Byzantine Attacks Exploiting Penalties in Ethereum PoS

Ulysse Pavloff
Université Paris-Saclay, CEA, LIST
Palaiseau, France

Yackolley Amoussou-Guenou
Université Paris-Panthéon-Assas, CRED
Paris, France

Sara Tucci-Piergiovanni
Université Paris-Saclay, CEA, LIST
Palaiseau, France

Abstract—In May 2023, the Ethereum blockchain experienced its first inactivity leak, a mechanism designed to reinstate chain finalization amid persistent network disruptions. This mechanism aims to reduce the voting power of validators who are unreachable within the network, reallocating this power to active validators. This paper investigates the implications of the inactivity leak on safety within the Ethereum blockchain. Our theoretical analysis reveals scenarios where actions by Byzantine validators expedite the finalization of two conflicting branches, and instances where Byzantine validators reach a voting power exceeding the critical safety threshold of one-third. Additionally, we revisit the probabilistic bouncing attack, illustrating how the inactivity leak can result in a probabilistic breach of safety, potentially allowing Byzantine validators to exceed the one-third safety threshold. Our findings uncover how penalizing inactive nodes can compromise blockchain properties, particularly in the presence of Byzantine validators capable of coordinating actions.

Index Terms—Ethereum, Inactivity Leak, Safety, Liveness, Blockchain

I. INTRODUCTION

Ethereum has transitioned to its proof-of-stake (PoS) protocol in September 2022, making a shift from proof-of-work to the more energy efficient proof-of-stake. The design of the Ethereum PoS, however, is quite intricate and recent research focused on its formalization, properties, and thorough analyses [1], [2]. The distinct feature of the Ethereum PoS protocol is the hybridization of classical Byzantine Fault-Tolerant (BFT) Consensus [3] within the framework of Nakamoto-style blockchains [4]–[6].

In Nakamoto-style blockchains, each peer maintains a local tree-like data structure. A deterministic rule, often termed the fork-choice rule, selects a chain from this structure, allowing for forks but ultimately reconciling to a single common chain.

In contrast, BFT Consensus blockchains [7], [8] operate without admitting forks in the blockchain. Through explicit voting for block proposals, where participants' votes align with their stake in the network, these blockchains ensure that once a block is added to the chain, it remains permanently incorporated, never revoked due to a fork. In blockchain terms, such a block is *finalized*, signifying settlement of all transactions within it. To establish a finalized chain, BFT Consensus blockchains use Byzantine-tolerant super-majority quorums, assuming a tolerance threshold below one-third of Byzantine voting power.

In terms of properties, BFT Consensus blockchains are always safe because they do not fork, but can stop growing during network partitions or erratic network behavior. On the other hand, Nakamoto-style blockchains can fork during partitions or erratic network behavior but are always live, because the tree will not stop to grow under these circumstances. The trade-off between Safety and Liveness during network partitions is a striking consequence of the CAP theorem [9], [10].

Instead, Ethereum PoS uniquely features a finalized chain as the root of a chain that can fork. Chain finalization relies on Byzantine voting quorums, while a fork-chain rule aids validators in resolving forks. Maintaining both finalized and non-finalized chains within a single data structure, Ethereum PoS strives for a delicate balance between safety and chain growth. Safety within Ethereum asserts the non-forkable nature of the finalized chain, while Liveness guarantees its continued growth, akin to BFT Consensus blockchains. However, in contrast to BFT Consensus blockchains, Ethereum PoS ensures that block addition – in the forkable part of the chain – remains unimpeded, even during network disruptions or failures [1].

Nonetheless, finalization procedures, based on Byzantine quorums, still carry the inherent limitation that if more than one-third of the voting power resides with honest validators who are unreachable within the network, the protocol may fail to achieve finalization. These validators are perceived as inactive and do not contribute to finalization. To mitigate this scenario, thereby restarting finalization in case of long periods of bad network behavior, the *inactivity leak*, was introduced [11]. Intuitively, the inactivity leak is a penalty mechanism that erodes the stake of inactive validators (and with that their voting power) to redistribute voting power to active validators. The inactivity leak can be seen as mechanism to restore Liveness during network partitions, which then poses a theoretical risk on Safety by the CAP theorem. Indeed, the initial introduction of the inactivity leak briefly mentions the prospect of conflicting finalized blocks, meaning a loss of Safety, however, it falls short in elucidating the exact events that could lead to these issues and their severity. This is particularly crucial in the context of potential interference by Byzantine validators, a variable that necessitates closer examination.

We address this question and give the first formal descrip-

---

[1] This attribute, essentially non-blocking Liveness, is denoted as Availability in [2].

tion of the inactivity leak and its impact on the protocol by considering different configurations in terms of initial Byzantine voting power and outcome in terms of Safety loss. We are interested for Safety loss in two distinctive outcomes: (1) the finalization of two conflicting chains, and (2) the break of the Safety threshold, meaning the Byzantine stake proportion of more than one-third.

Our theoretical analysis aims to pinpoint the significant role the inactivity leak mechanism plays in the Ethereum protocol. We first assess conflicting finalization when all validators are honest. By leveraging the CAP theorem, we emulate a network partition to establish the time it takes for the network to restore Liveness while breaking Safety because of network regions finalizing independently on different chains.

We then analyze, considering different Byzantine initial voting power proportions, the time it takes to break Safety under the same network condition and where Byzantine validators can coordinate and are not affected by network partition. We present attacks in which Byzantine validators can either accelerate the time that it takes to break Safety by finalizing conflicting chains or break the Safety threshold acquiring more than one-third of voting power. We also revisit a known attack called the Probabilistic Bouncing attack in light of the inactivity leak. The attack was initially identified as a means to delay finalization even during periods of good network conditions. We demonstrate that this attack can be exploited to break the Safety threshold in a probabilistic manner during network synchronous periods, i.e., without partition.

Let us note that the conditions required for these attacks to happen and persist, to the point of actually threatening Safety, make them unlikely in real-world settings, primarily due to the time scales involved. Nonetheless, Byzantine validators can arguably cause other validators to lose stake before compromising Safety. Indeed our work focuses on mechanisms penalizing inactive validators, akin to those found in other PoS blockchains such as Polkadot [12] or Tezos [13]. In this respect, we consider our analysis a crucial endeavor in exploring penalties mechanisms within Byzantine Fault Tolerance (BFT) analysis, aiming to provide valuable insights for future research.

The paper is organised as follows: we first present a description of the system model and blockchain properties (section II), prerequisites for understanding the following section detailing Ethereum's protocol (section III). Subsequently, we delve into the formalization of the inactivity leak (section IV) and go through the aforementioned scenarios(section V) for our analysis. Then comes the related work (section VI) before reaching our conclusions.

II. System Model & Blockchain Properties

We consider a system composed of a finite set Π of processes called *validators*. There are a total of n validators. Each validator owns a *stake*. The stake refers to the amount of cryptocurrency (ETH) owned by each validator, serving as a metric of their influence in the consensus protocol (cf. section III). Validators possess a unique public/private key pair

used for cryptographic signing, and can be identified by their public key. We assume that digital signatures cannot be forged. Validators have synchronized clocks[2]. Time is measured by periods of 12 seconds called *slots*, a period of 32 slots is called an *epoch* which serves as the largest time unit in the protocol. Throughout the remainder of this paper, we employ the term "proportion" concerning a validator set to denote the ratio of their combined stake to the total value staked. Initially capped at 32 ETH, the stake of individual validators has the potential to decrease.

Network: Validators communicate by message passing. We assume the existence of an underlying broadcast primitive, which is a best effort broadcast. This means that when a correct validator broadcasts a value, all the correct validators eventually deliver it. Messages are generated with a digital signature, providing a mechanism for cryptographic identification and validation in the protocol.

We assume a *partially synchronous model* [14], where the system transitions from an asynchronous state to a synchronous state after an a-priori unknown Global Stabilization Time (GST). Before GST there is an asynchronous period in which there is no bound on the message transfer delay Δ. After GST there is a known finite bound on Δ. Note that all messages sent before GST are received at most at time GST $+ \Delta$. Note that even if we have synchronized clocks, having an asynchronous network before GST still makes the system partially synchronous.

For a significant portion of our analysis, we assume a network configuration wherein, during asynchronous periods, honest validators are separated into two distinct partitions. Communication between these partitions is restricted, reflecting a scenario where two regions are temporarily isolated yet maintain internal communication. This setup emulates the scenario where two regions of the world are temporarily unreachable from each other, while maintaining unaffected communication within each region.

Fault Model: Validators fall into two categories: *honest* and *Byzantine*. Honest validators, also called correct validators, adhere to the protocol, while Byzantine validators may arbitrarily deviate from the protocol[3]. Following the literature [15], we allow for a strong adversary that can coordinate Byzantine validators, even across network partitions[4], thereby remaining unaffected by such partitions. But contrary to Castro and Liskov [15], the adversary does not manipulate message delay between honest validators. We denote by β_0 the initial proportion of Byzantine validators' stake, with $\beta_0 < 1/3$.

---

[2]Clocks can be offset by at most τ, this way, the offset can be captured as part of the network delay.

[3]Since in this paper we are only interested in the consensus part of the protocol, we only characterize validator's behavior. For clients submitting transactions, as in any blockchain, we assume they can be Byzantine without impact on our analysis.

[4]In our analysis, only the first three attacks leverage this power of the adversary because they occur during the asynchronous period, in which partition can occur. Meanwhile, the probabilistic bouncing attack happens in the synchronous period, so this power of the adversary is not necessary for the attack.

The Ethereum Proof-of-Stake (PoS) protocol aspires to achieve Byzantine Fault Tolerance (BFT), ensuring the preservation of Safety and Liveness properties for any initial Byzantine stake proportion (β_0) strictly below 1/3.

Ethereum PoS Properties: Validators keep a local data structure in form of a tree containing all the blocks perceived, then a consensus protocol helps to choose a unique chain in the tree. Ethereum has a particular trait that consists in having a *finalized* chain as prefix of a chain vulnerable to forks. A metaphor for this is that the finalized chain is the trunk that supports possibly various branches, and as time passes, the trunk grows and branches are trimmed[5].

Intuitively, the Safety property of Ethereum states that the finalized chain is not forkable, while the Liveness property states that the finalized chain always grows. The nuance with respect to classical consensus protocol is the existence of an Availability property on the entire chain that guarantees a constant growth of the chain despite failures and network partitions. We report here a simplified version of the formal Ethereum PoS properties defined in [2] for enhanced clarity and self-contained reference, as follows:

*Definition 1 (**Candidate chain**):* The candidate chain is the chain designated as the one to build upon according to the fork choice rule.
The candidate chain can be seen as the entire chain, from the genesis to the last block perceived by a validator. The blocks in the candidate chain can be finalized or not.

*Definition 2 (**Finalized block**):* A block is finalized for a validator if and only if the block cannot be revoked, i.e., it permanently belongs to the validator's candidate chain.

It stems from the definition that all the predecessors of a finalized block are also finalized.

*Definition 3 (**Finalized chain**):* The finalized chain is the chain constituted of all the finalized blocks.

The finalized chain is always a prefix of any candidate chain.

Safety, Availability, and Liveness are expressed as follows:

*Property 1 (**Safety**):* A blockchain achieves **safety** if for any two correct validators with a finalized chain, then one chain is necessarily the prefix of the other.

*Property 2 (**Availability**):* A blockchain is **available** if the following two conditions hold: (1) any correct validator is able to append a block to its candidate chain in bounded time, regardless of the failures of other validators and the network partitions; (2) the candidate chains of all correct validators are eventually growing.

*Property 3 (**Liveness**):* A blockchain is **live** if the finalized chain is eventually growing.

Accordingly to the aforementioned definitions, this paper considers forks within the finalized chain as a loss of Safety. As explained in the subsequent section, forks occurring within the candidate chain suffix, which has not yet been finalized, are resolved by the fork choice rule of the protocol. This rule determines the chain upon which validators vote and build. However, this rule has not been explicitly designed to

---

handle forks impacting the finalized chain. The protocol is not intended to fork the finalized chain, as the finalization process depends on a super-majority vote, ensuring Safety when the Byzantine stake is less than one third, i.e., $\beta_0 < 1/3$. We look at two types of Safety loss: (1) the finalization of two conflicting chains, and (2) the break of the Safety threshold, meaning the Byzantine stake proportion of more than one-third.

III. Ethereum protocol

In this work we study a specific mechanism of the Ethereum protocol, the *inactivity leak*. To help our explanation and analysis of the inactivity leak, we first introduce some[6] necessary understanding of the protocol.

A. Structure

In Ethereum PoS, time is measured in *slots* and *epochs*. A slot lasts 12 seconds. Slots are assigned with consecutive numbers; the first slot is slot 0. Slots are encapsulated in *epochs*. An epoch is composed of 32 slots, thus lasting 6 minutes and 24 seconds. The first epoch (epoch 0) contains from slot 0 to slot 31; then epoch 1 contains slot 32 to 63, and so on. These slots and epochs allow associating the validators' roles to the corresponding time frame. An essential feature of epochs is the *checkpoint*. A checkpoint is a pair block-epoch (b, e) where b is the block of the first slot of epoch e.

An epoch can be seen as a round of a BFT Consensus in which all validators vote to reach consensus about which checkpoint to finalize.
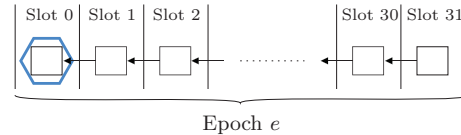

Fig. 1. Ethereum protocol Structure

B. Consensus

Validators have two main roles: *proposer* and *attester*. The proposer's role consists in proposing a block during a specific slot; this role is pseudo-randomly given to 32 validators by epoch (one for each slot). The attester's role consists in producing an attestation sharing the validator's view of the chain; this role is given once by epoch to each validator.

The attestation contains two votes, a *block vote* and a *checkpoint vote*. The block vote is used in the *fork choice rule* which determines the chain to vote and build upon for validators. As its name suggests, the checkpoint vote points to checkpoints constituting the chain. It is used to justify and finalize blocks to grow the finalized chain. Justification is the step prior to finalization. If validators controlling more than two-thirds of the stake make the same checkpoint vote, then

---

[5]In the the paper, we use the terms "chain" and "branch" interchangeably.

[6]We omit here a lot of details for the sake of space and simplicity. An interested reader must refer to [1], [2], [16], [17] for more thorough explanation of the protocol.

the checkpoint target is said justified. Finalization occurs when there are two consecutive justified checkpoints (one in epoch e and the following one in epoch $e+1$).

Let us note that if justification occurs only every other epoch, finalization is not possible.

C. Incentives

The Ethereum PoS protocol provides validators with rewards and penalties to incentivize timely responses reaching consensus. There are three different types of penalties: slashing, attestation penalties, and inactivity penalties. (i) *Slashing penalties.* Validators face slashing if they provably violate specific protocol rules, resulting in a partial loss of their stake and expulsion from the validator set. (ii) *Attestation penalties.* To incentivize timely and correct attestations (votes), the protocol rewards validators for adhering to the protocol and penalizes those who do not. If an attestation is missing or belatedly incorporated into the chain, its validator gets penalized. (iii) **Inactivity penalties.** Each epoch a validator is deemed inactive, its *inactivity score* increments. However, if the protocol is not in an inactivity leak, all inactivity scores are reduced. When finalization occurs regularly, a validator that is deemed inactive only gets attestation penalties. This changes when there is no finalization for four consecutive epochs: the inactivity leak begins. During the inactivity leak, that starts when there is no finalization for four consecutive epochs, all validators will get inactivity penalties directly linked to their stake and *inactivity score*. The inactivity score varies with the validator's activity. In addition to penalties, rewards are attributed for timely/correct attestation but not during the inactivity leak. Our analysis of the impact of the inactivity leak on the protocol takes into consideration the slashing and inactivity penalties across 5 different scenarios (cf. section V).

Having provided a comprehensive overview of the Ethereum PoS consensus mechanism, we are now well-positioned to delve into the specifics of the *inactivity leak*.

IV. INACTIVITY LEAK

The Ethereum PoS blockchain strives for the continuous growth of the finalized chain. In consequence, the protocol incentivize validators to finalize blocks actively. In the absence of finalization, validators incur penalties.

The inactivity leak, introduced in [11], serves as a mechanism to regain finality. Specifically, if a chain has not undergone finalization for four consecutive epochs, the inactivity leak is initiated. During the inactivity leak, the stakes of *inactive* validators are drained until active validators amount for two-thirds of the stake. A validator is labeled as inactive for a particular epoch if it fails to send an attestation or sends one with a wrong target checkpoint.

During the inactivity leak, there are no more rewards given to attesters[7], and additional penalties are imposed on inactive validators.

---

[7]Actually, the only rewards to remain are for the block producers and the sync committees.

A. Inactivity Score

The *inactivity score* is a dynamic variable that adjusts based on a validator's activity. The inactivity score of a validator is determined based on the attestations contained in the chain. It is important to note that if there are multiple branches, a validator's inactivity score depends on the selected branch. Within an epoch, being active on one branch implies[8] inactivity on another (for honest validators).

More precisely, the inactivity score is updated every epoch: if validator i is active, then its inactivity score is reduced by 1; otherwise, 4 is added to it. When the inactivity leak is not in place, every epoch the inactivity scores are decreased by 16, which often nullifies low inactivity scores.

During an inactivity leak, at epoch t, the inactivity score, $I_i(t)$, of validator i is:

$$\begin{cases} I_i(t) = I_i(t-1) + 4, \text{if } i \text{ is inactive at epoch } t \\ I_i(t) = \max(I_i(t-1) - 1, 0), \text{ otherwise.} \end{cases} \quad (1)$$

Each attester has thus an inactivity score that fluctuates depending on its (in)activity. In the protocol, the inactivity score is always superior to zero. A validator's inactivity for epoch t is determined by whether it sent an attestation for this epoch or if the sent attestation contains a correct checkpoint vote. Here "correct" implies that the target of the checkpoint vote belongs to the chain considering it.

B. Inactivity penalties

Validators that are deemed inactive incur penalties. Let $s_i(t)$ represent the stake of validator i at epoch t, and let $I_i(t)$ denote its inactivity score. The penalty at each epoch t is $I_i(t-1) \cdot s_i(t-1)/2^{26}$. Therefore, the evolution of the stake is expressed by:

$$s_i(t) = s_i(t-1) - \frac{I_i(t-1) \cdot s_i(t-1)}{2^{26}}. \quad (2)$$

C. Stake's functions during an inactivity leak

In this work, we model the stake function s (see Equation 2) as a continuous and differentiable function, yielding the following differential equation:

$$s'(t) = -I(t) \cdot s(t)/2^{26}. \quad (3)$$

We then explore three distinct validator's behaviors during inactivity leak, each influencing their inactivity score, and consequently, their stake.

(a) Active validators: they are always active.
(b) Semi-active validators: they are active every two epochs.
(c) Inactive validators: they are always inactive.

Note that, in case of a fork, this categorization depends on the specific branch under consideration as different branches may yield different evaluation of each validator's behavior.

Validators with these three behaviors experience a different evolution in their inactivity score: (a) Active validators have a constant inactivity score $I(t) = 0$; (b) Semi-active validators' inactivity score increases by 3 every two epochs ($+4-1$), their

---

[8]This is true as long as the chain differ for at least one epoch.

inactivity score on average at t is $I(t) = 3t/2$; (c) Inactive validators' inactivity score increases by 4 every epochs $I(t) = 4t$. Thanks to Equation 3 we can determine the evolution of the stake of each type of validators during an inactivity leak:

(a) Active validator's stake: $s(t) = s_0 = 32$.
(b) Semi-active validator's stake: $s(t) = s_0 e^{-3t^2/2^{28}}$.
(c) Inactive validator's stake: $s(t) = s_0 e^{-t^2/2^{25}}$.

This categorization is orthogonal to the Byzantine-Honest categorization. For instance, an honest validator can appear inactive in one branch due to poor connectivity or an asynchronous period (due to network partition or congestion). On the other hand, a Byzantine validator intentionally chooses one of these behaviors (e.g., being semi-active) to execute the attacks.
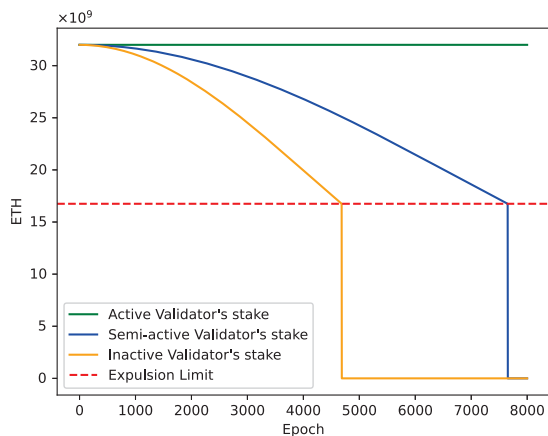

Fig. 2. This figure shows 3 different stake's trajectories in the event of an inactivity leak: the stake of a validator active every epoch, the stake of a validator active every other epoch (semi-active) and an inactive validator. The inactive validators get ejected at epoch $t = 4685$. The semi-active validators get ejected at $t = 7652$. For reference, 5000 epochs is about 3 weeks.

We illustrate in Figure 2 the evolution of the validators' stake depending on their behaviors. We also account for the ejection of validators with a stake lower or equal than 16.75.

Using these newly defined stake functions, we explore five different scenarios in section V.

The first scenario, with only honest validators, serves as a baseline to assess the impact of Byzantine validators. Even in this seemingly straightforward setting, Safety is compromised.

In the second scenario, Byzantine validators come into play and aim at expediting finalization on conflicting branches. They do so by performing slashable actions. Thus, they will get ejected from the set of validators once communication is restored among honest validators and evidence of their slashable offense is included in a block. We outline their impact based on their initial stake proportion. With an initial stake proportion of $\beta_0 = 0.2$, the finalization on conflicting chains occurs after 3107 epochs. With $\beta_0 = 0.33$, the conflicting finalization occurs only after 503 epochs.

Scenario		Outcomes
V-A	All honest	2 finalized branches
V-B1	Slashable Byzantine	2 finalized branches
V-B2	Non slashable Byzantine	2 finalized branches
V-B3	Non slashable Byzantine	$\beta > 1/3$
V-C	Probabilistic Bouncing attack	$\beta > 1/3$ probably

In the third and forth scenarios Byzantine validators show non-slashable behaviors. Indeed, Byzantine validators are semi-active, which means they are active on both chains but in a non-slashable manner. However, in the third scenario they aim at finalizing on conflicting branches as soon as possible – conflicting finalization in 556 epochs, for an initial stake proportion of $\beta_0 = 0.33$ – while in the fourth scenario they aim for their stake proportion to go over the 1/3 threshold.

The last scenario delves into the effect of the probabilistic bouncing attack regarding the Byzantine stake proportion taking the inactivity leak into account. In this attack Byzantine validators initially aim at delaying finality by being alternatively active (bounce) on both chains of a fork. This confuses honest validators that also bounce from one chain to the other. We detail how to find the distribution of honest validators' stake in this setting considering the inactivity penalties. We also cover how the Byzantine validators' stake proportion can go over 1/3 if their initial proportion is close enough to 1/3.

The scenarios unfold within the context of a partially synchronous network while offering a meticulous examination of the property of Safety and the evolution of the proportion of Byzantine validators. The variations in initial conditions and outcomes of each scenario are summarized in Table I.

V. ANALYSIS

In this section we study the robustness of the Safety property within the context of the inactivity leak. By construction, in case of a prolonged partition, two different chains can potentially be finalized, leading to conflicting finalized blocks. We delineate scenarios that can produce such predicament.

Considering the presence of Byzantine validators, we study how the proportion of Byzantine validators' stake evolves during an inactivity leak. Furthermore, we are interested in scenarios where the inactivity leak mechanism becomes the backbone of an attack strategy, potentially causing the proportion of Byzantine stakes to exceed the 1/3 security threshold (cf. subsubsection V-B3 and subsection V-C).

A. GST upper bound for Safety

In this first subsection, we look for an upper bound on GST before which no finalization on conflicting chains can happen in case of a partition. We study the case of an inactivity leak with these conditions: (i) Only honest validators, no Byzantine validators, and (ii) the network is asynchronous (before GST).

In case of catastrophic events, during an instance of a particularly disrupted network, an arbitrary large set of honest validators might be unreachable before GST. During this asynchronous period, the subset of validators still communicating with each other will continue to try to finalize new blocks. We assume that, within each partition, the message delay is bounded as in the synchronous period; however, communication between partitions is not restored before the GST. As mentioned in the system model, Byzantine validators can communicate between partitions without restriction but cannot manipulate the message delay between honest validators. The active validators must represent more than two-third of the stake to be able to finalize. After 4 epochs without finalization, the inactivity leak starts.

All the validators deemed inactive will get their stake reduced progressively. This will continue until the active validator's constitute at least two thirds of the stake and finalize anew.

a) Two finalized chains: A noteworthy scenario arises during asynchronous periods that can lead to a network partition and the creation of two distinct finalized chains. If this partition persists for an extended period, both chains independently drain the stakes of validators they consider inactive until they finalize again. Although the protocol permits this behavior by design, it results in the finalization of two conflicting chains, thereby compromising the Safety property.

This outcome is in line with Ethereum PoS prioritizing Liveness of Safety. But to the best of our knowledge, this corner case has not been discussed in details.

We can theoretically assess the time required to finalize both branches of the fork. Suppose honest validators remain on their respective branches due to the partition. In this case, by understanding the distribution of these validators across the partitions, we can compute the time it takes for the proportion of active validators' stake to return to 2/3 of the stake on each branch, permitting new finalization.

Let n_H and n_B denote the initial number of honest validators and Byzantine validators at the beginning of the inactivity leak ($n_H + n_B = n$). Additionally, n_{H_1} and n_{H_2} represents the number of honest validators active on branch 1 and on branch 2, respectively ($n_{H_1} + n_{H_2} = n_H$).

We denote by $p_0 = n_{H_1}/n_H$ the initial proportion of honest validators remaining active on branch 1, and $1 - p_0 = n_{H_2}/n_H$ represents the proportion of honest validators active on branch 2 (hence inactive on branch 1). In this first scenario, with only honest validators and no Byzantine validators, p_0 represents the proportion of all validators active on branch 1. Indeed, since $n_H/n = 1$ we have that $n_{H_1}/n_H \times n_H/n = p_0$.

We have assessed how validators' stakes vary based on their level of activity, we can in consequence express the ratio of active validators on branch 1 at time t:

$$\frac{n_{H_1}s_{H_1}(t)}{n_{H_1}s_{H_1}(t) + n_{H_2}s_{H_2}(t)}, \tag{4}$$

with s_{H_1} and s_{H_2} being the stake of honest active and inactive validators, respectively. We know the function of their stake according to time, and by dividing in the numerator and the denominator by the total number of validators ($n = n_H$), we can rewrite Equation 4 as:

$$\frac{p_0}{p_0 + (1 - p_0)e^{-t^2/2^{25}}}. \tag{5}$$

The initial stake value s_0 is factored out of the equation. This function is critical as the moment it reaches 2/3 or more, finalization can occur[9] on the branch.

To establish the upper bound on GST under which two conflicting branches finalize, we must find when finalization occurs on each branch, for each initial proportion of active validators p_0 and inactive validators $1 - p_0$. We simulate the evolution of the ratio of active validators (Equation 5) during an inactivity leak with different values of p_0 in Figure 3. The simulation starts with both active and inactive validators at 32 ETH. At epoch 0, the inactivity leak begins.

For $p_0 = 0.5$ or less, the ratio jumps to 1 at $t = 4685$, this is due to the fact that validators with a stake below 16.75 ETH are ejected from the set of validators. Conversely for $p_0 = 0.6$, the proportion of active validators do not jump drastically as 2/3 of active validators is regained before the ejection of inactive validators, permitting the active validator to finalize, hence ending the inactivity leak. Interestingly, with $p = 0.6$ we can see that the ratio still increases several epochs after the proportion of 2/3 of active validators' stake is reached. This is because the penalties for inactive validators take some time to return to zero.
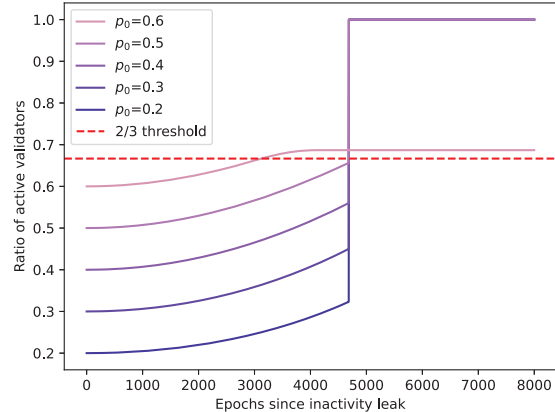

Fig. 3. Evolution of the ratio of active validators depending on the proportion p_0 of active validators on the branch. This follows the ratio given in Equation 5 before regaining 2/3 of active validators or the expulsion of inactive validators at epoch $t = 4685$.

As expected and shown by Figure 3, a chain with more active validators will regain finality quicker. To ascertain how

---

[9]With a proportion of two-thirds of validators' stake active justification then finalization can occur in 2 epochs.

quick, we seek when the ratio is equal to 2/3. Taking into account the expulsion[10] of inactive validators at $t = 4685$, we can find the value t at which the 2/3 threshold is reached:

$$t = \min\left(\sqrt{2^{25}[\log(2(1-p_0)) - \log(p_0)]}, 4685\right). \quad (6)$$

This calculation pertains to $0 < p_0 < 2/3$ (when there is less that 2/3 of active validators) ensuring that the epoch t can be computed.

Conflicting finalization occurs once the slowest branch to finalize has regained finality. Our observation highlights that the lower the proportion of active validators, the slower the branch will regain finality. Hence, the fastest way to reach finality on both chains would be for honest validators to be evenly proportionate with half validators active on one chain and the other half on the other chain ($p_0 = 1 - p_0 = 0.5$). In this case, the ratio of active validators amount to 2/3 on both chain at $t = 4685$ epochs (about 3 weeks). We can note here that even with the best configuration to finalize quickly on conflicting branches, it is impossible to be faster than 4685 epochs. Thus with only honest validators, whatever their proportion on each branch, the last chain to finalize will always finalize at $t = 4685$.

Finality on both chains is achieved precisely at 4686 epochs after the beginning of the inactivity leak. Adding an epoch is necessary after gaining 2/3 of active stake to finalize the preceding justified checkpoint. This finalization ends the inactivity leak which has lasted approximately 3 weeks. *Any network partition lasting longer than 4686 epochs will result in a loss of Safety because of conflicting finalization. This is an upper bound for Safety on the duration on inactivity leak with only honest validators.*

B. Upper bound decrease due to Byzantine validators

In a trivial setup with only honest validators, Safety does not hold if the inactivity leak is not resolved quickly. This prompts us to study the scenario in the presence of Byzantine validators to evaluate how much they will be able to hasten the conflicting finalization. We describe two possible outcomes, the first one violates Safety but Byzantine validators gets slashed, the second one violates Safety as well but no validators gets slashed. A slashing penalty entails an ejection of the validator set as well as a loss of part of the validator's stake. Both scenarios expedite the time t at which Safety is breached, with different velocity depending on the chosen method.

We study the inactivity leak with these conditions: (i) at the beginning, less than one third of the stake is held by Byzantine validators ($\beta_0 = n_B/n < 1/3$), the rest is held by honest validators ($1 - \beta_0 = n_H/n$); (ii) The network is asynchronous (before GST); and (iii) Byzantine validators control the network delay, allowing them to be active on both branches while preventing honest validators from even observing the branch on which they are not active. They

---

[10]We drew inspiration for this initial work from the insights presented in [17].

can manipulate message delays of honest nodes to simulate a partition between honest validators.

The situation is the following:

- Honest validators are divided into branches 1 and 2; a proportion $p_0 = n_{H_1}/n_H$ of the honest validators are active on branch 1 while a proportion $1 - p_0 = n_{H_2}/n_H$ are active on branch 2. (Meaning that on branch 1, a proportion $n_{H_1}/n_H \times n_H/n = p_0(1 - \beta_0)$ are honest and active and a proportion $n_{H_2}/n_H \times n_H/n = (1 - p_0)(1 - \beta_0)$ are honest and inactive).
- Byzantine validators are not restricted to either partition, they are connected to both.

1) With slashing: In the event of a fork during asynchronous times, Byzantine validators can be active on the two branches (Figure 4). Being active on two branches means sending correct attestations on both every epoch. Such behavior is considered a slashable offense, incurring penalties, but only if detected by honest validators. The slashable offense is punished once a proof of conflicting attestation during the same epoch has been included in a block. Thus, before GST, Byzantine validators could operate on both branches without facing punishment as long as honest validators are unaware of the conflicting attestations. Byzantine validators have control over the message delay before GST rending this behavior possible. They can thereby expedite the finalization on different branches.
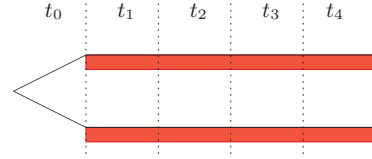
Fig. 4. Byzantine validators active on both chains of a fork at the same time during asynchronous times.

We study here the time needed for finalization to occur on conflicting branches depending on the proportion of Byzantine validators. The ratio of active validators at epoch t is:

$$\frac{n_{H_1}s_{H_1}(t) + n_B s_B(t)}{n_{H_1}s_{H_1}(t) + n_B s_B(t) + n_{H_2}s_{H_2}(t)}, \quad (7)$$

with s_{H_1}, s_B and s_{H_2} being the stake of honest active, Byzantine active and honest inactive validators, respectively. This can be rewritten as:

$$\frac{p_0(1-\beta_0) + \beta_0}{p_0(1-\beta_0) + \beta_0 + (1-p_0)(1-\beta_0)e^{-t^2/2^{25}}}, \quad (8)$$

where β_0 represents the initial proportion of Byzantine validators, and p_0 denotes the initial proportion of honest active validators. In contrast to the analysis with only honest validators (cf. Equation 5), here, Byzantine validators are present and active on both chains. Nonetheless, as before, we can obtain the ratio of active validators on the other branch just by interchanging p_0 and $1-p_0$. Finality on conflicting branches occurs when the last of the two branches finalizes. Similarly to

the previous example, the branch with the fewer initial honest active validators (p_0) will finalize the latest. This happens t epochs after the beginning of the inactivity leak, with

$$t = \min\left(\sqrt{2^{25}\left[\log(2(1-p_0)) - \log(p_0 + \frac{\beta_0}{1-\beta_0})\right]}, 4685\right). \tag{9}$$

Finality on conflicting branches is achieved the quickest when honest validators are evenly split between the branches of the fork, for $p_0 = 0.5$.

TABLE II
TIME BEFORE FINALIZATION ON CONFLICTING BRANCHES DEPENDING ON THE INITIAL PROPORTION OF BYZANTINE VALIDATORS β_0 FOR $p_0 = 0.5$ WITH SLASHING BEHAVIOUR BASED ON EQUATION 9.

β_0	t
0	**4685**
0.1	4066
0.15	3622
0.2	3107
0.33	502

Table II gives the epoch at which concurrent finalization occurs for $p_0 = 0.5$. This outline the rapidity at which finality can be regained depending on the initial proportion β_0 of Byzantine validators' stake. The table shows that 503 epochs (approximately 2 days) could suffice to finalize blocks on two different chains, but hypothetically it could be quicker than that. In fact, as β_0 gets closer to 1/3, the number of epochs required before concurrent finalization occurs (Equation 9) approaches 0. The explanation is that if β_0 were to start at exactly 1/3, then with $p_0 = 0.5$ it would mean that on each branch we would start with $p_0(1 - \beta_0) + \beta_0 = 2/3$ of active validators, hence finalizing immediately. This explains why if β_0 is very close to 1/3, the proportion of active validators reaches 2/3 rapidly. *Hence, Byzantine validators can expedite the loss of Safety. If their initial proportion is 0.33, they can make conflicting finalization occur approximately ten times faster than scenarios involving only honest participants.*

One can notice that if Byzantine validators act in a slashable manner, they will be penalized after the asynchronous period ends. However, the harm is already done. Once the finalization on two branches has occurred, the branches are irreconcilable with the current protocol. Next, we demonstrate that Byzantine validators can employ more subtle strategies to break Safety without slashable actions.

2) Without slashing: The Byzantine validators have a way to hasten the violation of the Safety property without incurring slashable offense. While not as rapid as being active on both branches simultaneously, they can be semi-active on both branches alternatively. Being semi-active on each branch means they are only active every other epoch. This approach diminishes their stake on each branch due to inactivity penalties. Nevertheless, at some point they will be able to finalize on two conflicting branches by being active two epochs in a row

on one branch then on the other (see Figure 5). Byzantine validators will be able to finalize when the proportion of their stake plus the proportion of the stake of honest active validators is above 2/3 on the branch (cf. Equation 10).
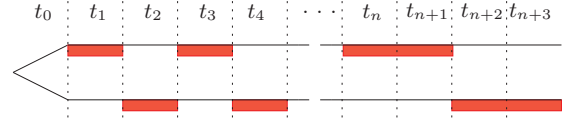

Fig. 5. Byzantine validators active on both branches of a fork alternatively during asynchronous times.

At that point, Byzantine validators must remain active for two consecutive epochs on each branch to finalized them both. If they are only semi-active, they can alternate justifications for checkpoints on each branch but will not achieve finalization. However, by maintaining activity for two consecutive epochs, first on one branch and then on the other, they ensure two sequential justifications, leading to the finalization of a checkpoint.

We gave the different evolution of stakes depending on the activity of validators (subsection IV-C). Now that Byzantine validators are semi-active, their stake follows the curve $s_0 e^{-3t^2/2^{28}}$. We simplify the ratio as previously and we get that finalization occurs on the branch when the ratio

$$\frac{p_0(1-\beta_0) + \beta_0 e^{-3t^2/2^{28}}}{p_0(1-\beta_0) + \beta_0 e^{-3t^2/2^{28}} + (1-p_0)(1-\beta_0)e^{-t^2/2^{25}}} \tag{10}$$

goes over 2/3. With β_0 and p_0 being the initial proportion of Byzantine validators and the proportion of honest active validators on the branch, respectively.

In contrast to the previous scenario, obtaining an analytic solution for t to determine the epoch when the ratio hits 2/3 is not straightforward. Therefore, we apply numerical methods on Equation 10 with initial parameters $p_0 = 0.5$ and $\beta_0 = 0.33$, resulting in a calculated t value of 555.65. Meaning, it will take 556 epochs to finalize, about 2 days and a half. As previously, the proximity of β_0 to 1/3 significantly influences the speed of finalization, as outlined in Table III and Figure 6. Figure 6 shows how the proportion of Byzantine affects the time of conflicting finalization. Let us notice that although the acceleration is not as pronounced as in the previous scenario, it remains noteworthy that Byzantine validators still exert a substantial impact on breaching Safety, while not committing any slashable offense. *Hence, Byzantine validators can expedite the loss of Safety without committing any slashable action. If their initial proportion is 0.33, they can make conflicting finalization occur approximately eight times faster than scenarios involving only honest participants.*

Another consequence of being "semi-active" on both branches is that Byzantine validators can decide when finalizing on each branch. Indeed, even when the proportion of their stake plus the proportion of honest and active validators' stake is above 2/3, finalization only occurs when the Byzantine validators stay active two consecutive epochs on the same chain.

TABLE III

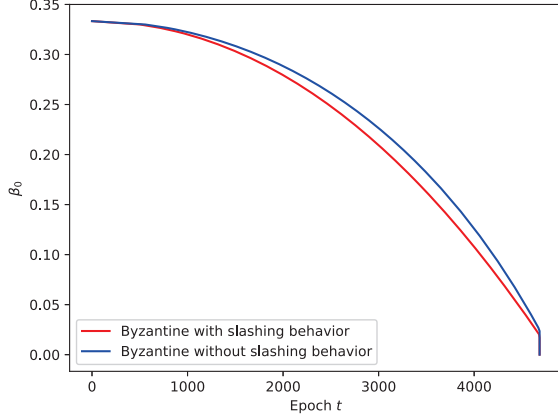| β_0	t
0	**4685**
0.1	4221
0.15	3819
0.2	3328
0.33	556

Fig. 6. Time before finalization on conflicting branches, depending on the initial proportion of Byzantine validators β_0 and whether they engage in slashable actions.

Being active for two epochs will justify the two consecutive epochs, thus finalizing an epoch.

There exists a scenario in which the Byzantine validators might delay finalization intentionally, aiming to increase their stake's proportion beyond the threshold of 1/3 without incurring slashing afterwards.

3) More than one third of Byzantine validators: One may ask, why would Byzantine validators aim at going over the 1/3 threshold? Indeed, we just shown that Safety can be broken regardless of β_0; is it not the ultimate goal of Byzantine validators? It is not obvious to determine what behaviour will harm the blockchain the most. We briefly discuss the impact Byzantine validators can have when they go over the 1/3 threshold in subsection V-C. We now examine what are the necessary conditions on β_0 and p_0 that permit Byzantine validator's stake to go over the one-third threshold.

The key ratio that translates into what we are looking for is the proportion of Byzantine validator's stake $\beta(t, p_0, \beta_0)$ over time:

$$\frac{\beta_0 e^{-3t^2/2^{28}}}{p_0(1 - \beta_0) + (1 - p)(1 - \beta_0)e^{-t^2/2^{25}} + \beta_0 e^{-3t^2/2^{28}}} \quad (11)$$

As expected at time $t = 0$, $\beta(0, p_0, \beta_0) = \beta_0$. Now, let us investigate when this ratio is above the threshold of 1/3, i.e.:

$$\beta(t, p_0, \beta_0) \geq 1/3 \quad (12)$$

The main difference with the previous scenario is that Byzantine validators seek to go over the 1/3 threshold, not to finalize quickly. This means that even after the proportion of honest active validators' stake and semi-active Byzantine validators' stake represent more than two-third of the stake on the branch, they do not finalize. Byzantine validators could finalize by staying active two epochs in a row, yet they do not do so in order to reach a higher stake proportion.

We construct a set containing the couples (p_0, β_0) that can lead to β to go over 1/3 (Equation 12). To do so, we take the point reached by the ratio when the validators deemed inactive are ejected. This point gives the highest value reachable[11] for a particular (p_0, β_0). For an intuition as to why this is the case, Figure 2 let us visualize that the biggest gap between semi-active Byzantine stake and honest inactive stake is at the moment of expulsion of the honest inactive validators. We have seen that inactive validators are ejected from the chain after 4685 epochs. We can thus evaluate the maximum ratio reachable $\beta_{\max}$ at time $t = 4685$ when the inactive validators gets ejected:

$$\beta_{\max}(p_0, \beta_0) = \frac{\beta_0 e^{-3 \times (4685)^2/2^{28}}}{p_0(1 - \beta_0) + \beta_0 e^{-3 \times (4685)^2/2^{28}}}. \quad (13)$$

When this ratio is greater than 1/3, Byzantine validators have reached their goal. We show with Figure 7 that Byzantine validators can actually go beyond the threshold of 1/3 on both branches simultaneously. The lower bound β_0 before this becomes possible is for $p_0 = 0.5$ when $\beta_0 = 1/(1 + 4e^{-3 \times (4685)^2/2^{28}}) = 0.2421$.

When the initial proportion of Byzantine validators is at least 0.2421, their proportion can eventually increase up to more than 1/3 of validators on both branches, exceeding the critical Safety threshold of voting power in each branch.

Having explored scenarios in which protocol vulnerabilities manifest exclusively before GST, we now focus on potential threats posed by Byzantine validators after GST. Given the acknowledged impact of the *Probabilistic Bouncing Attack* on Liveness [2], our study extends to take the inactivity leak into account.

C. Revisiting the Probabilistic Bouncing attack

This subsection revisits the Probabilistic Bouncing attack [2] showing that Byzantine validators could exceed the Safety threshold even during the synchronous period. Contrary to the previous scenarios, this one starts in the asynchronous period

---

[11]There exist more values that can lead to go over one-third when considering a special corner case. If the Byzantine validators strategically finalize just before the expulsion of honest inactive validators, the decrease in inactivity penalties might not occur quickly enough to prevent the ejection of honest inactive validators. In this particular scenario, Byzantine validators could potentially eject honest inactive participants while incurring fewer penalties themselves. This subtlety underscores the intricate dynamics at play during the inactivity leak.
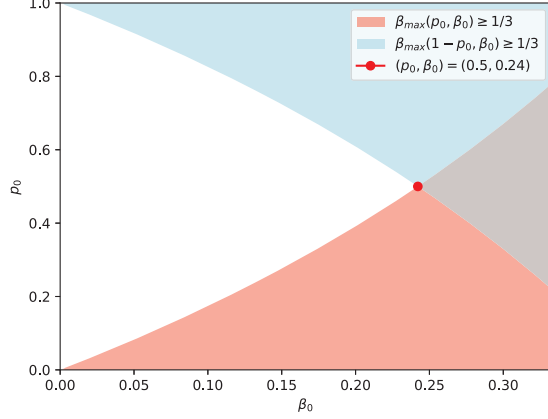
Fig. 7. Pairs (p_0, β_0) such that $\beta_{\max}(p_0, \beta_0) \geq 1/3$. This figure gives a lower bound for which (p_0, β_0) can result in the proportion of Byzantine validators exceeding 1/3 on both branches.

but unveils in the synchronous period. This demonstrates that the inactivity leak poses significant challenges even within the synchronous period, revealing its broader implication for blockchain security.

While analyzing the probabilistic bouncing attack [2], the authors did not take into account the penalties, here, we fill this gap.

Let us note, that there is no problem with conflicting finalization as the attack is progressing after GST, in the synchronous period. In synchronous time, there is not enough delay for honest validators to miss a finalization on another branch. There would need to be more than two-third of the active stakes owned by Byzantine validators to break Safety in the synchronous period.

We briefly discussed the differences in gravity between conflicting finalization and having more than 1/3 of the stake owned by Byzantine validators. We left the actual comparison and the in-depth analysis of the gravity of going beyond the infamous threshold as a future work.

We primarily focus on identifying specific scenarios that would disrupt the network. Thus, we give a detailed explanation of a scenario that could lead to Byzantine validators breaking the 1/3 threshold even during synchronous period (after GST).

Let us remind how the attack takes place.

a) Probabilistic Bouncing Attack [2]: The attack can be summarized as follows: (1) A favorable setup that partitions honest validators into two different views of the blockchain occurs. (2) At each epoch, Byzantine validators withhold their messages from honest validators, releasing them at the opportune time to make some honest validators change their view. (3) This attack continues as long as at least one Byzantine validators is proposer in the jth first slots of the epoch, where j is a parameter of the protocol. The probability of the attack to continue for k epochs with a proportion of $(1 - \beta_0)$ honest

validators is $(1 - (1 - \beta_0)^j)^k$.

We start by analyzing the outcome of a fork where a proportion p_0 of the *honest* validators start on chain A and $1 - p_0$ of the honest validators start on chain B.

We consider how would unfold a *Probabilistic Bouncing Attack* taking the inactivity leak into account. A probabilistic bouncing attack lasting more than 4 epochs will necessarily cause an inactivity leak. Knowing this, we analyze the stakes of honest and Byzantine validators in this setting.

For this attack to continue, at each epoch, Byzantine validators cast their vote with a different chain as their candidate chain. They are active on each chain alternatively. Due to their inactivity every 2 epochs, they will get ejected of the chain after a total of 7653 epochs (4 weeks and 6 days). Byzantine validators are active on each chain to ensure that justification only happens every two epoch, preventing finalization to occur.

For this attack to continue indefinitely, Byzantine validators must ensure honest validators are split on the two branches according to two conditions. A condition (a) that ensures that the honest validators are not enough to justify a chain on their own, and (b) that Byzantine validators can justify it afterwards with their withheld votes. It means that (a) p_0 must not represent more than 2/3 of the stake, and (b) the proportions p_0 of honest validators and β_0 of Byzantine validators must represent more than two-thirds of the total stake. The two necessary conditions are that (a) $p_0(1 - \beta_0) < 2/3$ and (b) $p_0(1 - \beta_0) + \beta_0 > 2/3$. For the attack to function, we get that:

$$\frac{2 - 3\beta_0}{3(1 - \beta_0)} < p_0 < \frac{2}{3(1 - \beta_0)}. \tag{14}$$

We can see that the closer β_0 is to 0, the closer p_0 has to be from 2/3. This is to be expected as otherwise the Byzantine validators would be unable to justify the checkpoint with withheld votes.

An illustration of an ongoing attack with the probability for honest validators to be on one chain or the other is depicted in Figure 8. We can see that at each epoch, a proportion p_0 is on one branch, whereas a proportion $1 - p_0$ is on the other.
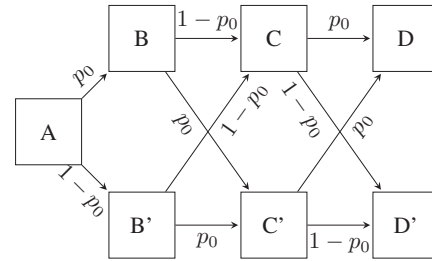

Fig. 8. This figure represents with a Markov chain the probability of an honest validator to change branch or not every epoch. During the attack, the Byzantine validators make sure that a proportion p_0 of honest validators are on one branch such that they can justify this branch later with their withheld votes (Equation 14).

Calculus: We are interested in the evolution of the proportion of Byzantine validators' stake β during the attack. To look at the proportion, we analyze the evolution of the

inactivity score over time for an honest validator randomly placed at each epoch. Looking at Figure 8, we see that after two epochs, there is a probability $p_0(1 - p_0)$ to have been on branch B for the two epochs, or on branch A for the two epochs. The probability to have been on both branches regardless of the order is $p_0^2 + (1 - p_0)^2$. From the point of view of a chain, the validators will be deemed inactive if they are active on the other chain. The probability of the inactivity score evolution after two epochs is the following:

$$\begin{cases} +8: & p_0(1 - p_0) \\ +3: & p_0^2 + (1 - p_0)^2 \\ -2: & p_0(1 - p_0) \end{cases} \quad (15)$$

We can notice that the time-dependent probability of the inactivity score is the convolution of two random walks. The first random walk moves +4 with probability p_0 and -1 with probability $(1 - p_0)$. The second is the opposite, it moves +4 with probability $(1 - p_0)$ and -1 with probability p_0.

We place ourselves in the continuous case to be able to continue our analysis and find the stake of validators with the inactivity score distribution over time. To do so we use the fact that a random walk follows a Gaussian distribution when time is big using the central limit theorem. The expectation of the two random walks are $(5p_0 - 4t)$ and $(1 - 5p_0)t$, respectively, with both having a standard deviation of $25p_0(1-p_0)$. We disregard here the fact that the actual inactivity score is bounded by zero for analytical tractability. Allowing for negative values in the inactivity score can result in a reward instead of a penalty, which leads to a scenario conservatively estimating the loss of stake. The convolution of these two random walks is the probability of the inactivity score I:

$$\phi(I, t) = \frac{1}{\sqrt{4\pi Dt}} \exp\left(-\frac{(I - Vt)^2}{4Dt}\right). \quad (16)$$

With $D = 25p_0(1 - p_0)$ and $V = 3/2$. It now remains to find the distribution function of the stake s. We rewrite here the differential equation of the stake depending on I previously described in Equation 17:

$$\frac{ds}{dt} = -\frac{I(t)s}{2^{26}}. \quad (17)$$

Due to space constraints, we omit the detailed process for finding the distribution of s. The underlying intuition is that we can return to a known problem with a derivative being equal to a Brownian motion. By integrating the Brownian motion, we find our solution. This lead to the distribution of s for any given time:

$$P(s, t) = \frac{2^{26}}{s\sqrt{\frac{4}{3}\pi Dt^3}} \exp\left(-\frac{(2^{26}\ln(s/32) + Vt^2/2)^2}{\frac{4}{3}Dt^3}\right). \quad (18)$$

The stake follows a log normal distribution for which the cumulative function is:

$$F(s, t) = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left[\frac{2^{26}\ln(s/32) + Vt^2/2}{\sqrt{\frac{4}{3}Dt^3}}\right] \quad (19)$$

Currently, the probability P does not reflect the actual stake according to time since validators get ejected at 16.75 ETH and are stuck at 32 ETH. To emulate this mechanism, since we know the cumulative distribution function we can compute the new probability law $\mathcal{P}$:

$$\mathcal{P}(x, t) = \begin{cases} F(a, t) & \text{if } x = 0 \\ P(x, t) & \text{if } a < x < b \\ 1 - F(b, t) & \text{if } x = b \end{cases} \quad (20)$$

With $a = 16.75$ and $b = 32$. This new probability law takes into account the fact that if the stake is lower than 16.75 ETH it becomes 0, and it is capped at 32 ETH. The explicit expression of $\mathcal{P}$ reads:

$$\begin{aligned} \mathcal{P}(x, t) = &\delta(x) \cdot F(a, t) + \delta(x - b) \cdot (1 - F(b, t)) \\ &+ [H(x - a) \times H(b - x)] \cdot P(x, t), \end{aligned} \quad (21)$$

where δ is the Dirac distribution, and H the Heaviside function. Figure 9 shows a visual representation of function $\mathcal{P}$.
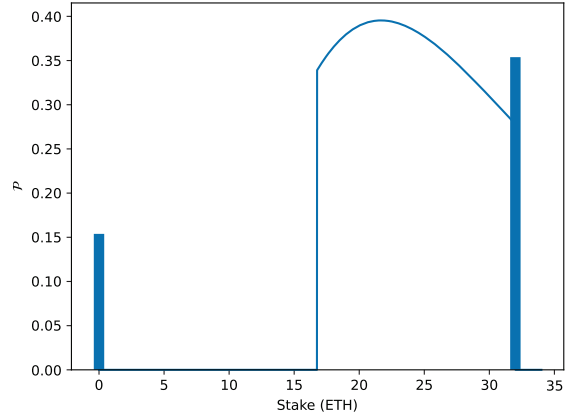

Fig. 9. This is a representation distribution of $\mathcal{P}$ at $t = 4024$ with exaggerated standard deviation to give a better intuition of the distribution behavior.

The associated cumulative distribution function $\mathcal{F}$ of $\mathcal{P}$ is :

$$\begin{aligned} \mathcal{F}(x, t) &= \int_0^x \mathcal{P}(s, t)ds \\ &= F(a, t) + H(x - a)[F(x, t) - F(a, t)] \\ &+ H(x - b)[1 - F(x, t)]. \end{aligned} \quad (22)$$

With this we can evaluate the ratio of Byzantine validators and determine with what probability it will go beyond 1/3. We denote by $s_B(t)$ the stake of a Byzantine validators and $s_H(t)$ the stake of an honest validator. We are looking for the probability such that

$$\beta(t) = \frac{\beta_0 s_B(t)}{\beta_0 s_B(t) + (1 - \beta_0)s_H(t)} > \frac{1}{3}, \quad (23)$$

depending on the probability of s_H that we now know. This translates in:

$$\mathcal{F}\left(\frac{2\beta_0}{1 - \beta_0}s_B(t), t\right), \quad (24)$$

where $s_{\mathrm{B}}(t)$ the stake of a Byzantine validator follows the stake of a semi-active validator.

We give a representation of Equation 24 for several values of β_0 with $p = 0.5$ (p_0 does not have much impact on the curve as it just changes the variance slightly) in Figure 10.

The figure gives insight as to how the proximity of β_0 to 1/3 can be harmful. The explication of this phenomenon is due to the mean of the log-normal distribution being equivalent to s_{B} when t is not too big. Looking at Equation 24 we can see that if $\beta_0 = 1/3$ then we are looking at $\mathcal{F}(s_{\mathrm{B}}(t), t)$ which explains why its probability is 0.5.

The probability rises abruptly right before the expulsion of Byzantine validators, however, it is unlikely the probabilistic bouncing attack would last that long. As an estimation, we can use the probability mentioned in [2] for an upper bound on the probability and to reach epoch 7000: $(1 - (1 - \beta_0)^8)^{7000}$ is equal to 1.01×10^{-121} for $\beta_0 = 1/3$. This negates all strategies of Byzantine validators that would need the probabilistic bouncing attack to last that long.

However, as Figure 10 shows, with β_0 nearing 1/3, Byzantine validators realistically have a high probability of quickly exceeding 1/3 of the stake, especially when considering the significant factor of the attack occurring on two branches. Meaning that if a validator is active during an epoch on one branch, it is inactive on the other. Hence, the probability can be doubled for each curve. We can comprehend this by considering the case of $\beta_0 = 1/3$: after two epochs, the Byzantine validators have been active on each branch once. If one branch has more validators that have been active on it for two epochs, the other branch will have honest validators incurring, on average, more penalties than the Byzantine validators. On this latter branch, the Byzantine stake will represent more than one-third of the total stake.

These results imply that, theoretically, within the synchronous period and with a proportion of Byzantine stake sufficiently close to 1/3 as well as a favorable initial setup, the probabilistic bouncing attack can pose a threat to the blockchain by allowing Byzantine validators to exceed the safety threshold of 1/3.

VI. RELATED WORK

While mechanisms similar to Ethereum's inactivity leak – punishing the lack of validator activity – exist elsewhere (e.g., [12], [13]), to the best of our knowledge, there has not been an analysis of the risk associated with potentially draining honest stake in a Byzantine-prone environment.

Initial efforts were made to intertwine the study of incentives with considerations of Liveness and Safety properties of the Ethereum protocol [18]. However, this early exploration discussed a preliminary version of the protocol [11] and did not include an analysis of the inactivity leak. The most recent version of the protocol by its founder [16] does not mention this mechanism. The inactivity leak still lacks a detailed examination, our work aims at filling this gap.

An investigation linking attestation penalties with the actions of Byzantine validators is presented in [19]. This work
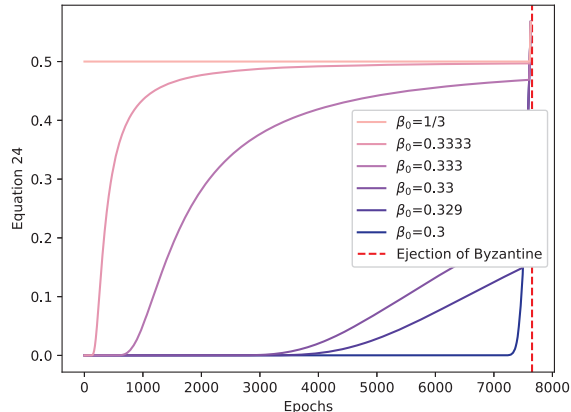

Fig. 10. We represent here Equation 24 according to time with various β_0.

demonstrates how Byzantine validators can maliciously cause penalties for honest validators. Although similar in the enterprise, we differ since we focus on penalties predominant during the inactivity leak, i.e., the inactivity penalties and slashing, since during this period attestation penalties tend to be less significant.

On another line of research, some works study consensus protocols through the lens of game-theory (e.g., [20]–[27]), where all agents are rational, i.e, they behave strategically and have a maximization objective. This work differs from them on many aspects, first, we study the Ethereum PoS protocol, and second, instead of considering rational agents, we analyze the impact classical agents considered in the distributed computing literature (correct and Byzantine) have on the Safety properties of the protocol under these incentive mechanisms.

VII. DISCUSSION & CONCLUSION

This paper presents the first theoretical analysis of the inactivity leak, designed to restore finalization during catastrophic network failure. We point out situations where *Byzantine actions expedite the loss of Safety, either with conflicting finalization or by increasing the Byzantine proportion over the one-third Safety threshold*. Interestingly, we showcase the possibility for Byzantine validators to exceed the one-third Safety threshold even during synchronous periods.

Our findings underscore the critical role of penalty mechanisms in BFT analysis. By shedding light on potential issues in the protocol design, we offer insights for future improvement and give tools to investigate them.

REFERENCES

[1] J. Neu, E. N. Tas, and D. Tse, "Ebb-and-flow protocols: A resolution of the availability-finality dilemma," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 446–465.

[2] U. Pavloff, Y. Amoussou-Guenou, and S. Tucci Piergiovanni, "Ethereum proof-of-stake under scrutiny," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC 2023, Tallinn, Estonia, March 27-31, 2023*. ACM, 2023, pp. 212–221.

[3] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*. USENIX Association, 1999, pp. 173–186.

[4] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 10401. Springer, 2017, pp. 357–388.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[6] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[7] L. Astefanoaei, P. Chambart, A. D. Pozzo, T. Rieutord, S. Tucci Piergiovanni, and E. Zalinescu, "Tenderbake - A solution to dynamic repeated consensus for blockchains," in *4th International Symposium on Foundations and Applications of Blockchain 2021, FAB 2021, May 7, 2021, University of California, Davis, California, USA (Virtual Conference)*, ser. OASIcs, vol. 92, 2021, pp. 1:1–1:23.

[8] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus," *CoRR*, vol. abs/1807.04938, 2018.

[9] E. A. Brewer, "A certain freedom: thoughts on the CAP theorem," in *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, PODC 2010, Zurich, Switzerland, July 25-28, 2010*. ACM, 2010, p. 335.

[10] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002.

[11] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *CoRR*, vol. abs/1710.09437, 2017.

[12] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White paper*, vol. 21, no. 2327, p. 4662, 2016.

[13] L. Goodman, "Tezos—a self-amending crypto-ledger white paper," *URL: https://www. tezos. com/static/papers/white paper.pdf*, vol. 4, pp. 1432–1465, 2014.

[14] C. Dwork, N. A. Lynch, and L. J. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, pp. 288–323, 1988.

[15] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*. USENIX Association, 1999, pp. 173–186.

[16] V. Buterin, D. Hernandez, T. Kamphefner, K. Pham, Z. Qiao, D. Ryan, J. Sin, Y. Wang, and Y. X. Zhang, "Combining ghost and casper," *arXiv preprint arXiv:2003.03052*, 2020.

[17] B. Edgington, *A technical handbook on Ethereum's move to proof of stake and beyond*. ETH2 Book, 2023-.

[18] V. Buterin, D. Reijsbergen, S. Leonardos, and G. Piliouras, "Incentives in ethereum's hybrid casper protocol," *Int. J. Netw. Manag.*, vol. 30, no. 5, 2020.

[19] M. Zhang, R. Li, and S. Duan, "Max attestation matters: Making honest parties lose their incentives in ethereum pos," *IACR Cryptol. ePrint Arch.*, p. 1622, 2023.

[20] Y. Amoussou-Guenou, B. Biais, M. Potop-Butucaru, and S. Tucci Piergiovanni, "Rational vs byzantine players in consensus-based blockchains," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, 2020, pp. 43–51.

[21] Y. Amoussou-Guenou, B. Biais, M. Potop-Butucaru, and S. Tucci-Piergiovanni, "Committee-Based Blockchains as Games between Opportunistic Players and Adversaries," *The Review of Financial Studies*, vol. 37, no. 2, pp. 409–443, 06 2023.

[22] B. Biais, C. Bisiere, M. Bouvard, and C. Casamatta, "The blockchain folk theorem," *The Review of Financial Studies*, vol. 32, no. 5, pp. 1662–1715, 2019.

[23] I. Eyal and E. G. Sirer, "Majority is not enough: bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[24] M. Fooladgar, M. H. Manshaei, M. Jadliwala, and M. A. Rahman, "On incentive compatible role-based reward distribution in algorand," in *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020*. IEEE, 2020, pp. 452–463.

[25] J. Y. Halpern and X. Vilaça, "Rational consensus: Extended abstract," in *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, G. Giakkoupis, Ed. ACM, 2016, pp. 137–146.

[26] M. H. Manshaei, M. Jadliwala, A. Maiti, and M. Fooladgar, "A game-theoretic analysis of shard-based permissionless blockchains," *IEEE Access*, vol. 6, pp. 78 100–78 112, 2018.

[27] T. Roughgarden, "Transaction fee mechanism design for the ethereum blockchain: An economic analysis of EIP-1559," *CoRR*, vol. abs/2012.00854, 2020.