PoSAT: Proof-of-Work Availability and Unpredictability, without the Work

Soubhik Deb[‡], Sreeram Kannan[‡], David Tse[⋆]
Email: soubhik@uw.edu, ksreeram@uw.edu, dntse@stanford.edu

[‡]University of Washington,
[⋆]Stanford University

Abstract. An important feature of Proof-of-Work (PoW) blockchains is full dynamic availability, allowing miners to go online and offline while requiring only 50% of the online miners to be honest. Existing Proof-of-stake (PoS), Proof-of-Space and related protocols are able to achieve this property only partially, either requiring the additional assumption that adversary nodes are online from the beginning and no new adversary nodes come online afterwards, or use additional trust assumptions for newly joining nodes. We propose a new PoS protocol PoSAT which can provably achieve dynamic availability fully without any additional assumptions. The protocol is based on the longest chain and uses a Verifiable Delay Function for the block proposal lottery to provide an arrow of time. The security analysis of the protocol draws on the recently proposed technique of Nakamoto blocks as well as the theory of branching random walks. An additional feature of PoSAT is the complete unpredictability of who will get to propose a block next, even by the winner itself. This unpredictability is at the same level of PoW protocols, and is stronger than that of existing PoS protocols using Verifiable Random Functions.

1 Introduction

1.1 Dynamic Availability

Nakamoto's invention of Bitcoin [25] in 2008 brought in the novel concept of a permissionless Proof-of-Work (PoW) consensus protocol. Following the longest chain protocol, a block can be proposed and appended to the tip of the blockchain if the miner is successful in solving the hash puzzle. The Bitcoin protocol has several interesting features as a consensus protocol. An important one is *dynamic availability*. Bitcoin can handle an uncertain and dynamic varying level of consensus participation in terms of mining power. Miners can join and leave as desired without any registration requirement. This is in contrast to most classical Byzantine Fault Tolerant (BFT) consensus protocols, which assumes a fixed and known number of consensus nodes. Indeed, Bitcoin has been continuously available since the beginning, a period over which the hashrate has varied over a range of 14 orders of magnitude. Bitcoin has been proven to be secure as long

as the attacker has less than 50% of the online hash power (the static power case is considered in [17, 25, 26] and variable hashing power case is considered in [18, 19]).

Recently proof-of-stake (PoS) protocols have emerged as an energy-efficient alternative to PoW. Instead of solving a difficult hash puzzle, nodes participate in a lottery to win the right to append a block to the blockchain, with the probability of winning proportional to a node's stake in the total pool. This replaces the resource intense mining process of PoW, while ensuring fair chances to contribute and claim rewards.

There are broadly two classes of PoS protocols: those derived from classical BFT protocols and those inspired by Nakamoto's longest chain protocol. Attempts at blockchain design via the BFT approach include Algorand [9, 20], Tendermint [7] and Hotstuff [35]. Motivated and inspired by Nakamoto longest chain protocol are the PoS designs of Snow White [4] and the Ouroboros family of protocols [2, 11, 21]. One feature that distinguish the PoS longest chain protocols from the BFT protocols is that they inherit the dynamic availability of Bitcoin: the chain always grows regardless of the number of nodes online. But do these PoS longest chain protocols provide the same level of security guarantee as PoW Bitcoin in the dynamic setting?

1.2 Static vs Dynamic Adversary

Two particular papers focus on the problem of dynamic availability in PoS protocols: the sleepy model of consensus [28] and Ouroboros Genesis [2]. In both papers, it was proved that their protocols are secure if less than 50% of the online nodes are adversary. This condition is the same as the security guarantee in PoW Bitcoin, but there is an additional assumption: *all adversary nodes are always online starting from genesis and no new adversary nodes can join.* While this static adversary assumption seems reasonable (why would an adversary go to sleep?), in reality this can be a very restrictive condition. In the context of Bitcoin, this assumption would be analogous to the statement that the hash power of the adversary is fixed in the past decade (while the total hashing power increased 14 orders of magnitude!) More generally, in public blockchains, PoW or PoS, no node is likely to be adversarial during the launch of a new blockchain token - adversaries only begin to emerge later during the lifecycle.

The static adversary assumption underlying these PoS protocols is not superfluous but is in fact *necessary* for their security. Suppose for the 1^{st} year of the existence of the PoS-based blockchain, only 10% of the total stake is online. Out of this, consider that all nodes are honest. Now, at the beginning of the 2^{nd} year, all 100% of the stake is online out of which 20% is held by adversary. At any point of time, the fraction of online stake held by honest nodes is greater than 0.8. However, both Sleepy and Genesis are not secure since the adversary can use its 20% stake to immediately participate in all past lotteries to win blocks all the way back to the genesis and then grow a chain *instantaneously* from the genesis to surpass the current longest chain (Figure 1(a)). Thus, due to this "costless simulation", newly joined adversary nodes not only increase the current online

adversary stake, but effectively increase past online adversary stake as well. See Appendix A.3 for further details on how costless simulation renders both sleepy model of consensus and Ouroboros Genesis vulnerable to attacks. In contrast, PoW does not suffer from the same issue because it would take a long time to grow such a chain from the past and that chain will always be behind the current longest chain. Thus, PoW provides an *arrow of time*, meaning nodes cannot "go back in time" to mine blocks for the times at which they were not online. This property is key in endowing PoW protocols with the ability to tolerate fully dynamic adversaries wherein both honest nodes and adversary can have varying participation (Figure 1(b)).

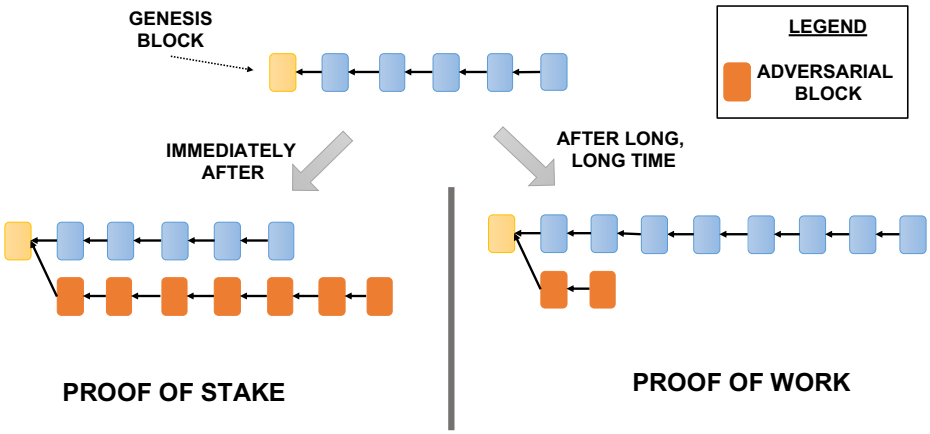


Fig. 1: (a) Newly joined nodes in existing PoS protocols can grow a chain from genesis instantaneously. (b) Newly joined miners in PoW protocol takes a long time to grow such a chain and is always behind.

We point out that some protocols including Ouroboros Praos [11] and Snowhite [4] require that nodes discard chains that fork off too much from the present chain. This feature was introduced to handle nodes with expired stake (or nodes that can perform key grinding) taking over the longest chain. While they did not specifically consider the dynamic adversary issue we highlighted, relying on previous checkpoints can potentially solve the aforementioned security threat. However, as was eloquently argued in Ouroboros Genesis [2], these checkpoints are unavailable to offline clients and newly joining nodes require advice from a trusted party (or a group inside which a majority is trusted). This trust assumption is too onerous to satisfy in practice and is not required in PoW. Ouroboros Genesis was designed to require no trusted joining assumption while being secure to long-range and key-grinding attacks. However, they are not secure against dynamic participation by the adversary: they are vulnerable to the aforementioned attack. This opens the following question:

Is there a fully dynamically available PoS protocol which has full PoW security guarantee, without additional trust assumptions?

1.3 PoSAT achieves PoW dynamic availability

We answer the aforementioned question in the affirmative. Given that arrow-of-time is a central property of PoW protocols, we design a new PoS protocol, PoS with Arrow-of-Time (PoSAT), also having this property using randomness generated from Verifiable Delay Functions (VDF). VDFs are built on top of iteratively sequential functions, i.e., functions that are only computable sequentially: $f^{\ell}(x) = f \circ f \circ ... \circ f(x)$, along with the ability to provide a short and easily verifiable proof that the computed output is correct. Examples of such functions include (repeated) squaring in a finite group of unknown order [8,31], i.e,, $f(x) = 2^x$ and (repeated) application of secure hash function (SHA-256) [23], i.e,, $f(x) = \text{HASH}(x)$. While VDFs have been designed as a way for proving the passage of a certain amount of time (assuming a bounded CPU speed), it has been recently shown that these functions can also be used to generate an unpredictable randomness beacon [14]. Thus, running the iteration till the random time L when $\text{RANDVDF}(x) = f^L(x) < \tau$ is within a certain threshold will result in L being a geometric random variable. We will incorporate this randomized VDF functionality to create an arrow-of-time in our protocol.

The basic idea of our protocol is to mimic the PoW lottery closely: instead of using the solution of a Hash puzzle based on the parent block's hash as proof of work, we instead use the randomized VDF computed based on the parent block randomness and the coin's public key as the proof of stake lottery. In a PoW system, we are required to find a string called "`nonce`" such that $\text{HASH}(\texttt{block}, \texttt{nonce}) < \tau$, a hash-threshold. Instead in our PoS system, we require $\text{RANDVDF}(\texttt{randSource}, pk, \texttt{slot}) < \tau$, where `randSource` is the randomness from the parent block, pk is the public key associated with the mining coin and `slot` represents the number of iterations of the RANDVDF since genesis. There are four differences, the first three are common in existing PoS systems: (1) we use "`randSource`" instead of "`block`" in order to prevent grinding attacks on the content in the PoS system, (2) we use the public-key "pk" of staking coin instead of PoW "`nonce`" to simulate a PoS lottery, (3) we use "`slot`" for ensuring time-ordering, (4) instead of using a HASH, we use the RANDVDF, which requires sequential function evaluation thus creating an "arrow of time".

The first two aspects are common to many PoS protocols and is most similar to an earlier PoS protocol [15], however, crucially we use the RANDVDF function instead of a Verifiable random function (VRF) and a time parameter inside the argument used in that protocol. This change allows for full dynamic availability: if adversaries join late, they cannot produce a costless simulation of the time that they were not online and build a chain from genesis instantaneously. It will take the adversary time to grow this chain (due to the sequential nature of the RANDVDF), by which time, the honest chain would have grown and the adversary will be unable to catch up. Thus, PoSAT behaves more like PoW (Figure 1(b)) rather than existing PoS based on VRF's (Figure 1(a)). We show

that this protocol achieves full dynamic availability: if $\lambda_h(t)$ denotes the honest stake online at t, $\lambda_a(t)$ denotes the online adversarial stake at time t, it is secure as long as

$$\lambda_h(t) > e\lambda_a(t) \qquad \text{for all } t, \tag{1}$$

where e is Euler's number $2.7182\ldots$.

We observe that the security of this protocol requires a stronger condition than PoW protocols. The reason for this is that an adversary can potentially do parallel evaluation of VDF on *all* possible blocks. Since the randomness in each of the blocks is independent from each other, the adversary has many random chances to increase the chain growth rate to out-compete the honest tree. This is a consequence of the nothing-at-stake phenomenon: the same stake can be used to grind on the many blocks. The factor e is the resulting amplification factor for the adversary growth rate. This is avoided in PoW protocols due to the conservation of work inherent in PoW which requires the adversary to split its total computational power among such blocks.

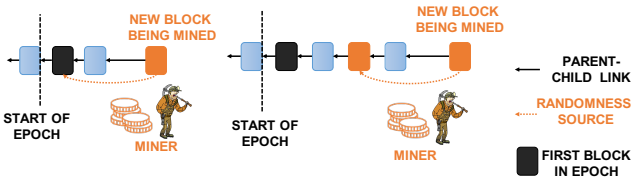


Fig. 2: Left: A node uses randomness from the first block of the epoch. Right: Since a node already won a block in the period, it uses that block's randomness.

We solve this problem in PoSAT by reducing the rate at which the block randomness is updated and hence reducing the block randomness grinding opportunities of the adversary. Instead of updating the block randomness at every level of the blocktree, we only update it once every c levels (called an epoch). The larger the value of the parameter c, the slower the block randomness is updated. The common source of randomness used to run the VDF lottery remains the same for c blocks starting from the genesis and is updated only when (a) the current block to be generated is at a depth that is a multiple of c, or (b) the coin used for the lottery is successful within the epoch of size c. The latter condition is necessary to create further independent winning opportunities for the node within the period c once a slot is obtained with that coin. This is illustrated in Figure 2. For $c = 1$, this corresponds to the protocol discussed earlier.

The following security theorem is proved about PoSAT for general c, giving a condition for security (liveness and persistence) under *all* possible attacks .

***Theorem** 1* (Informal). PoSAT with parameter c is secure as long as

$$\frac{\lambda_h^c(t)}{1 + \lambda_{\max}\Delta} > \phi_c\lambda_a(t) \qquad \text{for all } t, \tag{2}$$

where $\lambda_h^c(t)$ is the honest stake this is online at time t and has been online since at least $t-\Theta(c)$, Δ is the network delay between honest nodes, $\lambda_{\max}$ is a constant such that $\lambda_h^c(t) \leq \lambda_{\max}$ for all $t > 0$, ϕ_c is a constant, dependent on c, given in (21). $\phi_1 = e$ and $\phi_c \to 1$ as $c \to \infty$.

We remark that in our PoS protocol, we have a known upper bound on the rate of mining blocks (by assuming that the entire stake is online). We can use this information to set $1 + \lambda_{\max}\Delta$ as close to 1 as desired by simply setting the mining threshold appropriately. Furthermore, by setting c large, $\phi_c \approx 1$ and thus PoSAT can achieve the same security threshold as PoW under full dynamic availability. The constant ϕ_c is the amplification of the adversarial chain growth rate due to nothing-at-stake, which we calculate using the theory of branching random walks [32]. The right hand side of (2) can therefore be interpreted as the growth rate of a private adversary tree with the adversary mining on every block. Hence, condition (2) can be interpreted as the condition that the private Nakamoto attack [25] does not succeed. However, Theorem 1 is a *security theorem*, i.e. it gives a condition under which the protocol is secure under *all* possible attacks. Hence what Theorem 1 says is therefore that among all possible attacks on PoSAT, the private attack is the *worst* attack. We prove this by using the technique of blocktree partitioning and Nakamoto blocks, introduced in [12], which reduce all attacks to a union of private attacks.

We note that large c is beneficial from the point of view of getting a tight security threshold. However, we do require c to be finite (unlike other protocols like Ouroboros that continue to work under c being infinite). This is because the latency to confirm a transaction increases linearly in c (see Section 4). Furthermore, an honest node on coming online has to wait until encountering the next epoch beginning before it can participate in proposing blocks and the worst-case waiting time increases linearly with c. We note that the adversary cannot use the stored blocks in the next epoch, thus having a bounded reserve of blocks. The total number of blocks stored up by an adversary potentially increases linearly in the epoch size, thus requiring the confirmation depth and thus latency to be larger than $\Theta(c)$. By carefully bounding this enhanced power of the adversary, for any finite c, we show that PoSAT is secure.

Assuming $\lambda_{\max}\Delta$ to be small and c large, the comparison of PoSAT with other protocols is shown in Table 1.3. Here we use Λ_a to be the largest adversary fraction of the total stake online at any time during the execution ($\Lambda_a = \sup_t \lambda_a(t)$). Protocols whose security guarantee assumes all adversary nodes are online all the time effectively assumes that $\lambda_h(t) > \Lambda_a$. Thus existing protocols have limited dynamic availability (or compromise on the potential to join late without any trusted setup).

	Sleepy / Ourboros	Snow White / Praos	Genesis	Algorand	PoSAT
Dynamic Availability	$\lambda_h(t) > \Lambda_a$	$\lambda_h(t) > \Lambda_a$	$\lambda_h(t) > \Lambda_a$	No	$\lambda_h^c(t) > \phi_c \lambda_a(t)$
Trusted-set for Late-joining	Yes	Yes	No	NA	No
Predictability	Global	Local	Local	Local	None

1.4 PoSAT has PoW Unpredictability

Another key property of PoW protocols is their ability to be unpredictable: no node (including itself) can know when a given node will be allowed to propose a block ahead of the proposal slot. We point out that PoSAT with any parameter c remains unpredictable due to the the unpredictability of the RandVDF till the threshold is actually reached. We refer the reader to Fig. 2(a) where if the randomness source is at the beginning of the epoch it is clear that the unpredictability of the randomized VDF implies unpredictability in our protocol. However, in case the miner has already created a block within the epoch (Fig. 2(b)), the randomness source is now her previous block. This can be thought of as a continuation of the iterative sequential function from the beginning of the epoch and hence it is also unpredictable as to when the function value will fall below a threshold. Thus PoSAT achieves true unpredictability, matching the PoW gold standard, where even an all-knowing adversary has no additional predictive power.

The first wave of PoS protocols such as Sleepy model of consensus [28] and Ouroboros [21] are fully predictable as they rely on mechanisms for proposer election that provide global knowledge of all proposers in an epoch ahead of time. The concept of Verifiable Random Functions (VRF), developed in [13,24], was pioneered in the blockchain context in Algorand [9,20], as well as applied in Ouroboros Praos [11] and Snow White [4]. The use of a private leader election using VRF enables no one else other than the proposer to know of the slots when it is allowed to propose blocks. However, unlike Bitcoin, the proposer itself can predict. Thus, these protocols still allow *local* predictability. The following vulnerability is caused by local predictability: a rational node may then willingly sell out his slot to an adversary. In Ouroboros Praos, such an all-knowing adversary needs to corrupt only 1 user at a time (the proposer) adaptively in order to do a double-spend attack. He will first let the chain build for some time to confirm a transaction, and then get the bribed proposers one at a time to build a competing chain. Algorand is more resilient, but even there, in each step of the BFT algorithm, a different committee of nodes is selected using a VRF based sortition algorithm. These nodes are locally predictable as soon as the previous block is confirmed by the BFT - and thus an all-knowing adversary only needs to corrupt a third of a committee. Assuming each committee is comprised of K

nodes (K being a constant), the adversary only needs to corrupt $\frac{K}{3N}$ fraction of the nodes. Refer to Appendix A.4 for further details.

We summarize the predictability of various protocols in Table 1.3.

1.5 Related Work

Our design is based on frequent updates of randomness to run the VDF lottery. PoS protocols that update randomness at each iteration have been utilized in practice as well as theoretically proposed [15] - they do not use VDF and have **neither** dynamic availability nor unpredictability. Furthermore, they still face nothing-at-stake attacks. In fact, the amplification factor of e we discussed earlier has been first observed in a Nakamoto private attack analysis in [15]. This analysis was subsequently extended to a full security analysis against *all* attacks in [12,33], where it was shown that the private attack is actually the worst attack. In [33], the idea of c-correlation was introduced to reduce the rate of randomness update and to reduce the severity of the nothing-at-stake attack; we borrowed this idea from them in the design of our VDF-based protocol, PoSAT.

There have been attempts to integrate VDF into the proof-of-space paradigm [10] as well as into the proof-of-stake paradigm [1], [22], all using a VRF concatenated with a VDF. But, in [10], the VDF runs for a fixed duration depending on the input and hence is predictable, and furthermore do not have security proofs for dynamic availability. In [1], the randomness beacon is not secure till the threshold of $1/2$ as claimed by the authors since it has a randomness grinding attack which can potentially expand the adverarial power by at least factor e. There are three shortcomings in [22] as compared to our paper: (1) even under static participation, they only focus on an attack where an adversary grows a private chain, (2) there is no modeling of dynamic availablility and a proof of security and (3) since the protocol focuses only on $c = 1$, they can only achieve security till threshold $1/1 + e$, not till $1/2$. We note that recent work [6] formalized that a broad class of PoS protocols suffer from either of the two vulnerabilities: (a) use recent randomness, thus being subject to nothing-at-stake attacks or (b) use old randomness, thus being subject to prediction based attacks (even when only locally predictable). We note that PoSAT with large c completely circumvents both vulnerabilities using the additional VDF primitive since it is able to use old randomness while still being fully unpredictable.

We want to point out that dynamic availability is distinct and complementary to *dynamic stake*, which implies that the set of participants and their identities in the mining is changing based on the state of the blockchain. We note that there has been much existing work addressing issues on the dynamic stake setting - for example, the s-longest chain rule in [2], whose adaptation to our setting we leave for future work. We emphasize that the dynamic availability problem is well posed even in the static stake setting (the total set of stakeholders is fixed at genesis).

1.6 Outline

The rest of the paper is structured as follows. Section 2 presents the VDF primitive we are using and the overall protocol. Section 3 presents the model. Section 4 presents the details of the security analysis.

2 Protocol

2.1 Primitives

In this section, we give an overview of VDFs and refer the reader to detailed definitions in Appendix B.

Definition 1 (from [5]). *A* VDF $V = (\textsc{Setup}, \textsc{Eval}, \textsc{Verify})$ *is a triple of algorithms as follows:*

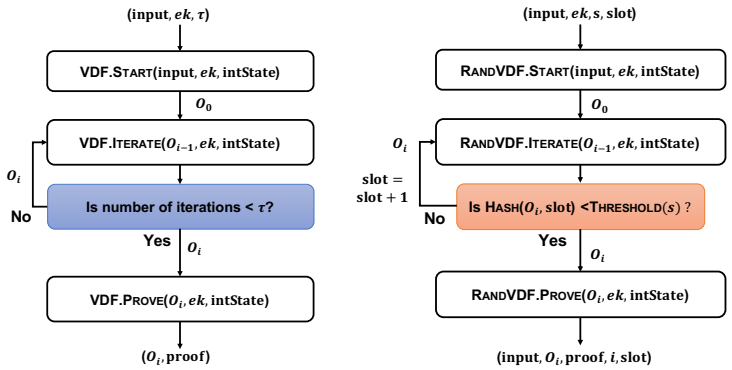
- $\textsc{Setup}(\lambda, \tau) \rightarrow \mathbf{pp} = (ek, vk)$ *is a randomized algorithm that produces an evaluation key* ek *and a verification key* vk*.*
- $\textsc{Eval}(ek, input, \tau) \rightarrow (O, proof)$ *takes an input* $\in \mathcal{X}$*, an evaluation key* ek*, number of steps* τ *and produces an output* $O \in \mathcal{Y}$ *and a (possibly empty)* $proof$*.*
- $\textsc{Verify}(vk, input, O, proof, \tau) \rightarrow Yes, No$ *is a deterministic algorithm takes an input, output, proof,* τ *and outputs* Yes *or* No*.*

VDF.Eval is usually comprised of sequential evaluation: $f^{\ell}(x) = f \circ f \circ ... \circ f(x)$ along with the ability to provide a short and easily verifiable proof. In particular, there are three separate functions VDF.Start, VDF.Iterate and VDF.Prove (the first function is used to initialize, the second one operates for the number of steps and the third one furnishes a proof). This is illustrated in Figure 3a on the left. While VDFs have been designed as a way for proving the passage of a certain amount of time, it has been recently shown that these functions can also be used to generate an unpredictable randomness beacon [14]. Thus, running the iteration till the random time L when $\textsc{RandVDF}(x) = f^{L}(x) < \tau$ generates the randomness beacon. This is our core transformation to get a randomized VDF. This is shown in Figure 3b on the right. Instead of running for a fixed number of iterations, we run the VDF iterations till it reaches a certain threshold. Our transformation is relatively general purpose and most VDFs can be used with our construction. For example, a VDF (which is based on squaring in a group of unknown order) is an ideal example for our construction [29, 34]. In the recent paper [14], for that sequential function, a new method for obtaining a short proof whose complexity does not depend (significantly) on the number of rounds is introduced - our protocol can utilize that VDF as well. They show furthermore that they obtain a continuous VDF property which implies that partial VDF computation can be continued by a different party - we do not require this additional power in our protocol.

For the RandVDF in PoSAT, as illustrated in Fig 3b, `slot` plays a similar role as the timestamps in other PoS protocols like [28]. The `slot` basically mentions the number of times the RandVDF has iterated since the genesis and when

the speed of the iteration of RandVDF is constant, `slot` is an approximation to the time elapsed since the beginning of the operation of the PoS system.

Normally, a VDF will satisfy *correctness* and *soundness*. And we require RandVDF to also satisfy correctness and soundness as defined in Appendix B.



(a) VDF.Eval$(input, ek, \tau)$ (b) RandVDF.Eval$(input, ek, s)$.

Fig. 3: VDF.Eval$(\texttt{input}, ek, \tau)$ requires the number of iterations that VDF.Iterate should run. On the other hand, RandVDF.Eval$(\texttt{input}, ek, \texttt{s}, \texttt{slot})$ requires the expected number of number of iterations RandVDF.Iterate (denoted by `s`) must run.

A key feature of VDF is that if the VDF takes T steps, then the prover should be able to complete the proof in time (nearly) proportional to T and the verifier should be able to verify the proof in (poly)-logarithmic time. This makes it feasible for any node that receives a block to quickly verify that the VDF in the header is indeed correctly computed, without expending the same effort that was expended by the prover. We refer the reader interested in a detailed analysis of these complexities to Section 6.2 in [29] for the efficiency calculation or Section 2.3 in [14].

2.2 Protocol description

The pseudocode for the PoSAT is given in Algorithm 1.

Algorithm 1 PoSAT

```
1: procedure Initialize( )                                   ▷ all variables are global
2:     blkTree ← Sync()                                      ▷ syncing with peers
3:     unCnfTx← φ                                            ▷ pool of unconfirmed txs
4:     parentBlk ← blkTree.Tip()                             ▷ tip of the longest chain in blkTree
5:     randSource ← None                                     ▷ will be updated at next epoch beginning
6:     slot ← None                                           ▷ will be updated at next epoch beginning
7:     return False
8: procedure PosLeaderElection(coin)
9:     (RandVDF.ek, RandVDF.vk),(Sign.vk, Sign.sk) ← coin.Keys()
10:    stake ← coin.Stake( SearchChainUp(parentBlk))         ▷ update the stake
11:    s ← UpdateThreshold(stake)                            ▷ update the threshold
12:    input ← randSource
13:    // Calling RandVDF.Eval
14:    (input, output, proof, randIter, slot) ← RandVDF.Eval(input, ek, s, slot)
15:    randSource ← output                                   ▷ update source of randomness
16:    state ← Hash(parentBlk)
17:    content ← ⟨unCnfTx, coin, input, randSource, proof, randIter, state, slot⟩
18:    return ⟨header, content, Sign(content,Sign.sk)⟩
19: procedure ReceiveMessage(X)                              ▷ receives messages from network
20:    if X is a valid tx then
21:        unCnfTx ← unCnfTx ∪ {X}
22:    else if IsValidBlock(X) then
23:        if parentBlk.Level() < X.Level() then
24:            ChangeMainChain(X)                            ▷ if the new chain is longer
25:            parentBlk ← X ▷ update the parent block to tip of the longest chain
26:            if X.Level() % c == 0 then
27:                randSource ← X.content.randSource
28:            else
29:                randSource ← randSource
30:            if participate == True then
31:                RandVDF.Reset()                           ▷ reset the RandVDF
32:            // Epoch beginning
33:            if (X.Level() % c == 0) & (participate == False) then
34:                slot ← X.content.slot
35:                participate = True
36: procedure IsValidBlock(X)                                ▷ returns true if a block is valid
37:    if not IsUnspent(X.content.coin) then return False
38:    if ParentBlk(X).content.slot ≥ X.content.slot then
39:        return False                                      ▷ ensuring time ordering
40:    s ← UpdateThreshold(ParentBlk(X))
41:    if Hash(X.content.{randSource,slot}) > Threshold(s) then return False
42:    // verifying the work
43:    return RandVDF.Verify(X.coin.vk, X.content.{input, randSource, proof, randIter})
44: procedure Main( )                                        ▷ main function
45:    participate = Initialize()
46:    StartThread(ReceiveMessage)                           ▷ parallel thread for receiving messages
47:    while True do
48:        if participate == True then
49:            block = PosLeaderElection(coin)
50:            SendMessage(block)                            ▷ broadcast to the whole network
```

Initialization. An honest coin n on coming online, calls INITIALIZE() where it obtains the current state of the blockchain, $\texttt{blkTree}$, by synchronizing with the peers via SYNC() and initializes global variables. However, the coin n can start participating in the leader election only after encountering the next epoch beginning, that is, when the depth of the $\texttt{blkTree}$ is a multiple of c. This is indicated by setting $\texttt{participate}_n$ to False. Observe that if the coin n is immediately allowed to participate in leader election, then, the coin n would have to initiate RANDVDF.EVAL from the $\texttt{randSource}$ contained in the block at the beginning of the current epoch. Due to the sequential computation in RANDVDF, the coin n would never be able to participate in the leader elections for proposing block at the tip of the blockchain. In parallel, the coin keeps receiving messages and processes them in RECEIVEMESSAGE(). On receiving a valid block that indicates epoch beginning, $\texttt{randSource}_n$, $\texttt{slot}_n$ and $\texttt{participate}_n$ are updated accordingly (lines 27, 33, 34) for active participation in leader election.

Leader election. The coin n records the tip of the longest chain of $\texttt{blkTree}$ in $\texttt{parentBlk}_n$ (line 25) and contests leader election for appending block to it. RANDVDF.EVAL($\texttt{input}_n$, RANDVDF.ek_n, $\texttt{s}_n$) is used to compute an unpredictable randomness beacon that imparts unpredictability to leader election. The difficulty parameter $\texttt{s}_n$ is set proportional to the current $\texttt{stake}_n$ of the coin n using UPDATETHRESHOLD($\texttt{stake}_n$) and $\texttt{randSource}_n$ is taken as $\texttt{input}_n$. RANDVDF.EVAL($\texttt{input}_n, ek_n, \texttt{s}_n, \texttt{slot}_n$) is an iterative function composed of:

- RANDVDF.START($\texttt{input}_n$, RANDVDF.ek_n, $\texttt{IntState}_n$) initializes the iteration by setting initial value of $\texttt{output}_n$ to be $\texttt{input}_n$. Note that $\texttt{IntState}_n$ is the internal state of the RANDVDF.
- RANDVDF.ITERATE($\texttt{output}_n$, RANDVDF.ek_n, $\texttt{IntState}_n$) is the iterator function that updates $\texttt{output}_n$ in each iteration. At the end of each iteration, it is checked whether HASH($\texttt{output}_n, \texttt{slot}_n$) is less than THRESHOLD($\texttt{s}_n$), which is set proportional to $\texttt{s}_n$. If $\texttt{No}$, $\texttt{slot}_n$ is incremented by 1 and current $\texttt{output}_n$ is taken as input to the next iteration. If $\texttt{Yes}$, then it means coin n has won the leader election and $\texttt{output}_n$ is passed as input to RANDVDF.PROVE(.). Observe that the number of iterations, $\texttt{randIter}_n$, that would be required to pass this threshold is unpredictable which lends to randomness beacon. Recall that $\texttt{slot}_n$ is a counter for number of iterations since genesis. In a PoS protocol, it is normally ensured that the timestamps contained in each block of a chain are ordered in ascending order. Here, in PoSAT, instead we ensure that the $\texttt{slot}$ in the blocks of a chain are ordered, irrespective of who proposed it. This is referred to as *time-ordering*. The reader can refer to Appendix A.5 and A.6 for further details on what attacks can transpire if time-ordering is not ensured. The rationale behind setting THRESHOLD($\texttt{s}_n$) proportional to $\texttt{s}_n$ is that even if the stake $\texttt{s}_n$ is sybil over multiple coins, the probability of winning leader election in at least one coin remains the same. See Appendix A.2 for detailed discussion.

– RandVDF.Prove($\texttt{output}_n$, RandVDF.ek_n, $\texttt{IntState}_n$) operates on $\texttt{output}_n$ using RandVDF.ek_n and $\texttt{IntState}_n$ to generate $\texttt{proof}_n$ that certifies the iterative computation done in the previous step.

The source of randomness $\texttt{randSource}_n$ can be updated in two ways:

– a block, proposed by another coin, at epoch beginning is received (line 27)
– if coin n wins a leader election and proposes its own block (line 15).

While computing RandVDF.Eval(.), if a block is received that updates $\texttt{parentBlk}_n$, then, RandVDF.Reset() (line 31) pauses the ongoing computation, updates $\texttt{s}_n$ and continues the computation with updated Threshold($\texttt{s}_n$). If $\texttt{randSource}_n$ is also updated, then, RandVDF.Reset() stops the ongoing computation of RandVDF.Eval(.) and calls PoSLeaderElection().

Content of the block. Once a coin is elected as a leader, all unconfirmed transactions in its buffer are added to the `content`. The `content` also includes the identity $\texttt{coin}_n$, $\texttt{input}_n$, $\texttt{randSource}_n$, $\texttt{proof}_n$, $\texttt{randIter}_n$, $\texttt{slot}_n$ from RandVDF.Eval(.). The `state` variable in the content contains the hash of parent block, which ensures that the content of the parent block cannot be altered. Finally, the header and the content is signed with the secure signature SIGN.sk_n and the block is proposed. When the block is received by other coins, they check that the time-ordering is maintained (line 38) and verify the work done by the coin n using RandVDF.Verify(.) (line 43). Note that the leader election is independent of the content of the block and content of previous blocks. This follows a standard practice in existing PoS protocols such as [2] and [28] for ensuring that a grinding attack based on enumerating the transactions won't be possible. The reader is referred to Appendix A.1 for further details. However, this allows the adversary to create multiple blocks with the same header but different content. Such copies of a block with the same header but different contents are known as a "forkable string" in [21]. We show in the section 4 that the PoSAT is secure against all such variations of attacks.

Confirmation rule. A block is confirmed if the block is $k-$deep from the tip of the longest chain. The value of k is determined by the security parameter.

3 Model

We will adopt a continuous-time model. Like the Δ-synchronous model in [26], we assume there is a bounded communication delay Δ seconds between the honest nodes (the particular value of latency of any transmission inside this bound is chosen by the adversary).

The blockchain is run on a network of N honest nodes and a set of adversary nodes. Each node holds a certain number of coins (proportional to their stake). We allow nodes to join and leave the network, thus the amount of honest/adversarial stake which is participating in the protocol varies as a function

of time. Recall that, as described in section 2, a coin coming online can only participate in the leader election after encountering the next epoch beginning. This incurs a waiting delay for the coin before it can actively participate in the evolution of the blockchain. Suppose that $\sigma(c)$ is the worst-case waiting delay, i.e., it refers to worst-case time. Based on this worst-case waiting delay, let $\lambda_h^c(t)$ be defined as the stake of the honest coins that are online at time t and has been online since at least time $t - \sigma(c)$. Also, let $\lambda_h(t)$ is defined as the stake of the honest coins that are online at time t and has encountered at least one epoch beginning. Thus, $\lambda_h(t)$ is the rate at which honest nodes win leader elections. Let $\lambda_a(t)$ be the stake controlled by the adversary. We assume these functions are fixed *a priori* deterministically, and they satisfy

$$\lambda_a(t) \leq (1-\eta)\lambda_h^c(t) \quad \forall \quad t > 0. \tag{3}$$

where $0 < \eta < 1$. Also, assume there exists constants $\lambda_{\min}, \lambda_{\max} > 0$ such that

$$\lambda_{\min} \leq \lambda_h^c(t) \leq \lambda_{\max} \quad \forall \quad t \geq 0. \tag{4}$$

The existence of $\lambda_{\max}$ is obvious since we are in a proof-of-stake system, and $\lambda_{\max}$ denotes the rate at which the leader elections are being won if every single stakeholder is online. We need to assume a minimum $\lambda_h^c(t)$ in order to guarantee that within a bounded time, a new block is created. An honest node will construct and publicly reveal the block immediately after it has won the corresponding leader election. However, an adversary can choose to not do so. By "private block", we refer to a block whose corresponding computation of RandVDF.Eval was completed by the adversary earlier than when the block was made public. Also, by "honest block proposed at time t", we mean that the computation of RandVDF.Eval was completed at time t and then the associated honest block was instantaneously constructed and publicly revealed.

The evolution of the blockchain can be modeled as a process $\{(\mathcal{T}(t), \mathcal{C}(t), \mathcal{T}^{(p)}(t), \mathcal{C}^{(p)}(t)) : t \geq 0, 1 \leq p \leq N\}$, N being the number of honest nodes, where:

- $\mathcal{T}(t)$ is a tree, and is interpreted as the *mother tree* consisting of all the blocks that are proposed by both the honest and the adversary nodes up until time t (including private blocks at the adversary).
- $\mathcal{T}^{(p)}(t)$ is an induced (public) sub-tree of the mother tree $\mathcal{T}(t)$ in the view of the p-th honest node at time t.
- $\mathcal{C}^{(p)}(t)$ is the longest chain in the tree $\mathcal{T}^{(p)}(t)$, and is interpreted as the longest chain in the local view of the p-th honest node.
- $\mathcal{C}(t)$ is the common prefix of all the local chains $\mathcal{C}^{(p)}(t)$ for $1 \leq p \leq N$.

The process evolution is as follows.

- **M0**: $\mathcal{T}(0) = \mathcal{T}^{(p)}(0) = \mathcal{C}^{(p)}(0), 1 \leq p \leq N$ is a single root block (genesis).
- **M1**: There is an independent leader election at every epoch beginning, i.e., at every block in the blocktree at level $c, 2c, ..., \ell c, ...$ The leader elections are won by the adversary according to independent Poisson processes of

rate $\lambda_a(t)$ at time t, one for every block at the aforementioned levels. The adversary can use the leader election won at a block at level ℓc at time t to propose a block at every block in the next $c-1$ levels $\ell c, \ell c+1, ..., \ell c+c-1$ that are present in the tree $\mathcal{T}(t)$. We refer the reader to Figure 4 for a visual representation.
- **M2**: Honest blocks are proposed at a total rate of $\lambda_h(t)$ at time t across all the honest nodes at the tip of the chain held by the mining node p, $\mathcal{C}^{(p)}(t)$.
- **M3**: The adversary can replace $\mathcal{T}^{(p)}(t^-)$ by another sub-tree $\mathcal{T}^{(p)}(t)$ from $\mathcal{T}(t)$ as long as the new sub-tree $\mathcal{T}^{(p)}(t)$ is an induced sub-tree of the new tree $\mathcal{T}^{(p)}(t)$, and can update $\mathcal{C}^{(p)}(t^-)$ to a longest chain in $T^{(p)}(t)$. [1]

We highlight the capabilities of the adversary in this model:

- **A1**: Can choose to propose block on multiple blocks of the tree $\mathcal{T}(t)$ at any time.
- **A2**: Can delay the communication of blocks between the honest nodes, but no more than Δ time.
- **A3**: Can broadcast private blocks at times of its own choosing: when private blocks are made public at time t to node p, then these blocks are added to $\mathcal{T}^{(p)}(t^-)$ to obtain $\mathcal{T}^{(p)}(t)$. Note that, under Δ-synchronous model, when private blocks appear in the view of some honest node p, they will also appear in the view of all other honest nodes by time $t+\Delta$.
- **A4**: Can switch the chain where the p-th honest node is proposing block, from one longest chain to another of equal length, even when its view of the tree does not change, i.e., $\mathcal{T}^{(p)}(t) = \mathcal{T}^{(p)}(t^-)$ but $\mathcal{C}^{(p)}(t) \neq \mathcal{C}^{(p)}(t^-)$.

It is to be noted that we don't consider the adversary to be *adaptive* in the sense that, although adversarial and honest nodes can join or leave the system as they wish, an adversary can never turn honest nodes adversarial. In order to defend against an adaptive adversary, key evolving signature schemes can be used [11]. However, in order to keep the system simple, we don't consider adaptive adversary.

Proving the security (persistence and liveness) of the protocol boils down to providing a guarantee that the chain $\mathcal{C}(t)$ converges fast as $t \to \infty$ and that honest blocks enter regularly into $\mathcal{C}(t)$ regardless of the adversary's strategy.

4 Security Analysis

Our goal is to generate a transaction ledger that satisfies persistence and liveness as defined in [17]. Together, persistence and liveness guarantee robust transaction ledger; honest transactions will be adopted to the ledger and be immutable.

Definition 2 (from [17]). *A protocol Π maintains a robust public transaction ledger if it organizes the ledger as a blockchain of transactions and it satisfies the following two properties:*

[1] All jump processes are assumed to be right-continuous with left limits, so that $\mathcal{C}(t), \mathcal{T}(t)$ etc. include the new arrival if there is a new arrival at time t.

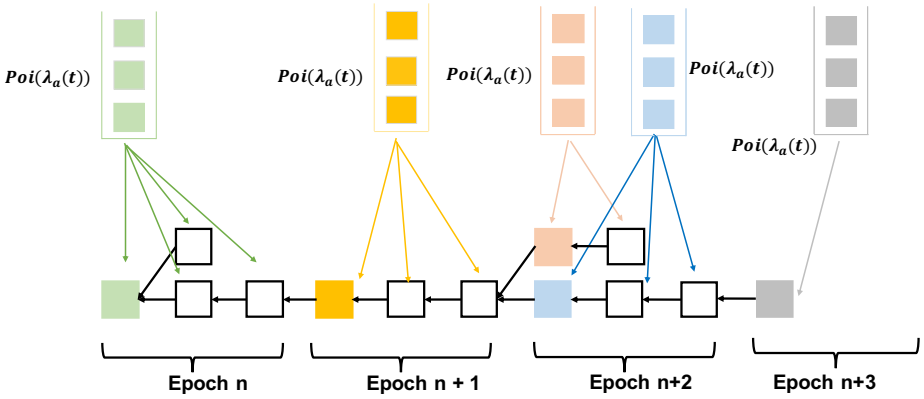


Fig. 4: There is a separate randomness generated for every block in the modulo c position. Blocks generated from that randomness at time t can attach to any block inside the next $c-1$ blocks that are present in the tree $\mathcal{T}(t)$.

- *(Persistence) Parameterized by $\tau \in \mathbb{R}$, if at a certain time a transaction* tx *appears in a block which is mined more than τ time away from the mining time of the tip of the main chain of an honest node (such transaction will be called confirmed), then* tx *will be confirmed by all honest nodes in the same position in the ledger.*
- *(Liveness) Parameterized by $u \in \mathbb{R}$, if a transaction* tx *is received by all honest nodes for more than time u, then all honest nodes will contain* tx *in the same place in the ledger forever.*

The theorem below shows that the the private attack threshold yields the true security threshold:

Theorem 1. *If*

$$\frac{\lambda_h^c(t)}{1+\lambda_{\max}\Delta} > \phi_c \lambda_a(t) \qquad \text{for all } t > 0,$$

then the PoSAT *generate transaction ledgers such that each transaction tx satisfies persistence (parameterized by $\tau = \rho$) and liveness (parameterized by $u = \rho$) in Definition 2 with probability at least $1 - e^{-\Omega(\rho^{1-\epsilon})}$, for any $\epsilon > 0$.*

In order to prove Theorem 1, we utilize the concept of blocktree partitioning and Nakamoto blocks that were introduced in [12]. We provide a brief overview of these concepts here.

Let τ_i^h and τ_i^a be the time when the i-th honest and adversary blocks are proposed, respectively; $\tau_0^h = 0$ is the time when the genesis block is proposed, which we consider as the 0-th honest block.

Definition 1. **Blocktree partitioning** Given the mother tree $\mathcal{T}(t)$, define for the i-th honest block b_i, the *adversary tree* $\mathcal{T}_i(t)$ to be the sub-tree of the mother

tree $\mathcal{T}(t)$ rooted at b_i and consists of all the adversary blocks that can be reached from b_i without going through another honest block. The mother tree $\mathcal{T}(t)$ is partitioned into sub-trees $\mathcal{T}_0(t), \mathcal{T}_1(t), \ldots \mathcal{T}_j(t)$, where the j-th honest block is the last honest block that was proposed before time t.

The sub-tree $\mathcal{T}_i(t)$ is born at time τ_i^h as a single block b_i and then grows each time an adversary block is appended to a chain of adversary blocks from b_i. Let $D_i(t)$ denote the depth of $\mathcal{T}_i(t)$; $D_i(\tau_i^h) = 0$.

Definition 2. [30] The j-th honest block proposed at time τ_j^h is called a *loner* if there are no other honest blocks proposed in the time interval $[\tau_j^h - \Delta, \tau_j^h + \Delta]$.

Definition 3. Given honest block proposal times τ_i^h's, define a honest fictitious tree $\mathcal{T}_h(t)$ as a tree which evolves as follows:

1. $\mathcal{T}_h(0)$ is the genesis block.
2. The first honest block to be proposed and all honest blocks within Δ are all appended to the genesis block at their respective proposal times to form the first level.
3. The next honest block to be proposed and all honest blocks proposed within time Δ of that are added to form the second level (which first level blocks are parents to which new blocks is immaterial) .
4. The process repeats.

Let $D_h(t)$ be the depth of $\mathcal{T}_h(t)$.

Definition 4. (**Nakamoto block**) Let us define:

$$E_{ij} = \text{event that } D_i(t) < D_h(t - \Delta) - D_h(\tau_i^h + \Delta) \text{ for all } t > \tau_j^h + \Delta. \quad (5)$$

The j-th honest block is called a *Nakamoto block* if it is a loner and

$$F_j = \bigcap_{i=0}^{j-1} E_{ij} \quad (6)$$

occurs.

See Figure 5 in [12] for illustration of the concepts of blocktree partitioning and Nakamoto blocks.

Lemma 1. *(Theorem 3.2 in [12]) (***Nakamoto blocks stabilize***) If the j-th honest block is a Nakamoto block, then it will be in the longest chain $\mathcal{C}(t)$ for all $t > \tau_j^h + \Delta$.*

Lemma 1 states that Nakamoto blocks remain in the longest chain forever. The question is whether they exist and appear frequently regardless of the adversary strategy. If they do, then the protocol has liveness and persistence: honest transactions can enter the ledger frequently through the Nakamoto blocks, and once they enter, they remain at a fixed location in the ledger. More formally, we have the following result.

Lemma 2. *(Lemma 4.4 in [12]) Define* $B_{s,s+t}$ *as the event that there is no Nakamoto blocks in the time interval* $[s, s+t]$ *where* $t \sim \Omega\left(\left[\frac{c-1}{\phi_c - 1}\right]^2\right)$. *If*

$$P(B_{s,s+t}) < q_t < 1 \tag{7}$$

for some q_t *independent of* s *and the adversary strategy, then the* PoSAT *generates transaction ledgers such that each transaction* tx *satisfies* persistence *(parameterized by* $\tau = \rho$*) and* liveness *(parameterized by* $u = \rho$*) in Definition 2 with probability at least* $1 - q_\rho$.

In order to prove Lemma 2, we proceed in five steps as illustrated in Fig. 5.

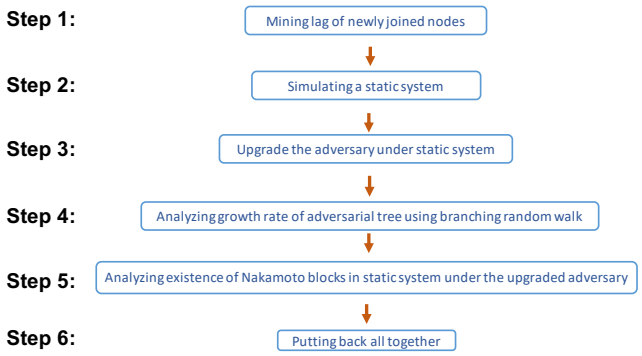


Fig. 5: Flowchart of the proof for Lemma 2.

4.1 Step 1: Mining lag of newly joined nodes

From section 3, recall that $\lambda_h(t)$ is defined as the stake of the coins that are online at time t but has encountered at least one epoch beginning. That implies, within an epoch, $\lambda_h(t)$ is the effective honest stake that can be used to contribute towards the growth of the canonical chain; it remains constant and gets updated only at the epoch beginning. In order to analyze the effect of this lag in a honest node to start mining, we simulate a new dynamic available system, *dyn*2, where, at time t, an has been online in the original dynamic system since at least time $t - \sigma(c)$, where, $\sigma(c) > 0$. Recall that $\lambda_h^c(t)$ be defined as the stake of the coins that are online at time t in the original dynamic system and has been online since at least $t - \sigma(c)$. Clearly, $\lambda_h^c(t)$ is the rate at which the honest nodes win leader election at time t in *dyn*2. We have the following relationship between the original dynamic available system and *dyn*2.

Lemma 3. *For the dynamic available system dyn*2 *and for all* $s, t > 0$, *define* $B_{s,s+t}^{dyn2}$ *as the event that there are no Nakamoto blocks in the time interval* $[s, s+t]$.

Let κ_0 *be the solution for the equation* $\ln\left(\frac{\lambda_{\max}}{\lambda_{\min}}(1+\kappa)\right) = \kappa$. *Then, for* $\sigma(c) = c\Delta + \frac{c(1+\kappa)}{\lambda_{\min}}$ *and* $\kappa >> \kappa_0$, *we have*

$$P(B_{s,s+t}) \leq P(B^{dyn2}_{s,s+t}) + e^{-O(\kappa)}.$$

The proof is given in Appendix C.

4.2 Step 2: Simulating a static system

Without loss of generality, we assume that the adversarial power is boosted such that $\lambda_a(t) = (1-\eta)\lambda^c_h(t)$. Let λ_h be some positive constant. Taking $dyn2$ as the base, we simulate a static system, $ss0$, where both honest nodes and adversary win leader elections with constant rates λ_h and $\lambda_a = (1-\eta)\lambda_h$, respectively. This requires, for a local time $t > 0$ in $dyn2$, defining a new local time $\alpha(t)$ for $ss0$ such that

$$\lambda^c_h(u)du = \lambda_h d\alpha \implies \alpha(t) = \int_0^t \frac{\lambda^c_h(u)}{\lambda_h}du. \tag{8}$$

Additionally, for every arrival of an honest or adversarial block in $dyn2$ at a particular level at a tree, there is a corresponding arrival in $ss0$ at the same level in the same tree. For a time t in the local clock of $dyn2$, let $\Delta^{ss0}(t)$ be the network delay of $dyn2$ measured with reference to the local clock of $ss0$. Using (8), we have

$$\frac{\lambda_{\min}}{\lambda_h}\Delta \leq \Delta^{ss0}(t) \leq \frac{\lambda_{\max}}{\lambda_h}\Delta. \tag{9}$$

We have the following relationship between $dyn2$ and $ss0$.

Lemma 4. *Consider the time interval* $[s, s+t]$ *in the local clock of* $dyn2$. *For the static system* $ss0$, *define* $B^{ss0}_{\alpha(s),\alpha(s+t)}$ *as the event that there are no Nakamoto blocks in the time interval* $[\alpha(s), \alpha(s+t)]$ *in the local clock of* $ss0$. *Then,*

$$P(B^{dyn2}_{s,s+t}) = P(B^{ss0}_{\alpha(s),\alpha(s+t)}).$$

The proof for this lemma is given in Appendix D.

4.3 Step 3: Upgrading the adversary

As the occurrence of Nakamoto blocks is a race between the fictitious honest tree and the adversarial trees from the previous honest blocks, we next turn to an analysis of the growth rate of an adversary tree. However, the growth rate of an adversarial tree would now depend on the location of the root honest block within an epoch which adds to the complexity of the analysis. To get around this complexity, we simulate a new static system, $ss1$ in which the adversary, on winning a leader election after evaluating RandVDF.Eval and

appending a block to an honest block (that is, growing a new adversarial tree), is given a gift of chain of $c-1$ extra blocks for which the adversary doesn't have to compute RandVDF.Eval. Thus, the adversary has to compute only one RandVDF.Eval for the chain of first c blocks in the adversarial tree. At this point, the adversary can assume a new epoch beginning and accordingly update `randSource`. Hereafter, the evolution of `randSource` follows the rules in $ss0$. Note that the local clock for both the static systems $ss0$ and $ss1$ are same. Now, we have the following relationship between $ss0$ and $ss1$.

Lemma 5. *Consider the time interval $[s, s+t]$ in the local clock of dyn2. For the static system ss1, define $B^{ss1}_{\alpha(s),\alpha(s+t)}$ as the event that there are no Nakamoto blocks in the time interval $[\alpha(s), \alpha(s+t)]$ in the local clock of ss1. Then,*

$$P(B^{ss0}_{\alpha(s),\alpha(s+t)}) \leq P(B^{ss1}_{\alpha(s),\alpha(s+t)}).$$

The proof for this lemma is given in Appendix E.

For analyzing $P(B^{ss1}_{\alpha(s),\alpha(s+t)})$, we first consider an arbitrary static system $ss2$ where both honest nodes and adversary win leader elections with constant rates λ_h and λ_a, respectively, the honest nodes follows PoSAT, the adversary has similar additional power of gift of chain of $c-1$ blocks as in $ss1$ but the network delay is a constant, say Δ'. For some $s', t' > 0$ in the local clock of the static system $ss2$, we will determine an upper bound on $P(B^{ss2}_{s',s'+t'})$ in Sections 4.4 - 4.5 and then use this result to obtain an upper bound on $P(B^{ss1}_{\alpha(s),\alpha(s+t)})$ in Section 4.6.

4.4 Step 4: Growth rate of the adversarial tree

For time $t' > 0$, let $\hat{\mathcal{T}}_i(t')$ represents the adversarial tree in $ss2$ with i^{th} honest block as its root. The depth $D_i(t')$ at time t' in the local clock of $ss2$ is defined as the maximum depth of the blocks of $\hat{\mathcal{T}}_i(t')$ at time t'. In Lemma 6, we evaluate the tail bound on $D_i(t')$.

Lemma 6. *For $x > 0$ so that $\eta_c \lambda_a t' + x$ is an integer,*

$$P(D_i(t') \geq \phi_c \lambda_a t' + cx) \leq e^{-\theta^*_c t'} e^{(\eta_c \lambda_a t' + x - 1)\Lambda_c(\theta^*_c)} g(t'). \tag{10}$$

where $\phi_c = c\eta_c$, $g(t') = \sum_{i_1 \geq 1} \int_0^{t'} \frac{\lambda_a^{i_1} u^{i_1 - 1} e^{-\lambda_a u}}{\Gamma(i_1)} e^{\theta^*_c u} du$, $\Lambda_c(\theta_c) = \log(-\lambda_a^c / \theta_c(\lambda_a - \theta_c)^{c-1})$ and θ^*_c is the solution for the equation $\Lambda_c(\theta) = \theta \dot{\Lambda}_c(\theta)$

Details on the analysis of $\hat{\mathcal{T}}_i(t')$ and the proof of Lemma 6 are in Appendix F.

4.5 Step 5: Existence of Nakamoto blocks

With Lemma 6, we show below that in the static system $ss2$ in the regime $\phi_c \lambda_a < \frac{\lambda_h}{1 + \lambda_h \Delta'}$, Nakamoto blocks has a non-zero probability of occurrence.

Lemma 7. *If*

$$\phi_c \lambda_a < \frac{\lambda_h}{1 + \lambda_h \Delta'},$$

then, in the static system ss2, there is a $p > 0$ such that the probability of the $j-th$ honest block being a Nakamoto block is greater than p for all j.

The proof of this result can be found in Appendix G.2.

Having established the fact that Nakamoto blocks occurs with non-zero frequency, we can bootstrap on Lemma 7 to get a bound on the probability that in a time interval $[s', s' + t']$, there are no Nakamoto blocks, i.e. a bound on $P(B_{s',s'+t'})$.

Lemma 8. *If*

$$\phi_c \lambda_a < \frac{\lambda_h}{1 + \lambda_h \Delta'},$$

then for any $\epsilon > 0$, there exist constants $\bar{a}_\epsilon, \bar{A}_\epsilon$ so that for all $s' \geq 0$ and $t' > \max\left\{\left(\frac{2\lambda_h}{1-\eta}\right)^2 \left(\frac{c-1}{\phi_c - 1}\right)^2, \left[(c-1)\left(\Delta' + \frac{1}{\lambda_{\min}}\right)\right]^2\right\}$, we have

$$P(B^{ss2}_{s',s'+t'}) \leq \bar{A}_\epsilon \exp(-\bar{a}_\epsilon t'^{1-\epsilon}) \tag{11}$$

where $\bar{a}_\epsilon$ is a function of Δ'.

The proof of this result can be found in Appendix G.3.

4.6 Step 6: Putting back all together

In this section, we use the results from Section 4.5 to upper bound $P(B^{ss1}_{\alpha(s),\alpha(s+t)})$ and hence, $P(B_{s,s+t})$.

Using equation 8, we have $\phi_c \lambda_a(t) < \frac{\lambda^c_h(t)}{1+\lambda_{\max}\Delta} \iff \phi_c \lambda_a < \frac{\lambda_h}{1+\lambda_{\max}\Delta}$. Then, we have the following lemma:

Lemma 9. *If*

$$\phi_c \lambda_a(t) < \frac{\lambda^c_h(t)}{1 + \lambda_{\max}\Delta},$$

then for any $\epsilon > 0$ there exist constants $\bar{a}_\epsilon, \bar{A}_\epsilon$ so that for all $s \geq 0$ and $t > \max\left\{\left(\frac{2\lambda_h}{1-\eta}\right)^2 \left(\frac{\lambda_h}{\lambda_{\min}}\right)\left(\frac{c-1}{\phi_c - 1}\right)^2, \left(\frac{\lambda_h}{\lambda_{\min}}\right)\left[(c-1)\left(\Delta + \frac{1}{\lambda_{\min}}\right)\right]^2\right\}$, we have

$$P(B_{s,s+t}) \leq \bar{A}_\epsilon \exp(-\bar{a}_\epsilon t^{1-\epsilon}). \tag{12}$$

The proof for this result is given in Appendix H. Then, combining Lemma 9 with Lemma 2 implies Theorem 1.

5 Discussion

In this section, we discuss some of the practical considerations in adopting PoSAT.

A key question in PoSAT is what is the right choice of c? If c is low, say 10, then the security threshold is approximately 1.58. At $c = 10$, the protocol is fully unpredictable and the confirmation latency is not too high. Also, any newly joining honest node has to wait for around 10 inter-block arrivals before it can participate in leader election. Thus, if there is a block arrival every second, then, the node has to wait for 10 secs. In any standard blockchain, there is always a bootstrap period for the node to ensure that the state is synchronized with the existing peers and 10 secs is negligible as compared to the bootstrap period.

In PoSAT, a separate RandVDF needs to be run for each public-key. In a purely decentralized implementation, all nodes may not have the same rate of computing VDF. This may disadvantage nodes whose rate of doing sequential computation is slower. One approach to solve this problem is to build open-source hardware for VDF - this is already under way through the VDF Alliance. Even under such a circumstnace, it is to be expected that nodes that can operate their hardware in idealized circumstances (for example, using specialized cooling equipment) can gain an advantage. A desirable feature of our protocol is that gains obtained by a slight advantage in the VDF computation rate are bounded. For PoSAT, a combination of the VDF computation rate and the stake together yields the net power weilded by a node, and as long as a majority of such power is controlled by honest nodes, we can expect the protocol to be safe.

In our PoSAT specification, the difficulty parameter for the computation of RandVDF.Eval was assumed to be fixed. This threshold was chosen based on the entire stake being online - this was to ensure that forking even when all nodes are present remains small, i.e., $\lambda_{\max}\Delta$ remained small. In periods when far fewer nodes are online, this leads to a slowdown in confirmation latency. A natural way to mitigate this problem is to use a variable mining threshold based on past history, similar to the adaptation inherent in Bitcoin. A formal analysis of Bitcoin with variable difficulty was carried out in [18, 19], we leave a similar analysis of our protocol for future work.

In our protocol statement, we have used the RandVDF directly on the randomness prevRand and the public key. The RandVDF ensures that any other node can only predict a given node's leadership slot at the instant that it actually wins the VDF lottery. However, this still enables an adversary to predict the leadership slots of nodes that are offline and can potentially bribe them to come online to favor the adversary. In order to eliminate this exposure, we can replace the hash in the mining condition by using a verifiable random function [13, 24] (which is calculated using the node's secret key but can be checked using the public key). This ensures that an adversary which is aware of all the public state as well as private state of all *online* nodes (including their VRF outputs) still cannot predict the leadership slot of any node ahead of the time at which they can mine the block. This is because, such an adversary does not have access to the VRF output of the offline nodes.

There are two types of PoS protocols: one favoring liveness under dynamic availability and other favoring safety under asynchrony. BFT protocols fall into the latter class and lack dynamic availability. One shortcoming of the longest chain protocol considered in the paper is the reduced throughput and latency compared to the fundamental limits; this problem is inherited from the Nakamoto consensus for PoW [25]. However, a recent set of papers address these problems in PoW (refer Prism [3], OHIE [36] and Ledger Combiners [16]). Adaptations of these ideas to the PoSAT protocol is left for future work. Furthermore, our protocol, like Nakamoto, does not achieve optimal chain quality. Adopting ideas from PoW protocols with optimal chain quality, such as Fruitchains [27], is also left for future work.

Finally, while we specified PoSAT in the context of proof-of-stake, the ideas can apply to other mining modalities - the most natural example is proof-of-space. We note that existing proof-of-space protocols like Chia [10], use a VDF for a fixed time, thus making the proof-of-space challenge predictable. In proof-of-space, if the predictability window is large, it is possible to use slow-storage mechanisms such as magnetic disks (which are asymmetrically available with large corporations) to answer the proof-of-space challenges. Our solution of using a RandVDF can be naturally adapted to this setting, yielding unpredictability as well as full dynamic availability.

References

1. Azouvi, S., McCorry, P., and Meiklejohn, S. Betting on blockchain consensus with fantomette. *arXiv preprint arXiv:1805.06786* (2018).
2. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., and Zikas, V. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018), ACM, pp. 913–930.
3. Bagaria, V., Dembo, A., Kannan, S., Oh, S., Tse, D., Viswanath, P., Wang, X., and Zeitouni, O. Proof-of-stake longest chain protocols: Security vs predictability. *arXiv preprint arXiv:1910.02218* (2019).
4. Bentov, I., Pass, R., and Shi, E. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive 2016* (2016), 919.
5. Boneh, D., Bonneau, J., Bünz, B., and Fisch, B. Verifiable delay functions. In *Annual international cryptology conference* (2018), Springer, pp. 757–788.
6. Brown-Cohen, J., Narayanan, A., Psomas, A., and Weinberg, S. M. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation* (2019), pp. 459–473.
7. Buchman, E., Kwon, J., and Milosevic, Z. The latest gossip on BFT consensus, 2018.
8. Cai, J.-Y., Lipton, R. J., Sedgewick, R., and Yao, A.-C. Towards uncheatable benchmarks. In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference* (1993), IEEE, pp. 2–11.
9. Chen, J., and Micali, S. Algorand. *arXiv preprint arXiv:1607.01341* (2016).
10. Cohen, B., and Pietrzak, K. The chia network blockchain. *https://www.chia.net/assets/ChiaGreenPaper.pdf* (2019).

11. David, B., Gaži, P., Kiayias, A., and Russell, A. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2018), Springer, pp. 66–98.
12. Dembo, A., Kannan, S., Tas, E. N., Tse, D., Viswanath, P., Wang, X., and Zeitouni, O. Everything is a race and nakamoto always wins. *ACM CCS, see also arXiv preprint arXiv:2005.10484* (2020).
13. Dodis, Y., and Yampolskiy, A. A verifiable random function with short proofs and keys. In *International Workshop on Public Key Cryptography* (2005), Springer, pp. 416–431.
14. Ephraim, N., Freitag, C., Komargodski, I., and Pass, R. Continuous verifiable delay functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2020), Springer, pp. 125–154.
15. Fan, L., and Zhou, H.-S. A scalable proof-of-stake blockchain in the open setting (or, how to mimic nakamoto's design via proof-of-stake), 2018. Cryptology ePrint Archive, Report 2017/656, Version 20180425:201821.
16. Fitzi, M., Gaži, P., Kiayias, A., and Russell, A. Ledger combiners for fast settlement. In *Theory of Cryptography Conference* (2020), Springer, pp. 322–352.
17. Garay, J., Kiayias, A., and Leonardos, N. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2015), Springer, pp. 281–310.
18. Garay, J., Kiayias, A., and Leonardos, N. Full analysis of nakamoto consensus in bounded-delay networks. Cryptology ePrint Archive, Report 2020/277, 2020. `https://eprint.iacr.org/2020/277`.
19. Garay, J. A., Kiayias, A., and Leonardos, N. The bitcoin backbone protocol with chains of variable difficulty. Cryptology ePrint Archive, Report 2016/1048, 2016. `https://eprint.iacr.org/2016/1048`.
20. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles* (2017), ACM, pp. 51–68.
21. Kiayias, A., Russell, A., David, B., and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference* (2017), Springer, pp. 357–388.
22. Long, J., and Wei, R. Nakamoto consensus with verifiable delay puzzle. *arXiv preprint arXiv:1908.06394* (2019).
23. Mahmoody, M., Moran, T., and Vadhan, S. Publicly verifiable proofs of sequential work. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science* (2013), pp. 373–388.
24. Micali, S., Rabin, M., and Vadhan, S. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)* (1999), IEEE, pp. 120–130.
25. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.
26. Pass, R., Seeman, L., and Shelat, A. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2017).
27. Pass, R., and Shi, E. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing* (2017), pp. 315–324.
28. Pass, R., and Shi, E. The sleepy model of consensus. In *International Conference on the Theory and Application of Cryptology and Information Security* (2017), Springer, pp. 380–409.

29. Pietrzak, K. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)* (2018), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
30. Ren, L. Analysis of nakamoto consensus. Tech. rep., Cryptology ePrint Archive, Report 2019/943.(2019). https://eprint. iacr. org . . . , 2019.
31. Rivest, R. L., Shamir, A., and Wagner, D. A. Time-lock puzzles and timed-release crypto.
32. Shi, Z. *Branching Random Walks*, vol. 2151 of *Lecture Notes in Mathematics*. Springer Verlag, New York NY, 2015.
33. Wang, X. e. a. Proof-of-stake longest chain protocol revisited. *arXiv preprint arXiv:1910.02218v2* (2018).
34. Wesolowski, B. Efficient verifiable delay functions. *Journal of Cryptology* (2020), 1–35.
35. Yin, M., Malkhi, D., Reiter, M. K., Gueta, G. G., and Abraham, I. Hotstuff: Bft consensus in the lens of blockchain. *arXiv preprint arXiv:1803.05069* (2018).
36. Yu, H., Nikolic, I., Hou, R., and Saxena, P. Ohie: blockchain scaling made simple. *arXiv preprint arXiv:1811.12628* (2018).

Appendix

A Suite of Possible Attacks Under Dynamic Availability

In this section, we describe the suite of possible attacks under dynamic availability in PoS systems. This attacks are possible even under static stake. We also discuss some design recommendations for mitigating against such attacks in PoS systems.

A.1 Content-grinding attack

Referring to Fig 3b, we note that the content of the block, namely the transactions, were not used in determining whether the Threshold(s) is satisfied or not. If we instead checked whether $\textsc{Hash}(O_i, \texttt{slot}, \texttt{transactionList}) < \textsc{Threshold}(s)$ instead of $\textsc{Hash}(O_i, \texttt{slot}) < \textsc{Threshold}(s)$ the protocol loses security due to the ability of adversary to choose the set of transactions in order to increase its likelihood of winning the leadership certificate.

In such a case, the adversary can get unlimited advantage by performing such content-grinding by parallel computation over different sets and orders of transactions. We note that in PoW the adversary does not gain any advantage by performing such content grinding, since it is equivalent to grinding on the Nonce, which is the expected behavior anyway.

A.2 Sybil attack

One natural attack in PoS for an adversary to sybil the stake contained in a single coin and distribute it across multiple coins which might increase the probability of winning a leader election from at least one of the coins. We describe next that having difficulty parameter in RandVDF.Eval proportional to the stake of the coin defends against such sybil attack. Let us consider H to be the value of the hash function on the output of VDF in an iteration, R to be range of this hash function and th be the difficulty parameter that is proportional to the stake of the coin. Suppose $p = P(\mathrm{H} < th) = \frac{th}{R}$, which is the probability of winning the leader election in each iteration of the VDF. If we sybil the stake into , let's say, three coins with equal stakes, then, the probability of winning leader election for each individual stake in each iteration of VDF is $P(\mathrm{H} < \frac{th}{3}) = \frac{th}{3R} = \frac{p}{3}$. This is due to the fact that difficulty parameter th is proportional to the stake. Hence, the probability of winning leader election in each iteration of VDF by at least one coin is $1 - \left(1 - \frac{p}{3}\right)^3$. However, as the VDFs are iterating very fast, we are in the regime $0 < p << 1$. Thus, by Binomial series expansion, $1 - \left(1 - \frac{p}{3}\right)^3 = 1 - (1 - 3\frac{p}{3} + O(p^2)) = p + O(p^2)$. Hence, this validates our design choice that difficulty parameter is proportional to the stake of the coin. This design choice is not unique to our design and is common in all longest chain based proof-of-stake protocols.

A.3 Costless simulation attack

Both sleepy model of consensus [28] and Ouroboros Genesis [2] have a weaker definition of dynamic availability: *all adversary nodes are always online starting from genesis and no new adversary nodes can join*, which makes them vulnerable to costless simulation attack as described next. In case of sleepy model of consensus [28], as shown in Fig 6, suppose that in the 1^{st} year of the existence of the PoS system, only 5% of the total stake, all honest, is online and actively participating in evolution of the blockchain. Consider that at the beginning of the 2^{nd} year, all 100% of the stake is online with 20% of the stake being controlled by the adversary. The adversary can costlessly simulate (with requirement of little

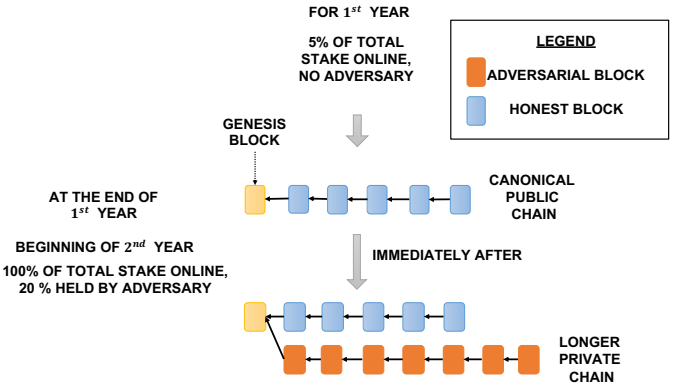


Fig. 6: Costless simulation attack for sleep model of consensus [28].

computational time) the eligibility condition in sleepy protocol across large range of values of time t, thus, constructing a longer private chain than the canonical public chain. In sleepy, under the fork-choice rule of choosing the longest chain, the private chain will be selected as the canonical chain once it is revealed by the adversary. In case of Ouroboros Genesis [2], as shown in Fig 7, the adversary can utilize the 20% stake under its control after the slot s_{change} to costlessly construct a private chain involving leader elections for the slots starting from the checkpoint slot $s_{checkpoint}$. Observe that in the slots $[s_{checkpoint}, s_{change}]$ of the operation of the PoS system, the canonical public chain evolved due to the participation of only 5% stake. Clearly, with high probability, for any s such that $s_{checkpoint} + s < s_{change}$, the private chain has more blocks in the range $[s_{checkpoint}, s_{checkpoint} + s]$ as compared to canonical public chain. Under the fork-choice rule of Ouroboros Genesis as described in Fig. 10 of [2], the private adversarial chain will be selected as the canonical chain when it is revealed.

If the fork-choice rule is to choose the longest chain, the design recommendation for defending against costless simulation attack is to make it *expensive* for the adversary to propose blocks for the past slots and create longer chain. For

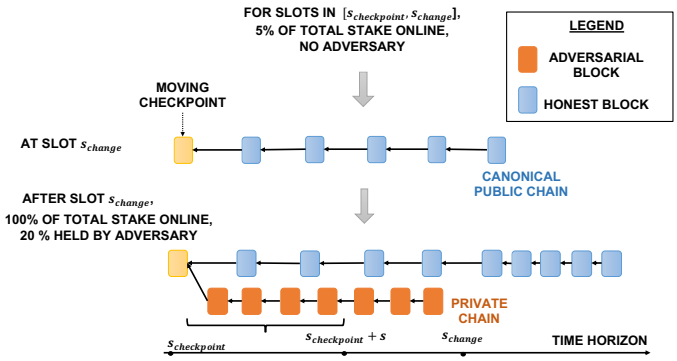


Fig. 7: Costless simulation attack for Ouroboros Genesis [2].

instance, in PoSAT, the adversary would have to initiate its RandVDF from the first block of the epoch where `randSource` is updated. Due to the sequential nature of the computation of RandVDF, with high probability, the adversary won't be able to create a private chain longer than the canonical chain.

A.4 Bribery attack due to predictability

A key property of PoW protocols is their ability to be unpredictable: no node (including itself) can know when a given node will be allowed to propose a block ahead of the proposal slot. In the existing PoS protocols, there are two notions of predictability depending on how the leader election winner is decided - *globally predictable* and *locally predictable*. Referring to Fig 8, using hash function for

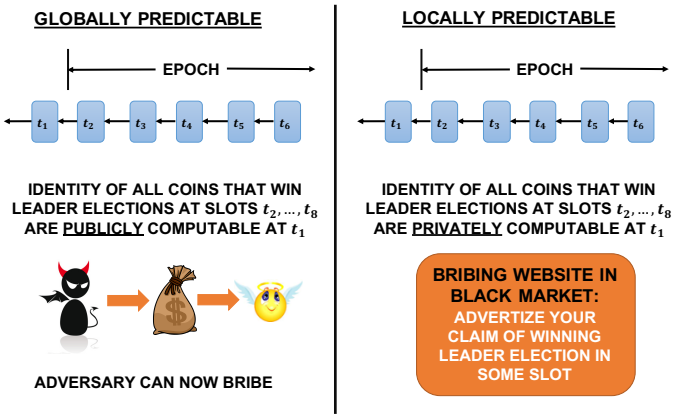


Fig. 8: Variations of bribery attack stemming from predictability.

deciding winner of leader election, as in [28], renders the identity of winners of

leader elections in future slots publicly computable. An adversary can now bribe a coin that is going to propose a block in a future slot to include or exclude a specific transaction of adversary's choice or influence the position on where to append the block. On the other hand, using verifiable random function (VRF) for deciding winner of leader election, as in Ouroboros Praos [11], Ouroboros Genesis [2] and Snow White [4], mutes the aforementioned public computability. However, a node owning a coin can still locally compute the future slots in which that coin can win the leader election. Now, the node can advertise its future electability in the black market.

The central idea on how to avoid such predictability is to ensure that a node owning a coin shouldn't learn about winning a leader election with that coin in slot s before the slot s. In PoSAT, owing to randomness of `randIter` in RandVDF, the node learns about winning a leader election for that coin in slot s only after completion of sequential execution of the RandVDF at slot s.

A.5 Private attack by enumerating blocks within an epoch

In PoSAT, at the beginning of each epoch, the `randSource` is updated. However,

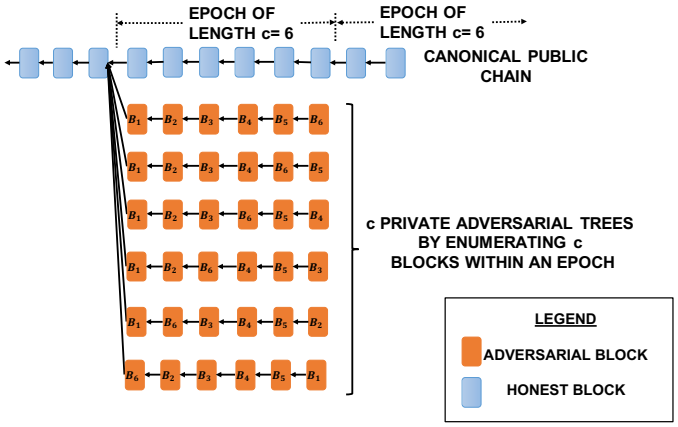


Fig. 9: Enumerating blocks when time-ordering is not required.

if the appropriate guardrail in the form of time-ordering (line 38 in Algorithm 1) is not put into place, then, this `randSource` update can provide statistical advantage to the adversary in creating longer private chain. To be specific, suppose that PoSAT doesn't require the `slot` in the blocks of a chain to be ordered in the ascending order. Then, as shown in Fig 9, the adversary can enumerate over the c blocks in the private adversarial tree to have c different `randSource` updates for the next epoch. This gives c distinct opportunities to the adversary to evolve the private adversarial tree which gives the aforementioned statistical advantage of order c in terms of inter-arrival time of the adversarial blocks.

With the guardrail of time-ordering in place, as in PoSAT, the aforementioned enumeration is not possible as the `slot` contained in the blocks of a chain are required to be in ascending order.

A.6 Long-range attack by leveraging randomness update

Updating `randSource` for a new epoch based on solely the last block of the previous epoch, as done in PoSAT, gives rise to an unique situation in which an adversary can mount a long-range attack to create a longer private adversarial chain. Referring to Fig 10, an adversary, with sufficiently large probability, can win at least one leader election in each epoch and publicly reveal the block associated with that leader election after appropriate delay so that the block ends up as the last block of that epoch. Observe that this strategy will give

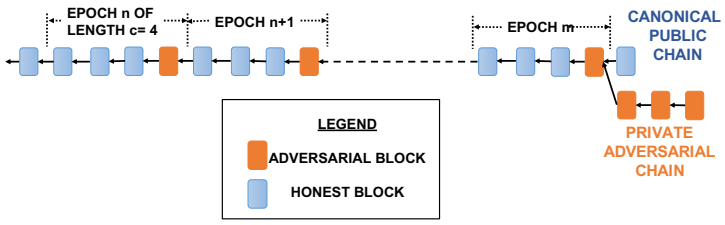


Fig. 10: Illustration of the long-range attack. Consider that $m > n$.

power to the adversary to dictate the `randSource` for each epoch. Moreover, the adversary, on proposing on winning a leader election for an epoch, can just move on to contesting a leader election for the next epoch. Thus, the adversary can privately behave as if $c = 1$ whereas the actual c might be greater than 1. With such a strategy, the adversary can win at least one leader election for many future epochs and publicly reveal the blocks associated with those leader elections in a time-appropriate manner. The adversary can continue this strategy until an appropriate epoch when it is able to win multiple leader elections and wants to do double spending. There are two design recommendations on how to protect against this long-range attack:

- requiring time-ordering of the blocks in a chain, as done in PoSAT, would ensure that even after the adversary behaves as if $c = 1$ and wins leader elections for future epochs, the blocks associated with those leader elections would fail the time-ordering (line 38 in Algorithm 1). This completely removes the aforementioned long-range attack.
- requiring that the `randSource` for a new epoch is dependent on all the blocks of the previous epoch. This strategy diminishes the amount of influence that an adversary can have on the `randSource` update.

B Supplementary for Section 2.1

We give a brief description of VDFs, starting with its definition.

Definition 3 (from [5]). *A VDF* $V = (Setup, Eval, Verify)$ *is a triple of algorithms as follows:*

- $Setup(\lambda, \tau) \to \mathbf{pp} = (ek, vk)$ *is a randomized algorithm that takes a security parameter* λ *and a desired puzzle difficulty* t *and produces public parameters* **pp** *that consists of an evaluation key* ek *and a verification key* vk*. We require Setup to be polynomial-time in* λ*. By convention, the public parameters specify an input space* $\mathcal{X}$ *and an output space* $\mathcal{Y}$*. We assume that* $\mathcal{X}$ *is efficiently sampleable. Setup might need secret randomness, leading to a scheme requiring a trusted setup. For meaningful security, the puzzle difficulty* τ *is restricted to be sub-exponentially sized in* λ*.*
- $Eval(ek, input, \tau) \to (O, proof)$ *takes an input* $\in \mathcal{X}$ *and produces an output* $O \in \mathcal{Y}$ *and a (possibly empty)* $proof$*. Eval may use random bits to generate the* $proof$ *but not to compute* O*. For all* **pp** *generated by* $Setup(\lambda, \tau)$ *and all input* $\in \mathcal{X}$*, algorithm* $Eval(ek, input, \tau)$ *must run in parallel time* τ *with* $poly(log(\tau), \lambda)$ *processors.*
- $Verify(vk, input, O, proof) \to Yes, No$ *is a deterministic algorithm takes an input, output and proof and outputs* Yes *or* No*. Algorithm Verify must run in total time polynomial in* $log\tau$ *and* λ*. Notice that* $Verify$ *is much faster than* $Eval$*.*

The definition for correctness and soundness for RandVDF is defined as follows:

Definition 4 (Correctness). *A* RandVDF V *is correct if for all* λ, τ*, parameters* $(ek, vk) \xleftarrow{\$} \textsc{Setup}(\lambda)$*, and all input* $\in X$ *, if* $(O, proof) \xleftarrow{\$} \textsc{Eval}(ek, input, \tau)$ *then* $\textsc{Verify}(vk, input, O, proof) = Yes$*.*

Definition 5 (Soundness). *A* RandVDF *is sound if for all algorithms* $\mathcal{A}$ *that run in time* $O(poly(t, \lambda))$

$$Pr\left[\begin{matrix}\textsc{Verify}(vk,input,O,proof)=Yes \\ O\neq\textsc{Eval}(ek,input,\tau)\end{matrix}\middle|\begin{matrix}pp=(ek,vk)\xleftarrow{\$}\textsc{Setup}(\lambda) \\ (input,O,proof)\xleftarrow{\$}\mathcal{A}(\lambda,pp,\tau)\end{matrix}\right] \leq negl(\lambda)$$

C Proof of Lemma 3

First, we prove the following lemma.

Lemma 10. *Define*

$$E_1 = \{\text{There is no epoch-beginning within the time interval } [t - \sigma(c), t]\} \quad (13)$$

and, let κ_0 *be the solution for the equation* $\ln\left(\frac{\lambda_{\max}}{\lambda_{\min}}(1+\kappa)\right) = \kappa$*. Then, for* $\sigma(c) = c\Delta + \frac{c(1+\kappa)}{\lambda_{\min}}$ *and* $\kappa >> \kappa_0$*, we have*

$$P(E_1) \leq e^{-O(\kappa)}.$$

Proof. Define $X_d, d > 0$, as the time it takes for D_h in the original dynamic available system to reach depth d after reaching depth $d-1$. Then, for some $d_0 > 0$, we have $E_1 = \left\{\sum_{d=d_0}^{d_0+c-1} X_d > \sigma(c)\right\}$. Observe that, due to our blocktree partitioning, $X_d = \Delta + Y_d$, where Y_d is a non-homogeneous exponential random variable. Therefore, by Chernoff bound, for any $v > 0$

$$
\begin{aligned}
P\left(\sum_{d=d_0}^{d_0+c-1} X_d > \sigma(c)\right) &\leq \mathbb{E}\left(e^{v\sum_{d=d_0}^{d_0+c-1} X_d - v\sigma(c)}\right) \\
&= \mathbb{E}\left(e^{v\sum_{d=d_0}^{d_0+c-1} Y_d}\right) e^{vc\Delta - v\sigma(c)} \\
&\overset{(a)}{=} e^{vc\Delta - v\sigma(c)} \mathbb{E}_{Y_{d_0}|X_{d_0-1}} e^{vY_{d_0}} \cdots \mathbb{E}_{Y_{d_0+c-1}|X_{d_0-1}\cdots X_{d_0+c-2}} e^{vY_{d_0+c-1}} \\
&\overset{(b)}{\leq} e^{vc\Delta - v\sigma(c)} \left(\frac{\lambda_{\max}}{\lambda_{\min} - v}\right)^c \\
&\overset{(c)}{=} e^{-v\frac{c(1+\kappa)}{\lambda_{\min}}} \left(\frac{\lambda_{\max}}{\lambda_{\min} - v}\right)^c
\end{aligned}
$$

where (a) is due to law of total expectation, (b) is due to the fact that, if $f_{Y_{d_0+i}|X_{d_0-1}\cdots X_{d_0+i-1}}(y)$ is the pdf of Y_{d_0+i} given $X_{d_0-1}\cdots X_{d_0+i-1}$, then $\lambda_{\min} \leq \lambda_h(t) \leq \lambda_{\max}$ implies $f_{Y_{d_0+i}|X_{d_0-1}\cdots X_{d_0+i-1}}(y) \leq \lambda_{\max} e^{-\lambda_{\min} y}$, (c) is by putting $\sigma(c) = c\Delta + \frac{c(1+\kappa)}{\lambda_{\min}}$. Optimizing over v implies that for $v = \lambda_{\min}\left(\frac{\kappa}{1+\kappa}\right)$, we have

$$
P\left(\sum_{d=d_0}^{d_0+c-1} X_d > \sigma(c)\right) \leq e^{c\left[-\kappa + \ln\left(\frac{\lambda_{\max}}{\lambda_{\min}}(1+\kappa)\right)\right]}
$$

Note that for $\kappa >> \kappa_0$, we have $P\left(\sum_{d=d_0}^{d_0+c-1} X_d > \sigma(c)\right) \leq e^{-O(\kappa)}$. □

Recall that, under the design of simulated system $dyn2$, if an honest coin has been online in the original dynamic available system for at least time $\sigma(c)$, then the coin can also contribute to the growth of the canonical chain in $dyn2$. From Lemma 10, we know that for $\sigma(c) = c\Delta + \frac{c(1+\kappa)}{\lambda_{\min}}$ and $\kappa >> \kappa_0$, this honest coin has encountered at least one epoch-beginning in the original dynamic available system with probability $1-e^{-O(\kappa)}$. That implies, with high probability $1-e^{-O(\kappa)}$, at time t, if an honest coin is contributing to the growth of the canonical chain in $dyn2$, then it is also contributing to the growth of the canonical chain in the original dynamical system. However, observe that at the same time t in the original dynamic available system, there might be other honest coins which became online after $t-\sigma(c)$ and have encountered at least one epoch-beginning. At time t, these coins would contribute to the growth of the canonical chain in the original dynamic available system but won't be contributing to the growth of the canonical chain in $dyn2$. Thus, $\lambda_h^c(t) \leq \lambda_h(t)$ with probability $1 - e^{-O(\kappa)}$. Consequently, $P(B_{s,s+t}) \leq P(B_{s,s+t}^{dyn2}) + e^{-O(\kappa)}$.

D Proof of Lemma 4

Observe that from (8), we have

$$\int_{t_1}^{t_2} \lambda_h^c(t)dt = \lambda_h \left[\alpha(t_2) - \alpha(t_1)\right] \tag{14}$$

Thus, $\alpha(t)$ is an increasing function in t. Then, we have the following lemma.

Lemma 11. *The ordering of events in the dynamic available system dyn2 is same as in the static system ss0.*

Proof. Suppose there are two events E_1 and E_2 that happen in $dyn2$ such that $t_{E_1} < t_{E_2}$, that is, E_1 happen before E_2 in $dyn2$. By contradiction, assume that E_2 happen before E_1 in the frame of reference of the static system $ss0$. By equation 8, that implies, $\alpha(t_{E_2}) < \alpha(t_{E_1})$. However, this contradicts the fact that $\alpha(t)$ is an increasing function in t. □

Suppose that $B_{s,s+t}$ happens in $dyn2$. This implies that, in $dyn2$, for every honest block b_j proposed at $\tau_j^h \in [s, s+t]$, there exists some minimum time $t_0 > \tau_j^h + \Delta$ and some honest block b_i proposed at τ_i^h such that

$$D_i(t_0) \geq D_h(t_0 - \Delta) - D_h(\tau_i^h + \Delta).$$

Due to Lemma 11, events in the evolution of the blockchain in $dyn2$ during the interval $[\tau_i^h, t_0]$ happens in the same order in the static system $ss0$ during the time interval $[\alpha(\tau_i^h), \alpha(t_0)]$. That implies the depth of the fictitious honest tree at time t in the local clock of $dyn2$ is same as the depth of the same fictitious honest tree at time $\alpha(t)$ in the local clock of $ss0$. This equivalence also carries over for the adversarial trees. Then, analysing the race between the fictitious honest tree $\mathcal{T}_h(t)$ and the adversarial tree $\mathcal{T}_i(t)$ with respect to the local clock of $ss0$, we can write

$$D_i(\alpha(t_0)) \geq D_h(\alpha(t_0 - \Delta)) - D_h(\alpha(\tau_i^h + \Delta))$$

That implies b_j is not a Nakamoto block in the static system $ss0$ too. Since, b_j is any arbitrary honest block with $\tau_j^h \in [s, s+t]$, therefore this is true for all honest blocks j' with $\tau_{j'}^h \in \{s, s+t\}$. Hence, $B_{s,s+t}^{dyn2} = B_{\alpha(s),\alpha(s+t)}^{ss0}$ which implies $P(B_{s,s+t}^{dyn2}) = P(B_{\alpha(s),\alpha(s+t)}^{ss0})$. This concludes our lemma.

E Proof of Lemma 5

For simulating the static system $ss1$, keep the sample path of the progress of the fictitious honest tree in both static systems $ss0$ and $ss1$ same. For some $t > 0$ in the local clock of $dyn2$, let $\mathcal{T}_i(t)$ represent the adversarial tree in $ss0$ with b_i as its root. Suppose $B_{\alpha(s),\alpha(s+t)}^{ss0}$ happens in $ss0$ for some $s, t > 0$ defined in the local clock of $dyn2$. That implies, for every honest block b_j proposed at

$\alpha(\tau_j^h) \in [\alpha(s), \alpha(s+t)]$, there exists some minimum time $\alpha(t_0) > \alpha(\tau_j^h + \Delta)$ and some honest block b_i proposed at $\alpha(\tau_i^h)$ such that

$$D_i(\alpha(t_0)) \geq D_h(\alpha(t_0 - \Delta)) - D_h(\alpha(\tau_i^h + \Delta)).$$

Now, for any arbitrary b_j, there are two cases:

1. If the tip of the fictitious honest tree at time $\alpha(t_0 - \Delta)$ is in the same epoch as the honest block b_i, then, the adversary can duplicate the first block in the adversarial tree $\mathcal{T}_i(t)$ of the static system $ss0$ and attach it to the block b_i of the simulated system $ss1$. However, in $ss1$, the adversary immediately gets a gift of $c-1$ blocks.
2. If the fictitious honest tree at time $\alpha(t_0 - \Delta)$ is in a different epoch as the honest block b_i, then, the adversary can duplicate the $\mathcal{T}_i(t)$ and prune it to contain all the blocks starting from the epoch that comes immediately after the epoch containing b_i in $\mathcal{T}_i(t)$. Then, in $ss1$, the adversary duplicates the first block in the adversarial tree $\mathcal{T}_i(t)$ of the static system $ss0$ and attaches it to the block b_i of the simulated system $ss1$ that immediately gifts a chain of $c-1$ blocks. The adversary then appends over that chain the pruned $\mathcal{T}_i(t)$.

Both cases clearly imply that at time $\alpha(t_0)$, there is an adversarial tree on b_i in $ss1$ whose depth is greater than $\mathcal{T}_i(t_0)$ in $ss0$. Thus, b_j is not a Nakamoto block in $ss1$. Hence, $P(B^{ss0}_{\alpha(s),\alpha(s+t)}) \leq P(B^{ss1}_{\alpha(s),\alpha(s+t)})$.

F Growth rate of Adversarial Tree $\hat{\mathcal{T}}_i(t)$

We first give a description of the (dual of the) adversarial tree consisting of super-blocks in terms of a Branching Random Walk (BRW).

Observe that due to the assumption on adversary in $ss2$, each adversarial tree $\hat{\mathcal{T}}_i(t')$ (with i^{th} honest block as its root), when analysed in the local clock of $ss2$, grows statistically in the same way, without any dependence on the level of the root. Without loss of generality, let us focus on the adversary tree $\hat{\mathcal{T}}_0(t')$, rooted at genesis. The genesis block is always at depth 0 and hence $\hat{\mathcal{T}}_0(0)$ has depth zero.

We can transform the tree $\hat{\mathcal{T}}_0(t')$ into a new random tree $\hat{\mathcal{T}}_0^s(t')$. Every c generations in $\hat{\mathcal{T}}_0(t')$ can be viewed as a single generation in $\hat{\mathcal{T}}_0^s(t')$. Thus, every block in $\hat{\mathcal{T}}_0^s(t')$, termed as *superblocks*, is representative of c blocks in $\hat{\mathcal{T}}_0(t')$. Consider B_0 to be the root of $\hat{\mathcal{T}}_0^s(t')$. The children blocks of B_0 in $\hat{\mathcal{T}}_0^s(t')$ are the descendent blocks at level c in $\hat{\mathcal{T}}_0(t')$. We can order these children blocks of B_0 in terms of their arrival times. Then, as the blocks in first $c-1$ levels of $\hat{\mathcal{T}}_0(t')$ are gift, the adversary didn't have to compute RandVDF.Eval for these blocks. Consider block B_1 to be the first block for which RandVDF.Eval was computed by the adversary. Therefore, the arrival time Q_1 of block B_1 is given by X_1 where X_1 is an exponential random variable in the static system $ss2$. On the other hand, the arrival time of the first child of B_1, call it $B_{1,1}$, is given by $Q_{1,1} = Q_1 + X_{1,1} + \cdots + X_{1,c}$, where $X_{1,i}$ is the inter-arrival time between the

$(i-1)^{th}$ and i^{th} descendent block of B_1. Note that, in the static system $ss2$, all the $X_{1,i}$'s are exponential with parameter λ_a, and they all are independent. Let the depth of the tree $\hat{\mathcal{T}}_0^s(t')$ be $D_0^s(t')$.

Each vertices at generation $k \geq 2$ in $\hat{\mathcal{T}}_0^s(t')$ can be labelled as a k tuple of positive integers $(i_1, \ldots, i_k)$ with $i_j \geq c$ for $2 \leq j \leq k$: the vertex $v = (i_1, \ldots, i_k) \in \mathcal{I}_k$ is the $(i_k - c + 1)$-th child of vertex $(i_1, \ldots, i_{k-1})$ at level $k-1$. At $k = 1$ generation, we have $i_1 \geq 1$ as the adversary is gifted $c-1$ blocks on proposing the first block for which it computes only one RandVDF.Eval. Let $\mathcal{I}_k = \{(i_1, \ldots, i_k) : i_j \geq 1 \text{ for } i_j = 1 \text{ and } i_j \geq c \text{ for } 2 \leq j \leq k\}$, and set $\mathcal{I} = \cup_{k>0} \mathcal{I}_k$. For such v we also let $v^j = (i_1, \ldots, i_j)$, $j = 1, \ldots, k$, denote the ancestor of v at level j, with $v^k = v$. For notation convenience, we set $v^0 = 0$ as the root of the tree.

Next, let $\{\mathcal{E}_v\}_{v \in \mathcal{I}}$ be an i.i.d. family of exponential random variables of parameter λ_a. For $v = (i_1, \ldots, i_k) \in \mathcal{I}_k$, let $\mathcal{W}_v = \sum_{j \leq i_k} \mathcal{E}_{(i_1, \ldots, i_{k-1}, j)}$ and let $Q_v = \sum_{j \leq k} \mathcal{W}_{v^j}$. This creates a labelled tree, with the following interpretation: for $v = (i_1, \ldots, i_j)$, the W_{v^j} are the waiting time for v^j to appear, measured from the appearance of v^{j-1}, and Q_v is the appearance time of v. Observe that starting from any $v \in \mathcal{I}_1$, we obtain a standard BRW. For any $v = (i_1, \cdots, i_k) \in \mathcal{I}_k$, we can write $Q_v = Q_v^1 + Q_v^2$ where Q_v^1 is the appearance time for the ancestor of v at level 1 while $Q_v^2 = Q_v - Q_v^1$.

Let $Q_k^* = \min_{v \in \mathcal{I}_k} Q_v$. Note that Q_k^* is the time of appearance of a block at level k and therefore we have

$$\{D_0(t') \geq ck\} = \{D_0^s(t') \geq k\} = \{Q_k^* \leq t'\}. \tag{15}$$

Fixing $i_1 \in \mathcal{I}_1$, let $Q_{k,i_1}^{2*} = \min_{v \in \mathcal{I}_k \text{ s.t. } v^1 = i_1} Q_v^2$. Observe that Q_{k,i_1}^{2*} is the minimum of a standard BRW with its root at the vertex i_1. Introduce, for $\theta_c < 0$, the moment generating function

$$\begin{aligned} \Lambda_c(\theta_c) &= \log \sum_{\substack{v \in \mathcal{I}_2 \\ v^1 = i_1}} E(e^{\theta_c Q_v^2}) = \log \sum_{j=c}^{\infty} E(e^{\sum_{i=1}^{j} \theta_c \mathcal{E}_i}) \\ &= \log \sum_{j=c}^{\infty} (E(e^{\theta_c \mathcal{E}_1}))^j = \log \frac{E^c(e^{\theta_c \mathcal{E}_1})}{1 - E(e^{\theta_c \mathcal{E}_1})}. \end{aligned}$$

Due to the exponential law of $\mathcal{E}_1$, $E(e^{\theta_c \mathcal{E}_1}) = \frac{\lambda_a}{\lambda_a - \theta_c}$ and therefore $\Lambda_c(\theta_c) = \log(-\lambda_a^c / \theta_c (\lambda_a - \theta_c)^{c-1})$.

An important role is played by θ_c^*, which is the negative solution to the equation $\Lambda_c(\theta_c) = \theta_c \dot{\Lambda}_c(\theta_c)$ and let η_c satisfy that

$$\sup_{\theta_c < 0} \left(\frac{\Lambda_c(\theta_c)}{\theta_c} \right) = \frac{\Lambda_c(\theta_c^*)}{\theta_c^*} = \frac{1}{\lambda_a \eta_c}.$$

Indeed, we have the following.

Proposition 1.

$$\lim_{k\to\infty}\frac{Q_k^*}{k}=\lim_{k\to\infty}\frac{Q_{k,i_1}^{2*}}{k}=\sup_{\theta_c<0}\left(\frac{\Lambda_c(\theta_c)}{\theta_c}\right)=\frac{1}{\lambda_a\eta_c},\quad a.s.$$

In fact, much more is known.

Proposition 2. *There exist explicit constants $c_1>c_2>0$ so that the sequence $Q_k^*-k/\lambda_a\eta_c-c_1\log k$ is tight, and*

$$\liminf_{k\to\infty}Q_k^*-k/\lambda_a\eta_c-c_2\log k=\infty, a.s.$$

Note that Propositions 1,2 and (15) imply in particular that $D_0(t')\le c\eta_c\lambda_a t'$ for all large t', a.s., and also that

$$\text{if } c\eta_c\lambda_a>\lambda_h \text{ then } D_i(t')>t' \text{ for all large } t', \text{ a.s..} \tag{16}$$

Let us define $\phi_c:=c\eta_c$, then $\phi_c\lambda_a$ is the growth rate of private c-correlated NaS tree. With all these preparations, we can give a simple proof for Lemma 6.

Proof. Consider $m=\eta_c\lambda_a t'+x$. Note that by (15),

$$\begin{aligned}P(D_0^s(t')\ge m)=P(Q_m^*\le t')&\le\sum_{v\in\mathcal{I}_m}P(Q_v\le t')=\sum_{v\in\mathcal{I}_m}P(Q_v^1+Q_v^2\le t')\\&=\sum_{v\in\mathcal{I}_m}\int_0^{t'}p_{Q_v^1}(u)P(Q_v^2\le t'-u)du\\&=\sum_{i_1\ge1}\sum_{i_2\ge c,\dots,i_m\ge c}\int_0^{t'}p_{Q_v^1}(u)P(Q_v^2\le t'-u)du\end{aligned}\tag{17}$$

For $v=(i_1,\dots,i_k)$, set $|v_{-1}|=i_2+\cdots+i_k$. Then, we have that Q_v^2 has the same law as $\sum_{j=1}^{|v_{-1}|}\mathcal{E}_j$. Thus, by Chebycheff's inequality, for $v\in\mathcal{I}_m$,

$$P(Q_v^2\le t'-u)\le Ee^{\theta_c^*Q_v^2}e^{-\theta_c^*(t'-u)}=\left(\frac{\lambda_a}{\lambda_a-\theta_c^*}\right)^{|v_{-1}|}e^{-\theta_c^*(t'-u)}.\tag{18}$$

And

$$\begin{aligned}\sum_{i_2\ge c,\dots,i_m\ge c}\left(\frac{\lambda_a}{\lambda_a-\theta_c^*}\right)^{|v_{-1}|}&=\sum_{i_2\ge c,\dots,i_m\ge c}\left(\frac{\lambda_a}{\lambda_a-\theta_c^*}\right)^{\sum_{j=2}^m i_j}\\&=\left(\sum_{i\ge c}\left(\frac{\lambda_a}{\lambda_a-\theta_c^*}\right)^i\right)^{m-1}=e^{(m-1)\Lambda_c(\theta_c^*)}.\end{aligned}\tag{19}$$

Combining (18), (19) and (17) yields

$$\begin{aligned} P(D_0^s(t') \geq m) &\leq e^{-\theta_c^* t'} e^{(m-1)\Lambda_c(\theta_c^*)} \sum_{i_1 \geq 1} \int_0^{t'} p_{Q_v^1}(u) e^{\theta_c^* u} du \\ &= e^{-\theta_c^* t'} e^{(m-1)\Lambda_c(\theta_c^*)} \sum_{i_1 \geq 1} \int_0^{t'} \frac{\lambda_a^{i_1} u^{i_1-1} e^{-\lambda_a u}}{\Gamma(i_1)} e^{\theta_c^* u} du \\ &= e^{-\theta_c^* t'} e^{(m-1)\Lambda_c(\theta_c^*)} g(t'). \end{aligned} \tag{20}$$

where $g(t') = \sum_{i_1 \geq 1} \int_0^{t'} \frac{\lambda_a^{i_1} u^{i_1-1} e^{-\lambda_a u}}{\Gamma(i_1)} e^{\theta_c^* u} du$. □

From proposition 1, we have

$$\phi_c = \frac{c\theta_c^*}{\lambda_a} \left(\frac{1}{\log\left(\frac{-\lambda_a^c}{\theta_c^*(\lambda_a - \theta_c^*)^{c-1}} \right)} \right), \tag{21}$$

where θ_c^* is the unique negative solution of

$$\Lambda_c(\theta_c) = \theta_c \dot{\Lambda}_c(\theta_c) \tag{22}$$

Note that $g(t')$ is an increasing function on t' and

$$\lim_{t' \to \infty} g(t') = \sum_{i_1 \geq 1} \left(\frac{\lambda_a}{\lambda_a - \theta_c^\star} \right)^{i_1} = \frac{\lambda_a}{-\theta_c^\star} \tag{23}$$

G Proofs

G.1 Definitions and Preliminary Lemmas

In this section, we define some important events which will appear frequently in the analysis and provide some useful lemmas.

Let V_j be the event that the $j-$th honest block b_j is a loner, i.e.,

$$V_j = \{\tau_{j-1}^h < \tau_j^h - \Delta'\} \bigcap \{\tau_{j+1}^h > \tau_j^h + \Delta'\}$$

Let $\hat{F}_j = V_j \bigcap F_j$ be the event that b_j is a Nakamoto block. Then, we can define the following "potential" catch up event in $ss2$:

$$\hat{B}_{ik} = \{D_i(\tau_k^h + \Delta') \geq D_h(\tau_{k-1}^h) - D_h(\tau_i^h + \Delta')\}, \tag{24}$$

which is the event that the adversary launches a private attack starting from honest block b_i and catches up the fictitious honest chain right before honest block b_k is mined.

Lemma 12. *For each j,*

$$P(\hat{F}_j^c) = P(F_j^c \cup V_j^c) \leq P\left(\left(\bigcup_{(i,k):0\leq i<j<k} \hat{B}_{ik}\right) \cup V_j^c\right). \tag{25}$$

Proof.

$$\begin{aligned}
&P(V_j \cap E_{ij}) \\
&= P(V_j \cap \{D_i(t') < D_h(t' - \Delta') - D_h(\tau_i^h + \Delta') \text{ for all } t' > \tau_j^h + \Delta'\}) \\
&= P(V_j \cap \{D_i(t' + \Delta') < D_h(t') - D_h(\tau_i^h + \Delta') \text{ for all } t' > \tau_j^h\}\ ) \\
&= P(V_j \cap \{D_i(\tau_k^{h^-} + \Delta') < D_h(\tau_k^{h^-}) - D_h(\tau_i^h + \Delta') \text{ for all } k > j\}) \\
&= P(V_j \cap \{D_i(\tau_k^h + \Delta') < D_h(\tau_{k-1}^h) - D_h(\tau_i^h + \Delta') \text{ for all } k > j\}).
\end{aligned}$$

Since $\hat{F}_j = F_j \cap V_j = \bigcap_{0\leq i<j} E_{ij} \cap V_j$, by the definition of $\hat{B}_{ik}$ we have $P(\hat{F}_j) \geq P\left(\left(\bigcap_{(i,k):0\leq i<j<k} \hat{B}_{ik}^c\right) \cap V_j\right)$. Taking complement on both side, we can conclude the proof. □

Let $R_m = \tau_{m+1}^h - \tau_m^h$. Then, V_j and $\hat{B}_{ik}$ can be re-written as:

$$\begin{aligned}
V_j &= \{\Delta' < R_{j-1}\} \bigcap \{R_j > \Delta'\} \\
\hat{B}_{ik} &= \left\{ D_i(\tau_i^h + \sum_{m=i}^{k-1} R_m + \Delta') \geq D_h(\tau_{k-1}^h) - D_h(\tau_i^h + \Delta') \right\}
\end{aligned} \tag{26}$$

Remark 1. By time-warping, R_m is an IID exponential random variable with rate λ_h.

Define X_d, $d > 0$, as the time it takes in the local clock of static system $ss2$ for D_h to reach depth d after reaching depth $d-1$. In other words, X_d is the difference between the times t_1 and t_2, where t_1 is the minimum time t' in the local clock of $ss2$ such that $D_h(t') = d$, and, t_2 is the minimum time t' in the local clock of $ss2$ such that $D_h(t') = d-1$.

Also, let $\delta_j^h = \tau_j^h - \tau_{j-1}^h$ and $\delta_j^a = \tau_j^a - \tau_{j-1}^a$ denote the inter-arrival time for honest and adversary arrival events in the local clock of static system $ss2$, respectively.

Proposition 3. *Let Y_d, $d \geq 1$, be i.i.d random variables, exponentially distributed with rate λ_h. Then, each random variable X_d can be expressed as $\Delta' + Y_d$.*

See Proposition $C.1$ in [12] for the proof.

Proposition 4. *For any constant a,*

$$P(\sum_{d=a}^{n+a} X_d > n(\Delta' + \frac{1}{\lambda_h})(1+\delta)) \leq e^{-n\Omega(\delta^2(1+\Delta'\lambda_h)^2)}$$

Proposition 4 is proved using chernoff bound and Proposition 3.

Proposition 5. *Probability that there are less than*

$$n\frac{\lambda_a(1-\delta)}{\lambda_h}$$

adversarial arrival events for which RandVDF.Eval *has been computed in the interval* τ_0^h *to* τ_{n+1}^h *is upper bounded by*

$$e^{-n\Omega(\delta^2\frac{\lambda_a}{\lambda_h})}$$

Proposition 5 is proven using the Poisson tail bounds.

Proposition 6. *For* $n > \frac{c-1}{\phi_c-1}$, *define* B_n *as the event that there are at least* n *adversarial block arrivals for each of which adversary computed* RandVDF.Eval *while* D_h *grows from depth* 0 *to* $n+c-1$*:*

$$B_n = \{ \sum_{i=1}^{n+c-1} X_i \geq \sum_{i=0}^{n} \delta_i^a \}$$

If

$$\phi_c\lambda_a < \frac{\lambda_h}{1+\lambda_h\Delta'},$$

then,

$$P(B_n) \leq e^{-A_1 n}e^{-A_2}$$

,

$$A_1 = -w\Delta' + \ln\left(\frac{\lambda_a+w}{\lambda_a}\right) + \ln\left(\frac{\lambda_h-w}{\lambda_h}\right)$$
$$A_2 = -(c-1)w\Delta' + (c-1)\ln\left(\frac{\lambda_h-w}{\lambda_h}\right)$$

such that $A_1 + \frac{A_2}{n} > 0$ *and,*

$$w = \frac{\lambda_h-\lambda_a}{2} + \frac{2n+c-1}{2(n+c-1)\Delta'} - \frac{\sqrt{[(n+c-1)\Delta'(\lambda_a-\lambda_h)]^2 + (2n+c-1)^2 + 2(n+c-1)\Delta'[(c-1)(\lambda_a+\lambda_h)+2(n+c-1)\Delta'\lambda_a\lambda_h]}}{2(n+c-1)\Delta'}$$

Proof. Using Chebychev inequality and proposition 3, for any $t > 0$, we have

$$\begin{aligned} P(B_n) &\leq E\left[\prod_{j=0}^{n} e^{-w\delta_i^a}\right] E\left[\prod_{j=1}^{n+c-1} e^{wX_i}\right] \\ &\leq \left[\frac{\lambda_a}{\lambda_a+w}\right]^n \left[\frac{e^{w\Delta'}\lambda_h}{\lambda_h-w}\right]^{n+c-1} \\ &= e^{-n\left[-\left(\frac{n+c-1}{n}\right)w\Delta' + \left(\frac{n+c-1}{n}\right)\ln\left(\frac{\lambda_h-w}{\lambda_h}\right) + \ln\left(\frac{\lambda_a+w}{\lambda_a}\right)\right]} \end{aligned}$$

Optimizing over w, we have

$$\frac{d}{dw}\left[-\left(\frac{n+c-1}{n}\right)w\Delta' + \left(\frac{n+c-1}{n}\right)\ln\left(\frac{\lambda_h - w}{\lambda_h}\right) + \ln\left(\frac{\lambda_a + w}{\lambda_a}\right)\right] = 0$$

$$(n+c-1)\Delta' w^2 + [(n+c-1)\Delta'(\lambda_a - \lambda_h) - (2n+c-1)]w + [n\lambda_h - (n+c-1)\lambda_a - (n+c-1)\Delta'\lambda_a\lambda_h] = 0$$

$$w = \frac{\lambda_h - \lambda_a}{2} + \frac{2n+c-1}{2(n+c-1)\Delta'} - \frac{\sqrt{[(n+c-1)\Delta'(\lambda_a - \lambda_h)]^2 + (2n+c-1)^2 + 2(n+c-1)\Delta'[(c-1)(\lambda_a+\lambda_h) + 2(n+c-1)\Delta'\lambda_a\lambda_h]}}{2(n+c-1)\Delta'}$$

Note that for $n > \frac{c-1}{\phi_c - 1}$, we have $\lambda_a\left(1 + \frac{c-1}{n}\right) < \phi_c\lambda_h < \frac{\lambda_h}{1+\Delta'\lambda_h}$. That implies $w > 0$.

Also, using $n > \frac{c-1}{\phi_c - 1}$, we have

$$-\left(\frac{n+c-1}{n}\right)w\Delta' + \left(\frac{n+c-1}{n}\right)\ln\left(\frac{\lambda_h - w}{\lambda_h}\right) + \ln\left(\frac{\lambda_a + w}{\lambda_a}\right) = A_1 + \frac{A_2}{n} > 0$$

□

Lemma 13. *For $k - i > \frac{\lambda_h(c-1)}{\lambda_a(\phi_c - 1)}$, there exists a constant $\gamma > 0$ such that*

$$P(\hat{B}_{ik}) \leq e^{-\gamma(k-i)} \tag{27}$$

Proof. Let $N(\tau_i^h, \tau_k^h + \Delta')$ be the number of adversarial arrivals for which RandVDF.Eval in was computed in $ss2$ in the interval $[\tau_i^h, \tau_k^h + \Delta']$. Define

$$\hat{C}_{ik} = \text{ event that } N(\tau_i^h, \tau_k^h + \Delta') + (c-1) \geq D_h(\tau_{k-1}^h) - D_h(\tau_i^h + \Delta')$$

Observe that $D_i(\tau_i^h, \tau_k^h + \Delta') \leq N(\tau_i^h, \tau_k^h + \Delta') + (c-1)$, where $c-1$ is due to the fact that blocks in first $c-1$ levels are gifted to the adversary on proposing the first block in the adversarial tree. Note that RandVDF.Eval was not computed by the adversary for these $c-1$ blocks. Then, we have

$$\hat{B}_{ik} \subseteq \hat{C}_{ik}.$$

$$
\begin{aligned}
P(\hat{B}_{ik}) \;&\leq\; P\left(N(\tau_i^h, \tau_k^h + \Delta') < (1-\delta)(k-i)\frac{\lambda_a}{\lambda_h}\right) \\
&\quad + P\left(\hat{C}_{ik} \mid N(\tau_i^h, \tau_k^h + \Delta') \geq (1-\delta)(k-i)\frac{\lambda_a}{\lambda_h}\right) \\
&\overset{(a)}{\leq}\; e^{-\Omega((k-i)\delta^2\lambda_a/\lambda_h)} + P\left(\hat{C}_{ik} \mid N_a(\tau_i^h, \tau_k^h + \Delta') \geq (1-\delta)(k-i)\frac{\lambda_a}{\lambda_h}\right) \\
&\overset{(b)}{\leq}\; e^{-\Omega((k-i)\delta^2\lambda_a/\lambda_h)} + \\
&\qquad \sum_{x=(1-\delta)(k-i)\frac{\lambda_a}{\lambda_h}}^{\infty} P\left(D_h(\tau_{k-1}^h) - D_h(\tau_i^h + \Delta') \leq x + c - 1 \mid N_a(\tau_i^h, \tau_k^h + \Delta') = x\right) \\
&\overset{(c)}{=}\; e^{-\Omega((k-i)\delta^2\lambda_a/\lambda_h)} + \sum_{x=(1-\delta)(k-i)\frac{\lambda_a}{\lambda_h}}^{\infty} e^{-A_1 x} e^{-A_2} \\
&\overset{(d)}{=}\; e^{-\Omega((k-i)\delta^2\lambda_a/\lambda_h)} + e^{-A_2}\frac{1}{1-e^{-A_3}} e^{-A_3(k-i)}
\end{aligned}
$$

where (a) is due to proposition 5 which says that there are more than $(1-\delta)(k-i)\lambda_a/\lambda_h$ adversarial arrival events in the time period $[\tau_i^h, \tau_k^h + \Delta']$ except with probability $e^{-\Omega((k-i)\delta^2\lambda_a/\lambda_h)}$, (b) is by union bound, (c) is by proposition 6 for $k - i > \frac{\lambda_h(c-1)}{\lambda_a(\phi_c - 1)}$, (d) is due to $A_3 = \frac{A_1(1-\delta)\lambda_a}{\lambda_h}$.

Hence,

$$P(\hat{B}_{ik}) < C_1 e^{-C_2(k-i)} \tag{28}$$

for appropriately chosen constants $C_1, C_2, > 0$ as functions of the fixed δ. Finally, since $P(\hat{B}_{ik})$ decreases as $k - i$ grows and is smaller than 1 for sufficiently large $k - i$, we obtain the desired inequality for a sufficiently small $\gamma \leq C_3$.

□

G.2 Proof of Lemma 7

For notational convenience, we will continue to use τ_i^h and τ_i^a as the arrival time of the $i-th$ honest and adversarial blocks in the static system $ss2$, respectively. In this proof, let $r_h := \frac{\lambda_h}{1+\lambda_h \Delta'}$. The random processes of interest start from time 0. To look at the system in stationarity, let us extend them to $-\infty < t' < \infty$. More specifically, define $\tau_{-1}^h, \tau_{-2}^h, \ldots$ such that together with $\tau_0^h, \tau_1^h, \ldots$, we have a double-sided infinite random process. Also, for each $i < 0$, we define an independent copy of a random adversary tree $\hat{\mathcal{T}}_i$ with the same distribution as $\hat{\mathcal{T}}_0$. And we extend the definition of $\hat{\mathcal{T}}_h(t')$ and $D_h(t')$ to $t' < 0$: the last honest block mined at $\tau_{-1}^h < 0$ and all honest blocks mined within $(\tau_{-1}^h - \Delta', \tau_{-1}^h)$ appear in $\hat{\mathcal{T}}_h(t')$ at their respective mining times to form the level -1, and the process repeats for level less than -1; let $D_h(t')$ be the level of the last honest arrival before t' in $\hat{\mathcal{T}}_h(t')$, i.e., $D_h(t') = \ell$ if $\tau_i^h \leq t' < \tau_{i+1}^h$ and the i-th honest block appears at level ℓ of $\hat{\mathcal{T}}_h(t)$.

These extensions allow us to extend the definition of E_{ij} to all i, j, $-\infty < i < j < \infty$, and define E_j and $\hat{E}_j$ to be:

$$E_j = \bigcap_{i<j} E_{ij}$$

and

$$\hat{E}_j = E_j \cap V_j.$$

Note that $\hat{E}_j \subset \hat{F}_j$, so to prove that $\hat{F}_j$ has a probability bounded away from 0 for all j, all we need is to prove that $\hat{E}_j$ has a non-zero probability.

Recall that we have defined the events V_j and $\hat{B}_{ik}$ in section G.1 of the appendix as:

$$V_j = \{\Delta' < R_{j-1}\} \bigcap \{R_j > \Delta'\}$$
$$\hat{B}_{ik} = \left\{ D_i(\tau_i^h + \sum_{m=i}^{k-1} R_m + \Delta') \geq D_h(\tau_{k-1}^h) - D_h(\tau_i^h + \Delta') \right\}$$

where R_m are i.i.d exponential random variable with mean $\frac{1}{\lambda_h}$.

Following the idea in Lemma 12 and using Lemma 14 and 15, we have

$$P(E_j \cap V_j) = P\left(\bigcap_{i<j} E_{ij} \cap V_j\right) = P\left(\left(\bigcap_{i<j<k} \hat{B}_{ik}^c\right) \cap U_j\right).$$

where $E_j = \bigcap_{i<j<k} \hat{B}_{ik}^c$ and $\hat{E}_j = E_j \cap U_j$. So, we just need to prove that $\hat{E}_j$ has a non-zero probability. Observe that, due to constant adversarial and honest mining rate and the growth rate of the adversarial tree being independent of level of its root in the static system $ss2$, $\hat{E}_j$ has a time-invariant dependence on $\{\mathcal{Z}_i\}$, which means that $p = P(\hat{E}_j)$ does not depend on j. Then we can just focus on $P(\hat{E}_0)$. This is the last step to prove.

$$\begin{aligned} P(\hat{E}_0) &= P(E_0|U_0)P(U_0) \\ &= P(E_0|U_0)P(R_0 > \Delta')P(R_{-1} > \Delta') \\ &= e^{-2\lambda_h \Delta'} P(E_0|U_0). \end{aligned}$$

where we used Remark 1 in the last step. It remains to show that $P(E_0|U_0) > 0$. We have

$$E_0 = \text{event that } D_i(\sum_{m=i}^{k-1} R_m + \Delta' + \tau_i^h) < D_h(\tau_{k-1}^h) - D_h(\tau_i^h + \Delta') \text{ for all } k > 0 \text{ and } i < 0,$$

then

$$(E_0)^c = \bigcup_{k>0, i<0} \hat{B}_{ik}. \tag{29}$$

Let us fix a particular $n > 2\lambda_h \Delta' > 0$, and define:

$$G_n = \text{event that} D_m(3n/\lambda_h + \zeta_m^h) = 0 \quad \text{for } m = -n, -n+1, \ldots, -1, 0, +1, \ldots, n-1, n$$

Then

$$\begin{aligned} P(E_0|U_0) &\geq P(E_0|U_0, G_n) P(G_n|U_0) \\ &= \left(1 - P(\cup_{k>0, i<0} \hat{B}_{ik}|U_0, G_n)\right) P(G_n|U_0) \\ &\geq \left(1 - \sum_{k>0, i<0} P(\hat{B}_{ik}|U_0, G_n)\right) P(G_n|U_0) \\ &\geq (1 - a_n - b_n) P(G_n|U_0) \end{aligned} \tag{30}$$

where

$$a_n := \sum_{(i,k): -n \leq i < 0 < k \leq n} P(\hat{B}_{ik}|U_0, G_n) \tag{31}$$

$$b_n := \sum_{(i,k): i < -n \text{ or } k > n} P(\hat{B}_{ik}|U_0, G_n). \tag{32}$$

Consider two cases:
Case 1: $-n \leq i < 0 < k \leq n$:

$$\begin{aligned} P(\hat{B}_{ik}|U_0, G_n) &= P(\hat{B}_{ik}|U_0, G_n, \sum_{m=i}^{k-1} R_m + \Delta' \leq 3n/\lambda_h) \\ &\quad + P(\sum_{m=i}^{k-1} R_m + \Delta' > 3n/\lambda_h|U_0, G_n) \\ &\leq P(\sum_{m=i}^{k-1} R_m + \Delta' > 3n/\lambda_h|U_0, G_n) \\ &\leq P(\sum_{m=i}^{k-1} R_m > 5n/(2\lambda_h)|U_0) \\ &\leq P(\sum_{m=i}^{k-1} R_m > 5n/(2\lambda_h))/P(U_0) \\ &\leq A_5 e^{-\gamma_1 n} \end{aligned}$$

for some positive constants A_5, γ_1 independent of n, k, i. The last inequality follows from the fact that R_i's are iid exponential random variables of mean

$1/\lambda_h$. Summing these terms, we have:

$$\begin{aligned} a_n &= \sum_{(i,k):-n\leq i<0<k\leq n} P(B_{ik}|U_0, G_n) \\ &\leq \sum_{(i,k):-n\leq i<0<k\leq n} A_5 e^{-\alpha_1 n} := \bar{a}_n, \end{aligned}$$

which is bounded and moreover $\bar{a}_n \to 0$ as $n \to \infty$.

Case 2: $k > n$ or $i < -n$:

For $0 < \varepsilon < 1$, let us define event W_{ik}^{ε} to be:

$$W_{ik}^{\varepsilon} = \text{event that } D_h(\zeta_{k-1}^h) - D_h(\zeta_i^h + \Delta') \geq (1-\varepsilon)\tfrac{r_h}{\lambda_h}(k-i-1). \tag{33}$$

Then we have

$$P(\hat{B}_{ik}|U_0, G_n) \leq P(\hat{B}_{ik}|U_0, G_n, W_{ik}^{\varepsilon}) + P(W_{ik}^{\varepsilon\, c}|U_0, G_n).$$

We first bound $P(W_{ik}^{\varepsilon\, c}|U_0, G_n)$:

$$\begin{aligned} P(W_{ik}^{\varepsilon\, c}|U_0, G_n) &\leq P(W_{ik}^{\varepsilon\, c}|\zeta_{k-1}^h - \zeta_i^h - \Delta' > \frac{k-i-1}{(1+\varepsilon)\lambda_h}) \\ &\quad + P(\zeta_{k-1}^h - \zeta_i^h - \Delta' \leq \frac{k-i-1}{(1+\varepsilon)\lambda_h}) \\ &\leq P(W_{ik}^{\varepsilon\, c}|\zeta_{k-1}^h - \zeta_i^h - \Delta' > \frac{k-i-1}{(1+\varepsilon)\lambda_h}) + e^{-\Omega(\varepsilon^2(k-i-1))} \\ &\leq e^{-\Omega(\varepsilon^4(k-i-1))} + e^{-\Omega(\varepsilon^2(k-i-1))} \\ &\leq A_6 e^{-\gamma_2(k-i-1)} \end{aligned} \tag{34}$$

for some positive constants A_6, γ_2 independent of n, k, i, where the second inequality follows from the Erlang tail bound (as $\zeta_{k-1}^h - \zeta_i^h$ is sum of IID exponentials due to time-warping) and the third inequality follows from Proposition 4.

Meanwhile, we have

$$
\begin{aligned}
&P(\hat{B}_{ik}|U_0, G_n, W_{ik}^{\varepsilon}) \\
\leq\ &P(D_i(\sum_{m=i}^{k-1} R_m + \Delta' + \zeta_i^h) \geq (1-\varepsilon)\frac{r_h}{\lambda_h}(k-i-1)|U_0, G_n, W_{ik}^{\varepsilon}) \\
\leq\ &P(D_i(\sum_{m=i}^{k-1} R_m + \Delta' + \zeta_i^h) \geq (1-\varepsilon)\frac{r_h}{\lambda_h}(k-i-1) \\
&\qquad |U_0, G_n, W_{ik}^{\varepsilon}, \sum_{m=i}^{k-1} R_m + \Delta' \leq (k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}) \\
&\quad + P(\sum_{m=i}^{k-1} R_m + \Delta' > (k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}|U_0, G_n, W_{ik}^{\varepsilon}) \\
\overset{(a)}{\leq}\ &P(\sum_{m=i}^{k-1} R_m + \Delta' > (k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}|U_0, G_n, W_{ik}^{\varepsilon}) \\
&\quad + e^{-\theta_c^*(k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}+\left(\frac{(1-\varepsilon)}{c}\frac{r_h}{\lambda_h}(k-i-1)-1\right)\Lambda_c(\theta_c^*)} g\left((k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}\right) \\
\overset{(b)}{=}\ &P(\sum_{m=i}^{k-1} R_m + \Delta' > (k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}|U_0, G_n, W_{ik}^{\varepsilon}) \\
&\quad + e^{-\theta_c^*\frac{k-i-1}{\lambda_h}\left[\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}-(1-\varepsilon)\frac{r_h}{\phi_c\lambda_a}\right]} e^{-\Lambda_c(\theta_c^*)} g\left((k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}\right)
\end{aligned}
$$

where (a) follows from Lemma 6, (b) follows from $\frac{\Lambda_c(\theta_c^*)}{\theta_c^*} = \frac{1}{\lambda_a\eta_c} = \frac{c}{\phi_c\lambda_a}$. The first term can be bounded as:

$$
\begin{aligned}
&P(\sum_{m=i}^{k-1} R_m + \Delta' > (k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}|U_0, G_n, W_{ik}^{\varepsilon}) \\
=\ &P(\sum_{m=i}^{k-1} R_m + \Delta' > (k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}|U_0, W_{ik}^{\varepsilon}) \\
\leq\ &P(\sum_{m=i}^{k-1} R_m + \Delta' > (k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h})/P(U_0, W_{ik}^{\varepsilon}) \\
\leq\ &A_7 e^{-\gamma_3(k-i-1)}
\end{aligned}
$$

for some positive constants A_7, γ_3 independent of n, k, i. The last inequality follows from the fact that $(r_h + \phi_c\lambda_a)/(2\phi_c\lambda_a) > 1$ and the R_i's have mean $1/\lambda_h$, while $P(U_0, W_{ik}^{\varepsilon})$ is a event with high probability as we showed in (34).

Then we have

$$\begin{aligned}
&P(\hat{B}_{ik}|U_0, G_n) \\
&\leq A_6 e^{-\alpha_2(k-i-1)} \\
&+ e^{-\theta_c^*(k-i-1)\frac{r_h(1-\varepsilon)}{\lambda_h \phi_c \lambda_a}\left[\frac{r_h+\phi_c\lambda_a}{2(1-\epsilon)r_h}-1\right]} e^{-\Lambda_c(\theta_c^*)} g((k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}) \\
&+ A_7 e^{-\gamma_3(k-i-1)}. \qquad (35)
\end{aligned}$$

Summing these terms, we have:

$$\begin{aligned}
b_n &= \sum_{(i,k):i<-n \text{ or } k>n} P(\hat{B}_{ik}|U_0, G_n) \\
&\leq \sum_{(i,k):i<-n \text{ or } k>n} [A_6 e^{-\alpha_2(k-i-1)} \\
&\quad + e^{-\theta_c^*(k-i-1)\frac{r_h(1-\varepsilon)}{\lambda_h \phi_c \lambda_a}\left[\frac{r_h+\phi_c\lambda_a}{2(1-\epsilon)r_h}-1\right]} e^{-\Lambda_c(\theta_c^*)} g\left((k-i-1)\frac{r_h+\phi_c\lambda_a}{2\phi_c\lambda_a}\frac{1}{\lambda_h}\right) \\
&\quad + A_7 e^{-\gamma_3(k-i-1)}] \\
&:= \bar{b}_n
\end{aligned}$$

Here, from (23), $g(.) \to \frac{\lambda_a}{-\theta_c^\star}$ as $n \to \infty$. Therefore, $\bar{b}_n$ is bounded and moreover $\bar{b}_n \to 0$ as $n \to \infty$ when we set ε to be small enough such that $\frac{r_h+\phi_c\lambda_a}{2(1-\varepsilon)r_h} < 1$.

Substituting these bounds in (30) we finally get:

$$P(E_0|U_0) > [1 - (\bar{a}_n + \bar{b}_n)]P(G_n|U_0) \qquad (36)$$

By setting n sufficiently large such that $\bar{a}_n$ and $\bar{b}_n$ are sufficiently small, we conclude that $P(\hat{E}_0) > 0$.

G.3 Proof of Lemma 8

We divide the proof in to two steps. In the first step, we prove for $\varepsilon = 1/2$. Recall that we have defined event $\hat{B}_{ik}$ as:

$$\hat{B}_{ik} = \text{event that } D_i(\textstyle\sum_{m=i}^{k-1} R_m + \Delta' + \zeta_i^h) \geq D_h(\zeta_{k-1}^h) - D_h(\zeta_i^h + \Delta').$$

And by Lemma 14, 15, 12, we have

$$\hat{F}_j^c = F_j^c \cup V_j^c = \left(\bigcup_{(i,k):i<j<k} \hat{B}_{ik}\right) \cup V_j^c. \qquad (37)$$

For $t' > \max\left\{\left(\frac{2\lambda_h}{1-\eta}\right)^2\left(\frac{c-1}{\phi_c-1}\right)^2, \left[(c-1)\left(\Delta' + \frac{1}{\lambda_{\min}}\right)\right]^2\right\}$, we have $\frac{\sqrt{t'}}{2\lambda_h} > \frac{\lambda_h}{\lambda_a}\left(\frac{c-1}{\phi_c-1}\right)$ and $\sqrt{t'} > (c-1)\left(\Delta' + \frac{1}{\lambda_{\min}}\right)$.

Divide $[s', s'+t']$ into $\sqrt{t'}$ sub-intervals of length $\sqrt{t'}$, so that the r th sub-interval is:

$$\mathcal{J}_r := [s' + (r-1)\sqrt{t'}, s' + r\sqrt{t'}].$$

Now look at the first, fourth, seventh, etc sub-intervals, i.e. all the $r = 1 \mod 3$ sub-intervals. Introduce the event that in the ℓ-th $1 \mod 3$th sub-interval, an adversary tree that is rooted at a honest block arriving in that sub-interval or in the previous ($0 \mod 3$) sub-interval catches up with a honest block in that sub-interval or in the next ($2 \mod 3$) sub-interval. Formally,

$$C_\ell = \bigcap_{j:\zeta_j^h \in \mathcal{J}_{3\ell+1}} U_j^c \cup \left(\bigcup_{(i,k):\zeta_j^h - \sqrt{t'} < \zeta_i^h < \zeta_j^h, \zeta_j^h < \zeta_k^h + \Delta' < \zeta_j^h + \sqrt{t'}} \hat{B}_{ik} \right).$$

We have

$$P(C_\ell) \leq P(\text{no arrival in } \mathcal{J}_{3\ell+1}) + 1 - p < 1 \tag{38}$$

for large enough t', where p is a uniform lower bound such that $P(\hat{F}_j) \geq p$ for all j. Also, we define the following event:

$\hat{C}_\ell =$ event that the honest fictitious tree grows by $c-1$ levels in sub-interval $\mathcal{J}_{3\ell+2}$

Observe that because of `randSource` being updated at each epoch beginning, for distinct ℓ, the events $C_\ell \bigcap \hat{C}_\ell$ are independent. Using Poisson tail bounds, for $\sqrt{t'} > (c-1)\left(\Delta' + \frac{1}{\lambda_{\min}}\right)$, we have $P(\hat{C}_\ell) \geq 1 - e^{-c_2\sqrt{t'}}$.

Introduce the atypical events:

$$B = \bigcup_{(i,k):\zeta_i^h \in [s', s'+t'] \text{ or } \zeta_k^h + \Delta' \in [s', s'+t'], i<k, \zeta_k^h + \Delta' - \zeta_i^h > \sqrt{t'}} \hat{B}_{ik},$$

and

$$\tilde{B} = \bigcup_{(i,k):\zeta_i^h < s', s'+t' < \zeta_k^h + \Delta'} \hat{B}_{ik}.$$

The events B and $\tilde{B}$ are superset of the events that an adversary tree catches up with an honest block far ahead. Then we have

$$\begin{aligned}
P(B^{static}_{s',s'+t'}) &\leq P\Big(\bigcap_{j:\zeta_j^h \in [s', s'+t']} U_j^c\Big) + P(B) + P(\tilde{B}) + P\Big(\bigcap_{\ell=0}^{\sqrt{t'}/3} C_\ell\Big) \\
&\leq P\Big(\bigcap_{j:\zeta_j^h \in [s', s'+t']} U_j^c\Big) + P(B) + P(\tilde{B}) + P\Big(\bigcup_{\ell=0}^{\sqrt{t'}/3} \hat{C}_\ell^c\Big) + P\Big(\bigcap_{\ell=0}^{\sqrt{t'}/3} C_\ell \cap \hat{C}_\ell\Big) \\
&\leq P\Big(\bigcap_{j:\zeta_j^h \in [s', s'+t']} U_j^c\Big) + P(B) + P(\tilde{B}) + \sum_{\ell=0}^{\sqrt{t'}/3} P(\hat{C}_\ell^c) + (P(C_\ell \cap \hat{C}_\ell))^{\sqrt{t'}/3} \\
&\leq e^{-c_1 t'} + P(B) + P(\tilde{B}) + e^{-c_2\sqrt{t'}} + (P(C_\ell))^{\frac{\sqrt{t'}}{3}}
\end{aligned} \tag{39}$$

for some positive constants c_1, c_2 when t' is large. Next we will bound the atypical events B and $\tilde{B}$. Consider the following events

$$D_1 = \{\#\{i : \zeta_i^h \in (s' - \sqrt{t'} - \Delta', s' + t' + \sqrt{t'} + \Delta)\} > 2\lambda_h t'\}$$
$$D_2 = \{\exists i, k : \zeta_i^h \in (s', s' + t'), (k - i) < \frac{\sqrt{t'}}{2\lambda_h}, \zeta_k^h - \zeta_i^h + \Delta' > \sqrt{t'}\}$$
$$D_3 = \{\exists i, k : \zeta_k^h + \Delta \in (s', s' + t'), (k - i) < \frac{\sqrt{t'}}{2\lambda_h}, \zeta_k^h - \zeta_i^h + \Delta' > \sqrt{t}\}$$

In words, D_1 is the event of atypically many honest arrivals in $(s' - \sqrt{t'} - \Delta', s' + t' + \sqrt{t'} + \Delta')$ while D_2 and D_3 are the events that there exists an interval of length $\sqrt{t'}$ with at least one endpoint inside $(s', s' + t')$ with atypically small number of arrivals. Since, by time-warping, the number of honest arrivals in $(s', s' + t')$ (in the local clock of the static system) is Poisson with parameter $\lambda_h t'$, we have from the memoryless property of the Poisson process that $P(D_1) \leq e^{-c_0 t'}$ for some constant $c_0 = c_0(\lambda_a, \lambda_h) > 0$ when t' is large. On the other hand, using the memoryless property and a union bound, and decreasing c_0 if needed, we have that $P(D_2) \leq e^{-c_0\sqrt{t'}}$. Similarly, using time reversal, $P(D_3) \leq e^{-c_0\sqrt{t'}}$. Therefore, again using the memoryless property of the Poisson process,

$$\begin{aligned} P(B) &\leq P(D_1 \cup D_2 \cup D_3) + P(B \cap D_1^c \cap D_2^c \cap D_3^c) \\ &\leq e^{-c_0 t'} + 2e^{-c_0\sqrt{t'}} + \sum_{i=1}^{2\lambda_h t'} \sum_{k:k-i>\sqrt{t'}/2\lambda_h} P(\hat{B}_{ik}) \qquad (40) \\ &\leq e^{-c_3\sqrt{t'}}, \qquad (41) \end{aligned}$$

for large t', where $c_3 > 0$ are constants that may depend on λ_a, λ_h and the last inequality is due to (27). We next claim that there exists a constant $\alpha > 0$ such that, for all t' large,

$$P(\tilde{B}) \leq e^{-c_6 t'}. \qquad (42)$$

Consider the following event

$$D_4 = \{\exists i, k : (k - i) < \frac{t'}{2\lambda_h}, \zeta_k^h - \zeta_i^h + \Delta' > t'\}.$$

Using Poisson tail bounds, we can show that $P(D_4) \leq e^{-c_4 t'}$. Now, we have

$$\begin{aligned} &P(\tilde{B}) \leq P(D_4) + P(\tilde{B} \cap D_4^c) \\ &\leq e^{-c_4 t'} + \sum_{i,k:k-i>t'/2\lambda_h} \int_0^{s'} P(\zeta_i^h \in d\theta) P(\hat{B}_{ik}, \zeta_k^h - \zeta_i^h + \Delta' > s' + t' - \theta) \\ &\leq e^{-c_4 t'} + \sum_i \int_0^{s'} P(\zeta_i^h \in d\theta) \sum_{k:k-i>t'/2\lambda_h} P(\hat{B}_{ik})^{1/2} P(\zeta_k^h - \zeta_i^h + \Delta' > s' + t' - \theta)^{1/2}. \end{aligned} \qquad (43)$$

The tails of the Poisson distribution yield the existence of constants $c', c'' > 0$ so that

$$P(\zeta_k^h - \zeta_i^h + \Delta' > s' + t' - \theta) \tag{44}$$

$$\leq \begin{cases} 1, & (k-i) > c'(s'+t'-\theta-\Delta') \\ e^{-c''(s'+t'-\theta-\Delta')}, & (k-i) \leq c'(s'+t'-\theta-\Delta'). \end{cases} \tag{45}$$

(27) and (44) yield that, for large enough t', there exists a constant $c_5 > 0$ so that

$$\sum_{k:k-i>t'/2\lambda_h} P(\hat{B}_{i,k})^{1/2} P(\zeta_k^h - \zeta_i^h > s'+t'-\theta-\Delta')^{1/2} \leq e^{-2c_5(s'+t'-\theta-\Delta')}. \tag{46}$$

Substituting this bound in (43) and using that $\sum_i P(\zeta_i^h \in d\theta) = d\theta$ gives

$$\begin{aligned} P(\tilde{B}) &\leq e^{-c_4 t'} + \sum_i \int_0^{s'} P(\zeta_i^h \in d\theta) e^{-2c_5(s'+t'-\theta-\Delta')} \\ &\leq e^{-c_4 t'} + \int_0^{s'} e^{-2c_5(s'+t'-\theta-\Delta')} d\theta \leq e^{-c_4 t'} + \frac{1}{2c_5} e^{-2c_5(t'-\Delta')} \\ &\leq e^{-c_6 t'}, \end{aligned} \tag{47}$$

for t' large and $c_6 = \min(c_4, c_5)$, proving (42).

Combining (41), (47) and (39) concludes the proof of step 1.

In step two, we prove for any $\varepsilon > 0$ by recursively applying the bootstrapping procedure in step 1. Assume the following statement is true: for any $\theta \geq m$ there exist constants $\bar{b}_\theta, \bar{A}_\theta$ so that for all $s', t' \geq 0$,

$$\tilde{q}[s', s'+t'] \leq \bar{A}_\theta \exp(-\bar{b}_\theta t'^{1/\theta}). \tag{48}$$

By step 1, it holds for $m = 2$. Also, for specific values of m that we will consider, we will have $t'^{\frac{m}{2m-1}} > \sqrt{t'}$.

Divide $[s', s'+t']$ into $t'^{\frac{m-1}{2m-1}}$ sub-intervals of length $t'^{\frac{m}{2m-1}}$, so that the r th sub-interval is:

$$\mathcal{J}_r := [s' + (r-1)t'^{\frac{m}{2m-1}}, s' + rt'^{\frac{m}{2m-1}}].$$

Now look at the first, fourth, seventh, etc sub-intervals, i.e. all the $r = 1 \mod 3$ sub-intervals. Introduce the event that in the ℓ-th $1 \mod 3$th sub-interval, an adversary tree that is rooted at a honest block arriving in that sub-interval or in the previous $(0 \mod 3)$ sub-interval catches up with a honest block in that sub-interval or in the next $(2 \mod 3)$ sub-interval. Formally,

$$C_\ell = \bigcap_{j:\zeta_j^h \in \mathcal{J}_{3\ell+1}} U_j^c \cup \left(\bigcup_{(i,k):\zeta_j^h - t'^{\frac{m}{2m-1}} < \zeta_i^h < \zeta_j^h, \zeta_j^h < \zeta_k^h + \Delta' < \zeta_j^h + t'^{\frac{m}{2m-1}}} \hat{B}_{ik} \right).$$

By (48), we have

$$P(C_\ell) \leq A_m \exp(-\bar{a}_m t'^{\frac{1}{2m-1}}). \tag{49}$$

Also, we define the following event:

$\hat{C}_\ell =$ event that the honest fictitious tree grows by $c-1$ levels in sub-interval $\mathcal{J}_{3\ell+2}$

Note that for distinct ℓ, the events $C_\ell \bigcap \hat{C}_\ell$ are independent. Also, from Lemma 10, assuming $t'^{\frac{m}{2m-1}} > \sqrt{t'} > (c-1)\left(\Delta' + \frac{1}{\lambda_{\min}}\right)$, we have $P(\hat{C}_\ell) \geq 1 - e^{-c_2 t'^{\frac{m}{2m-1}}}$ for some positive constant c_2.

Introduce the atypical events:

$$B = \bigcup_{(i,k):\zeta_i^h \in [s',s'+t'] \text{ or } \zeta_k^h+\Delta' \in [s',s'+t'], i<k, \zeta_k^h+\Delta'-\zeta_i^h > t'^{\frac{m}{2m-1}}} \hat{B}_{ik},$$

and

$$\tilde{B} = \bigcup_{(i,k):\zeta_i^h < s', s'+t' < \zeta_k^h+\Delta'} \hat{B}_{ik}.$$

The events B and $\tilde{B}$ are the events that an adversary tree catches up with an honest block far ahead. Following the calculations in step 1, we have

$$P(B) \leq e^{-c_3 t'^{\frac{m}{2m-1}}} \tag{50}$$

$$P(\tilde{B}) \leq e^{-c_6 t'}, \tag{51}$$

for large t', where c_1 and c_5 are some positive constant.

Then we have

$$\begin{aligned}
\tilde{q}[s', s'+t'] &\leq P(\bigcap_{j:\zeta_j^h \in [s',s'+t']} U_j^c) + P(B) + P(\tilde{B}) + P(\bigcap_{\ell=0}^{t'^{\frac{m-1}{2m-1}}/3} C_\ell) \\
&\leq P(\bigcap_{j:\zeta_j^h \in [s',s'+t']} U_j^c) + P(B) + P(\tilde{B}) + P(\bigcup_{\ell=0}^{t'^{\frac{m-1}{2m-1}}/3} \hat{C}_\ell^c) + P(\bigcap_{\ell=0}^{t'^{\frac{m-1}{2m-1}}/3} C_\ell \cap \hat{C}_\ell) \\
&\leq P(\bigcap_{j:\zeta_j^h \in [s,s+t]} U_j^c) + P(B) + P(\tilde{B}) + \sum_{\ell=0}^{t'^{\frac{m-1}{2m-1}}/3} P(\hat{C}_\ell^c) + (P(C_\ell \cap \hat{C}_\ell))^{t'^{\frac{m-1}{2m-1}}/3} \\
&\leq e^{-c_1 t'} + e^{-c_3 t'^{\frac{m}{2m-1}}} + e^{-c_6 t'} + e^{-c_2 t'^{\frac{m}{2m-1}}} + (A_m \exp(-\bar{a}_m t'^{1/(2m-1)}))^{t'^{\frac{m-1}{2m-1}}/3} \\
&\leq \bar{A}'_m \exp(-\bar{b}'_m t'^{\frac{m}{2m-1}})
\end{aligned}$$

for large t', where A'_m and b'_m are some positive constant.

So we know the statement in (48) holds for all $\theta \geq \frac{2m-1}{m}$. Start with $m_1 = 2$, we have a recursion equation $m_k = \frac{2m_{k-1}-1}{m_{k-1}}$ and we know (48) holds for all $\theta \geq m_k$. It is not hard to see that $m_k = \frac{k+1}{k}$ and thus $\lim_{k\to\infty} m_k = 1$. Now observe that for $m_k = \frac{k+1}{k}$, we have $t'^{\frac{m_k}{2m_k-1}} > \sqrt{t'}$ for $k > 1$.

So, for some constant $\bar{a}_\theta$ which is a function of Δ', we can rewrite (48) as

$$\tilde{q}[\alpha(s), \alpha(s+t)] \leq \bar{A}'_m \exp(-\bar{a}_\theta t^{1/\theta})$$

which concludes the lemma.

H Proof of Lemma 9

Let U_j be the event in $ss1$ that the $j-$th honest block b_j is a loner, i.e.,

$$U_j = \{\tau^h_{j-1} < \tau^h_j - \Delta\} \bigcap \{\tau^h_{j+1} > \tau^h_j + \Delta\}$$

Let $\hat{F}_j = U_j \bigcap F_j$ be the event that b_j is a Nakamoto block. We define the following "potential" catch up event in $ss1$:

$$\hat{A}_{ik} = \{D_i(\alpha(\tau^h_k + \Delta)) \geq D_h(\alpha(\tau^h_{k-1})) - D_h(\alpha(\tau^h_i + \Delta))\}, \tag{52}$$

which is the event that the adversary launches a private attack starting from honest block b_i and catches up the fictitious honest chain right before honest block b_k is proposed.

Next, define the following events

$$V^{ss1}_j = \{\alpha(\tau^h_{j-1}) < \alpha(\tau^h_j) - \frac{\lambda_{\max}}{\lambda_h}\Delta\} \bigcap \{\alpha(\tau^h_{j+1}) > \alpha(\tau^h_j) + \frac{\lambda_{\max}}{\lambda_h}\Delta\} \tag{53}$$

$$\hat{B}^{ss1}_{ik} = \{D_i(\alpha(\tau^h_k) + \frac{\lambda_{\max}}{\lambda_h}\Delta) \geq D_h(\alpha(\tau^h_{k-1})) - D_h(\alpha(\tau^h_i) + \frac{\lambda_{\max}}{\lambda_h}\Delta)\} \tag{54}$$

Lemma 14. *For any pair of i, k,*

$$\hat{A}_{ik} \subseteq \hat{B}^{ss1}_{ik}.$$

Proof. Using equation 8, we have

$$\begin{aligned}\alpha(\tau^h_k + \Delta) &= \int_0^{\tau^h_k + \Delta} \frac{\lambda^c_h(u)}{\lambda_h} du = \int_0^{\tau^h_k} \frac{\lambda^c_h(u)}{\lambda_h} du + \int_{\tau^h_k}^{\tau^h_k + \Delta} \frac{\lambda^c_h(u)}{\lambda_h} du \\ &\leq \alpha(\tau^h_k) + \frac{\lambda_{\max}}{\lambda_h}\Delta\end{aligned}$$

Similarly, $\alpha(\tau^h_i + \Delta) \leq \alpha(\tau^h_i) + \frac{\lambda_{\max}}{\lambda_h}\Delta$. Because $D_h(.)$ and $D_i(.)$ are increasing functions over their domain, we have

$$\begin{aligned}D_i(\alpha(\tau^h_k + \Delta)) &\leq D_i(\alpha(\tau^h_k) + \frac{\lambda_{\max}}{\lambda_h}\Delta) \text{ and} \\ D_h(\alpha(\tau^h_i + \Delta)) &\leq D_h(\alpha(\tau^h_i) + \frac{\lambda_{\max}}{\lambda_h}\Delta)\end{aligned}$$

□

Lemma 15. *For all j,*

$$V_j^{ss1} \subseteq U_j.$$

Proof. This can be proved using the fact that $\int_{\tau_{j-1}^h}^{\tau_{j-1}^h+\Delta} \frac{\lambda_h^c(u)}{\lambda_h} du \leq \frac{\lambda_{\max}}{\lambda_h}\Delta$ and $\int_{\tau_j^h}^{\tau_j^h+\Delta} \frac{\lambda_h^c(u)}{\lambda_h} du \leq \frac{\lambda_{\max}}{\lambda_h}\Delta$. □

By time-warping, R_m is an IID exponential random variable with rate λ_h. Let $\zeta_j^h = \alpha(\tau_j^h)$, that is, ζ_j^h is the time of mining of $j-$th honest block in the local clock of static system $ss1$. Similarly, we define $\zeta_j^a = \alpha(\tau_j^a)$ for the $j-$th adversarial block. Then, we can rewrite the event $\hat{B}_{ik}$ as:

$$\hat{B}_{ik}^{ss1} = \left\{ D_i(\zeta_k^h + \frac{\lambda_{\max}}{\lambda_h}\Delta) \geq D_h(\zeta_{k-1}^h) - D_h(\zeta_i^h + \frac{\lambda_{\max}}{\lambda_h}\Delta) \right\}.$$

Lemma 16. *In the static system $ss1$, for each j*

$$P(\hat{F}_j^c) = P(F_j^c \cup U_j^c) \leq P\left(\left(\bigcup_{(i,k):0\leq i<j<k} \hat{B}_{ik}^{ss1} \right) \cup (V_j^{ss1})^c \right). \tag{55}$$

This can be proved in a similar way as Lemma 12 and using Lemma 14, 15. Furthermore, defining X_d, $d > 0$, as the time it takes in the local clock of static system $ss1$ for D_h to reach depth d after reaching depth $d-1$, we have

Proposition 7. *Let Y_d, $d \geq 1$, be i.i.d random variables, exponentially distributed with rate λ_h. Then, each random variable X_d is less than $\Delta' + Y_d$, where $\Delta' = \frac{\lambda_{\max}}{\lambda_h}\Delta$.*

Proof. Let h_i be the first block that comes at some depth $d-1$ within $\mathcal{T}_h$. Then, in the local clock of static system, every honest block that arrives within interval $[\alpha(\tau_i^h), \alpha(\tau_i^h + \Delta)]$ will be mapped to the same depth as h_i, i.e., $d-1$. Hence, $\mathcal{T}_h$ will reach depth d only when an honest block arrives after time $\alpha(\tau_i^h + \Delta)$. Now, due to time warping, in the local clock of static system $ss1$, we know that the difference between $\alpha(\tau_i^h + \Delta)$ and the arrival time of the first honest block after $\alpha(\tau_i^h + \Delta)$ is exponentially distributed with rate λ_h due to the memoryless property of the exponential distribution. This implies that for each depth d, $X_d = \alpha(\tau_i^h + \Delta) - \alpha(\tau_i^h) + Y_d = \int_{\tau_i^h}^{\tau_i^h+\Delta} \frac{\lambda_h^c(u)}{\lambda_h} du + Y_d \leq \Delta' + Y_d$ for some random variable Y_d such that $Y_d, d \geq 1$, are IID and exponentially distributed with rate λ_h. □

Thus, for $\Delta' = \frac{\lambda_{\max}}{\lambda_h}\Delta$, Proposition 7 implies that both Proposition 4 and Proposition 6 are satisfied for the static system $ss1$. Therefore, for $\Delta' = \frac{\lambda_{\max}}{\lambda_h}\Delta$, a similar result holds for the event $\hat{B}_{ik}^{ss1}$ as in Lemma 13. Additionally, Lemma 6 holds for $ss1$. Then, substituting $\Delta' = \frac{\lambda_{\max}}{\lambda_h}\Delta$ and using Lemma 16, we have both Lemma 7 and Lemma 8 satisfy for the static system $ss1$. For a time $t > 0$ in the local clock of the dynamic available system $dyn2$, we have $\alpha(t) \geq \frac{\lambda_{\min}}{\lambda_h}t$. Then, using Lemma 3, Lemma 4, Lemma 5, we conclude the proof.