BeamHash III Short-Specification

Wilke Trei

March 28, 2020

1 Definition of a Solution and Naming

A valid BeamHash III solution is computed in six rounds with the first round being a *seeding phase* in which the algorithm is pseudo-randomly initialized using an element index and a global nonce *nonce*.

The elements created and processed during the rounds are called *step rows*. Each step row is composed of two sub-elements the *work bits* and an *index tree*. The index tree is tracking the way of its step rows through the rounds of the algorithm and has similar properties to the ones known from the Equihash 144/5 algorithm, namely

1. Each entry of the index tree needs to be unique.
2. Two index trees that are output round i can only be combined to a new element in round $i+1$ if the 24 least significant work bits of the two corresponding step rows are equal, i.e. xor to zero. In round 5 this applies to the lowest 48 work bits.
3. When two index trees get combined in a round to make a new step row, the one with the lowest entry is set first in memory.

This properties make the output of a new index tree that is combined from two parent trees of a previous round unique. The final solution to a BeamHash III problem is the index tree that is result of a combination in the 5th and last round. Hereby the seeding phase is considered to be round 0.

What distinguishes BeamHash III from Equihash is the generation of the initial step rows and the algorithm to derive new work bits from the combination of previous ones. This is described in the following paragraphs.

2 Seeding Phase

A typical BeamHash III work send by a solo node or a pool is given by a 32 byte (256 bit) pre-work that is a hash of the block header to mine on. To create an individual work to mine on the 32 byte pre-work is concatenated by a 16 byte (128 bit) nonce. We define the 32 byte output of

$$IndividualWork = Blake2B\,(pre-work \mid nonce)$$

to be the input for the BeamHash III seeding phase. Hereby the Blake2B is the standard implementation with "BeamHash" as personalization string in the initialization of Blake2B. The individual work is now input to the 0th round of the algorithm.

In this round of the algorithm in total 2^{25} initial step rows are generated. The index tree of each of this step rows only consists of its individual 25 bit index. Beside the index each step row has 448 initial work bits that are generated in seven chunks of 64 bits like follows.

$$WorkBits\,[i \cdot 64\ ..\ ((i+1) \cdot 64) - 1] = SipHash24\,(IndividualWork, (index \ll 3) + i)$$

3 Combination Phase

Before each combination phase the lowest 64 bits of each step row will be replaced in a *mixing* step. For this a serialization of the step row is created, i.e. its current index tree is concatenated to the its remaining work bits. The result will be padded with zeros or truncated to 512 bits. So the input of the mixing steps are 512 bit that also can be read as eight 64bit unsigned integers we will denote by s_i.

Then the mixing step is given as

$$WorkBits\,[0..63] = \left(\sum_{i=0}^{7} s_i \lll (29 \cdot (i+1) \pmod{64})\right) \lll 24.$$

The sums in this formula use 64 bit modular arithmetic. The $\lll$ character stands for the 64 bit left rotate function.

After the mixing step a combination step is applied. Let A and B be two step rows with completed mixing step that satisfy

$$WorkBits_A\,[0..23] = WorkBits_B\,[0..23]$$

then a new output step row C for the next round is created. This has a new working bit set given by

$$WorkBits_C\,[0..n] = WorkBits_A\,[24..n+23] \otimes WorkBits_B\,[24..n+23]$$

where the number n of left bits varies with the round following the following table. The index tree of C is composed by concatenating the two of A and B in the right order following the rules in the definition section.

Round	Left work bits	Index Tree Size
0	448	25
1	424	50
2	400	100
3	376	200
4	**288**	400
5	0	800

Note that intentionally in round 4 we drop the most significant left work bits and reduce to a size of 288 bits for remaining work bits that get involved into the mixing for round 5. This encourages implementations that do the mixing step of the next round before writing out the results of the current round. Thus at the end of round 3 - after round 4 mixing - the most significant 64 work bits can be dropped to make the total output of the round fit an ulong8 (512 bit) vector.

A solution to the problem defined by the pre-work and nonce will then be any resulting 800 bit index tree that was result to round 5 and satisfies the initially meant conditions on valid index trees.

4 Beam Hash III Solution Format and Implementation Details

Independent of the proof of work itself the following details are mandatory to know to create a valid beam block header using the new Beam Hash III PoW.

Originally Beam uses a 8 byte (64 bit) nonce and a solution size of $32 \cdot 26bit = 832bits = 104bytes$. This general format is *unchanged* for Beam Hash III. Beam Hash III has a total solution size of $32 \cdot 25bits = 800bits = 100bytes$ when packed ideally. Therefore we define the 4 most significant bytes of a Beam Hash III solution to be the *extra nonce* of a Beam Hash III solution. The extra nonce bytes can freely be chosen by the miner to extend the range of available nonces.

The 4 bytes of extra nonce are part of the solution when submitting a solution, so a Beam node and all Beam pools expect to receive a 8 byte nonce and a 104 byte solution as before, while the actual sizes are a 12 byte nonce and a 100 bytes solution.

That said, the extra nonce is as well part of the difficulty calculation - since it is part of the 104 solution bytes - but it also will be appended to the input of the blake2b part of the algorithm, so the solution depends on the extra nonce bytes.

Concretely the blake2b input is a message of length 32 bytes for the work from node, 8 bytes nonce and 4 bytes extra nonce, so 44 bytes total in the mentioned order.