FairPoS: Input Fairness in Permissionless Consensus

James Hsin-yu Chiang
Technical University of Denmark, Denmark

Bernardo David
IT University of Copenhagen, Denmark

Ittay Eyal
Technion, IC3, Haifa, Israel

Tiantian Gong
Purdue University, West Lafayette, USA

Abstract

In permissionless consensus, the ordering of transactions or inputs in each block is freely determined by an anonymously elected block leader. A rational block leader will choose an ordering of inputs that maximizes financial gain; the emergence of automatic market makers in decentralized finance enables the block leader to front-run honest trade orders by injecting its own inputs prior to and after honest trades. Front-running is rampant in decentralized finance and reduces the utility of the system by extracting financial value from honest trades and increasing demand for block-space. Current proposals to prevent input order attacks by encrypting user inputs are not permissionless, as they rely on small static committees to perform distributed key generation and threshold decryption. Such committees require party authentication, knowledge of the number of participating parties or do not permit player replaceability and are therefore not permissionless. Moreover, alternative solutions based on sequencing inputs in order of their arrival cannot prevent front-running in an unauthenticated peer-2-peer network where message arrival is adversarially controlled.

We present *FairPoS*, the first consensus protocol to achieve input fairness in the permissionless setting with security against adaptive adversaries in semi-synchronous networks. In FairPoS, the adversary cannot learn the plaintext of any client input *before* it is included in a block in the chain's common-prefix. Thus, input ordering attacks that depend on observing pending client inputs in the clear are no longer possible. In FairPoS, this is achieved via Delay Encryption (DeFeo *et al.*, EUROCRYPT 2021), a recent cryptographic primitive related to time-lock puzzles, allowing *all* client inputs in a given round to be encrypted under a key that can only be extracted after enough time has elapsed. In contrast to alternative approaches, the key extraction task in delay encryption can, in principle, be performed by any party in the permissionless setting and requires no distribution of secret key material amongst authenticated parties. However, key extraction requires highly specialized hardware in practice. Thus, FairPoS requires resource-rich staking parties to insert extracted keys into blocks, enabling light-clients to decrypt past inputs and relieving parties who join the execution from decrypting all inputs in the entire chain history. Realizing this in proof-of-stake is non-trivial; naive application of key extraction to proof-of-stake can result in chain stalls lasting the entire key extraction period. We overcome this challenge with a novel *key extraction protocol*, which tolerates adversarial delays in block delivery intended to prevent key extraction from completing on schedule. Critically, this also enables the adoption of a new *longest-extendable-chain* rule which allows FairPoS to achieve the same guarantees as Ouroborous Praos against an adaptive adversary.

2012 ACM Subject Classification Security and privacy → Privacy-preserving protocols

Keywords and phrases Front-running, Delay Encryption, Proof-of-Stake, Blockchain

Digital Object Identifier 10.4230/LIPIcs...

© Anonymous;
licensed under Creative Commons License CC-BY 4.0

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In permissionless consensus, the ordering privilege of the block leader is exploited in front-running [17], where adversarial inputs can be interleaved with honest inputs to extract financial value from the honest victim in applications such as automatic market makers [5]. Such behaviour financially penalizes the honest user, but also generates excess demand for block-space since front-running attacks [5] always require additional inputs from the adversary, inflicting block congestion at times, as acutely observed on Avalanche [2]. Current proposals to mitigate front-running with varying notions of input fairness violate assumptions underlying permissionless consensus such as Proof-of-Stake (PoS) [18].

A commonly proposed notion of input fairness requires encrypting inputs which are then decrypted after finalization[1]. To guarantee timely decryption and to avoid malicious parties withholding the reveal of the plaintext input, *threshold decryption* [31, 7] or *identity-based encryption* [32, 20] involving static committees have been proposed. The honest-majority committees with distributed private or master key material then guarantee the decryption of inputs as specified by the protocol. However, such protocols require authenticated parties to prevent Sybil interference and assume secure, private communication and are therefore not permissionless; ongoing research efforts to lift such protocols into the permissionless setting are discussed in Section 2.

Alternatively, the notion of sequencing transactions in the order of their arrival at honest consensus nodes has been proposed in [28, 27, 26, 13]. This is meaningful in the permissioned setting where communication between client and consensus node is fast, preventing an adversary to observe a pending transaction and then emit a front-running transaction which can then propagate faster than the victim's transaction. However, this notion of input order fairness does not easily translate to the permissionless setting, where transactions are propagated across a permissionless, unauthenticated peer-to-peer network where delay is adversarially controlled.

We introduce FairPoS, a PoS blockchain consensus protocol that achieves a novel notion of *input fairness* (Def. 6, Thm. 19) in permissionless consensus, while retaining the security guarantees of Ouroborous Praos [18]. As in Praos, we prove security against an adaptive adversary, which controls the network delay and corrupts parties as the protocol execution unfolds. Our novel notion of input fairness in permissionless consensus guarantees that the plain-text content of any finalized input (in the common-prefix) could not have been observed by the adversary prior to its finalization. FairPoS achieves this by encrypting inputs with the *delay encryption* scheme by DeFeo et al. [12], which improves on time-lock puzzles [33]. Classical time-lock puzzles store plaintext messages that can be obtained by clients after an *extraction process* that requires performing a known number of non-parallelizable sequential operations (*i.e.* requiring a certain minimum amount of time for recovering a message). However, naively encrypting inputs with time-lock puzzles requires a dedicated extraction process for each client input, which quickly becomes infeasible at higher throughput. Delay encryption, in contrast, allows all inputs in a block to be encrypted under a single unknown key, which can be extracted as time elapses. Hence, only a single key extraction is required for each block. The *extraction* procedure to recover the decryption key is parameterized to run in at least time d, and can be performed by any party with access to specialized hardware to ensure timely execution. This preserves adaptive security, as no relevant key

[1] In permissionless, longest-chain consensus, input finalization occurs when the block containing the input joins the common-prefix.

material is learned upon corruption of an honest party.

Still, it is not practical for non-staking parties or clients with limited resources to perform key extraction. First, we expect only resource-rich participants to have access to the specialized hardware [1] necessary to perform extractions in d time; otherwise, a long-running, non-trivial extraction cost would be imposed on clients following the blockchain and interacting with smart contract applications. Secondly, without any integrated mechanism to publicly expose extraction keys, any party joining the protocol would need to perform key extractions for all blocks beginning from genesis, which becomes rapidly more expensive at higher chain lengths. A key contribution of FairPoS is a novel *key extraction protocol*, requiring staking parties to insert the extracted keys from past blocks into later, child blocks of the same chain within a fixed schedule, thus ensuring decryption keys are made public in lock-step with chain growth. The challenge here is to prevent the arbitrary delay of adversarial blocks to impede chain growth if honest parties cannot finish key extractions on time due to delayed arrival of past blocks. In standard blockchain consensus, chains in the local view can immediately be extended, but parties in FairPoS can only extend a chain if past key extractions are completed *on time*. Note that it is not sufficient to require a block to arrive at an honest party within the maximum network delay, so that key extraction can begin as intended. The adversary can trivially deliver dishonest blocks to a subset of honest parties only with the maximum permitted *receipt delay*, and then induce an additional network delay as this dishonest block is relayed to others. The local receipt delay of this block at other parties must then exceed the limit, causing irreconcilable inconsistencies and potential chain stall. Although such attacks cannot be prevented as the adversary is permitted to delay messages up to a maximum bound, they are carefully addressed in FairPoS by incrementing receipt delay bounds for blocks which are further away from the chain tip (Fig. 3). The novel *longest-extendable-chain* rule then asserts this notion of timeliness of block arrivals, guaranteeing that any honest chain can be extended by another honest leader within a fixed time. We highlight honest chain growth as a critical property and formally prove that FairPoS achieves both input fairness whilst maintaining the security of Praos [18].

Paper overview. We provide an overview of related work in Section 2. In Section 3, we introduce Delay Encryption and an abstract model of Ouroborous Praos execution (δ-PoS). In Section 4, we then define our proposed notion of Input Fairness for permissionless consensus and present the FairPoS model, extending δ-PoS with delay encryption and a novel "longest-extendable-chain" selection rule. In Section 5, we gently introduce FairPoS and formally demonstrate that achieves input fairness whilst maintaining the asymptotic security of Ouroborous Praos (δ-PoS) against an adaptive adversary. Full Proofs of stated theorems and lemmas are provided in Appendix C.

2 Related Work

Encrypt-and-reveal with timed cryptography. The basic idea of encrypt-and-reveal is to encrypt client inputs, and then to decrypt these when they are finalized in the blockchain. The "blockchain state" implied by the ordering of inputs is then "revealed" as past inputs are decrypted over time. For "decryption" to be permissionless, it must be possible without the knowledge of any secret key material; here, time-lock puzzles [33] were first proposed, which permit any party with the ciphertext to compute the decryption key with d squarings in a group of unknown order. Similarly, timed commitments were proposed in [11], which are accompanied by zero-knowledge proof of well-formedness; namely, that the time-lock commitment is well-formed and opens after d squaring operations. Time-lock puzzles have

been formalized in the universal composability framework in Tardis [6], permitting secure composition with other protocols.

Mitigating front-running with time-lock puzzles in exchanges has been proposed in Clockwork [16]. In [25], blockchain inputs are encrypted with time-lock puzzles to prevent adversarial ordering based on the input plaintext. We note that this approach requires extracting decryption keys for each submitted input individually, which quickly becomes impractical for higher throughput levels. Open square [34] proposes a service which permits users to outsource the extraction of cryptographic time-locks to anonymous servers. De Feo et al. introduce the first time-lock primitive which permits all clients to encypt to the same session key, called Delay Encryption (§3.1), which improves on time-lock puzzles and minimizes "wasted" work spent on solving individual time-locks separately; thus, delay encryption represents the state-of-the-art in time-lock encryption.

In all timed-crypto primitives, the existence of a "fastest" hardware is assumed. Furthermore, it must assumed that this is accessible to all participants. We highlight two open challenges; firstly, there has been little research on parameterising time-cryptography for real-world hardware designs. Secondly, the "fastest" hardware design is likely to be specialized and costly such that in practice, only resource-rich participants are likely to have access to such extraction hardware. A key objective of FairPoS is to address the second challenge by requiring block leaders to include extracted keys in the blockchain, thereby minimizing redundant work and allowing non-staking users to decrypt past inputs in lockstep with chain growth.

Encrypt-and-reveal with threshold cryptography. An alternative to timed-cryptography is to rely on a dedicated committee to generate a distributed master key, permitting the encryption and subsequent decryption of inputs, after they are finalized in the longest chain. Threshold decryption [7, 31] imposes a significant overhead, as each encrypted input must be individually decrypted by the dedicated committee. We note a recent you-only-speak-once (YOSO) line of work that investigates cryptographic protocols via anonymously elected committees [9, 23, 15, 14, 21] promising player replaceability in the absence of party authentication. Protocols in the YOSO model allow for permissionless solutions with the same effect of the aforementioned solutions based on threshold encryption. The concept of Encryption to the Future [14] allows for encrypting messages in such a way that they can be decrypted only by the block leader of a later slot, also allowing for realizing "Witness Encryption on Blockchains" (WEB) as proposed in [23] using threshold Identity Based Encryption (IBE). However, protocols in the YOSO model require techniques for proactively secret-sharing state to future committees; accomplishing this with high throughput in a practical manner remains an open problem. The central technique in the WEB construction from [14] is efficiently implemented in [32], which allows clients to encrypt their inputs to a *future round number* in string form; the identity-specific key associated with the current round number is then jointly released by the committee, permitting the public decryption of client inputs encrypted to the same round. An alternative construction called signature-based witness encryption is proposed by McFly [20], which realizes a special case of WEB with an efficient instantiation of the threshold IBE technique from [14]. This approach permits the encryption of an input to the set of committee verification keys and a round specific reference string. McFly scales to larger committee sizes, promising practicality for permissioned consensus systems. We emphasize that such encrypt-and-reveal approaches are not permissionless; an authenticated party must be assigned to each protocol role in an execution, communication between committee members must be secure and private and parties are not arbitrarily replaceable.

Fair ordering. A line of research from authenticated consensus [28, 27, 26, 13] proposes a notion of *fair input ordering*. A block leader will order inputs based on their order of arrival. However, fair ordering is only meaningful in a setting with a secure connection between client and round leaders: in an unauthenticated, peer-to-peer gossip network setting common in massively distributed permissionless blockchain protocols, the receipt-order of messages is adversarially controlled, making it difficult to justify any notion of fair message arrival. A secure connection with the next block leader implies public knowledge of its identity, contradicting the permissionless setting. The work of [26] lifts the notion of receipt-order-fairness to the permissionless setting; however, we argue the adopted notion of transaction arrival ordering is difficult to justify when client messages are propagated across an unauthenticated, peer-2-peer network with adversarially controlled delivery schedules; the adversary observing a propagating client transaction can inject their own inputs and control the receipt order for each honest consensus node. Moreover, this work is only secure against static adversaries.

3 Preliminaries

3.1 Delay Encryption

The delay encryption (DE) scheme by De Feo *et al.* [12] consists of the following four algorithms: A global DE.Setup parameterized with a security parameter $\lambda \in \{0,1\}^*$ and delay parameter d generates public encryption (DE.pk) and extraction (DE.ek) keys. In each round, a public session $\mathsf{id} \in \{0,1\}^*$ is sampled, and DE.Encaps can be used to generate a pair (c, k) of a ciphertext c and a key k corresponding to id and the encryption key DE.pk. The DE.Extract algorithm runs in at least d time, and returns a session key idk, with which the DE.Decaps algorithm can compute a key k from ciphertext c for all (c, k) generated with the same session id and public paramaters from a given setup.

1. $\mathsf{DE.Setup}(\lambda, d) \to (\mathsf{DE.ek}, \mathsf{DE.pk})$
2. $\mathsf{DE.Encaps}(\mathsf{DE.pk}, \mathsf{id}) \to (c, k)$
3. $\mathsf{DE.Extract}(\mathsf{DE.ek}, \mathsf{id}) \to \mathsf{idk}$
4. $\mathsf{DE.Decaps}(\mathsf{DE.pk}, \mathsf{id}, \mathsf{idk}, c) \to k$

Delay encryption is an isogeny-based delay protocol, and similar to [19] is built from isogeny walks in graphs of pairing friendly supersingular elliptic curves. In implementations [19], such isogeny evaluations occupy memory space in the terabytes. Parties performing *Extract* are expected to deploy specialized FPGA hardware [1] in order to achieved the parameterized extraction time.

3.2 Longest-chain PoS model and security

We present a model of *longest-chain proof-of-stake* protocols, formalized by the Ouroborous line of work [30, 18, 3] and subsequent improvements [10, 29]. We adopt modelling approach in [30, 18, 3, 29], where the PoS protocol is modelled by two orthogonal components: the first describes the *leader election process* and the second part models the views of blockchain trees which result from a protocol execution *induced by a given leader schedule*.

Idealized leader elections. Time in PoS is divided into units named slots, each capturing the duration of a single protocol round. In a given round, a party with relative stake $\alpha \in (0, 1]$ becomes a slot leader for a given slot with probability

$$\phi(\alpha) = 1 - (1 - f)^{\alpha}$$

where parameter *active slot coefficient* f is the probability that a leader holding all stake will be elected leader in given slot: importantly, $\phi(\alpha)$ is maintained even if share α is split amongst multiple, virtual parties (eq. 2 in [18]). Let a characteristic string w be defined as a sequence of leader election results, where an election result at slot t is defined as follows.

$$w_t = \begin{cases} 0 & \text{a single honest leader} \\ 1 & \text{multiple honest leaders / adversarial leader} \\ \perp & \text{no leader} \end{cases}$$

In PoS [18], leader election is can be modelled by sampling characteristic strings from an idealized, *dominant distribution* $\mathcal{D}^f_\alpha$ that is strictly *more adversarial* than the *true* setting where the *adaptive adversary* corrupts up to $(1-\alpha)$ of the stake during the protocol execution (Theorem 8 in [18]). Thus, any security that holds in PoS executions induced by characteristic strings sampled from $\mathcal{D}^f_\alpha$ must also hold in the true protocol execution against an adaptive adversary dynamically corrupting up to $1-\alpha$ stake.

▶ **Definition 1** (Dominant distribution $\mathcal{D}^f_\alpha$ (Definition 11 in [18])). *For an adaptive adversary corrupting up to $1-\alpha$ stake fraction and active slot coefficient $f \in [0,1)$, the dominant distribution $\mathcal{D}^f_\alpha$ is defined by the following probabilities:*

$$p_\perp = 1 - f \qquad p_0 = \phi(\alpha) \cdot (1-f) \qquad p_1 = 1 - p_\perp - p_0 \tag{1}$$

▶ **Definition 2** (**Blocks, chains, trees and branches**). *A block $B = (\mathsf{sl}, \mathsf{st}, \mathsf{d}, \mathsf{ldr})$ generated at slot sl contains state $\mathsf{st} \in \{0,1\}^*$, data $\mathsf{d} \in \{0,1\}^*$ and party ldr that generated B and was the elected block leader of slot sl. A chain is a sequence of blocks $B_0, ..., B_n$ associated with a strictly increasing sequence of slots, where B_0 is the genesis block, and the state of B_i is $H(B_{i-1})$, $H(\cdot)$ denoting a collision-resistant hash function. We write $\mathcal{C}.\mathsf{tip}$ to denote the block at the tip of chain $\mathcal{C}$ and $\mathcal{C}_j$ to denote a block $B \in \mathcal{C}$ such that $B.\mathsf{sl} = j$. If such a block does not exist, $\mathcal{C}_j = \perp$. Let $\mathcal{C}^{\lceil k}$ denote the chain obtained from $\mathcal{C}$ by removing the last k blocks. Multiple chains form a tree if their blocks share state. A branch $\mathcal{B}$ in a tree $\mathcal{T}$ is a chain which ends with a leaf block. We write $\mathcal{C} \preceq \mathcal{B}$ to indicate $\mathcal{C}$ is a prefix of $\mathcal{B}$. When quantifying over all chains in a tree, $\forall \mathcal{C} \in \mathcal{T}$, we quantify over all prefixes of all tree branches. Let $\mathsf{len}(\mathcal{C})$ denote the number of blocks in chain $\mathcal{C}$.*

A model of δ-PoS executions. As in [22, 30, 18, 3], we model the execution of PoS initiated upon the activation of an environment $\mathcal{Z}$, which spawns both honest parties $\mathcal{H}$ and an adversary $\mathcal{A}$. Upon each activation by the environment, each party executes the protocol according to Figure 1, which precisely models the adversarial powers to influence the round-wise evolution of block tree structures in the local view parties as the full PoS protocol in [18], but omits details such as block proofs, signatures or individual leader election procedures such as evaluation of verifiable random functions. A given characteristic string w *induces* executions of our δ-PoS model that generate local tree structures identical to those resulting from a full PoS [18] protocol execution that *induces* a leader election sequence consistent with w and activates the same parties and adversarial actions.

Let the protocol execution state Γ_t in slot t, consist of honest party states, including the local block tree view $\mathcal{T}^{(i)}$ and the outbound message queue $\mathbf{m}^{(i)}$ for each honest party $i \in \mathcal{H}$. Further, let Γ_t include the blockchain tree view $\mathcal{T}^{\mathcal{A}}$ of the adversary.

$$\Gamma_t = \left(\{\mathcal{T}^{(i)}, \mathbf{m}^{(i)}\}_{i \in \mathcal{H}}, \mathcal{T}^{\mathcal{A}}\right) \tag{2}$$

The outbound message queue $\mathbf{m}^{(i)} = \{(\mathcal{C}, \mathcal{P}), ...\}$ in Γ_t is the set of broadcast, yet undelivered chains previously sent by honest party i. For each entry $(\mathcal{C}, \mathcal{P}) \in \mathbf{m}^{(i)}$, $\mathcal{C}$ was initially

δ-PoS consensus model

The δ-PoS state $\Gamma_t = \left(\{\mathcal{T}^{(i)}, \mathbf{m}^{(i)}\}_{i\in\mathcal{H}}, \mathcal{T}^{\mathcal{A}}\right)$ evolves in a single round $\Gamma_t \to^{w_{t+1}} \Gamma_{t+1}$ induced by environment $\mathcal{Z}$, adversary $\mathcal{A}$ and characteristic string w as follows.

Trees. Honest $\{\mathcal{T}^{(i)}\}_{i\in\mathcal{H}}$ and adv. $\mathcal{T}^{\mathcal{A}}$ evolve as follows:

T0 $\mathcal{T}_0^{(i)}$ consists of the genesis block.

T1 If $w_{t+1} = 0$, a single honest party runs **Extend** on the longest $\mathcal{C}$ in $\mathcal{T}^{(i)}$, which is then added to $\mathcal{T}^{(i)}$ and $\mathcal{T}^{\mathcal{A}}$.

T2 At any time during the round, adversary $\mathcal{A}$ may:

a) Run **Extend** on a chain in $\mathcal{T}^{\mathcal{A}}$ for slot $t+1$ if $w_{t+1} = 1$, upon which the extended chain is added to $\mathcal{T}^{\mathcal{A}}$.

b) Update any honest tree view $\mathcal{T}^{(i)}$ with an additional chain from $\mathcal{T}^{\mathcal{A}}$ or an honest message queue $\{\mathbf{m}^{(i)}\}_{i\in\mathcal{H}}$ (see **M2**).

Msgs. Honest message queues $\{\mathbf{m}^{(i)}\}_{i\in\mathcal{H}}$ evolve as follows:

M1 For chain $\mathcal{C}'$ extended by honest party i in the round (**T2**), the entry $(\mathcal{C}', \mathcal{H})$ is added to local message queue $\mathbf{m}^{(i)}$.

M2 At any time during the round, adversary $\mathcal{A}$ may deliver a chain from an entry $(\mathcal{C}, \mathcal{P}) \in \mathbf{m}^{(i)}$ to a subset of honest users $\mathcal{I} \subseteq \mathcal{H}$:

a) Entry $(\mathcal{C}, \mathcal{P})$ in $\mathbf{m}^{(i)}$ is updated to $(\mathcal{C}, \mathcal{P}')$, where $\mathcal{P}' = \mathcal{P} \setminus \mathcal{I}$.

b) Each $(\mathcal{C}, \mathcal{P}) \in \mathbf{m}^{(i)}$ must be delivered to all honest parties $\mathcal{H}$ by slot $\mathcal{C}.\mathsf{tip}.\mathsf{sl} + \delta$, and is then removed from $\mathbf{m}^{(i)}$.

Extend. To extend $\mathcal{C}$, party i generates $B = (t+1, H(\mathcal{C}.\mathsf{tip}), d, i)$ with an *ordering* of inputs $d = \{\mathsf{in}_i\}_{i\in[m]}$ input by environment $\mathcal{Z}$ for slot t.

Figure 1 Model of δ-PoS execution induced by environment $\mathcal{Z}$ and characteristic string w.

broadcast and added to the local message queue at slot $\mathcal{C}.\mathsf{sl} \leq t$. Each entry in $\mathbf{m}^{(i)}$ consists of a chain $\mathcal{C}$ and honest party subset $\mathcal{P} \subset \mathcal{H}$, which has yet to receive the message. $\mathcal{A}$ is required to deliver all honestly broadcast chains with a delay of no more than δ slots. The model executes round-wise beginning from initial state Γ_0, where the tree views of all parties consist only of the genesis block.

In each round from slot t to $t+1$, the leader is implied by by $w_{t+1} \in \{0, 1, \perp\}$. For a uniquely honest slot, the environment $\mathcal{Z}$ is permitted to activate any honest party to extend the longest chain in its local view, where the inputs for insertion in the block are provided by $\mathcal{Z}$. We interpret $w_{\mathsf{slot}} = 1$ as a strictly adversarial slot, since the adversary could affect the structure of local trees views in the same way as multiple honest leaders: namely, by producing multiple blocks associated with the same slot.

PoS Security. The seminal work on formalizing the Bitcoin backbone protocol [22] proved *liveness* and *persistence* of longest-chain proof-of-work (PoW) protocols in terms of common-prefix, chain growth and chain quality properties, which are also achieved for PoS in Ouroboros Praos [18]. We restate these below and formally prove them for FairPoS in Section 5.

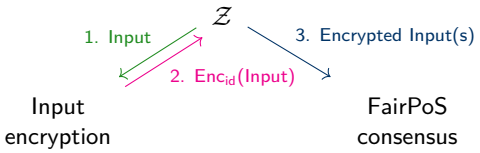
▶ **Definition 3 (Common prefix, k-CP; with parameter $k \in \mathbb{N}$).** *The chains $\mathcal{C}_1$, $\mathcal{C}_2$ possessed by two honest parties at the onset of the slots $t_1 < t_2$ are such that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$, where $\mathcal{C}_1^{\lceil k}$ denotes the chain $\mathcal{C}_1^{\lceil k}$ obtained by removing the last k blocks from $\mathcal{C}_1$, and $\preceq$ denotes the prefix relation.*

▶ **Definition 4 (Chain growth, (τ, s)-CG; with parameter $\tau \in (0, 1]$ and $s \in \mathbb{N}$).** *Consider the chains $\mathcal{C}_1$, $\mathcal{C}_2$ possessed by two honest parties at the onset of two slots t_1, t_2 with t_2 at least s slots ahead of t_1. Then it holds that $\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq \tau \cdot s$. We call τ the speed coefficient.*

▶ **Definition 5 (Chain quality, (μ, k)-CQ; with parameters $\mu \in (0, 1]$ and $k \in \mathbb{N}$).** *Consider any portion of length at least k of the chain possessed by an honest party at the onset of a round; the ratio of blocks originating from the adversary is at most $1 - \mu$. We call μ the chain quality coefficient.*

4 The FairPoS protocol

As in δ-PoS, we model a FairPoS execution that is induced by an environment $\mathcal{Z}$ and a characteristic string w; both protocol participants and adaptive adversary $\mathcal{A}$ are spawned by $\mathcal{Z}$, whereas w governs which parties may be activated by $\mathcal{Z}$ to generate blocks at each slot. The adaptive adversary is permitted to spend its corruption budget on honest parties anytime; in particular, it can observe any message emitted by an honest party, and then decide its corruption strategy based on the sent message of the honest user. In FairPoS, inputs provided to the block leader by $\mathcal{Z}$ are delay encrypted. Thus, a party must first be activated by $\mathcal{Z}$ with a plain-text input and execute the input encryption procedure (Fig. 2). Upon receiving encrypted inputs, $\mathcal{Z}$ can forward an ordering of encrypted client inputs to the elected block leader in the FairPoS execution (Fig. 4).



We introduce the FairPoS model in parts. In Section 4.1, we formally define the notion of input fairness in the permissionless setting for clients sending transactions to a FairPoS execution. In order for an encrypted input to reach the k-common-prefix, the duration implied by the delay parameter d must be sufficiently long. We then define the FairPoS input encryption procedure; our protocol deploys key evolving signature schemes as in Ouroborous Praos [18], which prevents the adaptive adversary from obtaining static key material upon corrupting an honest party that has just emitted an honestly signed input.

In Sections 4.2 and 4.3, we introduce the formal FairPoS protocol execution model, which extends δ-PoS with key extraction and a novel longest-extendable-chain selection rule. Here, encrypted inputs are generated as in Section 4.1 and given by the environment $\mathcal{Z}$ to parties executing the protocol. We first describe how the adversary can prevent key extraction processes to complete on time by delaying adversarial blocks, thereby motivating the design of the longest-extendable-chain selection rule in the FairPoS model, which mitigates such adversarial impedance and ensures honest chain growth independent of chosen delay encryption parameter d.

A security analysis of FairPoS follows in Section 5, where the precise relationship between input fairness, chain growth, common-prefix and chain quality properties is formalized.

FairPoS input encryption

Let KES = (Gen,Sign,Verify,Update) denote a key evolving signature scheme and SKE = (Gen, Enc,Dec) a symmetric-key encryption scheme. Let the genesis block of a chain contain a delay encryption parameter DE.pk and a chain tip imply account keys $\{\mathsf{KES.vk}_i\}_{i\in[n]}$.

Gen. Upon (GEN), set $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^k, T)$, return vk.

Sign. Upon $(\textsc{Sign}, \mathcal{C}, \mathsf{in})$

1. Let $\mathsf{id} = \mathcal{C}.\mathsf{tip}$, assert $\mathsf{vk} \in \mathsf{accts}(\mathcal{C}.\mathsf{tip})$
2. Set $\mathsf{pk} \leftarrow \mathsf{DE.pk}(\mathcal{C}_0)$
3. Compute $(c, k) \leftarrow \mathsf{DE.Encaps}(\mathsf{pk}, \mathsf{id})$.
4. Encrypt input with key k: $m \leftarrow \mathsf{SKE.Enc}_k(\mathsf{in})$.
5. Generate $\sigma \leftarrow \mathsf{KES.Sign}_{\mathsf{sk}}(c \,|\, m \,|\, \mathsf{id} = \mathcal{C}.\mathsf{tip})$.
6. Set $\mathsf{sk} \leftarrow \mathsf{Update}(\mathsf{KES.sk})$, erasing previous signing key.
7. Return (c, m, σ).

Figure 2 Input encryption procedure in FairPoS

4.1 Input fairness & encryption

▶ **Definition 6 (Input fairness, IF).** *Consider the chain $\mathcal{C}$ possessed by an honest party at the onset of a round, where k-CP holds true. Input fairness holds if for all blocks $B \in \mathcal{C}^{\lceil k}$: 1. the adversary cannot decrypt an honestly encrypted input in B before B is in the k-common-prefix; 2. encrypted inputs in B are eventually decrypted by all honest parties.*

Input fairness is conditioned on k-common-prefix property in FairPoS. Intuitively, the extraction delay d in FairPoS must be parameterized, such that the encrypted input can reach the common-prefix before d time passes. For simplicity, we denote d as time in slots. Note that input fairness permits an encrypted input to *not* become finalized and decrypted by the adversary: we argue this outcome is acceptable as the client transaction is not executed and thus cannot be exploited in any input ordering attacks. This is consistent with [7, 31, 32].

We sketch the input encryption procedure for FairPoS shown in Figure 2, where the environment $\mathcal{Z}$ provides the plain-text input for a party to encrypt and sign. For a block B, inputs are encrypted to a session id which is set to the chain tip that B is extending, such that $\mathsf{id} = \mathcal{C}.\mathsf{tip}$ and $\mathcal{C}.\mathsf{tip} = B.\mathsf{st}$. To ensure that a slot leader cannot insert an encrypted input to a *later* block, potentially deferring its insertion until the key extraction is completed, we ensure that the input is *bound* to a child block of $\mathcal{C}.\mathsf{tip}$ with a signature.

In the adaptive corruption setting, we deploy an efficient key evolving signature scheme (KES) [8, 24] as in Ouroborous Praos [18]. Such schemes evolve secret key material *forward* with each signature, thereby erasing any information that could be used to generate verifying signatures of past rounds (See Definition 20). An adaptive adversary can always corrupt an honest user who has just *broadcast* a newly delay encrypted and signed input; with static key material only, the adversary would learn the signature key and generate verifying signatures of the delay encrypted input to insert it into any later block, potentially *after* decrypting the encrypted input[2]. In other words, since it is the signature of an honest user that binds

[2] In practice, clients may be required to safeguard static key material. For modelling consistency, we assume clients performing the input encryption procedure can be adaptively corrupted just as parties participating in FairPoS consensus, motivating the use of key evolving signature schemes.

its encrypted input to a specific block, allowing the adaptive adversary to bind the same honestly encrypted input to a later block will violate input fairness in Definition 6.

For the public verification of such signatures, we assume the presence of logical accounts for all parties, each associated with a public, signature verification key inferred from the chain tip.

Malformed inputs. As shown in Fig. 2, it will be possible for adversarial parties to encrypt malformed inputs; elected, honest block leaders which included these encrypted inputs to their blocks cannot discern the validity of the plaintext message, as the key extraction procedure is intended to proceed until the input reaches the k-common-prefix.

We omit a detailed treatment of mitigation techniques, but sketch several possible approaches. (1) A zero-knowledge proof of well-formedness may accompany each encrypted input, proving well-formedness of the underlying input. (2) Fees may mitigate denial-of-service attacks from mal-formed, adversarial inputs; we note that the correct execution- or gas-dependent fee amount can also be proven in zero-knowledge for certain applications.

We also highlight the existence of consensus protocols with explicit focus on input finalization for high transaction-throughput [4]. Here, finalized yet invalid inputs are simply ignored when determining canonical, total ordering of the chain.

4.2 Introducing key extraction in FairPoS

In FairPoS, each block is associated with a session id string, to which inputs in the child block are encrypted, as required by the input encryption procedure in Fig. 2. The extraction of the session key idk required to decrypt the finalized, input ciphers is parameterized to take d slots or rounds. A key design goal of FairPoS is to ensure that the session keys for each block will be included in *later blocks* of the same chain so that lightweight clients following the protocol execution do not need to perform expensive extractions to observe the blockchain state. This suggests that the protocol must define a fixed "extraction schedule" parameter $D \geq d$, such that the session key of a block B must be included the earliest block following slot $B.\mathsf{sl} + D$ of a chain extending B. Defining such a fixed extraction schedule, however, is not trivial, as the adversary can delay its own blocks arbitrarily, and reveal them to a selected subset of honest users only, and thereby induce additional block delays for other honest parties as the dishonest block must be relayed over the network, further hindering timely completion of key extraction processes run by honest parties.

In this section, we provide preliminary definitions and intuition for how FairPoS achieves scheduled key extraction without breaking honest chain growth; the latter is formalized as the Δ-monotonicity property in Definition 7 which is also holds in Praos [18]. Then, we define *receipt delays* in Definition 8; this is the delay between the onset of a round and the local arrival time of a block associated with the same round. Given receipt delays, we then show a naive application of maximum receipt delays limits for blocks. The idea here is to ensure key extraction processes are initiated in a timely manner by requiring blocks to arrive within a maximum duration after their scheduled generation. We then show how the adversary can attack such a naive scheme and cause chain stalls (eq. 3). This attack is overcome in FairPoS by carefully incrementing maximum receipt delays for blocks further away from the chain-tip, as illustrated in Example 9. We demonstrate that FairPoS achieves Δ-monotonicity in Theorem 10, a key property required for FairPoS security (§5). A more formal treatment of the the full FairPoS consensus protocol follows in Section 4.3.

▶ **Definition 7** (Δ-**Monotonicity**, from [18]). *Let $\mathcal{T} = \bigcup_{i \in \mathcal{H}} \mathcal{T}^{(i)}$ be an honest tree rooted in genesis resulting from an execution of protocol π in a δ-synchronous network: it consists of*

all chains broadcast by honest parties. Further, let $\mathsf{depth}^{\mathcal{T}}(i)$ *denote the length of the chain in* $\mathcal{T}$ *extended by the uniquely honest leader of slot* i*.* $\mathcal{T}$ *exhibits the* Δ*-monotonicity property if the following holds true.*

For all uniquely honest slots (i, j) *s.t.* $j \geq i + \Delta$ *:*

$$\mathsf{depth}^{\mathcal{T}}(j) > \mathsf{depth}^{\mathcal{T}}(i)$$

In PoS executed in a δ-synchronous setting, δ-monotonicity is trivially achieved: any honest block tip must arrive in the view of other honest party after δ slots, and is thus considered as a chain candidate for extension by any honest leader applying the longest chain selection rule.

We introduce a formal notion of *receipt delay*, which, informally, quantifies how far a local key extraction process is "behind schedule" due to adversarial delay of block arrival.

▶ **Definition 8 (Receipt delay).** *Let* $\boldsymbol{r}^{(i)}(B) : \mathcal{B} \to \mathbb{Z}_0$ *be the delay in slots between* $B.\mathsf{sl}$ *and the local arrival of block* B *from the view of the party* (i)*. If* B *is an empty-block* $(\perp)$*, we define the receipt delay to be* $\perp$*.*

When the extraction window is defined as a slot interval preceding genesis, where slot indices are "negative", let the receipt delay is defined as $\perp$. We allow a receipt delay of $\perp$ to be interpreted as a receipt delay of 0.

Attack on receipt delay consistency. The receipt delay of each block B must be bounded by the protocol in order to guarantee a fixed future slot in which the honest party can complete the extraction of the session key of B and produce a valid block. However, note that the receipt delay is defined on the local view of a single party; the adversary can trivially achieve inconsistencies in receipt delays between honest parties, such that arrival of a block is considered "on-time" by on party, but "too late" by another. We illustrate such an attack on receipt delay consistency (Eq. 3) between honest parties.

$$\mathcal{A} \xrightarrow{\mathcal{C}_{\mathcal{A}}} \mathcal{P}_i \xrightarrow{\mathcal{C}_{\mathcal{A}} | B_i} \{\mathcal{P}_j\}_{j \in \mathcal{H}} \tag{3}$$

Consider a chain $\mathcal{C}_{\mathcal{A}}$, which for simplicity sake consists of adversarial blocks only. Then, let the adversary forward the chain to honest party $\mathcal{P}_i$, such that all receipt delays for each block in $\mathcal{C}_{\mathcal{A}}$ are exactly the maximum receipt delay permitted by the protocol. Should honest $\mathcal{P}_i$ extend $\mathcal{C}_{\mathcal{A}}$ and subsequently deliver $\mathcal{C}_{\mathcal{A}} \mid B_i$ to all other honest parties $\{\mathcal{P}_j\}_{j \in \mathcal{H}: j \neq i}$ with a network delay of δ slots, the receipt delays of blocks in $\mathcal{C}_{\mathcal{A}}$ in the view of the other honest parties will violate the protocol, as these will be $\mathbf{r}^{(j)}(B) = \mathbf{r}^{(i)}(B) + \delta$ for each $B \in \mathcal{C}_{\mathcal{A}}$; by assumption, $\mathbf{r}^{(j)}(B)$ is the maximum permitted by the protocol. Thus, the honest chain tip B_i will never be extended by other honest parties, as its prefix $\mathcal{C}_{\mathcal{A}}$ contains blocks with invalid receipt delays, violating Δ-monotonicity for any Δ; in the view of other parties, $\mathcal{C}_{\mathcal{A}} \mid B_i$ is an invalid chain.

FairPoS key extraction. We overcome this class of attacks in FairPoS by permitting increasing receipt delay limits for blocks that are farther away from the chain tip (Figure 4). The high-level idea is as follows: To achieve Δ-monotonicity (Def. 7), we consider honest block leaders of slots which are Δ slots apart. Even if the receipt delay view of a block may differ from one honest leader at slot t to the leader of a later slot $t + \Delta$, the leader at $t + \Delta$ will grant the same block a higher, maximum receipt delay to account for any additional network delays induced by the adversary, as shown above. Our protocol permits independent Choices of extraction delay (d), network delay (δ), montonicity paramater (Δ) satisfying Eq. 5. We

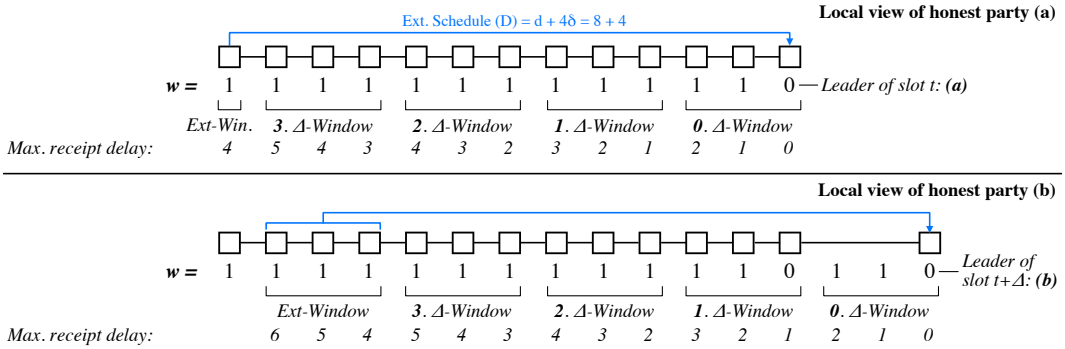


Figure 3 Key extraction in $(d{=}8, \delta{=}1, \Delta{=}3)$-FairPoS with maximally permitted receipt delays.

provide a discussion of parameterizations of (d, δ, Δ)-FairPoS in Sec. 4.3. Before formalising the FairPoS key extraction protocol, we provide an example parameteriziation of FairPoS for intuition.

▶ **Example 9.** We illustrate an execution of $(d{=}8, \delta{=}1, \Delta{=}3)$-FairPoS in Fig. 3, where the local chain view of honest parties (a) and (b) is shown. Here, party (a) is elected block leader At slot t, whereas party (b) is elected block leader at slot $t + \Delta$; for Δ-monotonicity to hold, the chain extended by (a) must be extendable by (b); in party (b)'s local view, this requires all blocks in the chain to respect the receipt delays imposed by the protocol, even if the adversary has induced *inconsistent receipt delays* between honest parties. For simplicity, we assume no leaderless slots ($\perp$) in this example.

Receipt delay limits ("max. receipt delay" in Fig. 3) imposed by FairPoS are organised in slot intervals, called **Δ-windows**. Beginning from the current slot, preceding slots are divided into slot windows of $\Delta = 3$ length. Each slot position within a n'th Δ-window is granted an *additional* receipt delay of delta in the subsequent $(n + 1)$'th Delta window. The **extraction window** defines the blocks for which the session keys must be extracted and included into the block of the current slot; note that the **extraction schedule** $(D = d + 4\delta = 12)$ is consistent with the maximum receipt delay $(4\delta = 4)$ imposed on blocks in the extraction window.

Fig. 3 illustrates a worst case scenario; honest party (a) is extending a fully adversarial chain, where the adversary has forwarded the blocks to party (a) only, with the maximally permitted receipt delays. Honest party (a) broadcasts the chain upon extending it in slot t, such that all blocks arrive at party (b) with an additional, worst case network delay $\delta = 1$. However, note that in the view of block leader party (b), each block of the chain in the n'th Δ-window is now in in the $(n + 1)$'th Δ-window, which tolerates an additional receipt delay of $\delta = 1$. Thus, this chain from party (a)is extendable by party (b) after Δ slots, implying Δ-monotonicity for $\Delta = 3$.

We highlight that each honest party has no knowledge of whether blocks where generated by an adversary or honest party. Further, no honest party can infer the true receipt delays in the local views of other honest parties. Informally, the FairPoS key extraction protocol is designed to accommodate the "worst case" adversarial interference as shown in this example.

4.3 The (d, δ, Δ)-FairPoS consensus protocol

We formalize the key extraction protocol introduced in §4.2 and present the full FairPoS execution model (Fig. 4) which also extends δ-PoS (Fig. 1) with a novel longest-extendable-chain selection rule. The design of FairPoS permits scheduled key extraction for each block

whilst maintaining Δ-monotoncity (Def. 10), essential for the security of FairPoS (§5).

Extraction window. A FairPoS protocol instance is parameterized with extraction schedule D, the duration after which the extracted session keys of a block are due in the next, available block. Here, we must account for the possibility of a *gap* between the chain tip and the current slot, for which no extracted keys could have been inserted. We therefore require a notion of an extraction window that denotes a window of preceding slots, for which associated blocks have key extractions due in the current slot. We first formalize $\mathsf{gap2tip}(t, \mathcal{C})$ as the number of *empty* slots between slot t and $\mathcal{C}.\mathsf{tip.sl}$.

$$\mathsf{gap2tip}(t, \mathcal{C}) = t - \mathsf{tip}(\mathcal{C}).\mathsf{sl} \quad \text{for} \quad t > \mathsf{tip}(\mathcal{C}).\mathsf{sl}$$

Then, the extraction window of current slot t and chain $\mathcal{C}$ with extraction schedule D can be defined as the range of slots, which are at least D slots in the past and are associated with blocks in $\mathcal{C}$ *without* key extractions already inserted in $\mathcal{C}$ due to a non-empty $\mathsf{gap2tip}(t, \mathcal{C})$.

$$\mathsf{extWin}_D(t, \mathcal{C}) = (t - D - \mathsf{gap2tip}(t, \mathcal{C}) : t - D\,] \tag{4}$$

Extraction schedule & Δ-windows. Let (d, δ, Δ)-FairPoS be parameterized by delay encryption parameter $d > 0$, maximum network delay $\delta \geq 0$ and desired monotonicity parameter $\Delta > \delta$ (Definition 7). Parameters d, δ and Δ are chosen to satisfy the following relation. Note that n denotes the number of Δ-windows in extraction schedule D (see Δ-windows in Fig. 3).

$$D = n\Delta \quad \text{where} \quad n = d/(\Delta - \delta) \text{ s.t. } n \in \mathbb{Z} \tag{5}$$

In words: the extraction schedule D is divisible by Δ. Furthermore, the number of Δ-windows in D must equal $d/(\Delta - \delta)$. We define the m'th "Δ-window" of current slot t as the following slot interval, consistent with Figure 3.

$$\Delta\mathsf{Win}_D(t, m) = (t - (m+1)\Delta : t - m\Delta] \quad \text{for} \quad m \in [0 : \frac{D}{\Delta}) \tag{6}$$

Chain extendability. A chain $\mathcal{C}$ is extendable by leader of slot t with local receipt delay view $\mathbf{r}^{(i)}$ if the receipt delays specific to each slot and Δ-window are satisfied, as implied by conditions ①, ② and ③ in Equation (7).

$$\mathsf{ext}(t, \mathcal{C}, \mathbf{r}^{(i)}) = \begin{cases} 1 & ① \wedge ② \wedge ③ \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned}
&① \quad \forall m \in [0 : \tfrac{D}{\Delta}) : \\
&\qquad \forall j \in \Delta\mathsf{Win}_D(t, m) : \quad \mathbf{r}^{(i)}(\mathcal{C}_j) \leq m\delta + (t - m\Delta - j) \\
&② \quad \forall j \in \mathsf{extWin}_D(t, \mathcal{C}) : \quad \mathbf{r}^{(i)}(\mathcal{C}_j) \leq \tfrac{D}{\Delta}\delta + (t - D - j) \\
&③ \quad \forall B \in \mathcal{C} : \qquad\qquad B \text{ contains valid } \mathsf{idk} \text{ for each block in its } \mathsf{extWin}
\end{aligned} \tag{7}$$

Concretely, condition ① reflects the maximum permitted receipt delays for each slot in the m'th Δ-window. Condition ② describes the maximum permitted receipt delays for slots in the extraction window (Eq. 4). Observe that these conditions are satisfied in the view of parties (a) and (b) in Figure 3 since the maximum receipt delays imposed by FairPoS are satisfied. Condition ③ states that all blocks in the chain must include valid session keys from past, parent blocks in their respective extraction windows.

▶ **Theorem 10.** *(Δ-Monotonicity of FairPoS) Every protocol execution of (d, δ, Δ)-FairPoS results in an honest tree $\mathcal{T}$ that exhibits the Δ-monotonicity property.*

Proof. (Sketch) Δ-monotonicity holds, because each slot in the m'th Δ-window is granted an additional δ receipt delay budget in the $m+1$'th Δ-window. Thus, if a chain is extendable (eq. 7) in the view of an honest party, the same chain must be extendable by all other parties following δ slots, despite the attack shown in eq. 3 that induces inconsistent receipt delays. Intuitively, the protocol can tolerate worst-case, receipt delay inconsistencies of δ slots from such attacks because each block is granted an additional δ in delay budget after Δ slots. The formal proof of Theorem 10 is stated in Appendix C. ◀

FairPoS execution model. We can now state the full FairPoS protocol and its execution model in Fig. 4. FairPoS extends the δ-PoS model in Figure 1 with the longest-extendable-chain selection rule, key extractions and the insertion of delay encrypted inputs, generated by the procedure in Figure 2. The protocol execution state of (d, δ, Δ)-FairPoS is extended with local receipt delays:

$$\Gamma_t = \left(\{\mathcal{T}^{(i)}, \mathbf{m}^{(i)}, \mathbf{r}^{(i)}\}_{i \in \mathcal{H}}, \mathcal{T}^{\mathcal{A}}\right) \tag{8}$$

Views in initial state Γ_0 contain a genesis block which includes public parameters $\mathsf{DE.pk}, \mathsf{DE.ek}$. Importantly, we *require* the adversary $\mathcal{A}$ and each honest party to perform exactly one extraction step for each pending key extraction in each round of the execution. Thus, no party or adversary gains a time advantage in extracting session keys from blocks (See E1 in Figure 4). It remains to formally define the longest extendable chain selection in the local view of an honest party.

Longest-extendable-chain selection. In FairPoS, an honest slot leader choses to extend the *longest extendable chain* in its local tree view $\mathcal{T}^{(i)}$, where chain extendability was previously defined in Eq. 7.

$$\mathsf{maxExtChain}(t, \mathcal{T}^{(i)}, \mathbf{r}^{(i)}) = \underset{\mathcal{C} \in \mathcal{T}^{(i)} : \mathsf{ext}(t, \mathcal{C}, \mathbf{r}^{(i)})}{\arg\max} \mathsf{len}(\mathcal{C}) \tag{9}$$

4.4 Parameterization of FairPoS

We note that in parameterizing (d, δ, Δ)-FairPoS, the maximum network delay δ is conservatively chosen to reflect network realities. For given δ, monotonicity parameter Δ and key extraction delay d can be chosen quasi-independently. For given choices of $\Delta > \delta$, any d is permitted that satisifies Equation (5); namely, d must be any multiple of $(\Delta - \delta)$. In Section 5, we show that the probability of input fairness being violated by the adversary falls exponentially in d (Theorem 19). Thus, the security of input fairness can be chosen independently from that of common-prefix, chain growth and chain quality properties. This is important for utility; a larger d increases the robustness of input fairness, but this comes at a cost of the freshness of blockchain state.

Freshness of visible blockchain state. All proposed encrypt-and-reveal approaches (Section 2) to input fairness naturally accept a comprise in the "freshness" of visible blockchain state; a delay parameter d naturally induces a gap between the current slot, and the most recent block in the past with already decrypted inputs. In practice, clients must submit inputs which are then applied to a blockchain state which is not yet decrypted - in other words, clients must generate inputs based on older blockchain state. In applications with frequent

(d, δ, Δ)-FairPoS consensus model

The (d, δ, Δ)-FairPoS state $\Gamma_t = \left(\{\mathcal{T}^{(i)}, \mathbf{m}^{(i)}, \mathbf{r}^{(i)}\}_{i \in \mathcal{H}}, \mathcal{T}^{\mathcal{A}}\right)$ evolves in a single round induced by environment $\mathcal{Z}$, adversary $\mathcal{A}$ and characteristic string w as follows.

Trees. Honest $\{\mathcal{T}^{(i)}\}_{i \in \mathcal{H}}$ and adversarial $\mathcal{T}^{\mathcal{A}}$ evolve as in δ-PoS (Fig. 1) except that the honest leader i extends the **longest-extendable chain** (Eq. 9) in its view $\mathcal{T}^{(i)}$.

Msgs. Honest message queues $\{\mathbf{m}^{(i)}\}_{i \in \mathcal{H}}$ evolve as in δ-PoS (Fig. 1).

Receipt delays. The receipt delays $\{\mathbf{r}^{(i)}\}_{i \in \mathcal{H}}$ in Γ_t evolve as follows:

R1 For each chain $\mathcal{C}$ delivered to honest party i by $\mathcal{A}$ at slot $t+1$: for each newly seen block $\mathcal{C}_j \in \mathcal{C}$, party i records $\mathbf{r}^{(i)}(\mathcal{C}_j) \leftarrow t + 1 - \mathcal{C}_j.\mathsf{sl}$.

Extraction. Each honest party $i \in \mathcal{H}$ and adversary $\mathcal{A}$ performs:

E1 In each round, exactly a single Extract step on each block in its local view for which key extraction is pending.

Extend. Party i generates $B = (t, H(\mathcal{C}.\mathsf{tip}), d_{\mathsf{ext}} | d_{\mathsf{ins}}, P_i)$ where

- $d_{\mathsf{ext}} = (\mathsf{idk}, \mathsf{idk}', ...)$ are extractions of blocks in $\mathcal{C}$ within $\mathsf{extWin}_D(t, \mathcal{C})$ (Eq. 4).
- $d_{\mathsf{ins}} = \{(c_j, m_j, \sigma_j)\}_{j \in [n]}$ is an ordering of encrypted inputs from enviroment $\mathcal{Z}$; Upon receiving d_{ins} from $\mathcal{Z}$, party i asserts for each entry; $\exists \mathsf{vk}_j \in \mathsf{accts}(\mathcal{C}.\mathsf{tip}) : 1 \leftarrow \mathsf{KES.Verify}_{\mathsf{vk}_j}((c_j | m_j | \mathcal{C}.\mathsf{tip}), \sigma_j)$.

Then, party i adds chain $\mathcal{C}' = \mathcal{C} \mid B$ to its local view.

Figure 4 Model of FairPoS execution induced by environment $\mathcal{Z}$ and characteristic string w. Encrypted inputs are generated in the input procedure in Fig 2 of §4.1.

inputs accessing the same public state, such as in decentralized finance [35], this likely results in increased number of invalid or reverted transactions, affecting the utility of the blockchain. In decentralized exchanges, for example, the market price is constantly adjusted with each submitted trade. A trade order with a price limit informed by an outdated price can become invalidated, as the older, lower price is no longer be available when the encrypted trade order is finally decrypted. We emphasize that the trade-off between utility and robustness of input fairness (Thm. 19) in FairPoS can be adjusted independently of the structural blocktree properties of common-prefix (Thm. 15), chain growth (Thm. 16) and chain quality (Thm. 17); in some applications a small, yet non-neglible probability of front-running may be acceptable for a fresher, decrypted blockchain state offered by a smaller delay extraction d parameter.

Decryption gap. Parameterizations of (d, δ, Δ)-FairPoS always imply an extraction schedule greater than the extraction delay parameter ($D > d$) for non-zero network delay parameter δ (Eq. 5). Let $D - d$ denote the "decryption gap"; informally, it represents how much sooner the consensus node performing key extraction can decrypt a block compared to a light-weight client, which must wait for the extracted key to be included in a later block. This "slack" in extraction schedule D is necessary to maintain consensus security due to adversarial block delays attacking receipt delay consistency between honest users illustrated in §4.2. We emphasize that the decryption gap $D - d$ does not represent a compromise in input fairness or structural security properties; instead, it reflects the efficacy of the FairPoS key extraction protocol, which aims to provide extracted keys to lightweight clients as soon as possible after these are extracted.

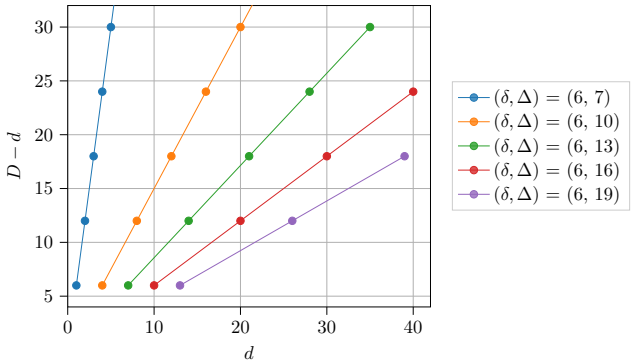


Figure 5 The *decryption gap* $(D - d)$ between key extraction schedule (D) and key extraction delay (d) is shown for selected parameterizations of (d, δ, Δ)-FairPoS. Decryption gap $D - d$ can be interpreted as the "efficacy" of the key extraction protocol in FairPoS, which aims to provide extracted keys to blocks as soon as possible.

Figure 5 shows the "decryption gap" $D - d$ for selected protocol parameters $(d, \delta = 6, \Delta)$. We highlight the first trade-off between $\Delta - \delta$ and decryption gap $D - d$; a higher monotonicity parameter Δ in $\Delta - \delta$ implies slower chain extendability, but can be exchanged for a smaller decryption gap, implying higher chain utility for light clients. Furthermore, delay encryption delay parameter d and decryption gap $D - d$ are linearly correlated for given δ and Δ; designing a secure key extraction protocol with sub-linear decryption gap represents an interesting problem for future work.

5 FairPoS security

Having presented a formal model of (d, δ, Δ)-FairPoS we proceed to demonstrate that it satisfies common-prefix (Def. 15), chain growth (Thm. 16), chain quality (Thm. 17) and input fairness (Thm. 19).

Informally, common-prefix is demonstrated by showing that any execution of (d, δ, Δ)-FairPoS produces local blocktree views which could have resulted from a Δ-PoS execution; since common-prefix is interpreted as a structural branching property of local blocktree views, this allows us to directly inherit common-prefix from Δ-PoS previously proven in Praos [18]. Chain growth and chain quality are implied by Δ-monotonicity of FairPoS, previously shown in Theorem 10. Finally, input fairness in FairPoS is inferred from common-prefix, chain growth and chain quality.

5.1 Common-prefix in FairPoS

Demonstrating k-common-prefix in FairPoS is accomplished by formally relating the tree views generated in an execution of (d, δ, Δ)-FairPoS with those that could have resulted from δ-PoS. Let the *honest tree* of protocol execution state

$$\Gamma_t = \left(\{\mathcal{T}^{(i)}, \mathbf{m}^{(i)}, \mathbf{r}^{(i)}\}_{i \in \mathcal{H}}, \mathcal{T}^{\mathcal{A}}\right)$$

be given be the union of honest tree views at slot t:

$$\mathcal{T}^{\mathcal{H}}(\Gamma_t) = \bigcup_{i \in \mathcal{H}} \mathcal{T}^{(i)} \tag{10}$$

In the analysis of PoS [18, 29], the branching structure of the honest tree informs us about events where a local chain was abandoned for a longer, alternative branch according to the *longest chain* selection rule. Informally, the common-prefix property is violated if the k-common-prefix shared between prior and newly adopted chains if their shared prefix is "too short". We begin our analysis with the (viable) branches which could have been adopted by honest parties in the first place.

Viable branches in PoS. A viable branch $\mathcal{B}$ in a tree $\mathcal{T}$ must exceed all honest chains-tips generated more than δ slots prior to $\mathcal{B}.\mathsf{tip.slot}$: this property arises from the honest application of the *longest chain rule*. The honest leader of $\mathcal{B}.\mathsf{tip.slot}$ must have received any honest chain-tips generated δ slots prior and considered them as *alternative candidates* for extension. Therefore, the *viable* branch $\mathcal{B}$ must exceed these in length. For branch $\mathcal{B} \in \mathcal{T}$, we first formalize its set of alternative chain candidates as follows.

$$\begin{aligned} &\mathbf{altChns}_\delta(\mathcal{B}, \mathcal{T}) = \\ &\{\, \mathcal{C} \subseteq \mathcal{T} \mid \mathcal{C}.\mathsf{tip.ldr} \in \mathcal{H} \ \wedge\ \mathcal{B}.\mathsf{tip.sl} > \mathcal{C}.\mathsf{tip.sl} + \delta \,\} \end{aligned} \tag{11}$$

For a PoS protocol execution state $\Gamma_t = \left(\{\mathcal{T}^{(i)}, \mathbf{m}^{(i)}\}_{i \in \mathcal{H}}, \mathcal{T}^{\mathcal{A}}\right)$, the set of well-formed, viable branches is formalized as the set of branches with lengths exceeding their honest, alternative chains.

$$\begin{aligned} &\mathbf{viableBranches}_\delta^{\mathsf{PoS}}(\Gamma_t) = \\ &\{\forall \mathcal{B} \in \mathcal{T}^{\mathcal{H}}(\Gamma_t) \mid \mathsf{len}(\mathcal{B}) > \underset{\mathcal{C} \in \mathbf{altChns}_\delta(\mathcal{B}, \mathcal{T}^{\mathcal{H}}(\Gamma_t))}{\arg\max} \mathsf{len}(\mathcal{C}) \ \} \end{aligned} \tag{12}$$

Viable chains in FairPoS. The notion of viable branches must be strengthened for FairPoS since the *longest-extendable-chain* rule introduces additional constraints for the adoption of a chain in the local honest tree view. Let the *extendable prefix* of a branch $\mathcal{B}$ in the view of honest parties at slot t be defined as the "longest extendable prefix" of a branch.

$$\mathbf{extPrefix}(t, \mathcal{B}, \{\mathbf{r}^{(i)}\}_{i \in \mathcal{H}}) = \underset{\mathcal{C} \preceq \mathcal{B} \,:\, \exists i \in \mathcal{H} \,:\, \mathsf{ext}(t, \mathcal{C}, \mathbf{r}^{(i)})}{\arg\max} \mathsf{len}(\mathcal{C}) \tag{13}$$

For a (d, δ, Δ)-FairPoS state, let the set of viable chains be defined as the extendable prefixes (Equation (13)) of branches in the honest tree with lengths which exceed those of its alternative chains (Equation (11)) generated Δ slots prior: by the Δ-monotonicity property (Theorem 10), these chains must have been extendable by the leader that generated the respective prefix and considered these as candidates for extension. Viable chains in FairPoS are defined as;

$$\begin{aligned} &\mathbf{viableChains}_\Delta^{\mathsf{FairPoS}}(\Gamma_t) = \{\, \mathcal{C} \subseteq \mathcal{T}^{\mathcal{H}}(\Gamma_t) \mid \ ① \wedge ② \,\} \\ &① \ \exists \mathcal{B} \in \mathcal{T}^{\mathcal{H}}(\Gamma_t) : \mathcal{C} = \mathbf{extPrefix}(t, \mathcal{B}, \{\mathbf{r}^{(i)}\}_{i \in \mathcal{H}}) \\ &② \ \mathsf{len}(\mathcal{C}) > \underset{\mathcal{C}' \in \mathbf{altChns}_\Delta(\mathcal{C}, \mathcal{T}^{\mathcal{H}}(\Gamma_t))}{\arg\max} \mathsf{len}(\mathcal{C}') \end{aligned} \tag{14}$$

In words; a viable chain in FairPoS and the local honest view must ① be an extendable prefix and ② this prefix must also exceed its alternative chains in length.

Next, we restate the *divergence* notion from Praos [18, 29] which formally describes the *magnitude* of branching caused by the switching between viable chains.

▶ **Definition 11 (Divergence).** *For two chains $\mathcal{C}_1$ and $\mathcal{C}_2$, define their divergence to be the quantity*

$$\mathsf{div}(\mathcal{C}_1, \mathcal{C}_2) = \mathsf{min}(\mathsf{len}(\mathcal{C}_1), \mathsf{len}(\mathcal{C}_2)) - \mathsf{len}(\mathcal{C}_1 \cap \mathcal{C}_2)$$

where $\mathcal{C}_1 \cap \mathcal{C}_2$ denotes the common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. We extend this notion of divergence to the protocol execution state Γ resulting from the execution of protocol π induced by characteristic w in the δ-synchronous setting: here, the maximum divergence over any two viable chains is quantified.

$$\mathsf{div}_\delta^\pi(\Gamma) = \mathsf{max}_{\mathcal{C}_1,\mathcal{C}_2 \in \mathsf{viableBranches}_\delta^\pi(\Gamma)} \mathsf{div}(\mathcal{C}_1, \mathcal{C}_2)$$

Finally, we define the divergence of a characteristic string w to be the maximum divergence observable over all states which could have resulting from protocol executions induced by w. More formally, let $\mathsf{exec}_\delta^\pi(\Gamma_0, w)$ denote all possible executions of π beginning with state Γ_0 which could have been induced by w in δ-synchronous network. Then the divergence of a characteristic string w is defined as:

$$\begin{aligned} \mathsf{div}_\delta^\pi(w) &= \mathsf{max}_{\Gamma \in \mathsf{reachable}_\delta^\pi(\Gamma_0,w)} \mathsf{div}_\delta^\pi(\Gamma) \\ \textit{where } \mathsf{reachable}_\delta^\pi(\Gamma_0, w) &= \{\Gamma \mid \exists \lambda \in \mathsf{exec}_\delta^\pi(\Gamma_0, w) : \Gamma_0 \xrightarrow{\lambda} \Gamma\} \end{aligned} \tag{15}$$

For PoS, the probability that that the divergence exceeds k-blocks over an execution of R slots, is given by the following theorem from [18]. Bounding the probability of divergence naturally implies common-prefix security.

▶ **Theorem 12.** *(PoS Divergence, Theorem 4 in [18]) Let active slot coefficient $f \in (0, 1]$ and α be such that $\alpha(1-f)\Delta = (1+\epsilon)/2$ for some $\epsilon > 0$. Further, let w be a string drawn from $\{0, 1, \perp\}^R$ according to D_α^f. Then we have $\Pr_{w \leftarrow\$ D_\alpha^f}[\mathit{div}_\Delta^{\mathit{PoS}}(w) \geq k] \leq \exp(\ln(R) - \Omega(k - \Delta))$.*

Towards demonstrating k-common prefix in FairPoS, we first present a central theorem, which states that for all executions of (d, δ, Δ)-FairPoS in the δ-synchronous setting, there exists an execution of PoS in the Δ-synchronous setting, such that viable chains of the d, δ, Δ-FairPoS honest tree are *equivalent* ($\equiv$) to the viable branches of the PoS honest tree: here, we define the equivalence of chains such that only their structural properties are considered, formally stated in Definition 21.

▶ **Theorem 13. (Equivalent trees)** For any (d, δ, Δ)-FairPoS execution λ induced by a charactistic string $w \in \{0, 1, \perp\}^*$, $\Gamma_0 \to^\lambda \Gamma$, there exists a Δ-PoS execution λ' induced by same w, $\Gamma_0' \to^{\lambda'} \Gamma'$, such that the viable chains in Γ are equivalent to the viable branches in Γ'.

$$\begin{aligned} \forall w \in \{0,1,\perp\}^* : \quad & \forall \lambda \in \mathsf{exec}_\delta^{\mathsf{FairPoS}}(\Gamma_0, w), \Gamma_0 \xrightarrow{\lambda} \Gamma : \\ & \exists \lambda' \in \mathsf{exec}_\Delta^{\mathsf{PoS}}(\Gamma_0', w), \Gamma_0 \xrightarrow{\lambda'} \Gamma' : \\ \mathbf{viableChains}_\delta^{\mathsf{FairPoS}}(\Gamma) &\equiv \mathbf{viableBranches}_\Delta^{\mathsf{PoS}}(\Gamma') \end{aligned}$$

We refer to the full proof of Theorem 13 in Appendix C; here, we demonstrate how to match any honest and adversarial action in Δ-PoS with a corresponding honest or adversarial action in (d, δ, Δ)-FairPoS such that the viable branches of (d, δ, Δ)-FairPoS evolve in lockstep with the viable chains in the Δ-PoS execution.

Since divergence is defined over viable chains and viable branches, we can infer Corollary 14 from Theorem 13.

▶ **Corollary 14.** $\forall w \in \{0, 1, \perp\}^* : \mathit{div}_\delta^{\mathit{FairPoS}}(w) \leq \mathit{div}_\Delta^{\mathit{PoS}}(w)$

This allows us to infer k-common-prefix from bounding the probability of the event $\mathsf{div}_\Delta^{\mathsf{PoS}}(w) > k$ for $w \leftarrow\$ \mathcal{D}_\alpha^f$ in PoS [18].

▶ **Theorem 15. (k-Common prefix in FairPoS)** *Let $\mathcal{A}$ be an an adaptive adversary against the protocol (d, δ, Δ)-FairPoS that corrupts up to $(1-\alpha)$ stake, where α be such that $\alpha(1-f)\Delta = (1+\epsilon)/2$ for active slot coefficient $f \in (0,1]$ and some $\epsilon > 0$. The probability that $\mathcal{A}$ makes the protocol violate the k-common-prefix property in a δ-synchronous environment throughout a period of R slots is no more than* $\exp(\ln R + \Delta - \Omega(k))$.

Proof. Let w be drawn from dominant distribution $\mathcal{D}_\alpha^f$ (Equation (1)), with honest stake α and parameter f satisfying $\alpha(1-f)\Delta = (1+\epsilon)/2$ for some $\epsilon > 0$. From Corollary 14 and Theorem 12 we infer the following:

$$\Pr_{w \leftarrow \mathcal{D}_\alpha^f}[\mathsf{div}_\delta^{\mathsf{FairPoS}}(w) \geq k] \leq \Pr_{w \leftarrow \mathcal{D}_\alpha^f}[\mathsf{div}_\Delta^{\mathsf{PoS}}(w) \geq k] \leq \exp(\ln R + \Delta - \Omega(k)) \tag{16}$$

From Corollary 14, $\mathsf{div}_\delta^{\mathsf{FairPoS}}(w) \geq k \implies \mathsf{div}_\Delta^{\mathsf{PoS}}(w) \geq k$, implying the left equality in Equation (16). The right inequality is inferred from Theorem 12. ◀

5.2 Chain growth, chain quality and input fairness

Both chain growth and chain quality of FairPoS can be derived from Δ-monotonicity of FairPoS (Theorem 10) and probabilities bounding security failure from PoS [18]. We refer to Appendix C for the full proofs of the following security theorems.

▶ **Theorem 16. ((τ, s)-Chain growth in (d, δ, Δ)-FairPoS)** Let $\mathcal{A}$ be an an adaptive adversary against the protocol (d, δ, Δ)-FairPoS that corrupts up to $(1-\alpha)$ stake, where α be such that $\alpha(1-f)\Delta = (1+\epsilon)/2$ for active slot coefficient $f \in (0,1]$ and some $\epsilon > 0$. Then the probability that $\mathcal{A}$ makes the protocol violate the chain growth property with parameters $s \geq 4\Delta$ and $\tau = c\alpha/4$ throughout a period of R slots, is no more than $\exp(-c\alpha s/(20\Delta) + lnR\Delta + O(1))$, where c denotes the constant $c := c(f, \Delta) = f(1-f)^\Delta$.

▶ **Theorem 17. ((μ, k)-Chain quality in (d, δ, Δ)-FairPoS)** Let $\mathcal{A}$ be an adaptive adversary against the protocol (d, δ, Δ)-FairPoS that corrupts up to $(1-\alpha)$ stake, where α be such that $\alpha(1-f)\Delta = (1+\epsilon)/2$ for active slot coefficient $f \in (0,1]$ and some $\epsilon > 0$. Then the probability that $\mathcal{A}$ makes FairPoS violate the chain quality property with parameters k and $\mu = 1/k$ throughout a period of R slots, is no more than $\exp(\ln R - \Omega(k))$.

Input fairness is obtained from chain growth, common prefix and chain quality. Informally, given time d and chain growth rate τ, we can determine common-prefix and chain quality parameters such that a decrypted input must have sufficient time (d slots) to reach finalization or lie in an abandoned chain.

▶ **Lemma 18. (Input fairness from CG, CP and CQ in (d, δ, Δ)-FairPoS)** Input fairness is implied in an execution of (d, δ, Δ)-FairPoS, in which (τ, d)-chain growth, $(d\tau(\tau - \delta/(\Delta-\delta)) - 1)$-common prefix and $(1/(D+1), D+1)$-chain quality hold, where $D = d\Delta/(\Delta-\delta)$

In the full proof of Lemma 18 in Appendix C, we derive the relation between the security parameters of chain growth, common-prefix and chain quality shown above such that there is always sufficient time for an honestly encrypted input to reach the common-prefix; here we must carefully accomodate "time advantages" granted to the adversary. Firstly, observe that whenever there are empty slots between the chain tip and current slot, the adversary is granted time to decrypt the current session key without any progress towards finalization. Secondly, in the case of a leading adversarial block-span, the adversary is granted another head start in completing key extraction, as it can generate the block span prior to their associated slots and begin key extraction ahead of time.

▶ **Theorem 19. (Input fairness in (d, δ, Δ)-FairPoS)** Let $\mathcal{A}$ be an adaptive adversary against the protocol (d, δ, Δ)-FairPoS that corrupts up to $(1 - \alpha)$ stake, where α be such that $\alpha(1 - f)\Delta = (1 + \epsilon)/2$ for active slot coefficient $f \in (0, 1]$ and some $\epsilon > 0$. Then, the probability that $\mathcal{A}$ makes the FairPoS violate the input fairness property falls exponentially with d.

Theorem 19 is proven in Appendix C with Lemma 18 which relates the delay extraction parameter d with the security parameters of k-common-prefix, (τ, s)-chain growth and (μ, k)-chain quality.

6 Conclusion

We contribute FairPoS, the first longest-chain, proof-of-stake protocol achieving input fairness against an adaptive adversary. When adopting the leader election procedure from Ouroborous Praos [18] or one that induces leader sequences (*i.e.* characteristic strings) consistent with the *dominant distribution* (Definition 1), FairPoS achieves input fairness in addition to the common-prefix, chain-growth and chain-quality properties of PoS [18]. We note that Kiayas *et al.* [29] provide tighter bounds for k-common prefix in PoS. Applying this updated analysis framework to security of FairPoS is planned as future work.

References

1 VDF Alliance. VDF Alliance Official Wiki. `https://supranational.atlassian.net/wiki/spaces/VA/overview`, 2022.

2 Avalanche. Apricot Phase Four: Snowman++ and Reduced C-Chain Transaction Fees. `https://medium.com/avalancheavax/apricot-phase-four-snowman-and-reduced-c-chain-transaction-fees-1e1f67b42ecf`, 2021.

3 Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 913–930, 2018. `https://doi.org/10.1007/978-3-319-78375-8_3`.

4 Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 585–602, 2019. `https://doi.org/10.1145/3319535.3363213`.

5 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. Maximizing extractable value from automated market makers. *arXiv preprint arXiv:2106.01870*, 2021. `https://arxiv.org/pdf/2106.01870`.

6 Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. Tardis: a foundation of time-lock puzzles in uc. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part III*, pages 429–459. Springer, 2021. `https://doi.org/10.1007/978-3-030-77883-5_15`.

7 Joseph Bebel and Dev Ojha. Ferveo: Threshold Decryption for Mempool Privacy in BFT networks. *Cryptology ePrint Archive*, 2022. `https://eprint.iacr.org/2022/898`.

8 Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, August 1999. `doi:10.1007/3-540-48405-1_28`.

9 Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In Rafael Pass

and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 260–290. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64375-1_10.

10 Erica Blum, Aggelos Kiayias, Cristopher Moore, Saad Quader, and Alexander Russell. Linear consistency for proof-of-stake blockchains. *arXiv preprint arXiv:1911.10187*, 2019. https://arxiv.org/abs/1911.10187.

11 Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000 Proceedings*, pages 236–254. Springer, 2000. https://link.springer.com/content/pdf/10.1007/3-540-44598-6.pdf#page=248.

12 Jeffrey Burdges and Luca De Feo. Delay encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 302–326. Springer, 2021. https://doi.org/10.1007/978-3-030-77870-5_11.

13 Christian Cachin, Jovana Mićić, and Nathalie Steinhauer. Quick Order Fairness. *arXiv preprint arXiv:2112.06615*, 2021. https://arxiv.org/abs/2112.06615.

14 Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. Encryption to the future - A paradigm for sending secret messages to future (anonymous) committees. In *ASIACRYPT 2022, Part III*, LNCS, pages 151–180. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22969-5_6.

15 Ignacio Cascudo, Bernardo David, Lydia Garms, and Anders Konring. YOLO YOSO: Fast and simple encryption and secret sharing in the YOSO model. In *ASIACRYPT 2022, Part I*, LNCS, pages 651–680. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22963-3_22.

16 Dan Cline, Thaddeus Dryja, and Neha Narula. ClockWork: An Exchange Protocol for Proofs of Non Front-Running.

17 P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *IEEE Symposium on Security and Privacy*, pages 910–927. IEEE, 2020. https://doi.org/10.1109/SP40000.2020.00040. doi:10.1109/SP40000.2020.00040.

18 Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018. https://doi.org/10.1007/978-3-319-78375-8_3.

19 Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 248–277. Springer, 2019. https://doi.org/10.1007/978-3-030-34578-5_10.

20 Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. Mcfly: Verifiable encryption to the future made practical. *Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/433.

21 Andreas Erwig, Sebastian Faust, and Siavash Riahi. Large-scale non-interactive threshold cryptosystems through anonymity. Cryptology ePrint Archive, Report 2021/1290, 2021. https://eprint.iacr.org/2021/1290.

22 Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 281–310. Springer, 2015. https://doi.org/10.1007/978-3-662-46803-6_10.

23 Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. In *PKC 2022, Part I*, LNCS, pages 252–282. Springer, Heidelberg, May 2022. doi:10.1007/978-3-030-97121-2_10.

24 Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 332–354. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_20.

25 Fredrik Kamphuis, Bernardo Magri, Ricky Lamberty, and Sebastian Faust. Revisiting transaction ledger robustness in the miner extractable value era. In *International Conference on Applied Cryptography and Network Security*, pages 675–698. Springer, 2023. https://doi.org/10.1007/978-3-031-33491-7_25.

26 Mahimna Kelkar, Soubhik Deb, and Sreeram Kannan. Order-fair consensus in the permissionless setting. In *Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop*, pages 3–14, 2022. https://doi.org/10.1145/3494105.3526239.

27 Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, Strong Order-Fairness in Byzantine Consensus, 2021. https://eprint.iacr.org/2021/1465.

28 Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In *Annual International Cryptology Conference*, pages 451–480. Springer, 2020. https://doi.org/10.1007/978-3-030-56877-1_16.

29 Aggelos Kiayias, Saad Quader, and Alexander Russell. Consistency of proof-of-stake blockchains with concurrent honest slot leaders. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 776–786. IEEE, 2020. https://doi.org/10.1109/ICDCS47774.2020.00065.

30 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017. https://doi.org/10.1007/978-3-319-63688-7_12.

31 Dahlia Malkhi and Pawel Szalachowski. Maximal Extractable Value (MEV) Protection on a DAG. *arXiv e-prints*, pages arXiv–2208, 2022. https://arxiv.org/abs/2208.00940.

32 Peyman Momeni. Fairblock: Preventing blockchain front-running with minimal overheads. Master's thesis, University of Waterloo, 2022. https://eprint.iacr.org/2022/1066.

33 Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996. http://bitsavers.trailing-edge.com/pdf/mit/lcs/tr/MIT-LCS-TR-684.pdf.

34 Sri Aravinda Krishnan Thyagarajan, Tiantian Gong, Adithya Bhat, Aniket Kate, and Dominique Schröder. Opensquare: Decentralized repeated modular squaring service. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3447–3464, 2021. https://doi.org/10.1145/3460120.3484809.

35 Sam M Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J Knottenbelt. Sok: Decentralized finance (defi). *arXiv preprint arXiv:2101.08778*, 2021. https://arxiv.org/abs/2101.08778.

A Key evolving signature schemes

We present the formal definitions of key evolving signature scheme of [8, 24]

▶ **Definition 20.** *A key evolving signature scheme*

$$KES = (Gen, Sign, Verify, Update)$$

is a tuple of algorithms such that:

1. $Gen(1^k, T)$ *is a probabilistic key generation algorithm that takes as input a security parameter* 1^k *and the total number of periods* T*, outputting a key pair* $(KES.sk_1, KES.vk)$*, where* $KES.vk$ *is the verification key and* $KES.sk_1$ *is the initial signing key (we assume that the period* j *to which a signing key* $KES.sk_j$ *corresponds is encoded in the signing key itself).*
2. $Sign_{KES.sk_j}(m)$ *is a probabilistic signing algorithm that takes as input a secret key* $KES.sk_j$ *for the time period* $j \leq T$ *and a message* m*, outputting a signature* σ_j *on* m *for time period* j *(we assume that the period* j *for which a signature* σ_j *was generated is encoded in the signature itself).*
3. $Verify_{KES.vk}(m, \sigma_j)$ *is a deterministic verification algorithm that takes as input a public key* $KES.vk$*, a message* m *and a signature* σ_j*, outputting* 1 *if* σ_j *is a valid signature on message* m *for time period* j *and* 0 *otherwise.*
4. $Update(KES.sk_j)$ *is a probabilistic secret key update algorithm that takes as input a secret key* $KES.sk_j$ *for the current time period* j *and outputs a new secret key* $KES.sk_{j+1}$ *for time period* $j+1$*. We define* $KES.sk_{T+1}$ *as the empty string and set it as the output of* $Update(KES.sk_T)$*.*

Correctness: *for every key pair* $(KES.sk_1, KES.vk) \leftarrow Gen(1^k, T)$*, every message* m *and every time period* $j \leq T$*, the following holds.*

$$Verify_{KES.vk}(m, Sign_{KES.sk_j}(m)) = 1$$

B Chain equivalence

▶ **Definition 21** (Equivalence $\equiv$)**.** *Two chains* $\mathcal{C}_0$ *and* $\mathcal{C}_1$ *are equivalent,* $\mathcal{C}_0 \equiv \mathcal{C}_1$*, if* $\mathcal{C}'_0 = \mathcal{C}'_1$*, where* $\mathcal{C}'$ *is obtained from* $\mathcal{C}$ *with the following procedure:*

- *Let* $\mathcal{C} = (B_0, ..., B_R)$ *and set* $B'_0 \leftarrow (0, \epsilon, \epsilon, 0)$*,* $\mathcal{C}' \leftarrow B'_0$*,*
- *For* $k \in [1, R]$*:*
  - $B'_k \leftarrow (B_k.sl, H(B'_{k-1}), \epsilon, B_k.ldr)$
  - $\mathcal{C}' \leftarrow \mathcal{C}'|B'_k$

We lift this definition of equivalence to chain sets (or trees), where each chain in one set has exactly one equivalent chain in the other.

C Proofs

▶ **Theorem 10.** *(Δ-Monotonicity of FairPoS) Every protocol execution of* (d, δ, Δ)*-FairPoS results in an honest tree* $\mathcal{T}$ *that exhibits the* Δ*-monotonicity property.*

Proof. Δ-monotonicity holds if every chain-tip that is honestly generated at slot i is considered a candidate for extension by the first honest leader at or following slot $i + \Delta$. In other words, the proof obligation is to demonstrate honest chain extendability (Equation (7)) holds for every honest chain-tip, Δ slots following its generation. We prove this by induction:

Base case (Genesis). The genesis block at slot 0 is trivially extendable at any slot $\mathsf{sl} \geq \Delta$. Recall that we permit negative chain indices, e.g. $\mathcal{C}_{-j}$ for $j \in \mathbb{Z}$, which denote "empty blocks", for which the local receipt delay $\mathbf{r}^{(i)}(\mathcal{C}_{-j})$ is always $\perp$, and by the definition of receipt delay (Definition 8) is interpreted as 0. Thus, when extending genesis, every "empty block" in the extraction window and the "m"th Δ-window will not violate the receipt delay bounds imposed by the *extendable-chain* criteria in Equation (7).

Inductive case (Slot i). A chain $\mathcal{C}$ honestly extended at slot i must fulfill the receipt delay constraints in Equation (7) in the view of the slot leader. This honest leader will have rebroadcast every block in $\mathcal{C}$ upon arrival: thus, in the worst case, all other honest parties will have received blocks in $\mathcal{C}$ with an additional receipt delay of δ slots compared to the leader of slot i. The following must hold for $\mathcal{C}$ in the view of any honest leader of slot $j \geq i + \Delta$:

- For $m \in [0 : n)$: each block of $\mathcal{C}$ in the "m"-th Δ-window (Equation (6)) in the view of leader of slot i is now in the "$m + 1$"-th Δ-window in the view of the honest leader of slot j, it is permitted an additional *worst-case* additional delay of δ by the extendable chain definition (Equation (7)).
- Blocks in the "$n - 1$"th window of $\mathcal{C}$ in the view of leader i are in the extraction window of leader of slot j, and are also permitted an additional *worst-case* delay of δ (Equation (7)).

◀

▶ **Theorem 13. (Equivalent trees)** For any (d, δ, Δ)-FairPoS execution λ induced by a charactistic string $w \in \{0, 1, \perp\}^*$, $\Gamma_0 \rightarrow^{\lambda} \Gamma$, there exists a Δ-PoS execution λ' induced by same w, $\Gamma'_0 \rightarrow^{\lambda'} \Gamma'$, such that the viable chains in Γ are equivalent to the viable branches in Γ'.

$$\forall w \in \{0,1,\perp\}^* : \quad \forall \lambda \in \mathsf{exec}_{\delta}^{\mathsf{FairPoS}}(\Gamma_0, w), \Gamma_0 \xrightarrow{\lambda} \Gamma : \\ \exists \lambda' \in \mathsf{exec}_{\Delta}^{\mathsf{PoS}}(\Gamma'_0, w), \Gamma_0 \xrightarrow{\lambda'} \Gamma' :$$

$$\mathbf{viableChains}_{\delta}^{\mathsf{FairPoS}}(\Gamma) \equiv \mathbf{viableBranches}_{\Delta}^{\mathsf{PoS}}(\Gamma')$$

Proof. Let $\mathsf{exec}_{\delta}^{\pi}(\Gamma, w_t)$ denote all possible *single round* executions of π in a δ-synchronous setting induced by $w_t \in \{0, 1, \perp\}$ at slot t, beginning with protocol state Γ. Further, we define the *honest extendable tree* of FairPoS state Γ_t as the union of the extendable prefixes (Equation (13)) of all tree branches in the view of honest parties.

$$\mathcal{T}_{\mathsf{ext}}^{\mathcal{H}}(\Gamma_t) = \bigcup_{\mathcal{C} \in \mathcal{T}^{\mathcal{H}} : \exists \mathcal{B} \in \mathcal{T}^{\mathcal{H}} : \mathcal{C} = \mathbf{extPrefix}(t, \mathcal{B}, \{\mathbf{r}^{(i)}\}_{i \in \mathcal{H}})} \mathcal{C}$$

For a FairPoS state Γ_t and PoS state Γ'_t where $\mathcal{T}_{\mathsf{ext}}^{\mathcal{H}}(\Gamma_t) \equiv \mathcal{T}^{\mathcal{H}}(\Gamma'_t)$ we observe that viable branches of Γ_t and viable chains of Γ'_t converge by definition.

$$\mathbf{viableChains}_{\delta}^{\mathsf{FairPoS}}(\Gamma_t) \equiv \mathbf{viableBranches}_{\Delta}^{\mathsf{PoS}}(\Gamma'_t)$$

We prove the theorem round-wise, by induction.

Base step ($0 \to 1$). For the first round executed on the genesis block (slot 0) and given a characterstic string w we must prove:

$$\begin{aligned} &\forall\, r \in \mathsf{exec}_{\delta}^{\mathsf{FairPoS}}(\Gamma_0, w_1)\ ,\ \Gamma_0 \xrightarrow{r} \Gamma_1 : \\ &\exists\, r' \in \mathsf{exec}_{\Delta}^{\mathsf{PoS}}(\Gamma'_0, w_1)\ ,\ \Gamma'_0 \xrightarrow{r'} \Gamma'_1 : \\ &\qquad \mathcal{T}_{\mathsf{ext}}^{\mathcal{H}}(\Gamma_1) \equiv \mathcal{T}^{\mathcal{H}}(\Gamma'_1) \end{aligned} \tag{17}$$

We describe the translation from r to r'. If w_1 is honest, then the elected honest parties will generate a block extending genesis in the rounds of both protocols (genesis is immediately extendable, as no key extractions are due in the block associated at slot 1). If w_1 is dishonest, then whatever the blocks the adversary generates in r is performed by the adversary in r': resulting extendable honest tree in Γ_1 and honest tree in Γ'_1 must be equivalent.

Induction step ($t \to t+1$). We must prove

$$\begin{aligned} &\forall\, r \in \mathsf{exec}_{\delta}^{\mathsf{FairPoS}}(\Gamma_t, w_{t+1})\ ,\ \Gamma_t \xrightarrow{r} \Gamma_{t+1} : \\ &\exists\, r' \in \mathsf{exec}_{\Delta}^{\mathsf{PoS}}(\Gamma'_t, w_{t+1})\ ,\ \Gamma'_t \xrightarrow{r'} \Gamma'_{t+1} : \\ &\qquad \mathcal{T}_{\mathsf{ext}}^{\mathcal{H}}(\Gamma_{t+1}) \equiv \mathcal{T}^{\mathcal{H}}(\Gamma'_{t+1}) \end{aligned} \tag{18}$$

By induction hypothesis it holds that this holds that $\mathcal{T}_{\mathsf{ext}}^{\mathcal{H}}(\Gamma_t) \equiv \mathcal{T}^{\mathcal{H}}(\Gamma'_t)$. For any honest or adversarial action in round r, we illustrate the respective action in round r', before arguing the equivalence of extendable honest trees in Γ_{t+1} and honest trees in Γ'_{t+1}.
Honest actions in $\mathsf{FairPoS}$ round r are also performed in PoS round r':

H1 If w_{t+1} is a uniquely honest slot, the longest (extendable) chains will be equivalent in the honest party's view, and the honest leader(s) will extend the equivalent chains of both protocol executions in rounds r and r' respectively. (By induction hypothesis, the the honest extendable tree in Γ_t is equivalent to the honest tree in Γ'_t). A newly extended chain from honest party i is added to the message queue $\mathbf{m}^{(i)}$ and $\mathcal{T}^{\mathcal{A}}$ in both r and r'.

Adversarial actions include dishonest block generation (T2 in Figure 1) and the delivery of messages (M2 in Figure 1). The translation of adversarial actions from r round to the r' is described.

A1 If the adversary adds dishonest blocks to $\mathcal{T}^{\mathcal{A}}$ at any adversarial slot $t' \leq t+1$ in the round r, this action is performed in r' round.

A2 If the adversary delivers a $\mathcal{C}$ with adversarial chain tip from $\mathcal{T}^{\mathcal{A}}$ to $\mathcal{T}^{(i)}$ in r:

a. If $\mathcal{C}$ is extendable by party i in state Γ_{t+1} following r, $\mathcal{A}$ delivers equivalent $\mathcal{C}'$ from $\mathcal{T}^{\mathcal{A}\prime}$ to $\mathcal{T}^{(i)\prime}$ in the r' round: $\mathcal{C}, \mathcal{C}'$ must both exist in the adversarial tree views in states Γ_t, Γ'_t due to **A1**.

b. Else if $\mathcal{C}$ *remains* unextendable in state Γ_{t+1} by party i, equivalent $\mathcal{C}'$ is not added to the equivalent honest tree $\mathcal{T}^{(i)\prime}$ in r'.

A3 If the adversary delivers a chain $\mathcal{C}$ from message queue $\mathbf{m}^{(i)}$ to an honest tree view $\mathcal{T}^{(i)}$ in r:

a. If $\mathcal{C}$ is extendable by party i in state Γ_{t+1} following r, $\mathcal{A}$ delivers equivalent $\mathcal{C}'$ to $\mathcal{T}^{(i)\prime}$ in the r' round.

b Else if $\mathcal{C}$ is not extendable by party i in state Γ_{t+1}, its equivalent $\mathcal{C}'$ is not added to the honest tree in round r': since $\Delta > \delta$, it is always possible for $\mathcal{A}$ defer an honest chain delivery for a longer delay in PoS (Δ-synchrony) than in the $\Delta\mathsf{PoS}$ (δ-synchrony) execution.

A4 If any $\mathcal{C} \in \mathcal{T}^{(i)}$ in an honest tree view *becomes* extendable in slot $t+1$ following round r in the view of party i.

a. If $\mathcal{C}$ and equivalent $\mathcal{C}'$ feature an honest chain tip it must be present in an honest message queue $\mathbf{m}^{(j)\prime}$ of the PoS execution (**H1**) and will be delivered to party i in round r'. An honest chain tip in a FairPoS execution can remain unextendable for up to Δ slots (Theorem 10), which is exactly the maximum message delay permitted for messages in $\mathbf{m}^{(j)\prime}$ in the PoS Δ-synchronous setting.

b. If $\mathcal{C}$ and equivalent $\mathcal{C}'$ feature an adversarial chain tip, $\mathcal{C}'$ cannot be in an honest tree view nor an honest message queue in the PoS execution prior to slot $t+1$: **A2** requires an adversarial chain tip in FairPoS to be extendable before it is introduced to an honest tree in PoS. Instead, any $\mathcal{C}'$ with adversarial tip must be in $\mathcal{T}^{\mathcal{A}}$, $\mathcal{T}^{\mathcal{A}\prime}$ of both executions (**A1**) at the beginning of the round, and $\mathcal{C}'$ can therefore still be delivered to $\mathcal{T}^{(i)\prime}$ from $\mathcal{T}^{\mathcal{A}\prime}$ by the adversary in round r'.

The extendable honest tree in Γ_{t+1} and honest tree in Γ'_{t+1} must be equivalent since for an addition of a *newly extendable chain* (**A2**(a), **A3**(a), **A4**) to the honest tree in FairPoS round r, the equivalent chain in PoS is added to the honest tree in round r'. ◀

▶ **Theorem 16. ((τ, s)-Chain growth in (d, δ, Δ)-FairPoS)** Let $\mathcal{A}$ be an an adaptive adversary against the protocol (d, δ, Δ)-FairPoS that corrupts up to $(1-\alpha)$ stake, where α be such that $\alpha(1-f)\Delta = (1+\epsilon)/2$ for active slot coefficient $f \in (0, 1]$ and some $\epsilon > 0$. Then the probability that $\mathcal{A}$ makes the protocol violate the chain growth property with parameters $s \geq 4\Delta$ and $\tau = c\alpha/4$ throughout a period of R slots, is no more than $\exp(-c\alpha s/(20\Delta) + lnR\Delta + O(1))$, where c denotes the constant $c := c(f, \Delta) = f(1-f)^{\Delta}$.

Proof. The proof of chain-growth in FairPoS closely follows that of Ouroborous Praos, as both analyses assume the dominant distribution (Definition 1) to model leader elections during protocol executions. Thus, we reproduce the main proof argument of Theorem 6 in [18] for the convenience of the reader, and refer to [18] for the derivation of the exact probability bounds, which are directly inferred from the dominant distribution.

Recall that the definition of chain growth requires that if the longest chain possessed by an honest party at the onset of some slot sl_1 is $\mathcal{C}_1$, and the longest chain possessed by a (potentially different) honest party at the onset of slot $\mathsf{sl}_2 \geq \mathsf{sl}_1 + s$ is $\mathcal{C}_2$, then $\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq \tau s$.

Let a uniquely honest slot (0) be Δ-right-isolated if it is followed with Δ consequent slots which are empty ($\perp$). Let $\mathcal{C}$ be a chain with a tip that is generated in a Δ-right-isolated uniquely honest slot. Then the next slot leader will necessarily consider $\mathcal{C}$ candidate for extension since

- Chain $\mathcal{C}$ must be have arrived in the view of all honest parties after δ slots in a δ-synchronous setting, where $\delta < \Delta$ (Equation (5)).
- Chain $\mathcal{C}$ must be extractable after Δ slots (Theorem 10).

Thus, in the view of all honest parties, chain $\mathcal{C}$ must be a candidate for extension according to the longest-extractable-chain rule (Equation (9)) in FairPoS.

Now, let $\hat{\mathsf{sl}}_1, ..., \hat{\mathsf{sl}}_h$ be the increasing sequence of all Δ-right-isolated uniquely honest slots among the slots in $T := \{\mathsf{sl}_1 + \Delta, \mathsf{sl}_1 + \Delta + 1, ..., \mathsf{sl}_2 - \Delta\}$. Observe that since $\hat{\mathsf{sl}}_1 \geq \mathsf{sl}_1 + \Delta$, the leader of $\hat{\mathsf{sl}}_1$ will append a block to a chain that is at least as long as $\mathcal{C}_1$, since $\mathcal{C}_1$ will be known to him and will be considered in the longest-extractable-chain selection rule. Therefore, the chain that the leader of $\hat{\mathsf{sl}}_1$ diffuses will be at least 1 block longer than $\mathcal{C}_1$. Analogously, the leader of every $\hat{\mathsf{sl}}_i$ will diffuse a chain that is at least 1 block longer than the chain diffused

by the leader of $\hat{\mathsf{sl}}_{i-1}$ since $\hat{\mathsf{sl}}_{i-1}$ is Δ-right-isolated. Finally, the chain diffused by the leader of $\hat{\mathsf{sl}}_h$ will be known to all parties at slot sl_2 and hence $\mathsf{len}(C2)$ will be at least as long as this chain. It follows that $\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq h$.

It remains to bound the number h of Δ-right-isolated uniquely honest slots among the slots with indices in T . To make our notation more flexible, let $H_T(x)$ denote the number of Δ-right-isolated uniquely honest slots among the slots from T in $x \in \{0,1,\perp\}^R$. From Theorem 6 in [18] we have for $c = f(1-\Delta)^\Delta$:

$$\Pr_{x \leftarrow \mathcal{D}_\alpha^f}[H_T(x) < c\alpha s/4] = \Delta \cdot e^{-\frac{c\alpha(s-3\Delta)}{20\Delta}}$$

Applying the union bound over R slots, we conclude that the probability that there is a chain growth violation with parameters s and $\tau = c\alpha/4$ is no more than

$$R\Delta \exp(-c\alpha(s-3\Delta)/(20\Delta)) = \exp(-c\alpha(s-3\Delta)/(20\Delta) + \ln R\Delta)$$

◀

▶ **Theorem 17. ((μ,k)-Chain quality in (d,δ,Δ)-FairPoS)** Let $\mathcal{A}$ be an adaptive adversary against the protocol (d,δ,Δ)-FairPoS that corrupts up to $(1-\alpha)$ stake, where α be such that $\alpha(1-f)\Delta = (1+\epsilon)/2$ for active slot coefficient $f \in (0,1]$ and some $\epsilon > 0$. Then the probability that $\mathcal{A}$ makes FairPoS violate the chain quality property with parameters k and $\mu = 1/k$ throughout a period of R slots, is no more than $\exp(\ln R - \Omega(k))$.

Proof. The proof of chain-growth in FairPoS closely follows that of Ouroborous Praos, as both analyses assume the dominant distribution (Definition 1) to model leader elections during protocol executions. Thus, for the convenience of the reader, we restate and adapt Lemma 4 in [18] and its main proof argument below, from which Theorem 17 follows.

▶ **Lemma 22.** *(Adapted from Lemma 4 in [18]) Let $k, \Delta \in \mathbb{N}$ and $\epsilon \in (0,1)$. Let A be an adaptive adversary against the protocol (d,δ,Δ)-FairPoS corrupting up to $(1-\alpha)$ stake for some $\alpha > 0$ satisfying $\alpha(1-f)^\Delta = (1+\epsilon)/2$. Let $B_1, ..., B_k$ be a sequence of consecutive blocks in a chain $\mathcal{C}$ possessed by an honest party. Then at least one block B_i was created in a Δ-right-isolated uniquely honest slot, except with probability $\exp(-\Omega(k))$.*

For convenience, let us call a slot good if it is Δ-right-isolated uniquely honest, and bad if it is neither empty nor good. Moreover, we call a block good (resp. bad) if it comes from a good (resp. bad) slot.

Towards contradiction, assume that all blocks $B_1, ..., B_k$ are bad. Let G_1 denote the latest good block preceding B_1 in $\mathcal{C}$, and G_2 the earliest good block appearing after B_k in $\mathcal{C}$ (or the last block of $\mathcal{C}$, if there is no good one). Note that all blocks between G_1 and G_2 are bad.

Let $\hat{\mathsf{sl}}_1$ (resp. $\hat{\mathsf{sl}}_2$) denote the good slot in which G_1 (resp. G_2) was created (if G_2 is not good, let $\hat{\mathsf{sl}}_2$ be the current slot). Denote by T the continuous sequence of slots between $\hat{\mathsf{sl}}_1$ and $\hat{\mathsf{sl}}_2$, excluding $\hat{\mathsf{sl}}_1$ and including $\hat{\mathsf{sl}}_2$. As we argued in the proof of Theorem 16, in each good slot in T the (unique) leader creates a block that has depth increased by at least 1 compared to the block from the previous good slot. Therefore, we have $\mathsf{depth}(G_2) \geq \mathsf{depth}(G_1) + g$, where g is the number of good slots in T . However, in chain $\mathcal{C}$ we have $\mathsf{depth}(G_2) \leq \mathsf{depth}(G_1) + b$, where b is the number of bad slots in the same sequence T . These two conditions can only be satisfied at the same time if $g \leq b$, we will now show that this is very unlikely.

We can bound $\Pr_{x \leftarrow \$ \mathcal{D}_\alpha^f}[g(x) \leq b(x)]$ as follows: we know that $\alpha(1-f)^\Delta = (1+\epsilon)/2$ and this implies that good slots are sampled with higher probability than bad slots. Therefore, $\Pr_{x \leftarrow \$ \mathcal{D}_\alpha^f}[g(x) \leq b(x)]$ falls exponentially with k. Lemma 22 directly implies Theorem 17. ◀

▶ **Lemma 18. (Input fairness from CG, CP and CQ in (d, δ, Δ)-FairPoS)** Input fairness is implied in an execution of (d, δ, Δ)-FairPoS, in which (τ, d)-chain growth, $(d\tau(\tau - \delta/(\Delta-\delta))-1)$-common prefix and $(1/(D+1), D+1)$-chain quality hold, where $D = d\Delta/(\Delta-\delta)$

Proof. The proof requires us to infer input fairness from chain growth, common-prefix and chain quality parameters as stated in the Lemma 18.

More informally, we frame the proof obligation as follows. At the onset of a given slot t, we are given a time budget of $\sim d$ slots, and must show that chain-growth will result in the sufficient growth in the length of the chain possessed by any honest party, such that for any encrypted input inserted at a block in slot t, it will reach the k-common-prefix within the given time budget, unless it becomes part of an abandoned branch.

Let $\mathcal{C}$ be the chain extended by an honest party at honest slot t. For simplicity, let us first assume that all slots are uniquely honest. An adversary begins the extraction of $\mathsf{id} = \mathcal{C}.\mathsf{tip}$ at the onset of its generation. Then, there remains $d-1$ slots between the encryption of the input at slot t and its decryption, since the input at t is encrypted with the parent block as session id. A chain growth rate of τ implies that the longest chain possessed by any honest party after $d-1$ time must increase by $k = \tau(d-1)$. Thus, in this naive scenario, input fairness would be implied by (τ, d)-CG and $(\tau(d-1))$-CP.

In the case that slots are *not all uniquely honest*, the adversary must be permitted a head-start in extracting the session key idk from any block: let us denote this the *time advantage*, which comes from two properties of the chain possessed by the honest user at the onset of slot t:

1. *Leading empty slots* between $\mathcal{C}.\mathsf{tip.sl}$ and current slot t. These empty slots represent a head-start the adversary has in decrypting inputs inserted in a child block of $h(\mathcal{C}.\mathsf{tip})$ generated at slot t.
2. *Leading adversarial block span* in $\mathcal{C}$ including $\mathcal{C}.\mathsf{tip}$, allowing it to generate blocks immediately after the extraction period d instead of waiting the full extraction schedule $D = d + n\delta$, as an honest party would. In the worst case, such a adversarial block span always leads up to $\mathcal{C}.\mathsf{tip}$, which is also adversarially generated, permitting the adversary an additional head-start in decrypting inputs.

For the (1) *leading empty slots*, the (τ, d)-CG property gives us the maximum number of slots in d, in which *no blocks* were generated for $\mathcal{C}$, namely $d(1-\tau)$. For the (2) *leading adversarial block span*, we quantify the *time advantage* gained from the leading adversarial block span as follows: for every D consecutive adversarial blocks, the adversary gains an additional $n\delta$ *time advantage*, since it does not have to wait the entire extraction schedule $D = d + n\delta$, where $n = D/\Delta = d/(\Delta - \delta)$. Note that we are granted $(1/(D+1), D+1)$-chain quality in Lemma 18, where $D = n\Delta = d\Delta/(n-\delta)$ as in Equation (5). We assume the worst case, namely, that all D slots leading up to t are indeed adversarial, and thus permit the adversary the maximum possible extraction time-advantage of $n\delta$ slots.

Thus the total time advantage in producing the adversarial block $\mathcal{C}.\mathsf{tip}$ obtained from (1) and (2) is given by $t_{\mathsf{adv}} = d(1-\tau) + n\delta = d(1 - \tau + \delta/(\Delta - \delta))$. Thus we require a *contracted* common prefix parameter $k = \tau(d - t_{\mathsf{adv}}) - 1$, in order for the honest input at slot t to either reach the common prefix before the adversary (with *time advantage*) can complete the key extraction of $\mathsf{id} = h(\mathcal{C}.\mathsf{tip})$, or not join the common prefix at all. Rewriting gives us $k = d\tau(\tau - \delta/(\Delta - \delta)) - 1$ ◀

▶ **Theorem 19. (Input fairness in (d, δ, Δ)-FairPoS)** Let $\mathcal{A}$ be an adaptive adversary against the protocol (d, δ, Δ)-FairPoS that corrupts up to $(1-\alpha)$ stake, where α be such

that $\alpha(1-f)\Delta = (1+\epsilon)/2$ for active slot coefficient $f \in (0,1]$ and some $\epsilon > 0$. Then, the probability that $\mathcal{A}$ makes the FairPoS violate the input fairness property falls exponentially with d.

Proof. With Lemma 18 we obtain input fairness from (τ, d)-CG and $(d\tau(\tau-\delta/(\Delta-\delta))-1)$-CP. Further, for an execution of (d, δ, Δ)-FairPoS for R slots,

- (τ, d)-CG with parameters $d \geq 4\Delta$ and $\tau = c\alpha/4$ is violated with probability no more than $\exp(-d\alpha c/(20\Delta) + lnR\Delta + O(1))$, where c denotes the constant $c := c(f, \Delta) = f(1-f)^{\Delta}$ (Theorem 16).
- k-CP with parameter $k = d\tau(\tau - \delta/(\Delta - \delta)) - 1$ is violated with probability no more than $\exp(\ln R + \Delta - \Omega(k))$ (Theorem 15).
- $(1/(D+1), D+1)$-CQ with parameter $D = d\Delta/(\Delta - \delta)$ is violated with probability no more than $\exp(\ln R + \Omega(D))$ (Theorem 17).

Probabilities above decline exponentially with increasing d. ◀