# Incentives in Casper the Friendly Finality Gadget

Vitalik Buterin
Ethereum Foundation

July 13, 2017

**Abstract**

We give an introduction to the incentives in the Casper the Friendly Finality Gadget protocol, and show how the protocol behaves under individual choice analysis, collective choice analysis and griefing factor analysis. We define a "protocol utility function" that represents the protocol's view of how well it is being executed, and connect the incentive structure to the utility function. We show that (i) the protocol is a Nash equilibrium assuming any individual validator's deposit makes up less than $\frac{1}{3}$ of the total, (ii) in a collective choice model, where all validators are controlled by one actor, harming protocol utility hurts the cartel's revenue, and there is an upper bound on the ratio between the reduction in protocol utility from an attack and the cost to the attacker, and (iii) the griefing factor can be bounded above by 1, though we will prefer an alternative model that bounds the griefing factor at 2 in exchange for other benefits.

## 1 Introduction

In the Casper protocol, there is a set of validators, and in each epoch validators have the ability to send two kinds of messages:

$$[PREPARE, epoch, hash, epoch_{source}, hash_{source}]$$

and

$$[COMMIT, epoch, hash]$$

Each validator has a *deposit size*; when a validator joins their deposit size is equal to the number of coins that they deposited, and from there on each validator's deposit size rises and falls as the validator receives rewards and penalties. For the rest of this paper, when we say "$\frac{2}{3}$ of validators", we are referring to a *deposit-weighted* fraction; that is, a set of validators whose combined deposit size equals to at least $\frac{2}{3}$ of the total deposit size of the entire set of validators. We also use "$\frac{2}{3}$ commits" as shorthand for "commits from $\frac{2}{3}$ of validators".

If, during an epoch $e$, for some specific checkpoint hash $h$, $\frac{2}{3}$ prepares are sent of the form

$$[PREPARE, e, h, epoch_{source}, hash_{source}]$$

with some specific $epoch_{source}$ and some specific $hash_{source}$, then $h$ is considered *justified*. If $\frac{2}{3}$ commits are sent of the form

$$[COMMIT, e, h]$$

then $h$ is considered *finalized*. The $hash$ is the block hash of the block at the start of the epoch, so a $hash$ being finalized means that that block, and all of its ancestors, are also finalized. An "ideal execution" of the protocol is one where, during every epoch, every validator prepares and commits some block hash at the start of that epoch, specifying the same $epoch_{source}$ and $hash_{source}$. We want to try to create incentives to encourage this ideal execution.

Possible deviations from this ideal execution that we want to minimize or avoid include:

- Any of the four slashing conditions get violated.
- During some epoch, we do not get $\frac{2}{3}$ commits for the $hash$ that received $\frac{2}{3}$ prepares.
- During some epoch, we do not get $\frac{2}{3}$ prepares for the same $(h, hash_{source}, epoch_{source})$ combination.

From within the view of the blockchain, we only see the blockchain's own history, including messages that were passed in. In a history that contains some blockhash $H$, our strategy will be to reward validators who prepared and committed $H$, and not reward prepares or commits for any hash $H\prime \neq H$. The blockchain state will also keep track of the most recent hash in

its own history that received $\frac{2}{3}$ prepares, and only reward prepares whose $epoch_{source}$ and $hash_{source}$ point to this hash. These two techniques will help to "coordinate" validators toward preparing and committing a single hash with a single source, as required by the protocol.

# 2 Rewards and Penalties

We define the following constants and functions:

- $p$: determines how the rewards and penalties paid or deducted from each validator decrease as the total deposit size increases
- $k$: a constant determining the base reward and penalty size
- $NCP$ ("non-commit penalty"): the penalty for not committing, if there was a justified hash which the validator *could* have committed
- $NCCP(\alpha)$ ("non-commit collective penalty"): if $\alpha$ of validators are not seen to have committed during an epoch, and that epoch had a justified hash so any validator *could* have committed, then all validators are charged a penalty proportional to $NCCP(\alpha)$
- $NPP$ ("non-prepare penalty"): the penalty for not preparing
- $NPCP(\alpha)$ ("non-prepare collective penalty"): if $\alpha$ of validators are not seen to have prepared during an epoch, then all validators are charged a penalty proportional to $NCCP(\alpha)$
- $f(e, LFE)$: a factor applied to all rewards and penalties that depends on the current epoch $e$ and the last finalized epoch $LFE$. Note that in a "perfect" protocol execution, $e - LFE$ always equals 1.

Note that preparing and committing does not guarantee that the validator will not incur $NPP$ and $NCP$; it could be the case that either because of very high network latency or a malicious majority censorship attack, the prepares and commits are not included into the blockchain in time and so the incentivization mechanism does not know about them. For $NPCP$ and $NCCP$ similarly, the $\alpha$ input is the portion of validators whose prepares and commits are *included*, not the portion of validators who *tried to send* prepares and commits.

When we talk about preparing and committing the "correct value", we are referring to the $hash$ and $epoch_{source}$ and $hash_{source}$ recommended by the protocol state, as described above.

We now define the following reward and penalty schedule, where a validator with deposit size $V_d$ gets a reward or penalty equal to $V_d$ times the values given below:

- Let $BIR = \frac{k}{D^p}$ (the "base interest rate")
- All validators get a reward of $BIR$
- If a validator did not prepare the correct value, they are penalized $BIR * f * NPP$
- If $p_p$ validators prepared the correct value, every validator is penalized $BIR * f * NPCP(1 - p_p)$
- If a validator did not commit the correct value, and the protocol sees that the correct value was justified so they *could* have committed, they are penalized $BIR * f * NCP$
- If $p_c$ validators committed the correct value, and the protocol sees that the correct value was justified so they *could* have committed, every validator is penalized $BIR * f * NCCP(1 - p_c)$

This is the entirety of the incentivization structure.

# 3 Claims

We seek to prove the following:

- Following the protocol is a Nash equilibrium; that is, if each validator has less than $\frac{1}{3}$ of total deposits, their economic welfare is maximized by preparing and committing the same value as everyone else.
- Even if all validators collude, the ratio between the harm incurred by the protocol and the penalties paid by validators is bounded above by some constant. Note that this requires a measure of "harm incurred by the protocol"; we will discuss this in more detail later.

- The *griefing factor*, the ratio between penalties incurred by non-attacking validators and penalties incurred by attacking validators, in any attack, is bounded above by some global constant, even in the case where the attacker has a majority of all validators.

# 4 Individual choice model

The individual choice analysis is simple. Suppose that some chain $C$ is the chain that you expect to be accepted as the main chain in the future (if all other validators are preparing and committing on $C$, then this will be the main chain because the fork choice rule takes commits into account). Suppose $H$ is the most recent block hash on $C$, and $(epoch_{source}, hash_{source})$ is the source data that $C$ expects validators to prepare with. A validator can avoid penalties by sending a prepare with $H, epoch_{source}, hash_{source}$ and sending a commit with $H$. Sending a commit does restrain the validator's future behavior because of the PREPARE_COMMIT_CONSISTENCY slashing condition, but if more than $\frac{2}{3}$ of validators are preparing and committing then this is not an issue because the epoch will itself receive $\frac{2}{3}$ prepares and so the validator will be able to use the current epoch as their source in the next epoch. Hence, it is in a validator's interest to be preparing and comitting on the same chain as everyone else.

# 5 Collective choice model

To model the protocol in a collective-choice context, we will first define a *protocol utility function*. The protocol utility function defines "how well the protocol execution is doing". The protocol utility function cannot be derived mathematically; it can only be conceived and justified intuitively.

Our protocol utility function is:
$U = \sum_{e=0}^{e_c} -log_2(e - max[e\prime < e, e\prime finalized]) - M * F$
Where:

- $e$ is the current epoch, going from epoch 0 to $e_c$, the current epoch
- $e\prime$ is the last finalized epoch before $e$
- $M$ is a very large constant

- $F$ is 1 if a safety failure has taken place, otherwise 0

The second term in the function is easy to justify: safety failures are very bad. The first term is trickier. To see how the first term works, consider the case where every epoch where $emodN = 0$ is finalized and other epochs are not. The average total over each $N$-epoch slice will be roughly $\sum_{i=1}^{N} -log_2(i) \approx N * (log_2(N) - \frac{1}{ln(2)})$. Hence, the utility per block will be roughly $-log_2(N)$. This basically states that a blockchain with some finality time $N$ has utility roughly $-log(N)$, or in other words *increasing the finality time of a blockchain by a constant factor causes a constant loss of utility.* The utility difference between 1 minute finality and 2 minute finality is the same as the utility difference between 1 hour finality and 2 hour finality.

This can be justified in two ways. First, one can intuitively argue that a user's psychological estimation of the discomfort of waiting for finality roughly matches this kind of logarithmic utility schedule. Second, one can look at various blockchain use cases, and see that they are roughly logarithmically uniformly distributed along the range of finality times between around 200 miliseconds ("Starcraft on the blockchain") and one week (land registries and the like).

Now, we need to show that, for any given total deposit size, $\frac{loss\_to\_protocol\_utility}{validator\_penalties}$ is bounded. There are two ways to reduce protocol utility: cause a safety failure, and do not finalize epochs. In the first case, validators lose a large amount of deposits for violating the slashing conditions. In the second case, in a chain that has not been finalized for $k$ epochs, the penalty to attackers is $min(\frac{1}{3} * (NPP + NPCP), \frac{1}{3} * (NCP + NCCP)) * BIR * floor(log_2(k))$, which is equal to the loss of protocol utility multiplied by $BIR * min(\frac{1}{3} * (NPP + NPCP), \frac{1}{3} * (NCP + NCCP))$, which for any given total deposit size is a constant.

# 6 Griefing factor analysis

Griefing factor analysis is important because it is one way to quanitfy the risk to honest validators. In general, if all validators are honest, and if network latency stays below the length of an epoch, then validators face zero risk beyond the usual risks of losing or accidentally divulging access to their private keys. In the case where malicious validators exist, they can interfere in the protocol in ways that cause harm to both themselves and honest validators.

We can approximately define the "griefing factor" as follows:

A strategy used by a coalition in a given mechanism exhibits a *griefing factor* $B$ if it can be shown that this strategy imposes a loss of $B * x$ to those outside the coalition at the cost of a loss of $x$ to those inside the coalition. If all strategies that cause deviations from some given baseline state exhibit griefing factors less than or equal to some bound B, then we call B a *griefing factor bound*.

A strategy that imposes a loss to outsiders either at no cost to a coalition, or to the benefit of a coalition, is said to have a griefing factor of infinity. Proof of work blockchains have a griefing factor bound of infinity because a 51% coalition can double its revenue by refusing to include blocks from other participants and waiting for difficulty adjustment to reduce the difficulty and increase their rewards. With selfish mining, the griefing factor may be infinity for coalitions of size as low as 23.21%.

Let us start off our griefing analysis by not taking into account validator churn, so all dynasties are identical. Because the equations involved are fractions of linear equations, we know that small churn will only lead to small changes in the results. In Casper, we can identify the following deviating strategies:

1. A minority of validators do not prepare, or prepare incorrect values.

2. (Mirror image of 1) A censorship attack where a majority of validators does not accept prepares from a minority of validators (or other isomorphic attacks such as waiting for the minority to prepare hash $H1$ and then preparing $H2$, making $H2$ the dominant chain and denying the victims their rewards)

3. A minority of validators do not commit.

4. (Mirror image of 3) A censorship attack where a majority of validators does not accept commits from a minority of validators

Notice that, from the point of view of griefing factor analysis, it is immaterial whether or not a given epoch was prepared or committed. The reward and penalty schedule only pays attention to prepares and commits for the purpose of setting $f$, the value proportional to the logarithm of the time since the last finalized epoch. This value scales penalties evenly for all participants, so it does not affect griefing factors.

Let us now analyze the griefing factors:

| Attack | Amount lost by attacker | Amount lost by victims |
|---|---|---|
| Minority of size $\alpha < \frac{1}{2}$ non-prepares | $NCP * \alpha + NCCP * \alpha^2$ | $NCCP * \alpha * (1 - \alpha)$ |
| Majority censors $\alpha < \frac{1}{2}$ minority prepares | $NCCP * \alpha * (1 - \alpha)$ | $NCP * \alpha + NCCP * \alpha^2$ |

In general, we see a perfect symmetry between the non-commit case and the non-prepare case, so we can assume $\frac{NCCP}{NCP} = \frac{NPCP}{NPP}$. Also, from a protocol utility standpoint, we can make the observation that seeing $\frac{1}{3} \leq p_c < \frac{2}{3}$ commits is better than seeing fewer commits, as it gives at least some economic security against finality reversions, so we do want to reward this scenario more than the scenario where we get $\frac{1}{3} \leq p_c < \frac{2}{3}$ prepares. Another way to view the situation is to observe that $\frac{1}{3}$ non-prepares causes *everyone* to non-commit, so it should be treated with equal severity.

In the normal case, anything less than $\frac{1}{3}$ commits provides no economic security, so we can treat $p_c < \frac{1}{3}$ commits as equivalent to no commits; this thus suggests $NPC = 2 * NCC$.

# 7 Conclusions

The above analysis shows Casper's basic properties in the context of an individual-choice model, a collective-choice model where the validator set is modeled as a single player, and a model where one coalition is trying to cause other validators to lose money possibly at some cost to itself. Non-economic honest-majority models are out of scope, as is the proof that causing a safety failure requires a large number of slashed validators, as those topics are covered elsewhere. More complex economic attacks involving extortion, blackmail and validator discouragement are not covered here, although the griefing factor analysis made here does serve as a foundation for the analyses of these topics.

Optimal selfish mining strategies in Bitcoin; Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar: https://arxiv.org/pdf/1507.06183.pdf