THE KIMOTO GRAVITY WELL

Dr. Kimoto Chan

creates

MEGACOIN & THE KIMOTO GRAVITY WELL

And Kimoto said,

```
from pylab import *

one_day = 1440 / 2.5  # how many 2.5 min blocks per day

nmin = one_day / 4    # PastBlocksMin
nmax = one_day * 7    # PastBlocksMax
x = arange(nmin, nmax)  # PastBlocksMass

# start with 2.5 minute blocktime + some noise
t0 = 2.5 + randn(nmax) / 4
t1 = 2.5 + randn(nmax) / 4

# t0 has 20% more hashrate, so shorter blocktime in the beginning
t0[:one_day] = 2.5 / 1.2 + randn(one_day) / 4
# t1 has higher blocktime in the beginning
t1[:one_day] = 2.5 / 0.9 + randn(one_day) / 4

s = arange(nmax)

adjust0 = (arange(1, nmax + 1) * 2.5 ) / cumsum(t0)
adjust1 = (arange(1, nmax + 1) * 2.5 ) / cumsum(t1)

# the magic function
def kimoto(x):
return  1 + (0.7084 * pow((double(x)/double(144)), -1.228));

plot(x/one_day, kimoto(x))
plot(x/one_day, 1/kimoto(x))
plot(s/one_day, adjust0)
plot(s/one_day, adjust1)
legend(["kimoto","1/kimoto", "20% increase in hashrate", "10% drop in hashrate"])
xlabel("days")
ylabel("adjustment factor (target/actual blocktime)")
show()
```

and there was MEGACOIN.

www.megacoin.co.nz

"This is what makes time-dilated difficulty formulas possible: the Kimoto Gravity Well! It's taken me nearly sixty six years and my entire family fortune to realize the vision of that day. KGW = 1 + (0.7084 * pow((double(PastBlocksMass)/double(144)), -1.228)); "

Dr. Kimoto Chan

#1 The Newbie's Guide to the Kimoto Gravity Well

Many of you may have heard of Kimoto's Gravity Well and that it is supposedly a major part of what makes Megacoin unique from other cryptocurrencies. However, many of you may also not know what exactly it is and what makes it so special. If that is the case, then this guide is for you.

What Is a Mining Difficulty Readjustment Algorithm, Anyway?
To understand what the Gravity Well algorithm is and what it does, you first need to understand what a "mining difficulty readjustment algorithm" is and why is it important for all current cryptocurrencies based off of the original Bitcoin source code. First, let's pull a few important definitions from the Bitcoin wiki:

Difficulty
Difficulty is a measure of how difficult it is to find a new block compared to the easiest it can ever be.

Difficulty Readjustment (for Bitcoin)
The difficulty is adjusted every 2016 blocks based on the time it took to find the previous 2016 blocks. At the desired rate of one block each 10 minutes, 2016 blocks would take exactly two weeks to find. If the previous 2016 blocks took more than two weeks to find, the difficulty is reduced. If they took less than two weeks, the difficulty is increased. The change in difficulty is in proportion to the amount of time over or under two weeks the previous 2016 blocks took to find.

So basically, the "difficulty" of a coin determines how hard it is for miners to find and mint blocks of that coin. The more miners there are mining a coin, the faster blocks will be found and at the end of this difficulty readjustment period (approximately every two weeks for Bitcoin), the difficulty will change accordingly so that the number of coins minted will follow the intended distribution curve. This has worked well for Bitcoin (so far) because of it's extremely slow adoption rate in the early days and now because of the sheer number of miners on the network. However, this method of difficulty readjustment is flawed for new altcoins entering the market today for a number of reasons which I will discuss below.

The History of the Gravity Well Mining Difficulty Readjustment Algorithm
When Megacoin first launched, it used a more traditional difficulty readjustment algorithm based off of Bitcoin's original proposal. (*Author's note: I have forgotten what the original implementation was for Megacoin, but if anyone knows the details please let me know so I can put that here for perspective and history's sake.)* By this time, some SHA-256 coins had already felt the pain of difficulty readjustment problems due to the influx of ASIC miners and an activity known as "pool-hopping".

If you are familiar with cryptocurrency mining at all, you may already know that in most cases, solo mining is usually impossible without extremely powerful hardware due to the large number of people now aware of cryptocurrencies and willing to mine for them. Most miners mine through pools, which provide proportional payouts of coins based on the amount of hashing power you provide to the network. This mitigates some of the risk of mining in that you receive a steady stream of coins based on your network hashing rate, so even small-time miners can still earn their share of the pie. However, as pool mining became more popular and more altcoins arrived on the market, services known as "multipools" began to appear. These were special pools that allowed miners to automatically switch to the "most profitable" coin to mine based on the current exchange rates. However, these new multipools introduced some new problems to the cryptocurrency landscape, one of those being major difficulty readjustment woes.

As Megacoin began to rise in price several months after its inception, it started to become a target for these multipools. What happens when this occurs is that suddenly the Megacoin network gets barraged by an influx of new (and very powerful) miners. This causes the block confirmation time to plummet and subsequently causes the difficulty to skyrocket at the next difficulty readjustment. When this occurs, the mining profitability also drops due to the higher difficulty which then in turn causes all of the multipool miners to leave the network in search of the next most profitable coin. What remains is an extremely high difficulty and only the "core" group of Megacoin miners left to deal with the aftermath. In extreme cases, the difficulty may be so high in proportion to the number of miners left that the entire network grinds to a halt. This has happened in the past to Terracoin and Feathercoin, among others. The only solution if this occurs is to hard fork the coin in an attempt to readjust the difficulty (or change the difficulty readjustment algorithm) or simply grind out the mining at an extremely slow pace (during which time the coin is basically unusable) until enough blocks are found to make it to the next difficulty readjustment. The more blocks required until the next difficulty readjustment, the longer this period of unusability will be, and in some cases could mean the death of the coin completely unless drastic measures are taken.

When this happened to Megacoin, Kimoto decided to come up with a better way to perform difficulty readjustment, and the result is the Kimoto Gravity Well (which is now also used as the difficulty readjustment algorithm for Anoncoin as well after it met a similar fate as that described above). And thus, we have the Megacoin we know and love today. Next I will discuss what exactly the Gravity Well does and how it works to keep mining stable and fair for all Megacoin miners and users.

Gravity Well: Explained
Now that you know how the Gravity Well came to be, let's take a look at what exactly it does and how it works. At the most basic level, Kimoto has changed how difficulty readjustment works so that the difficulty is adjusted after every single block that is mined on the network. I'm not 100% sure about the exact mathematics behind the calculations, but so far since its introduction on the network the difficulty has adjusted smoothly and flawlessly no matter how many miners there are on the network and even throughout the huge price (and subsequent mining hash rate) increase we have seen over the past couple of weeks. This keeps

mining fair and secure for all miners and users of the coin, and prevents the rampant multipool abuse that was (and still is) common with most all other altcoins out on the market today. This is even more important to consider when one day ASIC miners are developed for Scrypt coins and a small number of miners will suddenly have access to extremely powerful mining hardware. If and when this occurs, a malicious (or simply greedy) miner can simply point his or her ASIC miner at any Scrypt-based coin and cripple it because of the extreme difficulty fluctuation this will cause. (This is actually what happened with Terracoin after SHA-256 ASICS began to flood the market.) Megacoin, however, will be safe from this type of malicious mining behavior due to the smooth difficulty readjustment that Kimoto's Gravity Well provides.

Hopefully this will act as a guide for new investors to Megacoin who may have heard about Gravity Well but are not quite sure what it means or what it even is. If any of you have anything else to add to this, please post! Information is power.

www.megacoin.co.nz

#2 The Gravity Well

```
unsigned int static KimotoGravityWell(const CBlockIndex* pindexLast, const CBlockHeader *pblock, uint64
TargetBlocksSpacingSeconds, uint64 PastBlocksMin, uint64 PastBlocksMax) {
	/* current difficulty formula, megacoin - kimoto gravity well */
	const CBlockIndex  *BlockLastSolved                 = pindexLast;
	const CBlockIndex  *BlockReading                    = pindexLast;
	const CBlockHeader *BlockCreating                   = pblock;
	                    BlockCreating                   = BlockCreating;
	uint64              PastBlocksMass                  = 0;
	int64               PastRateActualSeconds           = 0;
	int64               PastRateTargetSeconds           = 0;
	double              PastRateAdjustmentRatio         = double(1);
	CBigNum             PastDifficultyAverage;
	CBigNum             PastDifficultyAveragePrev;
	double              EventHorizonDeviation;
	double              EventHorizonDeviationFast;
	double              EventHorizonDeviationSlow;

   if (BlockLastSolved == NULL || BlockLastSolved->nHeight == 0 || (uint64)BlockLastSolved->nHeight < PastBlocksMin)
{ return bnProofOfWorkLimit.GetCompact(); }

	for (unsigned int i = 1; BlockReading && BlockReading->nHeight > 0; i++) {
		if (PastBlocksMax > 0 && i > PastBlocksMax) { break; }
		PastBlocksMass++;

		if (i == 1)	{ PastDifficultyAverage.SetCompact(BlockReading->nBits); }
		else		{ PastDifficultyAverage = ((CBigNum().SetCompact(BlockReading->nBits) - PastDifficultyAveragePrev) /
i) + PastDifficultyAveragePrev; }
		PastDifficultyAveragePrev = PastDifficultyAverage;

		PastRateActualSeconds			= BlockLastSolved->GetBlockTime() - BlockReading->GetBlockTime();
		PastRateTargetSeconds			= TargetBlocksSpacingSeconds * PastBlocksMass;
		PastRateAdjustmentRatio			= double(1);
		if (PastRateActualSeconds < 0) { PastRateActualSeconds = 0; }
		if (PastRateActualSeconds != 0 && PastRateTargetSeconds != 0) {
		PastRateAdjustmentRatio			= double(PastRateTargetSeconds) / double(PastRateActualSeconds);
		}
		EventHorizonDeviation			= 1 + (0.7084 * pow((double(PastBlocksMass)/double(144)), -1.228));
		EventHorizonDeviationFast		= EventHorizonDeviation;
		EventHorizonDeviationSlow		= 1 / EventHorizonDeviation;

		if (PastBlocksMass >= PastBlocksMin) {
			if ((PastRateAdjustmentRatio <= EventHorizonDeviationSlow) || (PastRateAdjustmentRatio >=
EventHorizonDeviationFast)) { assert(BlockReading); break; }
		}
		if (BlockReading->pprev == NULL) { assert(BlockReading); break; }
		BlockReading = BlockReading->pprev;
	}

	CBigNum bnNew(PastDifficultyAverage);
	if (PastRateActualSeconds != 0 && PastRateTargetSeconds != 0) {
		bnNew *= PastRateActualSeconds;
		bnNew /= PastRateTargetSeconds;
	}
   if (bnNew > bnProofOfWorkLimit) { bnNew = bnProofOfWorkLimit; }

   /// debug print
   printf("Difficulty Retarget - Kimoto Gravity Well\n");
   printf("PastRateAdjustmentRatio = %g\n", PastRateAdjustmentRatio);
   printf("Before: %08x  %s\n", BlockLastSolved->nBits, CBigNum().SetCompact(BlockLastSolved-
>nBits).getuint256().ToString().c_str());
   printf("After:  %08x  %s\n", bnNew.GetCompact(), bnNew.getuint256().ToString().c_str());

	return bnNew.GetCompact();
}
```

#3 An In Depth look at the Kimoto Gravity Well

The goal is to have a more adaptive way of adjusting the difficulty instead of just averaging the last 2016 blocks like bitcoin. This is needed because of multipools which might switch the coin they are mining, and a sudden change in hashrate can occur (both increasing or decreasing). Especially when a multipool switches away you get stuck too long with a too high difficulty.

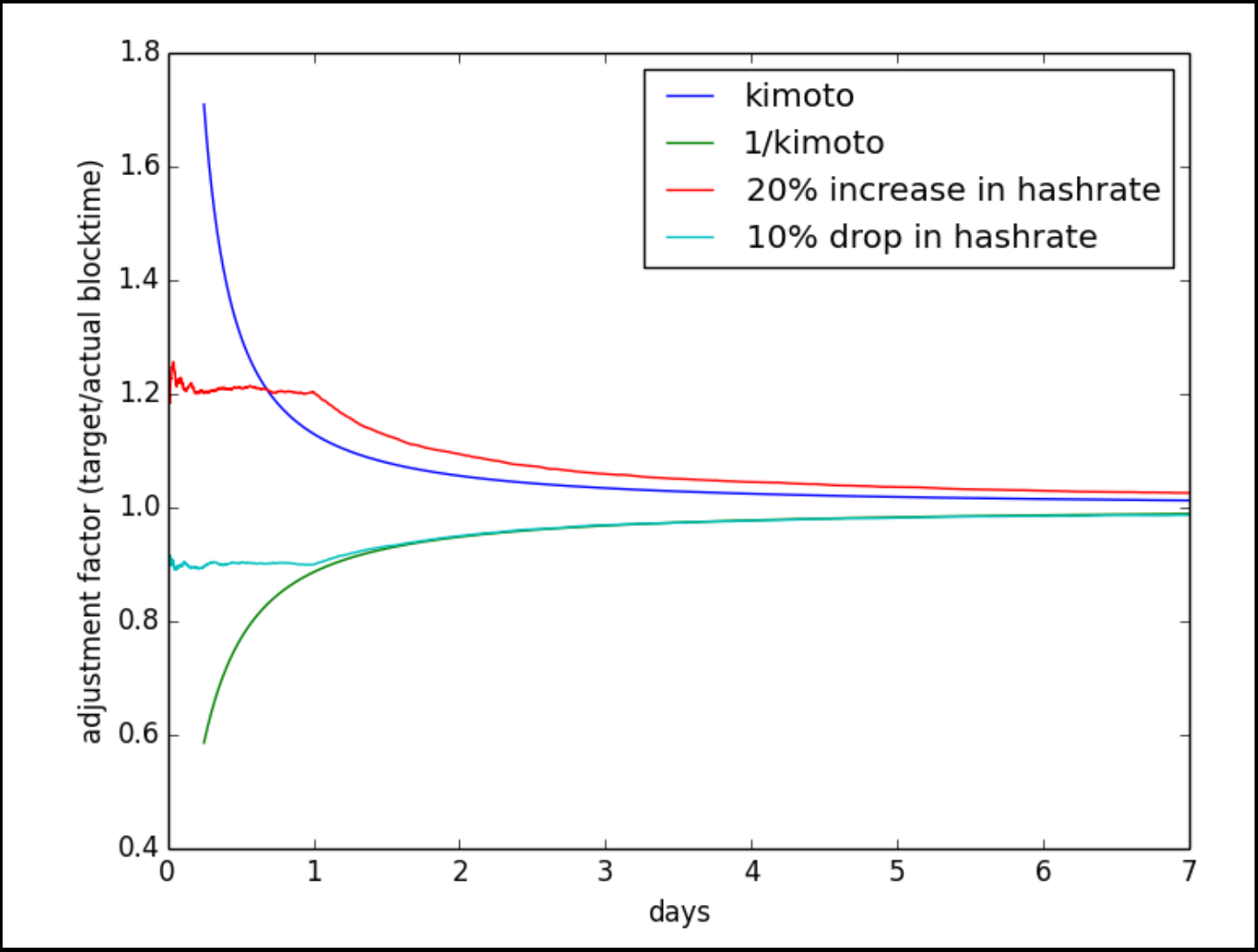
The algo loops backwards through the blocks, starting from the current one. The PastBlocksMass is just the number of blocks, so it starts at one and increases in each loop.

In each loop an adjustment factor is computed, which is the target block time divided by the actual block time, in a cumulative fashion, so at loop 10 we would have the 25 minutes target time divided by the time it actually took to compute the last ten blocks. When the hashrate increases, we get shorter times and an adjustment factor greater than one and vice versa.

The loop ends whenever the average adjustment factor is larger than the kimoto-value, or smaller than 1/kimoto-value. To understand this look at this python script and an example plot:

```
from pylab import *

one_day = 1440 / 2.5   # how many 2.5 min blocks per day

nmin = one_day / 4     # PastBlocksMin
nmax = one_day * 7     # PastBlocksMax
x = arange(nmin, nmax)   # PastBlocksMass

# start with 2.5 minute blocktime + some noise
t0 = 2.5 + randn(nmax) / 4
t1 = 2.5 + randn(nmax) / 4

# t0 has 20% more hashrate, so shorter blocktime in the beginning
t0[:one_day] = 2.5 / 1.2 + randn(one_day) / 4
# t1 has higher blocktime in the beginning
t1[:one_day] = 2.5 / 0.9 + randn(one_day) / 4

s = arange(nmax)

adjust0 = (arange(1, nmax + 1) * 2.5 ) / cumsum(t0)
adjust1 = (arange(1, nmax + 1) * 2.5 ) / cumsum(t1)

# the magic function
def kimoto(x):
    return  1 + (0.7084 * pow((double(x)/double(144)), -1.228));

plot(x/one_day, kimoto(x))
plot(x/one_day, 1/kimoto(x))
plot(s/one_day, adjust0)
plot(s/one_day, adjust1)
legend(["kimoto","1/kimoto", "20% increase in hashrate", "10% drop in hashrate"])
xlabel("days")
ylabel("adjustment factor (target/actual blocktime)")
show()
```

The script produces a figure like this Kimoto gravity well.



It shows two constructed examples when the hashrate increases and when it drops for one day.

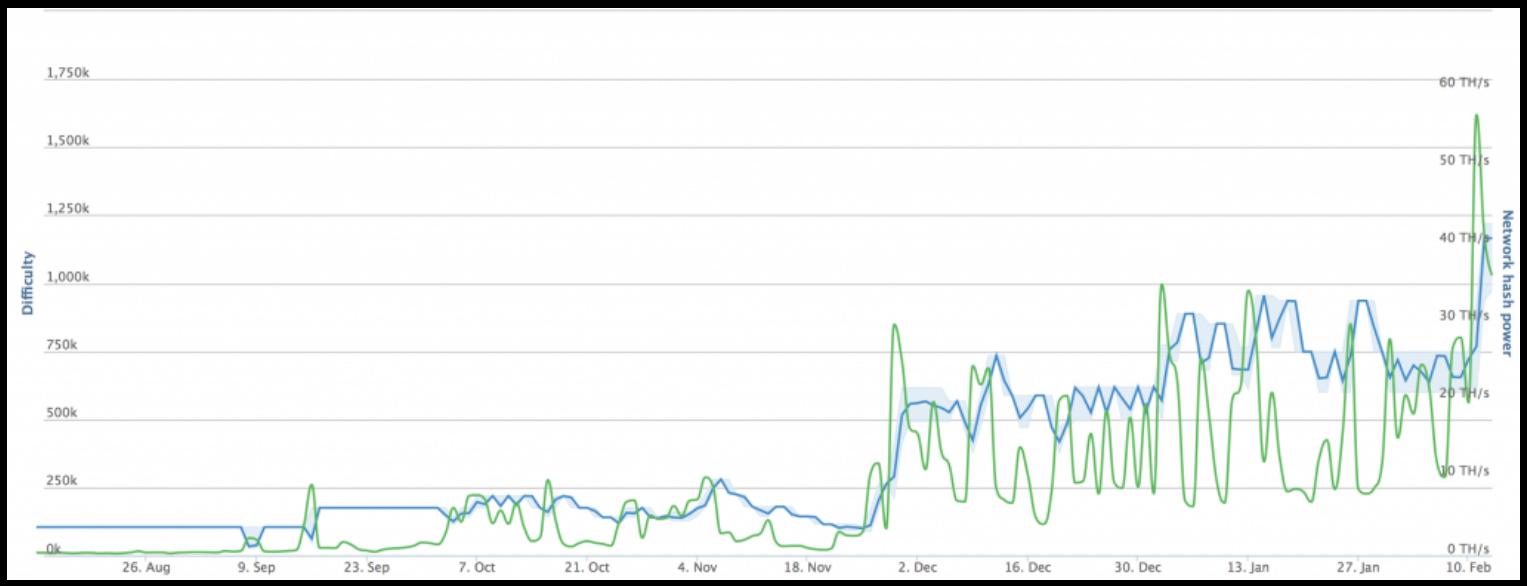
You see that the lines cross through the Kimoto formula at some point. This is when the algo exits and uses this adjustment factor to compute a new target/difficulty. For large adjustment factors this happens earlier than for the ones closer to one. This is to have a quick adaptation if the hashrate changes a lot, and a slower one if not - then we want a longer period to get a better average. The parameters of the Kimoto formula are adjusted in such a way that one roughly adjusts in one day to a 10% change and in seven days to a 1.2% change. A minimum of 144 blocks determine the new difficulty and at most 4032 (0.25 days or 7 days for a 2.5 minute blocktime).

What happens when the minimum of 144 blocks isn't reached or the maximum of 4032 blocks is exceeded?

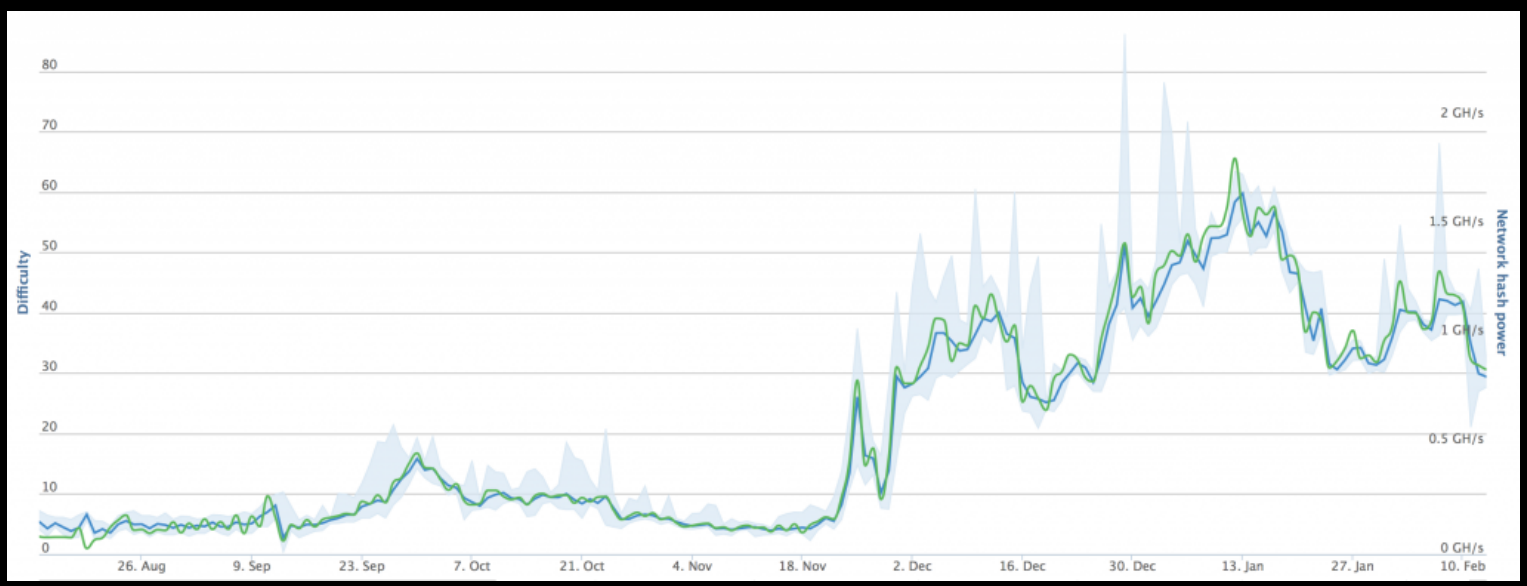
The minimum is 144 blocks. So if the adjustment factor is larger than 1+0.7084, then the factor for 144 blocks is taken. At 4032 the algo stops and the average for 4032 is taken no matter what. KGW gives an answer for both boundary cases.

#4 The result

Look at the results of such a decision by the example of the three graphs below. The first two - Feathercoin and Terracoin respectively. Blue lines show the level of difficulty, and green - the computing power of the network. As you can easily notice, a spike in Difficulty is always preceded by a sharp increase in computing power of the network due to the "raid" by multipools, after which the network capacity decreases and difficulty for some time, remains unchanged.



As you can see in the third Megacoin graph, changes occur more smoothly, thus making mining and difficulty rate more predictable and stable in the long run.



by Alienwalkerx and Dr. Kimoto Chan