Blossom Specification Review

Zcash

May 31, 2019 – Version 1.0

Prepared for

Jack Grigg
Daira Hopwood
Benjamin Winston

Prepared by

Thomas Pornin

Table of Contents

1 Table of Contents 2
2 Executive Summary 3
3 Blossom Network Upgrade Specification Review 4

Executive Summary

Synopsis

In March 2019, Zcash engaged NCC Group to perform a review of the protocol changes for the upcoming network upgrade, codenamed "Blossom." Two Zcash Improvement Proposals (ZIPs) were to be reviewed, describing changes to the issuance rate of new blocks and a split of the Founders' Reward into separate streams. One consultant was assigned, for a total of four person-days.

Scope

The review was to cover two specific ZIPs:

- ZIP 207[1]: Split of the Founders' Reward over several separate destination addresses
- ZIP 208[2]: Reduction of the target block spacing, from 150 seconds down to 75 seconds

A draft of the Zcash protocol specification, with these two ZIPs applied, was also provided.

Key Findings

The review did not find any significant security flaw in the two ZIPs.

The target block spacing change defined in ZIP 208 makes the difficulty adjustment algorithm more reactive when expressed in physical time (it is unchanged when expressed in blocks, but new blocks happen at twice the previous rate). This, in turn, increases the artificial difficulty inflation when a clock drift event occurs (six successive blocks emitted from nodes with a local clock skewed up to two hours in the future); hoever, under the usual hypothesis that attackers don't control a majority of the mining power, such events are very rare and seem hard to trigger maliciously. The extra inflation is still moderate and will be resorbed in less than two hours after the event. Increasing the damping factor in the adjustment algorithm can partially or even fully mitigate this issue if deemed important enough to warrant any action.

In a few places, some alternate wording is suggested to make the description more precise.

[1] https://github.com/zcash/zips/blob/c7e79e2909b4de618743d35d84b2b2d5fb884b7d/zip-0207.rst
[2] https://github.com/zcash/zips/blob/eb297d2454488d10fa3d76d582b2b11ba02f2e4e/zip-0208.rst

ZIP 207

ZIP 207[3] describes a change in the handling of the Founders' Reward. It is to be applied when the network is upgraded to the next protocol version (Blossom). In the current protocol version (Sapling), each block contains one *coinbase* transaction that corresponds to the 12.5 ZEC subsidy awarded to the new block; exactly 1/5th of that subsidy (2.5 ZEC) is sent to a specific address, which represents the reward to the network founders. As per Sapling rules, this reward mechanism was supposed to run until the first "halving" time, scheduled to happen when chain height 850,000 is reached. ZIP 207 replaces this reward with ten independent *funding streams* which collectively represent the same amount. The change is meant to increase transparency of how the reward is split between investors.

NCC Group verified that the proposed changes are appropriately specified:

- The proposed formulas stop the original Founders' Reward and start the funding streams at exactly the same time on chain height 640,000 (this is the scheduled activation height of the Blossom upgrade).
- The funding streams stop at the same time as the original Founders' Reward was supposed to stop, i.e. the first halving.
- The sum of the subsidy fractions for the funding streams in any post-Blossom block is *exactly* the same as the founders' reward in pre-Blossom blocks. Notably, all funding stream fractions have a denominator which is a divisor of 400, and the per-block subsidy is a multiple of 400 (when expressed in zatoshis), so all divisions are exact and no zatoshi is lost in rounding.
- The formulas properly take into account the consequences of the change of target block spacing, as defined in ZIP 208 (see next section). In particular, the first halving will still happen at the same physical time as initially scheduled.
- Each funding stream contains twelve addresses; each address is used for a period of roughly one month. The period is defined by the block height. The definition in ZIP 207 is exact: The time from the switch to the funding streams to the first halving time splits in exactly twelve complete periods, with no partial period.

NCC Group offers the following remarks:

Consensus rule on funding streams: ZIP 207's consensus rules stipulate that a block is acceptable only if the following property is fulfilled: *"The coinbase transaction in each block MUST contain at least one output per active funding stream that pays the stream's value to the stream's recipient address for the block's height."* This does not *stricto sensu* rule out the case of a coinbase transaction which pays *more* than the stream value to a given stream's recipient address. However, the example code in ZIP 207 enforces an exact match:

```
    for (const CTxOut& output : block.vtx[0].vout) {
        for (auto it = requiredStreams.begin(); it != requiredStreams.end(); ++it) {
            if (output.scriptPubKey == it->first && output.nValue == it->second) {
                requiredStreams.erase(it);
                break;
            }
        }
    }
```

The draft specification also mandates an exact match (section 7.9, page 93): "The transaction MUST include at least one output that pays exactly FundingStream$_i$.Value(height) zatoshi with...". Also, the current (pre-Blossom) rule on the Founders' Reward also uses an exact match.

NCC Group therefore suggests to amend the ZIP 207 wording from "that pays the stream's value" into "that pays exactly the stream's value" in order to match the draft specification and the example code.

Typographical error: In the draft specification, section 3.10 (page 17), a space is missing between "transaction" and "MUST."

[3] https://github.com/zcash/zips/blob/c7e79e2909b4de618743d35d84b2b2d5fb884b7d/zip-0207.rst

ZIP 208

ZIP 208[4] defines a change in the target block spacing. In the current Zcash network, the difficulty threshold for the proof-of-work is dynamically adjusted to the observed issuance rate so that, *on average*, new blocks are mined at a given nominal rate of one block every 150 seconds. In order to increase transaction throughput and lower confirmation latency, ZIP 208 increases the rate to one block every 75 seconds. This change impacts all processing that uses the block height as a time scale.

NCC Group verified that ZIP 208 takes into account the new target block spacing in the difficulty adjustment formulas and the founders' reward (which is at the same time replaced with the funding streams, as described in the previous section):

- Along the new scale, the first halving, when per-block subsidies are halved and the funding streams stop operating, will happen at block height 1,060,000 (instead of 850,000 in the previous scale). Subsequent halvings will happen every 1,680,000 blocks, exactly twice as much as in the previous scale, corresponding to the doubled block issuance rate.
- Per-block subsidy is halved, since there are twice as many blocks per time period. The overall rate of creation of new ZECs is unchanged. The funding streams are expressed as fractions of the per-block subsidy, and thus keep being 1/5th of the created ZECs.

Difficulty adjustment: The difficulty is adjusted with an automatic mechanism that increases or decreases the proof-of-work (PoW) threshold based on an *averaging window* of 17 blocks: If the last 17 blocks were issued "too fast" (as per the target spacing), then the difficulty is increased (i.e. the threshold to meet with Equihash is lowered); similarly, if the blocks were issued "too slowly," then the difficulty is lowered. A "damping factor," that divides the difficulty adjustment on each block, and maximum change boundaries, are applied to ensure that the change in difficulty is gradual and does not enter a chaotic feedback loop.

The protocol's basis of time is the `nTime` field of block headers: This is the time at which the miner started working on that block and expressed along the miner's own notion of the current time. Since there is no mechanism in the Zcash protocol to maintain clock synchronization, individual nodes may use different times. Three mechanisms are used in Zcash to cope with such variations:

- Whenever a physical time is inferred from the `nTime` fields of blocks, no single block header is used; instead, the *median* time of 11 successive blocks is used.
- A consensus rule is that a block is not acceptable unless its `nTime` is strictly greater than the median time obtained from the 11 previous blocks.
- A property of `zcashd` (Zcash's official implementation) is that it rejects blocks whose `nTime` is more than two hours (7200 seconds) in the future, as per the local node's own notion of the current time. This is not a consensus rule, since different nodes may use different notions of the current time; moreover, as time passes, an unacceptable block may become acceptable.

As will be detailed below, these properties mean that miner nodes with maliciously or accidentally misconfigured clocks may induce undue difficulty changes, but only up to that which can be achieved within a two-hour window and subject to the gradual changes implied by the damping factor. Such actions would constitute only a weak form of denial-of-service, artificially decreasing the frequency of issuance of new blocks by a relatively low factor, and for a limited amount of time (after four hours, the normal rate should be restored).

ZIP 208 does **not** change the main parameters of the difficulty adjustment algorithm. In particular, median times are still computed over 11 blocks, and the averaging window is still 17 blocks. This implies that the adjustment algorithm's behavior is unchanged when expressed in blocks: a given increase or decrease in total network mining power will imply the same adjustment in difficulty, spread over the same number of blocks. Since block issuance is increased, the *physical time* taken to enact an adjustment is reduced.

[4] https://github.com/zcash/zips/blob/eb297d2454488d10fa3d76d582b2b11ba02f2e4e/zip-0208.rst

However, this conservation of behavior (with a 2x speeding up factor, in physical time) is not complete, because the 7200-second maximum clock drift is not modified. Consider the case of a mining node which, through malicious action or configuration accident, has its internal clock two hours "in the future". Since nodes use median time over 11 successive blocks, a single block with a wrong `nTime` will not be of much consequence. However, if 6 of the last 11 blocks are two hours in the future, then the next block will necessarily also be two hours in the future, because the mining nodes will adjust the `nTime` they produce to match the consensus rule expressed above (in `zcashd`, this is done by the `UpdateTime()` function[5]). In that case, all subsequent blocks will be issued with sub-second apparent hashing times until physical time catches up with this artificial two-hour drift. During that time, the adjustment algorithm will believe that block issuance is way too fast, and ramp up difficulty as much as is allowed by the damping factor and maximum adjustment boundaries. With the reduced target block spacing, more blocks will be issued in these two hours, allowing for a comparatively larger artificial difficulty inflation.

Numerical simulations show that in the situation above (six successive new blocks have `nTime` two hours in the future), the maximum difficulty inflation factor after two hours is 1.95 when target block spacing is 150 seconds; with the new proposed target block spacing of 75 seconds, this inflation factor increases up to 3.24. In that sense, the reduced target block spacing, with the two-hour rule unchanged, increases the magnitude of the difficulty adjustment response when faced with six successive blocks with incorrect timing. In physical time, though, the inflation still occurs over the same time (two hours) and is resorbed as quickly (less than two extra hours, since the boundary on difficulty reduction is less stringent).

In order to keep the exact same adjustment dynamics (when counted along the height time scale) as in the pre-Blossom network, the two hours rule would have to be reduced by the same factor as the target block spacing, i.e. down to one hour (3600 seconds). However, this rule was implemented in order to cope with accidental clock drifts, in particular misconfiguration of daylight-saving time changes, and such accidents will not decrease in magnitude after the Blossom upgrade. As such, NCC Group does not recommend reducing the acceptance delay. If the increased difficulty inflation implied by the Blossom upgrade is deemed unacceptable, then possible mitigations include the following:

- The number of blocks used for median time computations could be increased, so that more deviant miner clocks can be tolerated.
- A higher damping factor will reduce the reactivity of the difficulty adjustment algorithm, and therefore the inflation in case of a clock drift event. Note that the reduced target block spacing mechanically makes the adjustment algorithm *more* reactive when measured in physical time; there is thus some margin for damping factor modifications.

Blossom Upgrade Transition: At the exact block on which the Blossom upgrade is applied (block 640,000), the target block spacing suddenly changes. The difficulty adjustment algorithm will react to it as if the collective mining power was suddenly halved. This is explained in ZIP 208 in these words: "The change in the effective value of PoWTargetSpacing will cause the block spacing to adjust to the new target, at the normal rate for a difficulty adjustment." As explained above, this "normal rate" is unchanged when counted in blocks. In physical time, the rate is suddenly faster (unless the damping factor is also increased). NCC Group suggests amending the wording in ZIP 208 into: "[...] at the normal rate for a difficulty adjustment with post-Blossom parameters."

Median Time Formula: The formula for extraction of a median value in a set, in section 7.6.3 of the draft specification (page 86), is:

$$\text{median}(S) := \text{sorted}(S)_{\text{ceiling}(\text{length}(S)/2)}$$

This formula is not defined if S is an empty set. In the formulas of section 7.6.3, when the height parameter is greater than 0 but less than PoWAveragingHeight, the Threshold(height) call indirectly invokes MedianTime() with parameter height − PoWAveragingWindow, i.e. a negative value. In that case, a median computation is performed over an empty set, which is not defined.

[5] https://github.com/zcash/zcash/blob/59e82c64e94504036239a994424bb88d72c4434a/src/miner.cpp#L99

In other words, the formulas, as stated in the specification, do not provide a meaningful threshold value for blocks with height 1 to 16, inclusive. These are some of the very first blocks issued, and thus already far in the past, and existing implementations apparently deal with them properly; therefore, this formal issue does not seem to have any tangible consequence. NCC Group nonetheless suggests fixing the formulas to match what the `zcashd` implementation does in that situation.

As a related note, the implementation of median time extraction in `zcashd` is the following (from `chain.h`[6]):

```
enum { nMedianTimeSpan=11 };

int64_t GetMedianTimePast() const
{
    int64_t pmedian[nMedianTimeSpan];
    int64_t* pbegin = &pmedian[nMedianTimeSpan];
    int64_t* pend = &pmedian[nMedianTimeSpan];

    const CBlockIndex* pindex = this;
    for (int i = 0; i < nMedianTimeSpan && pindex; i++, pindex = pindex->pprev)
        *(--pbegin) = pindex->GetBlockTime();

    std::sort(pbegin, pend);
    return pbegin[(pend - pbegin)/2];
}
```

Right now, this conforms to the specification as long as there are indeed 11 blocks to inspect. However, if `nMedianTimeSpan` were to be modified to an even value, or if `GetMedianTimePast()` were called on an ancient chain with an even number of blocks less than 11, the computation would return the "wrong" value. With `nMedianTimeSpan` equal to 11, the formula in the specification uses index 6, while the C++ code uses index 5: These values match, since the specification starts indexing at 1, while C++ array indexing starts at `0`. However, if `nMedianTimeSpan` were set to `10` instead, the C++ code would still use index 5 (with a start index of `0`) while the formula in the specification would also use index 5, but with a start index of 1.

NCC Group suggests amending the specification to match what the code implements. Since the code uses an odd value for `nMediaTimeSpan`, the discrepancy has no impact except for chain heights less than `10`.

The issues described above, related to median time computations, are not new to the Blossom upgrade; they were already present in previous protocol versions.

[6] https://github.com/zcash/zcash/blob/59e82c64e94504036239a994424bb88d72c4434a/src/chain.h#L283