CRYPTONOTE STANDARD 010 Albert Werner
Category: Main Track Marec Pliskov
Montag
CryptoNote
August 2014

CryptoNote Difficulty Adjustment

Abstract

This document is part of the CryptoNote Standards describing a peer-to-peer anonymous payment system. It defines the method for maintaining the rate at which blocks are generated.

Table of Contents

1. Introduction

CryptoNote uses a hash-based proof-of-work scheme to reach a distributed consensus among peers. When connected to the network, a peer considers the main branch of the blockchain (the branch with the most work done) to be the reference transaction history. See [CNS003] and [CNS009] for basic information on CryptoNote blockchain.

The work being done by peers is an iterated hash calculation

[CNS008]. Every block is considered valid only if the value of its hash is less than or equal to some target value. As the hash rate of the network changes, the target value is adjusted to keep the rate of block generation steady.

This document describes the exact algorithm for calculating this value.

2. Definitions

block: a set of data (payload) with a block header

block height: the distance between the block and the genesis block in the blockchain

blockchain: a tree structure of blocks

difficulty: a number that measures the amount of work necessary to find a block

proof of work: a way for a computer system to demonstrate that it expended a certain amount of computational resources in support of a particular decision

target: the maximal possible value of the hash of a block header for it to be considered valid

3. Algorithm Parameters

TargetTime: expected time to find a block. Default value: 120 seconds.

DiffWindow: the number of the previous blocks used by the algorithm to estimate the hash rate. The algorithm will adjust to small changes in the hash rate in approximately TargetTime*DiffWindow seconds. Default value: 720 blocks or one day.

DiffCut: the number of highest and lowest timestamp values to be ignored, as they are considered to be outliers. Default value: 60.

DiffLag: the number of last blocks that should be discarded previous to any subsequent computation. This is done to make it harder to create a blockchain fork with higher cumulative difficulty. Default value: 15.

4. Algorithm Description

The idea behind the algorithm is to estimate the hash rate of the network as a ratio of the total difficulty of several last blocks to the time it took the peers to find these blocks. However, individual timestamps are not reliable, so the array of timestamps is sorted and order statistics are used as more robust estimators for the actual time values. At the same time, the array of past difficulty values is accurate, so there is no need to sort it.

In addition to this, a few last blocks are completely ignored when computing the difficulty. This guarantees that in a blockchain fork of small length, blocks at the same height would have the same

difficulty. This makes it harder to make one of the chains have higher cumulative difficulty by manipulating timestamps.

In the first step of the algorithm, the exact blocks that are going to be used in the subsequent computation are determined. There are three cases:

- If the current branch has no more than DiffWindow blocks, all blocks are used.

- If the current branch has no less than DiffWindow blocks, but no more than DiffWindow+DiffLag blocks, the first DiffWindow blocks are used.

- Otherwise, the last DiffLag blocks are discarded and then the last DiffWindow blocks are used.

The number of blocks selected by this step will be denoted by N. The value of N is equal to either DiffWindow or the number of blocks in the current branch, whichever is smaller.

In the second step, the timestamps of the selected blocks are sorted, but the difficulty values are left in place.

In the third step, some of the first and some of the last blocks are discarded. The number of blocks to be discarded is determined as

follows:

- If N is not greater than DiffWindow - 2*DiffCut, no blocks are discarded.

- Otherwise, the first Ceiling((N + 2*DiffCut - DiffWindow) / 2) and the last Floor((N + 2*DiffCut - DiffWindow) / 2) are discarded.

The number of blocks that remain is either DiffWindow - 2*DiffCut or N, whichever is smaller.

In the last step, the value TotalT is computed as the difference between the timestamps of the last and the first blocks and the value TotalD is computed as the sum of the difficulty values of all blocks except the first (the first block is excluded because it was mined before the time indicated by its timestamp, so the time it took to mine it is not included in TotalT). If the value of TotalT is not positive, it is assumed to be 1. Then, the difficulty of the next block is Ceiling(TotalD / TotalT * TargetTime).

Special case: the difficulty of the first two blocks is 1.

The target value is computed as follows: Target = Floor((2^256 - 1) / Difficulty). Alternatively, it is possible to check the hash of a block without explicitly computing the target value: the block is valid if Hash*Difficulty < 2^256.

5. References

[CNS003] "CryptoNote Blockchain", CryptoNote Standard 003, September 2012.

[CNS008] "CryptoNight Hash Function", CryptoNote Standard 008, March

2013.

[CNS009] "CryptoNote Technology", CryptoNote Standard 009, August
2013.