Canonical Transaction Ordering for Bitcoin: A Critical Evaluation

This document is an evaluation of the Canonical Transaction Ordering proposal (CTOR), which seeks to alter the ordering of transactions within blocks on the Bitcoin Cash (BCH) network. (nChain believes BCH is the true Bitcoin.) After summarizing the proposal, we evaluate the proposed change against standard change management criteria.

For the reasons discussed below, we believe there is insufficient demonstration that the CTOR proposal will actually deliver its claimed benefits, and the risks of implementing such a disputed consensus change outweigh any unproven rewards. Therefore, we believe the CTOR proposal should not be implemented in any BCH implementation.

1 The CTOR Proposal

At present, transaction ordering within a BCH block is a loose form of partial ordering:

- The first transaction is the *Coinbase Transaction;*
- Where a transaction spends outputs of another transaction in the same block, the spending transaction must come after the transaction whose outputs are spent;
- All other transactions - those spending outputs from transactions in previous blocks - may appear in any order.

This is referred to as *Topological Transaction Ordering* (TTOR).

The *Canonical Transaction Ordering* (CTOR) proposal seeks to change the ordering of transactions within a block as follows:

- The first transaction is the *Coinbase Transaction;*
- All other transactions are lexographically sorted by transaction ID.

The CTOR proposal claims to have several advantages over TTOR, namely:

- Elimination of a class of scalability challenges
- Compact inclusion/exclusion proofs
- Opt-in transaction localization
- Efficiency improvements for block emission and propagation
- Simplified software implementations
- Potential attack vector mitigation

In a follow-up article, it was later claimed that CTOR is a prerequisite to sharding Bitcoin, itself positioned as the next step following the transition in CPU development from single-core performance advancements to multi-core products.

2 Community Reaction

CTOR has stimulated debate across the Bitcoin community, with arguments both for and against it being vigorously made. A selection of these are summarised below, making it apparent that CTOR faces significant opposition, and at the very least, serious questions about whether it should be implemented.

- In Private vs trustless sharding, Tom Zander (founder, *Flowee the Hub*) argues against CTOR as a prerequisite to sharding, noting that sharding can be achieved without affecting the consensus rules.
- Rawpool BCH Lab have produced a technical report (official English translation, community-provided English translation) which notes that the current TTOR implementation has had many years of evolution and incremental improvements but reserves further judgement until a mature CTOR implementation is delivered.
- Jonathan Toomim published Canonical block order, or: How I learned to stop worrying and love the DAG in which he observes that accounting for child-pays-for-parent structures is a significant cost during block construction, and that Graphene can be >7x more efficient without ordering concerns, he argues that CTOR allows for simplification of code, and finally concludes that CTOR is not a prerequisite of parallel validation.
- /u/awemany (member, *Bitcoin Unlimited*), with input from Tom Zander and others, critiques the CTOR proposal, opining that many of the motivations and arguments that lead to the CTOR solution are not valid when scrutinised.
- /u/Chris_Pacia (developer, *OpenBazaar*) offers a critique of /u/awemany's previous critique which re-states the motivations for CTOR, disagrees that parallelism can be achieved without it, and partly re-frames the debate to be around the removal of TTOR primarily with the introduction of CTOR a secondary concern.
- /u/markblundeberg (co-author, *Simple Ledger Protocol*) analyses the code changes between Bitcoin ABC version 0.17.1 and 0.18.1. He: (i) notes that the parallel algorithm proposed for validating CTOR blocks (referred to as *Out-Then-In* or OTI) works equally well with the existing TTOR (assuming one minor non-consensus change to carry a transaction ordinal in internal data structures); (ii) observes that Graphene can be implemented under current consensus rules by following a suggestion from Gavin Andresen; and (iii) draws a number of conclusions, including that the most disruptive part of the CTOR suggestion is the removal of TTOR, and that CTOR will not provide any near-term benefits for block validation, remaining undecided on long-term benefits.
- In a pair of forum posts, Steve Shadders (developer, *nChain* and Technical Director, *Bitcoin SV*) compares the cost of Merkle root re-computation when inserting transactions into a Merkle tree, as would be required by CTOR, versus the current optimisations already in place for appending transactions to the end, suggesting that a much larger internal change to bitcoind is necessary, namely replacing the Merkle tree structure with a Merklix tree.
- Andrew Stone (lead developer, *Bitcoin Unlimited*) published Why ABC's CTOR Will Not Scale in which he argues that the sharding proposal following on from CTOR

neither requires CTOR nor solves the motivating scalability concern, and that Graphene can work under current consensus rules making CTOR unnecessary for network optimisations.

3 Evaluating the CTOR Proposal

Any proposal to change an existing system should be evaluated against multiple criteria, including:

- Scope
- Risk
- Reward
- Cost to implement
- Time to market
- Maintenance implications
    - Availability of skilled resources
    - External SLA management
- Technical dependencies
- Non-functional requirements impacts

Each of these will be evaluated with both a Bitcoin-specific and a wider generalised IT system view.

3.1 Scope

3.1.1 Size of code change

The CTOR proposal is bounded in scope in terms of the code changes required to implement it. It is an internal change to the bitcoind daemon. This work has already been completed in Bitcoin ABC 0.18.1.

3.1.2 Infrastructure requirements

No additional infrastructure is required at this time.

3.1.3 Impact on upstream/downstream systems

CTOR is a consensus change. To avoid a chain split (regardless of intent), each fully validating node implementation operating on the Bitcoin Cash network must implement a compatible set of changes.

Mining pool software that uses the result of *getblocktemplate* as-is should be unaffected, however any software that itself builds blocks based on the data returned must be aware of the CTOR rules.

This does not directly impact SPV network clients.

3.1.4 Operational procedures

No additional procedures are required to operate a node with CTOR.

3.1.5 Support processes

Minimal changes to support processes are required. Teams must be aware of the new rule when determining the root cause of any rejected or orphaned blocks.

3.1.6 User training

Users of the Bitcoin Cash network should not be aware of any changes from CTOR. However, in the event of a chain split, users may observe their transactions as both confirmed and not confirmed, depending upon the nodes that their SPV client samples and whether or not competing blocks both include their transaction.

3.2 Risks

The CTOR proposal alters a current consensus rule. Any consensus rule change requires that all fully validating node implementations are modified with a consistent set of changes activated at the same time. This implies changing or deprecating code in each node whose behaviour is currently consistent across those implementations.

Even with sufficient testing, and even with sufficient evidence produced from multiple implementations on testnet, changes may introduce subtle bugs that are not immediately obvious. Edge cases that only show after significant uses may present, the result of which may be of any severity up to and including an inadvertent chain split. This is not without precedent.

As noted in the preceding *Community Reaction* section, there has been significant opposition against, and at a minimum, serious questions raised about, the CTOR proposal. Notably, Bitcoin Unlimited members overwhelmingly voted against the CTOR proposal (22 votes against; 5 in favour; 3 abstaining). The Bitcoin SV implementation will not feature CTOR.

Implementing a consensus change is risky, but implementing a consensus change when there is a significant divide of opinion within the community even more so. The risk assessment of the CTOR proposal is therefore ***high*.**

3.3 Rewards

To assess the potential rewards of (or *benefits of*, or *value delivered by*) the CTOR proposal, we considered the *Motivations for CTOR* described above and evaluate whether CTOR is likely to deliver upon these supposed goals.

3.3.1 Elimination of a class of scalability challenges

In the context of both the CTOR proposal and its follow-up article presenting a sharding strategy, *scalability* appears to refer to the ability to scale *compute resources*. Scalability may also refer to scaling for *capacity* or *resilience*. However, as these are not debated, this claim will be evaluated in the context of *scaling computational power*.

The CTOR proposal dedicates a sizeable section to the compute resources required to arrange randomly sorted items into topological order, with both offline and online prior art presented. CTOR is presented as a more compute-efficient alternative to TTOR for the purposes of sorting transactions within a block.

What the CTOR proposal fails to acknowledge is that transactions are received via the P2P network and accepted into the mempool *in topological order*; that any transaction that does not spend a UTXO set member (either through non-existence or through double-spend) is not admitted into the mempool. Simply maintaining a list of valid transactions in the order they are received (or maintaining an ordered list of transaction IDs regardless of underlying storage layout, or assigning an ordinal upon receipt) ensures that they can be presented within a block **without needing *any* compute resources to apply a topological sort**.

Placing the requirement for a given non-topological ordering onto transactions within a block introduces the need for additional computation. This can be done ahead of time, using an *insertion sort*, or transactions can be reordered at the point of transaction retrieval from underlying storage. The follow-up sharding article references a Merklix tree, a data structure that naturally sorts items on insert.

The existing TTOR compatible code has been incrementally optimised over time. Appending transactions to a list that underpins a Merkle tree does not require the recalculation of the entire tree; optimisations have been applied to grow the tree and convert the existing root to an internal node as transactions are appended and the tree grows taller. Inserting into a Merklix tree provides reasonable ordering but introduces the likelihood of requiring a full Merkle root recalculation (transactions that happen to sort into the equivalent of a Merkle append may be optimised as such).

Whilst the goal of scaling to handle increased transaction volumes is agreeable, the CTOR proposal offers no concrete evidence that CTOR lowers the utilisation of compute resource today, nor does it demonstrate clear scaling benefits. Furthermore, the CTOR proposal authors have not provided any test metrics or instrumentation data based upon a comparison of TTOR and CTOR node strategies. Nor has any conclusive argument been made that future scaling can *only* be achieved through a consensus change.

3.3.2 Compact inclusion/exclusion proofs

The CTOR proposal claims compact inclusion and exclusion proofs as a benefit. Whilst compact proofs are certainly desirable for the SPV client use case, no clear explanation of this claim is given.

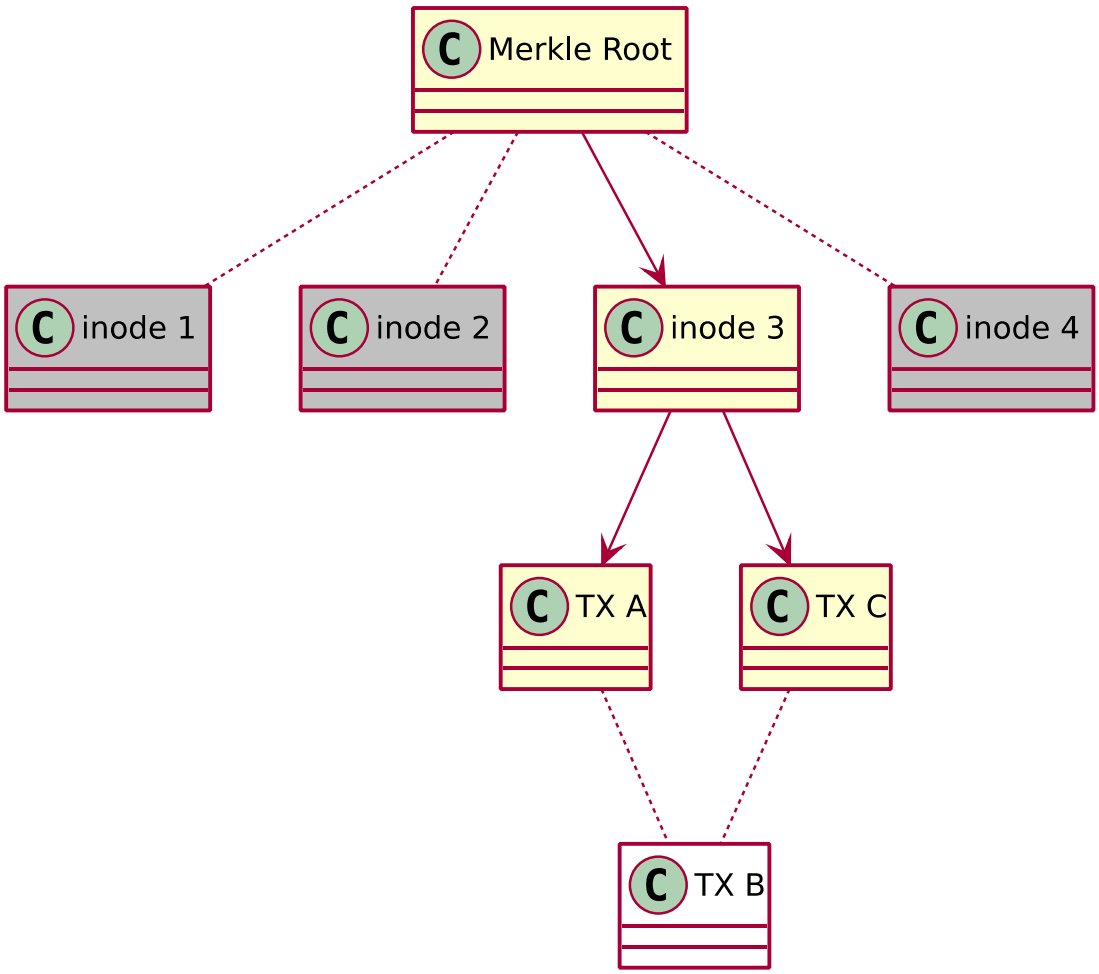
Neither is any explanation given for how exclusion proofs may be generated *at all*.

Merkle Proofs

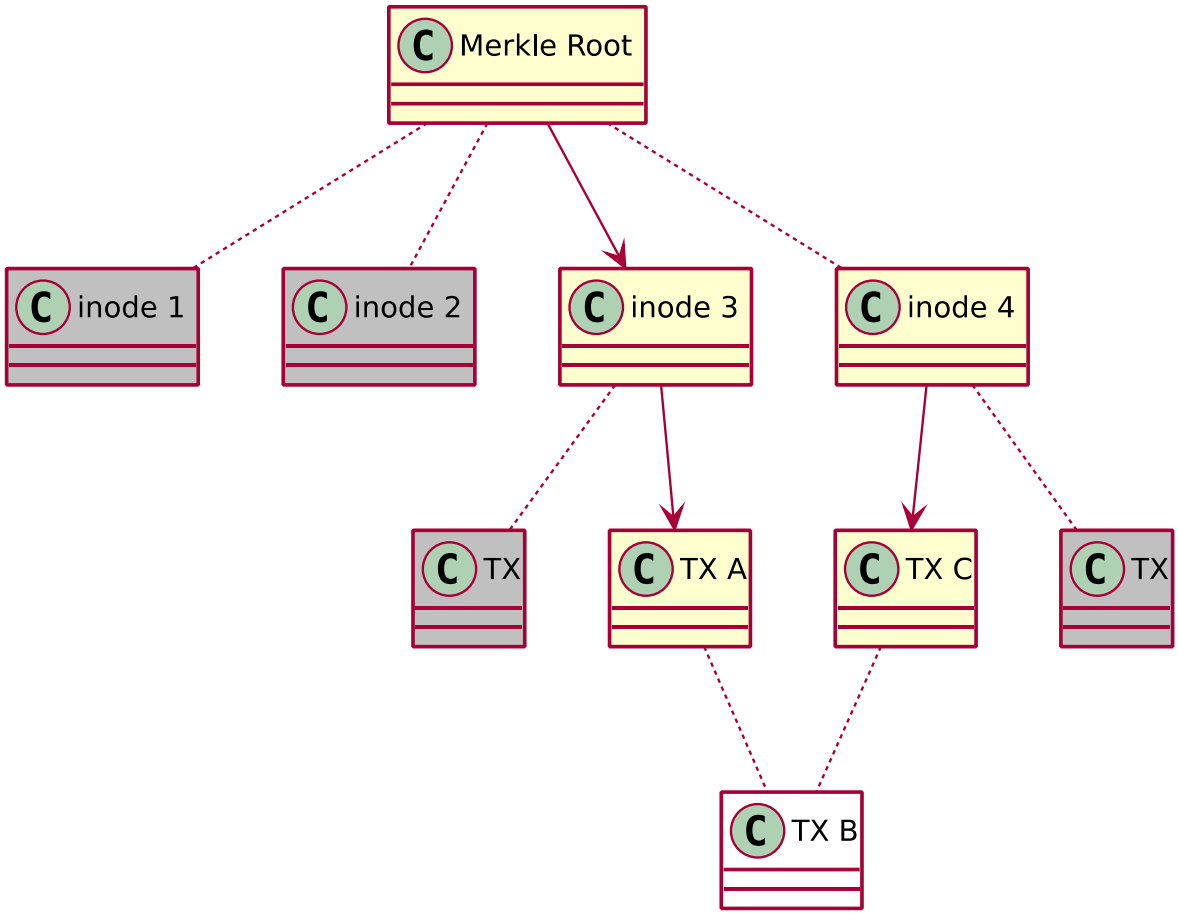
One possible interpretation of *compact proofs* is that *Merkle proofs* are somehow compacted.

No impact on the compactness of Merkle inclusion proofs is apparent from the CTOR proposal.

It is plausible to see how two leaf nodes (TX A and TX C in the diagram) that share a parent (and therefore common Merkle proof up to the final leaf node) cannot contain a transaction lexographically in between them (TX B in the diagram), as under CTOR rules the transactions should have been laid out in transaction ID order and therefore the transaction was not included:



It is not immediately obvious how the proof would hold if the queried transaction TX B fell between two leaf nodes TX A and TX C with different parents inode 3 and inode 4, as they would no longer share a common path in their Merkle proofs:



Even assuming that exclusion proofs *are* possible between contiguous leaf nodes with different parents, exclusion proofs are bound to the scope of a single block. To prove exclusion from the entire block chain, a proof must be generated for every block in the chain.

Given that exclusion proofs cannot be generated using this method for any block mined to date, whole-chain exclusion proofs cannot be generated.

No use case for exclusion proofs has been presented, nor is it conclusive that they are enabled by CTOR. No demonstration of inclusion proof compactness has been provided.

Range Restrictions

In a post titled [On compact proofs for token protocols], Joannes Vermorel (co-author, CTOR proposal) presents a concept for *compact token proofs*. The CTOR proposal could be referring to this article when referring to *compact inclusion/exclusion proofs*.

The article discusses two ways in which lightweight clients may download only some block data. The first is to request all transactions whose ID falls within a given range. An extension

of this idea is that by using a process like vanity mining, application users may deliberately target a given hash range to ensure that all transactions of a certain type (in the referenced article, token transactions) fall within it. The argument follows that this would allow light clients to request subsets of blocks by transaction ID range, and that this is only possible with CTOR.

While we collaborate with Joannes Vermorel on a project, we believe he is definitely incorrect on the above points.

- There is no guarantee that the submitter of a transaction will first vanity-mine a transaction ID within a target range.
- There is no mechanism to prevent another party vanity-mining the same range, thereby reducing the effectiveness of the scheme.
- To suggest that this vanity mining process and subsequent range-based transaction querying can only succeed if CTOR is enabled is to fundamentally fail to separate responsibilities of an IT system. If it is desirable to provide data queries by hash ID range, the underlying data store should be configured to support such queries, or if it cannot be, the data store should be migrated to a solution that can. An RPC endpoint for such a query pattern should be provided. Data storage/retrieval and lightweight client endpoint service are two separate responsibilities and should be treated as such. To conflate the issue with a third discreet responsibility, that of block propagation, is simply poor engineering practice.

The second suggested way in which lightweight clients may download only subsets of all block data is for clients to only download every n^{th} block, for some definition of n, meaning that the client would download only 1 of every n blocks to find relevant transactions. This suggestion acknowledges the impracticality of a submitter determining the block height at which a transaction will be mined and so will not be discussed further here.

3.3.3 Opt-in transaction localization

Transaction localization is the process whereby a first party repeatedly malleates a transaction (by any method, for example regenerating signatures) until the transaction ID falls within a range acceptable to the transaction creator. It is then submitted to the network and under CTOR will be proximate to other transactions malleated to fall within the same ID range.

The CTOR proposal suggests no benefits of this goal, although as discussed in the previous section, range-restricted lightweight client queries may be the underlying driver.

The follow-on sharding proposal suggests sharding transaction processing by using the transaction ID as a partition key. If the intent of the transaction localization suggestion is to allow those submitting transactions to the network to attempt to target a given shard then this is a flawed and possibly even dangerous suggestion.

There is no guarantee that a given node will run a specific number of shards, and therefore no guarantee that this will ensure localised transactions will be located on a specific shard. There is no stated benefit for this use case either. Finally, an adversary could use this behaviour by generating transactions with identifiers in a narrow range, with the intent of overloading one shard. This is a form of denial-of-service attack specific to the follow-on sharding proposal.

3.3.4 Efficiency improvements for block emission and propagation

The CTOR proposal (incorrectly) notes that CTOR shifts the data model from a *list* to a *set* of transactions. This is incorrect because transactions in a block are already a set. A list and a set differ only in that in a set, all elements are unique. Assuming this is simply a mistake and that the intent of the CTOR proposal was to highlight the shift in the model from a *set* to a *sorted set*, the CTOR proposal notes that this change allows for the application of well understood *set reconciliation* techniques to reduce the amount of data transferred during block emission and propagation.

Existing work in this area, such as Graphene, demonstrates one such technique. Graphene does not require any specific ordering, only that an ordering is stable between sender and receiver. Bitcoin Unlimited have an implementation that achieves most of the benefits of Graphene without changes to consensus rules, by including ordering information along with IBLT data.

Amaury Sechet of Bitcoin ABC observed that during the recent (September 1, 2018) BCH network stress test, *"graphene block were on average 43kb in size. 37kb to encode ordering, or 86% of the data"*.

It is true that with CTOR the ordering information could be omitted, leaving only the underlying set reconciliation data in the payload. However, this is a marginal improvement over an already impressive wire optimisation (BU's implementation); the benefits of CTOR on Graphene and similar block propagation techniques are minimal.

3.3.5 Simplified software implementations

The more complex software is, the harder it is to:

- verify
- maintain
- reason about
- upskill new developers in
- find bugs

It is therefore reasonable to aim to reduce complexity of software implementations.

The CTOR proposal discusses changing transaction validation code from the current one-pass algorithm to a two-pass *Out-Then-In* (OTI) algorithm. Both are relatively simple to understand, so whilst not more complex, this is certainly not less so.

It is debatable that any implementation of a node that scales across threads, processes or even machines no longer needs to be topological order-aware, and in such a scenario perhaps CTOR reduces the workload. However, given that tracking TTOR ordering at the point of sharing work across machines is trivial, the simplification case has not been demonstrated.

In its present form, CTOR does not achieve this. Instead, it adds additional behaviours into the codebase. When analysing all changes between Bitcoin ABC 0.17.1 and 0.18.1, it is hard to see any significant change in complexity.

3.3.6 Potential attack vector mitigation

The CTOR proposal contains an annex that suggests CTOR will be simpler to implement than TTOR for processing significant block sizes (beyond 10GB). The annex theorises that it follows that this simplicity indicates a lower potential for attack vectors in the future.

This claim is neither supported with adequate reasoning nor supported by evidence.

3.4 Cost to implement, time to market

At an initial look, it may not appear CTOR is a significant change because the development cost to make the change itself is not large. But the cost of testing that each node implementation has made a change compatible with every other implementation is much larger. Neither is quantified explicitly.

As the change itself is small the delivery time is low. The time to market should be elongated however due to the nature of the change. As a consensus change, the BCH development community should spend sufficient time compatibility testing nodes.

This has not even begun yet as there is still contention amongst node implementors, with some teams simply not implementing the CTOR proposal.

3.5 Maintenance implications

No impact on maintenance is perceived. The code change is small and well understood. No additional skills would be required to continue to work on the codebase after this change.

3.6 Technical dependencies

There are no additional technical dependencies introduced by the CTOR proposal.

The CTOR proposal is positioned as a dependency for future value delivery, primarily as a prerequisite to scaling for the purpose of handling increased transaction volumes and ultimately block sizes many orders of magnitude larger than we see today.

CTOR has not been sufficiently demonstrated to be necessary to enable future scaling.

3.7 Non-functional requirements impact

The code changes required to implement the CTOR proposal are conceptually small and the theoretical impact is negligible - that is, neither significantly positive nor negative.

No non-functional test results have been published to support this.

4 Evaluation Summary

While some of the CTOR proposal's goals may appear at first glance to be admirable, there is insufficient demonstration that those goals are actually achieved by the implementation of CTOR. Further, as a consensus change - and a highly contentious one - there is significant associated risk with implementing CTOR, without proven benefit. For these reasons, nChain believes the CTOR proposal should not be implemented.