Incentives in Casper the Friendly Finality Gadget

Vitalik Buterin
Ethereum Foundation

August 7, 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 show the connection between the incentive structure that we present and 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:

epoch
hash
$hash_{source}$
$epoch_{source}$

(a) **PREPARE**

epoch
hash

(b) **COMMIT**

Table 1: The schematic of the **PREPARE** and **COMMIT** messages.

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

$$[PREPARE, e, \mathrm{h}, e_\star, \mathrm{h}_\star] \tag{1}$$

with some specific $e_\star$ and some specific $\mathrm{h}_\star$, then h is considered *justified*. If $\frac{2}{3}$ sends a Commit of the form

$$[COMMIT, e, \mathrm{h}] \tag{2}$$

then h is considered *finalized*. The h is the block hash of the block at the start of the epoch, so a h being finalized means that block, and all of its ancestors, are finalized. An "ideal execution" of the protocol is one where, at the start of every epoch, every validator Prepares and Commits some block hash, specifying the same $e_\star$ and $\mathrm{h}_\star$. 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 [1] get violated.
- During some epoch, we do not get $\frac{2}{3}$ Prepares for the same $(h, \mathrm{h}_\star, e_\star)$ combination.
- During some epoch, we do not get $\frac{2}{3}$ Commits for the *hash* that received $\frac{2}{3}$ prepares. [there can be multiple hashes that received 2/3 prepares, right?]

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 is to reward validators who prepared and committed H, and not reward prepares or commits for any hash $H' \neq H$.

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 $e_\star$ and $\mathrm{h}_\star$ 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:

- BIR(**TD**): determines the base interest rate paid to a validator, taking as an input the current total quantity of deposited ether.
- BP($\mathbf{TD}, e, e_{\leftarrow}$): determines the "base penalty constant" - a value expressed as a percentage rate that is used as the "scaling factor" for all penalties; for example, if at the current time $\mathrm{BP}(\cdot,\cdot,\cdot) = 0.001$, then a penalty of size 1.5 means a validator loses 0.15% of their deposit. Takes as inputs the current total quantity of deposited ether **TD**, the current epoch e and the last finalized epoch $e_{\leftarrow}$. Note that in a "perfect" protocol execution, $e - e_{\leftarrow}$ always equals 1.
- NCP ("non-commit penalty"): the penalty for not committing, if there was a justified hash which the validator *could* have committed. NCP is a constant and $\mathrm{NCP} > 0$.
- NCCP(α) ("non-commit collective penalty"): if α 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(α). Must be monotonically increasing, and satisfy $NCCP(0) = 0$.
- NPP ("non-prepare penalty"): the penalty for not preparing. NPP is a constant and $\mathrm{NPP} > 0$.
- NPCP(α) ("non-prepare collective penalty"): if α of validators ($0 \leq \alpha \leq 1$) are not seen to have prepared during an epoch, then all validators are charged a penalty proportional to $NCCP(\alpha)$. Must be monotonically increasing, and satisfy $\mathrm{NPCP}(0) = 0$.

Aside from BP (which has no units), all expressions have units of *coins*.

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 α input is the proportion 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 $e_\star$ and $\mathrm{h}_\star$ recommended by the protocol state, as described above.

We now define the following reward and penalty schedule, which runs every epoch.

- Let **TD** be the current *total amount of deposited ether*, and $e - e_{\leftarrow}$ be the number of epochs since the last finalized epoch.
- All validators get a reward of $\mathrm{BIR}(\mathbf{TD})$ every epoch (eg. if $\mathrm{BIR}(\mathbf{TD}) = 0.0002$ then a validator with $10,000$ coins deposited gets a per-epoch reward of 2 coins)
- If the protocol does not see a Prepare from a given validator during the given epoch, they are penalized $\mathrm{BP}(\mathbf{TD}, e, e_{\leftarrow}) * \mathrm{NPP}$
- If the protocol saw Prepares from proportion p_P validators during the given epoch, *every* validator is penalized $\mathrm{BP}(\mathbf{TD}, e, e_{\leftarrow}) * \mathrm{NPCP}(1 - p_P)$
- If the protocol does not see a Commit from a given validator during the given epoch, and a Prepare was justified so a Commit *could have* been seen, they are penalized $\mathrm{BP}(\mathbf{TD}, E, e_{\leftarrow}) * \mathrm{NCP}$.
- If the protocol saw Commits from proportion p_C validators during the previous epoch, and a Prepare was justified so a validator *could have* committed, then *every* validator is penalized $\mathrm{BP}(\mathbf{TD}, e, e_{\leftarrow}) * \mathrm{NCCP}(1 - p_P)$

This is the entirety of the incentivization structure, though without functions and constants defined; we will define these later, attempting as much as possible to derive the specific values from desired objectives and first principles. For now we will only say that all functions output non-negative values for any input within their range.

3 Claims

We seek to prove the following:

1. If each validator has less than $\frac{1}{3}$ of total deposits, i.e., $\max_i(D) \leq \frac{\mathbf{TD}}{3}$, then preparing and committing the value suggested by the proposal mechanism is a Nash equilibrium.
2. 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.
3. The *griefing factor*, the ratio between penalties incurred by validators who are victims of an attack and penalties incurred by the validators that carried out the attack, even when the attacker holds a majority of the total deposit, the griefing factor is upperbounded by 2.

4 Individual choice analysis

The individual choice analysis is simple. Suppose that the proposal mechanism selects a hash h to Prepare for epoch e, and the Casper incentivization mechanism specifies some $e_\star$ and $\mathrm{h}_\star$. Because, as per definition of the Nash equilibrium, we are assuming that all validators except for one particular validator that we are analyzing is following the equilibrium strategy, we know that $\geq \frac{2}{3}$ of validators prepared in the last epoch and so $e_\star = e - 1$, and $\mathrm{h}_\star$ is the direct parent of h.

Hence, the PREPARE_COMMIT_CONSISTENCY slashing condition poses no barrier to preparing $(e, H, e_\star, \mathrm{h}_\star)$. Since, in epoch e, we are assuming that all other validators *will* Prepare these values and then Commit H, we know H will be a hash in the main chain, and so a validator will pay a penalty proportional to NPP (plus a further penalty from their marginal contribution to the NPCP penalty) if they do not Prepare $(e, H, e_\star, \mathrm{h}_\star)$, and they can avoid this penalty if they do Prepare these values.

We are assuming that there are $\frac{2}{3}$ prepares for $(e, H, e_\star, \mathrm{h}_\star)$, and so PREPARE_REQ poses no barrier to committing H. Committing H allows a validator to avoid NCP (as well as their marginal contribution to NCCP). Hence, there is an economic incentive to Commit H. This shows that, if the proposal mechanism

succeeds at presenting to validators a single primary choice, preparing and committing the value selected by the proposal mechanism is a Nash equilibrium.

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". Although the protocol utility function cannot be derived mathematically, it can only be conceived and justified intuitively.

Our protocol utility function is,

$$U = \sum_{k=0}^{e} -\log_2\left[k - e_{\leftarrow}\right] - M * F \ . \tag{3}$$

Where:

- e is the current epoch, starting from 0.
- $e_{\leftarrow}$ is the index of the last finalized epoch before e.
- M is a very large constant.
- F is an Indicator function. It returns 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 such that e mod N, for some N, is zero is finalized and other epochs are not. The average total over each N-epoch slice will be roughly $\sum_{i=1}^{N} -\log_2(i) \approx N * \left[\log_2(N) - \frac{1}{\ln(2)}\right]$. 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 a logarithmic schedule. At the very least, the difference between 3600sec and 3610sec finality feels much more negligible than the difference between 1sec finality and 11sec finality, and so the claim that the difference between 10sec finality and 20sec finality is similar to the difference between 1 hour finality and 2 hour finality should not seem farfetched. [1]

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: (i) cause a safety failure, and (ii) have $\geq \frac{1}{3}$ of validators not Prepare or not Commit to prevent finality. 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 $e - e_{\leftarrow}$ epochs, the penalty to attackers is at least,

$$\min\left[\text{NPP} * \frac{1}{3} + \text{NPCP}\left(\frac{1}{3}\right), \text{NCP} * \frac{1}{3} + \text{NCCP}\left(\frac{1}{3}\right)\right] * \text{BP}(\mathbf{TD}, e, e_{\leftarrow}) \ . \tag{4}$$

To enforce a ratio between validator losses and loss to protocol utility, we set,

$$\text{BP}(\mathbf{TD}, e, e_{\leftarrow}) \equiv \frac{k_1}{\mathbf{TD}^p} + k_2 * \lfloor \log_2(e - e_{\leftarrow}) \rfloor \ . \tag{5}$$

[1] 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). [add a citation for this or delete.]

what is p in the in the above equation?

The first term serves to take profits for non-committers away; the second term creates a penalty which is proportional to the loss in protocol utility.

This connection between validator losses and loss to protocol utility has several consequences. First, it establishes that harming the protocolexecution is costly, and harming the protocol execution more costs more. Second, it establishes that the protocol approximates the properties of a potential game [cite]. Potential games have the property that Nash equilibria of the game correspond to local maxima of the potential function (in this case, protocol utility), and so correctly following the protocol is a Nash equilibrium even in cases where a coalition has more than $\frac{1}{3}$ of the total validators. Here, the protocol utility function is not a perfect potential function, as it does not always take into account changes in the *quantity* of prepares and commits whereas validator rewards do, but it does come close.

6 Griefing factor analysis

Griefing factor analysis is important because it provides a way to quanitfy the risk to honest validators. In general, if all validators are honest, and if network latency stays below half [half, right?] the length of an epoch, then validators face zero penalties to their respective deposits. In the case where malicious validators exist, however, they can interfere in the protocol in ways that penalize themselves as well as honest validators.

We define the "griefing factor" as,

$$\mathcal{GF}(\mathfrak{D}, C) \equiv \max_{S \in strategies(T \setminus C)} \frac{loss(C)}{\min[0, loss(Players \setminus C)]} \ . \tag{6}$$

I need to work on this equation more. I don't like it yet.

Definition 1. *A strategy used by a coalition in a given mechanism has 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 have griefing factors less than or equal to some bound* B*, then we call* B *a* griefing factor bound. *[I plan to write this in terms of classical game theory.]*

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. With selfish mining, the griefing factor may be infinity for coalitions of size as low as 23.21%. [citation?]

Let us start off our griefing analysis by not taking into account validator churn, so the validator set is always the same. 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 H_1 and then preparing H_2, making H_2 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 any hash in a given epoch was justified or finalized. The Casper mechanism only pays attention to finalization in order to calculate $\text{BP}(D, e, e_{\leftarrow})$, the penalty scaling factor. This value scales penalties evenly for all participants, so it does not affect griefing factors.

Figure 1: Plotting the griefing factor as a function of the proportion of players coordinating to grief.

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

Let us now analyze the attack types:

In general, we see a perfect symmetry between the non-Commit case and the non-Prepare case, so we can assume $\frac{\text{NCCP}(\alpha)}{\text{NCP}} = \frac{\text{NPCP}(\alpha)}{\text{NPP}}$. 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 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 $\text{NPP} = 2 * \text{NCP}$. We can also normalize $\text{NCP} = 1$.

Now, let us analyze the griefing factors, to try to determine an optimal shape for NCCP. The griefing factor for non-committing is,

$$\mathcal{GF} = \frac{(1 - \alpha) * \text{NCCP}(\alpha)}{\alpha * (1 + \text{NCCP}(\alpha))} \,. \tag{7}$$

The griefing factor for censoring is the inverse of this. If we want the griefing factor for non-committing to equal one, then we could compute:

$$\alpha * (1 + \text{NCCP}(\alpha)) = (1 - \alpha) * \text{NCCP}(\alpha) \quad (8)$$

$$\frac{1 + \text{NCCP}(\alpha)}{\text{NCCP}(\alpha)} = \frac{1 - \alpha}{\alpha} \quad (9)$$

$$\frac{1}{\text{NCCP}(\alpha)} = \frac{1 - \alpha}{\alpha} - 1 \quad (10)$$

$$\text{NCCP}(\alpha) = \frac{\alpha}{1 - 2\alpha} \quad (11)$$

Note that for $\alpha = \frac{1}{2}$, this would set the NCCP to infinity. Hence, with this design a griefing factor of 1 is infeasible. We *can* achieve that effect in a different way - by making NCP itself a function of α; in this case, NCCP = 1 and NCP = $\max[0, 1 - 2 * \alpha]$ would achieve the desired effect. If we want to keep the formula for NCP constant, and the formula for NCCP reasonably simple and bounded, then one alternative is to set $\text{NCCP}(\alpha) = \frac{\alpha}{1-\alpha}$; this keeps griefing factors bounded between $\frac{1}{2}$ and 2.

7 Pools

In a traditional (ie. not sharded or otherwise scalable) blockchain, there is a limit to the number of validators that can be supported, because each validator imposes a substantial amount of overhead on the system. If we accept a maximum overhead of two consensus messages per second, and an epoch time of 1400 seconds, then this means that the system can handle 1400 validators (not 2800 because we need to count prepares and commits). Given that the number of individual users interested in staking will likely exceed 1400, this necessarily means that most users will participate through some kind of "stake pool".

There are several possible kinds of stake pools:

- **Fully centrally managed**: users $B_1 \ldots B_n$ send coins to pool operator A. A makes a few deposit transactions containing their combined balances, fully controls the Prepare and Commit process, and occasionally withdraws one of their deposits to accommodate users wishing to withdraw their balances. Requires complete trust.
- **Centrally managed but trust-reduced**: users $B_1 \ldots B_n$ send coins to a pool contract. The contract sends a few deposit transactions containing their combined balances, assigning pool operator A control over the Prepare and Commit process, and the task of keeping track of withdrawal requests. A occasionally withdraws one of their deposits to accommodate users wishing to withdraw their balances; the withdrawals go directly into the contract, which ensures each user's right to withdraw a proportional share. Users need to trust the operator not to get their deposits penalized, but the operator cannot steal the coins. The trust requirement can be reduced further if the pool operator themselves contributes a large portion of the coins, as this will disincentivize them from staking maliciously.
- **2-of-3**: a user makes a deposit transaction and specifies as validation code a 2-of-3 multisig, consisting of (i) the user's online key, (ii) the pool operator's online key, and (iii) the user's offline backup key. The need for two keys to sign off on a prepare, Commit or withdraw minimizes key theft risk, and a liveness failure on the pool side can be handled by the user using their backup key.
- **Multisig managed**: users $B_1 \ldots B_n$ send coins to a pool contract that works in the exact same way as a centrally managed pool, except that a multisig of several semi-trusted parties needs to approve each Prepare and Commit message.
- **Collective**: users $B_1 \ldots B_n$ send coins to a pool contract that that works in the exact same way as a centrally managed poolg , except that a threshold signature of at least portion p of the users themselves (say, $p = 0.6$) needs to approve each Prepare and Commit messagge.

We expect pools of different types to emerge to accomodate smaller users. In the long term, techniques such as blockchain sharding will make it possible to increase the number of users that can validate directly, and extensions to allow validators to temporarily "drop out" from the validator set when they are offline can mitigate liveness risk.

8 Conclusions

The above analysis gives a parametrized scheme for incentivizing in Casper, and shows that it is a Nash equilibrium in an uncoordinated-choice model with a wide variety of settings. We then attempt to derive one possible set of specific values for the various parameters by starting from desired objectives, and choosing values that best meet the desired objectives. This analysis does not include non-economic attacks, as those are covered by other materials, and does not cover more advanced economic attacks, including extortion and discouragement attacks. We hope to see more research in these areas, as well as in the abstract theory of what considerations should be taken into account when designing reward and penalty schedules.

Future Work. [fill me in]

Acknowledgements. We thank Virgil Griffith for review.

9 References

References

[1] V. Buterin. Minimal slashing conditions, 03 2017.

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

Potential games; Dov Monderer and Lloyd Shapley: `http://econpapers.repec.org/article/eeegamebe/v\_3a14\_3ay\_3a1996\_3ai\_3a1\_3ap\_3a124-143.htm`

Appendix

10 Unused text

[This is where text goes that for which a home hasn't been found yet. If no home is found, it will be deleted.]

Two other reasons to participate in stake pools are (i) to mitigate *key theft risk* (i.e. an attacker hacking into their online machine and stealing the key), and (ii) to mitigate *liveness risk*, the possibility that the validator node will go offline, perhaps because the operator does not have the time to manage a high-uptime setup.