# Doomslug: block confirmation with single round of communication, and a finality gadget with guaranteed liveness

Alex Skidanov
Near Protocol
/AlexSkidanov
alex@nearprotocol.com

December 31, 2019, last update January 31, 2020

**Abstract**

In this paper we present Doomslug[1] – a construction that allows a set of block producers in a blockchain protocol to create blocks with a single round of communication per block such that each block is irreversible unless at least one block producer gets their stake slashed. We also present nightshade finality gadget that requires more rounds of communication, but provides stronger and more common safety guarantee that a block is irreversible unless one third of the total stake is slashed. We also show that under partially synchronous network both Doomslug and Nightshade Finality Gadget have liveness.

## 1 Introduction

The are two dominant approaches to producing blocks in Proof-of-Stake systems: running a BFT consensus such as Tendermint [1] on each block, or having block producers produce blocks on some schedule, like for example in Ouroboros [2].

The former has a minor disadvantage that it requires at least two rounds of communication between the block producers, but also a bigger disadvantage that the protocol stalls if more than one third of all the block producers simultaneously go offline.

The latter has a disadvantage that a block cannot be considered final until at least several other blocks are built on top of it. Also, depending on implementation, such block producer schedules often either rely on having synchronized wall clocks, or allows block producers[2] to grieve neighboring block proposers.

[1]Doomslug is a character in science fiction novel Skyward by Brandon Sanderson. Brandon Sanderson allows using character names with attribution.

[2]Throughout this paper we call the set of all the participants producing blocks *block producers*. We call a participant that proposes block at particular height a *block proposer*

In practice we would prefer to have a reasonable finality guarantee after just one round of communication, and maintain liveness even if close to one half of the block producers are offline.

In this paper we present a particular construction that consists of a fast block producing gadget called Doomslug, and a slower finality gadget called Nightshade Finality Gadget.

Assuming there are $n = 2t + 1$ block producers, Doomslug has the following properties:

1. **Safety**. A produced block cannot be reverted unless at least one block producer has their stake slashed.
2. **Liveness**. Under partially synchronous network assumption from any reachable state of the system for as long as at least $t + 1$ honest block producers are online, a block will be produced in finite time.
3. **Performance**. Under synchronous network assumption, a honest block proposer will produce a block with just one round of communication.

Assuming there are $n = 3f + 1$ block producers, Nightshade Finality Gadget has the following properties:

1. **Safety**. A finalized block cannot be reverted unless at least one third of the total stake of all the block producers is slashed. Or, in other words, unless $f$ block producers are slashed.
2. **Liveness**. Under partially synchronous network assumption from any reachable state of the system for as long as at least $2f + 1$ honest block producers are online, a block will be finalized in finite time.
3. **Performance**. Under synchronous network assumption, two consecutive honest block proposers will finalize a block within two rounds of communication.

## 2 Block Production Gadget

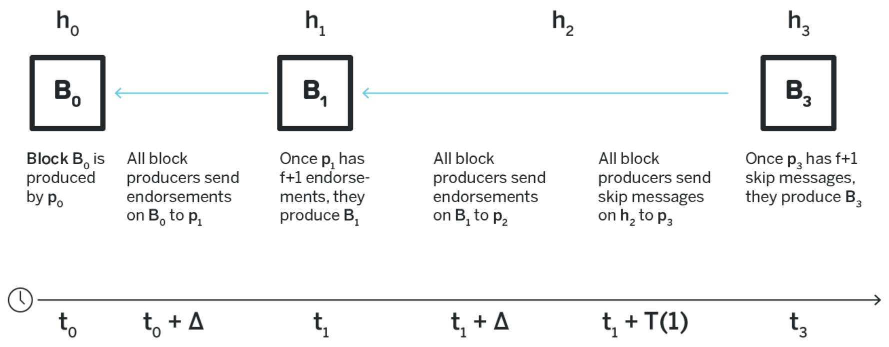
### 2.1 Overview

In this section we provide an overview of block production gadget. In the section 2.2 we provide a more formal definition.

Throughout this section we assume that there's a total of $n = 2t + 1$ block producers, of which $t + 1$ are honest, online and follow the protocol, while the remaining $t$ can be offline or arbitrarily deviate from the protocol.

In Doomslug each block has a height, and each height $h$ has a particular block producer who is assigned to produce a block at $h$, whom we call throughout the paper the *block proposer for h*. We denote the block proposer for $h$ as $BP(h)$.

Whenever a block $B$ is produced at height $h$, it is broadcast by $BP(h)$ to all other block producers. Once a particular block producer $p$ receives $B$, they



h0
h1
h2
h3
B0
B1
B3
Block B0 is produced by p0
All block producers send endorsements on B0 to p1
Once p1 has f+1 endorsements, they produce B1
All block producers send endorsements on B1 to p2
All block producers send skip messages on h2 to p3
Once p3 has f+1 skip messages, they produce B3
t0
t0 + Δ
t1
t1 + Δ
t1 + T(1)
t3

Figure 1: An example timeline of doomslug block production

create an endorsement message $E(B, p)$, unless they previously promised to skip endorsing height $h$. The block proposer for height $h + 1$ cannot produce their block unless they have messages $E(B, p)$ (or messages indicating some block producers suggest to skip $B$, see below) from at least $t + 1$ block producers. They must include such messages into the produced block. The set of such messages is called *witness*.

Given some function $T(x)$ each block producer that didn't receive the block for height $h+1$ after time $T(1)$ sends another message $S(h+1, 1, p)$ to $BP(h+2)$ which indicates that $p$ suggests to skip 1 block starting at height $h + 1$, and promises not to endorse any block at height $h+1$. The block proposer $BP(h+2)$ for height $h+2$ can produce their block if they collect $t+1$ messages $S(h+1, 1, p)$. If some block $B'$ is produced at height $h+1$ after some block producers started sending $S(h + 1, 1, p)$, the block producers that sent $S(h + 1, 1, p)$ cannot send $E(B', p)$, that would constitute a slashable behavior. Other block producers can and shall send $E(B', p)$ to the block proposer for $h + 2$. In such case the block proposer for $h + 2$ must include any combination of $S(h + 1, 1, p)$ and $E(B', p)$ from $t + 1$ block producers in the witness of the block they produce.

Similarly, if the block $B''$ produced by the proposer for $h + 2$ is missed by some block producers, they would send either

1. $S(h + 1, 2, p)$ if they also missed $B'$ and time $T(1) + T(2)$ passed since they received $B$, indicating that they suggest to skip two blocks starting at $h + 1$, or

2. $S(h + 2, 1, p)$ if time $T(1)$ passed since they received $B'$, indicating that they suggest to skip one block starting at $h + 2$.

The block proposer at height $h+3$ must collect any combination of $S(h+1, 2, p)$, $S(h + 2, 1, p)$ and $E(B'', p)$ from $t + 1$ block producers and include them in the block they produce.

If some block $B_{h+1}$ includes $t+1$ messages $E(B_h, p)$ for its previous block $B_h$, then $B_h$ is final in a sense that reverting it would require at least one block producer committing a slashable behavior.

## 2.2 Formal Definition

In this section we provide a more formal definition of the block production gadget.

For the purposes of $B$ we define the following attributes:

$$\mathit{prev}(B) \text{ – previous block of } B;$$
$$\mathit{height}(B) \text{ – the height of } B;$$
$$\mathit{witness}(B) \text{ – witness for } B;$$

There are three kinds of messages that block producers send to each other over the network:

| | |
|---|---|
| $\mathit{Block}(B)$ – | announcement of $B$; |
| $E(B,p)$ – | an endorsement of $B$ by $p$; |
| $S(h,n,p)$ – | a skip-message suggesting to skip $n$ blocks starting at height $h$ by $p$; |

Each $\mathit{Block}(B)$ message must be signed by the corresponding block proposer $BP(\mathit{height}(B))$. Each $E(B,p)$ and $S(h,n,p)$ message must be signed by $p$.

The $\mathit{witness}(B)$ on the block is a set of endorsement and skip messages from at least $t+1$ block producers.

In a valid block message $\mathit{Block}(B)$ each message $m$ in $\mathit{witness}(B)$ must satisfy the following conditions:

1. If $m$ is an endorsement message $E(B', p)$, then

$$B' = \mathit{prev}(B) \wedge \mathit{height}(B') + 1 = \mathit{height}(B)$$

2. If $m$ is a skip-message $S(h,n,p)$, then

$$h \leq \mathit{height}(\mathit{prev}(B)) + 1 \wedge h + n = \mathit{height}(B)$$

.

We also define the following slashing conditions:

1. If $\exists(B, B') : B \neq B' \wedge \mathit{height}(B) = \mathit{height}(B')$, then $BP(\mathit{height}(B))$ is slashed.

2. If $\exists(B, h, n, p) : E(B,p) \wedge S(h,n,p) \wedge h \leq \mathit{height}(B) < h + n$, then $p$ is slashed.

The first condition ensures that the same block producer doesn't propose two blocks for the same height, and the second condition ensures that a block producer that suggested to skip a certain height didn't also produce an endorsement for a block at that height.

## 2.3 Algorithm

**Algorithm 1** Doomslug subroutines

```
initialization
    $witnesses = MultiMap < height \to message >$
    ▷ largest height for which a block with $t+1$ endorsements was seen
    $h_{final} = -1$
    ▷ largest height which $p$ endorsed or promised to skip
    $h_{max} = -1$
    ▷ the tip of the chain
    $B_{tip} = genesis$

procedure processTimer(h)
    if $h_{max} < h$ then
        $h_{max} \leftarrow h$
        send $S(height(B_{tip}) + 1, h - height(B_{tip}), p)$ to $BP(h+1)$
    startTimer($h+1$)

procedure startTimer(h)
    interrupt $skip\_timer$
    $skip\_timer$ = schedule processTimer($h$) after $T(h - h_{final})$

function validWitnessMessage($m$, $B_{prev}$)
    $m$ is $S(h, n, p')$ such that $h \leq height(B_{prev})$, or $m$ is $E(B, p')$ such that $B = B_{prev}$

procedure checkBlockProduction(h, delay)
    $witness \leftarrow \{m \in$ get$(witnesses, h) |$ validWitnessMessage$(m, B_{tip})\}$
    if $BP(h) = p \land h > height(B_{tip}) \land$ length$(witness) \geq t+1$ then
        if $delay \land$ not $witness$ contains $t+1$ endorsements then
            schedule checkBlockProduction($h$, $false$) after $T(h - h_{final})/2$
        else
            $B_{tip}$ = newBlock$(prev = B_{tip}, height = h, witness)$
            broadcast $Block(B_{tip})$
```

The algorithm of Doomslug is described in listings 1 and 2. In the algorithm each subroutine and message handler is considered to be handled atomically.

The block producers exchange blocks, endorsements and skip messages. Upon receiving a block, if the block has height higher than the highest known

**Algorithm 2** Doomslug message handlers

```
upon $E(B,p')$ do
    insert($witnesses$, $height(B)+1$, $E(B,p')$)
    checkBlockProduction($height(B)+1$, $true$)

upon $S(h,n,p')$ do
    insert($witnesses$, $h+n$, $S(h,n,p')$)
    checkBlockProduction($h+n$, $true$)

upon $Block(B)$ do
    broadcast $Block(B)$
    if $height(B) > height(B_{tip})$ then
        $B_{tip} = B$
        startTimer($height(B_{tip})+1$)
        if $h_{max} < height(B)$ then
            $h_{max} = height(B)$
            send $E(B,p)$ to $BP(height(B)+1)$
    if $witness(B)$ contains $t+1$ endorsements and $height(B) > h_{final}$ then
        $h_{final} = height(B)$
        startTimer($height(B_{tip})+1$)
    checkBlockProduction($height(B)+1$, $true$)

```

tip, the block producer updates the tip. They also immediately send an endorsement if they haven't promised to skip the height previously.

Block producers produce a block whenever they have endorsements or skip messages from $t+1$ block producers. It is necessary for liveness that a block producer would wait for half of $T(h-h_{final})$ before producing a block at height $h$ unless they have $t+1$ endorsements. The block production is handled in the `checkBlockProduction` subroutine. Note that while the block producer waits, enough endorsements might accumulate, which would trigger another invocation of `checkBlockProduction` and produce the block before the timer in the original invocation runs out.

We omit from the listings the maintenance of seen message, and assume that each block producer handles each unique message exactly once, even if they have received it multiple times. This in particular means that if a block producer $p_1$ received a block message from $p_2$ and re-broadcasted it, $p_2$ will not process the block message again upon receiving it.

By definition of the endorsement message it must endorse the block with height immediately preceding the height of the block including it in its witness. Therefore for the block producer for height $h$ to include endorsements on tip $B_t$, it must be that $h = height(B_t)+1$. Otherwise the block proposer for $h$ can only include skip-messages in its witness.

## 2.4 Analysis

### 2.4.1 Safety

In this section we provide a formal proof of block production gadget safety.

**Theorem 1** (Block Production Gadget Safety)**.** *If a block $B$ with $prev(B) = B_1$ has endorsements on $B_1$ from $t + 1$ block producers, then $B_1$ is irreversible (meaning that no block $B_2$ with equal or higher height can be produced such that neither $B_1$ nor $B_2$ are in the ancestry of each other) unless at least one block producer is slashed.*

*Proof.* Say it is not the case, and another valid block $B_2$ is produced such that neither $B_1$ nor $B_2$ are in the ancestry of each other. Recall that $height(B_2) \geq height(B_1)$. Consider a block $B_2'$ in the ancestry of $B_2$ such that $height(B_2') \geq height(B_1)$ and $height(prev(B_2')) < height(B_1)$. Consider two cases, as shown on figure 2:

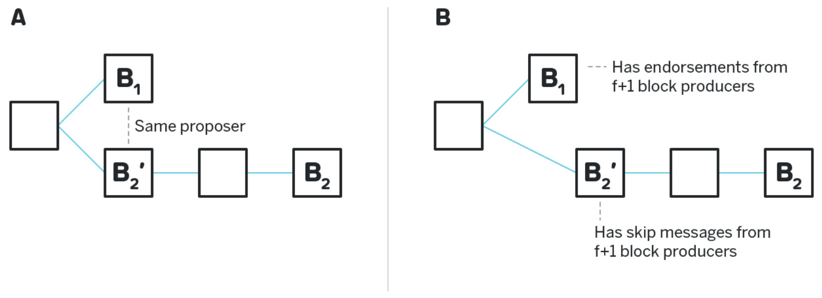

A
B1
Same proposer
B2'
B2
B
B1
Has endorsements from f+1 block producers
B2'
B2
Has skip messages from f+1 block producers

Figure 2: Reverting a block that has doomslug finality

1. $height(B_2') = height(B_1)$. Then $BP(height(B_1))$ is slashed for proposing two blocks at the same height.

2. $height(B_2') > height(B_1)$. Since $height(prev(B_2')) < height(B_1)$, the $witness(B_2')$ cannot contain any endorsement messages (since by definition an endorsement can only endorse a block if it's height immediately precedes the height of the block containing the endorsement). Thus, the $witness(B_2')$ contains at least $t + 1$ skip-messages. There are also at least $t + 1$ endorsement messages on $B_1$ in the $witness(B)$. Naturally, there's at least one block producer $p$ that signed both of such messages. Consider their skip message $S(h, n, p)$ in the $witness(B_2')$ and their endorsement message $E(B_1, p)$ in the $witness(B)$. A skip-message included in $B_2'$ by definition has the property that

$$h \leq height(prev(B_2')) + 1 \wedge h + n = height(B_2')$$

.

Since $height(prev(B_2')) < height(B_1)$, and $height(B_2') > height(B_1)$, it follows that

$$h \leq height(B_1) \wedge h + n > height(B_1)$$

which is exactly the slashing condition for endorsing a block at a height that was previously suggested to be skipped, and thus $p$ is slashed.

In either case at least one block producer is slashed, which proves the theorem.

□

### 2.4.2 Liveness

For liveness proof we will assume a partially synchronous network. In other words, we assume that there's a moment in time (unknown to us) called *global stabilization time*, or GST for short, such that there exists some value $\Delta$ (unknown to us) such that any message between any two honest block producers sent after GST will be delivered in time not exceeding $\Delta$. We also assume that all the messages sent before GST will be delivered to their corresponding recipients within $\Delta$ after GST. We choose such $T(x)$ that:

$$\forall q : \exists x : T(x) \geq q \wedge$$
$$\forall x | x \geq 2 : T(x) \geq T(x-1)$$

We say a block $B$ has *doomslug finality* if at least one honest block producer saw another block $B'$ such that the $witness(B')$ has at least $t+1$ endorsements on $B$. Note that according to theorem 1 a block that has *doomslug finality* is irreversible unless at least one block producer is slashed.

**Lemma 1.** *For any height $h$, at least one block at height $h' > h$ will be produced after GST.*

*Proof.* Say it is not the case, and there's some height $h$ such that no block at height higher than $h$ will be produced after GST. Let $h_{max}$ be the largest height for which a block is ever produced, let such block be $B_{max}$. Consider the smallest height $h_{next} > \max(h, h_{max}+1)$ for which no block producer produced an endorsement, and which no block producer promised to skip, before they received and processed $B_{max}$, and such that $BP(h_{next})$ is online and honest.

As each block producer $p$ receives $B_{max}$, they start a timer, which at some point reaches height $h_{next}$ for the first time. Let's show that $p$ will send a skip message to $BP(h_{next})$ once the timer reaches height $h_{next}$. Indeed, they would only not send a skip message if they previously promised to skip or endorsed a height equal to or greater than $h_{next}$. They didn't endorse or skip such height before receiving $B_{max}$ by definition of $h_{next}$. They didn't skip $h_{next}$ after receiving $B_{max}$ because this is the first time the timer reaches $h_{next}$ since receiving and processing $B_{max}$. They couldn't endorse a block at height higher or equal to $h_{next}$ because by assumption no such block was ever produced.

Therefore each block producer $p$ will eventually send a skip message to $BP(h_{next})$. Once $t + 1$ block producers sent such skip messages to $BP(h_{next})$, $BP(h_{next})$ can and will produce a block at $h_{next}$. □

**Lemma 2.** *Let $B_e$ be the block with largest height $h_e$ that has doomslug finality at some point in time.*

*Consider some height $h$ for which an honest block producer $p$ sent a skip message. For any height $h' < h$ it must be that either $p$ sent an endorsement message on some block at $h'$, or sent a skip message that promises to skip $h'$, or observed a block with a height $h_b, h' < h_b < h$.*

*If a message to endorse or skip $h'$ was indeed sent, unless block with higher height than $h_e$ has doomslug finality at some point, the time that passed between sending an endorsement or a skip on $h'$ and sending a skip for $h$ is at least $T(h - h_e)$*

*Proof.* The first part of the lemma can be proven by induction. The base of the induction is the case when $h-h' = 1$. The only way to send a skip message on $h$ is to call `processTimer`($h$), which can be only be scheduled from `startTimer`($h$), which in turn is only called when an endorsement or a skip message is being sent on $h - 1$. For a message not to be sent, it must be that $h_{max} \geq h - 1$. But $h_{max} < h$, otherwise the skip message on $h$ would not have been sent. Thus for a skip message or an endorsement message on $h - 1$ not to be sent when we call to `startTimer`($h$), it must be that $h_{max} = h - 1$, but the only way for $h_{max}$ to assume such a value is for an endorsement message or a skip message on $h - 1$ to be sent earlier.

For a case when $h - h' > 1$ by the same reasoning as above either a skip message or an endorsement message was sent for $h - 1$. If it was a skip message, by induction the lemma holds. If it was an endorsement message, then a block is known at height $h - 1$, and $h' < h - 1 < h$, thus lemma also holds.

In both cases above the skip message or the endorsement message could not have been sent after the `startTimer`($h$) was called, because whenever we send a skip or an endorsement message, we always restart the timer. Thus, the skip or the endorsement message on $h'$ must have been sent before the timer for $h$ was started, which was at least $T(h - h_e)$ before the call to `processTimer`($h$), which in turn was $T(h - h_e)$ before the skip message for $h$ was sent by $p$. □

**Lemma 3.** *Let $B_e$ be the block with largest height $h_e$ that has doomslug finality at some point in time.*

*Unless another block with higher height has doomslug finality at some point, if an honest block producer $p$ sent a skip message for height $h$ such that $T(h-h_e) > 8\Delta$ at time $t_0$, no later than $t_0 - 7\Delta$ all honest block producers have either sent a skip message promising to skip $h - 1$, or an endorsement on some block at height $h - 1$.*

*Proof.* Consider $height(B_{tip})$ and $h_{final}$ from perspective of $p$. $p$ updated both $B_{tip}$ and $h_{final}$ more than $\Delta$ before $t_0$, since updating either would have restarted the timer, and $p$ hasn't restarted their timer for at least $8\Delta$ before $t_0$.

It's easy to show that all the honest block producers have the same $height(B_{tip})$ and $h_{final}$. They can't have smaller values, because within $\Delta$ from $p$ learning the higher value they would have learnt it as well, and vise versa.

Consider the block producer $p_{min}$ for whom the later of the moments when they updated $B_{tip}$ or $h_{final}$ last time was the earliest among all the block producers, call that moment $t_{min}$, and the block producer $p_{max}$ for whom such a moment was the latest, call it $t_{max}$. It's easy to show that $t_{max} - t_{min} < \Delta$.

Thus, for all the block producers the last call to `startTimer`$(height(B_{tip})+1)$ was within $\Delta$ from each other. Since no honest block producer restarted their timers since then, their last calls to all heights higher than $height(B_{tip} + 1$ also happened within $\Delta$ from each other.

If $p$ called `processTimer`$(h)$ at $t_0$, they must have called `startTimer`$(h)$ exactly $t_0 - T(h - h_e) \leq t_0 - 8\Delta$. All other honest block producers must have called `startTimer`$(h)$ within $\Delta$, which is no later $t_0 - 7\Delta$. As they called to `startTimer`$(h)$, they either sent a skip message on $h - 1$, or an endorsement message on some block at height $h-1$, or already had $h_{max}$ equal to or exceeding $h - 1$.

If for some honest block producer $p'$ $h_{max}$ was equal to $h - 1$, then at some point preceding $t_0 - 7\Delta$ they sent a skip message promising to skip $h - 1$ or an endorsement message on a block at height $h - 1$.

If $h_{max}$ was greater than $h-1$, before $t_0-7\Delta$ $p'$ sent a skip message on some height exceeding $h - 1$, because $height(B_{tip}) < h$. But then according to 2 they sent a skip message or an endorsement message on $h - 1$ before that moment.

This exhaustively covers all possible cases. □

**Lemma 4.** *Let $B_e$ be the block with largest height $h_e$ that has doomslug finality at some point in time.*

*Unless another block with higher height has doomslug finality at some point, there's exists a height $h'$ such that for any height $h > h'$ with $T(h - h_e) > 8\Delta$, a block will be produced as long as the $BP(h)$ is online and honest.*

*Proof.* According to lemma 1, at least one block will be produced after GST. Let $h'$ be the height of any such block, that was produced after GST.

Say the lemma is incorrect, and there's a height $h > h'$, such that $T(h-h_e) > 8\Delta$ and $BP(h)$ is online and honest, such that no block is produced for height $h$. According to lemma 1, there will be a block produced for a height higher than $h$. Let the height of the first such block produced be $h_1$, and the block itself be $B_1$. $height(B_1) - height(prev(B_1))$ must be at least two, because $height(B_1) > h$ and $height(prev(B_1)) < h$ (since no block at height $h$ exists by assumption, and $h_1$ is the first block at the height higher than $h$ for which a block is produced). Therefore $witness(B_1)$ cannot contain any endorsement messages in it, and thus contains at least $t+1$ skip messages. At least one such message is from an honest block producer, because there's less than $t + 1$ malicious block producers.

According to lemma 2, since no block between heights $h - 1$ and $h'$ exists at the time $h'$ is produced, $p$ sent the skip message for height $h$ or an endorsement on some block at height $h$ at least $T(h_1 - h_e) \geq T(h - h_e)$ before they sent a skip message to $BP(h')$. Let's call the moment such a skip message was sent $t_0$. Note that no block at heights higher than or equal to $h$ was produced until $t_0$, because no block at height $h$ is never produced by assumption, and the first block to be produced at a height higher than $h$ is $h'$ which is produced after $t_0$.

According to lemma 3, since $T(h - h_e) \geq 8\Delta$, no later than $t_0 - 7\Delta$ all the honest block producers sent a promise to skip height $h - 1$ or an endorsement message on some block at height $h - 1$. Therefore no later than $t_0 - 6\Delta$ $BP(h)$ collected $t+1$ such messages, and produced a block at height $h$, and no later than $t_0 - 5\Delta$ all honest block producers saw such a block, causing a contradiction.

□

**Theorem 2** (Block Production Gadget Liveness). *For any height $h$, at least one block at height $h' > h$ will be produced after GST that has doomslug finality.*

*Proof.* Let $B_e$ be the block with largest height $h_e$ that has doomslug finality at some point in time. Say the theorem is incorrect, and it is possible that no block at height height that $h_e$ will ever have doomslug finality.

According to lemma 4, there's a height $h'$ such that for any height $h > h'$, $T(h - h_e) > 8\Delta$ with an honest block producer $BP(h)$ a block for height $h$ will be produced. Consider one such height $h$ that both the block producer for $h$ and for $h - 1$ are honest, and $T(h - 1 - h_e) > 8\Delta$. Given that there are more than half honest block producers, such height always exists.

Consider the first moment in time when $BP(h)$ has skip messages or endorsements on its current head from at least $t + 1$ block producers. Since the number of malicious actors is at most $t$, at least one such endorsement or a skip message came from an honest block producer, say such a block producer is $p$. When $BP(h)$ ultimately produces the block, it doesn't have $t+1$ endorsements, by assumption that no block with *doomslug finality* is published with a height higher than $h_e$. Therefore $BP(h)$ will wait for at least $T(h - h_e)/2 > 4\Delta$ between the moment $t_0$ they first received an endorsement or a skip message for height $h$ from an honest actor until the moment $t_1$ $BP(h)$ actually produced the block.

First consider the case that the first message from $p$ for height $h$ received by $BP(h)$ was a skip-message promising to skip height $h - 1$. According to lemma 3, since $T(h-1-h_e) \geq 8\Delta$, no later than $t_0 - 7\Delta$ all the honest block producers sent a promise to skip height $h-2$ or an endorsement message on some block at height $h-2$ to $BP(h-1)$. Therefore no later than $t_0 - 6\Delta$ $BP(h-1)$ collected $t + 1$ such messages, and produced a block at height $h - 1$, and no later than $t_0 - 5\Delta$ all honest block producers saw such a block, including $p$. But then $p$ either immediately endorsed it, or already had $h_{max} \geq h - 1$, in both cases it would not send the skip message for $h - 1$ at $t_0$, causing a contradiction.

Thus, it must be that any message sent to $BP(h)$ by any honest block producer was an endorsement. Note that those block producers that did send the endorsements did it within $\Delta$ from each other. Indeed, block producers only send endorsements after receiving a block, and the time since the first honest block producer receiving a block until the last honest block producer receiving the same block never exceeds $\Delta$.

Since $BP(h)$ waits at least $4\Delta$ since receiving the first endorsement until producing the block, and all the honest block producers send their endorsements within $\Delta$ from each other, $BP(h)$ will include all the endorsements sent by honest block producers in its witness.

It is now sufficient to show that no honest block producer skipped sending their endorsement. For a honest block producer to skip sending an endorsement they needed to either send an endorsement for a block at a higher height, or promise to skip the same or higher height. Consider two cases:

1. There was at least one block with height $h'$, $h' > h$ that existed before the block at height $h$ was produced. Consider the lowest such $h'$. The witness of the block at height $h'$ doesn't contain any endorsements, because $h' > h$ and $prev(h') < h$. Thus it contains skip messages from $t+1$ block producers, at least one of which is from an honest block producer. Since at least one honest block producer sent a skip message for height $h' - 1$, all the block producers, including $p$, sent a skip messages or endorsements on $h' - 2 \geq h' - 1$ according to lemma 3.

2. No block with height higher than $h$ was produced before the block at height $h$ was produced. In this case for $p$ to skip endorsing the block at $h - 1$ they must have previously sent a skip message on height $h - 1$ or higher.

In both cases $p$ has sent a skip message on some height $h_s$, $h_s \geq h - 1$ such that no block is known to them with heights between $h-1$ and $h_s$. According to lemma 2 $p$ has sent a skip message or an endorsement message on height $h - 1$. But we already showed that they couldn't have sent the skip message on $h - 1$, and therefore it must be that they sent an endorsement message on $h - 1$.

Thus all the honest block producers sent an endorsement message to $BP(h)$ on the block produced for height $h-1$. As we showed above $BP(h)$ would include all such endorsement messages, and therefore the block at height $h$ contains at least $t + 1$ endorsements on the block at height $h - 1$. □

## 3 Finality Gadget[3]

In this section we describe a Casper-like finality gadget, that we call Nightshade Finality Gadget, or Casper NFG.

Throughout this section we assume that there's a total of $n = 3f + 1$ block producers, of which $2f + 1$ are honest, online and follow the protocol, while the remaining $f$ can be offline or arbitrarily deviate from the protocol.

[3]Nightshade finality gadget was previously presented in [3]. The construction here replaces the definition of weight, provides guaranteed liveness and differs in other minor details.

## 3.1 Approvals

Approvals in our construction have the same purpose as votes in Casper FFG [4]. An approval by a block producer $v$[4] is an ordered tuple

$$\langle v, p, r \rangle$$

where $p$ is the block that is being approved, $r$ is a so-called *reference block*. The reference block must be in the ancestry of $p$.

We define a score $s(b)$ and a weight $w(b)$ of a block in section 3.4. For any two approvals any particular block producer issues

$$\langle v, p_1, r_1 \rangle$$

and

$$\langle v, p_2, r_2 \rangle$$

it must be that

1. $r_1 \neq r_2 \wedge s(r_1) \geq s(p_2) \wedge w(r_1) > w(p_2)$, or
2. $r_1 \neq r_2 \wedge s(r_2) \geq s(p_1) \wedge w(r_2) > w(p_1)$, or
3. $r_1 = r_2 \wedge (p_1 \in A(p_2) \vee p_2 \in A(p_1))$

where $A(b)$ denotes all the blocks in the ancestry of $b$.

In other words, if two approvals have the same reference block, the blocks they approve must be on the same chain, and if they have different reference blocks, the ranges of scores and weights of $(r, p)$ must not have any points in common, and moreover the approval that has higher scores must also have higher weights. Creating two approvals that violate these rules is a slashable behavior.

## 3.2 Algorithm

We alter the algorithm presented in section 2.3 in the following way:

When a block producer $v$ receives a block $b'$ witness of which has $t+1$ endorsements on some other block $b$ (thus, $b$ has *doomslug finality* from perspective of $v$), if *height*$(b)$ is the largest height for which $v$ knows a block to have *doomslug finality* and if $b$ is on the canonical chain according to the fork choice rule (see section 3.4 below), $v$ will produce an approval for $b$. $v$ will then repeatedly send such approval with any endorsement or skip-messages they sent to block proposers until the approval is included in some block and that block is finalized.

When producing an approval on $b$, if the last block $v$ approved is in the ancestry of $b$, $v$ should use the same reference block as in the last approval.

[4]While inconsistent with previous sections, we will use $v$ to refer to block producers when describing the finality gadget, and use $p$ to refer to the parent blocks of the approvals

Otherwise the reference block must be the lowest score block in the ancestry of $b$ that has higher weight and at no smaller score than the last block $v$ approved. If no such block exists in the ancestry of $b$, $v$ should skip producing an approval. Such approvals by construction cannot trigger the two slashable conditions listed above. See figure 3.

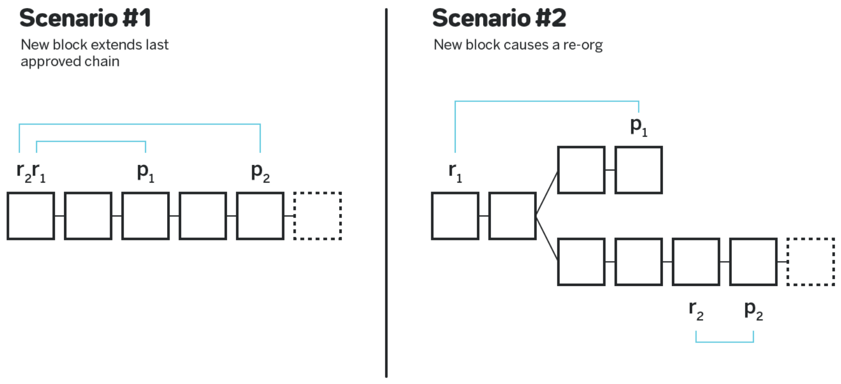

Scenario #1
New block extends last approved chain
r2r1
p1
p2
Scenario #2
New block causes a re-org
p1
r1
r2
p2

Figure 3: Approvals for a block that extends the last known canonical chain and a block that causes a reorg

## 3.3 Finality

For an approval $a = \langle v, p, r \rangle$ and a block $b$ we call $a$ a pre-vote on $b$ iff

1. $b$ is in the ancestry of $p$;
2. $r$ is in the ancestry of $b$.

We say a block $b_p$ has a *quorum pre-vote* for block $b$, if $b$ is in the ancestry of $b_p$ and there are $2f + 1$ block producers for whom there's an approval

$$a = \langle v, p, r \rangle$$

included in a block in the ancestry of $b_p$ such that $p$ is in the ancestry of $b_p$ and $a$ is a pre-vote on $b$.

For an approval $a = \langle v, p, r \rangle$ and a block $b$ we call $a$ a pre-commit on $b$ iff

1. $b$ is in the ancestry of $p$;
2. $r$ is in the ancestry of $b$;
3. There's a block $b_p$ in the ancestry of $p$ that has a quorum pre-vote on $b$.

We say a block $b_p$ has a *quorum pre-commit* for block $b$, if $b$ is in the ancestry of $b_p$ and there are $2f+1$ of block producers for whom there's an approval

$$a = \langle v, p, r \rangle$$

included in a block in the ancestry of $b_p$ such that $p$ is in the ancestry of $b_p$ and $a$ is a pre-commit on $b$.

As will be shown in section 3.5.1, no two blocks, assuming neither is in the ancestry of the other, can have a quorum pre-commit by the same set of block producers, unless at least $f$ of the block producers are slashed. Thus, a block that has a quorum pre-commit in practice can be considered final.

## 3.4 Fork Choice Rule

Here we define how a canonical chain should be chosen if multiple competing chains exist.

We first define concepts of the *weight* and *score* of a block, and then define the fork choice rule.

We define **block weight** of a block $B$ (denoted as $w(B)$) to be some function such that $w(B) > w(prev(B))$. For example, the weight can be just the height of the block.

Let $HQV(b)$ be the heaviest (in terms of weight) block for which a quorum pre-vote exists in the ancestry of $B$.

The **score** of a block (denoted as $s(B)$) is then:

$$s(B) = w(HQV(B))$$

To choose a canonical chain among a set of chains, one first computes the *score* of the tip of each chain, and then chooses the chain with the higher score of the tip.

If the scores are equal for the tips of two chains, the chain which has a tip with higher height is chosen. This is necessary to make the doomslug safety guarantee in section 2.4.1 meaningful (which states that a block is irreversible if no block with the same or higher height can be created).

## 3.5 Analysis

### 3.5.1 Safety

In this section we prove that two blocks such that neither is in the ancestry of the other cannot both be finalized (as in, have a quorum pre-commit) by the same block producers set.

For a chain with a tip $t$ and a block $b$ that is finalized in such a chain, we define as $QV(t, b)$ the lowest score block in the ancestry of $t$ that has a quorum pre-vote on $b$.

**Lemma 5.** *If a block $b_1$ is finalized in a chain with a tip $t_1$, no block $b_2$ such that $w(b_2) > w(b_1)$ can have a quorum pre-vote by the same block producers* *set in a chain with some tip $t_2$ if neither $b_1$ nor $b_2$ are in the ancestry of one another, unless at least $f$ block producers committed a slashable act.*

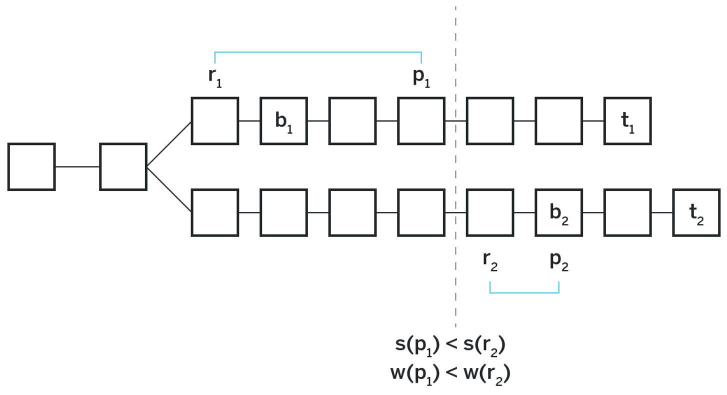


Figure 4: If a block $b_1$ is finalized on one chain, no heavier block can get a quorum pre-vote on any other chain

*Proof.* Say it is not the case, and there's such block $b_2$. Consider such block $b_2$ with the lowest height.

By definition of block finality, there's $2f + 1$ block producers that have an approval of a form $\langle v, p_1, r_1 \rangle$ such that $QV(t_1, b_1)$ is in the ancestry of $p_1$. Note that since $QV(t_1, b_1)$ has a quorum pre-vote on $b_1$, the score of $p_1$ is at least $w(b_1)$.

Since $b_2$ has a quorum pre-vote in the chain originating at $t_2$, there are $2f+1$ block producers that have an approval of a form $\langle v, p_2, r_2 \rangle$ such that $b_2$ is in the ancestry of $p_2$ and $r_2$ is in the ancestry of $b_2$. At least $f$ block producers have approvals of both forms.

Consider such block producer. Recall that $w(b_1) \leq w(b_2)$. Since $b_2$ is in the ancestry of $p_2$, and $r_1$ is in the ancestry of $b_1$, it follows that $w(r_1) \leq w(b_1) \leq w(b_2) \leq w(p_2)$. Since $p_2$ and $p_1$ are on two different chains, unless the block producer committed a slashable act, the ranges $(w(r_1), w(p_1))$ and $(w(r_2), w(p_2))$ do not intersect, and since $w(r_1) \leq w(p_2)$, it must be that $w(p_1) < w(r_2)$. Recall that for two approvals such that $w(p_1) < w(r_2)$ it is also required that $s(p_1) \leq s(r_2)$, as shown on figure 4.

Since the score of $p_1$ is at least $w(b_1)$, the score of $r_2$ is at least $w(b_1)$. But that implies that there is a block $b_2'$ with weight greater or equal to $w(b_1)$ that has a quorum pre-vote in the ancestry of $b_2$.

Thus, for a block $b_2$ with a weight higher than $w(b_1)$ that is not on the same chain as $b_1$ to exist, another such block $b_2'$ must exist in its ancestry, which contradicts the assumption that $b_2$ is the earliest such block. □

**Theorem 3** (Finality Gadget Safety)**.** *If a block $b_1$ is finalized in a chain with a tip $t_1$, any participant that has seen $t_1$ will have $b_1$ in their canonical chain, unless at least $f$ block producers committed a slashable act.*

*Proof.* Say it is not the case, and another chain is chosen as canonical. Since the chain originating at $t_1$ has a quorum pre-vote for $b_1$, its score is at least $w(b_1)$. Thus, the canonical chain that does not include $b_1$ must have a score of at least $w(b_1)$, otherwise it will not be chosen by the fork choice rule. By definition of score, that implies that there is a block other than $b_1$ with weight equal or greater than $w(b_1)$ that has a quorum pre-vote. This contradicts lemma 5. $\square$

### 3.5.2 Liveness

To show that Nightshade Finality Gadget has liveness under partially synchronous network, we first consider the case when no block producer commits a slashable act that causes two blocks $b_1$ and $b_2$ have *doomslug finality*, while neither $b_1$ nor $b_2$ are in the ancestry of one another.

**Lemma 6.** *For as long as no block producer commits a slashable act that causes two blocks $b_1$ and $b_2$ to have doomslug finality, while neither $b_1$ nor $b_2$ is in the ancestry of one another, any block $b$ on the canonical chain with doomslug finality will be finalized by the Nightshade Finality Gadget in finite time.*

*Proof.* Note that for as long as no two blocks $b_1$ and $b_2$ are produced such that they both have *doomslug finality* and are not in the ancestry of one another, according to algorithm described in section 3.2, the reference block of all the approvals for a particular block producer $v$ will be the same.

Consider some point in time $t$ at which $b$ is known by all honest block producers and is on the canonical chain.

Consider the largest score $s_{max}$ and the largest weight $w_{max}$ for which at least one block producer created an approval with such a score or weight in its parent block. Note that $s_{max}$ cannot be larger than the score of the tip of the canonical chain, since the canonical chain by definition has the tip with the largest score. Thus any block built on the canonical chain will have score at least $s_{max}$.

According to theorem 2, within finite amount of time since $t$ a new block that has *doomslug finality* will be produced. And then in finite time since then yet another block with *doomslug finality* will be produced. Since the weight is increasing, eventually a block with weight $w > w_{max}$ will have *doomslug finality*. All the honest block producers will be able to create an approval on it.

When $b_1$ is broadcasted, all honest block producers will create an approval such that

1. its reference block is $b_1$ or is in the ancestry of $b_1$.
2. its parent block is $b_1$ or has $b_1$ in its ancestry.

The above follows from the definition of the approval.

Since the approvals propagate and get included in the blocks unconditionally, within finite amount of time there will be some block $b_2$ which has in itself or in some blocks in its ancestry approvals from $2f + 1$ honest block producers of the form described above. From perspective of $b_2$, $b_1$ will have quorum pre-vote.

Again due to theorem 2 within finite amount of time there will be some block $b_3$ that has $b_2$ in its ancestry that has *doomslug finality*. Each honest block producer will then generate an approval such that

1. its reference block is in the ancestry of $b_1$, since the reference block in all the approvals in the absence of forks is the same.

2. its parent block has $b_3$ in its ancestry.

Since the approvals propagate and get included in the blocks unconditionally, within finite amount of time there will be some block $b_4$ which has in itself or in some blocks in its ancestry approvals from $2f + 1$ honest block producers of the form described above. From perspective of $b_4$ $b_1$ will have quorum pre-commit.

Thus, within finite amount of time since $t$ $b_1$ will become final. Once $b_1$ is final, all the blocks in its ancestry, including $b$, are also final. □

To provide a formal proof for the following theorem we would need to better define how slashing works and propagates, which is outside the scope of this paper. Instead we provide a less formal proof that only relies on an assumption that once a block producer is slashed, within finite amount of time they can no longer produce blocks, endorsements, skip-messages and approvals, and therefore cannot commit slashable acts.

**Theorem 4** (Finality Gadget Liveness). *From any moment in time $t_0$, within finite amount of time at least one block that was not previously finalized by the Nightshade Finality Gadget will be finalized.*

*Proof.* Say it is not the case.

According to theorem 2, within some finite time after $t_0$ some block $b$ that previously didn't have *doomslug finality* will have *doomslug finality*. Due to lemma 6 unless some block producer commits a slashable act, $b$ will get finalized by the Nightshade Finality Gadget. Since we assume no new block can get finalized by the Nightshade Finality Gadget, it implies that at least one block producer $v$ committed some slashable act.

Within finite amount of time $v$ can no longer commit slashable acts. Within finite amount of time after that a new block $b'$ will have *doomslug finality*. Again, for that block to not get finalized by the Nightshade Finality Gadget, at least one *other* block producer $v'$ needs to commit some slashable act.

Since the number of malicious actors is finite, at some point all the malicious actors will be slashed, and no block producers capable of committing slashable acts will remain. At that point the very next block that will have *doomslug finality* will become finalized due to lemma 6, causing a contradiction. □

## 4 Conclusion

In this paper we presented a block production gadget and a finality gadget constructions that under synchronous network allow block producers produce blocks such that after a single round of communication a block is irreversible unless at least one block producer is slashed, and after two rounds of communication a block is irreversible unless one third of all the block producers are slashed.

While we require synchronous network for the performance guarantees, the safety is guaranteed even under asynchronous network, and the liveness is guaranteed under partially synchronous network.

Doomslug and Nightshade Finality Gadget are both used in NEAR Protocol, which is a sharded blockchain protocol. The sharding design of NEAR is presented in [5]. NEAR reference client is being developed as an opensource project, you can follow it at `https://github.com/nearprotocol/nearcore`.

## References

[1] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938, 2018.

[2] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO*, pages 357–388. Springer, 2017.

[3] Alex Skidanov. Fast finality and resilience to long range attacks with proof of space-time and casper-like finality gadget, 2019. `http://near.ai/post`.

[4] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

[5] Illia Polosukhin Alex Skidanov. Nightshade: Near protocol sharding design, 2019. `http://near.ai/nightshade`.