SCDO – A Scalable, Efficient, and Decentralized Blockchain System

The SCDO Project Team, scdo.pro

Aug 2020

1. Introduction

SCDO is a new peer-to-peer public blockchain system which allows decentralized value transfer in a secure and efficient manner. It has an account-balance model, supports general smart contracts and is EVM compatible. It has an original ZPoW consensus algorithm that guarantees the safety of the SCDO mainnet. It provides a sharding framework and a STEM subchain protocol to make the whole SCDO system scalable. SCDO's ecology consists of SCDO mainnet, subchains, SDKs, applications and the global user/developer communities. The vision of SCDO is to build up such a decentralized ecology with powerful infrastructure.

Since the seminal paper [1] by Satoshi Nakamoto creates a purely peer-to-peer version of digital cash system – now the well-known Bitcoin, many different digital token systems are invented, such as Ethereum[2], Zcash[3], Algorand[4], etc. Among them, Ethereum introduces a completed smart contract framework (written in a Turing-complete language) into digital cash systems. A smart contract is a piece of computer code which can be executed when certain conditions are met. The invention of smart contracts establishes as the cornerstone for blockchain applications. The critical goals that these digital cash systems try to realize are security, efficiency and decentralization. More precisely, a secure blockchain guarantees that the ownership and transfer of digital cash work as expected; an efficient blockchain guarantees that a transaction is fulfilled in a reasonable time; a decentralized blockchain allows a peer-to-peer transaction to be completed without trusted third parties in the middle, like banks, payment centers in the current centralized financial payment system. The three goals are often declared to form an impossible triangle, that is at most two of them can be met in a single blockchain system. Application potential of blockchain technology is vast. Another critical problem is how a single blockchain system supports heterogeneous applications from different industries such as finance, communications, cloud computing, manufacturing, etc.

In this white paper, we present SCDO's solutions to the problems mentioned above.

For efficiency: SCDO provides a sharding framework. The SCDO mainnet can be configured to have as many shards as needed. However, from a practical viewpoint, the SCDO mainnet has 4 shards. The sender and recipient of a peer-to-peer transaction can belong to different shards or to the same shard.

For security and decentralization: apart from providing a robust peer-to-peer network and private key and public key framework, a new proof of work consensus algorithm ZPoW is invented. ZPoW focuses on scientific computation rather than hash speed. ZPoW is CPU friendly. GPU loses its advantage when it executes ZPoW. It uses less ram and more intense CPU computation. It is hard to design an ASIC chip for ZPoW as we can see in the near

future. These special characteristics make the SCDO mainnet less vulnerable to the dominance of a few miners so that the SCDO mainnet is stronger in security and levels the play field for miners.

For heterogeneous applications: SCDO mainnet supports various applications itself. What's more, SCDO proposes and implements the Stem Subchain Protocol. It can support applications in different fields. For example, if an application needs high throughput and is less concerned on decentralization, then a smart contract, named the stem smart contract is registered on the SCDO mainnet (or root chain) which acts as the "umbilical cord" between the root chain and the subchain. The root chain provides the security for the subchain and the subchain can choose its consensus algorithm, for example a PoS or a PBFT consensus algorithm, which has high throughput.

In the next few sections, we will describe each solution mentioned above.

2. Notations and terminologies

In this section we state a few notations which are used in later sections.

Public and private keys: a pair of integers of fixed length. The pair is created by an elliptic-curve function. The public key is used as an address to receive transactions. The private key is used to sign a transaction sent to the recipient.

Transaction: a fixed number of data fields. Each field records a piece of information of a peer-to-peer operation, such as the number of tokens to be sent, the receiver's public key, the sender's signature, and the other miscellaneous information. as illustrated in Fig 1.

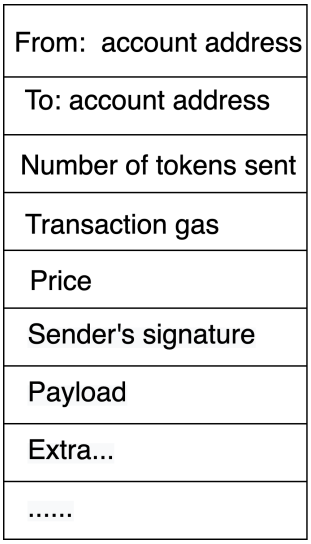

Fig. 1 An illustration of a transaction

Debt: a cross-shard transaction. It has the same structure as the transaction defined above.

Block: includes four metadata parts—header hash, header, transaction list (tx list), and debt list. The header contains basic information about the block, such as the previous block header hash, the creator of the block, the number of blocks recorded in the SCDO mainnet so far, the current timestamp, and the root hash of the transaction Merkle tree. Tx list is a set of transactions to be processed. Debt list is a set of debts, mainly recording the transactions across different shards (it will be explained later). The max number of transactions and debts in a block is determined by the max block size (set to 1M bytes in the SCDO mainnet), which is about 6000. A block structure is illustrated in Fig. 2.

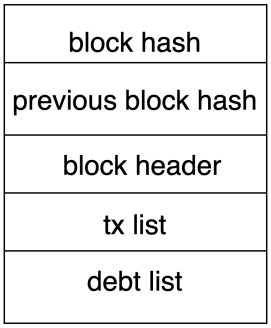

Fig. 2 An illustration of a block structure

Merkle tree: a binary tree, each intermediate node has two children. In the SCDO system, a Merkle tree is used to record transactions (debts) in such a way. Each leaf node contains the data of a transaction, and an intermediate node records the hash of its two children.

Smart contract: a piece of code executable in Ethereum Virtual Machine(EVM) [2]. It may have input and output, behaving as a function.

Account: has a pair of keys, a counter (for counting transactions), the current token balance, the contract code (it may be empty), and the internal storage for storing additional data. The counter is to guarantee that each transaction from the account is only processed once. When the contract code is empty, the account is controlled by the private key. It can send transactions/messages by signing them with the private key. Otherwise, only the contract code is activated. It is allowed to read from and write to the internal storage to send a transaction or to create another contract code.

Chain of block or blockchain: a linked list of blocks, a block can trace its previous or parent block with ease by the previous block header hash stored in the block. A blockchain is illustrated in Fig. 3.

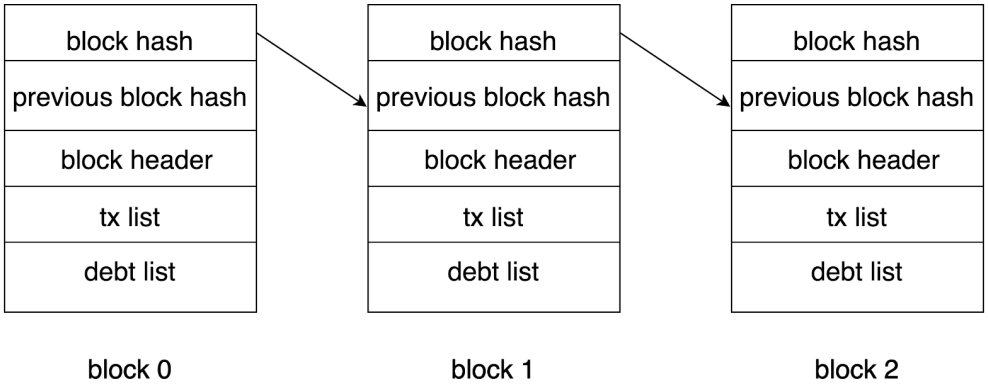

Fig. 3 A chain of blocks

Mining node (or miner, or full node): it is a node of SCDO peer to peer network. It stores all the necessary information to create a block and run ZPoW. Through ZPoW, all full nodes are racing for the privilege to record the next block in the SCDO blockchain.

Light node: a node of SCDO peer to peer network with the minimum information so that it can send/receive SCDO tokens or post a smart contract to the SCDO blockchain.

A node is either a mining node or light node. It has an account as part of itself.

Now we are ready to state the sharding framework, ZPoW consensus algorithm, and the Stem Subchain Protocol.

3. SCDO Sharding Framework

The sole aim of the SCDO sharding framework is to improve the efficiency of transactions of the SCDO mainnet. Each shard can be viewed as a blockchain on its own. The SCDO shard framework provides the basic functions such that peer-to-peer transactions can be processed efficiently and correctly.

In theory, the SCDO mainnet can have as many shards as desired. However, from a practical viewpoint the SCDO mainnet is configured to have 4 shards. This is how the four shards formed. We partition the nodes and accounts into 4 subsets. Each subset forms one shard. A function is provided to create a pair of public and private keys for a given shard. An account data structure is created with a key pair. An account can perform various transactions in the SCDO mainnet.

There are two different types of accounts in the SCDO: external accounts and contract accounts. External accounts can interact with other external accounts in any shard while contract accounts can only interact with accounts within the same shard.

For simplicity, we use one shard (the local shard) to explain how transactions are packed into blocks and then recorded.

For a transaction tx, only one of the three cases happens: (1) The sender and the recipient are in the local shard, (2) The sender is in the local shard, and the recipient is in a different shard, and (3) the recipient is in the local shard and the sender is not. For (1) and (2), tx is added to the Tx list, otherwise it is put into the Debt list of a block.

A mining node keeps and updates the following meta data:

1. A chain of blocks (called the local chain, the full chain): blocks created and confirmed by the mining nodes of the local shard so far.
2. A chain of block header hashes for each other shard (called a light chain). It plays a critical role in processing transactions among different shards.
3. A transaction Merkle Tree: it records transactions sent from the local shard.
4. A debt Merkle Tree: it records transactions sent to the local shard from the other shards.
5. A receipt Merkle Tree: it records transaction receipts. The information included in a receipt mainly is the transaction fee, the hash of the transaction, and how much “gas” (SCDO tokens) is used if the transaction is issued from a contract.
6. A list of transactions and debts received but not packed in confirmed block(s) yet.

Fig. 4 illustrates the local chain and the light chains.

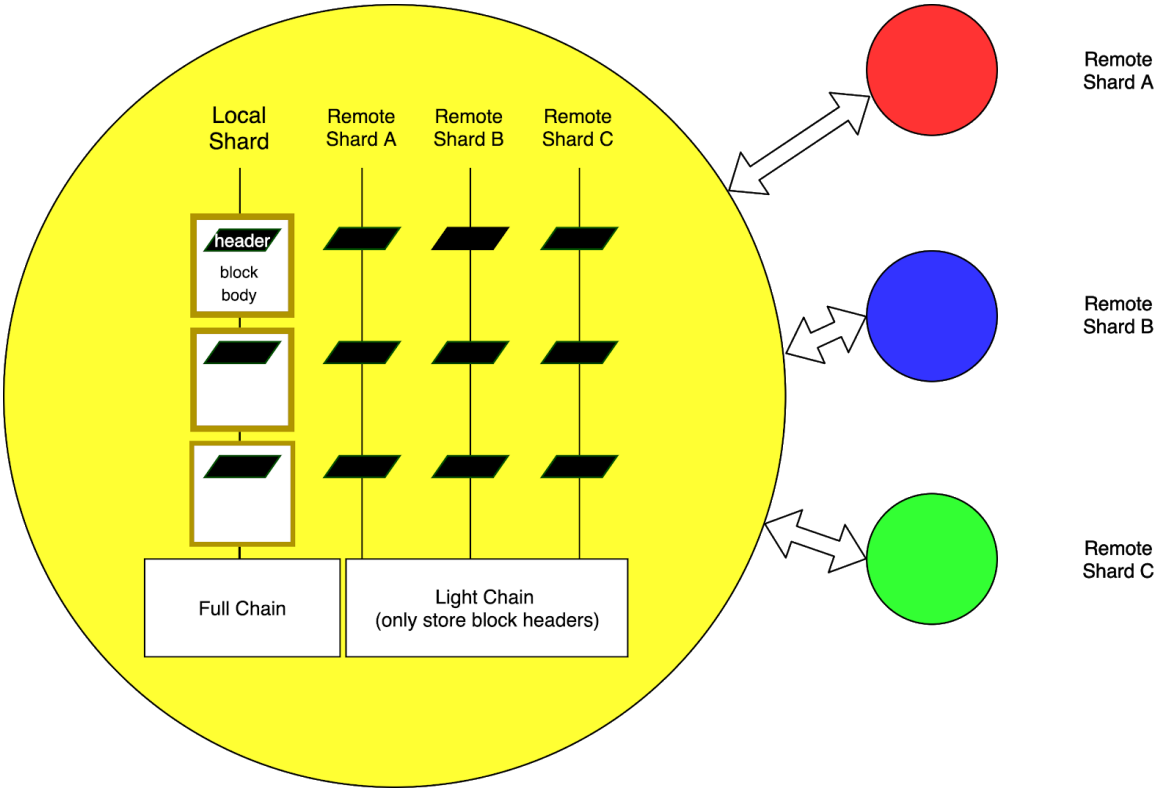

Fig. 4 Local chain and light chains illustration

To make SCDO sharding framework work, the information of a transaction tx between two shards and here between the local shard and any other shard, is recorded twice in the SCDO mainnet. tx is recorded in the Tx List in the sender's shard and in the Debt List in the recipient's shard. After a certain amount of blocks are confirmed in the sender's shard, then miners create a debt with respect to tx, put it in the outgoing debt pool, and broadcast it to the peer-to-peer network of SCDO.

When a debt reaches the recipient's shard, then it is added to the incoming debt pool. When a mining node Nd picks a debt dt from the incoming debt pool and before it adds dt to a new block, Nd sends a request to some mining nodes in the sender's shard SS to verify this debt. If this debt exists and has been contained in a confirmed block, then the block header hash blh is returned. Nd verifies that if blh is already in the light chain relating to SS. If yes, dt is added to the block. If not, it waits for the next round.

The local chain and light chains are the critical components of the SCDO sharding framework, which guarantee to record all transactions correctly.

4. SCDO Consensus Algorithm and Mining

The blockchain mining scheme of the proof of work (PoW) is usually the following. There is a consensus algorithm, say F(*). For a given block b and a nonce nc, F(b,nc) will return a computed value. If the computed value satisfies a given condition, then nc is the solution. The mining node which finds the solution first is the winner to record b. The popular PoW consensus algorithm from Bitcoin is a hash function. A mining node computes F(b,nc) by augmenting nc so that the return value satisfies a pre-given condition (difficulty). As soon as such a nonce is found, the mining node broadcasts the news to the entire network. If recognized by other mining nodes, the block b is recorded in the blockchain. Some tokens are awarded to the mining node for being the first to find the nonce.

The consensus algorithm of the SCDO mainnet belongs to the class of proof of work algorithms. The PoW of Bitcoin is the most well-known one in this class. Winning or not is dependent on the computing power of the devices running PoW algorithms. There are mainly three types of chips to run PoW algorithms: ASIC chips, GPUs and CPUs. ASIC chips are extremely fast for hash functions. GPU is very efficient for parallel computing. CPU is good for general purpose computing and very economical but is slower for hash rate and parallel computing. The challenge for PoW consensus algorithms is that a few miners with ASIC or GPU devices may dominate the entire network, which results in fewer miners and less decentralization. We keep this challenge in our mind when designing the consensus algorithm for the SCDO mainnet. The consensus algorithm which we invent and implement is named as ZPoW. It is CPU friendly. There is no obvious advantage if it runs on GPU powered devices. And there are probably no ASIC chips available in the foreseeable future.

In this section, we will state ZPoW and mention a few desired properties and leave the mathematics to another article.

The development of ZPoW has two phases. The first phase of ZPoW (ZPoW1) is an algorithm based on the computation of the determinant of random matrices. One property of this algorithm is that it cannot be implemented in parallel. GPUs do not have much advantage over CPUs. This algorithm is also difficult for ASIC design. The second phase of ZPoW (ZPoW2) will be a combination of multiple child algorithms. Miners can choose one of the child algorithms to mine blocks. ZPoW2 will use a unique dynamic mechanism to adjust the difficulties of different child algorithms so that it is almost impossible for the miners working on the same child algorithm to produce consecutive blocks. More specifically, a child algorithm which produces consecutive blocks will have a higher difficulty than other child algorithms. Meanwhile, ZPoW2 invents pre-computing proof to discourage miners from switching among the child algorithms. Pre-computing proof makes sure the miners' previous work on a child algorithm can be used to increase chances for mining future blocks. At a higher level, this innovative ZPoW2 algorithm will make it almost impossible to perform 51% attack.

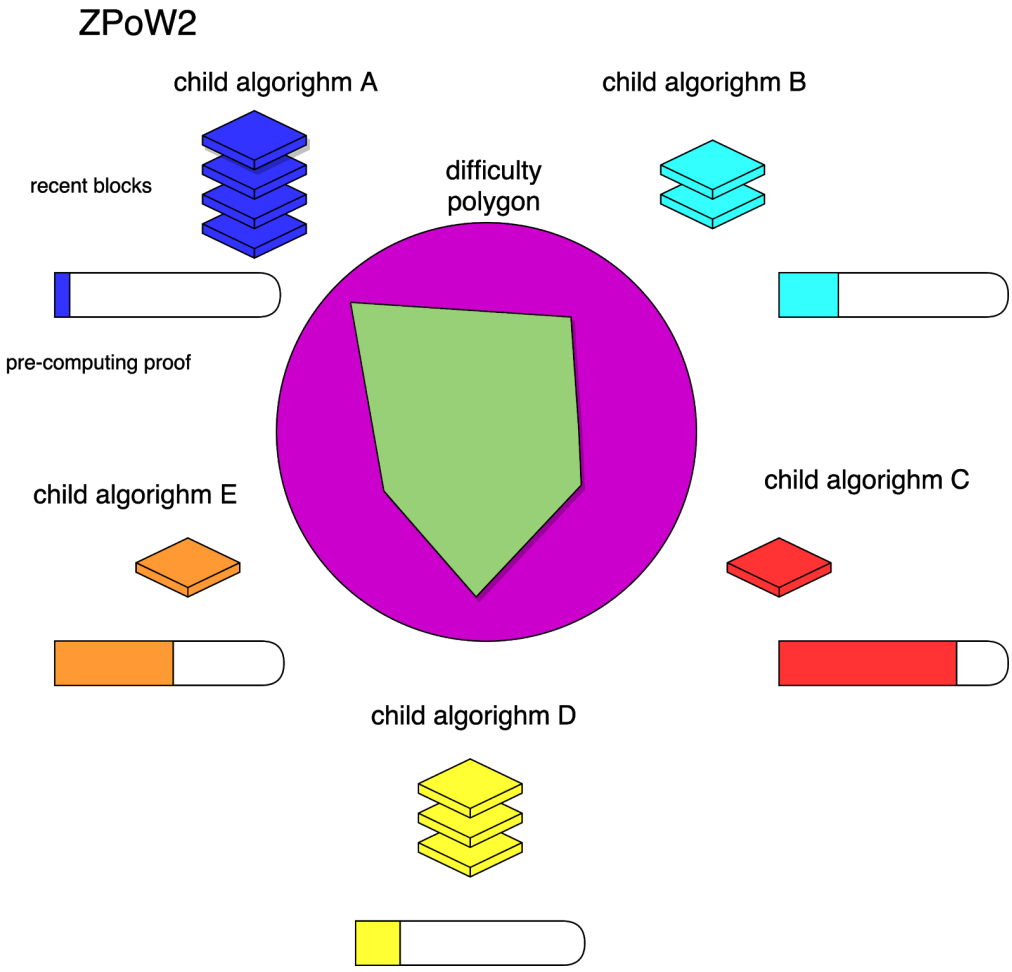

Fig. 5 An illustration of ZPoW2

The following is a brief introduction of ZPoW1:

Let CF(block_hash, nonce) be the function with two parameters which is invented in SCDO. The first is a hash string, and the second an integer. The output is an *nxn* matrix with 0,1,2

coefficients. In SCDO, n is set to be 30. One can view that CF produces a verifiable pseudo random matrix. Let det(M) denote a function to compute the determinant of a square matrix M. The definition and form of CF will be given in a separate paper.

The main characteristics of CF are: 1. It cannot be implemented in parallel. 2. Its computing process is verifiable. 3. The return value is verifiable. We are ready to state ZPoW1.

SCDO ZPoW1 Algorithm:

Input: bh, the header hash of a new block; n_0, the initial nonce (an integer); d, the difficulty (a large positive integer);

Initial Step:

$n:=n_0$;
dt:=0;

Loop Step:

dt:=det(CF(bh,n));
If dt>= d then exit the loop;
Otherwise n=n+1;

If an outside stop signal is received, the process is aborted;

Broadcast step:

Broadcast the block header hash and block header to the network.

We briefly mention here the determinant distribution P of the matrices that CF(*,*) computes. From our vast test cases that the tail part of P is a smooth curve and fits nicely with an exponential distribution with λ is around 1/1.4e+14.

Let T be the time between two consecutive blocks (or block time) in the local shard, which is the duration from computing the first block to the beginning of computing the second block. Then T roughly follows an exponential distribution with $\lambda = 1/20$. The average of T is 20 seconds.

CF(*) is designed to be a sequential process. The major portion of time of ZPoW is in computing CF(*). This limits the computing power of GPU and ASIC chips.

5. Stem Subchain Protocol-SCDO Subchain Framework

In this section, we introduce the Stem Subchain Protocol. A more detailed research article will be published in the future.

Stem subchain protocol provides a Layer-2 scalability solution for SCDO mainnet. With this protocol, SCDO mainnet (root chains) can protect the safety of funds on child chains while child chains can implement different consensus algorithms to speed up the transactions.

There is no limit on the number of child chains. This subchain protocol is an important infrastructure to scale up the SCDO system.

Stem subchain protocol aims to provide a full set of useful features. It supports account-based child chains, general smart contracts on child chains, safe and flexible deposit/exit operations, fast access to the account balance in the Stem smart contract which is the “umbilical cord”, and customizable state finality of the child chain.

The child chains of SCDO can be designed for different applications, for example:

1. Fast and private transaction channels: a group of users send transactions on the child chain and relay the updated states to the root chain;
2. A data warehouse recording important information such as logistics information: the history of the data is endorsed by SCDO root chain;
3. Decentralized finance: child chain users exchange/borrow SCDO on root chain and offer the equivalent assets on child chains;
4. Independent chains with their own tokens which are exchangeable with SCDO.

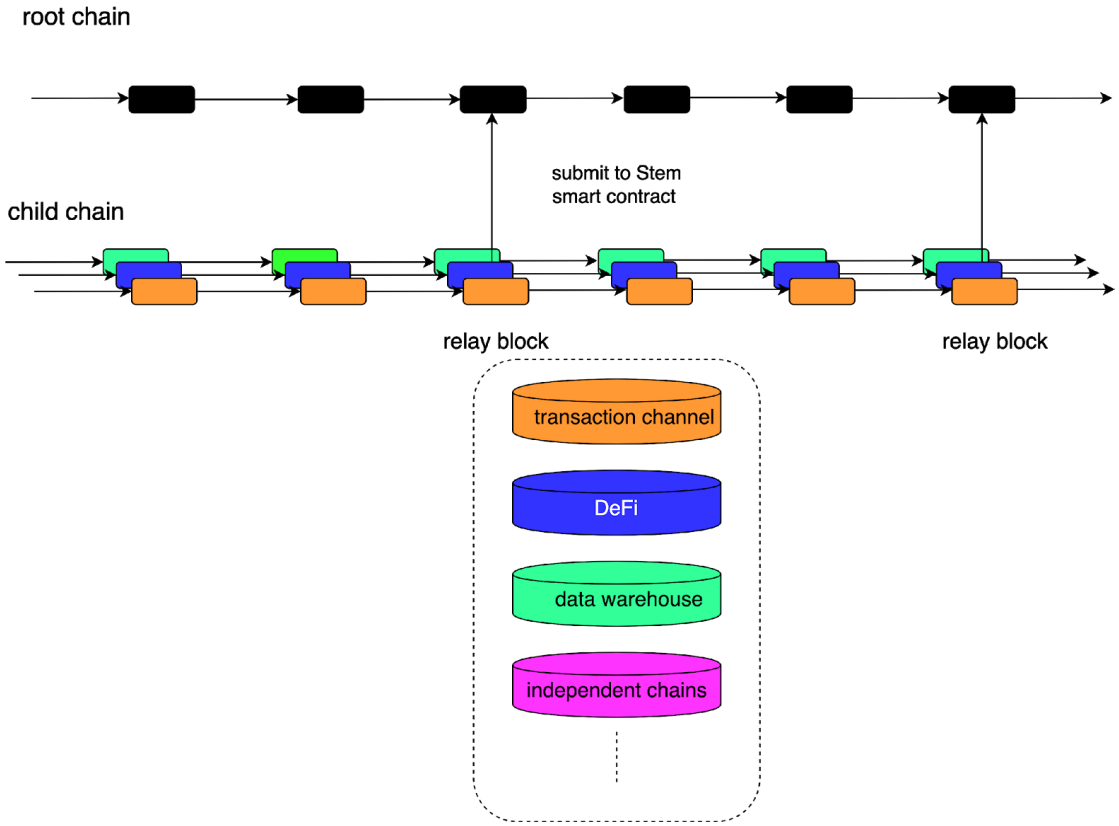

Fig. 6 Root chain and heterogeneous child chains

Stem subchain protocol has three major components: a child chain, a Stem smart contract on the root chain (a local chain in the SCDO mainnet), and a client interface. The child chain, which can be created by anyone, supports various consensus mechanisms such as PoS and

DPoS. The Stem smart contract acts as an anchor of the child chain on the root chain. The client interface is used to interact with the child chain and the Stem smart contract.

There are three roles in Stem subchain protocol: creator, operator, and user. Creator is the creator of the Stem smart contract. A certain amount of deposit (SCDO tokens) is required for the contract creation. Operators are registered in the Stem smart contract and are responsible for block recording in the child chain. They can be registered at the creation of the Stem smart contract or later. Users can deposit funds in the Stem smart contract and receive corresponding amounts of tokens of the child chain. They are then able to engage in the activities in the child chain. Since the operators could be malicious, Stem subchain protocol enables users to supervise the operators without sacrificing the efficiency of block production.

The account balance of operators and users are stored in the Stem smart contract. Operators or creators submit relay blocks to the Stem smart contract with a certain frequency. Relay blocks update the balances in the Stem smart contract and provide some hashes to document the updated states. If users noticed their balances are incorrect, they can submit a challenge to the most recent relay block requesting operators to provide proof. Correct proof removes the corresponding challenges. Relay blocks and the associated states are confirmed if no challenges exist after a challenge period.

References

1. Bitcoin: A Peer-to-Peer Electronic Cash System,Satoshi Nakamoto, https://bitcoin.org/bitcoin.pdf.

2. Ethereum White Paper, Vitalik Buterin, https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf.

3. Zerocash: Decentralized Anonymous Payments from Bitcoin, Eli Ben-Sasson. Etl, https://whitepaper.io/document/13/zcash-whitepaper

4. Algorand: Scaling Byzantine Agreements for Cryptocurrencies, Algorand, Yossi Gilad, etl, https://dl.acm.org/doi/pdf/10.1145/3132747.3132757