CryptoNote Blocktree: Partially Asynchronous Metablock Consensus

Thaer Khawaja
thaer.khawaja@gmail.com
www.getmasari.org

Abstract

One of the biggest problems to Blockchain scalability is block size, where too large a block will have propagation time and centralization concerns, while one too small will cause a network's failure to converge on a main chain. In practice, one can argue a lower bound of approximately 15 seconds, given proper management of orphaned blocks and a small upper bound on block size [3]. Therefore, in order to increase transaction throughput while maintaining the secure concept of a Proof of Work (PoW) block in a permissionless decentralized network, one must be able to mine multiple blocks in parallel. This paper proposes Blocktree, a tree-partitioned structure that would fundamentally extend that of the Blockchain's linear nature.

1 Introduction

In order to scale a permissionless and decentralized network securely, we take advantage of a property in CryptoNote coins called the Key Image, a globally unique key that ensures an output cannot be spent more than once[2]. This same property will be the underlying basis behind the Blocktree scaling proposal, using what we will be defining as the Metablock, showing how a Blocktree can be derived in a way that would maximize transaction throughput while maintaining resilience to adversarial attacks, while retaining pre-existing cryptographic primitives being used.

2 Metablocks

We define the blocktree as a depth balanced tree, $T = (V, E, M)$, where (V, E) corresponds to vertices and edges in k parallel subchains, where k is the width of the tree at any depth d, and M is the set of Metablocks defined at those depths. Given a width k at a given depth d, a Metablock m_d represents the k blocks at that depth as the hash of the PoW hashes, $m_d = Hash(v_j^d, v_{j+1}^d, \ldots, v_{j+k}^d)$. Each block v_j^d includes in its header the metablock m_{d-1}, and will be equivalent to the parent if $k == 1$.

2.1 Consensus Resolution

The blocktree is balanced when the depth of connected edges from the genesis block root node to any top block subchain leaf node is at most 1 higher than the subchain top block depth, which will be the protocol-enforced asynchronization factor. Metablocks are calculated retroactively when mining at depth $d + 1$, where a given subchain block header defines the blocks it's observing at d in order to mine at $d + 1$. The balanced tree requirement is maintained under this asynchronization factor since miners would need to converge on lagging subchains in order to be able to mine on any of the subchains at $d + 1$. With these properties, the Metablock is the synchronizing "chain" across the different subchains, and follows the same intractability guarantees as defined in the Gambler's Ruin problem[1].

2.2 Difficulty Weight Adjustment

The difficulty calculation would occur over m_d depths, using sum of the k corresponding block solve times as the input, and the per v_j^d block difficulty would be $W_s(v_j^d) = W(m_d)/k$. Weight W is used in this case to distinguish from difficulty in a similar manner to weights defined for SECOR[3].

3 Asynchronous Subchains

Using metablocks, it allows us to partition the traditional blockchain model into partially asynchronous parallel subchains $S = (s_0, s_1, \ldots, s_{k-1})$ while maintaining a balanced tree that is resistant to adversarial attacks. At every depth d, each subchain top block has to reference the other $k-1$ subchains via m_{d-1}; the base case reduces to a regular blockchain where width $k=1$, depth d is the same as the height, and the metabblock is equivalent to a parent reference v_{d-1}. When $k > 1$, each subchain $s \in S$ is freely able to mine independent of other subchains, and the partitioned transactions they're able to mine directly correlate to their width index in the tree. In the case of CryptoNote[2], the partitioning scheme would occur on the key image which ensures that no double spend occurs across subchains. Each subchain would have with it a partition index, which will be necessary for identifying valid transactions.

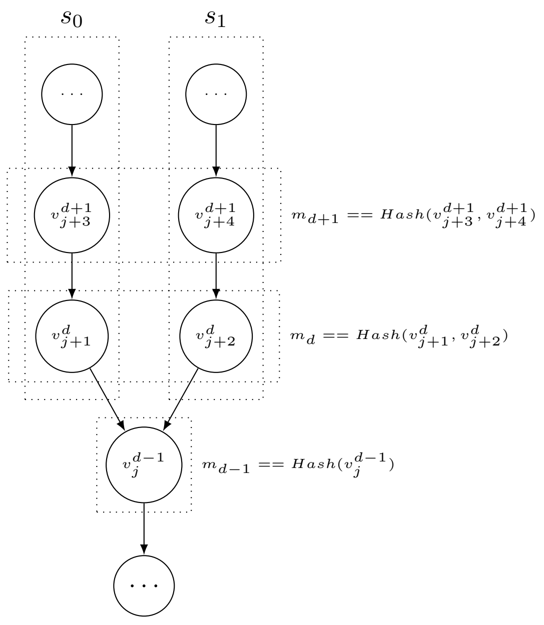
s_0
s_1
$\cdots$
$\cdots$
v_{j+3}^{d+1}
v_{j+4}^{d+1}
$m_{d+1} == Hash(v_{j+3}^{d+1}, v_{j+4}^{d+1})$
v_{j+1}^{d}
v_{j+2}^{d}
$m_{d} == Hash(v_{j+1}^{d}, v_{j+2}^{d})$
v_{j}^{d-1}
$m_{d-1} == Hash(v_{j}^{d-1})$
$\cdots$

Figure 1: Metablock and subchain illustration during a k=2 autoscale event

3.1 Alternative Metablocks

Since metablocks are retroactively defined given arbitrary top blocks from the subchain set S, we can get cases where one miner is using an alternative subchain s_i and therefore a different metablock; this case has a higher likelihood when no block has yet to be mined at $d+1$. In this situation, including mid-mined top blocks at any given depth, the alternative subchain s'_i would replace s_i as the main subchain, resolving the consensus resolution conflict as it would be the metablock with the most cumulative difficulty weight. In the worst-case scenario, storing only the metablock hash in the header would have an $O(n^2)$ complexity in trying to find an alternative metablock, however it should be $O(1)$ as the different parent block at d would be in the revised top subchain block reference at $d+1$.

4 Partitioned Transactions

In order to prevent double spends in these asynchronous subchains, transactions need to be partitioned by the key image so that the subchains are disjoint in their block content. To achieve this, each transaction's outs must belong to a single partition. In the base case where the tree width is 1, a regular blockchain, all key images belong to that partition. In cases of n subchain partitions where $n \geq 2$, modulo arithmetic would indicate which subchain a key image is allowed to be mined in. This has the side effect of sometimes needing multiple transactions when it would've otherwise been a single transaction. As long as the partially asynchronous height limit of the tree is less than the minimum number of blocks needed to unlock a transaction, ring members shouldn't be affected when forming the transaction, where stealth addresses do not need to be partitioned. If a key image was mined on more than one subchain, it would fail consensus due to the partitioning constraint, which continues to ensure that an output is spent only once through the key image without any further modifications.

5 Dynamic Scaling

Scaling the blocktree would be based on modulating the subchain width k with respect to transaction throughput given a scaling protocol. For instance, picking 2^{σ} as our scaling function, it allows for a structured re-partitioning. The base case is $k = 2^0 = 1$, where it's in a regular blockchain state. For $k > 1$ when scaling up, parent of a new top block v_j^{d+1} at s'_i would be from the old subchain at $s_{\lfloor i/2 \rfloor}$, and when scaling down, the parent of s'_i would be $Hash_d(s_{2i}, s_{2i+1})$. The partition function itself would also be part of the consensus protocol, as it would enforce a uniform distribution of the transactions and prevents subchain double spends.

5.1 Mining Decentralization

A fundamental advantage from tree partitioning is that each individual block can remain easier to compute, while still maintaining network security. The Metablock hash here is important because it would protect against adversarial attacks that otherwise one only needs to attack a single subchain if there was no metablock reference. With this, smaller mining pools will be able to be more competitive against bigger ones as their effective mean emission rates differences will be negligible.

6 Future Research

This Blocktree proposal doesn't address subchain observation or storage requirements, as those can and should be tackled independently. For example, with respect to subchain observation, one can research an approach that ties a wallet's address (or subaddress) to a single subchain, which could allow partial observation with cheaper storage and compute costs. Also, given that there's a way to cryptographically enforce partitioning invariants, partitioning by stealth addresses in the future may be the better solution than by key images.

7 Conclusion

This Blocktree proposal resolves the transaction throughput issue in permissionless cryptocurrencies via a tree-like data structure that allows asynchronous parallel chains to be mined, while maintaining the same PoW properties that maintain its security, with no new cryptographic primitives introduced. This asynchronous design, coupled with future advancements in cryptographic primitives, should make for feasible scalable and high throughput real world use case applications.

References

[1] Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System.* bitcoin.org/bitcoin.pdf (2008)
[2] Saberhagen, Nicolas. *CryptoNote v 2.0* https://cryptonote.org/whitepaper.pdf (2013)
[3] Khawaja, Thaer. *Simple Extended Consensus Resolution* https://getmasari.org/secor.pdf (2018)