Arya Chain Backbone Proofs

Aarno Labs

December 9, 2024

1 Introduction

In this document we formalize the Proof of Burn (POB) mechanism that backs the Arya blockchain and we provide proofs that this mechanism enables consensus between agents in the presence of a bounded adversary. POB is a consensus mechanism in which participants must destroy currency in order to mine a block. Other blockchain consensus mechanisms include Proof of Work (POW), in which miners expend computational energy to mine a block, and Proof of Stake (POS), in which miners place some currency in a locked stake wallet that allows them to mine blocks at a rate proportional to their share of the total staked currency across the network. In this sense, a POS system is similar to an interest bearing bank account.

We employ POB rather than POW to enable our agent to run on resource constrained devices that may be unable to bear the computational burden of a POW system. We employ POB rather than POS because POB provides natural bounds on the length of a campaign as monetary supply monotonically decreases. POB also frees our agents from having to reason about the percentage of currency they must individually stake to ensure network-level security properties.

Iain Stewart was the first to describe a POB system [4]. In his design, Stewart draws an analogy between burning currency and purchasing POW mining hardware. Stewart's POB system gives participants the right to compete for blocks for some bounded amount of time after burning currency. This mining time is bounded to model both the fact that mining hardware increases in performance over time (therefore making miners with older hardware less likely to successfully solve POW puzzles faster than miners with newer hardware), and that hardware eventually breaks down. Stewart's algorithm was first implemented in Slimcoin [1], a cryptocurrency offering POB as a consensus option (in addition to POW and POS).

The Arya POB algorithm is significantly different from Stewart's. The first major difference is that burning currency provides a miner with the immediate right to mine in the Arya system, rather than buying a miner the ability to compete for blocks over some time period. Another major difference is that our POB mechanism uses round robin mining, meaning that (in a system comprised

of only honest participants) exactly one miner produces a block per round, and all participants can independently compute which participant's turn it is to mine in any given round. We are able to make these changes because the Arya chain is private and all participants are known.

In essence, Stewart's design attempts to solve a different problem than ours does. Stewart's POB algorithm facilitates a public blockchain for exchanging currency between an unknown number of participants, while our POB algorithm enables a private network of autonomous agents to securely make global decisions based on an assemblage of independently gathered local information. Given the vastly different motivations, the similarities in our designs end at the fact that they both enable consensus through the destruction of currency.

Karantias et al. [3] provide proofs for a POB scheme following Stewart's design. However, as our design is significantly different from Stewart's, these proofs do not apply to our system. These proofs are also not easily adapted to our system due to the intertwining of core consensus mechanisms and application level algorithms that result in the use of primitives and structures that simply have no analog in our system.

Instead, we base our proofs heavily on the work of Garay et al. [2] in which the authors extract the core consensus mechanism of Bitcoin into the *Bitcoin backbone*.[1] They prove that this minimal backbone, consisting only of block structure information without regard to transactions or other higher level structures, provides the core security and consensus guarantees of the Bitcoin protocol. They then use those proofs to prove properties of applications built on top of the Bitcoin backbone, such as Bitcoin transactions themselves, as well as byzantine agreement protocols.

We similarly extract the backbone of our POB system and prove security and consensus properties over it. We adopt, with slight modifications to support the differences in our systems, the formalism and ideal properties of [2]. As such, a strong understanding of the [2] formalism is required to understand the proofs contained in this document.

2 Formalism

This section details our formalism, relative to the formalism presented in [2]. We adopt all parts of the [2] model, with the exception of those we present in Section 2.2.

2.1 Important Properties Adopted from POW Model

This section highlights some of the important features we adopt from the Bitcoin POW model. This section is intended to serve only as a refresher for [2], but does not serve as a replacement for reading the paper!

[1] We build on the updated full version of the paper available at `https://eprint.iacr.org/2014/765.pdf`

Rounds The POB algorithm proceeds in rounds. In every round, every honest player runs a single iteration of the POB algorithm. The adversary $\mathcal{A}$ is activated last to give them the strongest advantage by virtue of observing all information from the round prior to acting.

Communication Communication between parties is reliable. Any message sent by P_s to P_d during round r will be received (placed on P_d's RECEIVE tape) by round $r+1$. Honest participants only communicate through DIFFUSE, which broadcasts messages to all participants. Corrupted participants and the adversary may send messages to individual participants.

Adversary We assume the existence of a polynomially bounded adversary $\mathcal{A}$. During its turn, $\mathcal{A}$ may corrupt an honest player P by sending P a CORRUPT message. From then on, every time it is P's turn, $\mathcal{A}$ is activated instead. $\mathcal{A}$ may corrupt zero, one, or more than one player per round. Corruption is permanent; there is no mechanism for uncorruption. Of course, the adversary may control corrupted parties to act honestly. Although this adversary is perhaps unrealistically strong in the real world, keep in mind it also captures effects such as poor or adversarial network connections to make up for our assumption that communication between parties is reliable.

Honest Majority Assumption We assume that a majority of players are honest. That is, the adversary may not corrupt a majority of players.

Fixed Participants We assume that participants are fixed at the start of the protocol. We plan to remove this assumption in the future. We discuss this more in Section 5.

Notation We adopt the notation from [2]. We provide a brief review of this notation in Table 1.

2.2 Differences From POW Model

This section details differences in our model. We present algorithmic differences in Section 3.

Synchronized Clock Assumption We assume that all parties have a synchronized clock. This allows parties to know what round they are currently in for the purpose of round robin mining. This assumption is not unreasonable as parties could, for example, use NTP to keep their clocks in sync. As discussed in Section 5, we plan to remove this assumption in the future.

$\mathcal{A}$	Adversary
P	Player
P_i	Player with id i
$\mathcal{C}$	Chain
B	Block
r	Round
n	Total number of players
t	Total number of corrupted players
ε	Empty block (often genesis)
λ	Security parameter
κ_s	Length of hash function output (we assume $\kappa_s = \Omega(\lambda)$)
$\text{len}(\mathcal{C})$	Returns the length of $\mathcal{C}$
$\text{head}(\mathcal{C})$	Returns the head of $\mathcal{C}$
$\mathcal{C}^{\lceil k}$	The prefix of $\mathcal{C}$ ending k blocks from $\text{head}(\mathcal{C})$
$\mathcal{C}_1 \preceq \mathcal{C}_2$	$\mathcal{C}_1$ is a prefix of $\mathcal{C}_2$

Table 1: Summary of notation we use from [2].

Notion of Funds We expand the backbone to include a notion of funds. Each player starts with s funds. The only mechanism for spending funds in the backbone is to mine, which costs b units of currency. There is no mechanism to transfer or otherwise increase a player's funds.

Fund Check External Function We introduce a fund check function $F(\mathcal{C}, P)$ that returns the funds P has on $\mathcal{C}$. We leave this function unspecified, but demonstrate that it is constructable by providing a possible construction:

$$F(\mathcal{C}, P) = s - |B_P| \cdot b$$

where B_P is the set of blocks P has mined in $\mathcal{C}$.

Mine Check External Function We introduce a mine check function $\mathcal{M}(r)$ that takes a round as input, and returns an index i that uniquely identifies which player should mine this round. Again, we leave this function unspecified but note that it is constructable (one simple construction uses modular arithmetic on r with the total number of participants as the modulus).

Agent Authentication We add the ability for players to authenticate each other through asymmetric cryptographic primitives and a mapping K from player id to public key for each player. This prevents $\mathcal{A}$ from impersonating honest players. We detail this change more in Section 3.

Unbounded Use Of $H(\cdot)$ We permit unbounded use of $H(\cdot)$. This is safe in our non-POW based system as it is not necessary to even the playing field between participants through hash restrictions. Our equivalent restriction to even the ability to mine is starting every player with s units of currency.

b	Funds burnt in each block
s	Starting funds for each player
K	Mapping from player ids to public keys
privkey	Private key
pubkey	Public key
κ_{pubkey}	Length of a public key (we assume $\kappa_{pubkey} = \Omega(\lambda)$)
σ	Cryptographic signature
$\mathsf{sign}(m, k)$	Sign message m with private key k
$\mathcal{M}(r)$	Returns the id of the player who should mine in round r
$F(\mathcal{C}, P)$	Returns funds P has on $\mathcal{C}$
$x :: x'$	x' is appended onto x

Table 2: Summary of additional notation we use.

All Rounds Uniquely Successful Or Unsuccessful By our algorithm definitions detailed in Section 3, we have no notion of a round that is successful, but not uniquely successful. All rounds are by definition either uniquely successful or unsuccessful. As such, we do not differentiate between (or even use) $X(\cdot)$ and $Y(\cdot)$ in our proofs.

Additional Notation We provide a summary of the additional notation we use in Table 2.

3 Proof of Burn Backbone

In this section we present the POB backbone definitions and algorithms. The backbone describes the structure of the blockchain and the blocks within, as well as the underlying algorithms for achieving consensus among participants. The backbone is responsible for providing the security and consensus guarantees that blockchains strive to achieve. The backbone does not provide semantics for transactions or other higher level structures that may be found in the body of a block, but it does provide a foundation for a formalism of higher level structures.

For a more detailed explanation of the POW backbone, which this section builds heavily on, please see [2].

3.1 Algorithms

Here we explain the algorithms our proof of burn backbone uses. Since our POB model is not too dissimilar from the POW model, there is significant overlap between our algorithms and the algorithms presented in [2]. Therefore, we only present the differences between our sets of algorithms and elide that which is identical.

One difference is that we define a *block* as a tuple of the form $B = \langle s, x, i, \sigma \rangle$

where

$$s \in \{0,1\}^{\kappa_s}$$
$$x \in \{0,1\}^*$$
$$i \in [0,n)$$
$$\sigma \in \{0,1\}^*$$

We remove *ctr* from the block triple in [2], as we have no need for a counter because our POB scheme does not require a player to repeatedly modify a portion of the block for probabilistic generation (our system is not probabilistic). We also add i, which is a unique identifier per player, as well as σ, which is a cryptographic signature over $s :: x$. We add identifiers and cryptographic signatures to prevent the adversary from impersonating honest players. This is a key difference between our formalism and [2], in which players have no mechanism for authentication, thus enabling $\mathcal{A}$ to spoof messages from honest players.

To accommodate our different definition of a block, our block validation predicate is also different. We define $\mathsf{validblock}^K_{\mathcal{C}}(B)$ as

$$(F(\mathcal{C}, P_i) \geq b) \wedge i \in [0,n) \wedge \mathsf{verifysignature}(s :: x, K(i), \sigma)$$

where $K : [0,n) \to \{0,1\}^{\kappa_{pubkey}}$ is a mapping from player ids to public keys and $\mathcal{C}$ is the chain to verify player i's funds under. The first clause checks that player i has sufficient funds to mine B on chain $\mathcal{C}$. The second clause checks that i is well formed. The third clause checks the validity of the cryptographic signature σ of $s :: x$ (x appended to s) with respect to the public key $K(i)$.

The final notational difference is that our backbone uses only $H(\cdot)$ and does not use $G(\cdot)$. Additionally, our system puts no restrictions on the number of times a player can invoke $H(\cdot)$ in a round. There are no other differences in notation between our POB backbone and the POW backbone from [2].

Chain validation. We present the POB chain validation algorithm in Algorithm 1. Our validate function is very similar to the one found in [2], but we have adapted it to support our different block structure.

Chain comparison. Our chain comparison algorithm is identical to the one found in [2] (Algorithm 2 in their paper). Therefore, we elide it from this document.

Proof Of Burn. We present our proof of burn algorithm pob in Algorithm 2. This algorithm is the most significant difference between our POB based system and the POW based system presented in [2]. pob provides the mechanism that secures our backbone protocol. The pob function is parameterized by the hash function $H(\cdot)$, the mine selector function $\mathcal{M}(\cdot)$, the round r, the player's id i, and the player's private key *privkey*. pob takes as input a value x to be inserted into the chain, as well as the chain $\mathcal{C}$ to extend.

Algorithm 1 The *chain validation predicate*, parameterized by the public key mapping $K(\cdot)$, the hash function $H(\cdot)$, and the content validation predicate $V(\cdot)$. The input is the chain $\mathcal{C}$.

function validate($\mathcal{C}$)
  $b \leftarrow V(\mathbf{x}_{\mathcal{C}})$
  if $b \wedge (\mathcal{C} \neq \varepsilon)$ **then** ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
    $\langle s, x, i, \sigma \rangle \leftarrow \text{head}(\mathcal{C})$
    $s' \leftarrow H(s, x)$
    repeat
      $\langle s, x, i, \sigma \rangle \leftarrow \text{head}(\mathcal{C})$
      if $\mathsf{validblock}^{K}_{\mathcal{C}^{\lceil 1}}(\langle s, x, i, \sigma \rangle) \wedge (H(s, x) = s')$ **then**
        $s' \leftarrow s$ ▷ Retain hash value
        $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$ ▷ Remove head from $\mathcal{C}$
      else
        $b \leftarrow$ False
      end if
    until $(\mathcal{C} = \varepsilon) \vee \neg b$
  end if
  return b
end function

If player i is successful ($\mathcal{M}(r) = i$), then pob extends $\mathcal{C}$ with a new block B containing a reference to the previous block s, the value x, the player's id i, and a cryptographic signature of $s :: x$ proving that player i generated B. Note that pob does not insert into the chain any explicit fund deductions from player i, as $F(\cdot, \cdot)$ can compute player mine deductions implicitly from the number of blocks each player has in the chain.

If player i is unsuccessful ($\mathcal{M}(r) \neq i$), then pob returns $\mathcal{C}$ unmodified.

Backbone protocol. Our backbone protocol algorithm is identical to the one in [2], but we replace their call to pow with a call to our pob function. Additionally, we modify any validate calls to call our validate function instead.

3.2 Desired backbone properties

In this section we formally define the two properties that we will prove in Section 4. These properties are very similar to their equivalents in [2], with some minor parameter and bound changes to reflect the differences in our systems. Let $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$ be the view of party P after the completion of an execution with environment $\mathcal{Z}$, running protocol Π, and adversary $\mathcal{A}$.

Definition 1 (Common Prefix Property)**.** The common prefix property Q_{cp} with parameter $k \in \mathbb{N}$ states that for any pair of honest players P_1, P_2, adopting the chains $\mathcal{C}_1, \mathcal{C}_2$ at rounds $r_1 \leq r_2$ in $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$ respectively, it holds that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$.

Algorithm 2 The *proof of burn* function, parameterized by the hash function $H(\cdot)$, the mine selector function $\mathcal{M}(\cdot)$, the round r, the player's id i, and the player's private key *privkey*. The input is $(x, \mathcal{C})$.

function $\mathsf{pob}(x, \mathcal{C})$
  if $\mathcal{C} = \varepsilon$ **then** ▷ Get last block hash
    $s \leftarrow 0$
  else
    $\langle s', x', i', \sigma' \rangle \leftarrow \text{head}(\mathcal{C})$
    $s \leftarrow H(s', x')$
  end if
  $B \leftarrow \varepsilon$
  if $\mathcal{M}(r) = i$ **then**
    $B \leftarrow \langle s, x, i, \mathsf{sign}(s :: x, privkey) \rangle$ ▷ Mine block
  end if
  $\mathcal{C} \leftarrow \mathcal{C} :: B$ ▷ Extend chain
  return $\mathcal{C}$
end function

This property states that all honest players achieve consensus on a common prefix of the chain. Therefore, all honest participants can assume that all other honest participants agree on chain state up until a block k from the end.

Definition 2 (Chain Growth Property)**.** The chain growth property Q_{cg} with parameter $k \in \mathbb{N}$ states that for any honest party P that has a chain $\mathcal{C}$ in $\text{VIEW}^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, it holds that after any k consecutive rounds it adopts a chain that is at least 1 block longer than $\mathcal{C}$.

This property states that the chain continues to grow as time goes on and does not get stuck. Together, the two properties ensure that even in the presence of a bounded adversary, progress is made and honest parties can still continually form consensus amongst themselves.

4 Proof Of Burn Backbone Proofs

In this section, we prove the common prefix and chain growth properties of the POB backbone. While all of the proofs in this section take inspiration from those in [2], two of them are identical or nearly identical to those found in [2]. We reproduce those two proofs here for completeness, but do not claim to take credit for them. Those proofs are the proofs of Lemma 3 and Theorem 2. Our proof of Lemma 3 is identical to the proof of Lemma 6 in [2], and our proof of Theorem 2 is nearly identical to the proof of the theorem by the same name in [2], with the exception that it uses our own lemmas (which are themselves original work) rather than the lemmas defined in [2]. In both cases, we have significantly compressed these proofs and implore the reader to see the original paper for more detail.

All other proofs in this section have significant differences to those in [2] and are our own original work.

4.1 Chain Growth

Lemma 1 (Honest Growth Lemma)**.** *If in round r, $\mathcal{M}(r) = i$ where P_i is an honest party, P_i has a chain $\mathcal{C}$ of length ℓ, and $F(\mathcal{C}, P_i) \geq b$, then in round $r+1$, all honest parties have a chain of length at least $\ell + 1$.*

Proof. By the backbone algorithm, P_i broadcasts a chain $\mathcal{C}_h$ of length $\ell + 1$ in round r. $\mathcal{A}$ also sends any combination of chains $\mathcal{C}_0 \ldots \mathcal{C}_m$ where $m \geq 0$ through any corrupted parties (if $m = 0$ or $t = 0$, $\mathcal{A}$ broadcasts no chains) to any combination of honest players.

Let R_j be the set of chains each participant P_j receives from round r, which are necessarily received by round $r + 1$. Although different players may receive different sets of chains, note that $\forall j,\ 0 \leq j < n.\ \mathcal{C}_h \in R_j$. For any honest player P_k with chain $\mathcal{C}_k$, let $\mathcal{C}'_k = \text{maxvalid}(\mathcal{C}_k, R^*_k)$ where $\mathcal{C}_k$ is the chain P_k had in round r and R^*_k represents the elements of R_k as a comma separated list of arguments. Since P_i is honest and $F(\mathcal{C}, P_i) \geq b$, $\mathcal{C}_h$ is a valid chain of length $\ell + 1$. Because maxvalid returns the longest valid chain and honest parties are guaranteed to receive broadcasts from all other honest parties, $\mathcal{C}'_k$ must have length at least $\ell + 1$. Because it is longer than $\mathcal{C}_k$, P_k will adopt $\mathcal{C}'_k$. □

Lemma 2 (Corrupted Growth Lemma)**.** *Let $\mathcal{M}(r) = i$ where P_i is a corrupted party. If in round r, an honest party P has a chain $\mathcal{C}$ of length ℓ, then in round $r + 1$, all honest parties have a chain of length at least ℓ.*

Proof. Let $\mathcal{A}$ send any combination of chains $\mathcal{C}_0 \ldots \mathcal{C}_m$ where $m \geq 0$ through any corrupted parties (if $m = 0$ or $t = 0$, $\mathcal{A}$ broadcasts no chains) to any combination of honest players in round r.

Let R_j be the set of chains each participant P_j receives. Note that different players may receive different sets of chains. For any honest player P_k with chain $\mathcal{C}_k$, let $\mathcal{C}'_k = \text{maxvalid}(\mathcal{C}_k, R^*_k)$ where $\mathcal{C}_k$ is the chain P_k had in round r and R^*_k represents the elements of R_k as a comma separated list of arguments. In order for $\text{len}(\mathcal{C}'_k) \geq \ell$, it must be true that at least one argument to maxvalid is at least as long as ℓ. More formally, it must be true that

$$(\text{len}(\mathcal{C}_k) \geq \ell) \vee (\exists c \in R_k.\ \text{len}(c) \geq \ell)\,.$$

Let r' be the round at which P_i adopted or extended $\mathcal{C}$. We consider two cases. First, consider the case that $r' < r$. In this case, P_i, being honest, has diffused $\mathcal{C}$ prior to round r, and all other honest participants have seen it by round $r' + 1$. Therefore, since P_k is honest and has seen $\mathcal{C}$, it must be the case that $\mathcal{C}_k$ is at least as long as $\mathcal{C}$, thus satisfying the first clause.

The second case is that $r' = r$. In this case, P_i diffused $\mathcal{C}$ in round r therefore $\mathcal{C} \in R_k$, thus satisfying the second clause. With both clauses satisfied, it must be true that $\text{len}(\mathcal{C}'_k) \geq \ell$. □

Definition 3 (Typical execution)**.** An execution is *typical* if for any chain $\mathcal{C}$ the following hold.

1. For all honest players P_i, $F(\mathcal{C}, P_i) \geq b$.
2. No insertions, copies, or predictions occurred (see Definition 8 in [2] for the exact definition of these terms).

Theorem 1 (Chain Growth)**.** *In a typical execution, the chain growth property (Definition 2) holds with parameter $k > t$.*

Proof. Assume that at round r, an honest player P has a chain of length ℓ. Let $S = \{s \mid r \leq s < r + k\}$ and $N = \{n \mid \forall s \in S.\ n = \mathcal{M}(s)\}$. By the round robin nature of the burning algorithm and the fact that $t < k$, there must exist an $h \in N$ such that P_h is honest. Let round $r_h \in S$ be the round in which P_h was selected to burn a block. Additionally, let $\mathcal{C}_h$ be the chain P_h has at round r_h. By the typical execution hypothesis, it holds that $F(\mathcal{C}_h, P_h) \geq b$. By applying the honest growth lemma (Lemma 1), we know that if P had a chain of length ℓ' on round r_h, then on round $r_h + 1$, P had a chain of at least length $\ell' + 1$.

For all other rounds $s \in S$ where $s \neq r_h$, either the honest growth lemma (Lemma 1) or the corrupted growth lemma (Lemma 2) applies. Since neither of these lemmas can shorten the length P's chain, it is clear that after k rounds, (on round $r + k + 1$), P has a chain of at least length $\ell + 1$. □

4.2 Common Prefix

Lemma 3 (Uniqueness Lemma)**.** *Suppose the k-th block B of a chain $\mathcal{C}$ was computed by an honest party in a uniquely successful round. Then the k-th block of a chain $\mathcal{C}'$ either is B or has been computed by the adversary.*

Proof. This proof is identical to the proof of Lemma 6 in [2]. □

Lemma 4 (Bounded Adversarial Blocks Lemma)**.** *In typical execution, for any chain $\mathcal{C}$, $|\beta_{\mathcal{A}}| \leq \left\lfloor \frac{t \cdot s}{b} \right\rfloor$ where $\beta_{\mathcal{A}}$ is the set of blocks mined by $\mathcal{A}$ in $\mathcal{C}$.*

Proof. Assume towards contradiction that there exists a chain $\mathcal{C}$ such that $|\beta_{\mathcal{A}}| > \left\lfloor \frac{t \cdot s}{b} \right\rfloor$. Then the total amount of funds burned across all blocks in $\beta_{\mathcal{A}}$ is $|\beta_{\mathcal{A}}| \cdot b$. Let f be the maximum amount of funds available to $\mathcal{A}$ at any point in the construction of $\mathcal{C}$. In order to build $|\beta_{\mathcal{A}}|$ legal blocks, it must be the case that

$$f \geq |\beta_{\mathcal{A}}| \cdot b$$

We conservatively set f to be equal to all corrupted parties having their starting amount of funds. As there is no mechanism by which a party can acquire more funds, this is the maximum amount of funds any party can ever have.

$$t \cdot s \geq |\beta_{\mathcal{A}}| \cdot b$$

Division by b, followed by taking the floor of both sides (which has no effect on the right hand side as $|\beta_{\mathcal{A}}| \in \mathbb{N}$) yields

$$\left\lfloor \frac{t \cdot s}{b} \right\rfloor \geq |\beta_{\mathcal{A}}|,$$

which contradicts our previous assumption that $|\beta_{\mathcal{A}}| > \left\lfloor \frac{t \cdot s}{b} \right\rfloor$. □

Lemma 5 (Common Prefix Lemma)**.** *Assume a typical execution and consider two chains $\mathcal{C}_1$ and $\mathcal{C}_2$. If $\mathcal{C}_1$ is adopted by an honest party at round r and $\mathcal{C}_2$ is either adopted by an honest party at round r or diffused at round r and has* $\text{len}(\mathcal{C}_2) \geq \text{len}(\mathcal{C}_1)$*, then $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil k} \preceq \mathcal{C}_1$ for $k > 2 \cdot \left\lfloor \frac{t \cdot s}{b} \right\rfloor$.*

Proof. Assume toward contradiction an execution in which the assumptions of the lemma hold, but either $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2$ or $\mathcal{C}_2^{\lceil k} \not\preceq \mathcal{C}_1$ for some $k > 2 \cdot \left\lfloor \frac{t \cdot s}{b} \right\rfloor$. We define the syntax $B_{i,j}$ to refer to the j-th block in chain $\mathcal{C}_i$. Let B^* be the last block computed by an honest party in the common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. If no such block exists, let $B^* = \varepsilon$. Additionally, let p^* be the position that B^* occupies in both $\mathcal{C}_1$ and $\mathcal{C}_2$.

Now, let

$$S_1 = \{B_{i,j} \mid i = 1 \wedge p^* < j < \text{len}(\mathcal{C}_1)\}$$
$$S_2 = \{B_{i,j} \mid i = 2 \wedge p^* < j < \text{len}(\mathcal{C}_2)\}$$

We additionally define the injective function $f : S_1 \to S_2$ as

$$f(B_{1,j}) = B_{2,j}$$

to build a mapping between S_1 and S_2 based on the position of blocks in $\mathcal{C}_1$ and $\mathcal{C}_2$. Lastly, let the function $\mathcal{H}(B)$ evaluate to *true* if and only if B was mined by an honest participant.

Now, we claim that for all $B \in S_1$, it must be true that

$$(\mathcal{H}(B) \oplus \mathcal{H}(f(B))) \vee \neg(\mathcal{H}(B) \vee (\mathcal{H}(f(B)))).$$

This is trivially true if B lies in the common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$ (B and $f(B)$ must be adversarial by the definition of B^*), and follows directly by the uniqueness lemma (Lemma 3) for the rest.

Because at least one block in each position between p^* and $\text{len}(\mathcal{C}_1)$ from $\mathcal{C}_1$ or $\mathcal{C}_2$ is adversarial, it follows that in at least one chain, half or more blocks from B^* to the block at position $\text{len}(\mathcal{C}_1)$ on that chain are adversarial. Let this chain be $\mathcal{C}'$. In the event that both chains satisfy the above criteria, let—without loss of generality—$\mathcal{C}' = \mathcal{C}_1$.

Let $B_{\mathcal{A}}$ be the set of adversarial blocks in $\mathcal{C}'$. We know that

$$|B_{\mathcal{A}}| \geq \frac{\text{len}(\mathcal{C}_1) - p^*}{2}$$

By the definition of p^*, we know that $k \leq \text{len}(\mathcal{C}_1) - p^*$.

$$|B_{\mathcal{A}}| \geq \frac{k}{2}$$

From the lemma statement, we also know that $k > 2 \cdot \lfloor \frac{t \cdot s}{b} \rfloor$. Therefore,

$$|B_{\mathcal{A}}| > \left\lfloor \frac{t \cdot s}{b} \right\rfloor.$$

However, this violates the bounded adversarial blocks lemma (Lemma 4), thus contradicting the assumption we made at the beginning of the proof, which demonstrates that it must be the case that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil k} \preceq \mathcal{C}_1$. □

Theorem 2 (Common Prefix)**.** *In a typical execution the common prefix property (Definition 1) holds with parameter* $k > 2 \cdot \lfloor \frac{t \cdot s}{b} \rfloor$.

Proof. Consider chains $\mathcal{C}_1$ and $\mathcal{C}_2$ adopted by parties P_1, P_2, and corresponding rounds r_1 and r_2, in violation of the common prefix property. Now, let r be the smallest round $r_1 \leq r \leq r_2$ such that an honest party P_2' adopts a chain $\mathcal{C}_2'$ such that $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2'$.

We consider two cases. First, in case $r = r_1$, at round r two honest parties have adopted chains $\mathcal{C}_1$ and $\mathcal{C}_2'$ for which it holds that $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2'$. This violates the Common Prefix Lemma (Lemma 5) and we obtain a contradiction.

Second, consider the case $r_1 < r$. Let $\mathcal{C}_1'$ be the chain P_2' adopted at round $r - 1$. We claim that

$$\left(\mathcal{C}_2'^{\lceil k} \preceq \mathcal{C}_1'\right) \vee \left(\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_1'\right) \vee \left(\text{len}(\mathcal{C}_2'^{\lceil k}) \geq \text{len}(\mathcal{C}_1^{\lceil k})\right) \rightarrow \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2'^{\lceil k}$$

The conclusion implies that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2'$, contradicting the definition of r. Thus, to finish the proof, we show the truth of each of the hypotheses and the truth of the implication.

The first hypothesis follows from the Common Prefix Lemma (Lemma 5). This is because $\mathcal{C}_2'$ was diffused at round $r-1$ and $\text{len}(\mathcal{C}_2') > \text{len}(\mathcal{C}_1')$. The second is immediate from the definition of r. The third follows from $\text{len}(\mathcal{C}_2') \geq \text{len}(\mathcal{C}_1)$, which holds because $\mathcal{C}_2'$ was preferred by P_2' to $\mathcal{C}_1$ at round r. Finally, to see the implication, note that both $\mathcal{C}_2'^{\lceil k}$ and $\mathcal{C}_1^{\lceil k}$ are prefixes of $\mathcal{C}_1'$, so the smallest must be contained in the longest. □

5 Future Work

Although our proofs ensure that progress is made, they make no claims as to the quality of that progress. To that end, we would like to prove a theorem similar to the Chain Quality Property in [2]. This property would put a bound on adversarial blocks in our chain. It is intuitively clear that such a bound exists due to the fund cap on players and the fact that these funds monotonically

decrease, but we would like to formalize this bound. We have so far considered this to be lower priority than the other two major properties we prove because at this phase in the program there are no adversaries that can take over an agent and begin inserting adversarial data into the blockchain.

After adding a chain quality proof, we would like to add protocol layer proofs that build on our backbone proofs to both give formal semantics to the transactions in the Arya chain, and also to verify that the effects of those transactions are secure. Additionally, we would like to verify the partition resilience properties of the Arya chain. Some of those (such as that consensus can still be reached by players on either side of the partition amongst themselves) trivially fall out of the backbone proofs already in this paper. However, we would also like to verify that our partition resolution merge algorithm is sound.

Lastly, we would like to loosen some of the assumptions we make. Most notably, we would like to eliminate the assumption that players are fixed. This assumption is currently necessary because without a formalism for transactions, there is no mechanism by which new players can be introduced.

We would also like to eliminate the assumption that player clocks are synchronized. This seems doable to us, but at the expense of less clean proofs and worse bounds. Still, it is an important assumption to eliminate.

References

[1] Slimcoin. `http://slimco.in/`.

[2] 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.

[3] Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros. Proof-of-burn. Cryptology ePrint Archive, Report 2019/1096, 2019. `https://eprint.iacr.org/2019/1096`.

[4] Iain Stewart. Proof of burn, 2012. `https://en.bitcoin.it/wiki/Proof_of_burn`.