QuarkCoin: Noble Intentions, Wrong Approach

One of the more interesting alternative cryptocurrencies to have come out of the dungeon that is the Bitcointalk altcoin forum is something called “Quark” (or sometimes “QuarkCoin”) – a cryptocurrency that aims to be “super secure” by employing multiple advanced hash algorithms in place of Bitcoin’s plain SHA256. The currency has acquired significant interest in the past few weeks, especially so because it has been picked up not just inside the Bitcoin community, but also among two mainstream figures: Bill Still, an American journalist, film producer and author responsible for the films *The Money Masters* and *The Secret of Oz*, and Max Keiser, host of a show with millions of viewers on the Russia Today TV network. So what is this new coin that seems to have so suddenly come out of nowhere, and why has it succeeded in getting so much attention where even Peercoin and Primecoin have failed?

Quark differs from Bitcoin in three key ways: its proof of work algorithm, its block interval and its distribution model. The proof of work algorithm in any (Bitcoin-like) cryptocurrency is the function that miners must compute in order to create valid blocks; in Bitcoin, for example, a valid block must have a SHA256 hash starting with `000000000000000000` (that’s fifteen hexadecimal zeroes). Because SHA256 is essentially a pseudorandom function, the only way to create such a block is to keep trying, making an average of 2^{60}attempts, before you eventually create a block that is valid. Artificially making block creation so hard is a security measure; it ensures that attackers will not be able to flood the network with illegitimate blocks, and therefore a fraudulent transaction history, without having more computing power than the entire legitimate network combined.

The proof of work algorithm in Quark is more complicated than Bitcoin, but not excessively so; instead of using just one hash function as Bitcoin does, Quark uses six: BLAKE, Blue Midnight Wish, Groestl, JH, Skein and Keccak. The six algorithms are implemented in series, with nine steps; three of the steps randomly apply one of two out of these six functions depending on the value of a bit. The point of this is twofold. First, it is intended to make Quark more resistant against the “black swan” risk of a single hash function getting cracked. Second, it is intended to make the currency secure against specialized hardware or even GPUs; “Being only CPU mined,” the introduction reads, “this coin offers the average individual the rewards of mining.”

The block interval in Bitcoin is 10 minutes, meaning that the "difficulty" (ie. the number of zeroes that a valid proof of work must have its SHA256 hash start with) automatically adjusts so that the network produces one block per ten minutes. In Quark, the interval is an ambitious 30 seconds. The distribution model in Bitcoin is an exponential decay model: for the first 210000 blocks (~4 years), 50 BTC is released per 10 minutes, for the next 210000 blocks 25 per 10 minutes, then 12.5 per 10 minutes, and so on in an exponential decay until eventually issuance will stop entirely in 2140. Quark's issuance model is a similar exponential decay, but much faster; it starts off at 2048 QRK per block for three weeks, then 1024 for three weeks, and so on until it reaches 1 QRK per block after about seven months. Unlike Bitcoin, however, Quark then stays at 1 QRK per block forever – a "permanent linear inflation" model whose inflation rate will start off at 0.5%.

So What Are The Flaws?

Unfortunately, although Quarkcoin tries to make a number of bold and daring improvements on the Bitcoin parameters, it arguably fails in its objectives on almost every count. We can go through the various changes that Quarkcoin made, and see that almost each and every one of them either does a substandard job of doing what it is intended to do or even introduces problems of its own.

First, the hashing algorithms. As described above, the intent of having six hash algorithms is (1) to protect against "black swan" attacks on hash algorithms, and (2) to make the coin unfriendly to specialized hardware. The first purpose seems reasonable at first glance; if one of the six hash functions get cracked, that particular block will always be found instantly, and the other five hash functions will remain standing. However, the way that it was done manages to be simultaneously superfluous and inadequate.

One thing that must be understood about hash functions is that, unlike most public-key algorithms, hash functions are often very opaque in their implementations, relying on complicated permutations and arbitrary substitutions and transforms rather than elegant mathematics involving modular exponentiation or elliptic curve points. The design of hash functions attempts to maximize properties known as diffusion, confusion and nonlinearity – essentially, professional cryptographers literally come together and try to figure out how to make a function as opaque and jumbled up as possible so that no one, including the cryptographers themselves, can figure out what's going on inside.

As a result, hash functions tend to naturally have many built-in redundancies, and it shows. When the MD5 hash function was cracked, it went down slowly. In 1993, researchers first found a "pseudo-collision" – two changes to an internal parameter called an initialization vector that lead to the same output. In 1996, researchers found a "collision" – two inputs that produce the same output – to one specific internal component of MD5, the compression function. It was not until 2004 that these insights were converted into a full collision attack on MD5 itself. Even today, MD5 is actually not fully broken; althoush collisions, finding `X` and `Y` such that `MD5(x) = MD5(y)`, can be done in only a million computational steps, pre-images, or finding `X` such that `MD5(x) = Y` for a prespecified `Y`, still take over 2^{100}steps (although no longer quite the initial 2^{128}. Hence, hash functions like SHA256 are already highly redundant and black-swan proof. In fact, if a critical black swan event does occur, it will likely be something like P=NP or quantum computing that affects all hash functions at once.

Furthermore, there is one place where the algorithm does not use redundant hash functions: the Merkle tree. Quark's Merkle tree still uses good old SHA256. What's more, an attack on Bitcoin's Merkle tree does not even

need the harder pre-image attack – a mere collision attack, albeit a highly specialized one, will suffice to make a double spend and even fork the entire network. The process is simple: make two transactions, `T1: A -> B` and `T2: A -> A`, such that `hash(T1) = hash(T2)`. Publish `T1`. Then, publish `T2` later and spread around blocks containing `T2` in place of `T1`. Now, suppose B tries to spend the bitcoins that he received in a transaction `T3`. Some nodes, which have `T1`, will see that `T3` is spending the bitcoins from `T1` and thus recognize `T3` as legitimate, and eventually a miner will make a block, B1, containing `T3`. Other miners, that have `T2`, however, will see `T3` as invalid because it is spending bitcoins that have been sent to A, and thus reject `B1`. They will eventually make a new block `B2` without `T3`. From there, the blockchain will split in half, with some blocks following `B1` and others following `B2`. All this requires only a relatively simple collision attack against SHA256, and Quark does nothing against this.

The second application of Quark's multi-hash mechanism is its resistance to ultraefficient mining through specialized hardware. However, the combined hash function created by composing BLAKE, groestl and the other functions does not have any particularly special properties; it is simply a hash function which takes up nine times as many lines of code. Producing specialized hardware devices (ie. ASICs) for mining it will certainly take nine times as much work, but once they exist they will be every bit as efficient as Bitcoin ASICs. They only do not exist now because there is not enough interest in Quark. A better way to make an ASIC-resistant coin is to use so-called "memory-hard hash functions" – functions that take a large amount of memory, as well as time, to calculate, so devices that attempt to do the computation millions of times in parallel will need to have petabytes of memory on hand; Litecoin does this, as does Primecoin with its sieve-based prime number chain algorithm, albeit unintentionally.

Next, we come to Quark's 30 second block time. The idea of making blockchains with faster confirmations is a seductive one; Litecoin started the trend with its 2.5-minute blocks, Primecoin has 1-minute blocks, and now even faster coins like Krugercoin exist with 15-second blocks. Although accepting payments with any of these currencies is equally near-instant, as confirmations are not really required for security in most applications, such currencies have definite advantages in high-security applications such as gambling sites and depositing to exchanges. However, below roughly one minute such currencies run into two problems. First, there is the issue of "stale" blocks – when a miner finds a valid block, it takes about twelve seconds for that block to propagate through the network, and if any other miners find a valid block in those twelve seconds their work is essentially wasted. With a 10-minute block time, this is only a two percent decrease in de-facto network security. With a 2.5-minute block time, it becomes eight percent and with a 1-minute block time it becomes about 17% – significant, but far from fatal. Below one minute, however, these stales start to seriously threaten the security of the network.

The second issue is one of centralization. Suppose that miners are now organized into mining pools, where one mining pool necessarily has more market share than the others. Suppose this top mining pool has a 25% market share, and its next competitor is at 15%, and the baseline stale rate is 33%. Solo miners have 67% efficiency due to the stale rate. The 15% mining pool, however, itself mines the block 15% of the time, and so starts immediately working on the next block without delay. Hence, the 15% pool's stale rate is only 33% * 0.85, or 28% – or 72% efficiency. The 25% pool enjoys 75% efficiency. Thus, new miners have an incentive to join the largest pool, making it even more powerful. This effect inevitably leads to large centralized mining pools which, combined with the reduces network security, means that one mining pool will almost certainly have de-facto

control over the entire network. With Bitcoin and its 2% stale rate, this is not a significant issue. With Primecoin at 17%, this is a moderate concern. With Quark, this is arguably a fundamental flaw.

One possible solution to the first problem comes from a recent paper by Yonatan Sompolinsky and Aviv Zohar which suggests a way to actually include stale blocks as part of the blockchain; if a new currency integrated this, it may be able to address the security loss potentially down to even below a 10-second block time. However, the solution does not fully address the second concern; if these stale blocks that are part of the chain receive their mining rewards, then the currency becomes much weaker because there is no longer any incentive not to support fraudulent double-spending attacks, and if the stale blocks do not receive their rewards the efficiency centralization issue remains. Perhaps with some compromise, however, 10-30 second block times can be made viable in the future.

Finally, we have the distribution model. 50% of all Quark units were distributed within three weeks, a much steeper distribution curve than Bitcoin or any other cryptocurrency except perhaps Mastercoin and Ripple. Many have come to call his model a de-facto premine, “premine” being the technical term for when a currency is created with a number of units already in the hands of some centralized party. The Quark developers have made a post addressing this concern, saying that their currency is more fairly distributed than any other altcoin, showing that the percentage of all Quark units owned by the top 100 addresses (59%) is actually right in the middle of those of other major cryptocurrencies (cf. Namecoin at 56%, Litecoin at 48% and Peercoin at 64%).

However, this is somewhat misleading; of all of these other currencies, the percentage in question is the percentage out of those coins that are already in circulation. In the case of Litecoin, Peercoin and Namecoin, there are still many millions of currency units left to be distributed – and there is no particular reason why these new currency units should go to the same early adopters who were lucky enough to secure entire percentages of the currency’s money supply earlier in its life cycle. With Quark, the currency will take 100 years for its money supply to expand by 50%, so the 59% is not likely to go down by much any time soon. Ultimately, it is very hard to create a currency with a fair distribution model, and any specific technical approach will likely only last a few months before people with money and resources catch up to it at least to some extent. The only stable approach may be to simply premine it and then give out 0.001% to the first 100000 people that show up with a passport or at least a unique cell phone number.

The concerns that Quark seeks to address – Bitcoin’s slow block times, the unfairness of altcoin distribution and the threat of specialized hardware – are all very valid issues, and Quark’s recent success highlights the importance of these problems. However, Quark in its current state is not the solution. Litecoin and Primecoin are both very valid alternatives that seek to target many of the same goals, and have done so much more successfully. Hopefully, in time an even better cryptocurrency will be developed that will actually have all of the desired properties that we currently want; and ultimately every attempt and even every mistake made along the way is a stepping stone to achieving that goal.

Vitalik Buterin is a co-founder of Bitcoin Magazine who has been involved in the Bitcoin community since 2011, and has contributed to Bitcoin both as a writer and the developer of a fork of bitcoinjs-lib, pybitcointools and multisig.info, as well as one of the developers behind Egora. Now, Vitalik's primary job is as the main developer of Ethereum, a project which intends to create a next-generation smart contract and decentralized application platform that allows people to create any kind of decentralized application on top of a blockchain that can be imagined.