Defi data oracle using MAP protocol

6.4 IPFS and Privacy computing

MAP protocol also can provide support for IPFS storage coupled with privacy computing functionality. IPFS is a hypermedia transmission protocol based on content addressing and peer-to-peer network. It allows participants in the network to store, request, and transmit verifiable data to each other. Privacy preserving database could be built on IPFS network. Authorized access to this privacy perserving database and the privacy transmission function can be realized. Atlas embeded with MAP protocol is a privacy computing layer2 of filecoin, which is the incentive mechanism of IPFS. Any other blockchain could interoperate with Atlas to access to the privacy perserving database on IPFS.

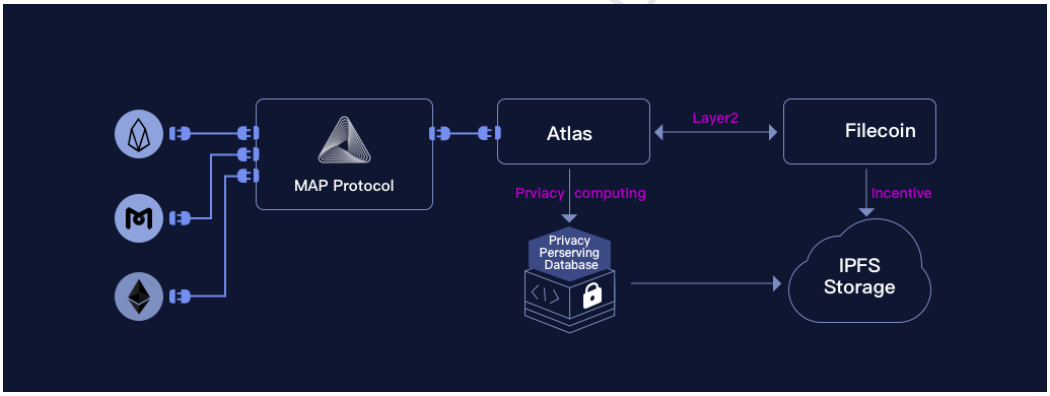

Figure 8: IPFS and privacy computing using MAP protocol

References

[1] Aggelos Kiayias, Nikolaos Lamprou, and Aikaterini-Panagiota Stouka. Proofs of proofs-of-work with sublinear complexity. International Conference on Financial Cryptography and Data Security, pages 61-78. Springer, 2016.

[2] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. Non-interactive proofs of proof-of-work, 2017.

[3] Goodrich, M., Mitzenmacher, M.: Invertible bloom lookup tables. In: Conf. on Comm., Control, and Computing. pp. 792-799 (Sept 2011)

[4] Benedikt Bünz, Lucianna Kiffer, Loi Luu, and Mahdi Zamani. FlyClient: Super-Light Clients for Cryptocurrencies,2019

[5] Micali Silvio, Rabin Michael O, Vadhan Salil P. Verifiable random functions. Proceedings of the 40th IEEE Symposium on Foundations of Computer Science, 120-130, 1999

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

A Categories of consensus

There are dozens of different consensus mechanisms in blockchain design. We list some them below in our way:

POW: Starting with Bitcoin, the POW consensus algorithm can be regarded as the longest time-tested consensus algorithm in the blockchain field. It provided the strongest degree of decentralization and provide a safe consensus base for the entire ecology. Other chains in the ecosystem can directly use ULVP to read the shared data on POW public chains. However, there is a big problem with the consensus of the POW type. A large amount of computing power is concentrated on the Bitcoin network, and the security of other POW type public chains cannot be guaranteed by sufficient computing power.

POS: The POS consensus algorithm is a supplement to the POW consensus algorithm, which first appeared in the PPCoin project. It was born to solve the waste of POW resources. Ethereum 2.0 used Casper which is also a kind of POS consensus algorithm. In the future, the underlying public chain of ecology may use it and provide shared global data for use by other application public chains in the ecosystem, just like the POW-type public chain. The shared data on it can also be directly read by other chains in the ecosystem through ULVP. The MAP Standard Chain we mentioned earlier will use POS consensus based on Ouroboros protocol.

Other Proof of X(POX): This includes Proof of Space, Proof of Elapsed-Time, and various other POX consensuses. These consensuses have a common feature, that is, they can be independently verified by any node without additional information (the same is true for POW and POS, but it was discussed separately above because of its particularity). These types of consensus are suitable for some public chains with special needs because of their

characteristics. At the same time, their data can also be read by other chains in the ecosystem through ULVP.

Delegated Proof of X(DPOX): This includes various types of delegated public chains, including Delegated proof of work and Delegated proof of stake. The characteristic of these consensuses is that some delegates need to be elected, and then these delegates perform operations such as consensus generation. However, these election processes may be off-chain or unpredictable, and therefore cannot be directly verified by other nodes. However, because the size of the cluster participating in the consensus has shrunk, such a public chain has strong scalability and is very suitable for some practical application scenarios. The data of these chains cannot be directly read by other chains in the ecosystem through ULVP. At this time, these public chains need to be anchored on some other type of public chain and various core security data (such as representative's vote information, representative's identification, etc.) on the anchor public chain needed to be updated regularly. At this time, the security of interoperation requires additional consideration of the security of the anchor public chain to obtain these core security data.

B Design of ULVP

If a node wants to query a predicate relate to one blockchain using SPV, he needs to download a header chain as we mentioned above. Then, he can request some randomly selected full node to relay the corresponding Merkle branch proof. After that, he can verify the Merkle branch proof based on his local stored header chain or valid header from relayer. However for ULVP, the first step is to validate one target block that exists in the blockchain using MMR and random sampling. The second step is similar to SPV, the verifier needs to get the Merkle branch proof corresponding one predicate to relate to the target block and verify this proof. I will elaborate the first step and show why this is called Ultra Light Verification Protocol.

Firstly, I want to introduce a new data structure named Merkle Mountain Range(MMR). This is a variation of the Merkle tree. The leaf nodes of MMR are block headers. Figure 9 shows the MMR construct by 12 block headers(from B0 to B11). The root of this MMR would be store in B12. Then, it is easy to provide Merkle branch proof corresponding any block from B0 to B11 if B12 was given. For example, if one verifier node needs to verify B5(blue marked) which was contained in the blockchain whose tail

block is B12, then the prover can generate the Merkle branch proof(green marked) and relay it to the verifier. The verifier node can recover the root based on the Merkle branch proof and compare the recovering root to the root stored in B12. If the two roots are the same, then it means B5 is indeed in this blockchain, otherwise means B5 is not.

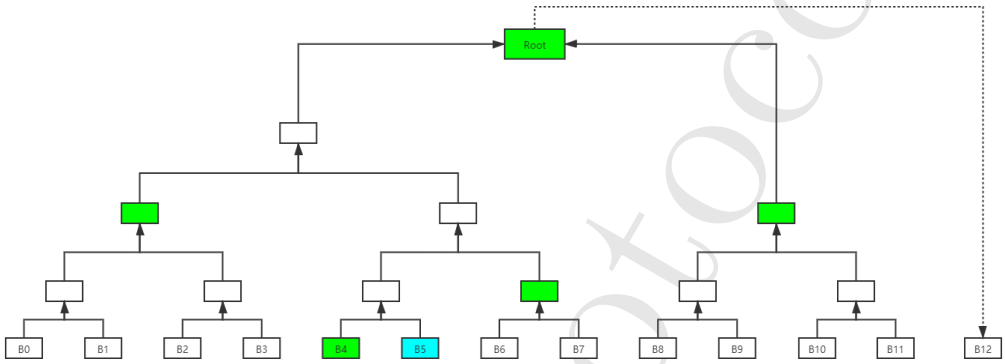

Figure 9: Merkle mountain range construct by block headers

The first step of ULVP is to validate one target block that exists in the blockchain. Thanks to MMR introduced above, this problem is equivalent to prove one given tail block of a blockchain is indeed the tail block of the main network. For any POX-based blockchain,it would allow any node to verify the validity of each block individually ensuring that the block creator has spent (or burnt) a certain amount of a resource uniquely for this block. For POW, the resource is hash power, and for POS, the resource is stake. Under the assumption that adversary cannot control more than 50% resource of the whole network. This means that the adversary can not craft more blocks than honest majority. Thus if the adversary want to cheat a verifier, he need to create some malicious blocks. ULVP would design a sampling algorithm to make sure the verifier detects malicious blocks with overwhelming probability. Thus the verifier could use a small sampling set of block headers to check the validity of the tail block of the main network.

For example, for POW-based blockchain the resource used for validity is hash power.Let us denote the advantage of the honest node is α. This means that the adversary can generate α blocks when an honest party generates one block. Let us see Figure 10: we set $\alpha = 0.4$, then, when an honest party generates 5 blocks, the adversary can mine 2 blocks(yellow blocks). If he

wants to cheat the verifier, he will need to fill 4 malicious blocks(blue blocks) to exceed the honest chain. If the algorithm can sample enough so that the verifier can detect these malicious blocks, then this algorithm can be used to prove the tail block one prover claim is indeed filled by all valid-mined blocks. Thus, the verifier can compare the length of all the valid candidate tail blocks and simply select the longest one. This would guarantee that the verifier picks the exact same tail block as the honest majority picks.

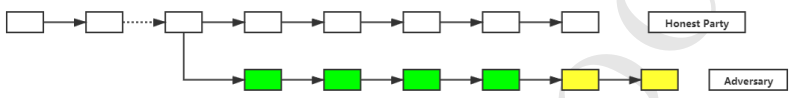

Figure 10: Adversary fill some invalid blocks to exceed honest party

Now the problem becomes how to design a sampling algorithm to make sure the verifier detects malicious blocks with overwhelming probability under the POW set. It can prove that for any sampling distribution over the blocks defined by a non-increasing PDF f, there exists another distribution with an equal or higher probability of catching the adversary. Thus, the optimal distribution over the blocks must be defined by an increasing PDF. Furthermore, Bunz et al [4] prove that the optimal distribution defined by the PDF $g(x) = \frac{1}{(x-1)ln\delta}$ maximizes the probability of catching any adversary that optimizes the placement of invalid blocks. And if we use this optimal distribution, we can sample $\lambda * log_{\frac{1}{\alpha}} n$ blocks to bound the failure probability of catching the malicious blocks below $2^{-\lambda}$, where n is the length of the blockchain.

C Details of SMART

Based on SMART implementation, MAP can provide the scalability of the interactive chain and introduce other assets on the chain. The payment system built on SMART has tens of thousands of TPS throughput in the real network environment, and the confirmation time can be reduced to 2-3 seconds.

MAP-VM is a virtual machine running contract code in SMART. MAP-VM is a high-performance virtual machine based on WebAssembly. WebAssembly is a new generation code format that can be run directly in the

browser. Compared with the EVM used by Ethereum, MAP-VM can run code at a speed close to the native computer hardware speed, while also providing a more flexible contract interface.

Compared with V8, Chakra, Spidermonkey, and other execution engines, MAP-VM is designed for consensus execution and enhances various features to adapt to contract code execution. Since the WebAssembly open standard is still developing, many new features are being added to the standard, and MAP-VM uses a subset of WebAssembly.

In addition to implementing the WebAssembly standard specification, MAP-VM provides the following enhanced features:

For soft floating-point arithmetic, the operation of the contract depends on the execution environment of a deterministic operation. MAP-VM uses soft floating-point arithmetic. Compared with the floating-point arithmetic of the local CPU instruction set, it can provide absolute consistency.

Deterministic execution, the contract code has a definite and consistent running result in different environments, and all abnormal call and operation errors are accurately handled. Compared with v8, MAP-VM, Spidermonkey's highly optimized JIT engine can guarantee the absolute consistency of the calculation process and calculation results.

Efficient measurement. In MAP protocol's consensus, both computing and storage are scarce resources. MAP-VM uses the gas mechanism similar to Ethereum. It can accurately calculate the computing resource consumption of the code while the wasm is executed, without affecting the overall performance.

MAP-VM is a stack-based virtual machine that can provide function library export and import custom function interfaces like JavaScript. MAP-VM will import a standard set of host functions, defined as MEI (MAP Environment Interface). MEI provides basic APIs to support the operation of wasm, including algorithm signature, digest operation, block data, state storage, and other operations.

In SMART, Delta is used to implement MAP's runtime code. Delta contract language is a JavaScript-like programming language. Delta's language features are specifically designed for contract platforms, and the ABI interface is closer to WebAssembly. Unlike Solidity on Ethereum, the Delta language supports more native base types, which can provide higher performance. The Delta contract code is directly compiled into the wasm format. Compared with the EVM contract compiled by Solidity, the Delta code has a ten-fold improvement in terms of calculation performance, providing greater

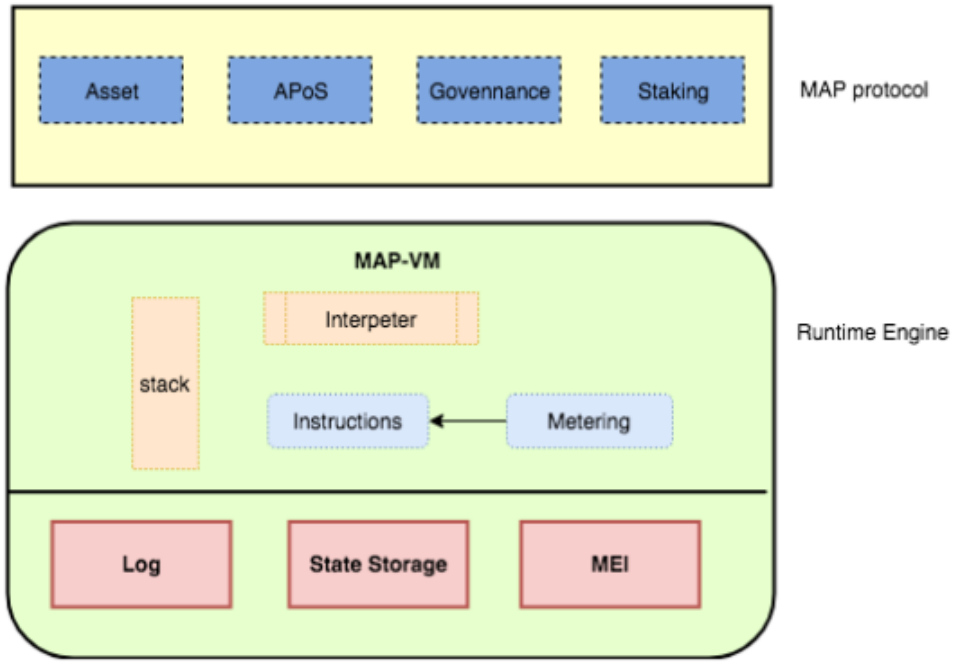

Figure 11: SMART SYSTEM

throughput for transaction processing.

The Delta language is based on static typing and is syntactically similar to JavaScript. In order to adapt to the contract operating environment, Delta has a built-in State statement and Event function, which can directly link the MAP ABI defined by SMART.

The most important improvement in the Delta language is support for the global state. In Solidity, you need to define variables and mapping and other modifiers in the contract to declare the storage structure in the global state. Delta will use native types to support storage type declarations providing safer type operations for states. Contract developers can more intuitively define operations on global resources and design contract security codes.

In the SMART architecture, the runtime protocol can be written and implemented in Rust or Delta language and can be compiled into the underlying wasm blob by the compiler. MAP-VM loads the compiled wasm code and waits for transaction execution after initializing the global environment. The runtime code running on MAP-VM can export a series of application

interfaces to providing functions such as consensus protocol, multi-currency payment, and community voting. MAP protocol defines the following basic agreements:

(1) staking

(2) asset

(3) government

In SMART, a series of MABI calls are defined as messages. The messages in MAP-VM are different from the transactions in Ethereum. In Ethereum, account addresses are explicitly divided into external accounts and contract accounts. When an ordinary transfer is initiated, the sender must specify the Value field of the amount, and if it is a contract call, the Input field of the contract must be specified. In MABI calls, all transactions use a unified message format, which is eventually decoded into WebAssembly code for execution, no distinction between transfer calls and contract calls.

In addition to the current state of the account, MAP protocol will separately store the execution process of the transaction and save the data in the Log. Log supports quick query for data indexing. When the user wants to determine that a transaction is executed correctly and obtains the expected state, he can query the transaction log according to the index through the transaction hash to extract the execution data from the stored data.

The transaction in MAP protocol is a general signed message data. Unlike Ethereum, the transaction structure in MAP protocol does not have a separate value field for the transfer amount but instead uses a similar system contract to provide the transfer function. When the user initiates a transfer, the sender needs to use a signature package contract to call the data, and then send it to the p2p network. The sender will not get any return value, but a unique transaction hash. After the transaction is packaged by consensus and executed by SMART, users can query the account status after the transaction.

A transaction in MAP protocol can be represented by a state transfer function:

$$S' = SMART - Exec(account, func, input, S). \tag{1}$$

S is the state of the account before the transaction, the input is encoded contract call data, func is the signature of the function called by the contract, and account is the sender can support multiple signature formats.

MAP prtocol can get the sender information from the signing data, and after decoding the transaction data to get the function signature and input data parameter. After the transaction data is verified and sorted, it is executed in MAP-VM to change the user's status. MAP's account system is different from Bitcoin and Ethereum. MAP's account is not the public key hash of a certain signature algorithm, but an account ID that supports multiple signature algorithm formats. Thanks to the account format of the multi-signature algorithm, MAP can natively support a variety of crypto assets with different signatures to realize multi-currency payment scenarios.

MAP-VM will accumulate gas consumption during the execution of transactions. VM has built-in WebAssembly measurement, which can accumulate all instruction calculations, stack occupation, and resource consumption of state storage. When insufficient gas and execution errors occur, the VM will deduct the gas consumption and return the state to the account state before the transaction.

In Bitcoin and Ethereum, when the community wants to modify the consensus protocol, it needs to provide an incompatible version. After the miner agrees and adopts it, a hard fork is initiated at a specific height. If the community disagrees with the hard fork, users will have to face governance issues such as community splits and replay attacks. Benefiting from SMART architecture design and on-chain governance, each running node can initiate a new runtime proposal without a hard fork, and after the vote is passed, the protocol consensus can be upgraded on the designated height chain.

D Details of cross-chain atomic swap based on MAP

The detail design of the cross-chain atomic swap is listed below Figure 12:

Suppose A want to exchange 1 ETH to 10EOS with B,

1. A and B would first lock their coins in two locked contract(contractA and contractB). These two contract can be unlocked either with a hashlock or with timelock.

2. After this is settled, B can deployed a swap contract(contractM) in MAP standard chain which can take the key of the hashlock as input.

3. If B decide to complete the swap, he need to relay the key to swap contract(contractM) before the timelock expired. MAP protocol would deal with the swap. If B regret,he can wait the timelock expired and request the swap contract to refund A and B with their initial deposit.

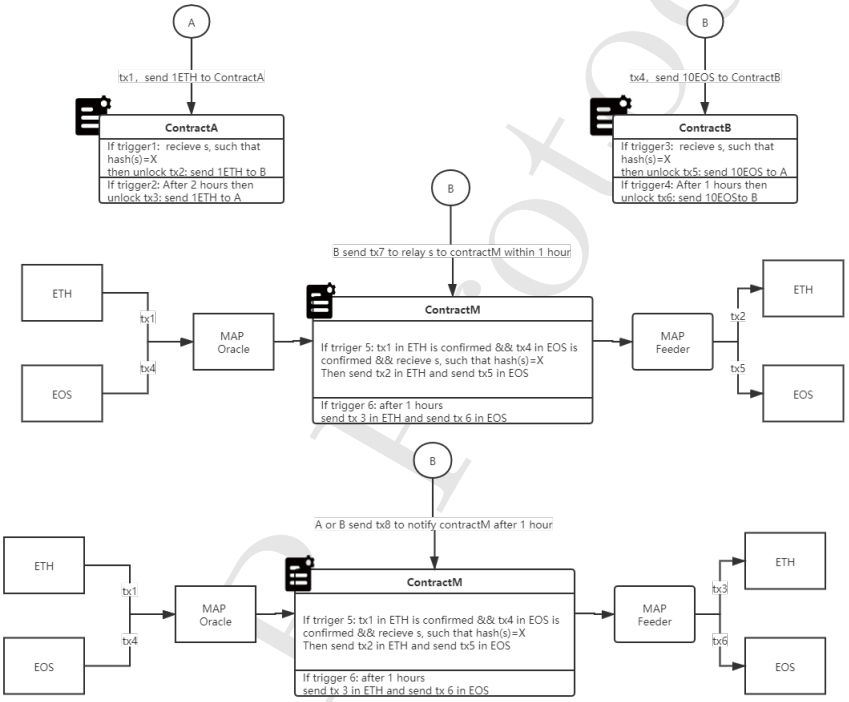

Figure 12: Cross-chain swap design