Player-Replaceability and Forensic Support are Two Sides of the Same (Crypto) Coin

Peiyao Sheng[1], Gerui Wang[1], Kartik Nayak[2], Sreeram Kannan[3], and Pramod Viswanath[1,4]

[1] University of Illinois at Urbana-Champaign, IL
{psheng2, geruiw2}@illinois.edu
[2] Duke University, NC
kartik@cs.duke.edu
[3] University of Washington at Seattle, WA
ksreeram@ece.uw.edu
[4] Princeton University, NJ
pramodv@princeton.edu

Abstract. *Player-replaceability* is a property of a blockchain protocol that ensures every step of the protocol is executed by an unpredictably random (small) set of players; this guarantees security against a fully adaptive adversary and is a crucial property in building permissionless blockchains. *Forensic Support* is a property of a blockchain protocol that provides the ability, with cryptographic integrity, to identify malicious parties when there is a safety violation; this provides the ability to enforce punishments for adversarial behavior and is a crucial component of incentive mechanism designs for blockchains. Player-replaceability and strong forensic support are both desirable properties, yet, none of the existing blockchain protocols have *both* properties. Our main result is to construct a new BFT protocol that is player-replaceable *and* has maximum forensic support. The key invention is the notion of a "transition certificate", without which we show that natural adaptations of extant BFT and longest chain protocols do not lead to the desired goal of simultaneous player-replaceability and forensic support.

1 Introduction

Byzantine fault tolerant state machine replication (BFT SMR) protocols allow a group of parties to agree on a common sequence of values submitted by external clients. The core security guarantee provided by BFT SMR is that as long as a certain fraction of parties are honest, i.e., follow the protocol, then these parties achieve consensus with respect to a time evolving ledger regardless of the actions of the remaining Byzantine parties that deviate from the protocol. Of particular interest are secure and efficient BFT SMR protocols: efficiency is measured in terms of commit latency and communication complexity [14,26,4,1,37,25], and security is measured by tolerating the maximum number of Byzantine parties under various network and cryptographic assumptions [16,17,28,14,15,2,22].

Security guarantee of BFT protocols is one-sided, addressing the scenario when the number of Byzantine parties is less than a certain threshold. *Forensic support* addresses the other side: what happens when the number of Byzantine parties exceeds the allowable threshold? Several recent works focus on designing secure BFT protocols that also have an additional goal of accountability, i.e., the ability to detect faulty behavior through an irrefutable proof upon security violation [12,7,30,31,36]. A recent work [35] has formally defined forensic support of BFT protocols, providing a unified framework to compare and contrast different designs; [35] provides a detailed analysis of canonical BFT protocols (e.g., PBFT [9,8], HotStuff [37], VABA [3], and Algorand [18,10,11]) with respect to their support for forensics on detecting Byzantine behavior. A key takeaway from this work is that, while forensic support depends heavily on the implementation details of the protocol, deterministically secure protocols with $\mathsf{poly}(n)$ communication complexity (here, n is the number of parties in the protocol) have protocol variants with maximum forensic support, i.e., the maximum number of Byzantine parties can be identified irrefutably using simply the transcript available at one of the honest parties.

An entirely different aspect of BFT protocols has emerged with the advent of blockchains and the desire to support the participation of a very large number of players ("permissionless" participation): communication-efficiency (i.e., have sub-quadratic communication complexity) combined with security against a fully adaptive adversary; e.g., Algorand [10] and Ouroboros Praos [13]. Such protocols are commonly referred to as "player-replaceable" since they rely on verifiably selecting small subgroups of truly random parties in each round, thus achieving adaptive security and communication efficiency. Of specific interest are secure blockchain protocols that offer both desired properties: forensic support *and* player-replaceability. We begin by observing that no extant blockchain protocols offer both properties. For instance, HotStuff excels in efficiency and forensic support but is not player-replaceable; Algorand is player-replaceable but has non-existent forensic support [35]. Indeed, no extant blockchain protocol appears to have both player-replaceability and strong forensic support.

Could this status-quo be not coincidental? Player replaceability implies security even though different players are corrupted at different rounds of the protocol; perhaps this strong property inherently rules out the ability to identify malicious parties when security is violated? Understanding the core relationships between player-replaceability and forensic support properties of BFT protocols is the main goal of this paper. Our main result is dissenting: we construct a new BFT protocol that is player-replaceable *and* has strong forensic support (i.e., detecting the maximum number of Byzantine nodes with the minimum number of honest transcripts).

In this paper, we divide our investigation based on two stylistically different families of player-replaceable protocols: BFT protocols and longest-chain protocols. A summary of our results is presented in Table 1.

Main result: a player-replaceable BFT protocol with strong forensic support. We first present a novel player-replaceable BFT protocol with strong

Table 1: Comparison of forensic properties among different protocols

	Protocol	Byzantine threshold (t)	Player replaceability	Forensic support (d)	
BFT Protocols	Algorand [10]	$n/3$	Yes	None	0
	HotStuff [37]		No	Strong	$\lceil n/3 \rceil$
	Player-replaceable HotStuff (§3.1)		Yes	None	0
Longest-chain Protocols	OBFT [23]	$n/3$	No	Moderate	$n-2f$
	Ouroboros [24]	$n/2$[①]	No		$< (n-2f)\kappa/n$
	Ouroboros Praos [13]		Yes		$< (n-2f)\kappa/n - T(n,\kappa)$[②]
Our Result	**Algorithm 3**	$n/3$	**Yes**	**Strong**	$\lceil \lambda/3 \rceil$[③]

[①] In Ouroboros and Praos, forensic support is discussed even when $f < t$.
[②] κ is a parameter for longest-chain confirmation, $T(n, \kappa) > 0$.
[③] λ is the expected size of committee.

forensic support in the partially synchronous setting, where "strong" implies that the most number of Byzantine nodes can be detected with the least number of honest transcripts. In particular, we show that when the total fraction of Byzantine parties is less than $(1-\epsilon)2/3$ (ϵ is a positive constant) and the expected committee size is λ, our forensic protocol can detect at least $\lceil \lambda/3 \rceil$ Byzantine parties when safety violations occur. Due to idiosyncratic constraints imposed by player-replaceability, traditional analyses of forensic support [35] do not immediately apply. For instance, a core component of the forensic support analysis of existing BFT protocols relies on identifying parties that perform two or more actions that are incompatible with each other with respect to the protocol specification [35]. The forensic protocol determines appropriate quorums and uses quorum intersection arguments to identify culpable parties. However, with player replaceability, when n is large, *it is extremely unlikely that the same player will be selected twice; thus access to incompatible actions performed by the same player, especially across different rounds (or views) of the protocol, may be unavailable.* One of our key innovations is the notion of "transition certificates", maintained and shared by each party in each round – this ensures that if Byzantine parties vote incorrectly in a round resulting in a safety violation, there is sufficient information to detect misbehavior.

Forensic analysis for longest-chain protocols. Bitcoin, the prototypical longest-chain protocol, demonstrates ideal player-replaceability: not only is the next proposer not predictable, but also the mined block safe from any later tampering. Indeed, the longest chain rule has inspired both BFT and proof-of-stake (PoS) based player-replaceable blockchain protocols, e.g., Ouroboros family, including Ouroboros BFT (OBFT), Ouroboros, and Ouroboros Praos (referred to simply as Praos in this paper). We first prove that OBFT, as a binary consensus protocol, can hold $n-2f$ replicas culpable where n and f represent the total number of replicas and the number of actual faults respectively. On the other hand, the number of Byzantine parties detected in Ouroboros and Praos

is bounded by $2\kappa(n-2f)/n$ where κ is the confirmation depth. The bound is a result of the randomized leader election process in the two protocols. It is noteworthy that there is no forensic support when $f > n/2$. However, even if Ouroboros and Praos have Byzantine threshold $t < n/2$, the random election results in possible executions with safety violation when $f < t$. We observe that the safety violation of longest chain protocols can be identified when an honest replica finalizes two conflicting blocks and observes more than one longest chain. In the case of Ouroboros family, this is used to identify malicious leaders who propose more than one valid block in the same round.

Outline. We describe the security model and definitions in §2. §3 contains our main result: the construction of a new BFT protocol endowed with both player-replaceability and strong forensic support. Longest-chain protocols are naturally aligned with the player-replaceability property and we explore their forensic support properties in §4. §5 concludes the paper with a discussion of the relationship between player-replaceability and forensic support. The topic of this paper has not been broached in any prior work, to the best of the authors' knowledge. Works, other than those already referenced, are tangential to the core content here; the connections are discussed for completeness in Appendix A. The practicality and parameter choices of our protocol are formally described in Appendix B.

2 Model and Definitions

We consider a network with n nodes interacting via all-to-all communication. Prior to the protocol execution, each node generates its key pair honestly and sends its public key to all other nodes. The adversary can adaptively corrupt nodes at any time during the protocol execution after the trusted setup. Nodes that are never corrupted are referred to as honest. The total number of nodes corrupted by the adversary in an execution is denoted as f. The maximum number of corrupted nodes the protocols can tolerate is denoted as t.

Network setting. We consider synchronous and partially synchronous network settings. In a synchronous protocol, a message sent at time T by a sender node is guaranteed to arrive at the receiver node by time $T + \Delta$, where Δ is a bounded network delay. In a partially synchronous protocol, there exists an unknown global stabilization time (GST), after which all transmissions between two honest nodes arrive within a bounded network delay Δ [16].

Blockchains and state machine replication (SMR). The goal of blockchains (state machine replication [34]) is to build a public ledger that provides clients a totally ordered sequence of transactions. The key security properties a blockchain protocol should provide are those of safety and liveness. Safety: no two honest nodes finalize two different blocks at the same position in the ledger. Liveness: every valid transaction is eventually finalized by every honest node. We use blockchain and SMR interchangeably and refer to nodes that run blockchain protocols as "replicas" or "players".

Player-replaceability is a property of blockchain protocols. As presented by Chen and Micali [10], a protocol is player-replaceable if each step of the protocol execution is conducted by an independently and randomly selected subset of players. A player-replaceable protocol achieves both adaptivity and communication efficiency since the adaptive adversary cannot predict the committee membership ahead of time and only a subset of parties (typically sublinear) need to communicate in each round (hence the communication complexity is subquadratic).

Forensic support for blockchains. The notion of forensic support for Byzantine Agreement (BA) was introduced by [35]. Forensic support refers to the ability to identify misbehaving replicas whenever there is a safety violation (two honest replicas finalize different blocks at the same position). The number of replicas that can be held culpable when $t < f \leq m$ is captured by the parameter d. Here, m denotes the bound on Byzantine replicas under which the forensic support can be provided. In BA, transcripts of honest parties are needed to obtain irrefutable proof of culprits *after* clients detect a safety violation. The number of transcripts to decide culpability of replicas is denoted by k. In the blockchain setting, we adapt the definition of k to denote the number of transcripts required to detect safety violations and construct the culpability proof.

Definition 1. ***(m, k, d)-Forensic Support***. *[35] If $t < f \leq m$ and there is a safety violation, then using the transcripts of all messages received from k honest replicas during the protocol, a client can provide an irrefutable proof of culpability of at least d Byzantine replicas.*

Cryptographic primitives. All protocols we discuss in this paper use collision resistant cryptographic hash functions and digital signatures (that are adaptively secure for achieving player-replaceability). $\langle x \rangle$ denotes the signed message x. The intersection of two aggregated signatures refers to the set of replicas who sign both messages. We use verifiable random functions (VRFs) [27] to choose a random subset of replicas to be the leader or committee in a round. In our model, VRF has two functions: $\texttt{VRF}(x)$ and $\texttt{VerifyVRF}_{pk}(msg, x)$. $\texttt{VRF}(x)$ returns two values: a $hash$ and a proof π. The hash is a $HASHLEN$-bit value, normalized by $2^{HASHLEN}$, i.e., $hash \in [0, 1)$. It is uniquely determined by sk and x, and indistinguishable from a random value to anyone that does not know sk. The proof π enables anyone that knows pk to verify the value by $\texttt{VerifyVRF}_{pk}$. In our protocol, $(hash, \pi)$ is always appended to a message and hence not explicitly specified. $\texttt{VerifyVRF}_{pk}(msg, x)$ verifies that $hash$ is the correct value computed from x by using π. The appended $hash$ value is denoted by $msg.vrf$. We omit the notation of sk, pk and the appended $(hash, \pi)$ when the context is clear. In some of the protocols, VRFs may be used to elect leaders and/or committees to obtain player-replaceability, i.e., every step of the protocol is executed by a potentially new set of parties. This approach was pioneered in [10] to construct protocols secure under fully adaptive adversaries that are also efficient, i.e., subquadratic communication complexity.

3 Main Results

Our main result is the first player-replaceable BFT protocol that has strong (maximum) forensic support. In particular, we construct a partially synchronous, player-replaceable BFT protocol (§3.2) that tolerates $t = (1-\epsilon)n/3$ Byzantine faults for safety and liveness while providing forensic support with $t < f \leq (1-\epsilon)2n/3$, where ϵ is a positive constant.

We start with a warm-up protocol (§3.1) that makes HotStuff [37] player-replaceable using ideas in Algorand [10] but it is shown to lack forensic support. Inspired by the learnings, we design our protocol which is equipped with an additional step, called *certified transition*, to obtain both player-replaceability and strong forensic support. We provide the forensic protocol and formally prove that when there is a safety violation, the protocol can hold at least $\lceil \lambda/3 \rceil$ Byzantine replicas culpable with irrefutable proof (§3.3).

3.1 Warmup: HotStuff Made Player-replaceable

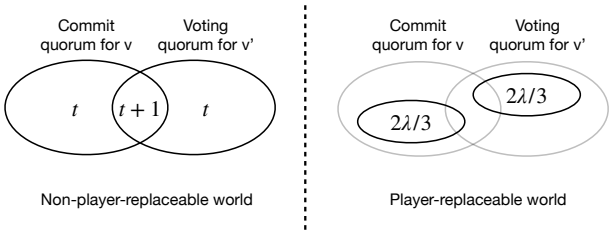

Fig. 1: Comparison of non-player-replaceable and player-replaceable worlds.

The most intuitive approach to obtain both properties is to start with a protocol with strong forensic support and make it player-replaceable. For instance, we can start with HotStuff [37] (or Tendermint [6]) and make it player-replaceable using techniques from Algorand [10]. Specifically, in each round of voting, replicas perform cryptographic sortition to determine whether they are eligible to vote in the current round. Such a sortition is publicly verifiable and produces a randomly and independently selected voting committee in each round.

While the use of sortition enables player-replaceability, the protocol still falls short of providing forensic support. In HotStuff with forensic support, when there is a safety violation, the forensic support protocol can always map it back to a set of culpable parties that have performed (at least) two contradictory actions, thus not following the protocol. Such behavior is observable even in non-trivial violations that happen across rounds. In particular, in a non-player-replaceable world, since a majority of honest parties guard the safety of the commit, there does not exist enough votes for a different value when $f \leq t$. When $f > t$, it has been shown in [35] that we can detect $t+1$ such parties whenever there is a safety violation. Intuitively, the idea uses a quorum intersection between the $2t+1$ parties that send a commit message for the first committed value v and a specific set of $2t+1$ parties that vote on a different value v' later (cf. left figure

in Figure 1), where the voting quorum for a different value consists of honest parties who hold stale states, and Byzantine parties trying to violate the safety.

Unfortunately, with player-replaceable protocols, such an argument does not apply. Since only a small λ-sized fraction (λ is a security parameter) of parties are chosen each time, it is highly likely that a replica is elected in the committee only once. As is shown in Figure 1 on the right, the quorums from the committee in two distinct rounds are mutually exclusive with high probability, due to which the continuity of participation of a single replica is lost. In other words, we cannot distinguish between Byzantine replicas from honest replicas with stale states since in any of the cases, since Byzantine replicas can deliberately mimic the behavior of honest replicas who suffer long message delays. By contradiction, suppose forensic support is possible in some case, i.e., some Byzantine replicas voting for a different value are made accountable, there must exist a corresponding scenario where these accountable replicas are honest (and their behaviors are simulated by Byzantine parties in the first case). Thus, we may not have any forensic support under this circumstance when $f > t$ while still being safe and live when $f \leq t$.

3.2 Construction of a Player-Replaceable BFT Protocol with Strong Forensic Support

Intuition. To address the above concern, the intuition of our protocol is to enforce replicas to wait for enough messages (2/3 of the committee size) to form a *transition certificate (TC)* of each round r before entering round $r + 1$. Waiting for messages of round r ensures that a replica's state is up-to-date at the beginning of round $r + 1$, then the scenario where honest parties are blamed due to message delays are no longer possible. Therefore, no honest replicas have stale states and we can distinguish honest replicas who suffer long message delays from Byzantine replicas, and have strong forensic support. Starting from this intuition, we design the protocol with safety and liveness properties as well as strong forensic support.

Protocol overview. The protocol proceeds in a sequence of consecutive rounds where each round lasts for at least 4Δ time (as measured by each replica's own clock). In each round, a set of leaders and a committee will be self-selected from all replicas using cryptographic sortition. The role of a leader is to collect *votes* from committee members and generate a *quorum certificate (QC)* from the votes. It then proposes a *block* that contains the QC to all replicas. The role of a committee member is to wait for the leader's proposal and, if it is valid, to vote for it. We first describe the sortition process used for election, then define the data structures used in the protocol, and finally present the protocol.

Cryptographic sortition. We use cryptographic sortition to choose a random subset of parties as leaders or committee members, by using VRF [27]. A replica determines its eligibility to be the next leader or the committee member by computing VRF from the random seed, the round, and the role ("*leader*" or "*committee*"), i.e., $\texttt{VRF}(seed||curRound||role)$, $role \in \{\text{"}leader\text{"}, \text{"}committee\text{"}\}$.

If the VRF hash value is smaller than a threshold, the replica is eligible and when it fulfills its role by broadcasting a message, it accompanies the VRF output (hash value and proof) thus allowing other replicas to verify its eligibility. For a message m, we denote the accompanied VRF hash value as $m.vrf$. The threshold is set to τ/n for leader and λ/n for committee where τ and λ are the expected number of leaders and the committee size, respectively. Hence, to validate cryptographic sortition of a message m, a replica calls `VerifyVRF` and checks whether $m.vrf < \tau/n$ or λ/n appropriately. To ensure that some block is proposed in each round with high probability, parameter τ should be chosen much larger than 1, e.g., Algorand [18] chooses $\tau = 26$. We denote $t_H \leftarrow \lceil 2\lambda/3 \rceil$ as the number of votes used to form a QC.

Cryptographic sortition enables player-replaceability in a straightforward manner. In each round, a new leader and committee are elected privately, i.e., only the elected parties know their eligibility before they fulfill their roles. To be resilient to strongly adaptive adversaries, the protocol can use ephemeral keys as in Algorand [11]. For simplicity, we use the same random seed in the genesis block for cryptographic sortition in all rounds. The protocol can be enhanced with a frequently refreshing random seed, as in Algorand [18]. Cryptographic sortition also works in a proof-of-stake setting if eligibility is weighted by stakes.

Blocks and quorum certificates. Client requests are batched into *blocks*. Each block references its predecessor (parent) with the exception of the genesis block which has no predecessor. A block proposed in round r, denoted b_r, has the following format: $b_r := (cmd, parent, justify)$. *cmd* denotes client commands to be committed, *parent* denotes the hash of the parent block of block b_r, and *justify* stores the quorum certificate (QC) for the parent block. A QC for a block b_r consists of at least t_H vote messages. A QC contains the hash, the round number of the block, and metadata such as signatures and accompanying VRF outputs of the vote messages. Notice that we abuse the notation *qc.block* to refer to the actual block instead of the block hash when the context is clear. A block is said to be valid if its parent is valid (genesis block is always valid) and client requests in the block meet application-level validity conditions. A block b_r extends a block $b_{r'}$ if $b_{r'}$ is an ancestor of b_r. Note that a block extends itself. Two blocks b_r and $b'_{r'}$ are conflicting if they do not extend one another.

Full protocol. Each replica maintains a lock denoted as *lockedQC* initialized as $qc_{genesis}$, and a set of $(id, lockedQC)$ pairs for every round, where id is the replica identifier. Each round proceeds as follows. (The full protocol is also presented in Algorithm 3, Appendix C.)

- **Propose.** A replica checks its potential leader eligibility using cryptographic sortition (line 7). Leader can construct a new QC and update its own lock once receiving t_H votes for the same block. Then the leader collects commands and proposes a new block extending from *lockedQC.block*.
- **Process proposals.** Unlike HotStuff, a replica waits for a fixed length period (period $[0, 2\Delta)$) for proposals in case there are multiple leaders eligible to propose (line 13). When a replica receives multiple proposals, it chooses the

one with the smallest VRF hash (line 15). At time 2Δ of this round, all replicas check the validity of the block and the *safety rule* to ensure the new block extends from *lockedQC.block* (line 17). If the block is valid and safe, replicas will update $lockedQC, TC$ properly. When three consecutive QCs are formed, the block is *directly* finalized, and all its previous blocks on the same chain will also be finalized *indirectly* (line 35).

- **Vote and timeout.** Then every replica checks its eligibility to vote for the round (line 20). The vote message is denoted as $\langle \text{VOTE}, r, b, lockedQC, TC[r-1] \rangle$, where r is the current round number, b is the hash of the block the committee replica votes for, $TC[r-1]$ is the set of locks collected from the last round (line 21). When $b = \emptyset$, the vote message serves as a timeout message and meanwhile contains the lock of the committee replica. When b is not empty, it is required that $b.justify = lockedQC$. For each round, replicas selected as committee broadcast their votes to all replicas.
- **Wait for locks.** All replicas cannot enter a round r until they receive t_H locks reported by the committee in round $r-1$. If a vote message in r is received from a replica whose lock has not been received in this round, the lock is added into a set $TC[r]$, and if the lock is more up-to-date, the replica updates its own lock (line 26). The replicas will also update $TC[r-1]$ given $TC^*[r-1]$ contained in the vote. At time 4Δ or later of a round r, replicas enter the next round $r+1$ if $|TC[r]| \geq t_H$.

Communication complexity. The communication complexity of Algorithm 3 is $O(n \cdot poly(\lambda))$ where λ is a security parameter denoting the committee size. In each round, only λ replicas in the committee broadcast messages and the TC in messages is $O(\lambda)$-sized.

3.3 Forensic Protocol and Proof of Forensic Support

When $f < (1-\epsilon)n/3$, the safety and liveness of Algorithm 3 are formally stated in Appendix C. When $f \geq (1-\epsilon)n/3$, it is possible that safety is violated, at which time the following forensic protocol in Algorithm 1 can provide forensic support proved in Theorem 1.

Theorem 1. *When $f \geq (1-\epsilon)n/3$, if two honest replicas finalize conflicting blocks,* Algorithm 1 *provides $((1-\epsilon)2n/3, 2, \lceil \lambda/3 \rceil)$-forensic support.*

Proof. Suppose two conflicting blocks are finalized by two honest replicas, let $b_r, b_{r'}$ be the first directly finalized blocks that are conflicting, w.l.o.g., suppose $r \leq r'$.

Case $r + 2 > r'$.
Culpability. If $r \leq r' < r + 2$, there are two quorums formed in r', these two $QC_{r'}$ intersect in $\lceil \lambda/3 \rceil$ replicas.These replicas should be Byzantine since the protocol requires a replica to vote for at most one block in a round.
Witnesses. In this case, the culpability proof can be constructed from two QCs generated in the same round (line 5-7, Algorithm 3).

Case $r + 2 \leq r'$.

Algorithm 1 Forensic protocol for Algorithm 3

```
1: upon receiving conflicting blocks finalized by two honest replicas do
2:     query the entire blockchain from the two honest replicas
3:     find the first block finalized by consecutive QCs in each chain, denoted by b_r, b_r'
4:     swap b_r, b_r' if r' < r                                   ▷ make sure r ≤ r'
5:     if r + 2 > r' then
6:         find two QC_r' on each chain
7:         return the intersection of two QC_r'
8:     else
9:         query TC[r + 1] from either of the honest replicas
10:        if all lockedQC in TC[r + 1] has round < r then
11:            find QC_r+1 that makes b_r be committed
12:            return the intersection of TC[r + 1] and QC_r+1
13:        else
14:            find block b_r* s.t.
                   (1) r + 2 ≤ r* ≤ r', and
                   (2) b_r' extends b_r*, and
                   (3) b_r conflicts with b_r*, and
                   (4) r* is the smallest round satisfying the above 3 conditions
15:            find QC for b_r*, denoted by QC_r*, return all replicas in QC_r*
```

Culpability. Since b_r is directly finalized in round $r+2$ (by QC_{r+1}), it must be the case that at least t_H committee replicas are locked on at least QC_r (if they are honest), and broadcast their votes with lock to all replicas. Then consider the first block b_{r^*} (possibly $b_{r'}$) that is conflicting with b_r and proposed after $r+1$. On the one hand, b_{r^*} must be extended from a block older than b_r since this is the first conflicting block proposed after r. On the other hand, only those replicas whose locks are staler than QC_r can vote for b_{r*}. Remember that in round $r+1$, at least t_H committee replicas broadcast lock QC_r (or higher lock). And for committee replicas in r^* to vote, they must collect a set $TC[\cdot]$ consisting of at least t_H locks from the committee in every round $< r^*$. If the lock of any one of them is still staler than QC_r, the intersection ($\lceil \lambda/3 \rceil$ replicas) of QC_{r+1} and $TC[r+1]$ is the set of committee replicas who equivocate in round $r+1$ hence are Byzantine (line 10-12, Algorithm 3). Otherwise all the committee replicas who vote for b_{r*} must be Byzantine (line 13-15, Algorithm 3).

Witnesses. In this case, there are two possible scenarios. (i) QC_{r+1} intersects $TC[r+1]$ in $\lceil \lambda/3 \rceil$ replicas, who are culpable since their votes in QC_{r+1} and $TC[r+1]$ are incompatible. (ii) All replicas in QC_{r^*} (at least t_H in total) are Byzantine because they should have received $TC[r+1]$ containing QC_r and update their locks to be at least QC_r, but they vote for a conflicting block b_{r^*} extending from a block older than r. These two cases indicate that with same-round safety violation, the witnesses can detect $\lceil \lambda/3 \rceil$ replicas. If same-round safety violation does not exist, at least t_H culprits can be detected.

4 Forensic Analysis for Player-Replaceable Longest-Chain Protocols

Longest-chain based protocols such as Bitcoin and Ouroboros are another family of SMR protocols. Compared to BFT protocols, they do not have explicit voting procedure and finalization of a block is probabilistic. In this section, we show that forensic support for longest-chain protocols targets leader proposals, and we investigate how player-replaceability influences the forensic properties by analyzing the Ouroboros protocol family, including Ouroboros BFT (OBFT) [23], Ouroboros [24], and Ouroboros Praos [13] (referred to as Praos).

4.1 Protocol Description

Ouroboros is a proof-of-stake blockchain protocol which tolerates up to 1/2 Byzantine stake under a synchronous network. Building on Ouroboros, Praos is a player-replaceable protocol secure under an adaptive adversary. On the other hand, OBFT is a deterministic permissioned derivative of Ouroboros for ledger consensus. We start with a general simplified description of Ouroboros family.

Each of the n replicas maintain the longest blockchain $C = b_0 \cdots b_r$ (b_0 is the genesis block and round number $r \geq 0$) containing a sequence of blocks. The length of a blockchain is the number of blocks on the chain, and the height of a block b_r is the length of chain $b_0 \cdots b_r$. Each block $b_r := (data, parent, proof)$ proposed in round r contains block data $data$, the hash of the parent block $parent$, and a block proof $proof$ that replicas can use to verify the validity of the block. The block is valid if its data and parent are valid, and it is signed by a certified round leader. The protocol proceeds in rounds, in round j, replica i performs the following.

- **Blockchain update.** The replica collects chains diffused by all valid leaders in the current round as set $\mathbb{C}$. Denote the longest valid chain (does not fork from C more than κ blocks) among $\mathbb{C}$ as C'. It updates its local longest chain C with C' if C' is strictly longer.
- **Blockchain extension.** If the replica is a leader in the current round, it generates a new block with a proof of leader (stored in *proof*). After the new block is appended to the local longest chain, the replica diffuses the new chain to other replicas.

The key distinguishing factors between the protocols we consider are the leader election process and the confirmation depth κ. We describe these in more detail for each of the three protocols below. Observe that we omit details related to message verification, stake distribution, randomness generation, etc. in our analysis since they do not matter in terms of forensic support. In terms of stake distribution, for our analysis we assume the stake of each of the parties are equal, although our analysis should be generalizable when the stakes are not equal.

4.2 Forensic Support for OBFT

We start with the simplest of the three protocols, OBFT, which is a deterministic permissioned protocol. In OBFT, leaders are elected in a round-robin manner,

Algorithm 2 Forensic protocol for OBFT

```
upon receiving conflicting outputs from two honest replicas do
    query the entire blockchain from the two honest replicas
    r, r' ← minimum / maximum round number among all blocks
    S ← ∅
    for i = r, r + 1, ··· , r' do
        if two conflicting blocks are generated in round i by a valid leader then
            add leader j of round i into S          ▷ j − 1 = (i − 1) mod n in OBFT
    return S
```

i.e., in round j, if $i-1 = (j-1) \mod n$, replica i is the round leader. Moreover, a block is committed if it is on the longest chain that is $\kappa = 3t+1$ blocks deep.

Security analysis. The key property satisfied by OBFT is that the blockchain is not forkable during a period of execution where the fraction of Byzantine leaders over all rounds is $< 1/3$ [23, Proposition 3.7]. Furthermore, under covert adversaries (who do not leave any verifiable evidence of misbehavior), the threshold of Byzantine leaders' fraction becomes $1/2$.

First, observe that the adversary can simply behave as covert adversary to fork the chain when $f > n/2$, in which case no cryptographic evidence will be left behind and therefore there is no forensic support (formally stated in Theoreum 7). Thus, we argue about forensic support only when $n/3 \leq f < n/2$. In such a situation, the adversary can undertake multiple malicious actions such as not extending any chain when it is the leader, extending a smaller chain, extending more than one chain, etc. Among these, the only detectable action is when a Byzantine leader proposes two or more blocks in the same round. Thus, given a safety violation (i.e., two or more chains of depth κ), if any leader proposes two different blocks in these chains in the same round, it is a Byzantine replica. The proof to hold it culpable are the two signed blocks.

The key part of our forensic analysis is to determine the minimum number of rounds that must equivocate to violate safety for an execution. The following lemma formally presents a bound on equivocating rounds (see proof in Appendix E.1).

Lemma 1. *If there exist two or more longest chains that diverge from height h until $h+l-1$, define $R_{\mathcal{H}} = |\{\omega_i = 0 | r \leq i \leq r'\}|$ as the number of rounds whose leader is unique and honest (where r, r' are the minimum / maximum round number among all blocks with height between h and $h+l-1$), $R_{\mathcal{A}} = |\{\omega_i = 1 | r \leq i \leq r'\}|$ as the number of rounds that possibly equivocate (generate more than one block), where $\forall i \in [r, r']$*

$$\omega_i = \begin{cases} 0 & \textit{the leader of block in round } i \textit{ is unique and honest} \\ 1 & \textit{otherwise} \end{cases}$$

Denote the number of rounds whose leader generates two or more blocks (for each chain) as X, then we have $X \geq R_{\mathcal{H}} - R_{\mathcal{A}}$.

Applying the above lemma to an execution of binary consensus version of OBFT, where the protocol terminates after $2n$ rounds and the majority bits of the first n rounds of blocks will be output (the first n blocks are finalized by κ-deep rule when $n = 3t + 1$), we can get the following theorem.

Theorem 2. *When $n/3 \leq f < n/2$, if two honest replicas output conflicting values,* Ouroboros BFT *has $(n/2,\ 2,\ n - 2f)$-forensic support.*

Proof. Suppose safety fails, i.e., when protocol terminates, there exist two longest chains finalized by honest replicas, whose majority bits among the first n rounds are different. Since OBFT uses round-robin leader election, we can apply Lemma 1 with $R = r' - r + 1 = n, R_{\mathcal{H}} = n - f$ and $R_{\mathcal{A}} \leq f$ (since some Byzantine replicas may stay silent), then $d = R_{\mathcal{H}} - R_{\mathcal{A}} \geq n - 2f$.

Notice that in Theorem 2, $d = n - 2f = 0$ when $f = n/2$, thus $m = n/2$. And two honest replicas are required to provide two different longest chains for irrefutable proof, thus $k = 2$. With two different chains, the forensic protocol to detect Byzantine leaders is to find these rounds whose leaders have proposed more than one values (Algorithm 2). Further, the impossibility for $f > n/2$ is formally stated in Appendix D.

4.3 Forensic Support for Ouroboros and Praos

In Ouroboros and Praos, the leader election process is randomized. In Ouroboros, in each round, a unique leader is elected randomly among the n replicas. In Praos, each replica evaluates a round dependent VRF independently. The probability of a replica i with relative stake α_i to be selected as a round leader is

$$\Pr[i \text{ is a leader}] = \phi_w(\alpha_i) = 1 - (1 - w)^{\alpha_i}$$

where w is the probability of electing a replica in a round when it holds all the stake. The probability of electing a leader with lesser stake is scaled as described above; due to this, zero, one or multiple leaders may be elected in a round.

Since the leader election process is randomized, "forkability" of the chain is not deterministic. However, when considering a single execution after safety is violated, since they are executed under the same longest chain rule, a similar intuition as in OBFT is still applicable for forensic support. Even though these protocols tolerate $t < n/2$ w.h.p., there may be executions with safety violation when the fraction of adversarial rounds is $< 1/2$ independent of f.[5] We formally discuss this in the following theorem.

Theorem 3. *For a given execution, if during rounds $[r, r + R - 1]$, two honest replicas finalize two conflicting blocks at height h (i.e., there exist two longest chains that diverge from height h to $h + \kappa - 1$), then* Ouroboros *has $(n/2,\ 2,\ d)$-forensic support where $d = (1 - \epsilon)\left(n\left(1 - (1 - 1/n)^R\right) - 2fR/n\right)$ except with $\exp(-\Omega(R))$ probability.*

[5] Our paper discusses the forensic ability after a safety violation happens. In particular, we ignore *when* such a violation happens. The probability of such a safety violation has been shown in Ouroboros [24, Figure 8].

We provide a short proof here, with complete proof in Appendix E.2. According to Lemma 1, the expected number of rounds whose leader equivocates $E[X] \geq E[R_{\mathcal{H}} - R_{\mathcal{A}}] \geq (n-2f)R/n$. However, some of these rounds may have the same leaders, denote D as the number of rounds whose leader has been selected before, then $E[d] \geq E[X-D] \geq (n-2f)R/n - (R - n\left(1-(1-1/n)^R\right))$. Finally, using a Chernoff bound, we have $d = (1-\epsilon)\left(n\left(1-(1-1/n)^R\right) - 2fR/n\right)$ for $\epsilon > 0$ except with $\exp(-\Omega(R))$ probability.

Corollary 1. *When $R/n = o(1)$,* Ouroboros *has $(n/2,\ 2,\ d)$-forensic support where $d = (1-\epsilon)(n-2f)\kappa/n$ except with $\exp(-\Omega(\kappa))$ probability.*

Proof. When $R/n = o(1)$, by binomial approximation, $E[D] \sim 0$. With $R \geq \kappa$, we have $E[d] \geq (n-2f)\kappa/n$. Using a Chernoff bound, we have $d = (1-\epsilon)(n-2f)\kappa/n$ for $\epsilon > 0$ except with $\exp(-\Omega(\kappa))$ probability.

The random leader election process of Ouroboros adds some uncertainty to the forensic analysis due to possibly duplicated equivocating leaders, which slightly impairs the forensic ability (though when κ/n is very small, this effect will be negligible). In comparison, since in Praos multiple leaders may be elected in a round, there may be equivocation even when all of them are honest. In this case, if multiple honest leaders are elected in round i, $\omega_i = 1$ per the definition in Lemma 1, the round may contribute to the violation of safety. But such leaders in these rounds should not be held culpable by the forensic protocol.

Recall the probability for any replica i to be elected as a leader in a round is defined as $\phi_w(\alpha_i)$. Observe that the probability that no one is elected as a leader in some round is

$$p_0 = \Pr[\text{no leader is elected}] = \prod_{i=1}^{n}(1-w)^{1/n} = 1-w$$

And the probability that only one honest or adversarial leader is elected is

$$p_1^{\mathcal{H}} = \Pr[\text{one honest leader}] = (n-f)(1-(1-w)^{1/n})(1-w)^{1-1/n} = (n-f)g(n)$$

$$p_1^{\mathcal{A}} = \Pr[\text{one adversarial leader}] = f(1-(1-w)^{1/n})(1-w)^{1-1/n} = fg(n)$$

where $g(n) = (1-(1-w)^{1/n})(1-w)^{1-1/n}$. Therefore,

$$\Pr[\text{multiple leaders are elected}] = p_2 = 1 - p_0 - p_1^{\mathcal{H}} - p_1^{\mathcal{A}} = w - ng(n)$$

Based on the analysis above, Praos has the following forensic support.

Theorem 4. *For a given execution, if during rounds $[r, r+R-1]$, two honest replicas finalize two conflicting blocks at height h (i.e. there exist two longest chains that diverge from height h until $h+\kappa-1$).* Ouroboros Praos *has $(n/2,\ 2,\ d)$-forensic support where $d = (1-\epsilon)\left((n-2f)g(n)R - T(n,R)\right)$ except with $\exp(-\Omega(R))$ probability, and $T(n,R) = 2(w-ng(n))R + (R - n + n(1-g(n))^R) > 0$.*

Corollary 2. *When* $R/n = o(1)$, Ouroboros Praos *has* $(n/2,\ 2,\ d)$*-forensic support where*

$$d = (1-\epsilon)((w-w^2)(n-2f)\kappa/n - (1-w+3w^2)\kappa)$$

except with $\exp(-\Omega(\kappa))$ *probability.*

The complete proof of Theorem 4 and Corollary 2 are presented in Appendix E.3 and E.4 respectively.

5 Discussion and Conclusion

We begin with two observations about the forensic properties for player-replaceable protocols in both families.

First, compared to quadratic complexity protocols analyzed in [35], player-replaceable protocols require fewer replicas to send messages; correspondingly, only fewer replicas ($O(\lambda)$ or $O(\kappa)$) can be held culpable when there is a safety violation even if the total number of Byzantine replicas are far higher ($O(n)$). For BFT style protocols, whenever forensic support is available, the number of culpable replicas is in the same proportion to the quorum size as in the non player-replaceable setting. Moreover, this number is independent of f. When there is no forensic support, no replica may be held culpable. On the other hand, since longest-chain style protocols are synchronous and can tolerate $t < n/2$, our detection is applicable in the regime $n/3 \leq f < n/2$. Safety violation is still possible since these protocols are randomized. In this case, we observe that the number of culpable replicas decreases linearly as f increases.

Second, qualitatively, the key difficulty with holding replicas culpable is related to potentially having a different set of replicas participating in each round. In BFT protocols, voting rules stipulate how previous actions impose restrictions on current behavior. Due to player-replaceability, voters' behaviors across rounds are less traceable, which can be utilized by adversary to conceal evidence of deviation. Thus, to construct a protocol with strong forensic support, we need to reconnect across-rounds actions. In our protocol, transition certificates serve as the link that the forensic protocol can use to identify culpable behavior. In longest-chain protocols, extending blocks are also used to certify the previous blocks. However, there is no evidence if a Byzantine replica appends blocks to a shorter chain. Thus the only culpable behavior our forensic protocol can detect is when a leader double proposes blocks (equivocates) in a round. As a consequence, the forensic analysis for longest-chain protocols only focuses on the same-round behavior. Player-replaceability can still adversely affect forensic analysis if multiple leaders are allowed to be elected in the same round. However, as seen in Corollary 2, this has a limited effect when κ/n is small.

Blockchain protocols perform two distinct roles: first, they are secure against adversarial behavior (by a fraction of participating nodes). Second, they are imbibed with incentives that encourage participation, and furthermore via honest behavior (i.e., following protocol). In this paper, forensic support of protocols serves the implicit role of incentives (identification of Byzantine action with cryptographic integrity strongly discourages deviation from following protocol). By

studying both strong security (i.e., against fully adaptive adversaries) and strong forensic support (i.e., identifying the maximum number of Byzantine nodes from just the transcripts of two honest nodes) we are considering both sides of the blockchain protocol (resistance to Byzantine behavior as well as incentives to promote honest behavior). Further, the identification of BFT protocols with both strong security and strong forensic support properties allows us to construct blockchain protocols with implicitly built incentive mechanisms. It is in these two senses that the title of this manuscript is constructed.

References

1. Abraham, I., Chan, T.H., Dolev, D., Nayak, K., Pass, R., Ren, L., Shi, E.: Communication complexity of byzantine agreement, revisited. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. pp. 317–326 (2019)
2. Abraham, I., Devadas, S., Dolev, D., Nayak, K., Ren, L.: Synchronous byzantine agreement with expected $O(1)$ rounds, expected $O(n^2)$ communication, and optimal resilience. In: International Conference on Financial Cryptography and Data Security. pp. 320–334. Springer (2019)
3. Abraham, I., Malkhi, D., Spiegelman, A.: Asymptotically optimal validated asynchronous byzantine agreement. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. pp. 337–346 (2019)
4. Abraham, I., Nayak, K., Ren, L., Xiang, Z.: Good-case latency of byzantine broadcast: a complete categorization. arXiv preprint arXiv:2102.07240 (2021)
5. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Theory of Cryptography Conference. pp. 137–156. Springer (2007)
6. Buchman, E., Kwon, J., Milosevic, Z.: The latest gossip on bft consensus. arXiv preprint arXiv:1807.04938 (2018)
7. Buterin, V., Griffith, V.: Casper the friendly finality gadget. arXiv preprint arXiv:1710.09437 (2017)
8. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems (TOCS) **20**(4), 398–461 (2002)
9. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OSDI. vol. 99, pp. 173–186 (1999)
10. Chen, J., Gorbunov, S., Micali, S., Vlachos, G.: Algorand agreement: Super fast and partition resilient byzantine agreement. IACR Cryptol. ePrint Arch. **2018**, 377 (2018)
11. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theoretical Computer Science **777**, 155–183 (2019)
12. Civit, P., Gilbert, S., Gramoli, V.: Polygraph: Accountable byzantine agreement. IACR Cryptol. ePrint Arch. **2019**, 587 (2019)
13. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 66–98. Springer (2018)
14. Dolev, D., Reischuk, R.: Bounds on information exchange for byzantine agreement. Journal of the ACM (JACM) **32**(1), 191–204 (1985)
15. Dolev, D., Strong, H.R.: Authenticated algorithms for byzantine agreement. SIAM Journal on Computing **12**(4), 656–666 (1983)

16. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM (JACM) **35**(2), 288–323 (1988)
17. Fischer, M.J., Lynch, N.A., Merritt, M.: Easy impossibility proofs for distributed consensus problems. Distributed Computing **1**(1), 26–39 (1986)
18. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. pp. 51–68 (2017)
19. Haeberlen, A., Kouznetsov, P., Druschel, P.: Peerreview: Practical accountability for distributed systems. ACM SIGOPS operating systems review **41**(6), 175–188 (2007)
20. Haeberlen, A., Kuznetsov, P.: The fault detection problem. In: International Conference On Principles Of Distributed Systems. pp. 99–114. Springer (2009)
21. Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: Annual Cryptology Conference. pp. 369–386. Springer (2014)
22. Katz, J., Koo, C.Y.: On expected constant-round protocols for byzantine agreement. In: Annual International Cryptology Conference. pp. 445–462. Springer (2006)
23. Kiayias, A., Russell, A.: Ouroboros-bft: A simple byzantine fault tolerant consensus protocol. IACR Cryptol. ePrint Arch. **2018**, 1049 (2018)
24. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Annual International Cryptology Conference. pp. 357–388. Springer (2017)
25. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative byzantine fault tolerance. In: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles. pp. 45–58 (2007)
26. LAMPORT, L., SHOSTAK, R., PEASE, M.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems **4**(3), 382–401 (1982)
27. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th annual symposium on foundations of computer science (cat. No. 99CB37039). pp. 120–130. IEEE (1999)
28. Mostefaoui, A., Moumen, H., Raynal, M.: Signature-free asynchronous byzantine consensus with t¡ n/3 and o (n2) messages. In: Proceedings of the 2014 ACM symposium on Principles of distributed computing. pp. 2–9 (2014)
29. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
30. Neu, J., Tas, E.N., Tse, D.: Ebb-and-flow protocols: A resolution of the availability-finality dilemma. arXiv preprint arXiv:2009.04987 (2020)
31. Neu, J., Tas, E.N., Tse, D.: Snap-and-chat protocols: System aspects. arXiv preprint arXiv:2010.10447 (2020)
32. Neu, J., Tas, E.N., Tse, D.: The availability-accountability dilemma and its resolution via accountability gadgets. arXiv preprint arXiv:2105.06075 (2021)
33. Ranchal-Pedrosa, A., Gramoli, V.: Blockchain is dead, long live blockchain! accountable state machine replication for longlasting blockchain. arXiv preprint arXiv:2007.10541 (2020)
34. Schneider, F.B.: Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Comput. Surv. **22**(4), 299–319 (dec 1990). https://doi.org/10.1145/98163.98167, `https://doi.org/10.1145/98163.98167`
35. Sheng, P., Wang, G., Nayak, K., Kannan, S., Viswanath, P.: Bft protocol forensics. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1722–1743 (2021)
36. Stewart, A., Kokoris-Kogia, E.: Grandpa: a byzantine finality gadget. arXiv preprint arXiv:2007.01560 (2020)

37. Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: Hotstuff: Bft consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. pp. 347–356 (2019)

A Related Work

Forensic support. The idea of holding misbehaving participants accountable has been discussed in earlier works [19,20] for distributed systems in general. Similar ideas have been discussed in other domains such as in secure multi-party computation with identifiable abort [21]. For SMR and blockchain, recent works [7,36,33] have discussed finality and accountability and designed their consensus protocols with the focus on accountability. A recent work [35] formally defined forensic support for Byzantine Agreement (BA) and analyzed it for protocols such as PBFT [9,8], HotStuff [37], VABA [3], and Algorand [18,11]. They show that except Algorand, the other protocols have forensic support depending on their implementation details. Another work [32] introduces the notion of accountable-safety that combines the traditional safety with the ability to hold Byzantine parties accountable, and shows that there exists a trade-off between accountable-safety and liveness. In [30], they proposed the class of snap-and-chat protocols, which combines a longest chain protocol with a BFT protocol to provide both availability and finality, and in reference [31] they showed that if the BFT protocol provides accountable-safety then the snap-and-chat protocol inherits accountable-safety. Nevertheless, to the best of our knowledge, this work is the first to investigate forensic support or accountability for longest chain protocols.

BFT protocols. Among Byzantine fault tolerant (BFT) protocols, HotStuff [37] is the first partially synchronous SMR protocol that enjoys a linear communication (with threshold signature) of view change and optimistic responsiveness. Forensic support for it was discussed in [35] at the cost of quadratic communication complexity (due to replacing threshold signatures with aggregated signatures). Algorand [18,11] is a player-replaceable proof-of-stake protocol that was shown to not have any forensic support. Our work explores forensic support for player-replaceable protocols achieving sub-quadratic communication complexity under different network settings and assumptions; the protocol is loosely inspired from HotStuff.

Longest chain protocols. The longest chain protocol [29] in Bitcoin and Ethereum uses proof-of-work, a mechanism where participants exhausting computation power to solve a hash puzzle and use the solution as the eligibility to propose a block. Proof-of-stake protocols such as Ouroboros [24] and Praos [13] keep the longest chain rule but replaces the energy-consuming puzzle to a stake weighted puzzle, which a stake-holder with higher stakes has higher probability to solve. These protocols are resilient to $(1/2 - \epsilon)$ adversarial computation power or stake; in work [35, Appendix B] it has been proven that forensic support cannot be provided when $t = \lceil n/2 \rceil - 1$. Ouroboros BFT [23] is a longest chain protocols with deterministic block proposing assignment and it tolerates $(1/3 - \epsilon)$ Byzantine adversary as well as $(1/2 - \epsilon)$ covert adversary [5].

B Practicality Analysis

We consider a network with n replicas, among which f replicas are corrupted by an adversary. Our protocol can tolerate up to $t = (1-\epsilon)n/3$ Byzantine faults for safety and liveness and hold at least $\lceil \lambda/3 \rceil$ Byzantine replicas culpable when $f > t$, where ϵ, n and λ are three parameters that together determine the failure probability of the protocol. In this section we analyze practical parameters that can be used in implementation to ensure safety and liveness with overwhelming probability when $f \leq t$.

The protocol employs cryptographic sortition to select committee in each round secretly. Every replica independently computes a hash value to determine its eligibility to vote with probability λ/n. In each round, we denote the number of selected honest and Byzantine replicas as two random variables H and B, both of which follow the binomial distribution with the probability mass function $Pr(H = h) = \phi(h; n-f, \lambda/n)$ and $Pr(B = b) = \phi(b; f, \lambda/n)$, where f is defined as below

$$\phi(k; n, p) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$$

The protocol guarantees safety when $H + 2B < 2t_H$ and liveness when $H \geq t_H$. Therefore we can calculate the probability that each condition fails.

$$\begin{aligned} Pr(\text{liveness fails}) &= Pr(H < t_H) \\ &= \sum_{h=0}^{t_H - 1} \phi(h; n-f, \lambda/n) \\ &= F(t_H - 1; n-f, \lambda/n) \end{aligned}$$

$$\begin{aligned} Pr(\text{safety fails}) &= Pr(H + 2B \geq 2t_H) \\ &= \sum_{b=0}^{\infty} \phi(b; f, \lambda/n) \sum_{h=\max(0, 2t_H - 2b)}^{\infty} \phi(h; n-f, \lambda/n) \\ &= \sum_{b=0}^{\infty} \phi(b; f, \lambda/n)(1 - F(\max(0, 2t_H - 2b - 1); n-f, \lambda/n) \\ &= 1 - \sum_{b=0}^{t_H - 1} \phi(b; f, \lambda/n) F(2t_H - 2b - 1; n-f, \lambda/n) \end{aligned}$$

where F is the cumulative distribution function of f. By union bound, the total failure probability does not exceed the sum of the above two probabilities. We begin by assuming, as stated in Appendix B of Algorand [18], that the total number of replicas (stakes) might be arbitrarily high, in which case $\phi(k; n, \lambda/n) = \lambda^k \exp(-\lambda)/k!$. We set the maximal tolerable total failure probability to $5 \cdot 10^{-9}$ and search for optimal ϵ given different committee size λ, the results are shown in Figure 2. When choosing the same committee size $\lambda = 2000$

as Algorand's implementation, we can tolerate 17.3% Byzantine fraction (slightly less than 20% used in Algorand). To further optimize the committee size, we may apply the same technique used in Algorand to balance safety and liveness conditions by adjusting the quorum threshold t_H (we use $t_H = \lceil 2/3\lambda \rceil$ whereas in Algorand, $t_H = 0.685\lambda$).

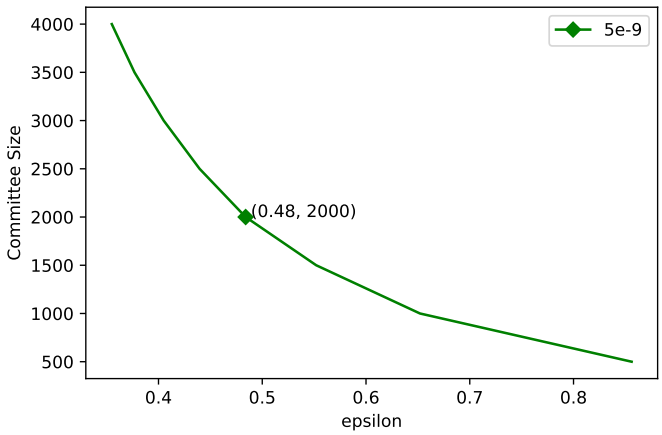

Fig. 2: The optimal Byzantine fraction parameter ϵ with different committee size λ to achieve the failure probability of $5 \cdot 10^{-9}$.

After selecting the committee size $\lambda = 2000$ and $\epsilon = 0.48$, we examine the impact of n. We evaluate the total number of replicas up to 400k, which is the estimated population in Ethereum. Figure 3 shows the overall failure probability increases when there are more replicas. This indicates the protocol will be more secure with larger population in the given setting.

C Full Protocols and Safety and Liveness of Algorithm 3

In this section, we provide the full protocols, as well as formal proofs for safety and liveness when $f < (1-\epsilon)n/3$. The safety follows from the following arguments.

Lemma 2. *For any valid QC, QC', if $QC.round = QC'.round$, $QC.block$ and $QC'.block$ are not conflicting except with $\exp(-\Omega(\lambda))$ probability.*

Proof. To show a contradiction, suppose QC, QC' are formed in the same round and $QC.block, QC'.block$ are conflicting, then at least $2t_H$ distinct votes are generated by the committee in the same round. Since honest replicas will only vote for one block, we require the following inequality to hold: $|H| + 2|B| \geq 2t_H$, where H and B are the set of honest and Byzantine committee members

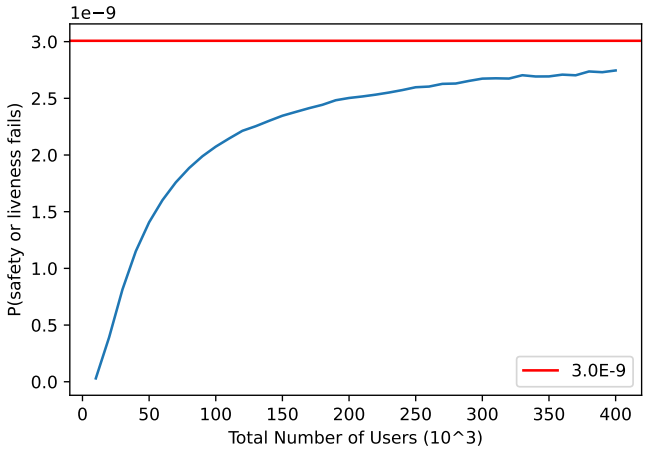

Fig. 3: The relation of failure probability and population size n given $\lambda = 2000, \epsilon = 0.48$. The red line represents the approximate probability when we assume there are arbitrary large population.

respectively. By Chernoff bound, the probability of this event is $\exp(-\Omega(\lambda))$ since there are at most $(1-\epsilon)n/3$ Byzantine replicas.

Theorem 5. *Any two conflicting blocks will not be both finalized by honest replicas except with* $\exp(-\Omega(\lambda))$ *probability.*

Proof. For contradiction, suppose two conflicting blocks are finalized by two honest replicas (see Figure 4). Let $b_r, b_{r'}$ be the first directly finalized blocks (finalized in $r+2, r'+2$ by two consecutive QCs) that are conflicting, w.l.o.g., assume $r < r'$. Further, we can assume $r+1 < r'$ due to Lemma 2. In round $r+2$, b_r is finalized, at least t_H committee replicas in round $r+1$ receive b_{r+1} and update lock to at least QC_r (the QC containing votes in round r) if they are honest.

Then consider the first block b_{r^*} (possibly $b_{r'}$) conflicting with b_r and proposed after $r+1$. The $b_{r^*}.justify$ must be formed in a round $< r$ because b_{r^*} is the first conflicting block after $r+1$. Since $b_{r'}$ is finalized, a QC for b_{r^*} must be formed, which means at least t_H votes are generated in round r^*.

To enter $r^* > r+1$, replicas need to collect at least t_H locks from committee members in $r+1$. Remember at least t_H committee replicas in round $r+1$ are locked on QC_r, then for every replica, $TC[r+1]$ must contain at least one QC_r reported by honest replica except with $\exp(-\Omega(\lambda))$ probability. Since honest replicas who are locked on QC_r cannot vote for a staler lock and $f < (1-\epsilon)n/3$, it is contradictory that a QC for b_{r^*} is formed.

Since the protocol is synchronous after GST, the liveness satisfies the following statements.

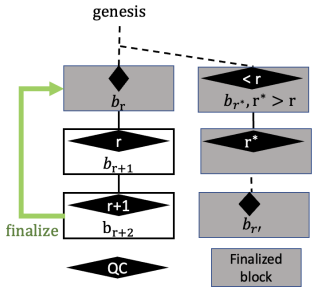

Fig. 4: A case where two conflicting blocks are both finalized.

Lemma 3. *After GST, if an honest replica enters some round r at time T, all honest replicas enter round r by time $T+4\Delta$ except with $\exp(-\Omega(\lambda))$ probability.*

Proof. Suppose the earliest honest replica enters round r at time T (after GST). If this honest replica is a committee replica in round r, it will broadcast $TC[r-1]$. Due to synchrony, all honest replicas will receive $TC[r-1]$ within $T+2\Delta$ time and enter round r. Otherwise, since there are t_H messages in $TC[r-1]$ sent by committee replicas in round $r-1$, at least t_H replicas have entered $r-1$ before $T-\Delta$, among which at least one is honest except for probability $\exp(-\Omega(\lambda))$ since $f < (1-\epsilon)n/3$. The honest committee replica in round $r-1$ will broadcast $TC[r-2]$, therefore at T, all honest replicas will enter round $r-1$. Then honest committee replicas in $r-1$ will broadcast votes before $T+2\Delta$. And since there are at least t_H honest committee replicas except with $\exp(-\Omega(\lambda))$ probability, all honest replicas can collect $TC[r-1]$ by $T+4\Delta$.

Lemma 4. *At any point after GST, if the highest QC is QC_r, and three consecutive rounds $r+1 \sim r+3$ have honest leaders, then b_{r+1} will be finalized within 16Δ time except with $\exp(-\Omega(\lambda))$ probability.*

Proof. After GST, when QC_r is sent by leader $r+1$, by Lemma 3, all honest replicas enter $r+1$ within 4Δ. If the leaders of $r+1 \sim r+3$ are honest, leader $r+1$ will propose b_{r+1} extending from the parent of b_r and all honest replicas are willing to vote. The leader $r+2$ can collect QC_{r+1} except with $\exp(-\Omega(\lambda))$ probability and propose b_{r+2}. Similarly leaders $r+3$ can collect enough votes to form QC_{r+2}. Then b_{r+1} is finalized within 3 rounds, each round costs at most 4Δ.

Theorem 6 (Liveness). *Every valid block will be eventually finalized by every honest replica.*

Proof. Since leader changes in every round randomly, the probability to elect consecutive 3 honest leaders is $> (2/3)^3$. Whenever a QC is formed and 4 honest leaders are elected consecutively, by Lemma 4, a block will be finalized within a time bound with overwhelming probability. Thus all honest replicas will eventually finalize all valid blocks.

D Impossibility when $f > n/2$ of OBFT

Theorem 7. *When $f \geq n/2$, if two honest replicas output conflicting values, $(n/2,\ n-f,\ d)$-forensic support is impossible with $d > 0$ for Ouroboros BFT.*

Proof. Suppose the protocol provides forensic support to detect $d \geq 1$ replicas with irrefutable proof. To prove this result, we construct two worlds where a different set of f replicas are Byzantine in each world but a forensic protocol cannot be correct in both worlds. We fix $f = n/2$, although the arguments will apply for any $f > n/2$.

Let there be two replica partitions P, Q, where $P = \{r_i \mid i \mod 2 = 1\}$ and $Q = \{r_i \mid i \mod 2 = 0\}$. All replicas in P (except r_1) have input value $v_i = 0$, all replicas in Q (except r_2) have input value $v_i = 1$, $v_1 = 1, v_2 = 0$.

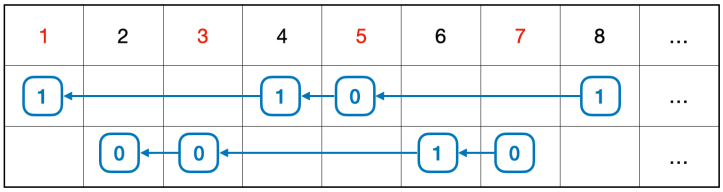

Fig. 5: Each row represents for a possible longest chain. In world 1, the leaders of odd rounds are Byzantine. In world 2, the leaders of even rounds are Byzantine.

World 1. Let P be Byzantine replicas in this world. When Byzantine replica i becomes the leader, it generate the block with input value but only sends the block to replica $i + 3$. Thus, there exists two longest chains as is shown in Figure 5. Honest replicas who receive the first chain will output value 1 while honest replicas who receive the second chain will output value 0. Thus there is a safety violation.

During the forensic protocol, all replicas send their transcripts to the client. Since the protocol has forensic support for $d \geq 1$, the forensic protocol determines some subset of P are culpable.

World 2. Let R be Byzantine replicas in this world. When Byzantine replica i becomes the leader, it generates the block with input value but does not follow the longest chain rule. It always appends another chain to create a fork. Assume honest replicas always choose to append on top of the longest chain sent by Byzantine replicas. Thus, the situation is the same as in World 1. Honest replicas who receive the first chain will output value 1 while honest replicas who receive the second chain will output value 0. Thus there is a safety violation.

Similarly, all replicas send their transcripts to the client and these transcripts are the same as those in World 1. So the forensic protocol outputs some subset of P as culpable replicas. In World 2, this is incorrect since replicas in P are honest. This completes the proof.

E Proof of Lemmas and Theorems

E.1 Proof of Lemma 1

Proof. Choose two longest chains (if there are more than two, choose any two of them). Let $B_{\mathcal{H}}$ and $B_{\mathcal{A}}$ be the number of blocks on these two chains that are generated in $R_{\mathcal{H}}$ and $R_{\mathcal{A}}$ rounds respectively. Since honest replicas will always generate blocks and extend from the longest chain when being leaders, then $l \geq R_{\mathcal{H}}$. Moreover, honest leaders will only generate one block per round, but their blocks may not be on these two chains, thus $B_{\mathcal{H}} \leq R_{\mathcal{H}}$. Byzantine replicas can choose to generate zero, one, and more than one block per round, thus $B_{\mathcal{A}} \leq R_{\mathcal{A}} + X$, then,

$$\begin{aligned} R_{\mathcal{H}} + R_{\mathcal{A}} + X \geq B_{\mathcal{H}} + B_{\mathcal{A}} = 2l \geq 2R_{\mathcal{H}} \\ \Longrightarrow X \geq R_{\mathcal{H}} - R_{\mathcal{A}} \end{aligned}$$

E.2 Proof of Theorem 3

Proof. For the given execution, denote the number of rounds whose leader equivocates as X. According to Lemma 1, we have $X \geq R_{\mathcal{H}} - R_{\mathcal{A}}$, which means $E[X] \geq E[R_{\mathcal{H}} - R_{\mathcal{A}}]$. The leader election of Ouroboros is uniformly random, therefore $E[R_{\mathcal{H}}] = (n-f)R/n$ and $E[R_{\mathcal{A}}] \leq fR/n$, we have

$$\begin{aligned} E[X] &\geq E[R_{\mathcal{H}} - R_{\mathcal{A}}] \\ &\geq \frac{(n-f)R}{n} - \frac{fR}{n} \\ &= \frac{(n-2f)R}{n} \end{aligned}$$

Among X rounds, there are possibly duplicated leaders selected more than once, and letting duplicated leaders equivocate is beneficial for adversary to reduce the number of replicas who leave verifiable evidence for misbehavior. Thus, $d \geq X - D$, where D is the number of rounds whose leader has been selected before among R rounds, i.e.,

$$D = \sum_{i=r}^{r+R-1} \mathcal{I}\left[\text{leader of round } i \text{ is selected among } [r, i-1]\right]$$

where $\mathcal{I}$ is the indicator function.

Let $Z_j^R = 0$ denote the event that the j-th replica was not selected in these R rounds and $Z_j^R = 1$ otherwise. Then, $E\left[Z_j^R\right] = 1 - (1-1/n)^R$.

Then the expected number of different replicas that have been selected as leaders among these R rounds is

$$E\left[\sum_{j=1}^{n} Z_j^R\right] = n\left(1 - \left(1 - \frac{1}{n}\right)^R\right)$$

which implies

$$E\left[D\right] = E\left[R - \sum_{j=1}^{n} Z_j^R\right] = R - n\left(1 - \left(1 - \frac{1}{n}\right)^R\right)$$

Therefore the expected number of Byzantine replicas that will be held culpable $E[d]$ during the execution is

$$\begin{aligned} E\left[d\right] &\geq E\left[X - D\right] \\ &\geq \frac{(n-2f)R}{n} - R + n\left(1 - \left(1 - \frac{1}{n}\right)^R\right) \\ &= n\left(1 - \left(1 - \frac{1}{n}\right)^R\right) - \frac{2fR}{n} \end{aligned}$$

Using a Chernoff bound, we have

$$d = (1-\epsilon)\left(n\left(1 - (1-1/n)^R\right) - 2fR/n\right)$$

for $\epsilon > 0$ except with $\exp(-\Omega(R))$ probability. $k = 2$ since two honest replicas are required to provide different chains, with which a forensic protocol similar to Algorithm 2 can be used to identify equivocating leaders. When $f > n/2$, a private chain attack is possible, thus $m = n/2$.

E.3 Proof of Theorem 4

Proof. For the given execution, denote the number of rounds whose leader equivocates as X, to satisfy the condition in Lemma 1, we have $X \geq R_{\mathcal{H}} - R_{\mathcal{A}}$ and therefore $E[X] \geq E[R_{\mathcal{H}} - R_{\mathcal{A}}]$. The election gives $E[R_{\mathcal{H}}] = p_1^{\mathcal{H}} R$ and $E[R_{\mathcal{A}}] \leq (p_1^{\mathcal{A}} + p_2)R$, then we have

$$E[X] \geq E[R_{\mathcal{H}} - R_{\mathcal{A}}] \geq p_1^{\mathcal{H}} R - (p_1^{\mathcal{A}} + p_2)R$$

In Praos, adversary can not only let duplicated leaders equivocate, but make rounds that have multiple leaders generate more than one blocks, thus $d \geq X - D - Y$ where D is a random variable representing for the number of rounds whose leader has been uniquely selected before among R rounds and Y is the number of rounds among R rounds that have multiple leaders. Therefore we have $E[Y] = p_2 R$ and

$$E[d] \geq E[X - D - Y] = E[X - D] - p_2 R$$

Let $Z_j^R = 1$ denote the event that the j-th replica was uniquely selected in these R rounds and $Z_j^R = 0$ otherwise. Then,

$$E\left[Z_j^R\right] = 1 - (1 - g(n))^R$$

Since $g(n)$ is the probability for a replica to be uniquely selected. Then the expected number of different replicas that have been selected as leaders among these R rounds is

$$E\left[\sum_{i=1}^{n} Z_i^R\right] = n - n(1-g(n))^R$$

and

$$E[D] = E\left[R - \sum_{i=1}^{n} Z_j^R\right] = R - n + n(1-g(n))^R$$

Therefore the expected number of Byzantine replicas that will be held culpable during the execution is

$$\begin{aligned}
E[d] &\geq E[X-D] - p_2 R \\
&\geq p_1^{\mathcal{H}} R - (p_1^{\mathcal{A}} + p_2)R - \left(R - n + n(1-g(n))^R\right) - p_2 R \\
&\geq p_1^{\mathcal{H}} R - p_1^{\mathcal{A}} R - 2p_2 R - \left(R - n + n(1-g(n))^R\right) \\
&= (n-2f)g(n)R - 2(w - ng(n))R - (R - n + n(1-g(n))^R) \\
&= (n-2f)g(n)R - T(n,R)
\end{aligned}$$

where $T(n,R) = 2(w-ng(n))R + (R - n + n(1-g(n))^R)$ for short and $T(n,R) > 0$. Then, using a Chernoff bound, we have

$$d = (1-\epsilon)\left((n-2f)g(n)R - T(n,R)\right)$$

except with $\exp(-\Omega(R))$ probability.

E.4 Proof of Corollary 2

Proof. Applying binomial approximation to the probability $g(n)$, we have

$$\begin{aligned}
g(n) &= (1-(1-w)^{1/n})(1-w)^{1-1/n} \\
&= (1-w)^{\frac{n-1}{n}} - (1-w) \\
&\sim (1-w)(1+\frac{w}{n}) - (1-w) \\
&= \frac{w(1-w)}{n} < \frac{1}{n}
\end{aligned}$$

Therefore, when $R/n = o(1)$, $Rg(n) = o(1)$, then we have $E[D] \sim R - ng(n)R$ and $T(n,R) \sim (1-w+3w^2)R$. Further, since $R \geq \kappa$

$$E[d] \sim (w-w^2)(n-2f)\kappa/n - (1-w+3w^2)\kappa$$

Using a Chernoff bound,

$$d = (1-\epsilon)((w-w^2)(n-2f)\kappa/n - (1-w+3w^2)\kappa)$$

except with $\exp(-\Omega(\kappa))$ probability.

Algorithm 3 A player-replaceable, partially synchronous SMR protocol

$t_H \leftarrow \lceil 2\lambda/3 \rceil$
$TC[0] \leftarrow \emptyset$, $Votes[0] \leftarrow \emptyset$
$lockedQC \leftarrow qc_{genesis}$ ▷ the lock variable
for $curRound \leftarrow 1, 2, \ldots$ **do**
▷ **At time 0 of *curRound***
  $TC[curRound] \leftarrow \emptyset$
  $Votes[curRound] \leftarrow \emptyset$
  if $\texttt{VRF}(seed||curRound||\text{''}leader\text{''}) < \tau/n$ **then** ▷ as the leader of $curRound$
    if $\exists h, s.t. |Votes[curRound-1][h]| \geq t_H \leftarrow \emptyset$ **then**
      $lockedQC \leftarrow$ QC generated from $Votes[curRound-1][h]$
    create block b^* where $b^*.justify \leftarrow lockedQC$; $b^*.cmd \leftarrow$ commands from clients; $b^*.parent \leftarrow lockedQC.block$
    broadcast $\langle \text{PROPOSAL}, curRound, b^*, TC[curRound-1] \rangle$
  $m \leftarrow \emptyset$
  upon receiving $m' \leftarrow \langle \text{PROPOSAL}, curRound, b^*, TC[curRound-1] \rangle$ from a leader whose cryptographic sortition is valid **do**
    if $(m = \emptyset) \vee (m'.vrf < m.vrf)$ **then**
      $m \leftarrow m'$ ▷ m is the proposal with the min VRF hash
▷ **At time 2Δ of *curRound***
  $blockHash = \emptyset$
  if m is not empty and $m.b^*$ extends from $lockedQC.block$ **then**
    ProcessProposal(m) (line 30)
    $blockHash = H(m.b^*)$
  if $\texttt{VRF}(seed||curRound||\text{''}committee\text{''}) < \lambda/n$ **then** ▷ as a committee member of $curRound$
    broadcast $\langle \text{VOTE}, curRound, blockHash, lockedQC, TC[curRound-1] \rangle$
  while $|TC[curRound]| < t_H$ or before 4Δ of $curRound$ **do**
    wait for $\langle \text{VOTE}, curRound, h^*, lockedQC, TC[curRound-1] \rangle$ whose cryptographic sortition is valid
▷ **At any time (triggered by receiving a vote)**
upon receiving $\langle \text{VOTE}, r, h^*, lockedQC^*, TC^*[r-1] \rangle$ s.t. sender's cryptographic sortition is valid **do**
  add Vote message to set $Votes[r][h^*]$
  if sender id has no entry in $TC[r]$ **then**
    $TC[r] \leftarrow TC[r] \cup \{(id, lockedQC^*)\}$
    $lockedQC \leftarrow \max_{round}\{lockedQC^*, lockedQC\}$ ▷ do not update in case of a draw
  $TC[r-1] \leftarrow TC[r-1] \cup TC^*[r-1]$ ▷ do not update in case an entry for an id exists
procedure ProcessProposal($\langle \text{PROPOSAL}, r, b^*, TC^*[r-1] \rangle$)
  $lockedQC \leftarrow \max_{round}\{b^*.justify, lockedQC\}$ ▷ do not update in case of a draw
  $TC[r-1] \leftarrow TC[r-1] \cup TC^*[r-1]$ ▷ do not update in case an entry for an id exists
  $b' \leftarrow b^*.parent$, $b \leftarrow b'.parent$
  if b, b', b^* are in consecutive rounds **then**
    finalize block b (directly) and all blocks before b (indirectly), execute commands in the finalized blocks