© 2020 Xuechao Wang

PROOF-OF-STAKE LONGEST CHAIN PROTOCOLS: SECURITY VS PREDICTABILITY

BY

XUECHAO WANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Professor Pramod Viswanath

ABSTRACT

The Nakamoto longest chain protocol is remarkably simple and has been proven to provide security against any adversary with less than 50% of the total hashing power. Proof-of-stake (PoS) protocols are an energy efficient alternative; however existing protocols adopting Nakamoto's longest chain design achieve provable security only by allowing long-term predictability, subjecting the system to serious bribery attacks. In this thesis, we prove that a natural longest chain PoS protocol with predictability similar to that of Nakamoto's PoW protocol can achieve security against any adversary with less than $1/(1+e)$ fraction of the total stake. Moreover we propose a new family of longest chain PoS protocols that achieve security against a 50% adversary, while only requiring short-term predictability. Our proofs present a new approach to analyzing the formal security of blockchains, based on a notion of *Nakamoto block*.

To my parents, for their love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my thesis advisor Prof. Pramod Viswanath for his sincere and patient guidance and support, without which this thesis would not have been possible. His sharp thinking, deep intuition, and brilliant insights of research problems have made him the best role model as a good researcher. I also want to thank Prof. Sewoong Oh and Prof. Sreeram Kannan from University of Washington, and Prof. David Tse from Stanford University, who had fruitful discussions with me on this thesis.

I would like to thank Prof. Michael Bailey, Prof. Nikita Borisov and Prof. Andrew Miller for serving as my qualifying exam committee. I would not be able to pursue my Ph.D. without their help. I would like to thank all the faculty who taught me during the past two years, including Prof. Jozsef Balogh, Prof. Nikita Borisov, Prof. Partha Dey, Prof. Burak Erdoğan, Prof. Bruce Hajek, Prof. Olgica Milenkovic, Prof. Andrew Miller, Prof. Ling Ren, Prof. Alexander Schwing, Prof. Lav Varshney and Prof. Pramod Viswanath.

Finally, I would like to express deepest gratitude to my parents for providing me with selfless love, support and encouragement throughout my years of study. To them I dedicate this thesis.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Proof-of-work and proof-of-stake blockchains

Bitcoin is the original blockchain, invented by Nakamoto. At the core is the permissionless consensus problem, which Nakamoto solved with a remarkably simple but powerful scheme known as the longest chain protocol. It uses only basic cryptographic primitives (hash functions and digital signatures). In the seminal paper [1] that introduced the original Bitcoin protocol, Nakamoto also showed that the protocol is secure against one specific attack, a private double-spend attack, if the fraction of adversarial hashing power, β, is less than half the hashing power of the network. This attack is mounted by the adversary trying to grow a long chain over a long duration in private to replace the public chain. Subsequently, the security of Bitcoin against *all* possible attacks is proven in [2], and further extended to a more realistic network delay model in [3].

The permissionless design (robustness to Sybil attacks) of Bitcoin is achieved via a proof-of-work (PoW) mining process, but comes at the cost of large energy consumption. Recently proof-of-stake (PoS) protocols have emerged as an energy-efficient alternative. When running a lottery to win the right to propose the next valid block on the blockchain, each node wins with probability proportional its 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 families of PoS protocols: those derived from decades of research in Byzantine Fault Tolerant (BFT) protocols and those inspired by the Nakamoto longest chain protocol. Attempts at blockchain design via the BFT approach include Algorand [4, 5] and Hotstuff [6]. The adaptation of these new protocols into blockchains is an active area of research and engineering [5, 7], with large scale permissionless deployment as yet untested.

Motivated and inspired by the time-tested Nakamoto longest chain protocol are the PoS designs of Snow White [8] and the Ouroboros family of protocols [9, 10, 11]. The inherent energy efficiency of the PoS setting comes with the cost of enlarging the space of adversarial actions. In particular, the attacker can "grind" on the various sources of the randomness, i.e., attempt multiple samples from the sources of randomness to find a favorable one. Since these multiple attempts are without any cost to the attacker this strategy is also known as a ***nothing-at-stake*** (NaS) attack. One way to prevent an NaS attack is to rely on a source of randomness on which a consensus has been reached. In Snow White [8] and the Ouroboros family [9, 10, 11], this agreed upon randomness is derived from the stabilized segment of the blockchain from a few epochs before. Each epoch is a fixed set of consecutive PoS lottery slots that use the same source of agreed upon randomness. However, this comes at a price of allowing each individual node to simulate and predict in advance whether it is going to win the PoS lottery at a given slot and add a new block to the chain. Further, as the size of each epoch is proportional to the security parameter κ (specifically, a block is confirmed if and only if it is more than κ blocks deep in the blockchain), higher security necessarily implies that the nodes can predict further ahead into the future. This is a serious security concern, as predictability makes a protocol vulnerable against other types of attacks driven by incentives, such as predictable selfish mining or bribing attacks [12].

1.2 Nakamoto-PoS

A straightforward PoS adoption of Nakamoto protocol, which in contrast to Ouroboros and Snow White can update randomness every block, runs as follows; we term the protocol as Nakamoto-PoS. The protocol proceeds in discrete time units called ***slots***, during which each node runs the "PoS lottery", a leader election with a winning probability proportional to the stake owned by the node – winners get to ***propose*** new blocks. Each node computes $hash = H(\text{time}, \text{secret key}, \text{parentBk}.hash)$, where the hash function H is a ***veri able random function*** (VRF) (formally defined in Appendix A), which enables the nodes to run leader elections with their secret keys (the output $hash$ is verified with the corresponding public key). The node n is elected a leader if $hash$ is smaller than a threshold $\rho \times \mathsf{stake}_n$, that is proportional to its stake stake_n, then the node n proposes a new block consisting of time, parentBk.$hash$, public key and $hash$, and appends it to the parent block. A detailed algorithmic description of this protocol is in Appendix E (with $c = 1$). Following Nakamoto's protocol, each honest node runs *only one* election, appending to the last block in the longest chain in its local view. Having the hash function depend on parentBk.$hash$ ensures that every appended block provides a fresh source of randomness, for the following elections. However, there is no consensus on the randomness used and the randomness is block dependent, creating opportunities for the adversary to mount a NaS attack by trying its luck at many different blocks.

The analysis of the security of the Nakamoto-PoS protocol is first attempted in [13]. Just like Nakamoto's original analysis, their analysis is on the security against a specific attack: the private double-spend attack. Due to the NaS phenomenon, they showed the adversary can grow a private chain faster than just growing at the tip, as though its stake increases by a factor of e. This shows that the PoS longest chain protocol is secure against the private double-spend against if the adversarial fraction of stake $\beta < 1/(1+e)$. The question of whether the protocol is secure against *all* attacks, or there are attacks more serious than the private double-spend attack, remains open. This is not only an academic question, as well-known blockchain protocols like GHOST [14] had been shown to be secure against the private attack, only to be shown not secure later [15, 16].

1.3 Main contribution

Methodological Contribution. In this thesis, we show that, under a formal security model (Chapter 3), the Nakamoto-PoS protocol is indeed secure against *all* attacks, i.e., it has persistence and liveness whenever $\beta < 1/(1 + e)$. One can view our result as analogous to what [2] proved for Nakamoto's PoW protocol. However, *how* we prove the result is based on an entirely different approach. Specifically, the security proofs of [2] are based on *counting* the number of blocks that can be mined by the adversary over a long enough duration (see Fig. 1.1), and showing that the longest chain is

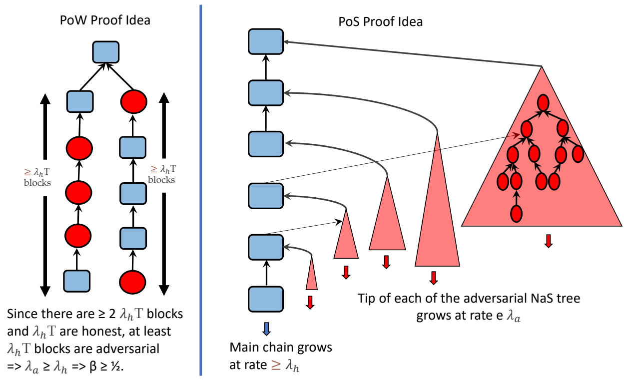

Figure 1.1: Notations: λ_a, λ_h are the rates at which the adversary and the honest nodes can mine a block on a given block, and T is the total duration. **Left**: In the PoW case, a counting argument shows that for the adversary to create a chain to match the longest chain, $\beta > 1/2$. This proof fails to work in the PoS case because there is no conservation of work and the total number of adversarial blocks that can be generated over a time duration T is exponentially larger than $\lambda_a T$. **Right**: Our proof technique. Race between main chain and adversarial trees: an honest block is a Nakamoto block if ***none*** of the previous NaS trees can ever catch up with the main chain downstream of the honest block. Security is proven by showing Nakamoto blocks occur at a non-zero frequency.

secure because the number of such adversarial blocks is less than the number of honest blocks whenever $\beta < 0.5$. This proof approach does not give non-trivial security results for the PoS protocol in question, because the number of adversarial blocks is exponentially larger than the number of honest blocks, due to the NaS phenomenon. Rather, our proof takes a dynamic view of the evolution of the blockchain, and shows that, whenever $\beta < 1/(1+e)$, there are infinite honest blocks, for which we use the notion of ***Nakamoto blocks*** (first defined in [17]), each having the property that none of the past adversary trees can ever catch up after the honest chain reaches the block (see Fig. 1.1). These Nakamoto blocks serve to stabilize the blockchain: when each such block enters the blocktree, complex as it may be, we are guaranteed that the entire prefix of the longest chain up to that block remains immutable in the future.

Although the adversary can propose an exponentially large number of blocks, perhaps surprisingly, the protocol can still tolerate a ***positive*** fraction

β of adversarial stake. On the other hand, the fraction that can be tolerated ($\frac{1}{1+e}$) is still less than the fraction for the longest chain PoW protocol ($\frac{1}{2}$). In [13] and [18], modifications of the longest chain protocol (called g-greedy and D-distance-greedy) are proposed, based on improvements to their security against the private double-spend attack. It has been shown in [19] that, unlike the longest chain protocol, these protocols are subject to worse public-private attacks, and they not only do not exhibit true improvements in security than the longest chain protocol, but in many cases, they do far worse.

New PoS Protocol Contribution. Taking a different direction, we propose a new family of simple longest chain PoS protocols that we call c-Nakamoto-PoS (Chapter 4); the fork choice rule remains the longest chain but the randomness update in the blockchain is controlled by a parameter c, the larger the value of the parameter c, the slower the randomness is updated. The common source of randomness used to elect a leader remains the same for c blocks starting from the genesis and is updated only when the current block to be generated is at a depth that is a multiple of c. When updating the randomness, the hash of that newly appended block is used as the source of randomness. The basic PoS Nakamoto protocol corresponds to $c = 1$, where the NaS attack is most effective. We can increase c to gracefully reduce the potency of NaS attacks and increase the security threshold. To analyze the formal security of this family of protocols, we combine our analysis for $c = 1$ with results from the theory of branching random walks [20]; this allows us to characterize the largest adversarial fraction β_c of stake that can be securely tolerated. As $c \to \infty$, $\beta_c \to 1/2$. We should point out that the Ouroboros family of protocols [10, 11] achieves security also by an infrequent update of the randomness; however, the update is much slower than what we are considering here, at the rate of once every constant multiple of κ, the security parameter. This is needed because the epoch must be long enough for the blockchain in the previous epoch to stabilize in order to generate the common randomness for the current epoch. Here, we are considering c to be a fixed parameter independent of κ, and show that this is sufficient to thwart the NaS attack. Technically, we show that even if c is small, there is no fundamental barrier to achieving any desired level of security κ. Hence, achieving a high level of security κ should not come at the cost of longer predictable window, and in this thesis we introduce a natural adoption of Nakamoto protocol to

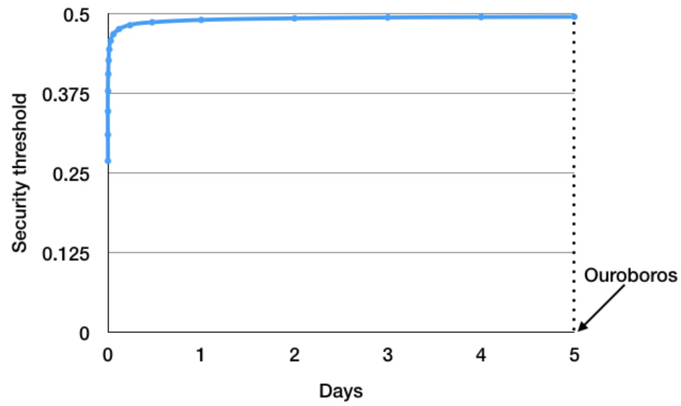

Figure 1.2: The security threshold β_c of c-Nakamoto-PoS against the prediction window, equaling to c times the inter-block time, which we set to be 20s, to match the implementation of Orouboros in Cardano. The Cardano project currently updates the common randomness every 5 days (21600 blocks, or 10κ), while the security threshold of c-Nakamoto-PoS can approach 1/2 with much higher randomness update frequency.

achieve this. The practical implication of this result is shown in Fig. 1.2, where we can see that c-Nakamoto can achieve comparable security with a much smaller prediction window than a current implementation of Orouboros as part of the Cardano project [21].

CHAPTER 2

PREDICTABILITY IN POS PROTOCOLS

2.1 Definition and evaluation of predictability in PoS protocols

Table 2.1: Our results decouple the prediction window W and the security parameter κ, achieving any combination of (W, κ). Prediction window W for other PoS protocols are strongly coupled with the security parameter $\kappa = \log(1/P_{\text{failure}})$. The maximum threshold of adversarial stake that can be tolerated by the PoS protocols while being secure is β . Nakamoto-PoS is the most basic way of extending Nakamoto protocol to the PoS setting. This was originally introduced in [13, 18] but with an incomplete security analysis. In Chapter 5 we show $\beta_c \approx 1/2 - \Theta(\sqrt{(1/c)\ln c})$ and numerically tabulate β_c (example: $\beta_c = 39\%$ for $c = 10$). [9, 10, 11] are the Ouroboros family, [8] is Snow White, [13] is g-Greedy, and [18] is D-Distance-Greedy.

	Longest Chain					BFT	Our results	
	[9]	[10]	[11]	[8]	[13] [18]	[4]	Nakamoto-PoS	c-Nakamoto-PoS
W	2κ	3κ	6κ	6κ	1	$\Theta(\kappa)$	1	c
β	$\frac{1}{2}$				unknown	$\frac{1}{3}$	$\frac{1}{1+e}$	β_c

In PoW protocols such as Bitcoin, no miner knows when they will get to propose the block until they solve the puzzle, and once they solve the puzzle, the block is inviolable (because the puzzle solution will become invalid, otherwise). This causality is reversed in proof-of-stake (PoS) protocols: a node eligible to propose a block knows ***a priori*** of its eligibility before proposing a block. This makes PoS protocols vulnerable to a new class of serious attacks not possible in the PoW setting. We briefly discuss these attacks here, deferring a detailed discussion to Appendix B.

Definition 1 (W-predictable)**.** Given a PoS protocol Π_{PoS}, let $\mathcal{C}$ be a valid blockchain ending with block B with a time stamp t. We say a block B enables w-length prediction, if there exists a time $t_1 > t$ and a block B_1 with a time stamp t_1 such that (i) B_1 can be mined by miner (using its private state and the common public state) at time t; and (ii) B_1 can be appended to $\mathcal{C}^0$ to form a valid blockchain for any valid chain $\mathcal{C}^0$ that extends $\mathcal{C}$ by appending $w-1$ valid blocks with time stamps within the interval (t, t_1). By taking the maximum over the prediction length over all blocks in Π_{PoS}, we say Π_{PoS} is W-predictable. W is the size of the prediction window measured in units of number of blocks.

Informally, a longest-chain PoS protocol has W length prediction window if it is possible for a miner to know that it is allowed to propose a block W blocks downstream of the present blockchain. We note that our definition is similar to the definition of W-locally predictable protocols in [12], where it has been pointed out that PoS protocols trade off between predictability and nothing-at-stake incentive attacks. In Table 2.1, we compare the prediction windows of various protocols. The longest-chain family of protocols of Ouroboros update randomness every epoch have prediction window equal to the epoch length. Furthermore, they require the epoch length to be proportional to the security parameter κ, since one error event is that a majority of block producers in an epoch are not honest (and in that case, they can bias the randomness of all future slots).

We note that in any PoS protocol with confirmation-depth κ (the number of downstream blocks required to confirm a given block), a simple bribing attack is possible, where a briber requests the previous block producers to sign an alternate block for each of their previous certificates. However, such attacks are overt and easily detectable, and can be penalized with slashing penalties. If the prediction window W is greater than the confirmation-depth κ, then the following ***covert*** (undetectable) attack becomes possible. An adversary who wants to issue a double-spend can create a website where nodes that have future proposer slots post their leadership certificates for a bribe. If the adversary gets more than $\kappa+1$ miners to respond to this request, then the adversary can launch the following attack: (1) collect the $\kappa+1$ leadership certificates, (2) issue a transaction that gets included in the upcoming honest block, (3) let the honest chain grow for κ blocks to confirm the transaction and receive any goods in return, then (4) create a double-spend against the previous transaction and (5) create a longer chain downstream of a block including the double-spend using the $\kappa + 1$ certificates. We note that this attack does not require participation from miners having a majority of the stake. Far from it, it only requires $\kappa + 1$ out of the next 2κ miners each holding a potentially infinitesimal fraction of stake. Furthermore, this attack does not require miners to double-sign blocks, making it indistinguishable from unexpected network latency and providing plausible deniability for the miners who take the bribe. Thus, it is a serious covert attack on security requiring only participants with a net infinitesimal stake to participate in it. We note that this attack is not covered in the popular adaptive adversary model [10, 5], since in that model, nodes are assumed not to have any agency and remain honest till the adversary corrupts them (based only on public state).

We note that it is ***not possible*** to mitigate the prediction issue by increasing the confirmation depth beyond the prediction window (which is equal to the epoch length). This is because the guarantees of existing protocols rely on the randomness of each epoch being unbiased and this guarantee fails to hold when a majority of nodes in an epoch are bribed through the aforementioned mechanism in order to bias the randomness.

While our discussion so far focused on longest-chain PoS protocols, we note that the prediction issue is even more serious in BFT based PoS protocols. PoS-based BFT protocols that work with the same committee or same proposer for many time-slots give raise to similar prediction based attacks. Even in protocols such as Algorand [4, 5], which require a new committee for each step of the BFT protocol, the entire set of committees for all steps of the BFT protocol for a given block is known once the previous block is finalized. This leads to a similar type of bribing attack where once a 2/3 majority of a BFT-step committee coordinate through a centralized website, they can sign a different block than the one the honest nodes agreed on. We note that since Algorand elects a small constant size committee (proportional to κ) for each round, a 2/3 majority of the committee can comprise a negligible total stake. Thus, in Algorand, even though the prediction window appears negligible, the confirmation delay is also small – thus leading to the same type of attack (detailed discussion deferred to Appendix B). A formal definition of prediction window for BFT-based PoS protocols is in Appendix B.4 where we evaluate the prediction window W for a canonical BFT based PoS protocol:

Algorand [4]. There is a strong coupling between the security parameter and prediction window for Algorand, and is tabulated in Table 2.1.

2.2 Summary

We have demonstrated that both longest-chain and BFT based protocols are highly vulnerable to prediction-based security attacks when coordinating through an external bribing mechanism, thus compromising the persistence and liveness of the system. These attacks are covert, i.e., the deviant behavior is not detectable and punishable on the blockchain, and require only an infinitesimal fraction of the stake to collude, thus significantly weakening the security of the protocol. This motivates the study of Nakamoto-PoS (with a very small prediction window) and the design of a new PoS protocol that has a prediction window much shorter than the confirmation depth and can be secure against adversaries with up to 50% of the stake. This state of affairs, together with the security results we prove in this thesis, are summarized in Table 2.1.

CHAPTER 3

SECURITY MODEL

3.1 Network and adversary model

A blockchain protocol Π is directed by an ***environment*** $\mathcal{Z}(1^{\kappa})$, where κ is the security parameter. This environment (i) initiates a set of participating nodes $\mathcal{N}$; (ii) manages nodes through an adversary $\mathcal{A}$ which ***corrupts*** a dynamically changing subset of nodes; (iii) manages all accesses of each node from/to the environment including broadcasting and receiving messages of blocks and transactions.

The protocol Π proceeds in discrete time units called ***slots***, each consisting of δ milliseconds (also called the slot duration), i.e. the ***time*** argument in the input to the hash function should be in δ millisecond increments. Each slot sl_r is indexed by an integer $r \in \{1, 2, \cdots\}$. A ***ledger*** associates at most one block to each slot among those generated (or proposed) by participating nodes, each running a distributed protocol. Collectively, at most one block per slot is selected to be included in the ledger according to a rule prescribed in the protocol Π. Similar to [9], we assume that the real time window for each slot satisfies that: (1) The time window of a slot is determined by a publicly-known and monotonically increasing function of current time; (2) every user has access to the current global time and any discrepancies between nodes' local time are insignificant in comparison with the slot duration.

We follow the security model of [2, 3, 10] with an ideal functionality $\mathcal{F}$. This includes ***diffuse*** functionality and ***key and transaction*** functionality as described below. With a protocol Π, adversary $\mathcal{A}$, environment Z, and security parameter κ, we denote by $\mathsf{VIEW}^{n,\mathcal{F}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa)$ the view of a node $n \in \mathcal{N}$ who has access to an ideal functionality $\mathcal{F}$.

We consider a semi-synchronous ***network model*** with bounded network delay similar to that of [3, 10] that accounts for adversarially controlled message delivery and immediate node corruption. All broadcast messages are delivered by the adversary, with a bounded network delay Δ millisecond. Let $\tau = \Delta/\delta$ be an integer. We model this bounded network delay by allowing the adversary to selectively delay messages sent by honest nodes, with the following restrictions: (i) the messages broadcast in slot sl_r must be delivered by the beginning of slot $sl_{r+\tau}$; and (ii) the adversary cannot forge or alter any message sent by an honest node. This is the so called ***delayed di use functionality*** (denoted by $\mathsf{DDi\ use}_\tau$ in [10]).

The dynamically changing set of ***honest*** (or uncorrupted) nodes $\mathcal{H} \subseteq \mathcal{N}$ strictly follows the blockchain protocol Π. The ***key registration*** functionality (from [9]) is initialized with the nodes $\mathcal{N}$ and their respective stakes $(\mathsf{stake}_1, \ldots, \mathsf{stake}_{jNj})$ such that the fraction of the initial stake owned by node n is $\mathsf{stake}_n / \sum_{m2N} \mathsf{stake}_m$. At the beginning of each round, the adversary can dynamically corrupt or uncorrupt any node $n \in \mathcal{N}$, with a permission from the environment $\mathcal{Z}$ in the form of a message $(\mathsf{Corrupt}, n)$ or $(\mathsf{Uncorrupt}, n)$. Even the corrupted nodes form a dynamically changing set, the total proportion of the adversarial stake is upper bounded by β all the time. For the honest nodes, the functionality can sample a new public/secret key pair for each node and record them. For the corrupted nodes, if it is missing a public key, the adversary can set the node's public key, and the public keys of corrupt nodes will be marked as such. When the adversary releases the control of a corrupted node, the node retrieves the current view of the honest nodes at the beginning of the following round.

Any of the following actions are allowed to take place. (i) A node can retrieve its public/secret key pair from the functionality. (ii) A node can retrieve the whole database of public keys from the functionality. (iii) The environment can send a message (Create) to spawn a new node, whose local view only contains the genesis block, and the functionality samples its public/secret key pair. (iv) The environment can request a transaction, specifying its payer and recipient. The functionality adjusts the stakes according to the transactions that make into the current ledger, as prescribed by the protocol Π. The adversary has access to the state of a corrupt node n, and will be activated in place of node n with restrictions imposed by $\mathcal{F}$.

3.2 Cryptographic primitives

Verifiable Random Function (VRF). Verifiable Random Functions (VRF), first introduced in [22], generates a pseudorandom number with a proof of its correctness. A node with a secret key sk can call $\text{VRF}\textsc{prove}(\cdot, sk)$ to generates a pseudorandom ***output*** $F_{sk}(\cdot)$ along with a ***proof*** $\pi_{sk}(\cdot)$. Other nodes that have the proof and the corresponding public key pk can check that the output has been generated by VRF, by calling $\text{VRF}\textsc{verify}(\cdot, \text{output}, \pi_{sk}(\cdot), pk)$. An efficient implementation of VRF was introduced in [23], which formally satisfy Definition 6 in Appendix A. This ensures that the output of a VRF is computationally indistinguishable from a random number even if the public key pk and the function $\text{VRF}\textsc{prove}$ is revealed.

Key Evolving Signature schemes (KES). We propose using forward secure signature schemes [24] to sign the transactions to be included in a generated block. This prevents the adversary from altering the transactions in the blocks mined in the past. Efficient Key Evolving Signature (KES) schemes have been proposed in [25, 10] where keys are periodically erased and generated, while the new key is linked to the previous one. This is assumed to be available to the nodes via the ideal functionality $\mathcal{F}$. This ensures immutability of the contents of the blocks.

CHAPTER 4

PROTOCOL DESCRIPTION

In this chapter, we explain our protocol following terminologies from [10] and emphasize the differences as appropriate. The ideal functionality $\mathcal{F}$ captures the resources available to the nodes in order to securely execute the protocol. When a PoS system is launched, a collection $\mathcal{N}$ of nodes are initialized. Each node $n \in \mathcal{N}$ is initialized with a coin possessing stake stake_n, a verification/signing key pair $(\mathsf{KES}.vk_n, \mathsf{KES}.sk_n)$, and a public/secret key pair $(\mathsf{VRF}.pk_n, \mathsf{VRF}.sk_n)$. The Key Evolving Signature key pair (KES) is used to sign and verify the content of a block, while the Verifiable Random Function key pair (VRF) is used to verify and elect leader nodes who generate new blocks. All the nodes and the adversary know all public keys $\{\mathsf{pk}_n = (\mathsf{KES}.vk_n, \mathsf{VRF}.pk_n)\}_{n2N}$. The *genesis* block contains all public keys and initial stakes of all nodes, $\{(\mathsf{pk}_n, \mathsf{stake}_n)\}_{n2N}$, and also contains a nonce in genesis.content.RandSource. This nonce is used as a seed for the randomness. The depth of a block in a chain is counted from the genesis (which is at depth zero). We denote the time at the inception of the genesis block as zero (milliseconds), such that the i-th slot starts at the time $\delta \cdot i$ milliseconds (since the inception of the genesis block). Nakamoto-PoS protocol is executed by the nodes and is assumed to run indefinitely. At each slot a node starts with a local chain $\mathcal{C}$, which it tries to append new blocks on.

4.1 Proposer selection

At each slot, a fresh subset of nodes are randomly elected to be the leaders, who have the right to generate new blocks. To be elected one of the leaders, each node first decides on where to append the next block, in its local view of the blocktree. This choice of a *parent block* is governed by the fork choice rule prescribed in the protocol. For example, in BitCoin, an honest node appends a new block to the deepest node in the local view of the blocktree. This is known as Nakamoto protocol. We propose s-***truncated longest chain rule*** that includes the Nakamoto protocol as a special case, which we define later in this chapter.

A random number of leaders are elected in a single slot, and the collective average block generation rate is controlled by a global parameter ρ that is adaptively set by the ideal functionality $\mathcal{F}$. The individual block generation rate is proportional to the node's stake. The stakes are updated continuously as the ledger is updated, but only a coin s blocks deep in the ledger can be used in the election (the same parameter s as used in the truncated longest chain rule), and is formally defined later in this chapter.

Concretely, at each slot, a node $n \in \mathcal{N}$ draws a number distributed uniformly at random in a predefined range. If this is less than the product of its stake and a parameter ρ (Algorithm 1 line 17), the node is elected one of the leaders of the slot and gains the right to generate a new block. Ideally, we want to simulate such a random trial while ensuring that the outcome (i) is ***veri able*** by any node after the block generation; (ii) is ***unpredictable*** by any node other than node n before the generated block has been broadcast; and (iii) is ***independent*** of any other events. Verifiability in (i) is critical in ensuring consistency among untrusted pool of nodes. Without unpredictability in (ii), the adversary can easily take over the blockchain by adaptively corrupting the future leaders. Without independence in (iii), a corrupted node might be able to grind on the events that the simulator (and hence the outcome of the election) depends on, until it finds one that favors its chances of generating future blocks. Properties (ii) and (iii) are challenges unique to PoS systems, as ***predicting*** and ***grinding attacks*** are computationally costly in PoW systems.

To implement such a simulator in a distributed manner among mutually untrusting nodes, [5, 10] proposed using Verifiable Random Functions (VRFs), formally defined in Appendix A. In our proposed protocol, a node n uses its secret key $\mathsf{VRF}.sk$ to generate a pseudorandom ***hash*** and a ***proof*** of correctness (Algorithm 1 line 16). If node n is elected a leader and broadcasts a new block, other nodes can verify the correctness with the corresponding public key $\mathsf{VRF}.pk$ and the proof (which is included in the block content). This ensures unpredictability, as only node n has access to its secret key, and verifiability, as any node can access all public keys and verify that the correctness of the random leader election.

The pseudorandom hash generated by $\mathrm{VRF}\textsc{prove}(x, \mathsf{VRF}.sk)$, depends on the external source of randomness, $(x, \mathsf{VRF}.sk)$, that is fed into the function. Along with the secret key $\mathsf{VRF}.sk$, which we refer to as the ***private*** source of randomness, we prescribe constructing a header x that contains the time (in a multiple of δ milliseconds) and a dynamically changing ***common*** source of randomness. Including the time ensures that the hash is drawn exactly once every slot. Including the common source of randomness ensures that the random elections cannot be predicted in advance, even by the owner of the secret key. Such ***private*** predictability by the owner of the secret key leads to other security concerns that we discuss in Appendix B.

A vanilla implementation of such a protocol might (a) update stakes immediately and (b) use the hash of the previous block (i.e. the parent of the newly generated block in the main chain as defined by the fork chain rule) as the common source of randomness. Each of these choices creates a distinct opportunity for an adversary to grind on, that could result in serious security breaches. We explain the potential threats in the following and propose how to update the randomness and the stake, respectively, to prevent each of the grinding attacks. A formal analysis of the resulting protocol is provided in Chapter 5.

4.2 Updating the common source of randomness

One way to ensure unpredictability by even the owner of the secret key is to draw randomness from the dynamically evolving blocktree. For example, we could use the hash of the parent block (i.e. the block that a newly generated block will be appended to). This hash depends only on the parent block proposer's secret key, the time, and the source of randomness included in the header of the parent block. In particular, this hash does not depend on the content of the parent block, to prevent an additional source of grinding attack. However, such a frequent update of the source creates an opportunity for the adversary to grind on. At every round, a corrupted node can run as many leader elections as the number of blocks in the blocktree, each appending to a different block as its parent. To mitigate such grinding attacks, we propose a new update rule for the source of randomness which we call

c-correlation.

A parameter $c \in \mathsf{Z}$ determines how frequently we update. The common source of randomness remains the same for c blocks, and is updated only when the current block to be generated is at a depth that is a multiple of c (Algorithm 1 line 19). When updating, the hash of that newly appended block is used as the source of randomness. When $c = 1$, this recovers the vanilla update rule, where a grinding attack is most effective. We can increase c to gracefully increase the security threshold. A formal analysis is provided in Appendix 5. When $c = \infty$, every block uses the nonce at the genesis block as the common source of randomness. This makes the entire future leader elections predictable in private, by the owners of the secret keys.

4.3 Dynamic stake

The stake of a node n (or equivalently that of the coin the node possesses) is not only changing over *time* as transactions are added to the blocktree, but also over which *chain* we are referring to in the blocktree. Different chains in the tree contain different sequences of transactions, leading to different stake allocations. One needs to specify which chain we are referring to, when we access the stake of a node. Such accesses are managed by the ideal functionality $\mathcal{F}$ (Algorithm 1 line 12).

When running a random election to append a block to a parent block b at depth $\ell - 1$ in the blocktree, a coin can be used for this election of creating a block with depth ℓ if and only if the coin is in the stake at the block with depth $\ell - s$ on the chain leading to block b. Accordingly, a node n has a winning probability proportional to $\mathsf{stake}_n(b)$ when mining on block b, where $\mathsf{stake}_n(b)$ denotes the stake belonging to node n as in the $(s-1)$-th block before b. Starting from an initial stake distribution $\mathsf{stake}_n(b_{\text{genesis}})$, we add to or subtract from the stake according to all transactions that (i) involve node n (or the coin that belongs to node n); (ii) are included in the chain of blocks from the genesis to the reference block b; and (iii) are included in the blockchain at least $s - 1$ blocks before b. Here, $s \in \mathsf{Z}$ is a global parameter.

When s=1, the adversary can grind on (the secret key $\mathsf{VRF}.sk$ of) the coin. For example, once a corrupted node is elected as a leader at some time slot and proposed a new block, it can include transactions in that block to transfer all stake to a coin that has a higher chance of winning the election at later time slots. To prevent such a grinding on the coin, a natural attempt is to use the stake in the block with depth $\ell - s$ when trying to create a block at depth ℓ on the main chain. However, there remains a vulnerability, if we use the Nakamoto protocol from BitCoin as the fork choice rule.

Consider a corrupted node growing its own ***private chain*** from the genesis block (or any block in the blocktree). A private chain is a blockchain that the corrupted node grows privately without broadcasting it to the network until it is certain that it can take over the public blocktree. Under the Nakamoto protocol, this happens when the private chain is longer (in the number of blocks) than the longest chain in the public blocktree. Note that the public blocktree grows at a rate proportional to ρ and the total stake of the nodes that append to the public blocktree. With a grinding attack, the private chain, which is entirely composed of the blocks generated by the corrupted node, can eventually take over the public blocktree.

Initially, the private chain grows at a rate proportional to ρ and the stake controlled by the corrupted node. However, after s blocks from the launch of the private chain, the corrupted node can start grinding on the private key of the coin; once a favorable coin is found, it can transfer the stake to the favored coin by including transactions in the first ancestor block in the private chain. This is possible as all blocks in the private chain belong to the corrupted node. It can alter any content of the private chain and sign all blocks again. With such a grinding attack (which we refer to as ***coin grinding***), the corrupted node can potentially be elected a leader every slot in the private chain, eventually overtaking the public blocktree. To prevent this private grinding attack, we propose using an s-truncation as the fork choice rule.

4.4 Fork choice rule

An honest node follows a fork choice rule prescribed in the protocol. The purpose is to reach a consensus on which chain of blocks to maintain, in a distributed manner. Eventually, such chosen chain of blocks produces a final ledger of transactions. Under the Nakamoto protocol, a node appends the next generated block to the longest chain in its local view of the blocktree. Unlike PoW systems, Nakamoto protocol can lead to serious security issues for PoS systems as discussed above. We propose using the following s-truncated longest chain rule, introduced in [11, 13].

At any given time slot, an honest node keeps track of one main chain that it appends its next generated block to. Upon receiving a new chain of blocks, it needs to decide which chain to keep. Instead of comparing the length of those two chains, as in Nakamoto protocol, we compare the creation time of the first s blocks after the fork in truncated versions of those two chains (Algorithm 1 line 34). Let b_{fork} be the block where those two chains fork. The honest node counts how long it takes in each chain to create up to s blocks after the fork. The chain with shorter time for those s blocks is chosen, and the next generated block will be appended to the newest block in that selected chain. When $s = \infty$, the stake is fixed since the genesis block, which leads to a system that is secure but not adaptive. This is undesirable, as even a coin with no current stake can participate in block generation. We propose using an appropriate global choice of $s < \infty$, that scales linearly with the security parameter κ. This ensures that the protocol meets the desired level of security, while adapting to dynamic stake updates. One caveat is that we only apply this s-truncation when comparing two chains that both have at least s blocks after those two chains forked. If one of the chain has less than s blocks after forking, we use the longest chain rule to determine which chain to mine on. This is necessary in order to ensure that s-truncation is only applied to chains with enough blocks, such that our probabilistic analysis results hold.

4.5 Content of the block

Once a node is elected a leader, all unconfirmed transactions in its buffer are added to the content (Algorithm 1 line 22). Along with the transactions, the content of the block also includes the identity of the coin that won the election, and the *hash* and *proof* from VRFprove($\cdot$). This allows other nodes to verify the accuracy of the leader election. A common source of randomness *RandSource* is also included, to be used in the next leader election. 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 forward secure signature $\mathsf{KES}.sk_n$.

Note that the content of the block is added after the leader election, in order to avoid any grinding on the content. However, this allows the adversary to create multiple blocks with the same header but different content. In particular, after one leader election, the adversary can create multiple blocks appending to different parent blocks, as long as those parent blocks share the same common source of randomness. Such copies of a block with the same header but different contents are known as a "forkable string" in [9] or "non-core blocks" in [13]. We show in Chapter 5 that the Nakamoto-PoS protocol is secure against all such variations of attacks. The protocol parameters used in our analysis are provided in Table 4.1.

Table 4.1: The parameters used in our analysis.

β	total proportion of the adversarial stake
δ	slot duration
Δ	network delay
κ	security parameter
c	correlation parameter
s	parameter in the fork choice rule
ϕ_c	maximum growth rate of a private tree

CHAPTER 5

SECURITY ANALYSIS OF THE STATIC STAKE PROTOCOL

In this chapter, we provide formal security analysis for the longest chain PoS protocol. We focus on $c = 1$, and will discuss the general c case at the end of this chapter and further in Appendix D. To simplify the expressions, we will consider the regime when the time slot duration δ is very small, so that the block generation processes can be modeled as Poisson. We will also assume the stake distribution is static in this chapter; the case of dynamic stake is discussed in Chapter 6.

We prove liveness and persistence of the protocol through understanding when the longest chain ***converges*** as time passes, regardless of the adversarial strategy. We first analyze this convergence in the setting when the network delay $\Delta = 0$. This setting contains the core ideas of the proof, and allows us to explain it with the simplest notations. Then we extend it to the case of positive network delay. Finally, we use these results to prove high probability guarantees on the liveness and persistence of the protocol.

5.1 Warmup: $\Delta = 0$

5.1.1 Random processes

In this setting, all the honest nodes have the same view of the blockchain, which can be modeled as a random process $\{(\mathcal{T}(t), \mathcal{C}(t)) : t \geq 0\}$. $\mathcal{T}(t)$ is a tree and $\mathcal{C}(t)$ is the public longest chain at time t. The tree $\mathcal{T}(t)$ is interpreted as consisting of all the blocks that are generated by both the adversary and the honest nodes up until time t, including blocks that are kept in private by the adversary. Note that $\mathcal{T}(t)$ consists of the honest blocks, mined at the tip of the longest chain, and all the blocks that the adversary ***can*** generate, by trying and winning the election lotteries at all possible locations of the blocktree. As such $\mathcal{T}(t)$ captures all the resources the adversary has at its disposal to attack at time t.

The longest chain protocol in Chapter 4 results in a process described as follows.

1. $\mathcal{T}(0) = \mathcal{C}(0)$ is a single root block (the genesis block).

2. $\mathcal{T}(t)$ evolves as follows: there are independent Poisson processes of rate λ_a at each block of $\mathcal{T}(t)$ (we call them the ***adversary*** processes), plus an additional independent Poisson process of rate λ_h (we call it the ***honest*** process) arriving at the last block of the chain $\mathcal{C}(t)$, i.e. an aggregate Poisson process of rate $\lambda_a + \lambda_h$ at that block (the tip of the longest chain), and rate λ_a at every other block of $\mathcal{T}(t)$. A new block is added to the tree at a certain block when a block is generated. An arrival from the honest process is called an honest block. An arrival from the adversary process is called an adversarial block.

3. The chain $\mathcal{C}(t)$ is updated in two possible ways : 1) an additional honest block is added to $\mathcal{C}(t)$ if an arrival from the honest process occurs; 2) an adversary can replace $\mathcal{C}(t\ )$ by another chain $\mathcal{C}(t)$ from $\mathcal{T}(t)$ which is equal or longer in length than $\mathcal{C}(t\ )$.[1] The adversary's decision has to be based on the current state of the process.

The longest chain protocol means that the honest nodes always propose on the tip of the current public longest chain $\mathcal{C}(t)$ (at rate λ_h, proportional to their stake). The adversary can propose on any block (at rate λ_a, again proportional to its stake). The adversary can change where the honest nodes act by broadcasting an equal or longer length chain using the blocks it has succeeded in proposing. Since the adversary can change where the honest nodes can propose even with an equal length new chain, that means the adversary is given the ability to choose where the honest nodes propose when there are more than one longest public chain.

Proving the liveness and persistence of the protocol boils down to providing a guarantee that the chain $\mathcal{C}(t)$ converges as $t \to \infty$ regardless of the adversary's strategy. We will show that this happens provided that $\lambda_a < \lambda_h/e$,

[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.

i.e.

$$\beta := \frac{\lambda_a}{\lambda_a + \lambda_h} < \frac{1}{1+e}.$$

Our key contribution here is defining an appropriate notion of ***Nakamoto Block*** and analyzing how frequently it occurs.

5.1.2 Nakamoto block

We first define several basic random variables and random processes which are constituents of the processes $\mathcal{T}(\cdot)$ and $\mathcal{C}(\cdot)$. Then we will use them to define the notion of Nakamoto block, and prove that indeed once it occurs, convergence of the longest chain will occur regardless of what the adversarial strategy is.

1. τ_i = generation time of the i-th honest block; $\tau_0 = 0$ is the generation time of the genesis block, $\tau_{i+1} - \tau_i$ is exponentially distributed with mean $1/\lambda_h$, i.i.d. across all i's.

2. $A_h(t)$ = number of honest blocks generated from time 0 to t. $A_h(t)$ increases by 1 at each time τ_i. $A_h(\cdot)$ is a Poisson process of rate λ_h.

3. $L(t)$ is the length of $\mathcal{C}(t)$. $L(0) = 0$. Note that since the chain $\mathcal{C}(t)$ increments by 1 for every honest block generation, it follows that for all i and for all $t > \tau_i$,

$$L(t) - L(\tau_i) \geq A_h(t) - A_h(\tau_i). \tag{5.1}$$

4. $\mathcal{T}_i = \{\mathcal{T}_i(s) : s \geq 0\}$ is the random tree process generated by the adversary starting from the i-th honest block. $\mathcal{T}_i(0)$ consists of the i-th honest block and $\mathcal{T}_i(s)$ consists of all adversarial blocks grown on the i-th honest block from time τ_i to $\tau_i + s$. Note that the $\mathcal{T}_i$'s are i.i.d. copies of the pure adversarial tree $\mathcal{T}^a$ (i.e. the tree $\mathcal{T}(t)$ when $\lambda_h = 0$).

5. $D_i(s)$ is the depth of the adversarial tree $\mathcal{T}_i(s)$.

Note that the overall tree $\mathcal{T}(t)$ is the composition of the adversarial trees $\mathcal{T}_0(t), \mathcal{T}_1(t - \tau_1), \ldots \mathcal{T}_i(t - \tau_i)$ where the i-th honest block is the last honest block that was generated before time t. How these trees are composed to form $\mathcal{T}(t)$ depends on the adversarial action on when to release the private chains. We make the following important definition.

Definition 2. (**Nakamoto block for** $\Delta = 0$) Define

$$E_{ij} = \text{event that } D_i(t - \tau_i) < A_h(t) - A_h(\tau_i) \text{ for all } t > \tau_j \tag{5.2}$$

and

$$F_j = \bigcap_{i=0}^{j\ \ 1} E_{ij}. \tag{5.3}$$

The j-th honest block is called a ***Nakamoto block*** if event F_j occurs.

We can interpret the definition of a Nakamoto block in terms of a *ctitious system*, having the same block mining times as the actual system, where there is a growing chain consisting of only honest blocks and the adversary trees are racing against this honest chain. The event E_{ij} is the event that the adversarial tree rooted at the i-th honest block does not catch up with the honest chain any time after the generation of the j-th honest block. Such a tree can be interpreted as providing resource for a possible attack at the honest chain. If E_{ij} occurs, then there is not enough resource for the i-th tree to attack after the j-th block. If F_j occurs, there is not enough resource for ***any*** of the previous trees to attack the honest chain.

Even though the events are about a fictitious system with a purely honest chain and the longest chain in the actual system may consist of a mixture of adversarial and honest blocks, intuitively the actual chain can only grow faster than the fictitious honest chain, and so we have the following key lemma.

Lemma 1. If the j-th honest block is a Nakamoto block, then it will be in the longest chain $\mathcal{C}(t)$ for all $t > \tau_j$. Equivalently, $\mathcal{C}(\tau_j)$ will be a prefix of $\mathcal{C}(t)$ for all $t > \tau_j$.

Proof. We will argue by contradiction. Suppose F_j occurs and let $t \ > \tau_j$ be the smallest t such that $\mathcal{C}(\tau_j)$ is not a prefix of $\mathcal{C}(t)$. Let b_h be the last honest block on $\mathcal{C}(t\ )$ (which must exist, because the genesis block is by definition honest). If b_h is generated at some time $t_1 > \tau_j$, then $\mathcal{C}(t_1\ )$ is the prefix of $\mathcal{C}(t\ )$ before block b_h, and does not contain $\mathcal{C}(\tau_j)$ as a prefix, contradicting the minimality of t . So b_h must be generated before τ_j, and hence b_h is the

i-th honest block for some $i < j$. The part of $\mathcal{C}(t\ )$ after block b_h must lie entirely in the adversarial tree $T_i(t\ \ - \tau_i)$ rooted at b_h. Hence,

$$D_i(t\ \ - \tau_i) < A_h(t\ ) - A_h(\tau_i) \leq L(t\ ) - L(\tau_i), \tag{5.4}$$

where the first inequality follows from the fact that F_j holds, and the second inequality follows from the longest chain policy (Equation (5.1)). From this we obtain that

$$L(\tau_i) + D_i(t\ \ - \tau_i) < L(t\ ), \tag{5.5}$$

which is a contradiction since $L(t\ ) \leq L(\tau_i) + D_i(t\ \ - \tau_i)$. □

We will show that Nakamoto block occurs infinitely number of times if $\beta < 1/(1+e)$, and will also give an estimate on how frequently that happens. This will imply persistence and liveness of the protocol with high probability guarantees.

Since the occurrence of the event F_j depends on whether the adversarial trees from the previous honest blocks can catch up with the (fictitious) honest chain, we next turns to an analysis of the growth rate of an adversarial tree.[2]

5.1.3 The adversarial tree via branching random walks

The adversarial tree $\mathcal{T}^a(t)$ is the tree $\mathcal{T}(t)$ when $\lambda_h = 0$, i.e. honest nodes not acting. Let the depth of the tree $\mathcal{T}^a(t)$ be denoted by $D(t)$ and defined as the maximum depth of its blocks. The genesis block is always at depth 0 and hence $\mathcal{T}^a(0)$ has depth zero.

We give a description of the (dual of the) adversarial tree in terms of a Branching Random Walk (BRW). Such a representation appears already in [26, 27], but we use here the standard language from, e.g., [28, 20].

Consider the collection of k tuples of positive integers, $\mathcal{I}_k = \{(i_1, \ldots, i_k)\}$, and set $\mathcal{I} = \cup_{k>0} \mathcal{I}_k$. We consider elements of $\mathcal{I}$ as labelling the vertices of a rooted infinite tree, with $\mathcal{I}_k$ labelling the vertices at generation k as follows: the vertex $v = (i_1, \ldots, i_k) \in \mathcal{I}_k$ is the i_k-th child of vertex $(i_1, \ldots, i_{k\ \ 1})$ at level $k - 1$. An example of labelling is given in Fig. 5.1. For such v we also

[2]The mean growth rate of this tree was analyzed in [13] using difference equations. Here, we are using the machinery of branching random walks, which not only gives us tail probabilities but also allow the extension to the c-correlated protocol, $c > 1$, easily.

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.

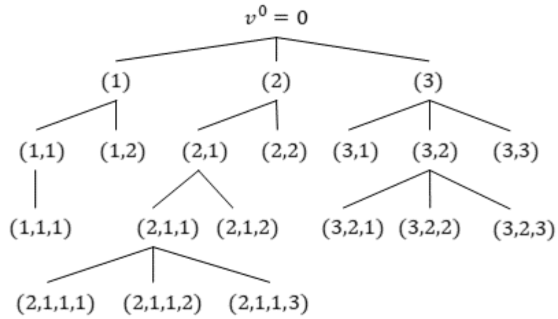

Figure 5.1: Labelling the vertices of a rooted infinite tree.

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 $\mathcal{W}_v = \sum_{j \le i_k} \mathcal{E}_{(i_1, \ldots, i_{k-1}, j)}$ and let $S_v = \sum_{j \le k} \mathcal{W}_{v^j}$. This creates a labelled tree, with the following interpretation: for $v = (i_1, \ldots, i_j)$, the W_{v^j} are the waiting for v^j to appear, measured from the appearance of v^{j-1}, and S_v is the appearance time of v. A moment's thought ought to convince the reader that the tree S_v is a description of the adversarial tree, sorted by depth.

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

$$\{D(t) \le k\} = \{S_k \ge t\}. \tag{5.6}$$

S_k is the minimum of a standard BRW. Introduce, for $\theta < 0$, the moment generating function

$$\begin{aligned} \Lambda(\theta) &= \log \sum_{v \in \mathcal{I}_1} E(e^{\theta S_v}) = \log \sum_{j=1}^{\infty} E(e^{\sum_{i=1}^{j} \theta \mathcal{E}_i}) = \log \sum_{j=1}^{\infty} (E(e^{\theta \mathcal{E}_1}))^j \\ &= \log \frac{E(e^{\theta \mathcal{E}_1})}{1 - E(e^{\theta \mathcal{E}_1})}. \end{aligned}$$

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

An important role is played by $\theta^* = -e\lambda_a$, for which $\Lambda(\theta^*) = -1$ and

$$\sup_{\theta<0}\left(\frac{\Lambda(\theta)}{\theta}\right) = \frac{\Lambda(\theta^*)}{\theta^*} = \frac{1}{\lambda_a e} = \frac{1}{|\theta^*|}.$$

Indeed, see e.g. [20, Theorem 1.3], we have the following.

Lemma 2.

$$\lim_{k\to\infty}\frac{S_k}{k} = \sup_{\theta<0}\left(\frac{\Lambda(\theta)}{\theta}\right) = \frac{1}{|\theta^*|}, \quad a.s.$$

In fact, much more is known, see e.g. [29].

Lemma 3. There exist explicit constants $c_1 > c_2 > 0$ so that the sequence $S_k - k/\lambda_a e - c_1 \log k$ is tight, and

$$\liminf_{k\to\infty} S_k - k/\lambda_a e - c_2 \log k = \infty, a.s.$$

Note that Lemmas 2, 3 and (5.6) imply in particular that $D(t) \le e\lambda_a t$ for all large t, a.s., and also that

$$\text{if } e\lambda_a > \lambda_h \text{ then } D(t) > \lambda_h t \text{ for all large } t\text{, a.s..} \tag{5.7}$$

We will need also tail estimates for the event $D(t) > e\lambda_a t + x$. While such estimates can be read from [20], we bring instead a quantitative statement suited for our needs.

Lemma 4. For $x > 0$ so that $e\lambda_a t + x$ is an integer,

$$P(D(t) \ge e\lambda_a t + x) \le e^{-x}. \tag{5.8}$$

Proof. We use a simple upper bound. Write $m = e\lambda_a t + x$. Note that by (5.6),

$$P(D(t) \ge m) = P(S_m \le t) \le \sum_{v\in\mathcal{I}_m} P(S_v \le t). \tag{5.9}$$

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

$$P(S_v \le t) \le Ee^{\theta^* S_v}e^{-\theta^* t} = \left(\frac{\lambda_a}{\lambda_a - \theta^*}\right)^{|v|} e^{-\theta^* t} = \left(\frac{1}{1+e}\right)^{|v|} e^{e\lambda_a t}. \tag{5.10}$$

But

$$\sum_{v2I_m} \left(\frac{1}{1+e}\right)^{jvj} = \sum_{i_1 \quad 1,\ldots,i_m \quad 1} \left(\frac{1}{1+e}\right)^{\sum_{j=1}^m i_j} = \left(\sum_{i \quad 1} \left(\frac{1}{1+e}\right)^i\right)^m = e^{\quad m}. \tag{5.11}$$

Combining (5.10), (5.11) and (5.9) yields (5.8). □

5.1.4 Occurrence of Nakamoto block

If the growth rate of the adversarial tree is greater than λ_h, then the adversary can always attack the honest chain by growing a side chain at a rate faster than the honest chain's growth rate and replace it at will. (5.7) immediately shows that if $\lambda_a > \lambda_h/e$, i.e. when the adversarial fraction $\beta > 1/(1+e)$, the growth rate of the adversarial tree is at least 1, and hence the private attack is successful. This is what [13] showed. The question we want to answer is what happens when $\beta < 1/(1+e)$? Will another attack work? We show below that in this regime, Nakamoto block has a non-zero probability of occurrence, and this implies that no attack works.

Lemma 5. If $\lambda_a < \lambda_h/e$, i.e. $\beta < 1/(1+e)$, then there exists a strictly positive constant $p > 0$ such that $P(F_j) \geq p$ for all j. Also, with probability 1, the event F_j occurs for infinitely many j's.

The proof of this result can be found in Appendix C.1.

5.1.5 Waiting time for Nakamoto block

In the previous section, we established the fact that the event F_j has $P(F_j) > p > 0$ for all j. This implies that the event F_j occurs infinitely often. But how long do we need to wait for such an event to occur? We answer this question in this section.

More specifically, we would like 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:

$$q[s, s+t] := P\Big(\bigcap_{j:\tau_j 2[s,s+t]} F_j^c\Big).$$

Lemma 6. If $\lambda_a < \lambda_h/e$, i.e. $\beta < 1/(1+e)$ then there exist constants a_2, A_2 so that, for any $s, t \geq 0$,

$$q[s, s+t] \;\; \leq \;\; A_2 \exp(-a_2\sqrt{t}). \tag{5.12}$$

The bound in (5.12) is not optimal, see Remark 1 below.

Proof. Define $R_j = \tau_{j+1} - \tau_j$, and let

$$B_{ik} = \text{event that } D_i(\textstyle\sum_{m=i}^{k\ \ 1} R_m) \;\geq\; (k-i-1). \tag{5.13}$$

(Notation as in (C.3).) Note that from Lemma 4 we have

$$\begin{aligned} P(B_{ik}) \;\; &\leq \;\; P\left(B_{ik}| \sum_{m=i}^{k\ \ 1} R_m < (k-i-1)\frac{\lambda_h + \lambda_a e}{2\lambda_a e}\frac{1}{\lambda_h}\right) \\ &\qquad\qquad + P\left(\sum_{m=i}^{k\ \ 1} R_m \geq (k-i-1)\frac{\lambda_h + \lambda_a e}{2\lambda_a e}\frac{1}{\lambda_h}\right) \\ &\leq \;\; e^{\ \frac{\lambda_h \ \ \lambda_a e}{2\lambda_h}(k\ \ i\ \ 1)} + A_1 e^{\ \ \alpha_1(k\ \ i\ \ 1)} \end{aligned} \tag{5.14}$$

for some positive constants A_1, α_1 independent of k, i. The first term in the last inequality follows from (5.8), and the second term follows from the fact that $(\lambda_h + \lambda_a e)/(2\lambda_a e) > 1$ and the R_i's are i.i.d. exponential random variables of mean $1/\lambda_h$. Then

$$F_j^c = \bigcup_{(i,k): i<j, k>j} B_{ik}. \tag{5.15}$$

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 adversarial 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,

$$C_\ell = \bigcap_{j:\tau_j \in \mathcal{J}_{3\ell+1}} \ \bigcup_{(i,k):\tau_j - \sqrt{t}<\tau_i<\tau_j, \tau_j<\tau_k<\tau_j+\sqrt{t}} B_{ik}.$$

Note that for distinct ℓ, the events C_ℓ's are independent. Also, we have

$$P(C_\ell) \le P(\text{no arrival in } \mathcal{J}_{3\ell+1}) + 1 - p < 1 \tag{5.16}$$

for large enough t.

Introduce the atypical events:

$$B = \bigcup_{(i,k):\tau_i \in [s,s+t] \text{ or } \tau_k \in [s,s+t], i<k, \tau_k - \tau_i > \sqrt{t}} B_{ik} \text{ , and} \tag{5.17}$$

$$\tilde{B} = \bigcup_{(i,k):\tau_i<s, s+t<\tau_k} B_{ik} \text{ .} \tag{5.18}$$

The events B and $\tilde{B}$ are the events that an adversarial tree catches up with an honest block far ahead. Consider also the events

$$D_1 = \{\#\{i : \tau_i \in (s-\sqrt{t}, s+t+\sqrt{t})\} > 2\lambda_h t\} \tag{5.19}$$

$$D_2 = \{\exists i, k : \tau_i \in (s, s+t), (k-i) < \sqrt{t}/2\lambda_h, \tau_k - \tau_i > \sqrt{t}\} \tag{5.20}$$

$$D_3 = \{\exists i, k : \tau_k \in (s, s+t), (k-i) < \sqrt{t}/2\lambda_h, \tau_k - \tau_i > \sqrt{t}\} \tag{5.21}$$

In words, D_1 is the event of atypically many honest arrivals in $(s-\sqrt{t}, s+t+\sqrt{t})$ 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 the number of honest arrivals in $(s, s+t)$ is Poisson with parameter $\lambda_h t$, we have from the memoryless property of the Poisson process that $P(D_1) \le e^{-c_0 t}$ for some constant $c_0 = c_0(\lambda_a, \lambda_h) > 0$. On the other hand, using the memoryless property and a union bound, and decreasing c_0 if needed, we have that $P(D_2) \le e^{-c_0\sqrt{t}}$. Similarly, using time reversal, $P(D_3) \le e^{-c_0\sqrt{t}}$. Therefore, again using the memoryless property of the Poisson process,

$$\begin{aligned} P(B) &\le P(D_1 \cup D_2 \cup D_3) + P(B \cap D_1^c \cap D_2^c \cap D_3^c) \\ &\le e^{-c_0 t} + 2e^{-c_0\sqrt{t}} + \sum_{i=1}^{2\lambda_h t} \sum_{k:k-i>\sqrt{t}/2\lambda_h} P(B_{ik}) \le c_1 e^{-c_2\sqrt{t}}, \end{aligned} \tag{5.22}$$

where $c_1, c_2 > 0$ are constants that may depend on λ_a, λ_h and the last inequality is due to (5.14). We next claim that there exists a constant $\alpha > 0$ so that, for all t large,

$$P(\tilde{B}) \leq e^{\;\alpha t}. \tag{5.23}$$

Indeed, we have that

$$\begin{aligned} P(\tilde{B}) &= \sum_{i<k} \int_0^s P(\tau_i \in d\theta) P(B_{ik}, \tau_k - \tau_i > s + t - \theta) \\ &\leq \sum_i \int_0^s P(\tau_i \in d\theta) \sum_{k:k>i} P(B_{i,k})^{1/2} P(\tau_k - \tau_i > s + t - \theta)^{1/2}. \end{aligned} \tag{5.24}$$

By (5.14), there exists $c_3 > 0$ so that

$$P(B_{i,k}) \leq e^{\;c_3(k\;\;i\;\;1)}, \tag{5.25}$$

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

$$P(\tau_k - \tau_i > s+t-\theta) = P(\tau_{k\;\;i} > s+t-\theta) \leq \begin{cases} 1, & (k-i) > c(s+t-\theta) \\ e^{\;c'(s+t\;\;\theta)}, & (k-i) \leq c(s+t-\theta). \end{cases} \tag{5.26}$$

Combining (5.25) with (5.26) yields that there exists a constant $\alpha > 0$ so that

$$\sum_{k:k>i} P(B_{i,k})^{1/2} P(\tau_k - \tau_i > s + t - \theta)^{1/2} \leq e^{\;2\alpha(t+s\;\;\theta)}. \tag{5.27}$$

Substituting this bound in (5.24) and using that $\sum_i P(\tau_i \in d\theta) = d\theta$ gives

$$\begin{aligned} P(\tilde{B}) &\leq \sum_i \int_0^s P(\tau_i \in d\theta) e^{\;2\alpha(t+s\;\;\theta)} \\ &\leq \int_0^s e^{\;2\alpha(t+s\;\;\theta)} d\theta \leq e^{\;\alpha t}, \end{aligned} \tag{5.28}$$

for t large, proving (5.23).

Continuing with the proof of the lemma, we have:

$$\begin{aligned} q[s, s+t] &\le P(B) + P(\tilde{B}) + P(\bigcap_{\ell=0}^{\sqrt{t}/3} C_\ell) = P(B) + P(\tilde{B}) + (P(C_\ell))^{\sqrt{t}/3} \\ &\le c_1 e^{-c_2 \sqrt{t}} + e^{-\alpha t} + (P(C_\ell))^{\frac{\sqrt{t}}{3}}, \end{aligned} \tag{5.29}$$

where the equality is due to independence, and in the last inequality we used (5.22) and (5.23). The lemma follows from (5.16). □

Remark 1. Iterating the proof above (taking longer blocks and using the bound of Lemma 6 to improve on $P(C_\ell)$ in (5.16) by replacing p with the bound from (5.12)) shows that (5.12) can be improved to the statement that for any $\theta > 1$ there exist constants a_θ, A_θ so that, for any $s, t > 0$,

$$q[s, s+t] \le A_\theta \exp(-a_\theta t^{1/\theta}). \tag{5.30}$$

5.2 Nonzero network delay: $\Delta > 0$

We will now extend the analysis in the above section to the case of non-zero delay.

In the case of zero network delay, the power of the adversary is in the adversarial blocks that it can generate by winning lotteries. We show that if $\beta < 1/(1+e)$, regardless of the adversarial strategy, there will be Nakamoto blocks occurring in the system once in a while. When the network delay is non-zero, the adversary has the additional power to delay delivery of honest blocks to create split view among the honest nodes. In the context of the security analysis of Nakamoto's PoW protocol, the limit of this power is quantified by the notion of ***uniquely successful*** round in [2] in the lock-step synchronous round-by-round model, and extended to the notion of ***convergence opportunity*** in [3] in the semi-synchronous model. (This notion is further used in [30] to provide a simpler security proof for Nakamoto's protocol.) They show that during these convergence opportunities, the adversary cannot create split view between honest nodes, because only one honest block is generated during a sufficiently long time interval. We combine our notion of Nakamoto block for zero delay with the notion of convergence opportunity to define a stronger notion of Nakamoto block for the non-zero delay case.

5.2.1 Random processes

We consider the network model in Chapter 3 with bounded communication delay, where all broadcast blocks are delivered by the adversary with maximum delay Δ. With this network model, the evolution of the blockchain can be modeled as a random process $\{(\mathcal{T}(t), \mathcal{C}(t), \mathcal{T}^{(p)}(t), \mathcal{C}^{(p)}(t) : t \geq 0, 1 \leq p \leq n\}$, where n is the number of honest nodes, $\mathcal{T}(t)$ is a tree, $\mathcal{T}^{(p)}(t)$ is an induced sub-tree of $\mathcal{T}(t)$ in the view of the p-th honest node at time t, and $\mathcal{C}^{(p)}(t)$ is the longest chain in the p-th tree. Then let $\mathcal{C}(t)$ be the common prefix of all the local honest chains $\mathcal{C}^{(p)}(t)$ for $1 \leq p \leq n$. The tree $\mathcal{T}(t)$ is interpreted as consisting of all the blocks that are generated by both the adversary and the honest nodes up until time t, including blocks that are kept in private by the adversary. The chain $\mathcal{C}^{(p)}(t)$ is interpreted as the longest chain in the local view of the p-th honest node at time t. The process is described as follows.

1. $\mathcal{T}(0) = \mathcal{T}^{(p)}(0) = \mathcal{C}(0) = \mathcal{C}^{(p)}(0), 1 \leq p \leq n$ is a single root block (the genesis block).

2. $\mathcal{T}(t)$ evolves as follows: there are independent Poisson processes of rate λ_a at each block of $\mathcal{T}(t)$ (we call them the ***adversary*** processes), plus an additional independent Poisson process of rate $\lambda_h^{(p)}$ (we call it the ***honest*** process) arriving at the last block of the chain $\mathcal{C}^{(p)}(t)$ (the tip of the local longest chain) for each $1 \leq p \leq n$, with $\sum_{p=1}^{n} \lambda_h^{(p)} = \lambda_h$. A new block is added to the tree at a certain block when an arrival event occurs at that node. An arrival from the honest process is called an honest block. An arrival from the adversary process is called an adversarial block.

3. The sub-tree $\mathcal{T}^{(p)}(t)$ for each $1 \leq p \leq n$ is updated in three possible ways : 1) an additional honest block can be added to $\mathcal{T}^{(p)}(t)$ by the adversary if an arrival event of the honest process with the p-th honest node occurs; 2) a block (whether is honest or adversarial) must be added to $\mathcal{C}^{(p)}(t)$ if it appears in $\mathcal{T}^{(q)}$ for some $q \neq p$ at time $t - \Delta$; 3) the adversary can replace $\mathcal{T}^{(p)}(t\ )$ by another sub-tree $\mathcal{T}^{(p)}(t)$ from $\mathcal{T}(t)$ as long as $\mathcal{T}^{(p)}(t\ )$ is an induced subgraph of the new tree $\mathcal{T}^{(p)}(t)$. The adversary's decision has to be based on the current state of the process.

4. $\mathcal{C}^{(p)}(t)$ is updated as follows for each $1 \leq p \leq n$: $\mathcal{C}^{(p)}(t)$ is the longest chain in the tree $\mathcal{T}^{(p)}(t)$ starting from the root block at time t. If there are more than one longest chain, tie breaking is in favor of the adversary.

5. $\mathcal{C}(t)$ is updated as follows: $\mathcal{C}(t)$ is the common prefix of all the local honest chains $\mathcal{C}^{(p)}(t)$ for $1 \leq p \leq n$ at time t.

The adversary can change where the honest nodes act by broadcasting an equal or longer length chain using the blocks it has succeeded in proposing. Since the adversary can change where the honest nodes can propose even with an equal length new chain, that means the adversary is given the ability to choose where the honest nodes propose when there are more than one longest public chain. Also the adversary has the ability to have one message delivered to honest nodes at different time (but all within Δ time).

5.2.2 Nakamoto block

We first define several basic random variables and random processes which are constituents of the processes $\mathcal{T}(\cdot)$ and $\mathcal{C}(\cdot)$. We make use of the terminology in [30].

1. τ_i = generation time of the i-th honest block; $\tau_0 = 0$ is the mining time of the genesis block, $\tau_{i+1} - \tau_i$ is exponentially distributed with mean $1/\lambda_h$, i.i.d. across all i's. Suppose an honest block B is generated at time τ_j. If $\tau_j - \tau_{j-1} > \Delta$, then we call B is a ***non-tailgater*** (otherwise, B is a ***tailgater***). If $\tau_j - \tau_{j-1} > \Delta$ and $\tau_{j+1} - \tau_j > \Delta$, then we call B is a ***loner***. Note that non-tailgaters have different depths and a loner is the only honest block at its depth.

2. $H_h(t)$ = number of non-tailgaters generated from time 0 to t.

3. $L^{(p)}(t)$ is the length of $\mathcal{C}^{(p)}(t)$ for each $1 \leq p \leq n$. $L^{(p)}(0) = 0$. Note that since every non-tailgater appears at different depth in the block tree, it follows that for all $t > s + \Delta$,

$$L^{(p)}(t) - L^{(p)}(s) \geq H_h(t - \Delta) - H_h(s). \tag{5.31}$$

4. $\mathcal{T}_i = \{\mathcal{T}_i(s) : s \geq 0\}$ is the random tree process generated by the adversary starting from the i-th honest block. $\mathcal{T}_i(0)$ consists of the i-th honest block and $\mathcal{T}_i(s)$ consists of all adversarial blocks grown on the i-th honest block from time τ_i to $\tau_i + s$. Note that the $\mathcal{T}_i$'s are i.i.d. copies of the adversarial tree $\mathcal{T}^a$.

5. $D_i(s)$ is the depth of the adversarial tree $\mathcal{T}_i(s)$.

We are now ready to put everything together to define Nakamoto blocks in general.

Definition 3. (**Nakamoto block for general** Δ) Define

$$\hat{E}_{ij} = \text{event that } D_i(t-\tau_i) < H_h(t-\Delta) - H_h(\tau_i) \text{ for all } t > \tau_j + \Delta, \tag{5.32}$$

$$\hat{F}_j = \bigcap_{0 \leq i<j} \hat{E}_{ij}, \tag{5.33}$$

$$U_j = \text{event that } j\text{-th honest block is a loner} = \{\tau_j - \tau_{j-1} > \Delta, \tau_{j+1} - \tau_j > \Delta\}, \tag{5.34}$$

and

$$\hat{U}_j = \hat{F}_j \cap U_j. \tag{5.35}$$

The j-th honest block is called a ***Nakamoto block*** if it is a loner and event $\hat{U}_j$ occurs.

And we have the following lemma, which justifies that Nakamoto blocks stabilize the blockchain for the non-zero delay case.

Lemma 7. If the j-th honest block is a Nakamoto block, then it will be in the chain $\mathcal{C}(t)$ (i.e. in all local chains $\mathcal{C}^{(p)}(t), 1 \leq p \leq n$) for all $t > \tau_j + \Delta$. This implies that the longest chain until the j-th honest block has stabilized.

Proof. We will argue by contradiction. Suppose $\hat{U}_j$ occurs and let $t^* > \tau_j + \Delta$ be the smallest t such that the j-th honest block is not contained in $\mathcal{C}(t)$. Let b_h be the last honest block on $\mathcal{C}^{(p)}(t^*)$ (which must exist, because the genesis block is by definition honest). If b_h is mined at some time $t_1 > \tau_j + \Delta$, then $\mathcal{C}^{(p)}(t_1^*)$ is the prefix of $\mathcal{C}^{(p)}(t^*)$ before block b_h, and does not contain the j-th honest block, contradicting the minimality of t . So b_h must be mined before time $\tau_j + \Delta$. And since the j-th honest block is a loner, we further know that b_h must be mined before time τ_j, hence b_h is the i-th honest block for some $i < j$. The part of $\mathcal{C}^{(p)}(t \ )$ after block b_h must lie entirely in the adversarial tree $\mathcal{T}_i(t \ - \tau_i)$ rooted at b_h. Hence, we have

$$D_i(t \ - \tau_i) < H_h(t \ - \Delta) - H_h(\tau_i) \leq L^{(p)}(t \ ) - L^{(p)}(\tau_i), \tag{5.36}$$

where the first inequality follows from the fact that $\hat{F}_j$ holds, and the second inequality follows from the longest chain policy (Equation (5.31)). From this we obtain that

$$L^{(p)}(\tau_i) + D_i(t \ - \tau_i) < L^{(p)}(t \ ) \tag{5.37}$$

which is a contradiction since $L^{(p)}(t \ ) \leq L^{(p)}(\tau_i) + D_i(t \ - \tau_i)$. □

Note that, Lemma 7 implies that if $\hat{U}_j$ occurs, then the entire chain leading to the j-th honest block from the genesis is stabilized after the j-th honest block is seen by all the honest nodes.

5.2.3 Occurrence of Nakamoto block

Lemma 8. If $\lambda_a < g/e \cdot \lambda_h$, i.e. $\beta < g/(g+e)$ with $g = e^{\ \lambda_h \Delta}$, then there is a $p > 0$ such that $P(\hat{U}_j) \geq p$ for all j. Also, with probability 1, the event $\hat{U}_j$ occurs for infinitely many j's.

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

5.2.4 Waiting time for Nakamoto block

We have established the fact that the event $\hat{U}_j$ has $P(\hat{U}_j) > p > 0$ for all j. In analogy to the zero-delay case, we would like 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:

$$\tilde{q}[s, s+t] := P(\bigcap_{j:\tau_j 2 [s,s+t]} \hat{U}_j^c).$$

The following lemma is analogous to Lemma 6 in the zero-delay case. Its proof is almost verbatim identical, and will not be repeated here.

Lemma 9. If $\lambda_a < g/e \cdot \lambda_h$, i.e. $\beta < g/(g+e)$, then there exist constants $\bar{a}_2, \bar{A}_2$ so that for all $s, t \geq 0$,

$$\tilde{q}[s, s+t] \leq \bar{A}_2 \exp(-\bar{a}_2 \sqrt{t}). \tag{5.38}$$

The improvement mentioned in Remark 1 applies here as well.

5.3 Persistence and liveness

We will now use Lemma 9 to establish the persistence and liveness of the basic longest chain PoS protocol.

In the absence of any adversary, each node will contribute to the final ledger as many blocks as their proportion of the stake. In the presence of an adversary, the chain quality property ensures that the contribution of the adversary is bounded. When $\beta < g/(g+e)$, we show that these properties hold with high probability, as stated in the following theorem.

Our goal is to generate a transaction ledger that satisfies ***persistence*** and ***liveness*** as defined in [2]. Together, persistence and liveness guarantees robust transaction ledger; honest transactions will be adopted to the ledger and be immutable.

Definition 4 (from [2])**.** 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:

- (Persistence) Parameterized by $\tau \in \mathsf{R}$, 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 $u \in \mathsf{R}$, 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 main result is that common prefix, chain quality, and chain growth imply that the transaction ledger satisfies persistence and liveness.

Theorem 1. Distributed nodes running Nakamoto-PoS protocol generates a transaction ledger satisfying *persistence* (parameterized by $\tau = \sigma$) and *liveness* (parameterized by $u = \sigma$) in Definition 4 with probability at least $1 - e^{-\Omega(\sqrt{\sigma})}$.

Proof. We first prove persistence and then liveness.

Lemma 10 (Persistence)**.** The public transaction ledger maintained by Nakamoto-PoS satisfies persistence parameterized by $\tau = \sigma$ with probability at least $1 - e^{-\Omega(\sqrt{\sigma})}$.

Proof. For a chain $\mathcal{C}_t$ with the last block generated at time t, let $\mathcal{C}_t^{\lceil\sigma}$ be the chain resulting from pruning a chain $\mathcal{C}_t$ up to σ, by removing the last blocks at the end of the chain that were generated after time $t - \sigma$. Note that $\mathcal{C}^{\lceil\sigma}$ is a prefix of $\mathcal{C}$, which we denote by $\mathcal{C}^{\lceil\sigma} \preceq \mathcal{C}$.

Let $\mathcal{C}_1$ be the main chain of an honest node P_1 at time t_1. Suppose a transaction tx is contained in $\mathcal{C}_1^{\lceil\sigma}$ at round t_1, i.e., it is confirmed by P_1. Consider a main chain $\mathcal{C}_2$ of an honest node P_2 at some time $t_2 \geq t_1$. The *σ-common prefix property* ensures that after pruning a longest chain, it is a prefix of all future longest chains in the local view of any honest node. Formally, it follows that $\mathcal{C}_1^{\lceil\sigma} \preceq \mathcal{C}_2$, which completes the proof.

We are left to show that the σ-common prefix defined below holds with a probability at least $1 - e^{-\Omega(\sqrt{\sigma})}$. This is a variation of a similar property first introduced in [2] for PoW systems. Ours is closer to a local definition of k-common prefix introduced in [31], which works for a system running for an unbounded time.

Definition 5 (σ-common prefix)**.** We say a protocol and a corresponding confirmation rule have a *σ-common prefix property* at time t, if in the view $\mathsf{VIEW}^{n,F}_{\Pi,A,Z}(\kappa)$ of a honest node n at time t, n adopts a longest chain $\mathcal{C}$, then any longest chain $\mathcal{C}'$ adopted by some honest node n' at time $t' > t$ satisfies $\mathcal{C}^{\lceil\sigma} \preceq \mathcal{C}'$.

Let $\mathcal{C}_t$ denote the longest chain adopted by an honest node with the last node generated at time t. There are a number of honest nodes generated in the interval $[t - \sigma, t]$, each of which can be in $\mathcal{C}_t$, $\mathcal{C}_{t'}$, or neither. We partition the set of honest blocks generated in that interval with three sets: $\mathcal{H}_t \triangleq \{H_j \in \mathcal{C}_t : \tau_j \in [t - \sigma, t]\}, \mathcal{H}_{t'} \triangleq \{H_j \in \mathcal{C}_{t'} : \tau_j \in [t - \sigma, t]\}$, and

$\mathcal{H}_{\text{rest}} \triangleq \{H_j \notin \mathcal{C}_t \cup \mathcal{C}_{t'} : \tau_j \in [t-\sigma, t]\}$, depending on which chain they belong to.

Suppose $\mathcal{C}_t^{\lceil\sigma} \not\preceq \mathcal{C}_{t'}$, and we will show that this event is unlikely. Under this assumption, we claim that none of the honest blocks generated in the interval $[t-\sigma, t]$ are ***stable***, i.e. for each honest block, there exists a time in the future (since the generation of that block) in which the block does not belong to the longest chain.

Precisely, we claim that $\mathcal{C}_t^{\lceil\sigma} \not\preceq \mathcal{C}_{t'}$ implies that F_j^c holds for all j such that $\tau_j \in [t-\sigma, t]$. This in turn implies that $P(\mathcal{C}_t^{\lceil\sigma} \not\preceq \mathcal{C}_{t'}) \leq P(\cap_{j:\tau_j \in [t-\sigma,t]} F_j^c)$. By Lemma 9, we know that the probability of this happening is low: $e^{-\Omega(\sqrt{\sigma})}$.

This follows from the following facts. (i) The honest blocks in $\mathcal{C}_t$ does not make it to the longest chain at time t': $H_j \notin \mathcal{C}_{t'}$ for all $H_j \in \mathcal{H}_t$. This follows from $\mathcal{C}_t^{\lceil\sigma} \not\preceq \mathcal{C}_{t'}$. (ii) The honest blocks in $\mathcal{C}_{t'}$ does not make it to the longest chain $\mathcal{C}_t$ at time t: $H_j \notin \mathcal{C}_t$ for all $H_j \in \mathcal{H}_{t'}$. This also follows from $\mathcal{C}_t^{\lceil\sigma} \not\preceq \mathcal{C}_{t'}$. (iii) The rest of the honest blocks did not make it to either of the above: $H_j \notin \mathcal{C}_t \cup \mathcal{C}_{t'}$ for all $H_j \in \mathcal{H}_{\text{rest}}$.

□

We next prove liveness.

Lemma 11 (Liveness)**.** The public transaction ledger maintained by Nakamoto-PoS satisfies liveness parameterized by $u = \sigma$ with probability at least $1 - e^{-\Omega(\sqrt{\sigma})}$.

Proof. Assume a transaction tx is received by all honest nodes at time t, then by Lemma 9, we know that with probability at least $1 - e^{-\Omega(\sqrt{\sigma})}$, there exists one honest block B_j mined at time τ_j with $\tau_j \in [t, t+u]$ and event F_j occurs, i.e., the block B_j and its ancestor blocks will be contained in any future longest chain. Therefore, tx must be contained in block B_j or one ancestor block of B_j since tx is seen by all honest nodes at time $t < \tau_j$. In either way, tx is stabilized forever. Thus, the lemma follows. □

□

5.4 Extending the analysis to general c

We can repeat the analysis in the previous sections for the c-Nakamoto-PoS for $c > 1$. Like for $c = 1$, the analysis basically boils down to the problem of analyzing the race between the adversarial tree and the fictitious purely honest chain. In Appendix D, we confine our analysis to a study of that race. It turns out that under c-correlation randomness, the adversarial tree is developed by another branching random walking process, but with a slowing growth amplification factor ϕ_c, such that $\phi_1 = e$ and $\phi_{\boldsymbol{1}} = 1$. So the security threshold is

$$\beta_c = \frac{e^{\lambda_h \Delta}}{e^{\lambda_h \Delta} + \phi_c}$$

and it goes from

$$\frac{e^{\lambda_h \Delta}}{e^{\lambda_h \Delta} + e}$$

for $c = 1$ to

$$\frac{e^{\lambda_h \Delta}}{e^{\lambda_h \Delta} + 1}$$

for $c = \infty$. Under the c-correlation protocol, we have

$$\phi_c = -\frac{c\theta_c}{\log(-\theta_c) + (c-1)\log(1-\theta_c)},$$

where θ_c is the unique negative solution of Equation (5.39)

$$-\log(-\theta_c) - (c-1)\log(1-\theta_c) = -1 + (c-1)\frac{\theta_c}{1-\theta_c}. \tag{5.39}$$

We numerically compute the value of ϕ_c and β_c with $\Delta = 0$ in Table 5.1.

Table 5.1: Numerically computed growth rate ϕ_c and stake threshold β_c with $\Delta = 0$.

c	1	2	3	4	5
ϕ_c	e	2.22547	2.01030	1.88255	1.79545
β_c	$\frac{1}{1+e}$	0.31003	0.33219	0.34691	0.35772

c	6	7	8	9	10
ϕ_c	1.73110	1.68103	1.64060	1.60705	1.57860
β_c	0.36615	0.37299	0.37870	0.38358	0.38780

In the deployment of the Ouroboros protocol in the Cardano project,[3] each slot takes 20 seconds and each epoch is chosen to be 5 days [21], that is a common randomness will be shared by 21600 blocks. In Fig. 1.2, we plot the security threshold β_c against c up to $c = 21600$ for our c-Nakamoto-PoS. It turns out the security threshold of c-Nakamoto-PoS can approach 1/2 very closely even when c is much less than 21600, which is the current randomness update frequency in the Cardano project.

[3] https://www.cardano.org

CHAPTER 6

SECURITY ANALYSIS OF THE DYNAMIC STAKE PROTOCOL

6.1 Permissionless PoS

One major advantage of the Nakamoto protocol in the proof of work setting is its permissionless setting: anyone can join (leave) the system by simply contributing (extricating) computing power for the mining process. Under the PoS setting the stakes are used in lieu of the computing power. To support a protocol that is close to a permissionless system, we need to handle the case when the stake is dynamically varying. Unlike the case of compute power, the entry/removal of stake has to be more carefully orchestrated: if a change in the stake takes effect immediately after the transaction has been included in the blockchain, then this gives an opportunity for the adversary to grind on the secret key of the header (time, secret key, common source of randomness). Concretely, once an adversary has generated a block, it can add a transaction that moves all its stake to a new coin with a new pair of public and secret keys. The adversary can keep drawing a new coin and simulating the next leader election, until it finds one that wins. This is a serious concern as the adversary can potentially win all elections.

To prevent such a grinding on the coin, we use s-truncation introduced in [11, 13]. s-truncation has two components: using stakes from ancestor blocks and a fork choice rule. The winning probability of a leader election uses the stake computed at an ancestor block which is s blocks above the parent block that is currently being mined on. However, this allows an adversary to launch a long range attack, for any value of $s < \infty$. To launch a long range attack, an adversary grows a private block tree. Once it has grown for longer than s blocks, then it can grind on the coin to win the election for the next block, and add the favorable transaction to the first ancestor block. As all blocks in this private tree are adversarial, the consistency of the transactions can be maintained by re-signing all intermediate blocks. This allows the adversary to win all elections after s private blocks, and eventually take over the honest block chain.

The s-truncation longest chain rule works as follows. When presented with a chain that forks from current longest chain, a node compares the two chains according to the following rule. Both chains are truncated up to s blocks after the forking. Whichever truncated chain was created in a shorter time (and hence denser) is chosen to be mined on. This ensures that honest block chains will be chosen over privately grown adversarial chains with long range attacks. A detailed description of the c-correlation, s-truncation, Nakamoto-PoS protocol is in Chapter 4.

6.2 Security of s-truncation scheme

We extend Theorem 1 to show that the above s-truncation scheme is secure.

Theorem 2. Under the dynamic stake setting, distributed nodes running Nakamoto-PoS protocol with a choice of $s = \Theta(\sigma)$ generates a transaction ledger satisfying ***persistence*** and ***liveness*** in Definition 4 with probability at least $1 - e^{\Omega(\sqrt{\sigma})}$.

Proof sketch. We prove it in three steps. First, we show that with static stake setting, common prefix, chain growth, and chain quality properties still hold for s-truncation protocol with $s = \Theta(\sigma)$. Second, we show that with dynamic stake setting and s stake update rule, the adversary can mine a private chain with consecutive s blocks that is denser than the public main chain only with a negligible probability.

We show in Chapter 5 that the choice of c only determines how many fraction of adversary the system can tolerate and not the security parameter κ. On the other hand, the choice of s is critically related to the target security parameter κ. By decoupling these two parameters c and s in the protocol, we can achieve any level of predictability (with an appropriate choice of c), while managing to satisfy any target security parameter κ (with an appropriate choice of s).

CHAPTER 7

CONCLUSION

We proposed a new family of PoS protocols and proved that our proposed PoS protocols have low predictability while guaranteeing security against adversarial nothing-at-stake attacks. We did not discuss the design of incentive systems that encourage users to follow the honest protocol. We point out that there are existing ideas that could be naturally adapted for our problem. For example, one way to minimize NaS attacks is to require users to deposit stake that can be slashed if the node has a provable deviation (for example, double-signing blocks) [8, 32]. Another important idea is that fruitchain-type incentive mechanisms [33] which protect against selfish mining in PoW can be ported to PoS protocols [9]. However, as pointed out in [12], the full problem of designing PoS protocols that strongly (instead of weakly) disincentivize NaS and selfish-mining attacks remains an important direction of future research. Finally, a detailed mathematical modeling of bribing attacks that consider interaction between the blockchain and external coordination mechanisms also remains open.

APPENDIX A

VERIFIABLE RANDOM FUNCTIONS

Definition 6 (from [23])**.** A function family $F_{(\,)}(\cdot) : \{0,1\}^{a(\kappa)} \rightarrow \{0,1\}^{b(\kappa)}$ is a family of VRFs is there exists a probabilistic polynomial-time algorithm $\textsc{Gen}$ and deterministic algorithms $\text{VRF}\textsc{prove}$ and $\text{VRF}\textsc{verify}$ such that $\textsc{Gen}(1^\kappa)$ outputs a pair of keys (pk, sk); $\text{VRF}\textsc{prove}(x, sk)$ computes $(F_{sk}(x), \pi_{sk}(x))$, where $\pi_{sk}(x)$ is the proof of correctness; and the algorithm $\text{VRF}\textsc{verify}(x, y, \pi, pk)$ verifies that $y = F_{sk}(x)$ using the proof π. Formally, we require

1. **Uniqueness:** no values $(pk, x, y_1, y_2, \pi_1, \pi_2)$ can satisfy the equation $\text{VRF}\textsc{verify}(x, y_1, \pi_1, pk) = \text{VRF}\textsc{verify}(x, y_2, \pi_2, pk)$ when $y_1 \neq y_2$.

2. **Provability:** if $(y, \pi) = \text{VRF}\textsc{prove}(x, sk)$, then $\text{VRF}\textsc{verify}(x, y, \pi, pk) = 1$.

3. **Pseudorandomness:** for any probabilistic polynomial-time algorithm $\mathcal{A} = (A_1, A_2)$, who does not query its oracle on x,

$$\Pr\left[z = z' \;\middle|\; \begin{array}{l} (pk, sk) \leftarrow \textsc{Gen}(1^\kappa); \\ (x, st) \leftarrow A_1^{\text{VRF}\textsc{prove}(\,)}(pk); \\ y_0 = F_{sk}(x); \\ y_1 \leftarrow \{0,1\}^{b(k)}; \\ z \leftarrow \{0,1\}; \\ z' \leftarrow A_2^{\text{VRF}\textsc{prove}(\,)}(y_z, st) \end{array} \right] \leq \frac{1}{2} + negl(\kappa).$$

This ensures that the output of a VRF is computationally indistinguishable from a random number even if the public key pk and the function $\text{VRF}\textsc{prove}$ is revealed.

APPENDIX B

PREDICTION INSPIRED BRIBING ATTACKS

Proof-of-work (PoW) protocols such as Nakamoto's protocol for Bitcoin achieve high security while maintaining a high unpredictability as to which miners can propose future blocks. A very attractive feature of this PoW protocol is that nodes that mine a valid block have no further ability to update the block after they have solved the mining puzzle, since the nonce seals the block making it tamper-proof. Thus no node knows whether they have the power to propose the block till they solve the puzzle, and once they solve the puzzle, they have no future rights to alter the content.

This causality is reversed in proof-of-stake (PoS) protocols: usually, the node that is eligible to propose a block knows ***a priori*** of its eligibility before proposing a block. This makes PoS protocols vulnerable to a new class of serious attacks not found in the PoW setting. We will show that a set of miners controlling an infinitesimal fraction of the stake can potentially completely undermine the security of the protocol. We demonstrate that the longer the prediction window, the more serious the attack space is. This raises an important questions as to whether it is possible to design a secure proof-of-stake protocol which has minimal prediction window.

We point out that an existing work [12] has already raised this issue that PoS protocols are forced make a tradeoff between predictability and NaS attacks. While that work mainly concerned itself with incentive attacks, our concern here is adversarial attacks that compromise consensus. We point out that even in the adversarial setting, all provably secure PoS protocols have a long prediction window; the main contribution of this thesis is to design such a protocol and show its security up to 50% of adversarial stake.

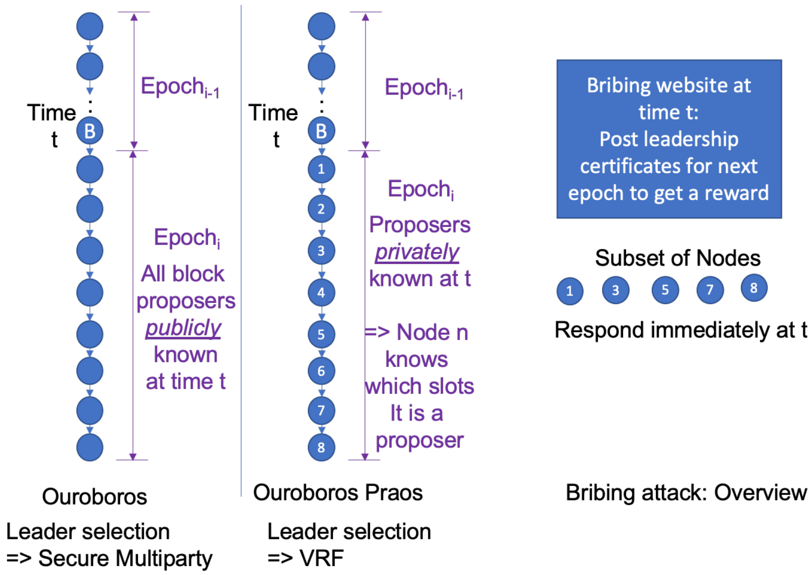

Figure B.1: Structure of bribing attack.

B.1 Adaptive adversaries and the VRF attack

A popular model that has been proposed to capture the effect of future prediction in the PoS setting is the so-called adaptive adversary model [10, 5]. In the adaptive adversary model, a node remains honest unless corrupted by an adversary (who can change who they are corrupting based on the public state). The adversary has a bound on how many nodes it can corrupt at any given time. To defend against an adaptive adversary model, many protocols have moved from global predictability of future block proposers (i.e., everyone knows who the future block leaders are) [9] to local predictability (i.e., each miner knows when in the future they will propose a block) [10, 5]. The local predictability is achieved using a Verifiable Random Function (VRF) [34] based leader-election.

However, the adaptive adversary model assumes that miners do not have any independent agency but rather only get corrupted based on an adversary's instructions. An adversary can easily circumvent this assumption by establishing a website where it can offer a bribe to anyone who posts their credentials for proposing blocks in an upcoming epoch of time. Thus even when the node's future proposer status is not public knowledge, this bribing website can solicit such information and help launch serious attacks (see Fig. B.1).

B.2 Longest chain protocols

We first consider longest-chain PoS protocols, in order to demonstrate our prediction attacks. We begin with a definition of prediction window of a protocol.

Definition 7 (W-predictable)**.** Given a PoS protocol Π_{PoS}, let $\mathcal{C}$ be a valid blockchain ending with block B with a time stamp t. We say a block B is $w(B)$-predictable, if there exists a time $t_1 > t$ and a block B_1 with a time stamp t_1 such that (i) B_1 can be mined by miner (using its private state and the common public state) at time t; and (ii) B_1 can be appended to $\mathcal{C}^0$ to form a valid blockchain for any valid chain $\mathcal{C}^0$ that extends $\mathcal{C}$ by appending $w - 1$ valid blocks with time stamps within the interval (t, t_1). By taking the maximum over the prediction parameter over all blocks in Π_{PoS}, i.e., let $W = \max_B w(B)$, we say Π_{PoS} is W-predictable. W is the size of the prediction window measured in units of number of blocks.

We note that our definition is similar to the definition of W-locally predictable protocols in [12]. We note furthermore that longest chain protocols also have a κ-deep confirmation policy, where a block embedded deep enough is deemed to be confirmed.

B.3 Prediction attack on W-predictable protocols

Let us consider a W-predictable protocol, where the prediction window W is longer than the confirmation window κ. We note that the prediction window of many existing protocols are quite large, as demonstrated in Table 2.1 and therefore this is a reasonable assumption. We will consider the alternative case ($W << \kappa$ in the upcoming section).

Consider block B that has been mined at time t (assume that B is W-predictable, since such a block exists by definition). The adversary launches a prediction attack by launching a website where it announces a reward for miners which possess a future block proposal slot. Some of the leaders respond to this call (these are shown with a red outer circle in Fig. B.2). We note that while the adversary requires $\kappa + 1$ leaders out of $2\kappa + 1$ slots to respond to the bribe, the total stake represented by these bribed leaders can

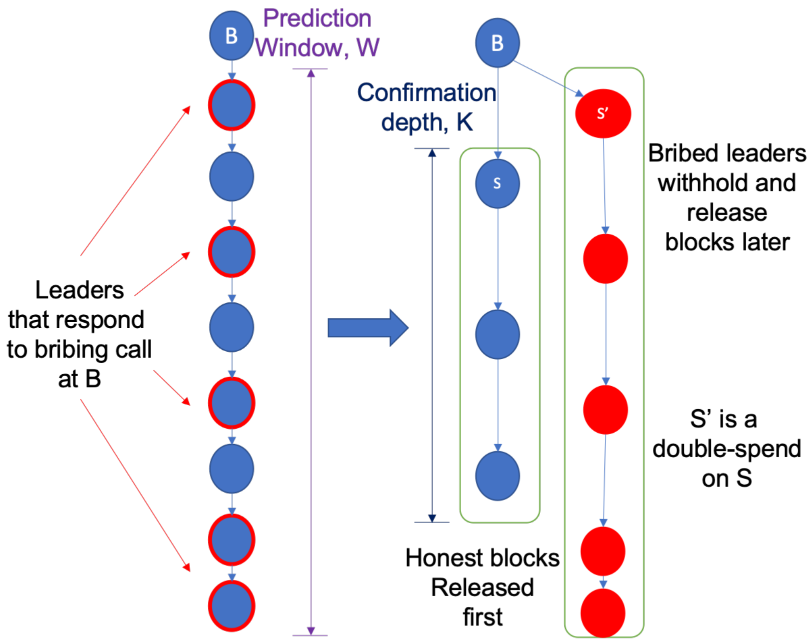

Figure B.2: Prediction attack on Ouroboros Praos.

be a very tiny fraction of the total stake. The adversary bribes these leaders to sign a forked version of the blockchain that it hoards till the block s is confirmed by a κ-deep honest chain. After that point, the adversary releases the hoarded blockchain to all the users thus switching the longest chain and confirming a block s^{0} (which contains a double-spend) instead of s. We note that this attack is indistinguishable from network delay since none of the bribed leaders sign multiple blocks with a single leadership certificate - thus nodes that participate in the bribing attack have plausible deniability.

We note that the previous attack can be launched whenever the prediction window W is larger than the confirmation depth κ. However, here we will briefly note that even when W is smaller than κ, the protocols still have a prediction problem. This is because in longest-chain PoS protocols such as Ouroboros, Praos, Snow White, the randomness is updated every epoch and a key assumption for the updated randomness to be unbiased is that a majority of the previous leaders were honest. However, by bribing the previous leaders, the adversary can bias the randomness (for example, by choosing a subset of proposers), thus leading itself to more favorable leadership slots in the upcoming epoch (and effectively enlarging the prediction window), as shown in Fig. B.3. These protocols offer no protection against these bribed

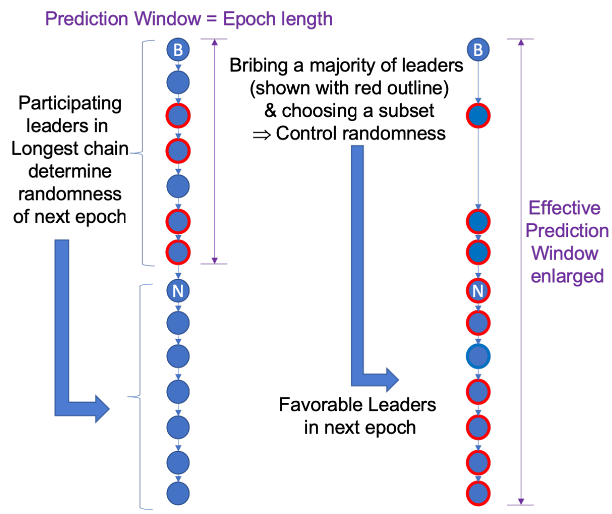

Figure B.3: Randomness grinding enlarges prediction ability.

randomness grinding attacks and hence their security parameter is limited to be of the same order as the epoch size. We note that while our proposed protocol has a structure similar to that of Ouroboros, our analysis proves security against adversarial randomness grinding (the so-called NaS attack).

B.4 Prediction attack on BFT-based PoS protocols

While we have focused on longest chain PoS protocols in the previous section, here we consider the other large family of PoS protocols that are based on Byzantine Fault Tolerant consensus (BFT). Some PoS-based BFT protocols work with the same committee for many time-slots thus giving raise to prediction based attacks. Furthermore, in many BFT protocols, a single leader proposes blocks till evicted for wrong-doing [6], thus making the prediction problem worse. Among BFT-based PoS protocols, the one with the least prediction window is Algorand [4, 5]. We will demonstrate a fatal prediction attack on Algorand (other BFT protocols which are even more predictable are naturally attacked as well).

In Algorand, at each round, a set of leaders and many sets of committees are elected using VRF from the previous finalized block. The leaders and committee members can construct their membership certificates from the previous finalized block's randomness, and others do not know their identities till they reveal themselves. The BFT consensus process proceeds in steps and

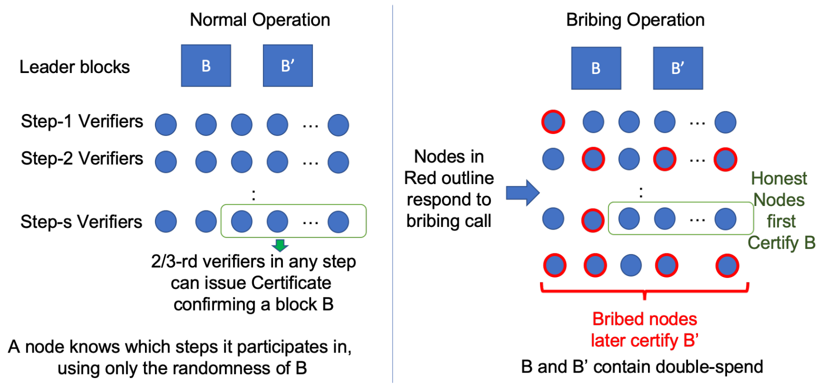

Figure B.4: Algorand bribing attack.

each step is run by a different committee - a block is considered finalized if it is voted on by a 2/3-majority of committee members at any step.

This feature is used to prove that Algorand is secure against adaptive adversaries. However, we show that Algorand is not immune to the bribing-based prediction attack (similar to the one for longest-chain protocols). Since the protocol does not use a sequence of blocks for confirmation, the prediction window defined for longest-chain protocols is not the appropriate measure for prediction-attacks in this protocol. Rather, the appropriate measure is what fraction of the committee participants are locally known when a block is proposed. For Algorand, all of the committee participants are known to themselves.

Suppose in a given round, there are two leaders blocks B and B^{0} (the latter block may contain a double-spend relative to B). The adversary solicits using a website the committee members to post their certificates. If the adversary is able to obtain a 2/3-quorum in any step, then the adversary can use that quorum to sign a different certificate than the one the honest nodes signed. In particular, the adversary waits for a honest quorum to sign a certificate for B in order to confirm the transaction and then signs a bribed quorum from a different step to certify B^{0}. This enables the adversary to reverse a confirmation. We note that none of the bribed miners have to double-sign a block since they would not have been elected in other steps of the quorum. We illustrate this bribing attack on Algorand in Fig. B.4.

We note that establishing a stalling-attack is even easier in this model - the adversary needs a 1/3-fraction of committee members in a step to remain silent in order to stall the progress of Algorand. Given that miners are randomly sampled, this quorum may hold a very small fraction of stake (which the adversary may compensate for using its bribe). This stalling attack can be used to launch extortion attacks demanding a large amount of money to un-stall the network.

B.5 Summary of prediction attacks

We have demonstrated that both longest-chain and BFT based protocols are highly vulnerable to prediction-based security attacks (compromising the safety and liveness of the system). This motivates the design of a new PoS protocol that can only be predicted with a look ahead window, much shorter than the confirmation window.

APPENDIX C

PROOFS FOR CHAPTER 5

C.1 Proof of Lemma 5

In this proof, we renormalize time such that $\lambda_h = 1$, and we set $\lambda_a = \lambda$ to simplify notations.

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 $\tau_{\ 1}, \tau_{\ 2}, \ldots$ such that together with $\tau_0, \tau_1, \ldots$ we have a double-sided infinite Poisson process of rate 1. Also, for each $i < 0$, we define an independent copy of a random adversarial tree $\mathcal{T}_i$ with the same distribution as $\mathcal{T}_0$.

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

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

Note that $E_j \subset F_j$, so to prove that F_j occurs for infinite many j's with probability 1, it suffices to prove that E_j occurs for infinite many j's with probability 1. This is proved in the following.

Define $R_j = \tau_{j+1} - \tau_j$ and

$$\mathcal{Z}_j = (R_j, \mathcal{T}_j). \tag{C.1}$$

Consider the i.i.d. process $\{\mathcal{Z}_j\}_{1<j<1}$. Now,

$$\begin{aligned}
E_{ij} &= \text{event that } D_i(t-\tau_i) < A_h(t) - A_h(\tau_i) \text{ for all } t > \tau_j \\
&= \text{event that } D_i(\tau_k-\tau_i) < A_h(\tau_k\ \ ) - A_h(\tau_i) \text{ for all } k > j \\
&= \text{event that } D_i(\tau_k-\tau_i) < A_h(\tau_k) - A_h(\tau_i) - 1 \text{ for all } k > j \\
&= \text{event that } D_i(\tau_k-\tau_i) < k-i-1 \text{ for all } k > j \\
&= \text{event that } D_i(\textstyle\sum_{m=i}^{k\ 1} R_m) < k-i-1 \text{ for all } k > j.
\end{aligned}$$

Hence $E_j = \cap_{i<j} E_{ij}$ has a time-invariant dependence on $\{\mathcal{Z}_i\}$. This means that $p = P(E_j)$ does not depend on j. Since $\{\mathcal{Z}_j\}$ is i.i.d. and in particularly ergodic, with probability 1, the long term fraction of j's for which E_j occurs is p, which is nonzero if $p \neq 0$. This is the last step to prove.

Let

$$E_0 = \text{event that } D_i(\textstyle\sum_{m=i}^{k\ 1} R_m) < k-i-1 \text{ for all } k>0 \text{ and } i<0 \tag{C.2}$$

and

$$B_{ik} = \text{event that } D_i(\textstyle\sum_{m=i}^{k\ 1} R_m) \geq k-i-1, \tag{C.3}$$

then

$$E_0^c = \bigcup_{k>0,i<0} B_{ik}. \tag{C.4}$$

Let us fix a particular $n > 0$, and define:

$$G_n = \text{event that } R_m < \tfrac{1}{n} \text{ for } m = -n, -n+1, \ldots, -1, 0, +1, \ldots, n-1. \tag{C.5}$$

Then

$$\begin{aligned}
P(E_0) &\geq P(E_0|G_n)P(G_n) && \text{(C.6)} \\
&= \left(1 - P(\cup_{k>0,i<0} B_{ik}|G_n)\right) P(G_n) && \text{(C.7)} \\
&\geq \left(1 - \sum_{k>0,i<0} P(B_{ik}|G_n)\right) P(G_n) && \text{(C.8)} \\
&\geq (1 - a_n - 2b_n - c_n) P(G_n), && \text{(C.9)}
\end{aligned}$$

where

$$a_n := \sum_{(i,k):\, -n \leq i<0<k \leq n} P(B_{ik}|G_n) \tag{C.10}$$

$$b_n := \sum_{(i,k):\, -n \leq i<0<n<k} P(B_{ik}|G_n) \tag{C.11}$$

$$c_n := \sum_{(i,k):i<-n<n<k} P(B_{ik}|G_n). \tag{C.12}$$

The last inequality (C.9) comes from the fact $P(B_{ik}|G_n) = P(B_{-k,-i}|G_n)$.

Using (5.8), we can bound $P(B_{ik}|G_n)$. Consider three cases:

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

For $k - i < \sqrt[4]{n}$, we have that

$$\begin{aligned} P(B_{ik}|G_n) &= P(D_i(\sum_{m=i}^{k-1} R_m) \geq k-i-1|G_n) \leq P(D_i(\frac{1}{\sqrt[4]{n^3}}) \geq 1) \\ &= P(\Gamma_\lambda \leq \frac{1}{\sqrt[4]{n^3}}) = 1 - e^{-\frac{\lambda}{\sqrt[4]{n^3}}} < \frac{\lambda}{\sqrt[4]{n^3}}, \end{aligned} \tag{C.13}$$

where $\Gamma_\lambda \sim \mathrm{Exp}(\lambda)$.

For $k - i \geq \sqrt[4]{n}$, we have that

$$\begin{aligned} P(B_{ik}|G_n) &= P(D_i(\sum_{m=i}^{k-1} R_m) \geq k-i-1|G_n) \\ &\leq P(D_i(\frac{k-i}{n}) \geq \sqrt[4]{n}-1) \\ &\leq P(D_i(2) \geq \sqrt[4]{n}-1) \leq e^{-\sqrt[4]{n}+1+2e\lambda}, \end{aligned} \tag{C.14}$$

where the last inequality follows from (5.8). Summing these terms, we have:

$$\begin{aligned} a_n &= \sum_{(i,k):\, -n \leq i<0<k \leq n} P(B_{ik}|G_n) \\ &\leq \sum_{(i,k):\, -n \leq i<0<k \leq n,\, k-i<\sqrt[4]{n}} \frac{\lambda}{\sqrt[4]{n^3}} + \sum_{(i,k):\, -n \leq i<0<k \leq n,\, k-i \geq \sqrt[4]{n}} e^{-\sqrt[4]{n}+1+2e\lambda} \\ &\leq \frac{\lambda}{\sqrt[4]{n}} + \sum_{(i,k):\, -n \leq i<0<k \leq n,\, k-i \geq \sqrt[4]{n}} e^{-\sqrt[4]{n}+1+2e\lambda} := \bar{a}_n, \end{aligned}$$

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

Case 2: $-n \le i < 0 < n < k$:

$$
\begin{aligned}
& P(B_{ik}|G_n) \\
\le\ & P\left(B_{ik}|G_n, 2 + \sum_{m=n}^{k-1} R_m < (k-i-1)\frac{1+\lambda e}{2\lambda e}\right) \\
& + P\left(2 + \sum_{m=n}^{k-1} R_m > (k-i-1)\frac{1+\lambda e}{2\lambda e}\right) \\
\le\ & e^{-\frac{1-\lambda e}{2}(k-i-1)} + A_1 e^{-\alpha(k-i-1)}
\end{aligned}
$$

for some positive constants A_1, α independent of n, k, i. The first term in the last inequality follows from (5.8), and the second term follows from the fact that $(1+\lambda e)/(2\lambda e) > 1$ and the R_i's have mean 1. Summing these terms, we have:

$$
\begin{aligned}
b_n &= \sum_{(i,k):\ -n \le i<0<n<k} P(B_{ik}|G_n) \\
&\le \sum_{(i,k):\ -n \le i<0<n<k} \left[e^{-\frac{1-\lambda e}{2}(k-i-1)} + A_1 e^{-\alpha(k-i-1)}\right] := \bar{b}_n,
\end{aligned}
$$

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

Case 3: $i < -n < n < k$:

$$
\begin{aligned}
& P(B_{ik}|G_n) \\
\le\ & P\left(B_{ik}|G_n, 2 + \sum_{m=-i}^{n-1} R_m + \sum_{m=n}^{k-1} R_m < (k-i-1)\frac{1+\lambda e}{2\lambda e}\right) \\
& + P\left(2 + \sum_{m=-i}^{n-1} R_m + \sum_{m=n}^{k-1} R_m > (k-i-1)\frac{1+\lambda e}{2\lambda e}\right) \\
\le\ & e^{-\frac{1-\lambda e}{2}(k-i-1)} + A_2 e^{-\alpha(k-i-1)}
\end{aligned}
$$

for some positive constants A_2, α independent of n, k, i. The first term in the last inequality follows from (5.8), and the second term follows from the fact that $(1+\lambda e)/(2\lambda e) > 1$ and the R_i's have mean 1.

Summing these terms, we have:

$$\begin{aligned} c_n &= \sum_{(i,k):i<\ \ n<n<k} P(B_{ik}|G_n) \\ &\leq \sum_{(i,k):i<\ \ n<n<k} \left[e^{\ \frac{1\ \ \lambda e}{2}(k\ \ i\ \ 1)} + A_2 e^{\ \alpha(k\ \ i\ \ 1)} \right] := \bar{c}_n, \end{aligned}$$

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

Substituting these bounds in (C.9) we finally get:

$$P(E_0) > [1 - (\bar{a}_n + 2\bar{b}_n + \bar{c}_n)]P(G_n). \tag{C.15}$$

By setting n sufficiently large such that $\bar{a}_n, \bar{b}_n$ and $\bar{c}_n$ are sufficiently small, we conclude that $P(E_0) > 0$.

C.2 Proof of Lemma 8

In this proof, we fix $g = e^{\ \lambda_h \Delta}$ and renormalize time such that $\lambda_h = 1$, and we set $\lambda_a = \lambda$ to simplify notations.

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 $\tau_{\ 1}, \tau_{\ 2}, \ldots$ such that together with $\tau_0, \tau_1, \ldots$ we have a double-sided infinite Poisson process of rate 1. Also, for each $i < 0$, we define an independent copy of a random adversarial tree $\mathcal{T}_i$ with the same distribution as $\mathcal{T}_0$.

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

$$\hat{E}_j = \bigcap_{i<j} \hat{E}_{ij}$$

and

$$\hat{V}_j = \hat{E}_j \cap U_j.$$

Note that $\hat{V}_j \subset \hat{U}_j$, so to prove that $\hat{U}_j$ occurs for infinite many j's with probability 1, it suffices to prove that $\hat{V}_j$ occurs for infinite many j's with probability 1. This is proved in the following.

Define $R_j = \tau_{j+1} - \tau_j$ and

$$\mathcal{Z}_j = (R_j, \mathcal{T}_j).$$

Consider the i.i.d. process $\{\mathcal{Z}_j\}_{1<j<1}$. Now,

$$\begin{aligned}
U_j \cap \hat{E}_{ij} &= U_j \cap \text{event that } D_i(t-\tau_i) < H_h(t-\Delta) - H_h(\tau_i) \text{ for all } t > \tau_j + \Delta \\
&= U_j \cap \text{event that } D_i(t+\Delta-\tau_i) < H_h(t) - H_h(\tau_i) \text{ for all } t > \tau_j \\
&= U_j \cap \text{event that } D_i(\tau_k \quad + \Delta - \tau_i) < H_h(\tau_k \quad) - H_h(\tau_i) \text{ for all } k > j \\
&= U_j \cap \text{event that } D_i(\textstyle\sum_{m=i}^{k\ 1} R_m + \Delta) < H_h(\tau_{k\ 1}) - H_h(\tau_i) \text{ for all } k > j.
\end{aligned}$$

Hence $\hat{E}_j \cap U_j = \bigcap_{i<j} \hat{E}_{ij} \cap U_j$ has a time-invariant dependence on $\{\mathcal{Z}_i\}$, which means that $p = P(\hat{V}_j)$ does not depend on j. Since $\{\mathcal{Z}_j\}$ is i.i.d. and in particularly ergodic, with probability 1, the long term fraction of j's for which $\hat{V}_j$ occurs is p, which is nonzero if $p \neq 0$. This is the last step to prove.

$$P(\hat{V}_0) = P(\hat{E}_0|U_0)P(U_0) = P(\hat{E}_0|U_0)P(R_0 > \Delta)P(R_{\ 1} > \Delta) = g^2 P(\hat{E}_0|U_0).$$

It remains to show that $P(\hat{E}_0|U_0) > 0$.

$$\hat{E}_0 = \text{event that } D_i(\textstyle\sum_{m=i}^{k\ 1} R_m + \Delta) < H_h(\tau_{k\ 1}) - H_h(\tau_i) \text{ for all } k > 0 \text{ and } i < 0. \tag{C.16}$$

Let

$$\hat{B}_{ik} = \text{event that } D_i(\textstyle\sum_{m=i}^{k\ 1} R_m + \Delta) \geq H_h(\tau_{k\ 1}) - H_h(\tau_i), \tag{C.17}$$

then

$$\hat{E}_0^c = \bigcup_{k>0, i<0} \hat{B}_{ik}. \tag{C.18}$$

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

$$G_n = \text{event that } D_m(3n) = 0 \text{ for } m = -n, -n+1, \ldots, -1, 0, +1, \ldots, n-1, n. \tag{C.19}$$

Then

$$
\begin{aligned}
P(\hat{E}_0|U_0) &\geq P(\hat{E}_0|U_0, G_n)P(G_n|U_0) && \text{(C.20)}\\
&= \left(1 - P(\cup_{k>0,i<0}\hat{B}_{ik}|U_0, G_n)\right) P(G_n|U_0) && \text{(C.21)}\\
&\geq \left(1 - \sum_{k>0,i<0} P(\hat{B}_{ik}|U_0, G_n)\right) P(G_n|U_0) && \text{(C.22)}\\
&\geq (1 - a_n - b_n)P(G_n|U_0), && \text{(C.23)}
\end{aligned}
$$

where

$$
\begin{aligned}
a_n &:= \sum_{(i,k):\ \ n\ \ i<0<k\ \ n} P(\hat{B}_{ik}|U_0, G_n) && \text{(C.24)}\\
b_n &:= \sum_{(i,k):i<\ \ n \text{ or } k>n} P(\hat{B}_{ik}|U_0, G_n). && \text{(C.25)}
\end{aligned}
$$

Using (5.8), we can bound $P(\hat{B}_{ik}|U_0, G_n)$. Consider two cases:
Case 1: $-n \leq i < 0 < k \leq n$:

$$
\begin{aligned}
P(\hat{B}_{ik}|U_0, G_n) &= P(\hat{B}_{ik}|U_0, G_n, \sum_{m=i}^{k\ \ 1} R_m + \Delta \leq 3n) + P(\sum_{m=i}^{k\ \ 1} R_m + \Delta > 3n|U_0, G_n)\\
&\leq P(\sum_{m=i}^{k\ \ 1} R_m + \Delta > 3n|U_0, G_n)\\
&\leq P(\sum_{m=i}^{k\ \ 1} R_m > 5n/2|U_0)\\
&\leq P(\sum_{m=i}^{k\ \ 1} R_m > 5n/2)/P(U_0)\\
&\leq A_1 e^{\ \ \alpha_1 n}
\end{aligned}
$$

for some positive constants A_1, α_1 independent of n, k, i. The last inequality follows from the fact that R_i's are i.i.d. exponential random variables of mean 1. Summing these terms, we have:

$$
a_n = \sum_{(i,k):\ \ n\ \ i<0<k\ \ n} P(B_{ik}|U_0, G_n) \leq \sum_{(i,k):\ \ n\ \ i<0<k\ \ n} A_1 e^{\ \ \alpha_1 n} := \bar{a}_n,
$$

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:

$$W_{ik}^{\varepsilon} = \text{event that } H_h(\tau_{k-1}) - H_h(\tau_i) \geq (1-\varepsilon)g(k-i-1).$$

Then we have

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

Let X_j be a Bernoulli random variable such that $X_j = 1$ if and only if $R_{j-1} > \Delta$, i.e., the j-th honest block is a non-tailgater. Since R_j's are i.i.d. exponential random variables with mean 1, we have that X_j's are also i.i.d. and $P(X_j = 1) = g$. By the definition of $H_h(\cdot)$, we have $H_h(\tau_{k-1}) - H_h(\tau_i) = \sum_{j=i+1}^{k-1} X_j$, then

$$\begin{aligned}
P(W_{ik}^{\varepsilon\, c}|U_0, G_n) &= P(\sum_{j=i+1}^{k-1} X_j < (1-\varepsilon)g(k-i-1)|X_0 = 1, X_1 = 1) \\
&= P(\sum_{j=i+1}^{-1} X_j + \sum_{j=2}^{k-1} X_j < (1-\varepsilon)g(k-i-1) - 2) \\
&\leq P(\sum_{j=i+1}^{-1} X_j + \sum_{j=2}^{k-1} X_j < (1-\varepsilon)g(k-i-3)) \\
&\leq A_2 e^{-\alpha_2(k-i-3)} \qquad \text{(C.26)}
\end{aligned}$$

for some positive constants A_2, α_2 independent of n, k, i. The last inequality follows from the Chernoff bound.

Meanwhile, we have

$$
\begin{aligned}
& P(\hat{B}_{ik}|U_0, G_n, W_{ik}^{\varepsilon}) \\
\leq\ & P(D_i(\sum_{m=i}^{k-1} R_m + \Delta) \geq (1-\varepsilon)g(k-i-1))|U_0, G_n, W_{ik}^{\varepsilon}) \\
\leq\ & P(D_i(\sum_{m=i}^{k-1} R_m + \Delta) \geq (1-\varepsilon)g(k-i-1)) \\
& \qquad\qquad\qquad | \ U_0, G_n, W_{ik}^{\varepsilon}, \sum_{m=i}^{k-1} R_m + \Delta \leq (k-i-1)\frac{g+\lambda e}{2\lambda e}) \\
+\ & P(\sum_{m=i}^{k-1} R_m + \Delta > (k-i-1)\frac{g+\lambda e}{2\lambda e}|U_0, G_n, W_{ik}^{\varepsilon}) \\
\leq\ & e^{-\frac{(1-2\varepsilon)g-\lambda e}{2}(k-i-1)} + P(\sum_{m=i}^{k-1} R_m + \Delta > (k-i-1)\frac{g+\lambda e}{2\lambda e}|U_0, G_n, W_{ik}^{\varepsilon}),
\end{aligned}
$$

where the first term in the last inequality follows from (5.8), and the second term can also be bounded:

$$
\begin{aligned}
& P(\sum_{m=i}^{k-1} R_m + \Delta > (k-i-1)\frac{g+\lambda e}{2\lambda e}|U_0, G_n, W_{ik}^{\varepsilon}) \\
=\ & P(\sum_{m=i}^{k-1} R_m + \Delta > (k-i-1)\frac{g+\lambda e}{2\lambda e}|U_0, W_{ik}^{\varepsilon}) \\
\leq\ & P(\sum_{m=i}^{k-1} R_m + \Delta > (k-i-1)\frac{g+\lambda e}{2\lambda e})/P(U_0, W_{ik}^{\varepsilon}) \\
\leq\ & A_3 e^{-\alpha_3(k-i-1)}
\end{aligned}
$$

for some positive constants A_3, α_3 independent of n, k, i. The last inequality follows from the fact that $(g+\lambda e)/(2\lambda e) > 1$ and the R_i's have mean 1, while $P(U_0, W_{ik}^{\varepsilon})$ is a event with high probability as we showed in Equation (C.26).

Then we have

$$
P(\hat{B}_{ik}|U_0, G_n) \leq A_2 e^{-\alpha_2(k-i-3)} + e^{-\frac{(1-2\varepsilon)g-\lambda e}{2}(k-i-1)} + A_3 e^{-\alpha_3(k-i-1)}.
$$

Summing these terms, we have:

$$
\begin{aligned}
b_n &= \sum_{(i,k):i<\ \ n \text{ or } k>n} P(\hat{B}_{ik}|U_0, G_n) \\
&\leq \sum_{(i,k):i<\ \ n \text{ or } k>n} \left[A_2 e^{\ \ \alpha_2(k\ \ i\ \ 3)} + e^{\ \ \frac{(1\ \ 2\varepsilon)g\ \ \lambda e}{2}(k\ \ i\ \ 1)} + A_3 e^{\ \ \alpha_3(k\ \ i\ \ 1)}\right] := \bar{b}_n,
\end{aligned}
$$

which is bounded and moreover $\bar{b}_n \to 0$ as $n \to \infty$ when we set ε to be small enough such that $(1 - 2\varepsilon)g - \lambda e > 0$.

Substituting these bounds in Equation (C.23) we finally get:

$$P(\hat{E}_0|U_0) > [1 - (\bar{a}_n + \bar{b}_n)]P(G_n|U_0). \tag{C.27}$$

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

APPENDIX D

GROWTH RATE OF C-CORRELATED ADVERSARY TREE

We set $\lambda_a = \lambda$ in this section to simplify notations.

The adversary is growing a private chain over the genesis block, under the c-correlation. As illustrated in Fig. D.1, the common source of randomness at a block is only updated when the depth is a multiple of c (Algorithm 1 line:19). We refer to such a block b with $\text{depth}(b)\%c = 0$ as a ***godfather-block***.

$$\text{RandSource}(b) :=$$
$$\begin{cases} \text{VRF}\textsc{prove}\big(\text{RandSource}(\text{parent}(b)), \text{time}, \mathsf{VRF}.sk(b)\big), & \text{if depth}\big(b\big)\%c = 0, \\ \text{RandSource}(\text{parent}(b)), & \text{otherwise.} \end{cases}$$

The randomness of a block changes only at ***godfather-blocks***. In other words, for $n \in \mathsf{Z}$, blocks along a chain at depths $\big\{nc, nc+1, nc+2, \cdots, nc+c-1\big\}$ share a common random number. Two blocks are called ***siblings-blocks*** if they have the same parent block. Given this shared randomness, the adversary now has a freedom to choose where to place the newly generated blocks. The next theorem provides a dominant strategy, that creates the fastest growing private tree.

Lemma 12. Under c-correlation, the optimal adversarial strategy to grow the tree fast is to only fork at the parents of godfather-blocks.

Proof. Note that under the security model, several types of grinding attacks are plausible. First, at depth multiple of c, the adversary can grind on the header of the parent of the godfather block, and run an independent election in every round. Secondly, for blocks sharing the same source of randomness, once an adversary is elected a leader, it can generate multiple blocks of the same header but appending on different blocks. However, adding multiple blocks with the same header cannot make the tree any higher than the optimal scheme.

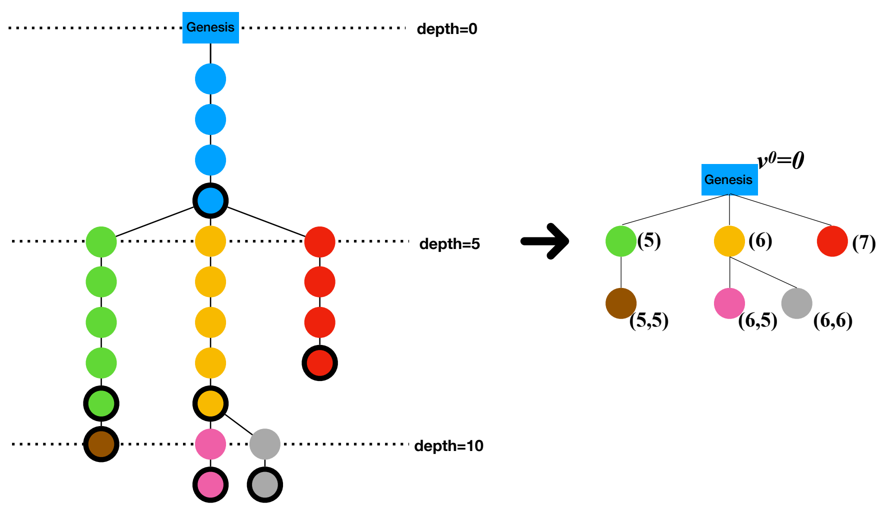

Figure D.1: An example of $T^{(a)}(t)$ with $c = 5$ under the optimal strategy to grow the private NaS tree. Blocks forking from the same godfather-block share the same common source of randomness, as shown by the colors. To grow the tree fast, it is optimal to grow a single chain until the next godfather block. Circles with black outlines indicate blocks that are currently mined on.

Sibling non-godfather blocks share a common source of randomness and thus "mining events" on these blocks are completely dependent. Specifically, for sibling non-godfather blocks, there is only one leader election in each round. As a result, for a particular non-godfather block b, the child-block of b's any younger sibling (a sibling block mined after block b) should share the same header and identical source of randomness with one of b's child-block. Thus it is not necessary for a non-godfather block to have sibling blocks, that is, mining a sibling to a non-godfather block does not increase the growth rate of the longest chain in the adversarial block tree. However, sibling god-father blocks have independent sources of randomness and thus mining multiple such block increase the growth rate of the longest chain. $\square$

Here we use the representation in Fig. D.1, whose growth rate is the same as the full NaS adversarial tree as shown in Lemma 12. We can transform the tree $T^{(a)}(t)$ in Fig. D.1 into a new random tree $T^0(t)$. Every c generations in $T^{(a)}(t)$ we can view as a single generation in $T^0(t)$; in the example of Fig. D.1, we have depth 0, 5, 10 etc. (i.e., all the godfather depths) as the generations in $T^0(t)$. $T^0(t)$ corresponds to a branching random walk. For example, the genesis block B_0 is the root of $T^0(t)$ at depth 0 with arrival time

0. The children blocks of B_0 in $T^0(t)$ are the descendant blocks at depth 5 in $T^{(a)}(t)$. We can order these children blocks in their arrival times. Consider block B_1 to be the first such block, then the arrival time of block B_1 is

$$S_1 = X_1 + X_2 + \ldots + X_c,$$

where X_i is the inter-arrival time between block at depth $i-1$ and block at depth i in $T^{(a)}(t)$. Note that all the X_i's are exponential with parameter λ, and they are all independent. Similarly, the arrival time S_i of the i-th child of root in $T^0(t)$ is a sum of $i+c-1$ i.i.d. exponential random variables with parameter λ.

Let the depth of the tree $T^a(t)$ and $T^0(t)$ be denoted by $D^a(t)$ and $D(t)$ respectively defined as the maximum depth of the blocks in the tree.

Similar to Chapter 5, each vertices at generation k can be labelled as a k tuple of positive integers $(i_1, \ldots, i_k)$ with $i_j \geq c$ for $1 \leq j \leq k$: 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 $k-1$. Let $\mathcal{I}_k = \{(i_1, \ldots, i_k) : i_j \geq c \text{ for } 1 \leq \text{j} \leq \text{k}\}$, 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. For $v = (i_1, \ldots, i_k) \in \mathcal{I}_k$, let $\mathcal{W}_v = \sum_{j \le i_k} \mathcal{E}_{(i_1,\ldots,i_{k-1},j)}$ and let $S_v = \sum_{j \le k} \mathcal{W}_{v^j}$. This creates a labelled tree, with the following interpretation: for $v = (i_1, \ldots, i_j)$, the W_{v^j} are the waiting for v^j to appear, measured from the appearance of v^{j-1}, and S_v is the appearance time of v.

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

$$\{D^a(t) \geq ck\} = \{D(t) \geq k\} = \{\underline{S}_k \leq t\}. \tag{D.1}$$

$\underline{S}_k$ is the minimum of a standard BRW. Introduce, for $\theta_c < 0$, the moment generating function

$$\begin{aligned}\Lambda_c(\theta_c) &= \log \sum_{v \in I_1} E(e^{\theta_c S_v}) = \log \sum_{j=c}^{\infty} E(e^{\sum_{i=1}^{j} \theta_c \mathcal{E}_i}) \\ &= \log \sum_{j=c}^{\infty} (E(e^{\theta_c \mathcal{E}_1}))^j = \log \frac{E^c(e^{\theta_c \mathcal{E}_1})}{1 - E(e^{\theta_c \mathcal{E}_1})}.\end{aligned}$$

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

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

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

Indeed, see [20, Theorem 1.3], we have the following.

Proposition 1.

$$\lim_{k \to \infty} \frac{S_k}{k} = \sup_{\theta_c < 0} \left(\frac{\Lambda_c(\theta_c)}{\theta_c} \right) = \frac{1}{\lambda \eta_c}, \quad a.s.$$

In fact, much more is known, see e.g. [29].

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

$$\liminf_{k \to \infty} S_k - k/\lambda\eta_c - c_2 \log k = \infty, a.s.$$

Note that Propositions 1,2 and (D.1) imply in particular that $D^a(t) \leq c\eta_c\lambda t$ for all large t, a.s., and also that

$$\text{if } c\eta_c\lambda > 1 \text{ then } D^a(t) > t \text{ for all large } t, \text{ a.s.} \tag{D.2}$$

Let us define $\phi_c := c\eta_c$, then $\phi_c\lambda$ is the growth rate of private c-correlated NaS tree. One can check that ϕ_c is the solution to the same equation as in [35], where the same problem is solved with a differential equation approach. [35] also proves the uniqueness of ϕ_c and provides an approximation for large c: $\phi_c = 1 + \sqrt{\frac{\ln c}{c}} + o\Big(\sqrt{\frac{\ln c}{c}}\Big)$.

We will need also tail estimates for the event $D(t) > \eta_c \lambda t + x$. While such estimates can be read from [20], we bring instead a quantitative statement suited for our needs.

Theorem 3. For $x > 0$ so that $\eta_c \lambda t + x$ is an integer,

$$P(D^a(t) \geq \phi_c \lambda t + cx) = P(D(t) \geq \eta_c \lambda t + x) \leq e^{\Lambda_c(\theta_c)x}. \tag{D.3}$$

Proof. We use a simple upper bound. Write $m = \eta_c \lambda t + x$. Note that by (D.1),

$$P(D(t) \geq m) = P(S_m \leq t) \leq \sum_{v \in \mathcal{I}_m} P(S_v \leq t). \tag{D.4}$$

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

$$P(S_v \leq t) \leq E e^{\theta_c S_v} e^{-\theta_c t} = \left(\frac{\lambda}{\lambda - \theta_c}\right)^{|v|} e^{-\theta_c t}. \tag{D.5}$$

And

$$\begin{aligned} \sum_{v \in \mathcal{I}_m} \left(\frac{\lambda}{\lambda - \theta_c}\right)^{|v|} &= \sum_{i_1 \geq c, \ldots, i_m \geq c} \left(\frac{\lambda}{\lambda - \theta_c}\right)^{\sum_{j=1}^m i_j} \\ &= \left(\sum_{i \geq c} \left(\frac{\lambda}{\lambda - \theta_c}\right)^i\right)^m = e^{\Lambda_c(\theta_c)m}. \end{aligned} \tag{D.6}$$

Combining (D.5), (D.6) and (D.4) yields (D.3). $\square$

APPENDIX E

NAKAMOTO-POS PROTOCOL PSEUDOCODE

Algorithm 1 on page 69 provides the pseudocode for our proposed Nakamoto-PoS Protocol.

Algorithm 1 Nakamoto-PoS (c, s, δ)

```
procedure Initialize( )
    BlkTree ← genesis                                   ▷ Blocktree
    parentBk ← genesis                                  ▷ Block to mine on
    unCnfTx ← ϕ                                         ▷ Blk content: Pool of unconfirmed txs
procedure PosMining(coin)
    while True do
        SleepUntil(SystemTime % δ == 0)                 ▷ System time is miner's machine time
        time ← SystemTime
        (KES.vk,KES.sk),(VRF.pk,VRF.sk) ← coin.Keys()
        // Update the stake according to the stake distribution in the s-th last block in the main chain.
        stakeBk ← SearchChainUp(parentBk, s)
        stake ← coin.Stake(stakeBk)
        ρ ←UpdateGrowthRate(parentBk)
        // Three sources of randomness: a common source (parentBk.content.RandSource), a private source (coinSecretKey), and time.
        header ← ⟨parentBk.content.RandSource , time⟩
        ⟨hash, proof⟩ ← VRFprove(header, VRF.sk)        ▷ Verifiable Random Function
        if hash < ρ × stake then                        ▷ Block generated
            // Update common source of randomness every c-th block in a chain as per c-correlation scheme
            if parentBk.Height() % c == c − 1 then RandSource ← hash
            else RandSource ← parentBk.content.RandSource
            state ← Hash(parentBk)
            content ← ⟨ unCnfTx, coin, RandSource, hash, proof, state ⟩ and break
    // Return header along with signature on content
    return ⟨header, content, Sign(content, KES.sk)⟩
// Function to listen messages and update the blocktree
procedure ReceiveMessage(X)                             ▷ Receives messages from network
    if X is a valid tx then
        undfTx ← unCnfTx ∪ {X}
    else if IsValidBlock(X) then
        X_fork ← the highest block shared by the main chain and the chain leading to X
        L_fork ← min(parentBk.Height(), X.Height()) - X_fork.Height()
        if L_fork < s then                              ▷ If the fork is less than s blocks
            if parentBk.Height() < X.Height() then
                ChangeMainChain(X)                      ▷ If the new chain is longer
        else                                            ▷ check s-truncated longest chain rule
            MainChainBk ← SearchChainDown(parentBk, X_fork, s)   ▷ find the s-th block down the main chain from fork
            NewChainBk ← SearchChainDown(X, X_fork, s)  ▷ find the s-th block down the new chain from fork
            if NewChainBk.header.time < MainChainBk.header.time then
                ChangeMainChain(X)                      ▷ If the new chain is denser
procedure IsValidBlock(X)                               ▷ returns true if a block is valid
    if not IsUnspent(X.content.coin) then return False
    if X.header.time > SystemTime then return False
    if VRFverify(X.header, X.content.hash,X.content.proof,X.content.coin.VRF.pk) then
        return True
    else
        return False
procedure Main( )
    Initialize()
    StartThread(ReceiveMessage)
    while True do
        block = PosMining(coin)
        SendMessage(block)                              ▷ Broadcast to the whole network
```

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[3] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2017.

[4] J. Chen and S. Micali, "Algorand," *arXiv preprint arXiv:1607.01341*, 2016.

[5] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[6] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: BFT consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[7] M. Baudet, A. Ching, A. Chursin, G. Danezis, F. Garillot, Z. Li, D. Malkhi, O. Naor, D. Perelman, and A. Sonnino, "State machine replication in the Libra blockchain," 2018.

[8] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 23–41.

[9] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[10] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 66–98.

[11] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 913–930.

[12] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," in *Proceedings of the 2019 ACM Conference on Economics and Computation*. ACM, 2019, pp. 459–473.

[13] L. Fan and H.-S. Zhou, "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.

[14] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[15] C. Natoli and V. Gramoli, "The balance attack against proof-of-work blockchains: The R3 testbed as an example," *arXiv preprint arXiv:1612.09426*, 2016.

[16] L. Kiffer, R. Rajaraman et al., "A better method to analyze blockchain consistency," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 729–744.

[17] A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Everything is a race and Nakamoto always wins," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 859–878.

[18] L. Fan, J. Katz, and H.-S. Zhou, "A large-scale proof-of-stake blockchain in the open setting," 2019, available online at http://www.fractalblock.com/assets/iching_consensus_protocol.pdf.

[19] V. Bagaria, A. Dembo, S. Kannan, S. Oh, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Proof-of-stake longest chain protocols: Security vs predictability," *arXiv preprint arXiv:1910.02218*, 2019.

[20] Z. Shi, *Branching Random Walks*, ser. Lecture Notes in Mathematics. New York NY: Springer Verlag, 2015, vol. 2151.

[21] R. Stütz, P. Gaži, B. Haslhofer, and J. Illum, "Stake shift in major cryptocurrencies: An empirical study," *arXiv preprint arXiv:2001.04187*, 2020.

[22] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[23] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *International Workshop on Public Key Cryptography*. Springer, 2005, pp. 416–431.

[24] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Annual International Cryptology Conference*. Springer, 1999, pp. 431–448.

[25] G. Itkis and L. Reyzin, "Forward-secure signatures with optimal signing and verifying," in *Annual International Cryptology Conference*. Springer, 2001, pp. 332–354.

[26] B. Pittel, "Notes on the heights of random recursive trees and random m-ary search trees," *Random Structures Alg.*, vol. 5, pp. 337–347, 1994.

[27] M. Drmota, "The height of increasing trees," *Annals of Combinatorics*, vol. 12, no. 4, pp. 373–402, 2009.

[28] E. Aïdékon, "Convergence in law of the minimum of a branching random walk," *The Annals of Probability*, vol. 41, no. 3A, pp. 1362–1426, 2013.

[29] Y. Hu and Z. Shi, "Minimal position and critical Martingale convergence in branching random walks, and directed polymers on disordered trees," *The Annals of Probability*, vol. 37, no. 2, pp. 742–789, 2009.

[30] L. Ren, "Analysis of Nakamoto consensus," Cryptology ePrint Archive, Report 2019/943.(2019). https://eprint. iacr. org . . . , Tech. Rep., 2019.

[31] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 585–602.

[32] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[33] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2017.

[34] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[35] X. Wang, G. Kamath, V. Bagaria, S. Kannan, S. Oh, D. Tse, and P. Viswanath, "Proof-of-stake longest chain protocols revisited," *arXiv preprint arXiv:1910.02218*, 2019.