THE HIVE: AGENT-BASED MINING IN LITECOIN CASH

Version 1.0

Iain CRAIG, Sebastian CLARKE, Michał WYSZYŃSKI, Federico DE GONZÁLEZ-SOLER, Thomas BEASLEY, Kasper Brandt HANSEN, Ilya HELLE, Joseph Paul MCMAHON, Sylvia TYNELSKA and Maciej ZAMS

Litecoin Cash Developers, London

Abstract. *Hive mining,* an agent-based block minting priority contest, is proposed as a means to help secure a cryptocurrency blockchain. Hive mining increases network security over pure proof-of-work, providing protection from 51% attacks. Proof-of-work miners can continue mining as usual with no idle time, while a highly democratised alternative mining solution is available to those without specialised mining hardware. We introduce the idea of on-chain agents, *worker bees,* which act as virtual mining rigs with a negligible energy cost.

1. Background and motivation

1.1 Alleviating the risk of 51% attacks

In common with several other alternative cryptocurrencies[1],[2], Litecoin Cash (LCC) in 2018 experienced a 51% attack[3], which can allow a bad actor who controls more than half the available network hashing power to attempt to double-spend a transaction.

This highlights a fundamental peril of a cryptocurrency with relatively small market share secured only by proof-of-work. In general, any such coin secured with a given block hashing algorithm is highly susceptible to a 51% attack, since it is expected that the total amount of hashpower available across all coins using the same hashing algorithm will dwarf the amount of hashpower typically available on the marginal coin[4].

In the case of Litecoin Cash, this risk is exacerbated by the fact that the hashing algorithm is SHA256. As this is the algorithm used by Bitcoin[5], the world number one cryptocurrency, a vast amount of SHA256 hashpower exists[6].

1.2 Avoiding alienation of existing miners and pools

Litecoin Cash's existing community is based on a strong network of SHA256 miners, and LCC began life as "the SHA256 fork of Litecoin"[7]. Therefore, it is important that any proposed solution enables SHA256 proof-of-work miners to continue unhindered, without any behaviour changes, and without needing to have mining hardware sitting idle at any point.

1.3 Democratising the mining process

It is desirable that those without dedicated mining hardware should still be able to help secure the network and earn the associated rewards, with negligible energy cost.

2. Introduction to hive mining

In a classical proof-of-work blockchain security scheme, miners compete by computing a large number of potential block hashes in an effort to find one which meets a difficulty target set by network consensus[5].

It is a worthwhile thought experiment to consider what would happen in this classical scheme if the mining difficulty was just reduced to zero and *any* block hash was accepted as valid by the network.

With the “work” part of proof-of-work now effectively bypassed, any core node on the network would be able to mine blocks easily.

This initially sounds quite attractive; democratised mining with a very low energy cost. However, in practice, everyone would mine these very cheap blocks and push them onto the network simultaneously, resulting in many potential candidate blocks to extend the chain. As miners would not know which candidate block they should pick to build on in order to extend the chain, many short orphan chains would result[5],[8].

While the network may reach some sort of lurching consensus, it would certainly involve a lot of small jumps forward, chain reorganisations, and cancelled transactions. This situation will be familiar to anyone who has observed the launch of a pure proof-of-work coin with an inadequate mining difficulty adjustment algorithm.

With difficulty at zero, and no cost element to producing a block, not only can the network not tell which candidate chain is worth more and should therefore have priority, but miners can also work on multiple chains at once with no penalty[9].

The point of this thought experiment is to demonstrate simply that the main purpose of proof-of-work, proof-of-stake, or indeed proof-of-*anything*, is to give the network a deterministic way of deciding *who has the right to mine/mint/forge the next block* that can be agreed on by all participants. A further requirement of this block contest, often fulfilled as a side-effect of the main act of proving, is that a miner should not be able to work on multiple chains at once without incurring a cost penalty[5],[9].

Hive mining is an alternative form of block contest, whereby the right to produce a block is secured by an agent working on behalf of a user.

These “worker bee” agents reside inside the blockchain itself. They are completely decentralised and are created when a special agent creation transaction is made by a user.

Once created, worker bees act as virtual mining rigs, and users who own them become “bee keepers”. When worker bees successfully mine blocks, the block rewards (block subsidy and any transaction fees) are paid to the bee keeper. Worker bees have extremely low energy cost, and so bee keepers do not require any specialised hardware to be able to produce blocks.

Worker bees have a limited lifespan, and creating a bee is a speculative action with an associated cost; this penalises adversaries attempting to work on multiple chains at once. The success of a given bee depends largely on the population of live bees across the network, although some bees may never find a block, while others may be disproportionately lucky.

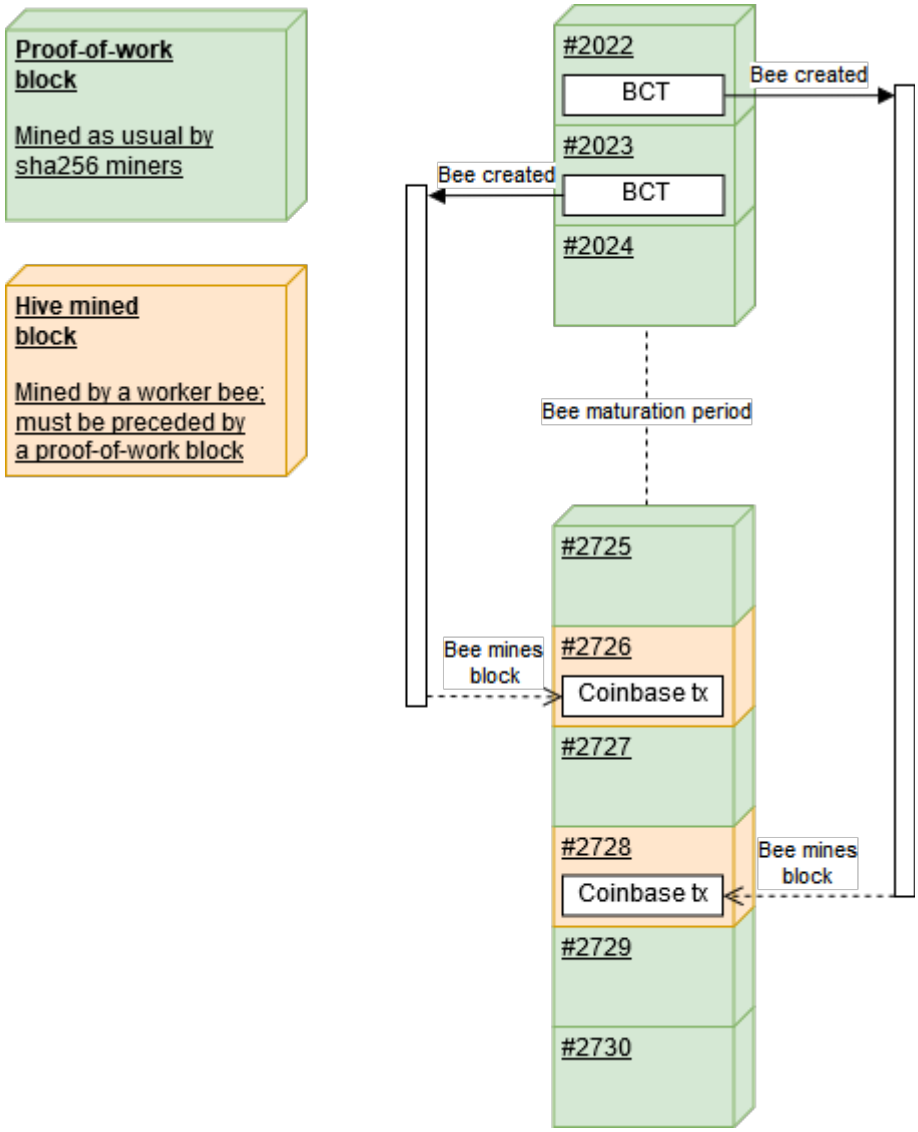

Fig 1: A worker bee added to the blockchain via a bee creation transaction (BCT) is able to mine blocks in the future for the duration of its lifespan

The concept of Hive mining, in which bees are created by destruction of coins, bears some similarity to previously employed methods for initial coin distribution whereby users demonstrating a burn of coins on one chain are allocated coins on a new chain. It is also related to Iain Stewart's proposed "Proof of Burn" concept[13], which attempts to secure a network in broadly similar way, but here the philosophy is inverted somewhat; our implementation is driven by a desire to *create agents* rather than solely to prove coin destruction.

3. Creation of worker bee agents

To create a worker bee, a user sends a transaction to a special unspendable address; e.g. CReateLitecoinCashWorkerBeeXYs19YQ. Note that everybody uses the same bee creation address.

The bee creation address parses as a valid address, but it's reasonably provable that nobody has a private key for it; the vanitygen utility estimates that to find a private key that gives a human-readable vanity address with that many deliberate characters would take 24 * 2 Ghz cores at least $1.7 * 10^{31}$ years (for a 50% chance of success).

A bee creation transaction (BCT) must have at least 1 output, which both pays a fixed bee creation fee to the unspendable address and encodes the *honey address* that will receive any block rewards earned in the future by bees created in this BCT.

Note that the honey address could be set by the user creating the bee if desired, defaulting to a new address generated by their wallet if none is specified. Optionally, additional output(s) in the BCT receive change if the potential bee keeper's wallet needs to use an overpaying input to pay the bee creation fee.

The first output of the BCT, vout[0], contains both the script pub key for the unspendable address, and the bee keeper's honey address:

```
unspendableBeeCreationScriptPubKey OP_RETURN OP_BEE honeyScriptPubKey
```

It is proposed that the required fee per bee is expressed as a percentage of current block reward. This calculation would include a hard minimum cost tuned so that by the time block rewards have ceased because all coins are issued, blocks would still be worth hive mining for transaction fees alone.

There is no need for a BCT to only purchase a single bee; instead, a BCT can be assumed to have created TOTAL_PAID / BEE_COST bees.

Furthermore, it would be possible to allow the user to optionally divert some of the funds that would otherwise have been destroyed to a community fund, to allow for future development costs.

It is very important that a user opting to divert otherwise destroyed funds to the community fund only be able to send at maximum a small percentage of the overall cost; otherwise, there is a risk that the development team (or whoever controls the community fund) would be able to create free bees.

It is also vital that a bee keeper receives exactly the same number of bees for the same overall cost, whether all creation fees are to be destroyed or some are to be diverted to the community fund.

For example, if a user destroying 100 coins creates 1000 bees, then destroying 90 coins while diverting 10 to the community fund must also create 1000 bees for the user.

An example BCT might look like this (only relevant fields shown; inputs must still be signed, etc, as usual):

```
  "vout": [
    {
      // BCT fee to unspendable addr; also encodes beekeeper's honey address
      "value": 90.0000000,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 64bef321f2d4e128f4f26016476eb425165d90ba OP_EQUALVERIFY
OP_CHECKSIG OP_RETURN OP_BEE OP_DUP OP_HASH160 3a9365c404bd7748f84a40213329da00bf7959b5
OP_EQUALVERIFY OP_CHECKSIG",
      }
    },
    {
      // Optional fund diversion to comm fund. Value combined with vout[0] to find beecount
      "value": 10.0000000,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 c9f2c053d8d9f5ab9f02deabcd58bfda774f4746 OP_EQUALVERIFY
OP_CHECKSIG",
        "type": "pubkeyhash",
        "addresses": [
          "CashCFfv8CmdWo6wyMGQWtmQnaToyhgsWr"
        ]
      }
    },
    {
      // Bee keeper's change address
      "value": 3485073.2997539,
      "n": 2,
      "scriptPubKey": {
        "asm": "OP_HASH160 3e9e71ca7cda38afc433228cac25e0e2df7fb628 OP_EQUAL",
        "type": "scripthash",
        "addresses": [
          "MDcFwEtTvMn81AhnaRiSZzsVTp29jD1jq5"
        ]
      }
    }
  ]
```

Once the bee creation transaction is accepted by the network, it appears with a fixed transaction ID in a block with a fixed height.

Bees only mature and become able to mint blocks when 1152 blocks have passed since the bee creation transaction. This is the expected number of new blocks to be added to the Litecoin Cash blockchain in 24 hours.

After maturation, bees have a lifespan of 16128 blocks (the expected number of blocks in 1 week) during which they have a chance to mint blocks. Therefore, when they reach a depth of 17280 blocks, they die.

The act of creating a BCT will be encapsulated in a new RPC call in the core wallet. In the QT wallet, the user would have a new tab, "The Hive", to show a table of the user's current bees. For each bee, this would show creation date, time to mature / time left to live, and mining results. The Create Bee interface would also be placed on that tab. A screenshot of the Hive tab is shown below:

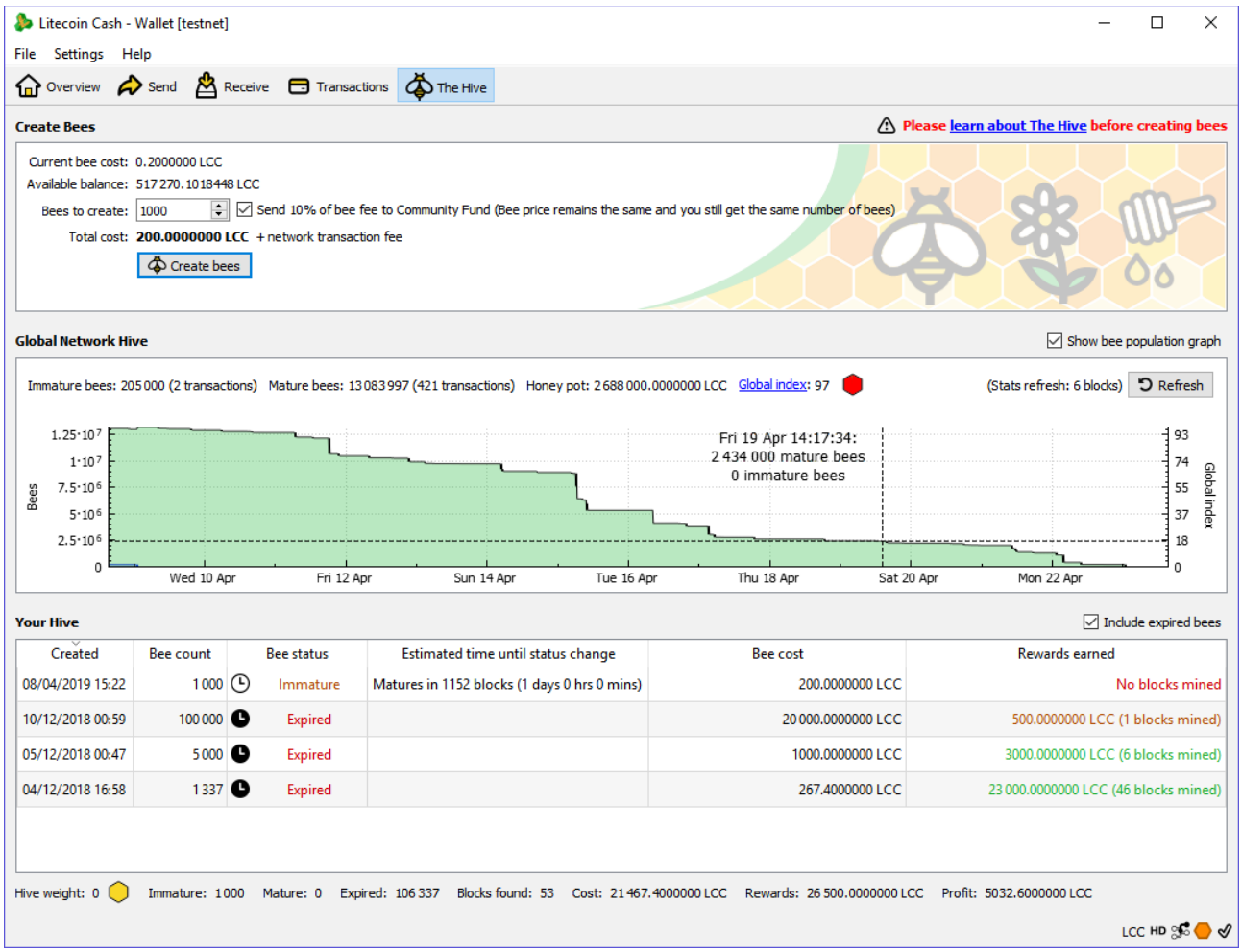

Fig 2: The Hive tab in the QT wallet

After maturation, bees have a lifespan of 16128 blocks (the expected number of blocks in 1 week) during which they have a chance to mint blocks. Therefore, when they reach a depth of 17280 blocks, they die.

Note the display of a “Global Index” metric. Because the potential hivemining block rewards available to the entire network over a given time period are fixed, there is a maximum bee population that can be sustained by the network while still expecting all bees to be profitable (ie, to return more block rewards than the cost of the bees).

The Global Index gives bee keepers a simple way to make a value judgement call before creating bees.

When Global Index is below 100 (green is still visible in the GI pie chart), all live bees are expected to be profitable.

When Global Index is equal to or greater than 100 (only red is visible in the GI pie chart), there are more bees than can be expected to be profitable, and it may be wise to wait for the population to decrease before creating new bees.

4. Bees at work: The minting process

4.1 Block creation

By way of example, let us suppose that the current blockchain height is 1565621, and the network needs to determine which bee gets to mint block 1565622.

We'll examine the process from the point of view of Alice, a bee keeper who has 4 current bees ("current" meaning their bee creation transactions are between depths 576 and 4608 in the best chain).

As soon as block 1565621 arrived, Alice's wallet calculated two values.

Firstly, a deterministic random value which is unpredictable but easily verifiable. We can do this by combining the block hashes of blocks at a variety of (hardcoded) depths between say 0 and 500000 blocks, ensuring our random value is well rooted in the blockchain:

```
string deterministicRandString =
       blocks[blockHeight].hash +
       blocks[blockHeight-13].hash +
       blocks[blockHeight-173].hash +
       blocks[blockHeight-1363].hash +
       blocks[blockHeight-27363].hash +
       blocks[blockHeight-496393].hash;
```

Secondly, her wallet calculates a bee hash target, `beeTargetHash`. This value is driven by an exponential moving average with very high dynamic range[10], which works to set `beeTargetHash` such that a regular frequency of hive mined blocks are issued for any given bee population. As a beneficial side effect of this, the more proof-of-work blocks that have elapsed since the last hive mined block, the higher (easier) the bee hash target is. The algorithm is given as follows; values for `maxTarget`, `emaWindowSize` and `emaDesiredSpacing` will be determined through simulation.

```
beeHashTarget = previousBeeHashTarget (default to highest (easiest) target maxTarget)
numPowBlocks = number of pow blocks since the previous hive mined block;
emaInterval = emaWindowSize / emaDesiredSpacing;
beeHashTarget *= (interval - 1) * emaDesiredSpacing + numPowBlocks + numPowBlocks;
beeHashTarget /= (interval + 1) * emaDesiredSpacing;
```

Both `deterministicRandString` and `beeHashTarget` can be calculated by any node on the network, and they'll all find identical values for both.

Alice's wallet now loops through each of her live bees, combining the deterministic random string and the bee's BCT transaction ID and hashing these to get a new hash – this bee's `beeHash`. Every bee therefore generates one hash per block. This hash is analogous to the best hash generated by a proof-of-work mining rig within the same time duration.

```
hash beeHash = sha256(deterministicRandString + bee.creationTransaction.ID + beeNonce);
```

The beeNonce is the index of this bee within its BCT; if a BCT created 1000 bees, then beeNonces 0-999 would be tried for that BCT.

As Alice's wallet iterates through her bees calculating each `beeHash`, it keeps track of the best (lowest) hash found. If, at the end of iterating through her bees, the best hash that Alice's wallet has found satisfies the condition `beeHash < beeTargetHash`, Alice has the right to mint a block.

Let's say that Alice has a currently living bee whose hash is lower than the target.

Knowing that she has a valid right to mint the block, Alice's wallet produces a block with a special coinbase transaction with 2 outputs, as follows (again, only relevant fields are shown):

```
  "vout": [
    {
      // Zero-value output identifies the bee and proves it's really minting for Alice
      "value": 0.0000000,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_RETURN OP_BEE 857 1564459 1
3166313466633534636663653165633533663561386433366438616566376165383438356534636436393362366
6343034633564323530343031376563643539
2067fc865031769d066e8bf5b29c532d120c752dfdb51f74193cdbcdf822dc44627e71113704d049b80a8fadfaa
521f09378b9d5b8a61de6fe18b66b670b64d3e0",
      }
    },
    {
      // Block reward (subsidy + fees) - must pay to claiming bee's correct honey address
      "value": 250.0000000,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 3a9365c404bd7748f84a40213329da00bf7959b5 OP_EQUALVERIFY
OP_CHECKSIG",
        "type": "pubkeyhash",
        "addresses": [
          "CMocLXWUEnMm36U2FfSgPQ23b5do52DKMb"
        ]
      }
    }
  ]
```

`vout[0]` is a zero-value unspendable output which is used both to identify the bee which is minting the block, and prove that it's really being minted on behalf of Alice.

The scriptPubKey of this output is encoded as follows:

```
OP_RETURN OP_BEE beeNonceVec bctHeightVec cfFlag txidVec messageProofVec
```

Note that the scriptPubKey's begins with `OP_RETURN` to allow us to place arbitrary data after that opcode while still having a script that parses as valid. `OP_BEE` is a custom opcode which identifies this coinbase transaction as being hive mined rather than merely being a strangely formatted proof-of-work coinbase transaction.

The next parts of the script allow us to identify the transaction ID of the bee's original bee creation transaction, and which bee from that BCT has solved the block.

beeNonceVec contains the nonce of the claiming bee, and bctHeightVec contains the chain height of the BCT, used as a helper in validation to prevent having to iterate the chain. cfFlag indicates to validating code that Alice diverts some otherwise-destroyed funds to the community address, and txidVec is the transaction ID of the BCT.

From Alice’s vout[0], we can see that the solving bee is bee #857, from her BCT found at block with transaction ID 1f14fc54cfce1ec53f5a8d36d8aef7ae8485e4cd693b6f404c5d2504017ecd59 (hex encoded in the output above), which can be found at block height 1564459.

The final part is the signature of a message consisting of just the current block number, signed with the private key of the coinbase transaction address.

`vout[1]` is the output that pays Alice the block reward (block subsidy and fees). Note that the receiving address of this output must be the honey address specified in the original bee creation transaction, which is also the address that must sign the block number message in `vout[0]`.

Note that the requirement of this message signature being present is the only reason that Alice must be online for her worker bee to mint a block. If the proof requirement is dropped, so that the coinbase output simply has to go to the specified future coinbase address from the BCT, it would mean that if Alice's wallet wasn't online then someone else could mint the block on her behalf.

Alice would receive her reward even though she's offline, and the network would keep on moving. This initially seems advantageous; it would mean that worker bees could mine on behalf of offline wallets, mobile/hardware wallets, and even paper wallets.

However, although someone could issue this block altruistically, it could also be done as part of a 51% attack; if the legitimate network doesn't need Alice to be around, neither does an attacker, and so they could use Alice's bee to further the chainwork on their withheld chain. The reason the message that Alice's wallet signs consists of the current block number is to prevent an attacker re-using Alice's "proof of bee" in a similar manner, in the event that the same bee is valid to mine another block in the future.

At this point Alice's wallet has determined that she has the right to mint the block, and assembled a block that includes the special coinbase transaction described above. The block is then pushed onto the network.

4.2 Block validation

Bob's wallet, receiving Alice's block, now needs to check if the block meets consensus. He first makes sure that the coinbase transaction has 2 inputs, and that the first has a zero output and that the script begins with `OP_RETURN OP_BEE`. He then extracts the transaction ID of Alice's bee (1f14fc54cfce1ec53f5a8d36d8aef7ae8485e4cd693b6f404c5d2504017ecd59), along with the bee nonce and BCT chain height.

> ASIDE: Because the bee creation transaction was paid to an unspendable output, it remains in the UTXO set. This means Bob's wallet doesn't need to have the txindex command line option enabled (which fully indexes all transactions at the expense of slower verification and much higher disk usage) to be able to easily check the outputs of Alice's BCT. Because of this use of the UTXO set, the core wallet should not need any custom databases or serialisation changes in order to support hive mining. It is also likely that the "Current Bees" UI tab and RPC call results can be dynamically built too, i.e. done without custom serialisation.

Proceeding with validating the hive mined block, Bob's wallet performs the equivalent to the following RPC call:

```
gettxout 1f14fc54cfce1ec53f5a8d36d8aef7ae8485e4cd693b6f404c5d2504017ecd59 0
```

This gives the first output of the BCT, `vout[0]`, and checks can now be made to ensure that (1) the transaction's depth is within bee lifespan range; (2) appropriate bee creation fee was paid; and (3) the correct unspendable address was sent to.

In addition, since Alice's coinbase transaction indicates cfFlag=1, Bob's wallet will perform:

```
gettxout 1f14fc54cfce1ec53f5a8d36d8aef7ae8485e4cd693b6f404c5d2504017ecd59 1
```

This retrieves the second BCT output, `vout[1]`, in order to confirm that the community fund contribution paid to the correct address and wasn't greater than a given percentage of the burnt funds.

The next check is to verify the message signature from the last part of `vout[0]`. Bob's wallet performs the equivalent of a verifymessage call, passing Alice's honey address as the public key and the message signature. The message itself should be the current block height (1565622).

Finally, Bob computes the `deterministicRandString` and `beeHashTarget` for the current block, then computes Alice's `beeHash` and checks it against `beeHashTarget`. If all checks pass, the block is considered validated.

This block validation process is fast and requires no costly iteration across historical blocks.

5. Integrating hive mining with proof-of-work

It is envisaged that hive mining is not the sole method used to secure the network. As explained in section 1, it is vital to not only retain Litecoin Cash's existing mining community, but to not disrupt it in any way. This means that hive mining must fit in with proof-of-work on the same blockchain.

Adopting a hybrid proof-of-work/hive mining system has several strong advantages. Let's consider the implications of the following rules:

- Bee hash target is based on number of proof-of-work blocks since the last bee hash target
- A hive mined block must be preceded by a proof-of-work block

This allows proof-of-work to continue to drive block issue speed without the risk of runaway hive mined blocks. It is likely that in practice, if a bee is valid to solve a block, the issue of that hive-mined block will come shortly after the preceding proof-of-work block, so the issue speed of proof-of-work blocks as driven by Dark Gravity Wave[11] will still provide the overall timekeeping aspect of the network. It is expected that high bee population, causing frequent issue of hive mined blocks, will cause an increase in proof-of-work difficulty, while lower bee population will cause proof-of-work difficulty to drop as Dark Gravity Wave attempts to compress block times to maintain the frequency target.

Proof-of-work blocks, and the requirement for them to come immediately before a hive mined block, also solve the problem alluded to in the thought experiment in section 2 whereby there should be a cost to an attacker for working on multiple chains at once.

Because there is no strict alternation between proof-of-work and hive mine blocks – a proof-of-work block can occur at any time – proof-of-work miners never need to stand idle, and no software

(or behaviour) modification is required of existing pools. They can continue unhindered, exactly as before deployment of hive mining.

The fact that a proof-of-work block is valid at any time also prevents the network becoming stuck indefinitely waiting for a `beeHash` that no bee is strong enough to create for the current block.

While these facts alone do not confound a 51% attack because a proof-of-work attacker can still mine pure proof-of-work blocks and ignore hive mining entirely, this can be overcome by allowing hive mined blocks to carry weight when considering which potential chain should win in a chain reorg scenario. The key to preventing a pure-proof-of-work 51% attacker is to give hive mined blocks sufficient weight. In fact, it seems that to fully frustrate an attacker with 51% of the network hashpower but no worker bees, hive mined blocks should carry *more* weight than proof-of-work blocks.

Chain work is currently calculated as follows:

$$chainWork = \sum_{n=0}^{n<blockHeight} expectedHashesToCalculate(blocks[n].difficulty)$$

That is to say, chain work accumulates as a function of difficulty in each block in the chain. We propose changing this definition as follows:

$$chainWork = \sum_{n \in pow\ blocks} expectedHashesToCalculate(blocks[n].difficulty)$$

$$+ \sum_{n \in hive\ blocks} k * expectedHashesToCalculate(blocks[n-1].difficulty) + (MAXINT - blocks[n].beeHash)$$

This awards each hive mined block a value dependant on the amount of work carried in the proof-of-work block immediately preceding it, with the constant *k* to be defined experimentally.

Note the additional term included in chainWork for hive mined blocks; this allows a priority decision to be made between two candidate chains where each are identical except for two alternative hive-mined blocks at tip. This can result if two valid bees have both submitted valid blocks onto the network at the same time.

While both blocks must already meet the `beeHashTarget` for them to both be considered valid, priority is given to the one with the lower hash.

6. Variations and refinements

These ideas are intended only for future discussion and will not be addressed in our initial implementation.

Within the current model, all parameters such as bee cost, chance to find a block, and bee lifespan, are largely static (or in the case of bee cost, driven by a simple relationship to block reward).

This allows us to avoid either custom serialisation or costly iteration in our initial implementation, but does have downsides. For one thing, the system can only sustain a maximum concurrent bee

population while still expecting all bees to be profitable; this gives rise to the Global Index discussed above.

It may be preferable in the future to drive more parameters in a dynamic way, in order to prevent this hard ceiling on bee population.

For example, bee cost or lifespan could be driven by current bee population. Alternatively, it may be desirable to have (for example) 3 different types of bees, each with a fixed cost. They could have different weights and/or lifespans. For example:

Bee type	Cost (% of 1 block reward)	Increased chance to mint	Lifespan	Creation address
Worker	1%	None	7 days	CReateLitecoinCashWorkerBeeXYs19YQ
Warrior	10%	Equal to 12 x worker	10 days	CReateLitecoinCashWarriorBeeHeS2YP
Queen	100%	Equal to 150 x worker	14 days	CReateLitecoinCashQueenBeeXXUiFgYA

As the idea of worker bees is motivated by the lifecycle of a natural organism, it may be of interest to examine additional parallels and incorporate more biologically-inspired ideas. Real bees are born, mature, and then spend their adult lives defending their hive and performing work such as honey production – analogous to our blockchain worker bees securing the network and earning block rewards.

Real bee hives have a hierarchy, and real bees hatch from eggs. A given egg may produce an individual optimised for a specific task; this process could be modelled within hive mining by deciding that BCTs result not in bees, but eggs.

A given egg could have a chance to produce a bee with different properties (solving weight, lifespan), or even to produce a queen bee which would itself have the ability to produce eggs. Under such a system, a lucky bee keeper may establish a multi-generational hive that will produce block rewards well into the future.

Note that these are examples of how the agent-based idea could be expanded, and will not necessarily be included in our initial implementation.

7. The prescient network: Seeing the future

All estimations of hashpower on a pure proof-of-work blockchain are derived from the time it took the network to solve a given block at a given difficulty. As such, proof-of-work hashpower can only ever be estimated *in the past*, once the block in question has been solved[12]. With hive mining, it is possible to get an exact count of the current bee population and therefore to accurately know the *current* network hive mining capability.

However, even more usefully, the presence within our design of the bee maturation period provides the network and community with an ability to not just monitor the current live bee population but the *upcoming* (immature) bee population.

At any time it is possible to gauge the bee population for the coming days. A given bee keeper motivated purely by individual profit will be able to make an informed value judgement as to whether or not to create additional bees, based on the weight of the bee keeper's own hive measured against the overall network weight.

From a different perspective, the wider community, upon seeing a large number of bees suddenly created, would have time to create bees of their own to offset any potential attack.

This is a subtle but important advantage over proof-of-work or indeed proof-of-stake. With a proof-of-work coin, it's not possible for a benevolent party to hoard hashpower in advance in order to protect the network from possible attack at some point in the future. With proof-of-stake, hoarding coins or coin age *is* possible, but without some knowledge of an impending increase in total network stake weight, the benevolent hoarder cannot deploy their stake without risking causing a 51% attack themselves.

With hive mining, coins can be hoarded by benevolent parties and deployed as bees when it is apparent that the network would benefit from additional bee population. However, attempting to do the same thing offensively will fail unless the attacker can also control over 51% of the hashpower at the same time.

Note that as explained in *4.2 Block Validation* above, it's impossible for an attacker to force another bee keeper's bees to work on their private chain, as a message in the coinbase transaction must be signed with the private key for the honey address.

A potential attack at first glance appears possible: An adversary could create a large quantity of bees and mature them on their own private chain, allowing the bees to kick in and work on the private chain when mature so that the private chain accumulates chainwork at the same rate as the public chain. However, this attack is thwarted by the maturation period; the attacker will be so behind on chainwork by the absence of bees on their private chain that it would be extremely difficult to keep up with the public chain by proof-of-work alone during the vital maturation period.

8. Comparisons to alternative schemes

8.1 Compared to pure proof-of-work

Hive mining has a very low energy cost; in an alternating proof-of-work/hivemining scheme, the overall energy cost over any given timeframe is expected to be approximately halved compared to pure proof-of-work, if the difficulty adjustment is set to produce approximately 50% of each block type.

Hive mining requires no investment in powerful hardware, democratising the mining process.

The additional chainwork dimension introduced by hive mining requires an adversary attempting a 51% attack control 51% of the bee population as well as 51% of the network hashpower; and as discussed in section 7 above, benevolent bee keepers have vastly greater visibility into an incoming attack.

8.2 Compared to proof-of-stake / hybrid proof-of-work and proof-of-stake

The requirement to invest in bees means the inherent inequality of proof-of-stake, where the richer get richer, is mitigated. In proof-of-stake, earning block rewards is a largely passive process for the staker, requiring only that their wallet is online (and sometimes not even that).

It is almost impossible for this scheme to produce a distribution that doesn't skew strongly in favour of large stakeholders, without much action from those stakeholders themselves.

By introducing an actual monetary cost to the process of producing blocks, we consider that hive mining provides a superior "proof of commitment" to proof-of-stake. And by requiring the user to actively create bees with some regularity, Hive mining additional increases proof of *engagement* of users to the community.

Hybrid proof-of-work/proof-of-stake can help with 51% protection in a similar manner to our proposed solution, but we contend that hive mining democratises the mining process more effectively.

Furthermore, as noted in section 7, a benevolent coin holder is not able to predict incoming stake weight attacks and so cannot leverage their holdings to assist in attack prevention.

9. References

[1] *More 51% blockchain attacks expected.* Lielacher, A. (2018). https://bravenewcoin.com/news/more-51-blockchain-attacks-expected/

[2] *Cryptocurrencies Losing Real Money to 51% Attacks.* Ausick, P. (2018). https://247wallst.com/investing/2018/06/09/cryptocurrencies-losing-real-money-to-51-attacks/

[3] *Litecoin Cash 51% Attack Highlights Insecurity of Smaller PoW Coins.* Buntinx, J.P. (2018). http://www.livebitcoinnews.com/litecoin-cash-51-attack-highlights-insecurity-of-smaller-pow-coins/

[4] *Analysis of hashrate-based double-spending.* Rosenfeld, M. (2012). https://bitcoil.co.il/Doublespend.pdf

[5] *Bitcoin: A peer-to-peer electronic cash system.* Nakamoto, S. (2009). https://bitcoin.org/bitcoin.pdf

[6] *Hash Rate. The estimated number of tera hashes per second (trillions of hashes per second) the Bitcoin network is performing.* Blockchain.info (2018). https://blockchain.info/charts/hash-rate

[7] *Litecoin Cash: The best of all worlds SHA256 Cryptocurrency.* The Litecoin Cash Foundation (2018). https://litecoinca.sh/downloads/lcc_whitepaper.pdf

[8] *How the Bitcoin protocol actually works*. Nielsen, M. (2013). http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/

[9] *Double-spending fast payments in bitcoin*. Karame, G.O., Androulaki and E., Capkun, S. (2012). In CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security, pp 906-917.

[10] *Difficulty control for blockchain-based consensus systems*. Kraft, D. (2016). In Peer-to-Peer Netw. Appl., March 2016, Volume 9, Issue 2, pp 397–413 https://www.domob.eu/research/DifficultyControl.pdf

[11] *Dark Gravity Wave*. Dash Core Group (Accessed June 2018). https://docs.dash.org/en/latest/introduction/features.html#dark-gravity-wave

[12] *Optimizing SHA256 in Bitcoin Mining.* Courtois N.T., Grajek M., Naik R. (2014). In Cryptography and Security Systems. CSS 2014: Cryptography and Security Systems pp 131-144

[13] *Proof of burn.* Stewart, I. (2012). https://en.bitcoin.it/wiki/Proof_of_burn