A Graceful Hash Transition For The Bitcoin Protocol (WIP)

Rusty Russell <rusty@rustcorp.com.au>

May 2013

Abstract

The Bitcoin protocol currently uses SHA-256 and RIPEMD-160 as its workhorse hashing protocols, in particular using SHA-256 for generating proofs-of-work[1 Satoshi]. Though SHA-256 is widely endorsed and has no known weaknesses[2], this may change over time. This paper proposes a gradual method for transitioning away from SHA-256 in the active Bitcoin network, assuming that the vast majority of miners and users are willing to upgrade. This reduces the problem of deciding the initial difficulty of the new hashing algorithm, and avoids the economic disruption which a sudden transition away from SHA-256 would cause. It is also the only clearly compelling reason to making incompatible changes to the bitcoin protocol[3 - https://en.bitcoin.it/wiki/Hardfork_Wishlist], so serves as a useful vehicle for considering hard forks.

1 Background

[SHA for mining, double-SHA for merkle hashes, SHA for bitcoin addresses [https://en.bitcoin.it/wiki/Technical_background_of_Bitcoin_addresses]

Mining difficulty, timestamps, difficulty adjustment. every 2016 blocks.

Satoshi on hash transition: [https://bitcointalk.org/index.php?topic=360.msg3520#msg3520]

2 Likely Timeline of SHA256

Theoretical weakening: impractical.

Practical weakening: seen in wild, or published

Duplicate hashes can be created.

Duplicate hashes can be created for any hashed value.

Massive weakening

Broken

3 Replacing SHA-256 in merkle hashes

FIXME: Trivial cutover for new blocks.

4 Replacing SHA-256 in Bitcoin Addresses

New prefix "2"? How to handle old bitcoins?

5 Replacing SHA-256 in Proof of Work

FIXME: Discussion of ASICS, investment, and thus why sudden transition will be resisted until it's too late, and why a return to CPU mining is likely to result in a less secure network (botnets et al).

This paper proposes a gradual transition over a number of difficulty changes (each one nominally 2 weeks long). For simplicity the examples and graphs use 26 transitions (ie. 1 year), but this is fairly arbitrary (see 6.2).

The amount time spent on the new hashes changes over S difficulty changes: the old hash is solved as before, but that forms the basis for a new hash (with a new nonce) which must also be solved. Both new and old hash have separate difficulties, balanced such that the time spent on the new hash is $\frac{1}{S}$ at first, $\frac{2}{S}$ next difficulty change, and so after S difficulty changes the new hash is used entirely and the transition is complete.

Unfortunately this scheme, while simple, has a problem: we cannot isolate the time taken on the old and new hashes separately. These can be expected to vary over the transition period due to both technical and other factors:

1. As the value of old hashing equipment drops, it may not be replaced if it breaks.

2. At some point the electricity cost of old hashing will make it unprofitable; this may happen simultaneously for several large miners.

3. Previous experience has shown that ASIC delivery is rarely on time, so predicting the increase in new hashing power is fraught,

4. The SHA2 algorithm may be significantly weakened during the transition, making old mining much easier.

In addition, estimating the initial new hash difficulty is problematic. Satoshi based the bitcoin initial difficulty on his own hashing rate, but we can be certain that more than one miner will exist in this transition. Problems include:

1. This initial hash rate estimate must be agreed by all, so it's best if it's not too critical (eg. can be selected by the lead developer without friction).

2. Miners may not disclose their new hashing capability to gain competitive advantage.

3. A new hash (eg. SHA3) may see rapid speedups due to intense development brought by using it for bitcoin mining.

4. There must be a delay between setting the initial hash rate and the start of the transition. During this time, any estimate could become obsolete.

5. Miners with a vested interest in the status quo may actively sabotage the new hash difficulty selection.

5.1 Alternate Difficulty Adjustment Solution

The proposed solution is simple, yet provides robustness against various scenarios. On odd difficulty adjustments (ie. 1st, 3rd, etc) we adjust the difficulty of both hashes the same way as done now: if blocks arrived 10% too rapidly, both new and old hashing is made 10% more difficult:

$NewHashTarget_{N+1} = NewHashTarget_N \frac{Duration_N}{1209600}$

$OldHashTarget_{N+1} = OldHashTarget_N \frac{Duration_N}{1209600}$

(Note that the difficulty adjustments are always clamped to a factor of 4, as in the current protocol).

On even adjustments, we don't adjust the old hash target at all:

$OldHashTarget_{N+1} = OldHashTarget_N$

Instead, we assume the old hash took exactly the right fraction of two weeks:

$OldTime_N = 1209600 \frac{S-N}{S}$

Hence the remainder of the duration is assumed to be taken up by the new hash:

$NewTime_N = Duration_N - OldTime_N$

As the ideal duration for the new hash would be $\frac{N}{S}$ of two weeks, we correct the new hash target by this ratio:

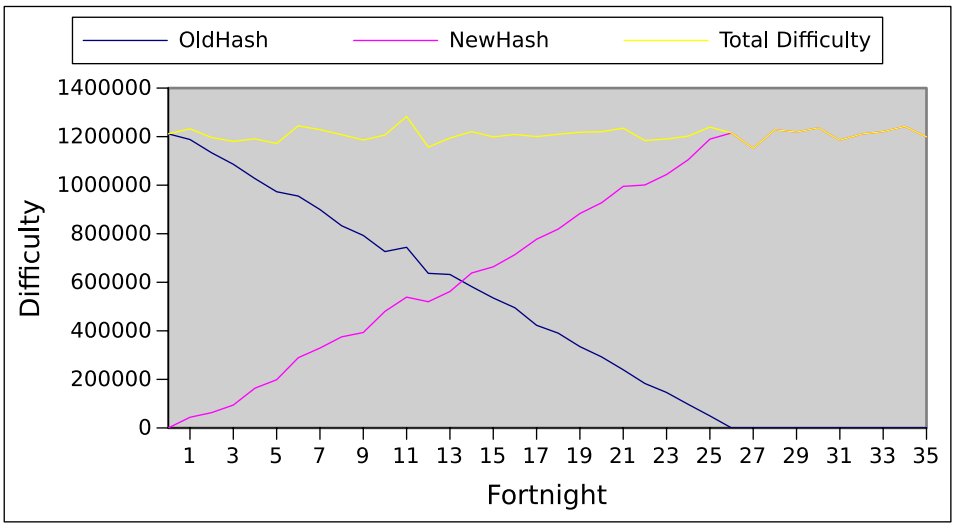
$NewHashTarget_{N+1} = NewHashTarget_N \frac{NewTime_N S}{1299600N}$

Again, this is clamped to within a factor of 4 of the last difficulty to avoid dramatic changes.

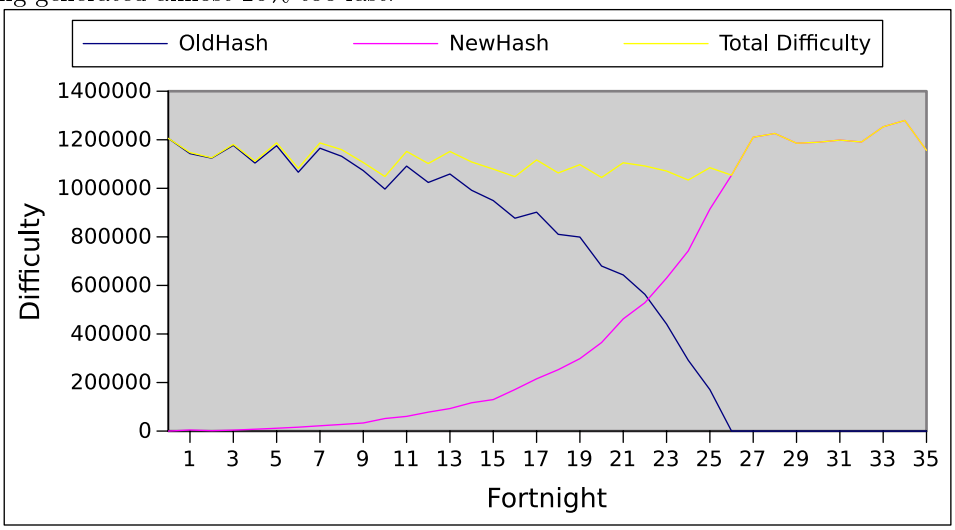
5.1.1 Results In Different Scenarios

A rough simulator was written, to explore different scenarios. For simplicity (and ease of graphing), the new hash and old hashes are assumed take the same amount of time, and transition occurs in 26 steps (S=26).

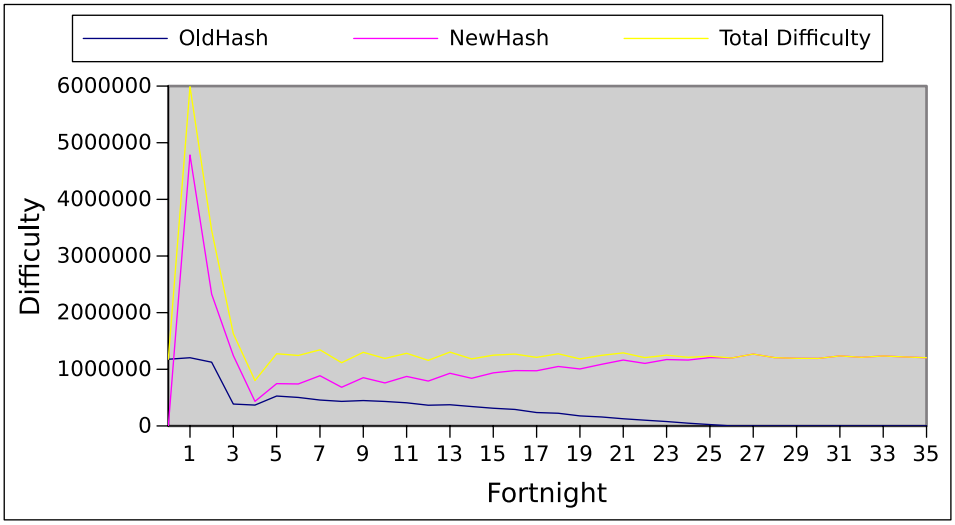
Firstly, if the initial calibration is correct, the time spent on the new hash ramps up, and time on the old hash ramps down, as expected, with the time spent on both staying constant:



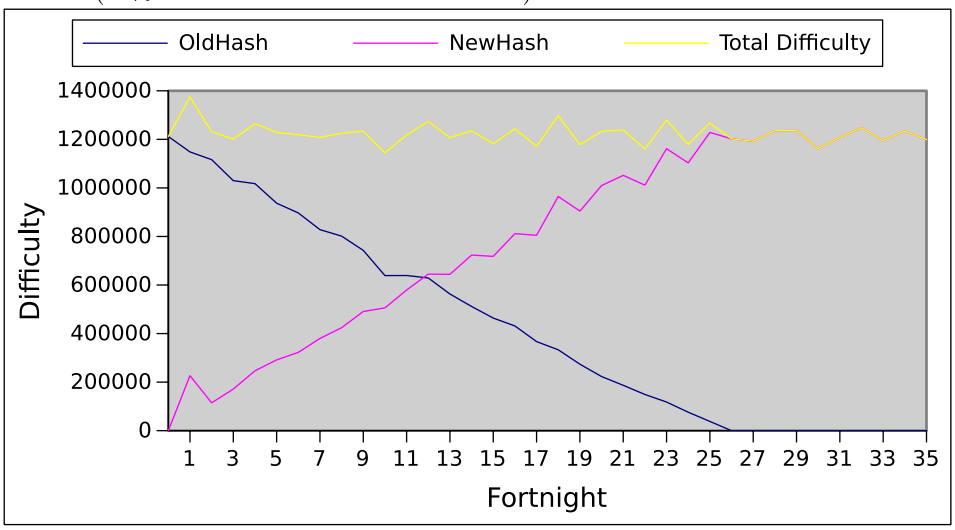
If the initial difficulty of the new hash is too easy by a factor of 100, results are still reasonable, as the new hash ramps up to adjust. At worst, blocks are being generated almost 20% too fast:



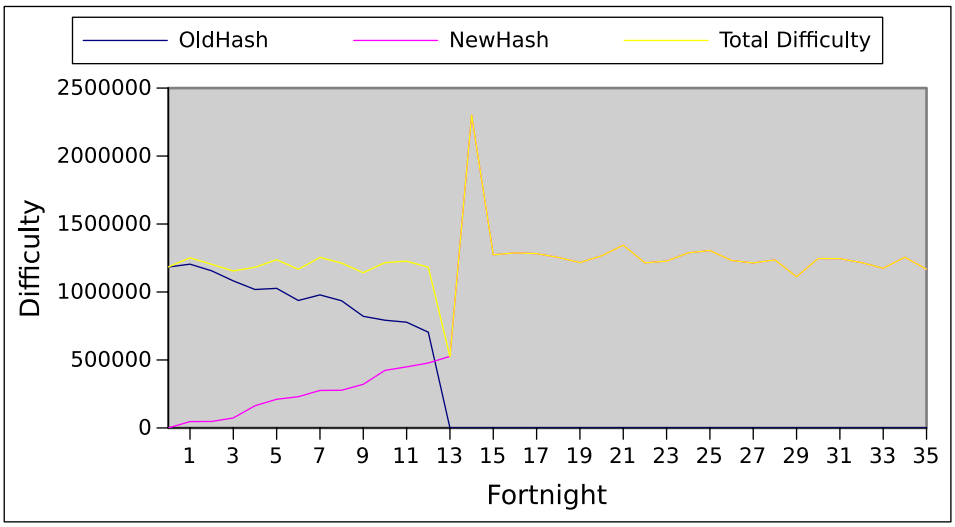
If the initial difficulty of the new hash is too hard by a factor of 100, results are poor (460% too slow on the initial introduction), but still recover quickly:



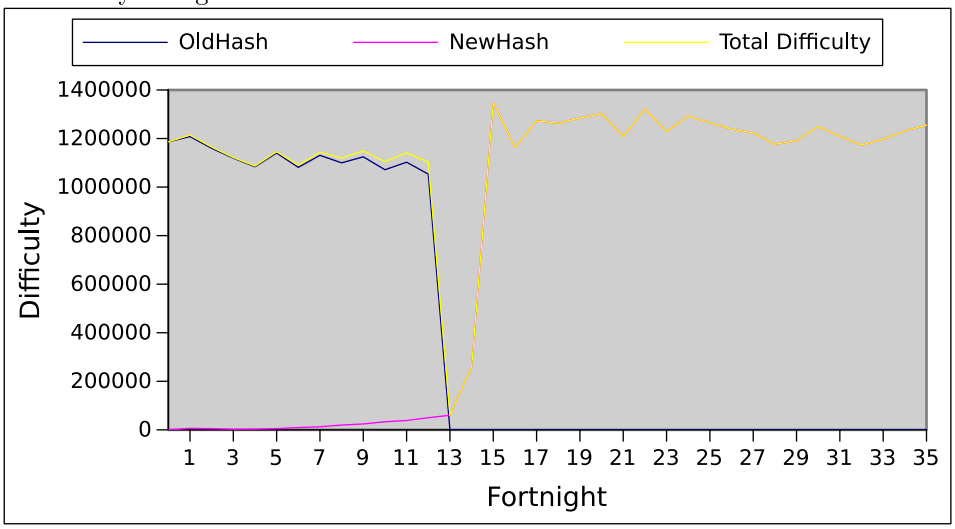
If the new hash difficulty is only too hard by a factor of 5, results are more reasonable (15% too slow on initial introduction):



To measure the case where SHA256 weakens during transition, we test the most radical case (a complete break) by setting difficulty to zero halfway through (N=13, S=26). We immediate produce blocks 50% too fast (as expected), then overadjust to produce them 95% too slowly, but then remain within 10% of the expected rate:



Finally, combining the "100 times too easy" case with the catasrophic break of SHA256 halfway through, we can see that the new hash has not ramped up sufficiently at the half-way mark: it is only 5% of the hashing power, so blocks are produced 2000% faster than usual (every 30 seconds or so) and it takes two more difficulty changes to correct.



5.2 Effect of Dramatic Reduction in Hash Difficulty

Let's assume an attacker secretly breaks SHA during the transition period, such that they can now freely generate old-hash blocks. To preform a "51% attack" they need to be able to solve the new hash faster than the rest of the network can solve both hashes. For example, once the new hash is taking over half the block generation time, they need 1/3 of the network hashing power to double-spend

at will.

6 Recommendations

6.1 Initial Hash Rate

Assuming we want to remain within 20% of a 10-minute block time, the initial difficulty setting should reflect between 1% and 500% of the real hash rate. It's better to be too low (eg. 0.1% of the real hash rate, thus have blocks too fast) than too high (eg. 5000% of the real hash rate, thus having blocks 3 times too slow).

A higher initial hash rate provides more protection against SHA2 failing (only if the network can maintain that hash rate though).

A hashing competition prior (using real bitcoins as a reward) could be used to establish the initial hash rate.

6.2 Transition Period

I've used a 1-year transition period, but a period which matches the expected hardware lifecycle of dedicated mining hardware (eg. 3 years) might be more economically rational. Obviously a slower transition means a longer exposure to any SHA2 weakening, but it's more robust to start a long transition early than to delay and have a faster transition.