SCDO – A Scalable, Efficient, and Decentralized Blockchain

The SCDO Project Team, scdo.pro

Aug 20, 2020

1. Introduction

SCDO is a new peer-to-peer public blockchain system that allows for decentralized value transfer in a secure and efficient manner. It uses an account-balance model, supports general smart contracts, and is EVM compatible. It features 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 entire SCDO system scalable. The SCDO ecosystem includes the SCDO mainnet, subchains, SDKs, applications, and global user/developer communities. The vision of SCDO is to build such a decentralized ecosystem with powerful infrastructure.

Since Satoshi Nakamoto's seminal paper [1] created a purely peer-to-peer version of a digital cash system – now known as Bitcoin – many different digital token systems have been invented, such as Ethereum [2], Zcash [3], Algorand [4], etc. Among them, Ethereum introduced a complete smart contract framework (written in a Turing-complete language) into digital cash systems. A smart contract is a piece of computer code that can be executed when certain conditions are met. The invention of smart contracts established the cornerstone for blockchain applications. The critical goals that these digital cash systems try to achieve are security, efficiency, and decentralization. More precisely, a secure blockchain guarantees that the ownership and transfer of digital cash works 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 and payment centers in the current centralized financial payment system. The three goals are often said to form an impossible triangle, where at most, two of them can be met in a single blockchain system. The 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 perspective, the SCDO mainnet has been configured with 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, SCDO provides a robust peer-to-peer network and a private key and public key framework. Additionally, a new proof-of-work consensus algorithm called ZPoW has been invented. Unlike traditional algorithms that prioritize hash speed, ZPoW focuses on scientific computation, making it more CPU-friendly. GPU can execute ZPoW also. Because it is difficult to design an ASIC chip for ZPoW, the SCDO mainnet is less vulnerable to the dominance of a few miners. These special characteristics make the SCDO mainnet stronger in security and level the playing field for miners.

For heterogeneous applications, SCDO mainnet supports various applications on its own. Moreover, SCDO proposes and implements the Stem Subchain Protocol, which can support applications in different fields. For instance, if an application requires high throughput and is less concerned about 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 security for the subchain, and the subchain can choose its consensus algorithm, such as a PoS or PBFT consensus algorithm, which offers high throughput. In the next few sections, we will describe each solution mentioned above.

2. Notations and terminologies

In this section, we will state a few notations that will be used in later sections.

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

Transaction: a fixed number of data fields that record a piece of information for a peer-to-peer operation. These fields include the number of tokens to be sent, the receiver's public key, the sender's signature, and other miscellaneous information, as illustrated in Fig 1.

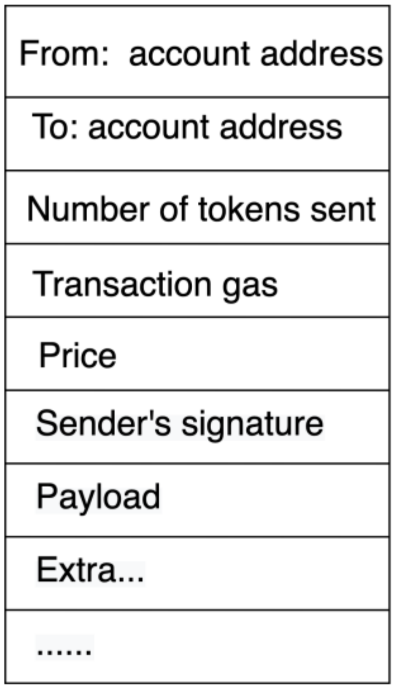

Fig. 1 An illustration of a transaction

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

A Block: a 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. The tx list is a set of transactions to be processed, while the debt list is a set of debts mainly recording transactions across different shards (which will be explained later). The maximum number of transactions and debts in a block is determined by the maximum block size, which is set to 1M bytes in the SCDO mainnet, roughly equivalent to 6000 transactions. The block structure is illustrated in Fig. 2.

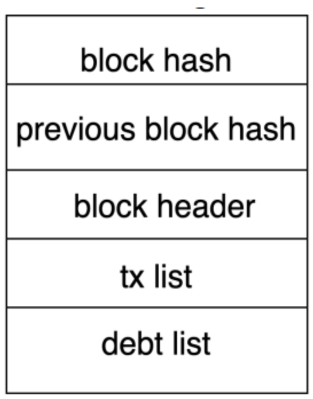

Fig. 2 An illustration of a block structure

A Merkle Tree: it is a binary tree in which each intermediate node has two children. In the SCDO system, a Merkle tree is used to record transactions or debts. Each leaf node contains the data of a transaction, and an intermediate node records the hash of its two children.

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

An Account：it 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.

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

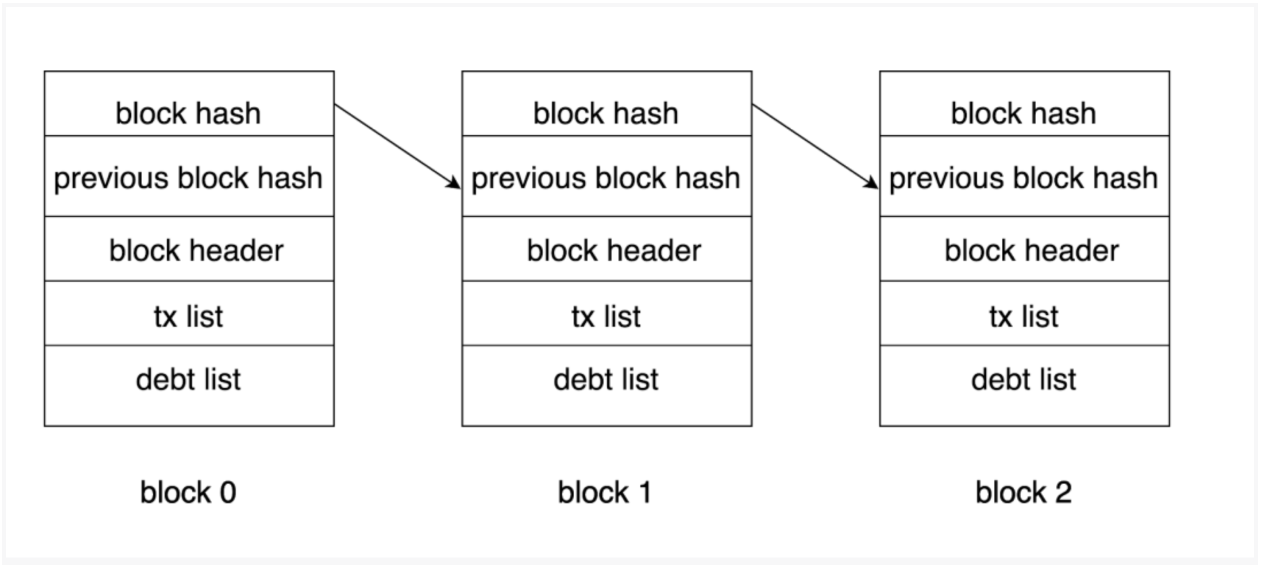

Fig. 3 A chain of blocks

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 primary objective of the SCDO sharding framework is to enhance the efficiency of transactions on the SCDO mainnet. Each shard can be considered as a separate blockchain. The SCDO sharding framework offers fundamental features that allow for efficient and accurate processing of peer-to-peer transactions.

In theory, the SCDO mainnet could have an unlimited number of shards. However, in practice, it is configured to have four shards. We accomplish this by dividing nodes and accounts into four subsets, with each subset forming a shard. A function is available to generate a pair of public and private keys for each shard, and an account data structure is created with a key pair. An account can perform various transactions on the SCDO mainnet.

There are two types of accounts on 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 in 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.

A mining node in a given shard keeps and updates the following meta data:

1. A chain of blocks (called the local chain or the full chain) that includes blocks created and confirmed by the mining nodes of the local shard so far.
2. A chain of block header hashes for each of the other shards (called a light chain), which plays a critical role in processing transactions among different shards.
3. A transaction Merkle Tree that records transactions sent from the local shard.
4. A debt Merkle Tree that records transactions sent to the local shard from the other shards.
5. A receipt Merkle Tree that records transaction receipts. The information included in a receipt mainly consists of 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 yet packed in confirmed block(s).

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

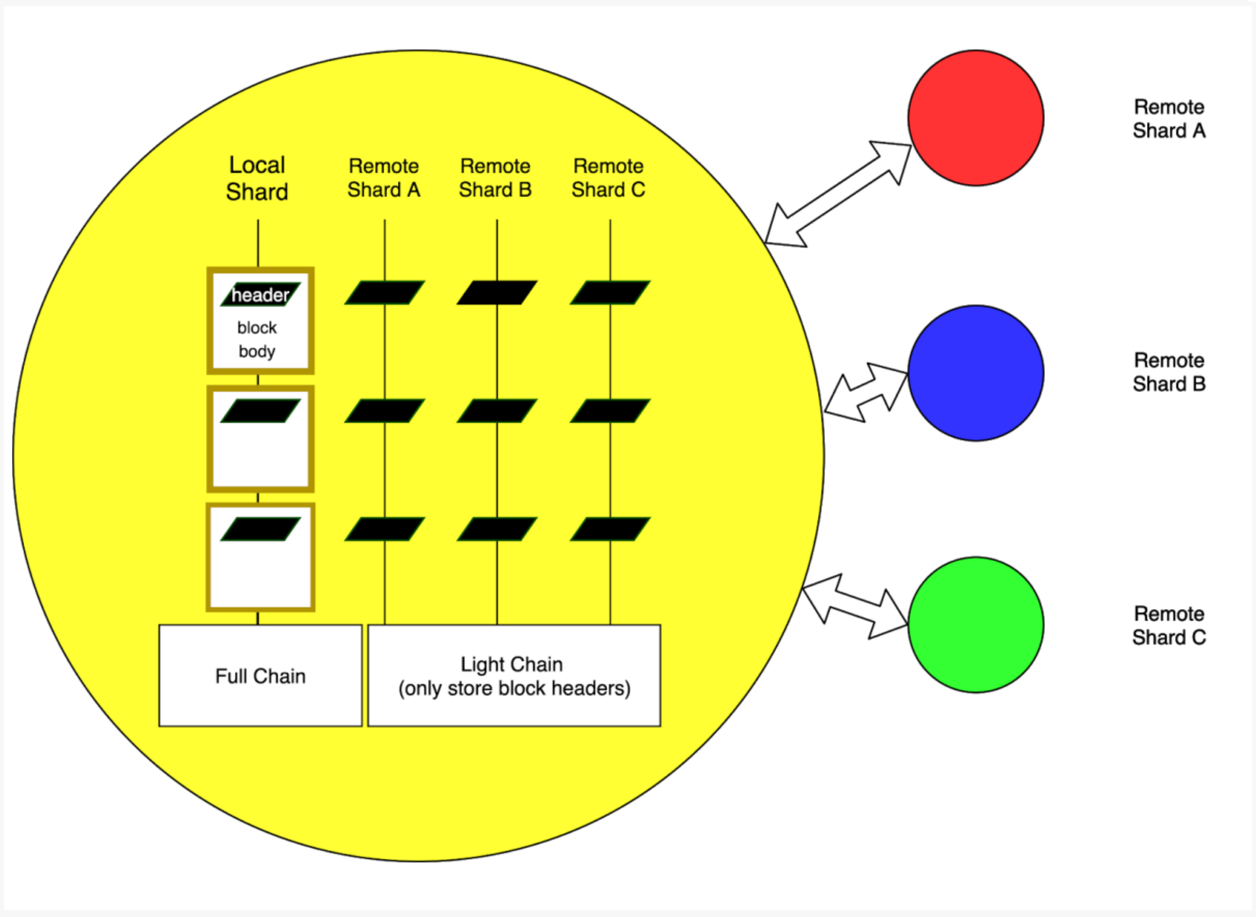

Fig. 4 Local chain and light chains illustration

In order for the SCDO sharding framework to work, the information of a transaction (tx) between two shards 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 number of blocks are confirmed in the sender's shard, 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, it is added to the incoming debt pool. When a mining node (Nd) selects 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 included in a confirmed block, the block header hash (blh) is returned. Nd verifies 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 proof of work (PoW) usually follows the process where a consensus algorithm, denoted as F(*,*), is used. For a given block b and nonce nc, F(b,nc) returns a computed value. If the computed value satisfies a given condition, then nc is the solution. The mining node that discovers the solution first is the winner to record b. The popular PoW consensus algorithm from Bitcoin uses a hash function. A mining node computes F(b,nc) by augmenting nc until the return value meets a predetermined difficulty. Once 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 added to the blockchain. The mining node that finds the solution nonce first is awarded some tokens.

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 kept this challenge in mind when designing the consensus algorithm for the SCDO mainnet. The consensus algorithm which we invent and implement is named ZPoW. It is CPU friendly and can run on CPU or GPU powered devices. GPUs are particularly good at parallel computations, making them more efficient for certain types of PoW algorithms. And there are probably no ASIC chips available for ZPoW in the foreseeable future.

In this section, we will introduce ZPoW and outline some of its desired properties. However, we will not delve into the mathematical details, which will be covered in a separate article.

The development of ZPoW consists of two phases. The first phase, ZPoW1, is an algorithm that computes the determinant of random matrices. One characteristic of this algorithm is that it cannot be implemented in parallel. Additionally, ASIC design for this algorithm is challenging. In the second phase of ZPoW (ZPoW2), multiple child algorithms are combined, and miners can choose one of these algorithms to mine blocks. ZPoW2 includes a unique dynamic mechanism that adjusts the difficulties of different child algorithms so that miners working on the same child algorithm cannot produce consecutive blocks. Specifically, a child algorithm that produces consecutive blocks will have a higher difficulty than other child algorithms. Moreover, ZPoW2 introduces a pre-computing proof to discourage miners from switching between child algorithms. The pre-computing proof ensures that miners' previous work on a child algorithm can increase their chances of mining future blocks. At a higher level, this innovative ZPoW2 algorithm makes it almost impossible to perform a 51% attack.

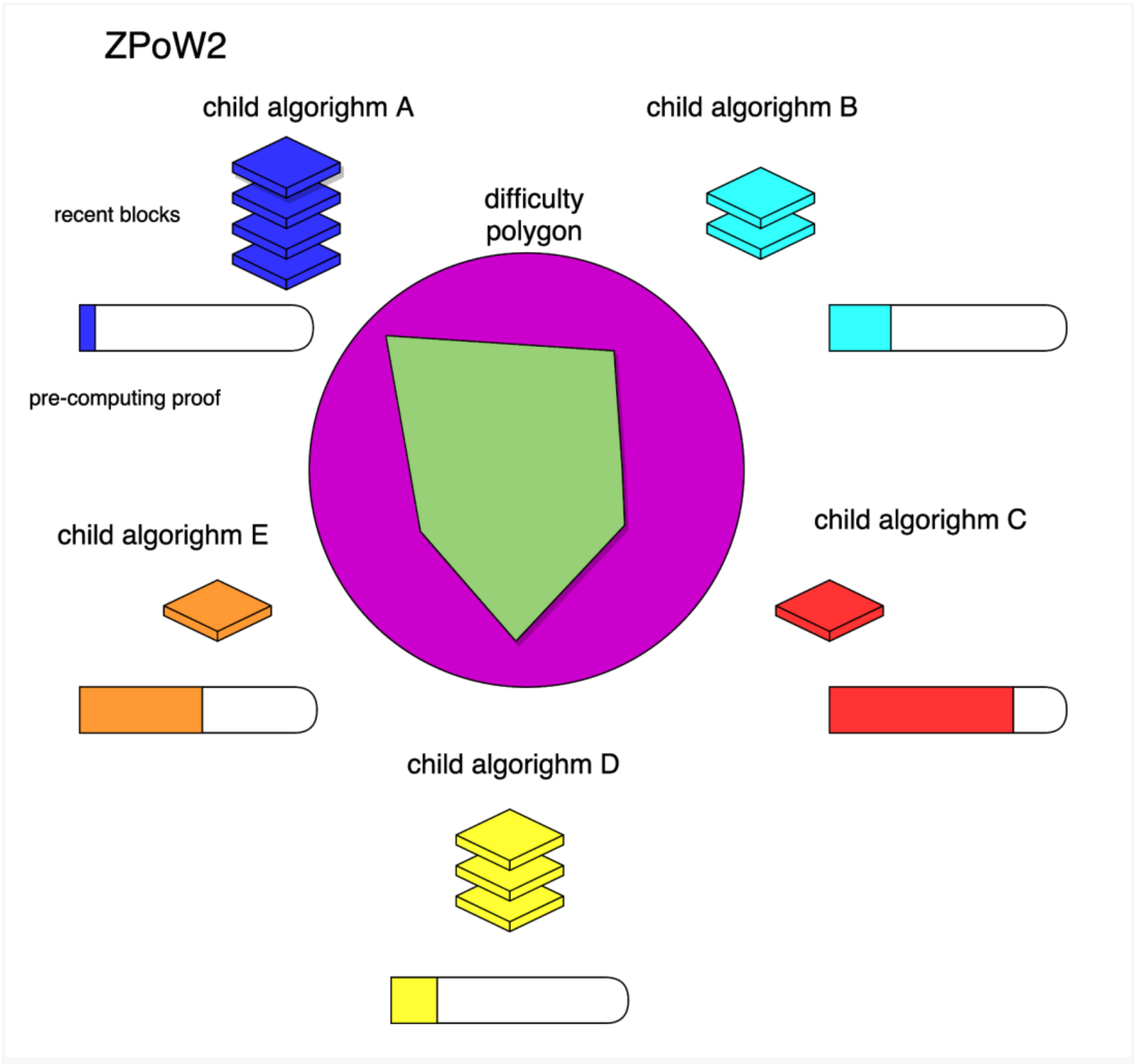

Fig. 5 An illustration of ZPoW2

The following is a brief introduction of ZPoW1:

Let CF(block_hash, nonce) be the function with two parameters invented in SCDO. The first parameter is a hash string, and the second is an integer. The output is an n x n matrix with 0, 1, or 2 coefficients. In SCDO, n is set to be 30. CF can be viewed as producing a verifiable pseudo-random matrix. The function det(M) computes 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 as follows: 1) it cannot be implemented in parallel; 2) its computing process is verifiable; and 3) the return value is verifiable. We are now 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. Based on our extensive testing, the tail of P forms a smooth curve and fits nicely with an exponential distribution with λ 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 value of T is 20 seconds.

CF(*,*) *is designed to be a sequential process, and the major portion of ZPoW's time is spent computing CF(*,*)*. This will possibly limit 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.

The Stem Subchain Protocol provides a Layer-2 scalability solution for the SCDO mainnet. With this protocol, the SCDO mainnet (root chain) can protect the safety of funds on child chains while child chains can implement different consensus algorithms to speed up transactions. There is no limit to the number of child chains. This subchain protocol is an important infrastructure for scaling up the SCDO system.

The Stem Subchain Protocol aims to provide a full set of useful features, including support for 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 serves as the "umbilical cord" between the root chain and the child chain), 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 can send transactions on the child chain and update the state on the root chain.
2. A data warehouse that records important information, such as logistics data, and the history of this data is endorsed by the SCDO root chain.
3. Decentralized finance: Child chain users can exchange/borrow SCDO on the root chain and offer equivalent assets on child chains.
4. Independent chains with their own tokens that can be exchanged with SCDO.

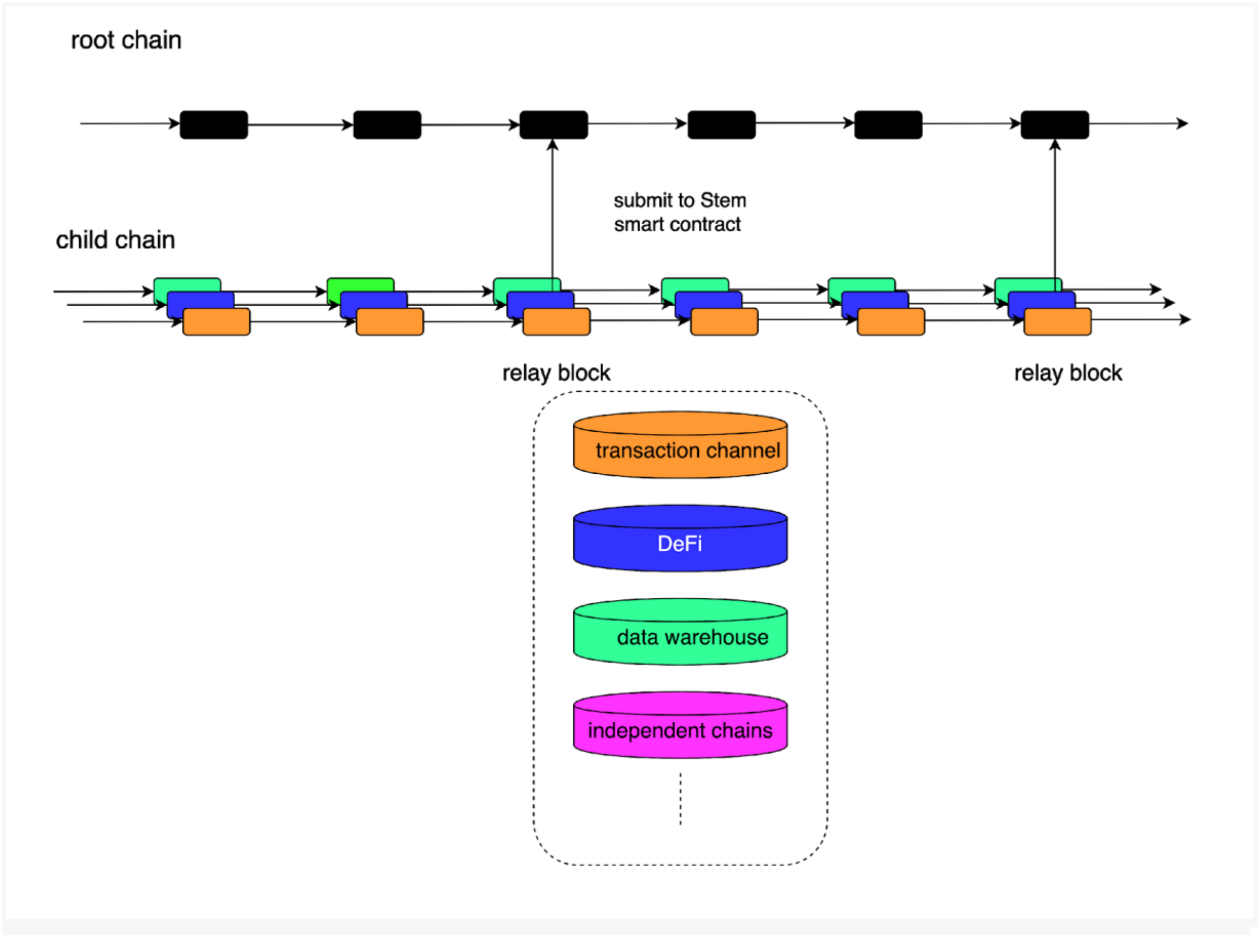

Fig. 6 Root chain and heterogeneous child chains

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 notice 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