Epoch and Difficulty

Sub-epoch: Sub-epoch N starts when sub-epoch N-1 ends (except for 0th sub-epoch), and it ends at the end of the first slot where 384 * (N+1) blocks have been included since genesis.

Epoch: Epoch N starts when epoch N-1 ends (except for 0th epoch), and it ends at the end of the first slot where 4608 * (N + 1) blocks have been included since genesis.

Difficulty: A constant that scales the number of iterations for a given proof of space. Iterations are computed as difficulty / quality.

Every 4608 blocks, the difficulty adjustment is automatically performed. This modifies two parameters: The slot_iterations parameter, and the difficulty parameter.

The sub_slot_iterations parameter is reset so a 600-second slot requires close to slot_iterations many iterations. The reset is done using the values from the last epoch to approximate the iterations-per-second ratio, concretely.

We'll define `epoch$` as the period beginning with the last block that was infused *prior* to the current epoch, and ending with the last block that was infused *in* the current epoch. Thus, `epoch$` is a slightly shifted period that occurs for each epoch.

The values `t1`,`i1` and `w1` denote the timestamp, iterations (since genesis), and weight (since genesis) at the beginning of `epoch$`. Along the same lines, `(t2, i2, w2)` are the values at the end of `epoch$`.

Here's how we calculate iterations per second:

```
iterations per second = floor(num iterations in last epoch / duration of last epoch)
                      = floor((i2 - i1) / (t2 - t1))
```

That is, the delta in total iterations from the start to the end of the epoch, divided by the delta in timestamps.

Sub-slot iterations is the total number of iterations per ten-minute sub-slot. Signage point interval iterations is sub-slot iterations divided by 64 (the number of signage points per sub-slot).

```
sub slot iterations = iterations per second * 600
sp interval iterations = floor(sub slot iterations / 64)
```

Note that we don't take the iterations and time exactly at the end of an epoch, but at the last infusion point of a block in an epoch (aka `epoch$`), the reason being simply that we only have timestamps available when blocks are infused.

```
weight/sec of last epoch = (new weight - old weight) / duration of last epoch
                         = (w2 - w1) / (t2 - t1)


new difficulty = (weight/sec * target seconds) / target number of blocks
               = ((w2 - w1) / (t2 - t1) * (4608/64) * 600) / 4608
```

This can be rearranged to use only one floor division:

```
new difficulty = floor(75 * (w2 - w1) / (16 * (t2 - t1)))
```

The sub-slot iterations are adjusted such that each slot lasts around 600 seconds. The difficulty is adjusted such that every challenge gets 32 blocks on average with fewer iterations than slot_iterations.

It is important to note that the VDF iterations per slot is not material to the weight. That is, if there were two identical worlds where VDF speeds were equal and space was equal, but the sub-slot iterations parameter was twice as high in one world, then the blockchain with the higher sub-slot iterations would get twice as many blocks included per slot, but each slot would take twice as long. The weight per second added to the chain would be the same in both cases.

Another way to look at it is that increasing sub-slot iterations increases the number of blocks per slot, but it also makes slots last longer, and thus has no effect on weight per second.

Sub Epochs

The challenge chain is completely separate and does not refer to anything in the rewards chain. If these chains stayed separate forever, an attacker with a faster VDF would be able to look into the far future and predict challenges. The attacker could create one block per slot, with limited space, thus creating a whole challenge chain. This would allow them to create plots and instantly create proofs of space for these plots that will win in the future, and then delete the plots (a long range replotting attack). This would enable them to fill their reward chain and increase their weight.

The solution to this is to periodically (every 384 blocks, which is an average of 2 hours) infuse the reward chain into the challenge chain. This means that the attacker can only perform the replotting attack for a few hours into the future. Plotting takes some time, but even if the attacker could replot instantly, the cost of a replotting attack will outweigh the benefits. This is because we do not infuse the *current* reward chain output; instead we infuse the *previous* sub-epoch's reward chain output (two hours in the past).

The cost of creating a plot includes the electricity to calculate all of the tables, the RAM necessary while creating this plot, and the fixed infrastructure costs (space, power, cooling, etc). Assuming the worst case scenario of a super fast VDF, and instant ASIC plotting - the benefits would be equivalent to the benefits of storing that plot on a HDD for a few hours. Note that this would not guarantee a winning plot; it would be the equivalent of storing a normal plot.

It is clear that this attack is not worthwhile, and that storing the plots is much cheaper.

The above explains why the sub-epoch interval should be kept relatively low. But why can't we further reduce it to lower than 2 hours to further disincentivize replotting attacks? The reason is that whenever non-canonical data is infused into the challenge chain (which the reward chain contains), an opportunity for grinding occurs. This means an attacker can possibly choose to include or exclude blocks to manipulate what the challenge will be 2 hours into the future. If this time is too short, they can gain a small space advantage by doing this more often.

Revision #2
Created 5 June 2023 14:43:32 by Admin
Updated 6 June 2023 07:30:59 by Admin