Rewards and Incentives in Casper the Friendly Finality Gadget

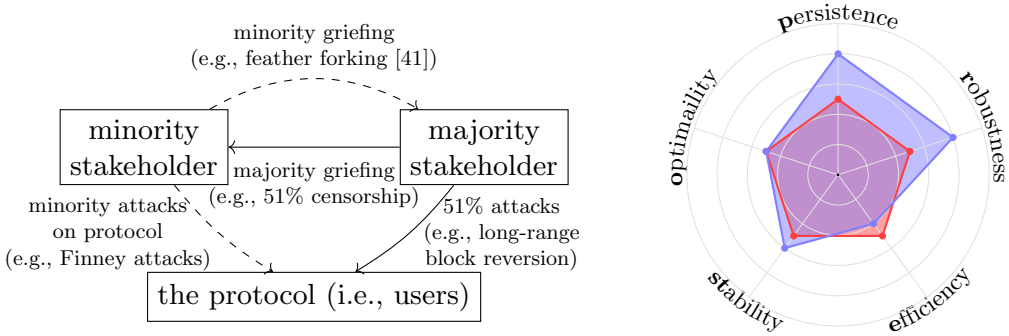
Vitalik Buterin[1], Daniël Reijsbergen[2] and Georgios Piliouras[2]

[1]Ethereum Foundation
[2]Singapore University of Technology and Design

Abstract. We present a formal description and analysis of the incentives in Casper the Friendly Finality Gadget, the Proof-of-Stake checkpointing protocol that will be used in the Ethereum blockchain. We find that it improves the stability, robustness, and persistence of the underlying blockchain, at a cost to efficiency. For stability, we show that for the new class of agents — called validators — it is a Nash equilibrium to follow the protocol. For robustness, we establish bounds on the damage that censoring attackers can do relative to their own losses, and show that their attacks only start to pay off after unrealistically long time periods. For persistence, we analyse the protocol's built-in mechanism for recovery from 51% attacks that is unavailable to pure Proof-of-Work protocols such as Ethereum's current version and Bitcoin.

1 Introduction

Proof-of-Work (PoW), the blockchain mechanism where 'miners' earn the right to extend a chain by provably spending computational power on block creation, is increasingly under scrutiny. Since it relies on the notion that the majority



(a) (b)

Fig. 1: Left: Triangle of harm [8]. Right: Illustrative comparison between Ethereum without (red) and with (blue) Casper FFG using PRESTO.

(in terms of computational power) is always right, a dishonest majority pool or coalition is unconstrained in its ability to misbehave. The main forms of misbehaviour are *censorship* (where the majority refuses to build on certain blocks, e.g., when they contain a blacklisted transaction), and so-called *51% attacks* like long-range block reversion (where a client must drop several blocks simultaneously after encountering a longer chain built in secret, which enables double-spending). In a 'large' network like Bitcoin, the vast amount of computational resources spent globally on mining provides security against 51% attacks, but at the cost of a colossal energy footprint [21]. However, the need for such security has been witnessed recently by 51% attacks on several smaller blockchain platforms, including Bitcoin Gold, Monacoin, Zencash, and Litecoin Cash [34].

One of the main alternatives to PoW is Proof-of-Stake (PoS), in which the right to propose blocks is earned not through computational power, but by temporarily locking tokens on the blockchain. This setting has a similar rent-seeking competition [6] that invites economic agents to participate (and hence protection against 51% attacks) as in PoW, but with the well-known benefit of a much lower energy cost. However, PoS has two additional lesser-known advantages. First, in PoS the 'stake' is always intrinsic to the blockchain, in contrast to the 'stake' in PoW (i.e., the computational resources that may be blockchain-specific, in the case of ASICs, or not). This allows the protocol to ensure that all participants are penalised when a minority is censored, instead of just the victims. Second, if aggrieved minorities in PoS fork away from the canonical chain [27], then the majority will lose most of their deposits on the new chain if they continue to vote on the old chain. This leads to a permanently diminished capacity to propose blocks for misbehaving stakeholders that is unavailable in any PoW context.

As these phenomena involve rewards and incentives, they belong to the field of *cryptoeconomics*. Hence, they fit well into the *PRESTO* framework [18], which compares the cryptoeconomic properties of different protocols along five axes. These are (in reverse order): ***o**ptimality* (are distributed computing guarantees provided under ideal conditions?), ***st**ability* (is it a Nash equilibrium to follow the protocol?), ***e**fficiency* (what is the protocol's throughput relative to its use of resources?), ***r**obustness* (how does the protocol cope when assumptions turn out to be invalid?), and ***p**ersistence* (can the protocol recover from serious attacks?).

In this paper, we explore and quantify this five-dimensional trade-off for Casper the Friendly Finality Gadget (FFG), the novel PoS checkpointing protocol for Ethereum. It was specifically designed [8] with curtailing majority attacks (the solid lines in Fig. 1a) in mind. In the new protocol, users can deposit tokens to be inducted into a new agent class: *validators*. The core task of the validators is to periodically vote on checkpoints, with the original application being that PoW miners prioritise recent checkpoints over long chains in their fork-choice rule. Ethereum's current medium-term goal is to apply Casper FFG on a separate PoS chain [9] that will not only implicitly perform PoW chain checkpointing, but also facilitate the transition to a more scalable, sharded [25] design. Parts of the new protocol are still under development, which is why we focus on the now-defunct Casper contract [24] which is an implementation of Ethereum Im-

provement Proposal 1011 [47]. Although the EIP has been deprecated [26,45] in favour of the sharding-minded approach, the implemented protocol is fully functional and allows us to reason about its core reward mechanism. Although this paper is focussed on Ethereum, Casper FFG can be overlaid onto any other block proposal mechanism, and would thus confer its security properties onto it.

Our results. This paper makes the following contributions (optimality is not treated, as the underlying blockchain's properties are retained).

- A formal description of the reward mechanism in Casper FFG, which by itself is a contribution to the scientific debate around PoW and PoS.
- For stability, we show that it is a Nash equilibrium for the validators to follow the protocol, and that selfish mining [28] profitability is slightly reduced.
- For efficiency, we find that although the protocol can require large overhead, the new two-chain approach allows for optimisations to ameliorate this.
- For robustness, we show that censoring attackers suffer at least one third of the damage done to their victims in the short term, and show that attacks only start to pay off after unrealistically long time periods, i.e., decades.
- For persistence, we show that the time to recover from majority attacks using the protocol's minority forking mechanism can be bounded according to the protocol designers' preferences (e.g., three weeks).

Outline. This paper is structured as follows. We first provide a first concise, formal description of the Casper FFG protocol in Section 2. We will discuss the short- and long-term incentives of the protocol agents in Section 4. We will analyse the minority fork mechanism in Section 5. Section 6 concludes the paper.

Related Work. Casper FFG is, to the best of our knowledge, unique among blockchain consensus protocols as a PoS overlay that provides checkpointing. Among the other applications of PoS, some use pure PoS (e.g., NXT [44], Blackcoin [50], Tezos [32], Algorand [17,31], Snow White [4,20], and Ouroboros [37]), PoS and PoW in parallel (e.g., PPCoin/Peercoin [38]), or variations of PoS (e.g., Proof-of-Importance in NEM [43], Distributed PoS in EOS.IO [23], and Proof-of-Activity [2,3]). Tendermint [39] started as a pure PoS mechanism, but has since evolved to a general-purpose consensus engine.

Regarding attacks against PoS, we note that Casper FFG does not suffer from the nothing-at-stake problem because of slashing [16], and is not vulnerable to stake-bleeding attacks [30] because of the checkpointing mechanism. In [5], the cost of 51% attacks against both PoW and PoS platforms are considered. In [6], the vulnerability of blockchain platforms to 51% attacks is explored from an economic perspective.

2 The Casper FFG Protocol

2.1 Protocol Execution

The core functionality of the Casper protocol is provided by a set of validators who transmit `vote` messages of the form $\langle \mathsf{v}, \mathrm{t}, h(\mathrm{t}), h(\mathrm{s}), \mathcal{S} \rangle$, where the entries

Notation	Description
ν	the validator index
t	the hash of the 'target' checkpoint
h(t)	the height of the target checkpoint in the checkpoint tree
h(s)	the height of the source checkpoint in the checkpoint tree
$\mathcal{S}$	signature of $(\nu, \text{t}, h(\text{t}), h(\text{s}))$ from the validator's private key

Table 1: Summary of `vote` message contents.

are explained in Table 1. For ease of notation, we assume that the full set of validators who will at any time participate in the protocol can be ordered, and that validators can be identified by their place in the ordering. The set of *active* validators at any given time is dynamic — to join or leave the active validator set, validators can respectively send `deposit` and `withdraw` (and potentially `logout`) messages. Typically, a validator will need to wait a long period (e.g., a month) after depositing before being able to withdraw. How this is implemented is discussed in more detail in Section 2.3 and Appendix B.

A *checkpoint* is any block with block number $i \cdot l$, where $i \in \{0, 1, \ldots\}$. Block 0 (which is also a checkpoint) denotes the genesis block. Here, $l \in \mathbb{N}$ denotes the epoch length — an epoch is defined as the contiguous sequence of blocks between two checkpoints, including the first checkpoint but not the later one. We will assume $l = 50$ throughout this paper [47]. The *epoch of a block* is the index of the epoch containing that hash, e.g., the epoch of blocks 200 and 249 is 4. Because validators' deposits change from block to block, we denote by $D^*_{\nu,i}$ the deposit of validator $\nu \in \mathbb{N}$ right before block $i \cdot l$. We accept fractional values to denote blocks between checkpoints: e.g., $D^*_{\nu,1.5}$ is validator ν's deposit right before block $1.5l$, so if $l = 50$, before block 75.

At any epoch i, two sets of validators are relevant: the *current* active validator set V_i and the *previous* active validator set V'_i. Typically, $V'_i = V_{i-1}$, but not if checkpoints are not being finalised (more on that later). Changes to the active validator set will not affect the analyses conducted in this paper, so will not be treated here in more detail: for more background, see Section 3 of [16].

We will use "p of validators" for any fraction p (e.g., $\frac{2}{3}$) to refer to any set of validators $S \subset \mathbb{N}$ at epoch i such that,

$$\sum_{\nu \in S \cap V_i} D^*_{\nu,i} \geq p \sum_{\nu \in V_i} D^*_{\nu,i} \quad \text{and} \quad \sum_{\nu \in S \cap V'_i} D^*_{\nu,i} \geq p \sum_{\nu \in V'_i} D^*_{\nu,i}.$$

That is, S must make up *both* p of the current validator set *and* p of the previous validator set. A checkpoint t is *justified* if, for some s (which must be an ancestor of t), there exist $\frac{2}{3}$ votes of the form $\langle \nu, \text{t}, h(\text{t}), h(\text{s}), \mathcal{S} \rangle$, and $h(\text{s})$ is itself justified. A checkpoint s is *finalised* if it is justified and a direct child of s is also justified. The genesis block is considered to be both justified and finalised, and serves as the base case for the recursive definition.

Casper considers votes for the 'wrong' targets and target epochs as invalid. In particular, the only valid target epoch is the current epoch. The only valid target on a specific chain is the relevant checkpoint block on that chain. Depending on

the block proposal mechanism, it is possible for different chains to exist on the network such that a single vote is considered valid on one chain and not on the other. Regarding the source epoch, the Casper contract [24] allows votes from any source epoch, and will indeed justify a checkpoint if there are $\frac{2}{3}$ votes from any single justified source epoch. However, it will only *reward* a vote if it uses the 'expected' source epoch, which is the last justified epoch.

For a summary of the notation used in this paper, see Appendix A.

2.2 Fault Types

We seek to disincentivise the following two core types of faults: *safety* faults, i.e., two incompatible checkpoints getting finalised, and *liveness* faults, i.e., a checkpoint not getting finalised during some epoch. We discuss both below.

Safety faults: It can be shown that any safety fault can only be caused by at least $\frac{1}{3}$ of validators violating one of the two Casper Commandments ("slashing conditions") — i.e., publishing (as validator ν) two distinct votes

$$\langle \nu, \mathrm{t}_1, h(\mathrm{t}_1), h(\mathrm{s}_1), \mathcal{S}_1 \rangle \quad \text{and} \quad \langle \nu, \mathrm{t}_2, h(\mathrm{t}_2), h(\mathrm{s}_2), \mathcal{S}_2 \rangle, \quad \text{such that either}$$

I. $h(t_1) = h(t_2)$ (publishing two distinct votes for the same target height),
II. $h(s_1) < h(s_2) < h(t_2) < h(t_1)$ (voting within the span of its other votes).

For more detail, see [16] or [42]. Any validator is able to broadcast a `slash` message that includes two potentially offending vote messages. If the `slash` message is found to be valid, then the offending validator's deposit is shrunk or destroyed and the sender receives 4% of this deposit as a 'finder's fee'.

The exact amount of ETH that is burned when a validator is found to have violated a slashing condition has evolved throughout the development of the Casper contract. Earlier versions had *total* slashing, where a `slash` message would cause a validator's entry in the contract to be deleted. However, later versions use *partial* slashing, where the degree to which the validator's deposit is shrunk depends to the total amount slashed 'recently' across the protocol — the reasoning is to punish more harshly when the overturning of finalised checkpoints becomes a realistic threat. Regardless of whether total or partial slashing is used, the damage done to an attacker is large enough to not impact base incentives and therefore stability. However, the lower bound of $\frac{1}{3}$ of deposits being slashed in any safety fault does contribute to robustness. Since slashing has already been treated in [16], it will not be discussed further in this paper.

Liveness Faults: Penalising liveness faults is more difficult, as there are several faulty behaviours that can cause a validator ν to fail to produce a valid vote, as we discuss below.

1. A valid vote by ν for the target epoch either never makes it onto the blockchain, or only outside the target epoch. This could be because
   (a) ν never cast the vote, or was too late when doing so,
   (b) a majority of the other validators are censoring ν,
   (c) a significant portion of the nodes decided not to propagate ν's vote,

(d) the network latency was too severe.

2. A vote by ν does make it onto a block inside the target epoch, but it is otherwise invalid. This could be because
   (a) the signature is invalid,
   (b) the source epoch does not use the last justified epoch, or
   (c) the target hash is invalid.

In case 1a, the fault clearly lies with ν. However, in cases 1b and 1c the fault lies with (a coalition of) malicious validators or nodes respectively. In case 1d, the fault does not even necessarily lie with participants in the blockchain. In case 2a, this could have been due to a fault by ν, or by one or more malicious or malfunctioning nodes tampering with ν's message before propagating. In case 2b, this is either the result of a mistake by ν, a safety fault (in which a previously finalised epoch is overturned), or because ν had an incomplete view of the network. In case 2c, the blockchain must have been forked when ν voted — again, this could have been caused by the network, a mistake by ν, misbehaving nodes (who blocked ν's view of the network), misbehaving miners (who were withholding blocks from ν), or a misbehaving coalition of validators (who finalised a chain that was different from the one preferred by the network).

The above relates to a fundamental problem with attributing liveness faults known as *speaker/listener fault equivalence* — see also [8]. Hence, in a liveness fault, we cannot unambiguously determine who was at fault, and this creates a fundamental tension between *disincentivising harm* and *fairness* — between sufficiently penalising validators who are malicious and not excessively penalising validators who are not at fault. After all, if the penalties cause innocent validators to not feel comfortable participating, then this itself reduces security.

2.3 Appearance of Protocol Messages on the Blockchain

Since the protocol messages in Casper FFG — `vote`, `slash`, `deposit`, and `withdraw` — impact the fork-choice rule on the main chain, clients need to reach consensus about their existence and ordering. In the following, we will discuss two ways to achieve this.

On the PoW Main Chain: In the iteration of Casper FFG as a contract, all the relevant messages are stored on the main blockchain as contract transactions. The reward variables (see Section 2.4) are updated by the clients via the contract state. The only change to the clients would be the fork-choice rule: given the choice between two chains, they would prefer the one with the greatest height of the last finalised checkpoint, and only prefer the one with the largest block height/difficulty if the greatest checkpoints heights are the same.

On a Separate Chain: This design uses two different blockchains: the traditional PoW chain — here called the *main chain* — and a separate chain called the *beacon chain*.[1] The core protocol messages (`vote` and `slash`) are

[1] The beacon chain is named for its originally envisaged role in producing a random beacon [22].

moved to the latter. Rewards are processed internally by clients. A contract on the main chain is still created to handle `deposit` and `withdraw` messages, and to process exchanges from ETH to/from the reward variables on the beacon chain. The consensus and block proposal mechanism on the beacon chain is still under active development, but will involve PoS (with the same validator set as Casper FFG). See Appendix B for more detail.

2.4 Rewards and Penalties

In this section, we present a description of the reward scheme in Casper FFG as implemented in the contract [24]: all the formulas are mathematical representations of the code therein. In the scheme, voters are rewarded (and non-voters penalised) implicitly, via two variables that are updated every epoch:

- the *deposit scale factor*, denoted by S, and
- the *reward factor*, denoted by ρ.

In the first block of each epoch, S is decreased, but a validators who votes receives a reward ρ that (more than) compensates for the loss in terms of their *scaled* deposit. Since S and ρ vary from epoch to epoch, they can be described using sequences $\{S_i\}_{i=0,1,\ldots}$ and $\{\rho\}_{i=0,1,\ldots}$ defined as follows: S_{-1} and ρ_{-1} are the initial values (which are hard-coded into the protocol), and S_i and ρ_i are their values during blocks in $(il, (i+1)l]$. See Figure 2 for an illustration. We assume for simplicity that the deposits are denominated in ETH instead of wei.

The following protocol parameters are important (all are positive):

- the *base interest factor*, denoted by γ,
- the *base penalty factor*, denoted by β, and
- the *total deposit dependence factor*, denoted by p.

These variables influence the reward factor. In particular, at the start of each epoch, if the total deposit size is greater than 0, then the reward factor is set to

$$\rho_i = \frac{\gamma}{\left(\max\left(\sum_{\nu\in V_i} S_{i-1}D^*_{\nu,i}, \sum_{\nu\in V'_i} S_{i-1}D^*_{\nu,i}\right)\right)^p} + \beta\cdot(\mathrm{ESF}_i - 2). \quad (1)$$

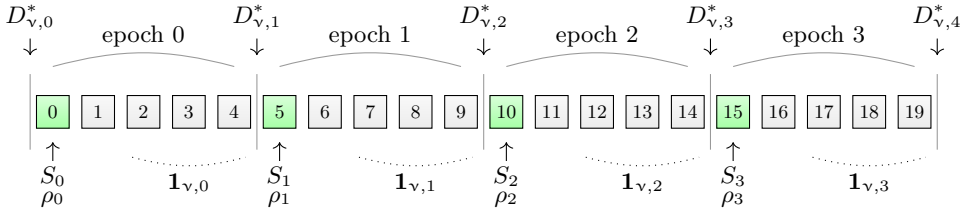


Fig. 2: Illustration of the points in the blockchain where the different variables are updated — here, $l = 5$.

Here, ESF_i denotes the *Epochs Since Finalisation* at epoch i, defined as i minus the height of the last finalised epoch. In the ideal situation where everyone always votes, the ESF always equals 2 (it requires two consecutive epochs to be justified for the first to be finalised), otherwise it is higher. So voters get a higher base reward if the ESF increases, i.e., if blocks fail to get finalised for several epochs in a row. (However, any gains are offset by the deposit scale factor.)

If a vote is 'correct' (voted for the expected block at expected target epoch, from expected target source) on a given chain, then the voter gets a reward equal (after rounding) to its deposit times the reward factor on this chain. Let

$$\mathbf{1}_{\nu,i} = \begin{cases} 1 \text{ if validator } v \text{ successfully voted during epoch } i, \\ 0 \text{ otherwise.} \end{cases}$$

The votes are typically sent when some fraction of the epoch has passed: in Pyethapp (one of the Ethereum clients that was used to test the Casper contract on Ethereum's test net) it is done after $\frac{1}{4}$ of the epoch [29]. For the base deposits at the start of an epoch, the following then holds:

$$D^*_{\nu,i} = (1 + \mathbf{1}_{\nu,i-1}\rho_{i-1})D^*_{\nu,i-1}. \tag{2}$$

Let m_i be defined as

$$m_i = \min\left(\frac{\sum_{\nu\in V_i}\mathbf{1}_{\nu,i}D^*_{\nu,i}}{\sum_{\nu\in V_i}D^*_{\nu,i}}, \frac{\sum_{\nu\in V'_i}\mathbf{1}_{\nu,i}D^*_{\nu,i}}{\sum_{\nu\in V'_i}D^*_{\nu,i}}\right)$$

Then the 'collective reward' C and scale factor S in epoch i are given as

$$C_i = \begin{cases} \frac{1}{2}m_i\rho_i \text{ if } \text{ESF}_i = 2, \\ 0 \quad \text{otherwise,} \end{cases} \qquad \text{and} \qquad S_i = \frac{1 + C_{i-1}}{1 + \rho_{i-1}}S_{i-1}. \tag{3}$$

We use ρ_{i-1} because the reward factor is changed after the deposit scale factor. Combining all this with (2), we obtain two expressions for the *scaled* deposit of validator ν at epoch i: right *before* checkpoint i, given by $D'_{\nu,i} \triangleq S_{i-1}D^*_{\nu,i}$, and right *after* checkpoint i, given by $D_{\nu,i} \triangleq S_iD^*_{\nu,i}$. For the latter, the following convenient relationship holds:

$$\begin{aligned} \frac{D_{\nu,i}}{D_{\nu,i-1}} &= \frac{S_i}{S_{i-1}}(1 + \mathbf{1}_{\nu,i-1}\rho_{i-1}) \\ &= (1 + C_{i-1})\frac{1 + \mathbf{1}_{\nu,i-1}\rho_{i-1}}{1 + \rho_{i-1}}. \end{aligned} \tag{4}$$

Note that $D_{\nu,i}$ increases monotonically in i for validators who always vote successfully. For more information about how to choose values for the parameters, see Appendix D — we treat $\gamma = 0.007$, $\beta = 0.0000002$, and $p = \frac{1}{2}$ as the benchmark setting for the figures in later sections.

2.5 Efficiency of Casper FFG

The protocol messages impact the efficiency of the protocol as they use the same client bandwidth and processing power as regular transactions, either implicitly or explicitly. The expected number of messages per epoch (especially `vote`) is proportional to the number of participating validators, which can be limited by imposing a minimum deposit size on the validators. With Casper FFG as a contract on the PoW chain, a minimum deposit of 1500 ETH was intended [47], leading to slightly over 900 expected validators [7]. With on average around 125 transactions per block (see `https://etherscan.io`) and 50 blocks per epoch, that would be 900 votes for every 6250 transactions, so around 12% of all transactions would be votes. Experiments suggested that if votes were allowed to consume as much 'gas' (i.e., computational cost) as all other transactions combined during the last three quarters of an epoch (when the votes were expected to come in), then only between 400 and 592 validators could be supported [46].

By contrast, the dual-chain approach allows for the utilisation of message processing optimisations (e.g., bit masks and signature aggregation [10,26]) as the messages are processed natively by the clients. The gains are expected to be so large that is considered possible to lower the minimum deposit size to 32ETH [26] (and possibly make it fixed), with a theoretical upper bound of $2^{22} \approx 4.2$ million validators [14]. In this case, even the `deposit` and `withdraw` messages may impose a considerable load, although again optimisations are available [10].

3 Stability and Robustness of Casper FFG

3.1 Stability

In this section we will consider the system's *stability*, i.e., whether it is optimal for all types of agents in the system to follow the protocol.

Miners: Currently, the incentives for miners are similar to those in Bitcoin, for which it has been shown in [28,48,36] to be a Nash equilibrium to follow the protocol for anyone who controls less than roughly $\frac{1}{3}$ of the mining power. The checkpointing mechanism has an impact on the profitability of selfish mining because it imposes limits on how many blocks can be reverted and when. Although the epoch length (50 blocks) is likely to be too large for the impact to be considerable, the stability in terms of miner incentives is slightly improved.

Miners also influence whether and when some Casper FFG protocol messages appear on the PoW chain (for `vote` messages, the 'when' is very important because delaying them beyond the end of the epoch renders them invalid). In the setting where Casper FFG exists only as a contract on the PoW chain (see Section 2.3), miners affect all of the protocol messages, whereas only `deposit` and `withdraw` messages are affected in the dual-chain setting. Hence, it is important for stability that miners do not have an incentive to obstruct these messages. They can be rewarded for the protocol messages via transaction fees — in the single-chain setting they also receive a fee for including vote messages equal to $\frac{1}{8}\rho$ (see line 298 of [24]). On the other hand, the impact that they have by

	loss to ν	loss to voters	loss to non-voters
justification regardless of ν	$\frac{\rho(1+\frac{1}{2}(m\rho+\alpha))}{1+\rho}$	$\frac{1}{2}\alpha\rho$	$\frac{\frac{1}{2}\alpha\rho}{1+\rho}$
no justification regardless of ν	$\frac{\rho}{1+\rho}$	0	0
ν is swing voter	$\frac{\rho(1+\frac{1}{2}m(1+\rho))}{1+\rho}$	$\frac{1}{2}m\rho$	$\frac{\frac{1}{2}m\rho}{1+\rho}$

Table 2: Losses to all groups of validators if ν abstains from voting.

obstructing messages is limited. If l is large enough, even a small minority of honest miners can get votes onto the blockchain in time with high probability. For refusal to publish `slash`, `deposit` or `withdraw` messages to have any effect, they must be kept off the blockchain indefinitely, which requires 51% censorship.

Should a 51% censorship attack by miners occur, then this is a question of persistence rather than stability, and as such will be discussed in Section 5.

Validators: The validators are the new class of agents introduced by Casper FFG. Their most important core action is to vote, so we will consider the impact on their deposits of producing a valid vote or not. In the following, we will consider the case of a validator ν who controls a deposit share $\alpha \in (0, \frac{2}{3})$. The combined fraction of ν and the validators who vote during epoch i is given by m_i (the $1-m_i$ who do not vote could be doing so because they have unwillingly gone offline). We consider a single epoch, so we write ρ instead of ρ_i for now and m for m_i. The impact of ν not voting is summarised in Table 2 — the impact depends on whether the epoch is justified (i.e., more than $\frac{2}{3}$ vote). Clearly, if ν does not vote, all validators lose a bit, but ν loses by far the most. Hence, ν is incentivised to follow the protocol. For more information on how the entries in Table 2 are derived, see Appendix C.

3.2 Short-Term Robustness

It can be seen in Table 2 that if any validator does not vote, all validators lose. Hence, it does not pay in the short term for a validator to cause another validator to not vote, e.g., by censoring. However, this does not rule out that this behaviour will ever happen, as those validators could be driven by external factors. This involves *robustness*: how does the system cope when some of the validators do not behave as expected given the model's payoffs? In the following, we aim to provide upper bounds on the damage that a validator ν can do relative to ν's own losses. We consider two measures: the ratio between the *absolute* losses of the other validators to ν's, called the Griefing Factor (GF) g, and the *relative* losses, called the Proportional Loss Ratio (PLR) r [15].

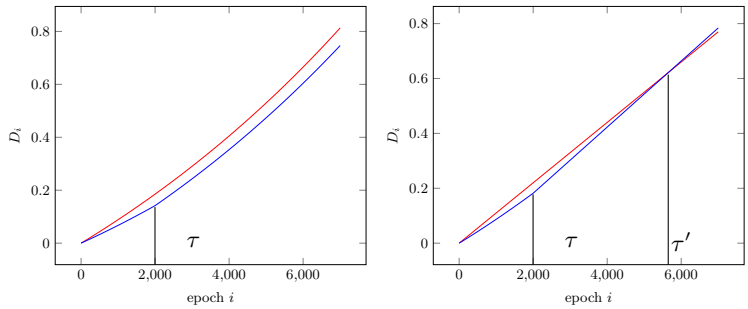
In general, it holds that if the attacker controls α of deposits, then $g = \frac{1-\alpha}{\alpha}r$ [15]. Table 3 displays the PLRs, obtained by dividing the relevant entries in Table 2 by each other and applying the equality above. All PLRs are bounded from above by $\frac{1}{2}$, and are much lower for small α. More concretely, if $\rho \approx 0$,

	voters PLR	non-voters PLR	voters GF	non-voters GF
always finalisation	$\frac{\frac{1}{2}\alpha(1+\rho)}{(1+\frac{1}{2}(m\rho+\alpha))}$	$\frac{\frac{1}{2}\alpha}{(1+\frac{1}{2}(m\rho+\alpha))}$	$\frac{\frac{1}{2}(1-\alpha)(1+\rho)}{(1+\frac{1}{2}(m\rho+\alpha))}$	$\frac{\frac{1}{2}(1-\alpha)}{(1+\frac{1}{2}(m\rho+\alpha))}$
never finalisation	0	0	0	0
ν is swing voter	$\frac{\frac{1}{2}m(1+\rho)}{(1+\frac{1}{2}m(1+\rho))}$	$\frac{\frac{1}{2}m}{(1+\frac{1}{2}m(1+\rho))}$	$\frac{\frac{1}{2}m(1-\alpha)(1+\rho)}{\alpha(1+\frac{1}{2}m(1+\rho))}$	$\frac{\frac{1}{2}m(1-\alpha)}{\alpha(1+\frac{1}{2}m(1+\rho))}$

Table 3: PLRs and GFs if ν abstains from voting.

then the PLRs are $\frac{1}{2}\alpha/(1+\frac{1}{2}\alpha)$ and $\frac{1}{2}m/(1+\frac{1}{2}m)$ respectively, so the PLR is between 0 (at $\alpha \approx 0$) and $\frac{1}{3}$ (at $\alpha > \frac{1}{3}$ and $m = 1$) whereas the GF is between 1 (at $\alpha \approx 0$) and $\frac{1}{3}$ (at $\alpha > \frac{1}{3}$ and $m = 1$). This means that a validator will always incur a relative cost that is higher than the damage done to other validators by abstaining.

3.3 Long-Term Stability/Robustness



(a) $p = 0$, $\gamma = 0.00017$, $D_0 = 100\,000$, $h = 0.25$, $\tau = 2\,000$

(b) $p = 1$, $\gamma = 10$, $D_0 = 100\,000$, $h = 0.25$, $\tau = 2\,000$

Fig. 3: Illustration of the long-term evolution of D for different values of p, when the attacker blocks other validators from voting until epoch $\tau = 2\,000$.

Although choosing not to vote is a form of misbehaviour, so is blocking other validators' votes. We will now consider the possibility that malevolent validators who control a fraction $\alpha \in (0, 1)$ of the total deposits have the power to *prevent* another group of validators from voting, who control a stake of h, for a period of τ epochs. There are two main ways in which they can achieve this. First, if a single powerful validator controls a large percentage of the deposits, then it may be possible to force this validator off the network for some time, e.g., through a DoS [35,49] or network partitioning (e.g., eclipse [33] or BGP hijacking [1])

attack. Second, if the coalition controls a large enough fraction of the stake, then it may be possible to perform a censorship attack against other validators by refusing to build on top of or justify (children of) their blocks.

In the short term, preventing a group of validators from voting will always hurt the profits of the other validators. This is obvious from the tables in the previous section. Unfortunately, the PLRs for censoring are inversely proportional to the PLRs for abstaining. Still, attackers hurt their own profits, which is not the case in, e.g., Bitcoin: because Bitcoin's PoW difficulty is regularly updated, censoring a minority is not just costless but profitable because the attackers' share of the (largely constant) mining rewards is increased.[2]

However, in the (very) long term, depending on the value of p, it may eventually become profitable for validators in Casper FFG to increase their share of the total deposit by hurting other validators' deposits. The reason is that if $p > 0$, then the interest paid out to validators increases as the total deposit decreases. In fact, it can be shown that for $p > 0$ there will always be some τ' in the future such that from that point on, it will have been more profitable to have conducted the attack than to not have — see Figure 3 for an illustration. If epochs remain being finalised, i.e., if $h < \frac{1}{3}$ and the proportion of otherwise non-voting validators is negligible, then $\tau' \approx (\sum_{\mathbf{v}} D_{\mathbf{v},0})^p/(p\gamma)$, where epoch 0 denotes the start of the censorship attack (see Appendix E for more detail). This approximation for τ' is remarkably insensitive to h and τ, see Figure 7 for an illustration. With the benchmark assumptions of $p = \frac{1}{2}$, $\gamma = 0.007$, and $\sum_{\mathbf{v}} D_{\mathbf{v},0} = 10\,000\,000$, $\tau' \approx 903\,500$, so roughly 20 years. Hence, it is impossible for censoring validators or those performing DoS attacks to make a profit within a realistic time frame. In reality, this type of attack may be even less profitable because the gains in terms of a higher interest rate may well be offset by new validators joining the protocol.

4 Persistence of Casper FFG

4.1 Types of Severe Attacks

In this section, we investigate the *persistence*, the ability of a protocol to recover from debilitating attacks. We begin with a discussion of four types of serious attacks on the protocol.

I. Attacks on PoW Chain by a Miner Majority: The most serious kind of attack on Ethereum's PoW chain is a majority attack by the miners, in particular majority censorship and n-confirmation double-spends for high n. As a checkpointing mechanism, Casper FFG protects against the latter type of attack by imposing a high cost on the reversion of blocks beyond finalised checkpoints.

[2] Although validators hurt their short-term profits by censoring, if the griefing factor is above 1, what they earn as a proportion of the total amount of money paid out by Casper increases. This could be an issue if the amount of tokens given to validators as interest is very large compared to the amount of tokens issued as block rewards. (After all, fewer tokens ⇒ higher value per token.)

Additionally, it provides a recovery mechanism against censorship if a coalition that controls at least $\frac{2}{3}$ of the total stake is willing to intervene. Because the fork-choice rule for honest miners and clients prioritises the highest finalised block on the beacon chain, this gives the validators' supermajority coalition the power to overrule the normal block proposal mechanism. To build an alternative main chain, the validators must ignore the standard fork-choice rule while the attack is ongoing, and only build on top of blocks that are mined by whitelisted miners. Although this is not an ideal situation to continue indefinitely, the fact that a recovery mechanism exists in itself disincentivises attackers — after all, the period during which they can maintain the attack is limited, and they lose their mining rewards while the system is being controlled by the validators.

II. Attack on PoW Chain by Validator Supermajority: Although the power of validators to overrule miners can be used for good, the supermajority coalition might also choose to abuse it. In principle, there is no agent class in the protocol that can overrule the validators (and if there were, the same question of 'who watches the watchmen' would apply to them). However, the minority can recover from this attack using a mechanism called the *minority fork*. It requires that honest nodes adapt their fork-choice rule by blacklisting a block's epoch/hash combination that was justified by the coalition (depending which approach in Section 2.3 is chosen, this block can either be on the PoW chain or the beacon chain). A new chain will then naturally emerge on top of the previous block. All validators who have voted using the blacklisted block or one of its descendants as a source will be unable to vote on the minority chain without violating a slashing condition, unless they wait until epochs resume being justified. Eventually the attacking coalition's share of the stake will have dropped from above $\frac{2}{3}$ to below $\frac{1}{3}$, at which point epochs can again be justified. In any case, the attackers suffer heavy losses.

When the blockchain has been forked, it seems likely that nodes (and the market for ETH) will prefer the chain on which the attack has been neutralised. To this end, it is necessary for those who initiate the fork that the misbehaviour of the coalition (or their lack of action) is clearly visible to outside parties, and to make sure that other validators are not unnecessarily penalised. This will be discussed in Section 5.2.

III. Attack by Miner Majority + Negligent Validators: Recovery against Attack I requires a supermajority of validators to be willing to manually overrule the miners. If a supermajority coalition cannot be formed because a significant number of validators is unwilling to intervene, then the only course of action left for the activist validators is to become a supermajority on their own chain after a fork, similar to the recovery mechanism against Attack II.

IV. Attack on Beacon Chain by Validator Majority: In the dual-chain setting, an additional type of majority attack by validators is a traditional 51% attack on the beacon chain. In this case the aggrieved party are the minority of validators whose beacon blocks (and perhaps `vote` or `slash` messages) fail to make it onto the canonical beacon chain. Because the validator set on the beacon

chain is the same as for FFG, the misbehaving validators cannot be overruled. As for Attack II and III, the minority fork is the natural recovery mechanism.

4.2 Initiating a Minority Fork

In the event of a minority fork, all validators, miners, and other clients need to decide whether to stay with the chain starting with the blacklisted hash— the *majority chain* — or follow the alternative chain starting after the blacklisted hash's predecessor — the *minority chain*. As mentioned earlier, when the network is forked, anyone who has voted using a block on the majority chain will lose a large part of their deposits on the minority chain — at least three quarters if they need to wait for justification to resume. Hence, it is crucial that validators know when a hash is about to be blacklisted so they can stop voting in advance. One approach is to build into the validator client a rule to stop voting once there is enough evidence of misbehaviour, which would make coordination automatic. Validators can then manually resume voting if, e.g., high latency or bad luck was mistaken for misbehaviour. For miners and clients, the conditions on the timing of their switch are less stringent — however, they still need to be able to make a well-informed choice about which chain to follow.

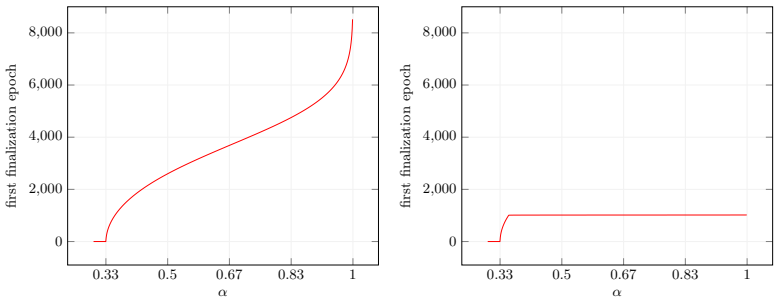
For victims of censorship, it is clear that they lose rewards — miners/validators lose block rewards when their blocks are overruled on the PoW/beacon chain, and validators lose FFG rewards when their votes do no make it onto the chain in time, or at all. Much higher losses than expected across an extended time period can motivate validators to stop voting and miners to be on alert for a fork. In general, unambiguous detection of censorship is an interesting topic for future research — more discussion can be found in Appendix F.

4.3 Typical Evolution of a Fork

After the fork has been initiated, epochs can initially not be finalised on the minority chain, so the penalty term in (1) starts to have an effect. Voting will yield increasingly large rewards, as the reward factor increases proportionally to the ESF (as discussed in Section 2.4), although the gains are undone by the change in the deposit scale factor at the start of every epoch. This is displayed in Figures 8 and 9 in Appendix G for the short and long term respectively.

If e is the first epoch to be finalised, then at the start of epoch $e+2$ the collective reward as defined in (**??**) changes abruptly from 0 to $\frac{1}{3}\rho_{e+1}$, which means that if a validator voted during all epochs up until then, then she will have gained approximately $\frac{2}{3}\rho_{e+1}$ over this period (see the rightmost graph in Figure 9). This is considerably less than what would have been earned under normal circumstances, but this is the price to pay for recovering the protocol.[3] Also note that even short downtimes will have relatively large penalties for fork participants in the epochs shortly before the first finalisation.

[3] For $p > 0$, the same long-term effects (higher effective interest due to smaller total deposit) as in Section 4.3 may come into play.



(a) Benchmark setting. (b) Added exponential penalty term.

Fig. 4: Epoch of the first finalisation on a minority chain where those who remain voting control a fraction $1 - \alpha$ of the total stake. This is done for both the benchmark setting of Section 2.4, and one where an exponential penalty term is added, as discussed in Appendix G. To create this graph, $\sum D = 10^6$ was used.

In Figure 4a, we have depicted the number of epochs needed for the protocol to recover. Naturally, the higher the stake of the validators that cannot vote on the minority chain, the longer it takes before their power has diminished sufficiently. If $\alpha = 0.67$, then it will take about 3600 epochs (roughly 58 days) for Casper to recover. If we modify the benchmark slightly to include an exponential penalty term after a fixed epoch threshold τ^*, then we can guarantee that the protocol recovers shortly after it is crossed regardless of how many participate in the fork. This is displayed in Figure 4b, where a threshold of 1000 epochs was used. In any case, the time until recovery is bounded regardless of the strength of the censoring majority, in both the benchmark and modified settings.

5 Conclusions

We have provided a formal description of the reward and incentive structure in Casper FFG, and analysed its properties using the PRESTO framework. We found that the PoS overlay mostly adds to the robustness and persistence of the framework. For robustness, we established bounds on the damage that misbehaving participants can do relative to their own losses. For persistence, we found that the maximum time before recovery from a majority attack can be set by the protocol. All of this comes at an implementation-specific cost to efficiency.

With the emergence of a wide range of blockchain consensus protocols comes the need to make their security trade-offs explicit. As we saw in this paper, the PRESTO framework allows us to consider not just well-studied properties such as stability and efficiency, but also lesser-know yet vital properties such as the ability to absorb or recover from serious attacks. More research is needed to make direct comparisons between protocols on PRESTO's five axes, such as in

Figure 1b, possible. Hence, future work will focus on analysing a greater variety of protocols using PRESTO.

Acknowledgements

The authors would like to thank Pieter Hartel, Stefanos Leonardos, and Chih-Cheng Liang for their helpful comments on a draft version of this paper. This work was supported in part by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2016NCR-NCR002-028) and administered by the National Cybersecurity R&D Directorate.

References

1. M. Apostolaki, A. Zohar, and L. Vanbever. Hijacking Bitcoin: Routing attacks on cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 375–392. IEEE, 2017.
2. I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In *International Conference on Financial Cryptography and Data Security*, pages 142–157. Springer, 2016.
3. I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending Bitcoin's proof of work via proof of stake [extended abstract]. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.
4. I. Bentov, R. Pass, and E. Shi. Snow White: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919, 2016.
5. J. Bonneau. Hostile blockchain takeovers (short paper). In *Proceedings of the 5th Workshop on Bitcoin and Blockchain Research*, 2018.
6. E. Budish. The economic limits of Bitcoin and the blockchain. Technical report, National Bureau of Economic Research, 2018.
7. V. Buterin. Parametrizing casper: the decentralization/finality time/overhead tradeoff, 2017. [Online]. Available: `https://medium.com/@VitalikButerin/parametrizing-casper-the-decentralization-finality-time-overhead-tradeoff-3f2011672735`. [Accessed: 3-9-2018].
8. V. Buterin. The triangle of harm, 2017. [Online]. Available: `https://vitalik.ca/general/2017/07/16/triangle_of_harm.html`. [Accessed: 3-9-2018].
9. V. Buterin. Beacon chain casper ffg rpj mini-spec, 2018. [Online]. Available: `https://ethresear.ch/t/beacon-chain-casper-ffg-rpj-mini-spec/2760`. [Accessed: 14-9-2018].
10. V. Buterin. Casper FFG and light clients, 2018. [Online]. Available: `https://ethresear.ch/t/casper-ffg-and-light-clients/2957`. [Accessed: 3-9-2018].
11. V. Buterin. CBC-Casper in the face of the new PoS+Sharding plans on Ethereum [comment], 2018. [Online]. Available: `https://ethresear.ch/t/cbc-casper-in-the-face-of-the-new-pos-sharding-plans-on-ethereum/2444/2`. [Accessed: 3-9-2018].
12. V. Buterin. Censorship detectors via 99% fault tolerant consensus, 2018. [Online]. Available: `https://ethresear.ch/t/censorship-detectors-via-99-fault-tolerant-consensus/2878`. [Accessed: 4-9-2018].
13. V. Buterin. Censorship rejection through "suspicion scores", 2018. [Online]. Available: `https://ethresear.ch/t/censorship-rejection-through-suspicion-scores/305`. [Accessed: 4-9-2018].
14. V. Buterin. Convenience link to Casper+Sharding chain v2.1 spec, 2018. [Online]. Available: `https://ethresear.ch/t/convenience-link-to-casper-sharding-chain-v2-1-spec/2332`. [Accessed: 4-9-2018].
15. V. Buterin. Discouragement attacks, 2018. [Online]. Available: `https://github.com/ethereum/research/blob/master/papers/other_casper/discouragement.pdf`. [Accessed: 3-9-2018].
16. V. Buterin and V. Griffith. Casper the Friendly Finality Gadget. *arXiv:1710.09437*, 2017.
17. J. Chen and S. Micali. Algorand. *arXiv:1607.01341*, 2016.
18. V. Chia, P. Hartel, Q. Hum, S. Ma, G. Piliouras, D. Reijsbergen, M. van Staalduinen, and P. Szalachowski. Rethinking blockchain security: Position paper. In *Proceedings of IEEE Blockchain 2018*, 2018.

19. J. Choi. Casper: The fixed income approach, 2017. [Online]. Available: https://ethresear.ch/t/casper-the-fixed-income-approach/218. [Accessed: 3-9-2018].
20. P. Daian, R. Pass, and E. Shi. Snow White: Robustly reconfigurable consensus and applications to provably secure proofs of stake. *Cryptology ePrint Archive*, 2016. http://eprint.iacr.org/2016/919.
21. Digiconomist. Bitcoin energy consumption index. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption. [Accessed: 3-9-2018].
22. J. Drake. Which BLS curve/DKG algorithm is going to be used for Casper? [comment], 2018. [Online]. Available: https://ethresear.ch/t/which-bls-curve-dkg-algorithm-is-going-to-be-used-for-casper/2273/7. [Accessed: 3-9-2018].
23. EOS.IO Community. EOS.IO technical white paper v2, 2017. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.
24. Ethereum. The Casper contract (simple_casper.v.py). [Online]. Available: https://github.com/ethereum/casper/blob/master/casper/contracts/simple_casper.v.py. [Accessed: 3-9-2018].
25. Ethereum. Sharding roadmap. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-roadmap. [Accessed: 14-9-2018].
26. Ethereum. Ethereum core devs meeting 40 notes, 2018. [Online]. Available: https://github.com/ethereum/pm/blob/master/All%20Core%20Devs%20Meetings/Meeting%2040.md. [Accessed: 3-9-2018].
27. Ethereum Wiki. Proof of stake FAQs - What would the equivalent of a 51% attack against Casper look like? [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs#what-would-the-equivalent-of-a-51-attack-against-casper-look-like. [Accessed: 3-9-2018].
28. I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.
29. K. Floersch, H. Hees, and Others. The pyethapp client's validator_service.py, 2017. [Online]. Available: https://github.com/karlfloersch/pyethapp/blob/dev_env/pyethapp/validator_service.py#L137. [Accessed: 11-9-2018].
30. P. Gazi, A. Kiayias, and A. Russell. Stake-bleeding attacks on proof-of-stake blockchains. *Cryptology ePrint Archive*, 2018. https://eprint.iacr.org/2018/248.
31. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.
32. L. Goodman. Tezos – a self-amending crypto-ledger: White paper, 2014. https://cdn.relayto.com/media/files/wMfrXRbQogudZech3tBA_white_paper.pdf.
33. E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *USENIX Security Symposium*, pages 129–144, 2015.
34. A. Hertig. Blockchain's once-feared 51% attack is now becoming regular, 2018. [Online]. Available: https://www.coindesk.com/blockchains-feared-51-attack-now-becoming-regular. [Accessed: 3-9-2018].
35. B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In *International Conference on Financial Cryptography and Data Security*, pages 72–86. Springer, 2014.
36. A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 365–382. ACM, 2016.

37. A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

38. S. King and S. Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake, 2012. `https://pdfs.semanticscholar.org/0db3/8d32069f3341d34c35085dc009a85ba13c13.pdf`.

39. J. Kwon. Tendermint: Consensus without mining, 2014.

40. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

41. A. Miller. Feather-forks: enforcing a blacklist with sub-50% hash power, 2013. [Online]. Available: `https://bitcointalk.org/index.php?topic=312668.0`. [Accessed: 3-9-2018].

42. O. Moindrot and C. Bournhonesque. Proof of stake made simple with Casper, 2017.

43. NEM Community. NEM technical reference, 2015. `https://nem.io/NEM_techRef.pdf`.

44. NXT Community. Nxt whitepaper, 2013. `https://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf`.

45. D. Ryan. Casper ♡ sharding. [Online]. Available: `https://medium.com/@djrtwo/casper-%EF%B8%8F-sharding-28a90077f121`. [Accessed: 3-9-2018].

46. D. Ryan. Gas cost profiling, optimization, and management, 2018. [Online]. Available: `https://github.com/ethereum/casper/issues/166`. [Accessed: 4-9-2018].

47. D. Ryan and C.-C. Liang. Ethereum improvement proposal 1011. [Online]. Available: `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1011.md`. [Accessed: 3-9-2018].

48. A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

49. M. Vasek, M. Thornton, and T. Moore. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In *International Conference on Financial Cryptography and Data Security*, pages 57–71. Springer, 2014.

50. P. Vasin. Blackcoin's proof-of-stake protocol v2, 2014. `https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf`.

Appendix

A List of Symbols

l	length (in terms of blocks) of an epoch
V_i	current active validator set at epoch i
V_i'	previous active validator set at epoch i
$D^*_{\nu,i}$	deposit of validator $\nu \in \mathbb{N}$ right before block $i \cdot l$
S_i	deposit scale factor right after checkpoint i
ρ_i	reward factor right after checkpoint i
$D'_{\nu,i}$	scaled deposit of validator ν right *before* checkpoint i (i.e., $D'_{\nu,i} = S_{i-1}D^*_{\nu,i}$)
$D_{\nu,i}$	scaled deposit of validator ν right *after* checkpoint i (i.e., $D_{\nu,i} = S_i D^*_{\nu,i}$)
γ	base interest factor (must be positive)
β	base penalty factor (must be positive)
p	total deposit dependence factor (must be positive, typically in $[0, 1]$)

Table 4: List of symbols.

B The Dual-Chain Approach

Although the dual-chain version of Ethereum is still under active development, we include a short discussion of the latest developments to this section for reference. First, to make the comparison more concrete, we have included a graphical representation of the implementation of Casper FFG as a contract on the PoW chain in Figure 5. Transactions would appear alongside 'regular' transactions, subject to certain requirements discussed in [47] (e.g., `vote` messages would appear after the regular transactions, and would have their own gas limit parallel to the one for the regular transactions).

Due to Ethereum's long-term goal of switching from a single blockchain to a sharded blockchain, a design that would facilitate this transition was chosen instead. As discussed in Section 2.3, this design uses two chains, the PoW chain

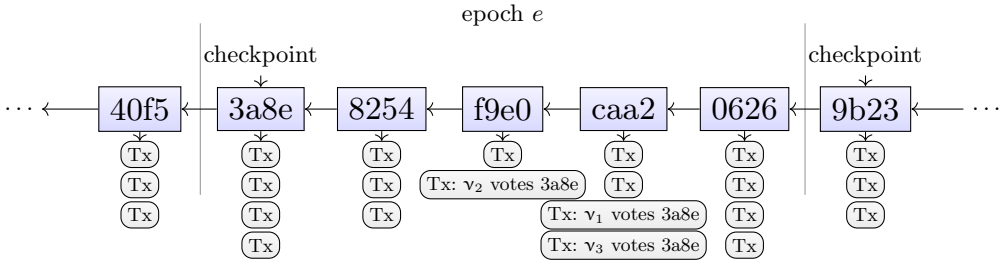


Fig. 5: Original set-up of Casper FFG: protocol messages are transactions to the Casper contract that appear on the main chain. In the above figure, the epoch length is 5, and three validators — ν_1, ν_2, and ν_3 — broadcast vote messages. Assuming that they control at least $\frac{2}{3}$ of the stake, epoch e is justified after the 4th block of e, and if epoch $e-1$ was also justified, then $e-1$ is finalised.

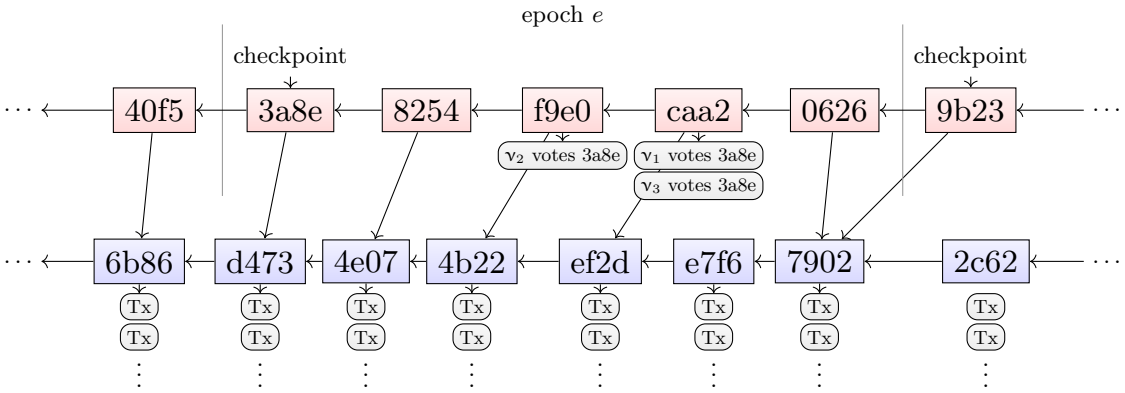


Fig. 6: Current design of Casper FFG: protocol messages are kept on a separate blockchain called the *beacon chain*. Each block on the beacon chain links to a block on the PoW main chain, although this mapping is not necessarily one-to-one. As checkpoints on the beacon chain are finalised, this indirectly imposes finalisation onto blocks on the main chain.

and a separate chain called the *beacon chain*. This is displayed in Figure 6. The core protocol messages (i.e., `vote` and `slash`) will be moved onto the beacon chain, whereas only entries to the validator set are handled on the main chain. Because ETH tokens only exist on the PoW chain, this does complicate the exchanging of tokens to/from validator accounts. In the proposed set-up [26], a contract is created on the main chain that allows potential validators to deposit 32 ETH, which is then burned, creating a log. This log is then used by the clients to determine whether a validator is in the validator set. It is not yet possible in this setting for clients to withdraw from the validator set — this will happen after a later hard fork. The evolution of reward-related variables (to be discussed in Section 2.4) is now processed and stored by the client software.

Each block on the beacon chain contains a reference in its header to a single block on the main chain — thus, by voting for and finalising blocks on the beacon chain, the validators are implicitly finalising blocks on the main chain. The block proposal mechanism on the shard chain is a naive chain-based PoS mechanism that is the subject of ongoing research by the Ethereum community. It will eventually be integrated with Casper CBC (Correct-by-Construction) [11].

The chain will most likely be vulnerable to traditional 51% attacks or censorship: if a majority of validators decide to pretend that they cannot receive messages from the others, then this chain will grow faster than any other chain, and this gives the majority coalition the power to censor vote and slash messages on the beacon chain. The recovery mechanism against this type of attack is the minority fork as described in Section 5. The vulnerability of the chain to other types of attacks such as feather-forking [41] remains to be seen.

Because most protocol messages have been moved off the PoW chain, miners do not have much control over the protocol messages, as discussed in Section 4.1. If a majority coalition among the miners on the PoW chain manages to create a fork going back to before the deposit messages, then they can censor some of the deposit messages and essentially hand-pick the validator set, leading to the existence of two chains with different chain difficulties yet the same checkpoint height, albeit using different validator sets. In this case, the clients will need to pick manually which chain they prefer, possibly by blacklisting a hash in the chain created by the misbehaving miners.

In an envisioned future design stage of Ethereum, the PoW chain will be deprecated in favour of a collection of *shard chains*. The beacon chain will then receive an additional load beyond Casper FFG messages, namely *'attestations'* [14] which serve to link the beacon chain to the shard chains, and implicitly the shard chains to each other. Again, this is very much an area of active research.

C Stability

In the following we discuss how to derive the entries of Table 2 in Section 4.2. To recap: we consider a validator ν, who controls (weighted) stake fraction α, and stops voting for τ epochs — we are interested at their profit after $T > \tau$ epochs. Without loss of generality, we assume that epoch 0 is the first epoch in which ν does not vote. In contrast to Section 4.3 we consider the short term rather than the long term — i.e., we assume that τ and T are small (rule of thumb: less than 10). This allows for a different set of simplifying assumptions. In particular, we assume that changes in the reward factor are small, so we assume that there is a single ρ such that $\rho \approx \rho_0 \approx \ldots \approx \rho_{T-1}$. We also assume that the fraction that does not vote regardless of ν is roughly constant and given by $1 - \mu$.

To measure the effects on the profits/losses of validators, we consider the relative increase in their scaled deposits after T epochs. We will consider the relative deposits right *after* the rescale, i.e., D_T/D_0. From (4) we know that if ν votes, then

$$D_{i+1}/D_i = 1 + C_i, \tag{5}$$

whereas if ν does not vote, then

$$D_{i+1}/D_i = \frac{1 + C_i}{1 + \rho}. \tag{6}$$

Here, C_i is also affected by whether ν voted in the previous epoch(s) or not. In particular, if $\mu - \alpha \geq \frac{2}{3}$, i.e., epochs are justified regardless of ν, then it follows from the definition of C_i in (**??**) that C_i equals $\frac{1}{2}\mu\rho$ if ν voted in epoch i and $\frac{1}{2}(\mu - \alpha)\rho$ if not. If $\mu < \frac{2}{3}$, i.e., epochs are not justified regardless of ν, then $C_i = 0$ regardless of ν's vote in epoch i. If $\mu \geq \frac{2}{3} > \mu - \alpha$, i.e., ν is a swing voter with respect to justification, then C_i equals 0 if ν did not vote during epoch i, but *also* if ν did not vote during epoch $i - 1$, because this would mean that the epoch can be justified but not finalised (as this requires two consecutive epochs to be justified).

scenario	validator type	relative deposit increase
always justification	ν	$\left(\frac{1+\frac{1}{2}(\mu-\alpha)\rho}{1+\rho}\right)^{\tau} \cdot \left(1+\frac{1}{2}\mu\rho\right)^{T-\tau}$
	voters	$\left(1+\frac{1}{2}(\mu-\alpha)\rho\right)^{\tau} \cdot \left(1+\frac{1}{2}\mu\rho\right)^{T-\tau}$
	non-voters	$\left(\frac{1+\frac{1}{2}(\mu-\alpha)\rho}{1+\rho}\right)^{\tau} \cdot \left(\frac{1+\frac{1}{2}\mu\rho}{1+\rho}\right)^{T-\tau}$
never justification	ν	$\left(\frac{1}{1+\rho}\right)^{\tau}$
	voters	1
	non-voters	$\left(\frac{1}{1+\rho}\right)^{T}$
ν is swing voter	ν	$\left(\frac{1}{1+\rho}\right)^{\tau} \cdot \left(1+\frac{1}{2}\mu\rho\right)^{T-\tau-1}$
	voters	$\left(1+\frac{1}{2}\mu\rho\right)^{T-\tau-1}$
	non-voters	$\left(\frac{1}{1+\rho}\right)^{\tau+1} \cdot \left(\frac{1+\frac{1}{2}\mu\rho}{1+\rho}\right)^{T-\tau-1}$

Table 5: Losses to the different validator groups if ν abstains from voting. In the rows corresponding to 'always justification' and 'never justification', the results hold for all $\tau \geq 0$ and $T \geq \tau$. For the 'ν is swing voter' rows, it must holds that $\tau \geq 1$ and $T \geq \tau + 1$.

Table 5 presents a summary of the relative deposit changes for the different scenarios and validator types. For the first entry, ν incurs the no-voting penalty of (6) τ times, and receives the voting reward of (5) $T - \tau$ times. For the second entry, the voters receive the reward of (5) T times, but τ times this is with a collective reward of $\frac{1}{2}(\mu - \alpha)\rho$ and $T - \tau$ times with a collective reward of $\frac{1}{2}\mu\rho$. Similarly, for the third entry, the non-voters receive the penalty of (6) but with different collective rewards. The other entries in the table are obtained in a similar manner, but such that the collective reward is 0 either always, or in the epochs when ν did not vote and the first epoch afterwards. Note that in the 'always justification' rows, we also assume that epoch -1 was justified.

Given Table 5, Table 2 is then constructed as follows: for the 'always justification' and 'never justification' rows, we take the corresponding values in Table 5 for $\tau = 0$, $T = 1$ and subtract from them the same entry for $\tau = 1$, $T = 1$. For the 'ν is swing voter' entries, we take the 'always justification' entry for $\tau = 0$, $T = 1$ as the base profit, and subtract from them the relative deposit change only after the same epoch, so $1/(1+\rho)$, 1, and $1/(1+\rho)$ respectively.

D Discussion of The Parameter Choice

In order to find appropriate values for γ, β, and p, there are several factors to consider. Regarding γ:

- Setting the reward too high makes the protocol expensive to operate.
- Setting the reward too low means that fewer people will be willing to deposit, and a smaller set of participants means that the protocol is less secure.
- Setting rewards too low also increases the possibility of *discouragement attacks* [15], where an attacker either performs a censorship attacks or finds a way to cause high latency (e.g., by splitting the network), causing validators to lose money; the threat of this may encourage validators to leave or discourage them from joining.

Regarding β:

- Setting the penalties too low means that it takes very long to recover from a large number of (weighted) validators going offline, or for a minority fork to start finalising epochs.
- Setting the penalties too high may make it appear risky for powerful validators to deposit if they are unsure whether they are able to stay online.

Regarding p:

- Setting the deposit size dependence higher means that validators can make a larger profit by performing censorship or DoS attacks against other validators.
- Setting the deposit size dependence too low means that the protocol does not automatically adjust the interest rate depending on how risky potential validators perceive depositing to be [19].
- Users are looking for not just low issuance and high security, but also *stability* of the level of issuance and security. Having rewards decrease as the total deposit size increases (i.e., setting p high) accomplishes the former goal trivially, and the latter goal by "trying harder" to attract validators when the total deposit size is small. However, low values of p mean that from the perspective of a single validator, the interest rate is stable (e.g., if you deposit when the interest rate is 5%, you know that this will not drop dramatically even if a large sum is deposited by other validators).

The benchmark parameters were set as $\gamma = 7 \cdot 10^{-3}$, $\beta = 2 \cdot 10^{-7}$, and $p = \frac{1}{2}$. Here, p was chosen to strike a balance between $p = 0$ (constant interest rate per validator) and $p = 1$ (constant total amount of interest paid out by the protocol). In particular, given $p = \frac{1}{2}$, γ and β were derived by reverse-engineering the constants from two desired outcomes [47]:

- $\approx$5% annual interest to validators if 10M ETH has been deposited (assuming perfect operation of the protocol).
- Offline validators lose 50% of deposits in 3 weeks (21 days), if 10M ETH has been deposited in total and 50% of (weighted) validators (5M ETH) go offline.

E Long-Term Evolution of Deposits

In Section 4.3, we stated that the time until a censorship attack start to pay off can be approximated by

$$\tau' \approx \frac{(\sum_{\nu} D_{\nu,0})^p}{p\gamma}. \tag{7}$$

We motivate this approximation in this section of the appendix. In the following, we assume that all the validators always votes successfully (if a minor fraction of the validators abstain, then in our analysis this would be largely equivalent to γ being lower, and τ' therefore being higher). Let $D_n = \sum_{\nu \in V} D_{\nu,n}$ and for brevity let $y = \frac{1}{2}\gamma$. Then

$$\frac{D_{\nu,n}}{D_{\nu,n-1}} \equiv (1 + C_{n-1}) = 1 + \frac{1}{2}\rho = 1 + \frac{y}{D_{n-1}^p}.$$

The same holds for the total deposit, leading to the recurrence relation

$$D_n = \left(1 + \frac{y}{D_{n-1}^p}\right) D_{n-1} = D_{n-1} + y D_{n-1}^{1-p}, \tag{8}$$

which in turn implies

$$D_n = \prod_{j=1}^{n-1} \left(1 + \frac{y}{D_{j-1}^p}\right) D_0.$$

The following approximations for three concrete values of p are easy to check by hand, and will inform the more general case later on:

$$p = 0 \,:\; D_n = D_0 \sum_{j=0}^{n} \binom{n}{j} y^j \approx D_0 e^{ny}$$

$$p = \tfrac{1}{2} \,:\; D_n \approx \left(1 + \frac{\frac{1}{2}ny}{\sqrt{D_0}}\right)^2 D_0$$

$$p = 1 \,:\; D_n \approx \left(1 + \frac{ny}{D_0}\right) D_0.$$

In particular, the following holds in general for $p \in (0, 1]$:

$$D_n \approx \left(1 + \frac{pny}{D_0^p}\right)^{1/p} D_0 \tag{9}$$

The following is the sketch of a proof via mathematical induction: substituting (9) into the final expression in (8) yields the following (where the approximation

is valid if y/D_0^p is small, and because $(1+x)^p \approx 1+px$ for small x):

$$\begin{aligned}
&\left(1+\frac{pny}{D_0^p}\right)^{1/p} D_0 + y\left(\left(1+\frac{pny}{D_0^p}\right)^{1/p} D_0\right)^{1-p} \\
&= D_0\left(\left(1+\frac{pny}{D_0^p}\right)\left(1+\frac{y}{D_0^p\left(1+\frac{npy}{D_0^p}\right)}\right)^{p}\right)^{1/p} \\
&\approx D_0\left(\left(1+\frac{pny}{D_0^p}\right)\left(1+\frac{py}{D_0^p\left(1+\frac{npy}{D_0^p}\right)}\right)\right)^{1/p} \\
&= D_0\left(1+\frac{p(n+1)y}{D_0^p}\right)^{1/p}
\end{aligned}$$

We will use (9) to motivate (7). Let $D_{\alpha,n}$ denote the deposits of the attackers at epoch n if they do not perform the attack, and $\mathfrak{D}_{\alpha,n}$ if they do. We then have

$$D_{\alpha,\tau'} \approx \left(1+\frac{p\tau' y}{D_0^p}\right)^{1/p} D_{\alpha,0}$$

and

$$\mathfrak{D}_{\alpha,\tau} \approx \left(1+\frac{(1-h)p\tau y}{D_0^p}\right)^{1/p} D_{\alpha,0}$$

and

$$\mathfrak{D}_{\alpha,\tau'} \approx \left(1+\frac{p(\tau'-\tau)y}{\mathfrak{D}_\tau^p}\right)^{1/p} \mathfrak{D}_{\alpha,\tau},$$

where

$$\mathfrak{D}_\tau \approx \left(1+\frac{(1-3h)p\tau y}{D_0^p}\right)^{1/p} D_0$$

This means that we need to solve the following for τ':

$$\left(1+\frac{p\tau' y}{D_0^p}\right)^{1/p} D_{\alpha,0} = \left(1+\frac{(1-h)p\tau y}{D_0^p}\right)^{1/p}\left(1+\frac{p(\tau'-\tau)y}{\mathfrak{D}_\tau^p}\right)^{1/p} D_{\alpha,0}$$

or equivalently

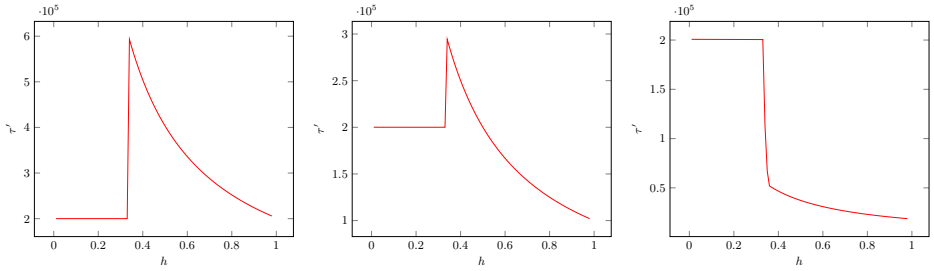
$$\frac{p\tau' y}{D_0^p} = \frac{(1-h)p\tau y}{D_0^p} + \frac{p(\tau'-\tau)y}{\mathfrak{D}_\tau^p} + \frac{(1-h)p\tau y}{D_0^p}\frac{p(\tau'-\tau)y}{\mathfrak{D}_\tau^p}$$

or

$$\tau' = (1-h)\tau + \frac{\tau'-\tau}{1+\frac{(1-3h)p\tau y}{D_0^p}}\left(1+\frac{(\tau'-\tau)+\tau p(\tau'-\tau)y}{D_0^p}\right).$$

Solving for τ' yields

$$\tau' = \frac{D_0^p + 3p\tau y - 3hp\tau y}{2py},$$



(a) $\tau = 10$ (b) $\tau = 100$ (c) $\tau = 1000$

Fig. 7: Values of τ' as a function of h, for different attack lengths τ. Here, $p = 0.5$, $\gamma = 0.01$, $D_0 = 1000000$. If $h < \frac{1}{3}$, then $\tau' \approx (\gamma p)^{-1} D_0^p = 200\,000$.

and since D_0^p is typically much larger than the other two terms in the numerator, this yields

$$\tau' \approx \frac{D_0^p}{2py} = \frac{D_0^p}{p\gamma},$$

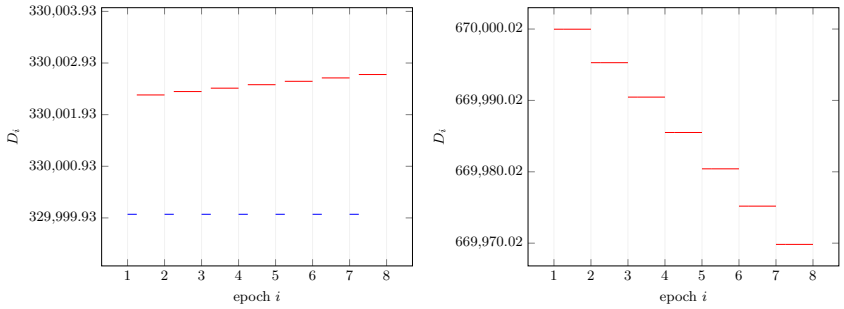
which is what we claimed in (7).

F Automated Censorship Detection

As discussed in Section 5.2, it is relatively easy for victims of censorship to notice that they are being censored. However, as discussed in Section 2.2, it is impossible for those who are neither victims nor attackers to determine if the absence of certain blocks or protocol messages is due to feigned censorship, actual censorship, or high latency. However, if they are able to identify 'suspicious' activity on the blockchain, then this could set off flags or in extreme cases cause a validator to stop voting until the issue has been resolved.

In [13], it was proposed that each client maintains a 'suspicion score' for each `vote` message that they've seen. This suspicion score is given by the time (as a timestamp or block height) at which a vote message appeared on the blockchain minus the time it was observed by the client, discounted in some way to give less importance to events in the distant past. If, e.g., the maximum of all these values across the blockchain is then above some threshold this could trigger a response from the client.

In [12], an approach was suggested based on the original Byzantine Generals paper [40] in the synchronous setting with signed messages. Here, validators have the ability to broadcast knowledge of blocks or votes that they have seen across the network: should other validator receive these messages within certain time constraints, they attach their signature to the message and propagate it further. This gives clients information about latencies and the connectivity of the validators, which can be used to check if the absence of certain votes or blocks matches what they would expect from the network.



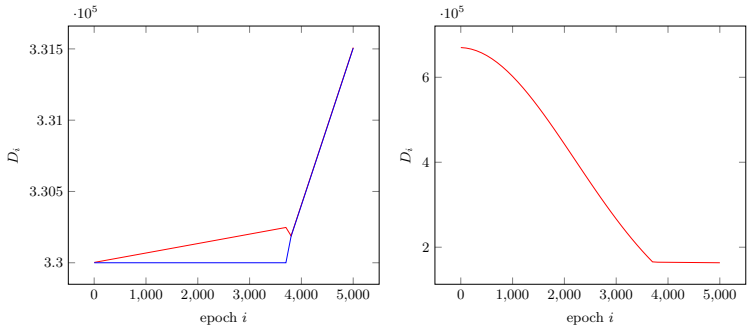
(a) Voting validator. (b) Non-voting validator.

Fig. 8: Evolution of the scaled deposits of voting and non-voting validators shortly after a minority fork. In the figure on the left, the red line represents the deposits after voting, and the blue line the deposits after the deposit scale factor update. The non-voting validators do not get rewards and their deposits shrink at the start of each epoch. In this graph, $\sum D = 10^6$.

Both approaches are interesting for future work.

G Effects of A Minority Fork

In this section we include figures displaying the evolution of validator deposits during a minority fork in the short and long term in Figure 8 and Figure 9 respectively. In both figures, the leftmost figure represents the deposits of those who vote on the minority fork — in the graphs on the left, the deposits immediately before (blue) and after (red) voting are displayed. Although the height of the blue lines remains constant, the values of the red lines increase as the reward factor increases. In the graphs on the right, the deposits of the non-voting validators are displayed, which only decrease.



(a) Voting validator. (b) Non-voting validator.

Fig. 9: Similar graph to Figure 8, but for a longer time scale. In this setting, finalization resumes after epoch 3706.