Filecoin-winningPoSt logic introduction

FileCoin (https://learnblockchain.cn/tags/FileCoin)

Lotus PoSt consists of two parts: winningPoSt and windowPoSt. winningPoSt is the proof of PoSt that needs to be provided when obtaining the right to produce blocks. From all valid Sectors, extract a Sector and spot check 66 leaves on the Sector.

The part of Lotus PoSt has changed from electionPoSt to two new PoSts, one is winningPoSt and the other is windowPoSt. Let me talk about winningPoSt first. winningPoSt, as the name suggests, is a PoSt performed at the time of winning. The so-called winning is to obtain the right to produce blocks.

To put it simply, winningPoSt is a randomly checked sector, and all 66 randomly checked merkle paths in this sector are correct. The code logic starts with the code of Lotus Go. Everything starts from the block-the mineOne function of the Miner structure of lotus/miner/miner.go.

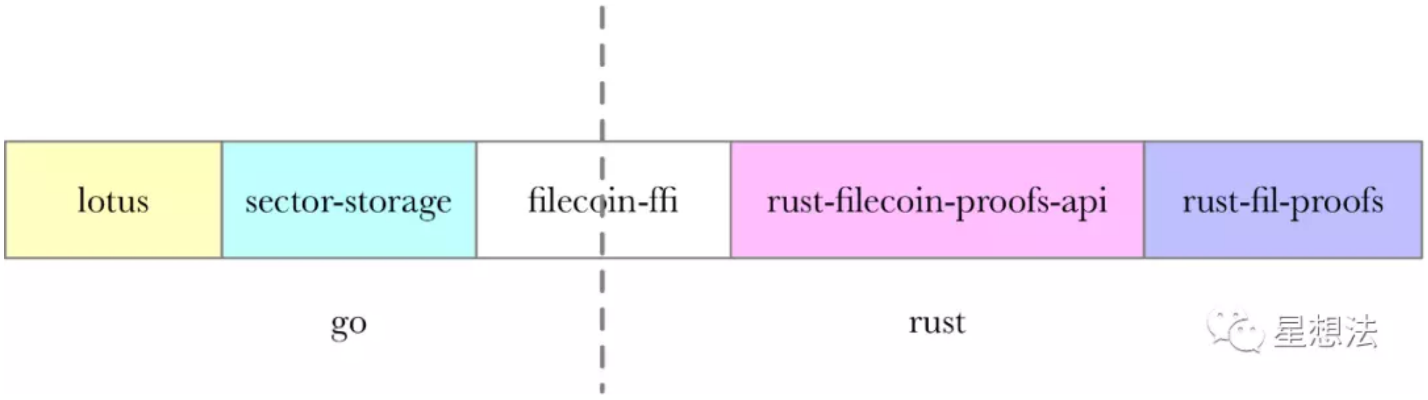
```
func (m *Miner) mineOne(ctx context.Context, addr address.Address, base *MiningBase) (*types.Block

 mbi, err := m.api.MinerGetBaseInfo(ctx, addr, round, base.TipSet.Key())

 rand, err := m.api.ChainGetRandomness(ctx, base.TipSet.Key(), crypto.DomainSeparationTag_WinningP
 prand := abi.PoStRandomness(rand)
 postProof, err := m.epp.ComputeProof(ctx, mbi.Sectors, prand)
```

Among them, the MinerGetBaseInfo function is to obtain some basic information, including the sector information that needs to be extracted. The ComputeProof function is to calculate the proof of winningPoSt.

Because the specific implementation of these logics is implemented in rust-fil-proofs, which is the rust language. From go to rust-fil-proofs, there are many interfaces:



The interface in the middle is not introduced, just look at the two API functions provided by rust-fil-proofs.

01 Setting the number of spot checks

The number of sectors and the total number of random-checked leaves are defined in rust-fil-proofs/filecoin-proofs/src/constants.rs:

```
pub const WINNING_POST_CHALLENGE_COUNT: usize = 66;
pub const WINNING_POST_SECTOR_COUNT: usize = 1;
```

In other words, it is necessary to extract a Sector from the effective Sector, and spot check 66 leaf nodes on this Sector.

02 Sector's spot check logic

The generate_winning_post_sector_challenge function implements the sector's spot check logic. The core logic is obviously how to spot check Sector? The specific logic is in the function of fallback::generate_sector_challenges:

```
let mut hasher = Sha256::new();
hasher.input(AsRef::::as_ref(&prover_id));
hasher.input(AsRef::::as_ref(&randomness));
hasher.input(&n.to_le_bytes()[..]);

let hash = hasher.result();

let sector_challenge = LittleEndian::read_u64(&hash.as_ref()[..8]);
let sector_index = sector_challenge % sector_set_len;
```

To put it simply, the provider_id, random information, and the random number of the sector are randomly checked for the sha256 hash calculation, and the calculation result is modulo the current limited number of sectors. That is, the sector_index is the id of the sector that was finally spot-checked.

03 Challenge's leaf spot check logic

generate_winning_post randomly checks the leaf nodes on the merkle tree (replica_r_last) formed by the randomly checked Sectors. The calculation logic of spot check leaf nodes is in the function of fallback::generate_leaf_challenge:

```
let mut hasher = Sha256::new();
hasher.input(AsRef::::as_ref(&randomness));
hasher.input(&sector_id.to_le_bytes()[..]);
hasher.input(&leaf_challenge_index.to_le_bytes()[..]);
let hash = hasher.result();

let leaf_challenge = LittleEndian::read_u64(&hash.as_ref()[..8]);

let challenged_range_index = leaf_challenge % (pub_params.sector_size / NODE_SIZE as u64);
```

The random information, sector id and challenge leaf number are hashed. The calculated result is modulo the total number of leaves. For 32G Sector, the number of leaves is 1G.

04 Zero-knowledge proof circuit

The calculation part of the zero-knowledge proof can be viewed in the rust-fil-proofs/post/fallback directory. The general logic module and structure can be viewed in the introduction of the previous article:
Filecoin-PoREP Circuit Introduction Let's (https://learnblockchain.cn/article/890)
talk about the Sector structure in rust-fil-proofs/post/fallback/circuit.rs. This structure represents a spot check. It can be seen from the synthesissize function:

```
// 1\. Verify comm_r
let comm_r_last_num = num::AllocatedNum::alloc(cs.namespace(|| "comm_r_last"), || {
comm_r_last
.map(Into::into)
.ok_or_else(|| SynthesisError::AssignmentMissing)
})?;

let comm_c_num = num::AllocatedNum::alloc(cs.namespace(|| "comm_c"), || {
comm_c
.map(Into::into)
.ok_or_else(|| SynthesisError::AssignmentMissing)
})?;

let comm_r_num = num::AllocatedNum::alloc(cs.namespace(|| "comm_r"), || {
comm_r
.map(Into::into)
.ok_or_else(|| SynthesisError::AssignmentMissing)
})?;

comm_r_num.inputize(cs.namespace(|| "comm_r_input"))?;
```

comm_r is input as public, and the other comm_r_last and comm_c are input as private.

```
// 1\. Verify H(Comm_C || comm_r_last) == comm_r
 {
 let hash_num = ::Function::hash2_circuit(
 cs.namespace(|| "H_comm_c_comm_r_last"),
 &comm_c_num,
 &comm_r_last_num,
 )?;

 // Check actual equality
 constraint::equal(
 cs,
 || "enforce_comm_c_comm_r_last_hash_comm_r",
 &comm_r_num,
 &hash_num,
 );
 }
```

Verify that comm_r is calculated by comm_c and comm_r_last.

```
// 2\. Verify Inclusion Paths
for (i, (leaf, path)) in leafs.iter().zip(paths.iter()).enumerate() {
 PoRCircuit::::synthesize(
 cs.namespace(|| format!("challenge_inclusion_{}", i)),
 Root::Val(*leaf),
 path.clone(),
 Root::from_allocated::(comm_r_last_num.clone()),
 true,
 )?;
}
```

Verify that the Merkle tree root can be calculated correctly from the leaf nodes.

to sum up:

Lotus PoSt consists of two parts: winningPoSt and windowPoSt. winningPoSt is the proof of PoSt that needs to be provided when obtaining the right to produce blocks. From all valid Sectors, extract a Sector and spot check 66 leaves on the Sector.

There are many original high-quality articles in my official account **star idea** , and you are welcome to scan the code and pay attention.

Published on 2020-04-29 17:43 Reading (2239) Credits (108)
Category: FileCoin (https://learnblockchain.cn/categories/FileCoin)

0 likes Favorites

Articles you may be interested in

Why is NFT different? How does Filecoin's distributed storage solution empower NFT? (https://learnblockchain.cn/article/2495) 21 views

IPFS Weekly 132 | The next gathering will showcase the cooperation between IPFS and NFT (https://learnblockchain.cn/article/2418) 69 views

IPFS helps expand ETH, Filecoin and DeFi to create the future together, and analyzes the powerful combination of IPFS and ETH (https://learnblockchain.cn/article/2390) 66 views

The Web3.0 China Summit came to a successful conclusion, Hu Feng, COO of Time Cloud: Filecoin needs long-termism! (https://learnblockchain.cn/article/2375) 94 views

Web 3.0 is coming, distributed storage plays an important role|Space Cloud invites you to participate in the Web3.0 China Summit and Distributed Storage Industry Conference (https://learnblockchain.cn/article/2349) 132 views

People's Daily Online: "Distributed storage opens up a market of 100 billion yuan", IPFS welcomes the new digital era! (https://learnblockchain.cn/article/2320) 186 views

Related questions

Which RPC interfaces are the three interface calls in the figure below the filecoin block explorer? ? ? (https://learnblockchain.cn/question/1505) 1 answer

What is the current progress of Filecoin? (https://learnblockchain.cn/question/4) 1 answer