Dynamic Slot Allotment consensus: the Mintlayer protocol[*]

Enrico Rubboli [†], RBB LAB, Dogana, RSM
Alberto De Luigi [‡], Milan, Italy
Luca Viviani [§], Minsk, Belarus

November 7, 2020

draft

Abstract

We present a byzantine fault tolerant solution for a distributed and permissionless ledger protocol, relying on the Bitcoin blockchain as a source of randomisation and time calculation to define the active roles the participants play in the network within a specific time frame. The protocol is designed to work in an asynchronous network. The Proof-of-Stake validation is secured against long range attacks by means of a checkpoint system incorporated in the protocol.

[*]Thanks to: RBB LAB, San Marino Innovation
[†]enrico@rbblab.com
[‡]mail@albertodeluigi.com
[§]luca@mintlayer.org

Contents

draft

1 Definitions

- **Validator**: a user running a full validating node.
- **Round**: a period of 1008 Bitcoin blocks.
- **Participant**: role played by a validator who has been selected in a round as a leader and a blocksigner. There are 1008 participant slots, a single validator can be associated with more participant slots.
- **Stake**: amount of coins locked in order to be selected as a participant for a round.
- **Proposer**: a participant creating a block
- **Slot**: one of the 1008 elements of a sequence which defines a committee and a leader. Each block in a round is associated with a slot. A single slot is associated with more blocks in a round (the slot order is repeated until the end of the round).
- **Leader**: slot owner selected through the consensus algorithm which has higher chances than the other participants in creating a valid block.
- **Committee**: subset of 80 participants associated with each slot.
- **Blocksigner**: one of the 80 participants of the committee allowed to countersign a block at a specific height (associated with a specific slot).
- **Signature Threshold**: minimum number of countersignatures required to accept a block as valid.
- **Signing Power**: accumulated weight of the signatures in a chain (or block) at a certain height.

2 Bitcoin Sidechain

Mintlayer is a Bitcoin sidechain governed by a dynamic group of users (participants) who generate blocks according to the consensus rules. The participants are selected through a POS mechanism. The security of this model ultimately relies on the Bitcoin blockchain for:

- **time calculation**: each Mintlayer block references to a Bitcoin block hash. This mechanism is exploited for three purposes:
  1. to allow the creation of checkpoints on the Bitcoin blockchain to avoid POS long-range attack (§11);
  2. to calculate the duration of Mintlayer rounds (§4)
  3. to enforce a minimum time delay which prevents the creation of parallel longest chains with a higher frequency of block generation (see the escaping stall clause §10)
- **randomization**: blocksigners and leaders (§6) associated to a round (§4) and to the respective slots (see §5 and §9) are randomically ordered using the Bitcoin block hash as a source of entropy

A user can apply for the role of participant inside a round (§4) by putting coins at stake (§13).

3 Block

Every Mintlayer block includes a reference to the hash of a Bitcoin block. If the Mintlayer block X references the Bitcoin block Y, the block X+1 necessarily references the hash of a Bitcoin block at a height higher or equal to Y. Several Mintlayer blocks can include the reference to the same Bitcoin block hash, since the frequency of Mintlayer blocks generation is about 5-10 blocks for each Bitcoin block.

Mintlayer nodes lay on a Bitcoin node: when Bitcoin chain reorganization happens, the Mintlayer node discards all Mintlayer blocks referencing to the hashes of orphaned Bitcoin blocks, eventually reorganizing on the "longest" Mintlayer chain in terms of signature power (§8).

draft

The average frequency of block generation is variable in a range of about 1-2 minutes (§7).

Mintlayer blocks are structured as follow:

1. the header contains the version of the block, the reference to the previous block in the form of hash link pointer, the timestamp, the bitcoin block hash, the optional checkpoint reference, the txs merkle root, the data merkle root, the block signatures, the utreexo merkleroot;
2. the data field contains an ordered list of monetary transactions, staking data, oracle's data, ACL data and notarization values.

The maximum size of a block is 1mb.

4 Round

A period of 1008 bitcoin blocks defines a Mintlayer round. It amounts to approximately 5,040 Mintlayer blocks (up to about 10,080 at maximum throughput). During a round, the participants alternate in the role of leaders and blocksigners, according to the slot order (§5) determined when the previous round ends.

The participants for the round x (active round) are selected among those validators who put coins locked-in at stake in one of the blocks of the round x-2 (auction round).

The participants cannot move their coins out of the staking address until the end of round x+1 (stake round), which means that the coins are frozen during at least part of round x-2 (auction round), the entire round x-1 (setup round), round x (active round) and x+1 (stake round). When a node discovers the new Bitcoin block at the end of the round (the 1008th Bitcoin block), it uses the hash of that block as a source of randomization to derive the list of blocksigners in the committee and leaders for the new round, which starts with the first Mintlayer block referencing the 1009th Bitcoin block hash since the beginning of the previous round.

It might be that two blocks at the same height are generated at the edge of two rounds, both valid because the first refers to the 1008th Bitcoin block hash, the second to the 1009th, so they rely on two different committees and both received enough countersignatures (§9). If two blocks are valid at the same height and with the same signature power (§8), then the validators orphan the block referencing the oldest Bitcoin block hash.

5 Slot

In a round there are 1008 participants' public keys and 1008 slots. A participant, represented by the public key specified during the staking in the auction round, is coupled with a slot. A single validator might represent more participants. The slot sequence is the order the participants are expected to follow in acting like leaders and blocksigners (§6) when generating and countersigning blocks.

The slot 1 is assigned to the public key of the first participant, which is then the leader deputed to the creation of the first block in the round; the second slot is assigned to the public key of the second participant, which is leader for the creation of the second block in the round, and so on. When 1008 blocks have been created, the slots order starts all over again and the first participant public key (slot 1) will be the leader deputed to the creation of block 1009.

The order is repeated until the end of the round (Mintlayer block z). The block z+1 is the first block generated in the new round, when the nodes stop considering the previous participant list of public keys as valid, and starts following the new order of slots. When the outcome of an auction is determined, a validator can win the position of leader for more than one slot. Statistically, the share of slots won is proportional to the weight of the stake.

6 Blocksigner, leader, proposer

The users participating in the governance of Mintlayer chain cover three roles: blocksigner, proposer and leader. The blocksigner is a participant allowed to countersign a block at a specific height (all blocks

draft

associated with that particular slot). Every participant of a round is also a blocksigner of at least one committee and leader for the creation of at least one block in that round. Within a round, potentially every participant, regardless of whether it is leader or not, can be a proposer for any block of that round.

The proposer is the participant who creates a block and broadcasts it to the network as an "open block", in order to collect the countersignatures from the blocksigners. For a block to be valid, there is a minimum threshold of 40 signatures that have to be collected.

The leader is a proposer who has an advantage under specific circumstances in proposing a valid block. In fact, the leader signature has more weight (signature power) than those of the other blocksigners for a block at a specific height, in particular, those blocks corresponding to the slot(s) won by the leader. For example, if a participant is the leader of slot 1, his signature is weighted more than that of any other proposer who will try to generate a valid block at height 1, 1009, 2017, 3025 and so on (assume the block at height 1 is the first of the round). Moreover, the leader is the only proposer able to collect countersignatures from the other blocksigners within the leader time window (§7).

7 Leader time window and counter-signatures

The validators keep counting the time passed since the last block. When a new valid block is received, the node resets the clock and starts counting from zero. A blocksigner's node never countersigns a proposed block unless 60 seconds have passed before the previous block, according to its local clock. Therefore, a block cannot be validated reaching the minimum threshold of countersignatures until 60 second have passed since the last valid block has been received according to the local clock of at least 40 blocksigners (which is the minimum countersignature threshold).

Also, a blocksigner never countersigns a block created by a different proposer than the leader, unless 120 seconds have passed since the node received the previous valid block.

8 Leader substitution and signature-power

If the leader's block is not received in the leader time window, starting from the 120^{th} second the blocksigners will countersign the first new valid block broadcasted either by the leader or by the slot owner immediately next to him in the 1008 slot sequence. A participant never countersigns two blocks at the same height. If no valid blocks are received at the 130th second, a blocksigner potentially countersign either a block proposed by the leader or by the first two slot owners immediately next in the order. The same logic is applied as time passes (i.e. 140th, 150th, etc.), allowing to countersign blocks proposed by a progressively broader list of participants: each 10 seconds a new slot owner is added to the list, the one which is immediately next in the sequence of 1008 slots determined for that round.

When a node is aware of alternative valid Mintlayer chains, the node reorgs on the chain with the highest accumulated signature-power.

To calculate the signature power, the countersignatures of all blocksigners are counted with the same weight (equal to 1), while the proposer's signature is weighted more, depending on the relative position of the proposer's slot to the slot of the current leader: the closer the slot owned by the proposer to the position of the leader's slot, the higher the weight. The leader signature has the maximum weight possible. This weighting system provides a deterministic way to select the longest chain in the unlikely case two valid blocks at the same height pass the countersignature threshold (40 signatures each).

If the mempool is empty of transactions, the proposer may wait an undetermined amount of time before the creation of the block, because the leader has no incentives in creating empty blocks and no losses if another participant replaces him.

9 Randomly chosen subset of blocksigners and minimum countersignature threshold

If the threshold was very low, a corrupted minimum number of slots could successfully broadcast valid blocks earlier than expected (outside the leader time window §7), increasing the frequency of block generation. It

draft

allows room for dangerous attacks. For example, if it is possible to countersign a block with a minimum threshold of 40 signatures out of 1008, an attacker staking only the 4% of the total share of coins could countersign each block he created and flood the network with valid blocks with much higher frequency of generation than the competing canonical chain, gathering more accumulated signature power even if he is not the leader and does not receive countersignatures from the blocksigners.

If the threshold is high, it may take too much time to collect enough signatures, affecting the network latency and delaying the valid block generation. Moreover and more importantly, with a high threshold the network may stall if there are missing blocksigners.

The solution to this trade-off problem is to define a subset of blocksigners (a committee) to countersign the blocks. Each slot is associated with a committee of 80 randomly chosen blocksigners out of 1008 participants. Each block coupled with a slot must be validated by collecting a minimum threshold of 40 countersignatures out of 80.

At the block z of the previous round it is determined the subset of 80 blocksigners associated with each slot. Every block is rejected if it doesn't reach 40 counter-signatures, unless the "escaping stall clause" requirements are met (§10).

This allows for a lower number of countersignatures required (better latency, faster relay) raising at the same time the staking requirements to perform an attack.

The 40/80 threshold has been chosen as the optimum solution considering the following tradeoff:

- a large threshold increase the latency and time required to relay valid blocks
- a small threshold compared to the committee size increases the chances for an attacker to countersign the blocks proposed by himself
- a large threshold compared to the committee size increases the probability of missing blocksigners and therefore stalled blocks
- a small subset allows for thinier threshold, but represents a smaller sample of population which then increases the hypergeometric probability of occurrence of attacks and stalled blocks

40 countersignatures out of 80 have minimal impact on latency and relay (and don't affect the blocksize, since they are aggregated), while the probabilities of attacks are minimal, as calculated below:

- Hypergeometric probabilities of success for a 30% signature power attack: in a population of 1008 participants' slots, suppose the 30% (302 slots) are in the hands of a malicious stakeholder. The probability for the attacker to reach the 40/80 threshold of countersignatures relying solely on his stake is the 0.006994%. This means that the attacker will be able to anticipate a leader creating a valid alternative block once over 14,297 blocks. It's very unlikely that the attacker will be able to create a second block on top of it (x+1), accumulating this way more signature power than the canonical chain (still at block height x). Even if it happened, the attacker will only be able to replace the leader at height x and won't be able to compromise the chain.

10 Escaping stall clause

It may occur that 40 or more blocksigners of the current committee are missing/offline, in this case the chain will stall. For this reason, there are conditions which allow the creation of valid blocks that do not meet the countersignature threshold.

A Mintlayer block at height higher than X and referring a Bitcoin block at height Y can be accepted without reaching the countersignature threshold only if:

1. it includes the hash of a bitcoin block at height Y+2. This avoids the possibility to create a parallel private chain with blocks generated at higher frequency than the canonical chain and therefore cumulating more signature power;
2. it is received 240 seconds after the block at height X (measured locally). In the unlikely case that two bitcoin blocks are created almost simultaneously (condition 1 is met in a few seconds), this further check avoids that a malicious participant could generate a valid block replacing the leader.

draft

The signature power of the escaping-stall block is calculated according to the general rule: the proposer signature is weighted depending on the relative position of the proposer's slot to the height of the block created, while the countersignatures of the blocksigners count as 1.

The impact of stall blocks on the chain is estimated to be minimal, calculated as follows:

- Hypergeometric probabilities of occurrence of a stall block when 30% participants are missing: if 30% of the participants (302 slots) are missing for an entire round, the probability that the threshold is not reached for a block is 0.006994%. Which means that, on average, every 14,297 blocks 1 is stalled because the threshold can't be met. In that case, it is necessary to apply the escaping stall clause and wait for 2 new bitcoin blocks, which are created on average in a timespan of 15 minutes, before to proceed with the creation of new valid Mintlayer blocks. It means that if 30% of the participants are missing, then there is the chance to have the chain stalled for about 15 minutes every 20 days.

11 Checkpoints

Everyone may include a marker on the Bitcoin blockchain with the hash of a Mintlayer block at any height (let's say height X). For this purpose, it's necessary to execute a Bitcoin transaction (Tx0) with an OP_RETURN and include this transaction in a Bitcoin block Y.

When the proposer generates the Mintlayer block at height X+n where n is a positive integer, it can include a reference to Tx0.

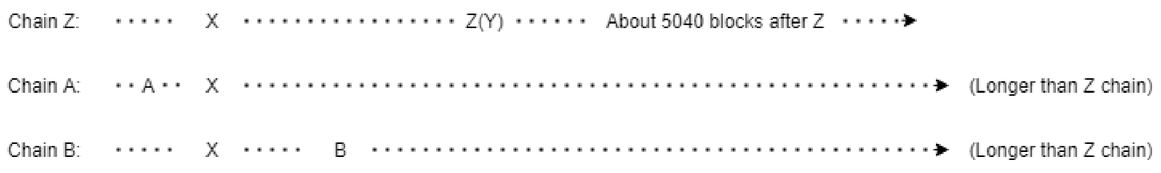
The other nodes will acknowledge the Mintlayer block as a checkpoint request, distinguished from the standard blocks precisely because it includes two hashes of the Bitcoin blockchain rather than only one, corresponding to the last block. The checkpoint request is at height Z, but the checkpoint is enforced at height X.

A validator can prune the chain starting from height X.

12 Checkpoint chain consolidation

After the 1008th confirmation on the Bitcoin network since block Z, every Mintlayer full node consolidates the blockchain up to block X (does not allow rollbacks affecting the chain before that block) placing a checkpoint on its local node. The count up to 1008 blocks considers only the new Bitcoin blocks hashes after block X included in the header of a Mintlayer block.

After having consolidated the blockchain up to X, if a node discovered a fork that happened before X, let's say in A (see the graph below) resulting in a longer chain in terms of signature power (chain A), then the blocks of that chain would be discarded. If instead the node discovered the existence of a longer chain B, but block B is after X, the node would discard the part of the chain Z after block B, reorganizing on the chain B even if block B is antecedent to the checkpoint request Z(Y), since the checkpoint is actually effective in block X and not block Z(Y).



If a node is synching the blockchain from the beginning and chain A and B are longer than Z, the new node will never locally place a checkpoint in block X. In case a user temporarily obtains the 51% of the stake, to avoid long range attacks on nodes that are synching the chain from the beginning the releases of the software may include updated checkpoints.

13 Coins at stake

The Mintlayer token (MLT) is used to apply for the participant role, by staking coins as described in §4. The tokens are put at stake using a lock-in function with a specific OP_CODE.

During the setup round, the algorithm of selection chooses only those validators who are staking more than 0.01% of the total token supply, which is an anti-spam threshold.

14 Transactions

Every proposer can collect fees on transactions paid using any token transferred on the Mintlayer chain. A user is free to set a desired amount in any token. Every proposer can signal in the block header a list of the tokens accepted and might censor any unwanted transaction. The wallets query a third party service to suggest a fee in the specified token. By default, every node accepts fees paid in Mint (MLT).

A single transaction can batch (or coinjoin) payments in different tokens, aggregating the signatures of every input in a single one, this way resulting in a size of about 70bytes per payment when more payments are aggregated. The aggregated inter-token payments can pay a single fee in a desired token, even if there is no amount of that token addressed to the recipient of the transaction.

It is possible to create tokens on Mintlayer with confidential transactions enabled. It's only optional, to preserve free space when confidentiality is unnecessary or inappropriate, as it happens for the security tokens.

15 Annex: attack vectors and checkpointing

15.1 Case 1: the attacker wants to boycott a soft fork

Let's consider an attacker who intends to boycott a soft fork adopted by a majority. The fork introduce a narrower limit on the blocksize. By definition, the attacker is a minority, which means he has less signing power than the majority proposing the fork.

The attacker wants to take advantage of the checkpoint system, forcing all nodes which are not yet updated to converge on his chain. In this case, the chain supported by the attacker is the legacy chain, while the forking chain is expected to be the canonical chain (if by definition it is the one supported by the majority of signing power).

Let's then assume that after the fork at Block S (S for "Small blocks"), the nodes that follow the new chain no longer allow blocks greater than 0.5 megabyte.

The attacker will therefore create an X Block after S with a bigger size than 0,5 megabyte, that the non-upgraded nodes will recognize as valid.

By definition, the attacker in the long run can't compete with the signature power of the canonical chain, therefore the proposed block X would ultimately be invalidated by all nodes which will reorganize on the canonical chain. However, the attacker tries to stop the chain reorganization using a checkpoint.

The attacker must qualify as a blocksigner in a round and when he is the leader he creates Block X, including the hash of X into a Bitcoin transaction Y(X) on the Bitcoin blockchain. At this point, he will wait for the next slot in which he is leader and create the checkpoint block Z(Y) in Mintlayer, including the hash of the Bitcoin block Y.

Assume that the output of the randomization algorithm established the most favorable slot order for the attacker, so he can create Z(Y) shortly after X, then he has to wait 1008 blocks before the nodes locally recognize X as checkpoints.

The nodes with updated software will never recognize the attacker's chain as valid, as their software will never reach Block X. On the other hand, non-upgraded nodes that may have temporarily followed the attacker's chain would recognize the attacker's checkpoint at height X only 1008 Bitcoin blocks after Z. The only way the attacker can successfully stop the fork upgrade is by:

1. having the majority of signing power until 1008 Bitcoin blocks have passed (which is excluded by hypothesis)

draft

2. making a sybil attack to the non-upgraded nodes so that they won't receive any data from the canonical chain until they see block Z. The chance of success is extremely unlikely and the estimated costs of constantly deceiving several fullnodes for a week is huge.

Even if successful, such an attack would only temporarily suspend the soft fork, but it wouldn't lead to a permanent split of the network in two branches of the blockchain, as we see in the next Case study.

15.2 Case 2: the network split

Following Case 1, let's assume that an attacker had successfully isolated the blocksigner nodes with sybil attack, such that two chains are formed and the attack lasts long enough to the point that the blocksigners on both chains decide to set up a checkpoint.

It is very likely that the network split affected only the Mintlayer network and not the Bitcoin network. If some Mintlayer nodes don't receive new Bitcoin blocks, the Mintlayer chain cannot go forward. If instead both chains were receiving Bitcoin blocks, it would mean that the Bitcoin network is also splitted, with miners actively working on both the chains.

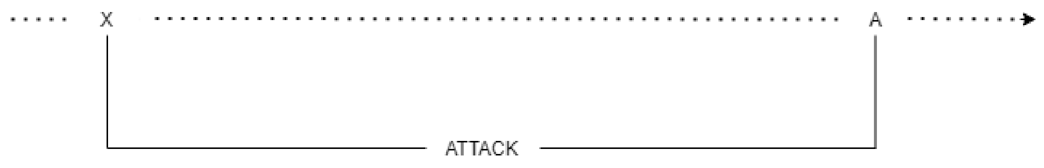
As an example we can imagine a hypothetical split between nodes in Asia and Europe. When the connection between the two continents is restored, or at least when there is a new contact, the nodes on the shorter Bitcoin chain will reorg on the longest chain. At that point, the presence of a checkpoint on the Mintlayer chain would be irrelevant, since the connection with the Bitcoin blockchain represents a tighter constraint than the checkpoint. Since the checkpoint includes blocks of the Mintlayer chain referring to Bitcoin block hashes forgotten by the network (because on a minority fork) those blocks and checkpoints are invalidated.

15.3 Case 3: POS reorg when the attacker has a temporary majority

Let's assume that an attacker obtains temporarily the majority of signing power, such that he is able to validate a checkpoint in X, cumulating more signing power than the alternative chain for a period up to block $Z(Y) + 1008$ bitcoin blocks. This would widen the definition of attacker, including an entity capable of keeping the majority of signing power for a large timespan, which actually counts at least 4,024 Bitcoin blocks (auction round, setup round, active round, stake round). In that case, the "attacker" would be effectively able to stop a fork positioning a checkpoint, but in that case his behaviour wouldn't properly fit the definition of attack. It's not possible to consolidate a soft fork without having enough signing power for such an amount of time.

However, an entity which obtained only temporarily a majority wouldn't be able to make long range attacks precisely because of the checkpoint system. The attack would happen by definition only after the attacker gave away his stake (his majoritarian power). A POS reorg attack is not effective on a range longer than the timespan between the last valid checkpoint and the moment the attacker gave away his 50+1 power (note that the most recent checkpoint request cannot be activated unless 1008 new Bitcoin blocks are generated on top of it).

Let's assume that the last checkpoint is in X, then the attacker can attempt a reorg spanning from the first block after X to the moment he still has the majority, assuming that it happens at block A.

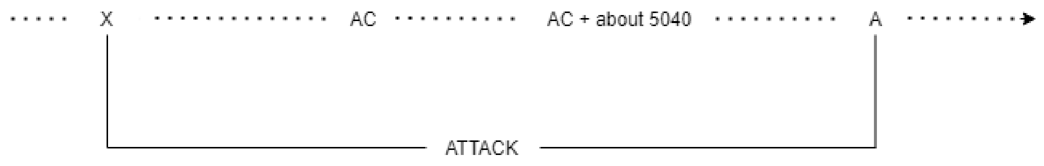


In this timeframe, the attacker could try to re-order or censor transactions, but cannot hard fork or double spend, because the nodes ignore whatever checkpoint placed in blocks built above invalid blocks.

draft

15.4 Case 4: the attacker has a temporary majority and creates valid checkpoints

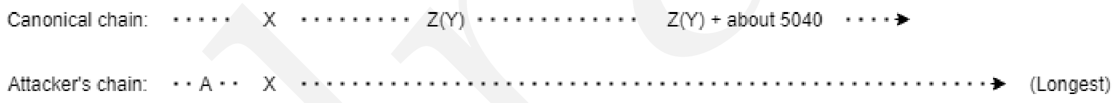
Suppose X is the last active checkpoint, AC is the checkpoint request, which is recognized as an active checkpoint only in AC+ about 5040 blocks. At height A the attacker loses his majority power.



The mere fact of creating a valid checkpoint in the period between X and A doesn't constitute a threat to the sidechain. The checkpoint cannot validate a chain which is invalid for the other nodes. Such an attack could only affect lightnodes developed in such a way that they do not validate the chain but yet recognize the checkpoints as valid. Fullnodes and properly developed light clients wouldn't be affected.

15.5 Case 5: long range attacks on nodes synching from the genesis

Let's assume the checkpoint X has been consolidated on the canonical chain. Meanwhile, an attacker who temporarily gained the majority of power before block X creates a private parallel chain starting from block A (before X) as it happens in a traditional POS long range attack. The attacker is able to accumulate more signature power on its chain with respect to the canonical chain.



Normally, the nodes of the network reject the attacker's chain even if it's longer because of checkpoint X. However, if a node synchronizes the chain from the genesis block it will see two competing chains:

- most of the peers relay the canonical chain with its own checkpoints
- the attacker relays the chain with its own checkpoints

A node cannot validate checkpoints on two different chains and once reconstructed the respective trees, it will start validating what appears to be the longest chain in terms of signature power, which in this case is the attacker's one.

Therefore, to avoid a long range attack against new nodes synching from the genesis block, the new software releases need to include the updated checkpoints made on the canonical chain.

draft