Sigmoid: ASIC-resistant Mineable KIP-7 Token

John Weligon
Electrical and Computer Engineering Dept.
sigmoid.token@gmail.com

I. Introduction

The advent of gigantic mining pools has long been a headache for a cryptocurrency field. Due to the possibility of 51% attack and block withholding attack, the very existence of the gigantic mining pools was one of the biggest threats to the security of proof-of-work cryptocurrencies. Not only in terms of security but also in terms of energy efficiency were those mining pools a nuisance. It is known that the process of creating Bitcoin to spend or trade consumes around 91 terawatt-hours of electricity annually, more than is used by Finland, a nation of about 5.5 million.

Various technical attempts have been made to solve this problem. One of them was to switch the consensus mechanism to proof-of-stake. Ethereum, which Sigmoid is going to be based on, is an example of cryptocurrency that is planning a switch from PoW to PoS mechanism. Though this changeover of the consensus mechanism can effectively alleviate the aforementioned issues, to miners can the whole changeover be seen as 'being thrown away like an old shoe.'

To miners, mineable ERC-20 tokens like 0xBitcoin were great substitutes for cryptocurrencies preparing for a changeover. The move was faster than expected. In the case of 0xBitcoin, the hash rate is already enormous 100Th/s. This is because there was an influx of miners from traditional cryptocurrencies. While the competition between 0xBitcoin miners is already fierce, the biggest drawback that wasn't revealed yet is that 0xBitcoin is prone to ASIC mining. This is because hashing algorithm used in 0xBitcoin, keccak-256, is not memory-hard and ASIC-friendly. If ASIC miners jump into the mining pool, the mining difficulty and the competition between miners will skyrocket.

Therefore, we bring another older attempt for decentralization: to use an ASIC-resistant hashing algorithm for proof of work. For example, Scrypt and Ethash, which were known to be GPU-friendly and ASIC-resistant, were respectively selected as a hashing algorithm by Litecoin and Ethereum. Those memory-hard hashing algorithms demonstrated their effectiveness for a long period though ASIC modules for those algorithms have now been developed and commercialized.

In this paper, a new ASIC-resistant KIP-7 token with a relatively new memory-hard hashing algorithm, **Sigmoid** is presented. For minting, Sigmoid uses Balloon hashing for its key derivation function, which is a NIST-recommended algorithm theoretically proven to be memory-hard. In this way, we expect to achieve thorough decentralization and bring about short-term energy-efficiency and equality through incapacitation of ASIC miners.

II. What is KIP-7?

Klaytn is an enterprise-grade, service-centric platform developed by Kakao, Korea. Klaytn, the currency running on the platform, is the 39th largest cryptocurrency in crypto market capitalization. KIP-7 is a fungible Klaytn Compatible Token (KCT) that has properties of uniformity and divisibility, which is very similar to ERC-20 from Ethereum.

The difference of KIP-7 from ERC-20 is that every token transfer/mint/burn must be tracked by event logs. This means that a transaction must be emitted for any action related to transfer/mint/burn. Another difference from ERC-20 is that the KIP-13 interface for each method group must be implemented. KIP-13 is a standard that defines a method to query whether a contract implements a certain interface or not.

III. Difficulty Adjustment and Rewarding System

A. Difficulty adjustment

Overall difficulty adjustment system of Sigmoid follows Digishield, the system originally used in Digibyte and adopted in Dogecoin. DigiShield is designed to solve "chain freezing" problem. If the mining difficulty becomes higher, as miners hop between various cryptocurrencies with their mining, a relatively new currency would end up with a "frozen" chain, where it takes too long to find a block. In order to prevent such a situation from happening, Digishield lets the difficulty "fall" very fast so that the miners from other currencies would come again to begin mining. Though Sigmoid doesn't need to worry about the "chain freezing" problem, Digishield was adopted in order to prevent the mining difficulty from being too high.

The rate of block creation is adjusted every 64 blocks to aim for 1 block creation per 600 Klaytn blocks, which is roughly 6 blocks per hour. All difficulty targets are bound within minimum and maximum difficulties of 2^{16} and 2^{234} respectively.

B. Rewarding system

Just like 0xBitcoin, maximum total supply of Sigmoid is 21 million tokens and the amount of SIG issued per block is set to decrease logarithmically, having a 50% reduction every time half of the remaining supply has been mined. In result, total supply of Sigmoid will never exceed 21 million.

The reward halving occurs when the reward era increases. The reward era increases if the tokens minted count has exceeded the maximum era supply which is calculated via the Eq. (1).

$$\text{max era supply} = \text{total supply} - \frac{\text{total supply}}{2^{\text{reward era}+1}} \qquad (1)$$

IV. Hashing algorithm

Sigmoid uses Balloon hashing for its key derivation function. Balloon hashing is a NIST-recommended key derivation algorithm which is theoretically proven to be memory-hard. Memory-hardness of this particular key derivation function enables Sigmoid to be ASIC-resistant. [2]

Another strength of Balloon hashing other than memory-hardness is its versatility. Since Balloon hashing is only a key derivation algorithm, it can use almost every hash function for its hashing / pseudo-random-number generation processes. By using hash functions that are already built in Solidity API, it is possible to reduce the gas consumed for validation of mining results. In the case of Sigmoid, the hash function is SHA-256.

A. Balloon Hashing

Like 0xBitcoin, to prevent pre-mining and Man-in-the-Middle attack, the digest of Sigmoid's hash function includes a recent Ethereum block hash and msg.sender's address. In addition, the digest also includes another variable named *cnt* to secure memory-hardness. *cnt* increases gradually as the hashing process continues and helps mixing the hash blocks.

```
func Balloon(block_t nonce,
      block_t salt, // previousBlockHash, msg.sender
      int s_cost, // Space cost (main buffer size)
      int t_cost): // Time cost (number of rounds)
   int delta = 3 // Number of dependencies per block
   int cnt = 0 // A counter (used in security proof)
   block_t buf[s_cost]): // The main buffer

   // Step 1. Expand input into buffer.
   buf[0] = hash(cnt++, salt, nonce)
   for m from 1 to s_cost-1:
      buf[m] = hash(cnt++, buf[m-1])

   // Step 2. Mix buffer contents.
   for t from 0 to t_cost-1:
      for m from 0 to s_cost-1:
      // Step 2a. Hash last and current blocks.
      block_t prev = buf[(m-1) mod s_cost]
      buf[m] = hash(cnt++, prev, buf[m])

      // Step 2b. Hash in pseudorandomly
      // chosen blocks.
      for i from 0 to delta-1:
         block_t idx_block = ints_to_block(t, m, i)
         int other = to_int(hash(cnt++, salt,
                  idx_block)) mod s_cost
         buf[m] = hash(cnt++, buf[m], buf[other])

   // Step 3. Extract output from buffer.
   return buf[s_cost-1]
```

In the case of Sigmoid, *s_cost* was set to $1024 * 1024/32$ to make the size of *buf* array 1MB, which is similar to another hashing function Scrypt. *t_cost* and *delta* were set to 3.

B. Switching hashing function

The drawback of using Balloon hashing for validation is that Balloon hashing process takes up too much gas. If the situation goes wrong, the price for minting will be more expensive than the value of the minted Sigmoid tokens. To prevent such a situation from happening, the owner, the developer in other words, has the authority to switch the hashing function from Balloon hashing to Keccak-256 and vice versa. If the gas price becomes too high, the hashing function will be changed to an easier one (Keccak-256) for convenience.

References

[1] https://github.com/0xbitcoin/white-paper-v2

[2] Boneh, Dan, Henry Corrigan-Gibbs, and Stuart Schechter. "Balloon hashing: A memory-hard function providing provable protection against sequential attacks." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2016.

***Index Terms*—ASIC-resistant, mineable, ERC-20, Ethereum, PoW**