

MoxieBFT: A Byzantine Consensus Protocol with Event-Completeness Commitments

Moxie Research
research@moxie.inc

October 27, 2025

Abstract

We present MoxieBFT, a HotStuff-family Byzantine consensus protocol for event-conditional settlement under partial synchrony. A valid proposal carries a phase-ordered payload $P_1 \prec P_2 \prec P_3$: Phase 1 resolves events, Phase 2 performs CP-CAM clearing on the post-resolution state, and Phase 3 settles the resulting transfers atomically. The protocol augments the voting kernel with an Event-Collect gate and a threshold-encrypted mempool subject to the commit-before-decrypt invariants CP1-CP4. We prove agreement, validity, view and height monotonicity, locked-QC safety, and liveness post-GST for $n \geq 3f + 1$ with quorum $q(n) = \lceil 2n/3 \rceil$; in particular, a five-validator deployment requires quorum four, not three. Liveness under partial synchrony is obtained by a pacemaker that advances only on current-target-view view-change quorums or quorum-backed NEWVIEW proofs, together with stale-certificate rebroadcast and a timeout parameter that dominates the WAN message and decryption budget. We separate cryptographic roles: BLS12-381 authenticates quorum votes and certificates, Ed25519 plus ML-DSA-65 cross-sign individually authored pacemaker and proposal messages, and the encrypted mempool uses a threshold decryption layer with an optional ML-KEM-768 confidentiality wrapper. A five-round lower bound shows that any protocol satisfying event-completeness, Byzantine safety, and commit-before-decrypt must pay five causal message rounds. The protocol admits a TLA⁺ safety specification and a companion pacemaker-liveness model that serve as formal grounding for the state-machine invariants and the WAN-liveness argument.

Contents

1	Introduction	3
2	System Model	4
2.1	Adversary model	5
2.2	Cryptographic role separation	6
3	Protocol Specification	6
3.1	Phase model	6
3.2	Consensus state	11
3.3	Transition function	12
3.4	Proposal handling	12
3.5	Vote handling	13
3.6	Event-Collect phase	13
3.7	View change	14
3.8	Proposer selection	15

4	Committee Selection	15
5	VRF and Randomness	16
5.1	Construction	16
5.2	RANDAO commit-reveal	16
5.3	Winner selection	17
6	Safety	17
6.1	Agreement	17
6.2	Validity	19
6.3	View monotonicity	19
6.4	Height monotonicity	19
6.5	Locked QC safety	20
6.6	Vote bound	21
7	Liveness	22
8	Consensus Pipelining	23
9	MEV Resistance	23
9.1	Threshold encryption	23
9.2	Ordering commitment protocol	25
9.3	Attack taxonomy and mitigation	25
9.4	Formal MEV model	25
9.5	Residual MEV: cross-block repositioning	26
10	Commit-Precondition Properties CP1-CP4	27
10.1	Five-Round Message-Complexity Lower Bound	33
11	Parallel Decryption	38
12	Batch BLS Aggregation	39
13	Event-Atomic Settlement	40
14	Security Budget	43
14.1	Slashing dilution floor	44
15	TLA⁺ Formal Grounding	45
16	Related Work	46
16.1	Classical and pipelined BFT consensus	46
16.2	DAG-based mempool and consensus	47
16.3	MEV-resistant mempool architectures	48
16.4	Composition Theorem	48
17	Open Problems	51
18	Conclusion	52

1 Introduction

Byzantine fault-tolerant (BFT) consensus protocols for blockchain settlement [1, 2, 3] treat block content as an opaque payload assembled by the leader. The consensus layer indifferently orders whatever the leader proposes. This is adequate for general-purpose state-machine replication, where execution follows consensus on a finalized order. It is inadequate for settlement systems whose semantics demand two further properties: that the block contains a complete, quorum-attested set of event resolutions on which the included transactions depend, and that the ordering of encrypted transactions be committed before their plaintexts can be observed.

Event-completeness. A block that settles event-conditional positions must carry a quorum-attested resolution of the underlying events. Under the leader-as-block-author model, a Byzantine leader can selectively omit attestations, delay resolution, or fabricate outcomes; the protocol admits no structural guarantee of completeness.

Frontrunning and resequencing. A validator who learns the plaintext of an encrypted order before the block is finalized may insert profitable trades that reorder the unsettled set. Threshold encryption of the mempool [6, 9] mitigates plaintext leakage, but requires precise integration with the consensus pipeline: decryption shares must be released only after the executed ordering is committed, and share release must be bound to the authoritative committed ordering.

Contribution. We present MoxieBFT as a systems-composition result. The three constituent components (HotStuff-2 [7] as Byzantine-consensus kernel, an oracle-attestation phase, and a commit-before-decrypt threshold-encrypted mempool adopting the consensus-layer pattern of F3B [9] under threshold hashed-ElGamal in G_1 of BLS12-381 [17]) are individually prior art. Our contribution is their composition under the Commit-Precondition invariants CP1-CP4, the five-round lower bound for the resulting class, and the composition theorem that discharges the end-to-end guarantees from the per-component guarantees.

The contributions, by section, are:

- (i) A six-phase BFT state machine (§3) that augments the HotStuff-2 voting pipeline with an oracle-attestation phase producing quorum-backed event-resolution certificates committed into block headers, together with a block-validity predicate that enforces the consensus-ordered payload discipline Phase 1 (event resolution) \rightarrow Phase 2 (CPCAM clearing) \rightarrow Phase 3 (atomic settlement).
- (ii) Proofs of safety (agreement, validity, view monotonicity, height monotonicity, locked-QC safety, vote bound; §6) and liveness under partial synchrony (§7). The agreement argument is factored through the HotStuff-2 unlocking lemma (Lemma 6.1), and the liveness argument makes the current-target-view pacemaker discipline explicit.
- (iii) A threshold-encrypted mempool with the CP1-CP4 commit-reveal-execute invariants (§10) and a frontrun-freeness theorem (Theorem 10.9) against the composed adversary $\mathcal{A}_{\text{MBFT}}$, with a separated cryptographic-role assignment for individually signed pacemaker messages, aggregated quorum certificates, and encrypted mempool confidentiality.
- (iv) A five-message-round lower bound $r(\Pi) \geq 5$ for the protocol class $\mathcal{P}_{\text{CP,BFT,EA}}$ satisfying CP1-CP4, $f < n/3$ Byzantine fault tolerance, and event-atomic settlement

(Theorem 10.17); the bound is established by four pairwise-merge impossibility lemmas and is tight on MoxieBFT under fusion of EVENT-COLLECT with PROPOSE at gate G_1 (Remark 10.18).

- (v) A composition theorem (Theorem 16.2) discharging agreement, liveness with bound $t_{ec} + 5\Delta + c\Delta$, frontrun-freeness, and observer opacity from the per-component guarantees of HotStuff-2, threshold hashed-ElGamal IND-CPA, and oracle-attestation liveness, under the CP1-CP4 invariants and the composed adversary $\mathcal{A}_{\text{MBFT}}$ of §16.4.
- (vi) A TLA⁺ safety specification for the six-phase state machine and a companion pacemaker-liveness model (§15) that provide formal grounding for the agreement, monotonicity, and current-view view-change invariants used throughout the paper.

The protocol specification is a deterministic pure function $(\text{state}, \text{message}, \text{time}) \mapsto (\text{state}', \text{actions})$. Every safety property is checked on every transition. All claims are stated against this state-machine specification and against the adversary models of §2.1 and §16.4.

2 System Model

Definition 2.1 (System). A *MoxieBFT system* consists of:

- A finite set of validators $\mathcal{V} = \{v_1, \dots, v_n\}$, where $n = |\mathcal{V}| \geq 3f + 1$ for a resilience parameter $f \in \mathbb{N}$.
- A partition $\mathcal{V} = \mathcal{H} \cup \mathcal{B}$, where \mathcal{H} is the set of *honest* validators and \mathcal{B} is the set of *Byzantine* validators, with $|\mathcal{B}| \leq f$.
- A point-to-point network with eventual delivery: for any message m sent by an honest validator at time t , there exists a delivery time $t' \geq t$ such that every honest validator receives m by time t' .
- A global stabilization time GST after which all messages between honest validators are delivered within a known bound Δ .

Definition 2.2 (Partial synchrony). The network is *partially synchronous*: there exists an unknown $\text{GST} \in \mathbb{R}_{\geq 0}$ and a known $\Delta > 0$ such that for all $t \geq \text{GST}$ and all honest validators v_i, v_j , any message sent by v_i at time t is received by v_j by time $t + \Delta$.

Definition 2.3 (View and height). A *view* $r \in \mathbb{N}$ indexes a round of the consensus protocol. A *height* $h \in \mathbb{N}$ indexes a position in the committed chain. Each view corresponds to exactly one block proposal attempt. Views advance monotonically; heights advance monotonically upon block commitment.

Definition 2.4 (Quorum). A *quorum* is a set $Q \subseteq \mathcal{V}$ with $|Q| \geq \lceil 2n/3 \rceil$. We write $q(n) := \lceil 2n/3 \rceil$ for this bound.

Remark 2.5 (Canonical shorthand). When $n = 3f + 1$ we have $q(n) = 2f + 1$, so the standard HotStuff notation $2f + 1$ is a canonical special case of $q(n)$. Throughout the paper, any quorum threshold written as $2f + 1$ should be read as $q(n)$ outside the canonical $n = 3f + 1$ regime. In particular, for a five-validator committee, $q(5) = \lceil 10/3 \rceil = 4$.

Proposition 2.6 (Quorum intersection). *For any two quorums Q_1, Q_2 , we have $Q_1 \cap Q_2 \cap \mathcal{H} \neq \emptyset$ whenever $n \geq 3f + 1$.*

Proof. $|Q_1 \cap Q_2| \geq |Q_1| + |Q_2| - n \geq 2\lceil 2n/3 \rceil - n$. We verify the inequality $2\lceil 2n/3 \rceil - n \geq f + 1$ for each residue class of n modulo 3 satisfying $n \geq 3f + 1$:

- $n = 3f + 1$: $2(2f + 1) - (3f + 1) = f + 1$.
- $n = 3f + 2$: $2(2f + 2) - (3f + 2) = f + 2 \geq f + 1$.
- $n = 3f + 3$: $2(2f + 2) - (3f + 3) = f + 1$.

In each residue class $|Q_1 \cap Q_2| \geq f + 1$. Since $|\mathcal{B}| \leq f$, at least one member of $Q_1 \cap Q_2$ is honest. \square

2.1 Adversary model

We fix a single adversary \mathcal{A} against which all safety and secrecy claims below are stated. The system-level adversary is a polynomial-time algorithm endowed with the capabilities enumerated below. A composition-analysis adversary $\mathcal{A}_{\text{MBFT}}$ (a specialization of \mathcal{A} to the intersection of component adversary models used in the composition setting) is defined in §16.4; the two coincide on all capability coordinates except corruption timing, where $\mathcal{A}_{\text{MBFT}}$ adopts the bounded-adaptive window of Remark 2.8.

Definition 2.7 (Byzantine adversary \mathcal{A}). \mathcal{A} is a probabilistic polynomial-time algorithm with the following capabilities:

- (A1) **Static corruption.** Before the protocol starts, \mathcal{A} selects a set $\mathcal{B} \subseteq \mathcal{V}$ of Byzantine validators with $|\mathcal{B}| \leq f = \lfloor (n - 1)/3 \rfloor$. Once fixed, the set is immutable for the entire execution horizon considered in the claim. Dynamic / adaptive corruption is outside our system model; see Remark 2.8 below for the bounded relaxation we adopt in §16.4.
- (A2) **Byzantine control.** \mathcal{A} has full control of each validator $v \in \mathcal{B}$: arbitrary deviation from protocol, arbitrary message contents subject only to signing-key constraints, arbitrary timing of messages. Byzantine validators possess their own signing keys sk_v ; honest validators' keys remain secret.
- (A3) **Network scheduling (post-GST).** After the global stabilization time GST (unknown to honest validators, known to \mathcal{A}), \mathcal{A} may schedule message delivery subject only to the bound $\text{delay}(m) \leq \Delta$ of Definition 2.2. The adversary may reorder concurrent messages arbitrarily and may delay any message by up to Δ , but may not exceed Δ .
- (A4) **Network scheduling (pre-GST).** Before GST, \mathcal{A} may delay messages arbitrarily (no bound). Liveness does not hold in this regime; safety claims must.
- (A5) **Computational assumptions.** \mathcal{A} is probabilistic polynomial time in the security parameter λ . The following hardness assumptions bound \mathcal{A} 's cryptographic advantage:
 - *BLS unforgeability*: BLS12-381 signatures are existentially unforgeable under chosen-message attack, reducing to co-CDH on (G_1, G_2, G_T, e) [8].
 - *Threshold hashed-ElGamal IND-CPA*: the mempool primitive of Definition 9.1 is IND-CPA-secure under CDH-with-KDF-in-ROM (equivalently DDH without ROM) in G_1 [17].
 - *Post-quantum signature hardness (optional)*: ML-DSA-65 (FIPS 204) is existentially unforgeable under chosen-message attack, reducing to module-lattice SIS / LWE. Required only for deployments specifying the hybrid-classical-plus-PQ signature mode of §4; the base classical mode does not depend on it.
 - *Post-quantum KEM hardness (optional)*: ML-KEM-768 is IND-CCA2-secure under module-LWE. Same scope as above.
 - *Collision resistance of H_{dom}* : the domain-separated hash of Definition 10.3 is collision-resistant; where the security reduction invokes random-oracle-ness of H_{dom} , that usage is stated explicitly at the call site.

- (A6) **Key-exposure bound.** A validator’s threshold-decryption share sk_i and consensus signing key sk_i^{BLS} are exposed to \mathcal{A} iff $v_i \in \mathcal{B}$. Honest-validator keys remain confidential throughout the execution; honest validators zeroize secret material after use.
- (A7) **Rushing.** \mathcal{A} may choose its messages after observing honest validators’ current-round messages, subject to the pre-/post-GST delivery bounds above.

Remark 2.8 (Adaptive corruption: scope and bound). Definition 2.7 fixes the corruption set statically. A strictly stronger (and widely-studied) model is *adaptive corruption*, in which \mathcal{A} selects new corruptions during execution, possibly targeting the next proposer or the current decryption-share holders. MoxieBFT’s safety and liveness statements in Sections 6-7 are proved in the static-corruption model. A bounded-adaptive extension is introduced in §16.4 ($T_{\text{window}} = 1$ view, forbidding within-view adaptive corruption), under which the static-model proofs apply verbatim. Fully adaptive safety is outside the scope of this paper.

Remark 2.9 (Out of scope). The adversary model does *not* cover: (i) denial-of-service attacks on non-validator clients (out of scope: these do not affect Byzantine safety by Liveness pre-GST); (ii) physical-layer attacks on validator hardware (RowHammer, side channels on the TRNG used for key generation; covered elsewhere by hardware-security assumptions); (iii) social-layer attacks on governance (bribery of off-chain decision makers; covered elsewhere by the slashing-dilution economic argument, not the BFT proof); (iv) adversarial stake concentration beyond f (outside the threat model by assumption; enforced operationally by staking-cap constraints).

2.2 Cryptographic role separation

The protocol logic is independent of any single concrete signature suite, but the roles are not interchangeable:

- (C1) **Individually authored pacemaker and proposal messages.** A proposer, a VIEWCHANGE sender, and a NEWVIEW leader may attach a cross-signature $\sigma_i^{\text{ind}} = (\sigma_i^{\text{Ed25519}}, \sigma_i^{\text{MLDSA}})$. The classical limb authenticates the message under current assumptions; the ML-DSA-65 limb provides post-quantum hedge security. The consensus proofs require only existential unforgeability for the individually signed channel.
- (C2) **Quorum-authenticated votes and certificates.** Votes, quorum certificates, event-resolution certificates, and CP3 share-release attestations are authenticated under BLS12-381 so that a quorum of signatures compresses to a constant-size certificate. This aggregation property is load-bearing for the linear-message HotStuff-family pipeline.
- (C3) **Encrypted mempool confidentiality.** The commit-before-decrypt path uses a threshold encryption layer for consensus-bound secrecy. An optional ML-KEM-768 wrapper can carry the DEM key or its re-encryption material across committee members without changing the safety proof: secrecy against early plaintext recovery still rests on the threshold layer and the CP1-CP4 gate structure.

3 Protocol Specification

3.1 Phase model

MoxieBFT augments the HotStuff-2 three-phase voting pipeline with a prepended EVENT-COLLECT phase and an explicit IDLE state between blocks. The resulting state machine comprises six phases.

Definition 3.1 (Consensus phases). The consensus phase of a validator is an element of:

$$\Phi = \{\text{IDLE}, \text{EVENTCOLLECTING}, \text{PREPARE}, \text{PRECOMMIT}, \text{COMMIT}, \text{DECIDE}\}$$

The linear progression within a successful round is:

$$\text{IDLE} \xrightarrow{\text{proposal}} \text{EVENTCOLLECTING} \xrightarrow{\text{timeout/early quorum}} \text{PREPARE} \xrightarrow{2f+1} \text{PRECOMMIT} \xrightarrow{2f+1} \text{COMMIT} \xrightarrow{2f+1} \text{D}$$

Definition 3.2 (Vote phases). The four voting phases correspond to the four HotStuff-derived phases:

$$\Phi_{\text{vote}} = \{\text{PREPARE}, \text{PRECOMMIT}, \text{COMMIT}, \text{DECIDE}\}$$

Each vote phase $\varphi \in \Phi_{\text{vote}}$ has a bijective correspondence with the consensus phase of the same name. The mapping $\pi : \Phi \setminus \{\text{IDLE}, \text{EVENTCOLLECTING}\} \rightarrow \Phi_{\text{vote}}$ is the identity on names.

The role of each phase is:

1. **EVENT-COLLECT.** The proposer aggregates pending oracle attestations, enforces a per-event quorum threshold, and assembles the attestation certificate QC_{EC} , an aggregate BLS signature over the block's event-resolution set, which is bound into the block header. Events that fail to reach quorum are deferred with escalation to a higher oracle tier (§3.6).
2. **PREPARE.** The leader broadcasts a proposal carrying the parent hash, event-resolution set, ordering commitment $R_{\text{committed}}$, VRF output and proof. Validators verify the proposal against the leader schedule, the block-commitment hashes, the attestation-certificate quorum, and the lock compatibility predicate, and broadcast PREPARE votes.
3. **PRECOMMIT.** On $2f+1$ PREPARE votes validators form $\text{QC}_{\text{Prepare}}$, set lockedQC to it, and broadcast PRECOMMIT votes. The lock is the central safety device: a locked validator votes only for proposals that extend the locked chain or carry a justification QC of strictly higher view than the lock.
4. **COMMIT.** On $2f+1$ PRECOMMIT votes validators form $\text{QC}_{\text{PreCommit}}$ and broadcast COMMIT votes. Threshold decryption shares are released in this phase, attested under the authoritative $R_{\text{committed}}$ fixed at $\text{QC}_{\text{Prepare}}$.
5. **DECIDE.** On $2f+1$ COMMIT votes threshold decryption completes, each validator recomputes R_{executed} from the revealed plaintext sequence, and provided $R_{\text{executed}} = R_{\text{committed}}$ broadcasts a DECIDE vote binding the final state root. On $2f+1$ DECIDE votes the block commits: height and view advance and the machine returns to IDLE.

Remark 3.3 (Phase count vs. round count). The state machine has six phases IDLE, EVENT-COLLECTING, PREPARE, PRECOMMIT, COMMIT, DECIDE, but the number of network rounds on the happy path is five. The fused PROPOSE/EVENTCOLLECTING gate is a single leader-to-all broadcast (the proposal message carries both QC_{EC} and $R_{\text{committed}}$), so the six-phase state machine executes in five sequential message rounds after GST (cf. Remark 10.18). The IDLE phase is a null-duration state between blocks and does not contribute a round.

Definition 3.4 (Block time budget). The total block interval is

$$t_{\text{block}} = t_{\text{ec}} + t_{\text{prep}} + t_{\text{pc}} + t_{\text{com}} + t_{\text{dec}} + t_{\text{decrypt}} + t_{\text{exec}},$$

where t_{ec} is the Event-Collect budget, $t_{\text{prep}}, t_{\text{pc}}, t_{\text{com}}, t_{\text{dec}}$ are the per-voting-phase maximum round-trip budgets, t_{decrypt} is the threshold-decryption budget of Definition 9.2, and t_{exec} is the deterministic-execution budget. The protocol does not fix numeric values for these budgets: any deployment is required only to satisfy

$$t_{\text{block}} \geq 2\text{GST} + \sum_{\varphi \in \Phi_{\text{vote}}} \ell_{\text{max}}(\varphi) + t_{\text{ec}} + t_{\text{decrypt}} + t_{\text{exec}},$$

where $\ell_{\text{max}}(\varphi)$ is the post-GST message-delivery bound governing phase φ . The inequality is the liveness-admissibility constraint: under it, Theorem 7.1 guarantees commit within one block interval after GST. Concrete budgets are a deployment-configuration choice, not a property of the protocol specification, and are outside the scope of this paper.

Phase-ordered block validity

Consensus authenticates a block hash together with a typed payload discipline. Every valid proposal carries an ordered decomposition $B = (P_1, P_2, P_3)$ where Phase 1 resolves events, Phase 2 performs CPCAM clearing and any other state transitions whose prices depend on the resolved event set, and Phase 3 settles the resulting transfers and residual state updates.

Definition 3.5 (Phase-valid block). A proposed block B is *phase-valid* if:

- (a) every transaction in P_1 is an event-resolution or event-settlement action whose inputs are determined solely by the parent state and the QC_{EC} -attested outcome set;
- (b) every transaction in P_2 is a CPCAM-clearing or trading action whose validation is performed against the post- P_1 state, so no Phase 2 action can reference an unresolved event;
- (c) every transaction in P_3 is a settlement, transfer, or residual post-clearing action whose validation is performed against the post- P_2 state; and
- (d) no transaction classified for a later phase appears before a transaction classified for an earlier phase.

Proposition 3.6 (Out-of-order proposals fail validation). *If a proposed block fails Definition 3.5, then no honest validator emits a PREPARE vote for it.*

Proof. Phase validity is part of proposal validation in §3.4. Any proposal that places a Phase 2 CPCAM action before its enabling Phase 1 event resolution, or a Phase 3 settlement action before the Phase 2 clearing state on which it depends, is rejected before the first vote is cast. Consensus therefore enforces the ordering Phase 1 \rightarrow Phase 2 \rightarrow Phase 3 as a consensus safety property. \square

Protocol class and the five-round lower bound

We now state and prove a message-round lower bound for the protocol class MoxieBFT instantiates. The lower bound is not a property of MoxieBFT's specific implementation; it is a property of any protocol simultaneously achieving the EVENT-COLLECT completeness guarantee, HotStuff-2 three-phase safety, and the CP1-CP4 commit-reveal-execute discipline of Section 10. It shows that MoxieBFT's five rounds are unavoidable under these preconditions.

Definition 3.7 (Protocol class Π_{CEP}). A consensus protocol π belongs to the class Π_{CEP} (Commit-Event-Pipelined) if each of its one-block executions satisfies:

- (a) **Event-completeness.** For every block B at height h with committed event-resolution certificate $C_e(h)$, every honest validator enters the voting-phase cascade for B only after it has received the aggregated attestation-quorum for $C_e(h)$. Equivalently: the set of messages on which the first voting action for B causally depends includes the $q(n) = \lceil 2n/3 \rceil$ attestation broadcast.
- (b) **HotStuff-2 three-phase safety.** There exist three voting phases $\varphi_1, \varphi_2, \varphi_3$ (Prepare, PreCommit, Commit) such that (i) a validator votes in φ_2 only after observing a $(2f + 1)$ -quorum certificate on φ_1 , and similarly for φ_3 after φ_2 ; (ii) a validator enters its locked state after φ_2 's quorum and unlocks only on a strictly higher-view φ_1 -QC.
- (c) **CP1-CP4 commit-reveal-execute.** Per Section 10: the ordering root $R_{\text{committed}}$ is attested at end of φ_1 ; decryption-share release uses the epoch-stable encryption domain label (CP3(a)) and is consensus-bound to a finalized φ_1 -QC via signed attestations and a state-machine gate (CP3(b)-(d)); and committed execution matches the committed root (CP4).

Theorem 3.8 (Five-round consensus-latency lower bound). *Let $\pi \in \Pi_{\text{CEP}}$ be any protocol in the Commit-Event-Pipelined class. Let ℓ_{med} be the median network latency. Then, in the worst case, committing and executing one block requires at least five sequential network message rounds of duration ℓ_{med} : formally, the critical path in any execution of π over one block contains at least five edges, each corresponding to a message dispatch whose receipt is causally required before the next dispatch can occur.*

The five rounds correspond to five causally-ordered gates G_1, \dots, G_5 (introduced formally in Theorem 10.17 and Definition 10.12): G_1 Propose (the leader broadcasts a block whose header simultaneously carries QC_{EC} , the oracle-attestation quorum for the block's event-resolution set, and the Merkle-root commitment $R_{\text{committed}}$; Event-Collect is fused with Propose), G_2 Prepare (the φ_1 -vote round), G_3 PreCommit (the φ_2 -vote round that forms the lock), G_4 Commit-decrypt (the φ_3 -vote round concurrent with decryption share release), and G_5 Decide (the final commit round binding R_{executed}).

Proof. We give a causal-dependency argument. For any protocol $\pi \in \Pi_{\text{CEP}}$ and any one-block execution σ , let $G(\sigma)$ be the message-causality DAG whose vertices are the message-send events in σ and whose edges are causal dependencies: $e = (m_1 \rightarrow m_2)$ iff the send of m_2 is in the same honest validator's execution and follows the receipt of m_1 (receive-before-send order on a single honest process). The block-commit event at any honest validator is a sink in $G(\sigma)$. The network-round complexity of σ is the length of the longest path in $G(\sigma)$.

We identify five vertices m_1, \dots, m_5 , one per gate G_1, \dots, G_5 , that lie on a common causal path terminating at commit:

Round 1 (Propose, m_1 , gate G_1). By Definition 3.7(a), any honest validator's first voting action for block B at height h is causally preceded by its receipt of the oracle-attestation quorum for the block's event-resolution set. Under the protocol class Π_{CEP} , we permit the attestations to ride into the block header simultaneously with the proposal broadcast: the leader assembles both the attestation set $C_e(h)$ and $R_{\text{committed}}$ and broadcasts them in a single proposal message m_1 . The fusion of Event-Collect with Propose into G_1 is consistent with the upper-bound tightness statement of Remark 10.18; any alternative that separates attestation broadcast from proposal merely lengthens the path. Hence m_1 is the proposal broadcast, and its receipt at each validator is required before any φ_1 -vote may be cast.

Round 2 (Prepare, m_2 , gate G_2). A validator casts a φ_1 -vote (m_2) only after receiving the proposal broadcast m_1 . Hence $m_1 \rightarrow m_2$ in $G(\sigma)$ along the validator's process-order edge (receive-before-send).

Round 3 (PreCommit, m_3 , gate G_3). A φ_2 -vote (m_3) is cast only after the validator observes a φ_1 -QC, which requires the validator to have first sent (and aggregated) φ_1 -votes in response to m_1 . Hence $m_2 \rightarrow m_3$.

Round 4 (Commit-decrypt, m_4 , gate G_4). A validator casts a φ_3 -vote (m_4) only after observing a φ_2 -QC, whose formation requires φ_2 -votes previously exchanged, each of which in turn required the φ_1 -QC. Hence $m_3 \rightarrow m_4$ along the process-order chain φ_2 -QC \rightarrow φ_3 -vote terminating at each validator; this contributes exactly one network round on the critical path (the intermediate QC formations are local computations once enough votes have been delivered). In the same gate, each validator releases its decryption share (under the epoch-stable encryption identity $\text{id}_{\text{enc}}(h, E)$ of Definition 10.3) together with the scheduling tag and attestation a_i of Definition 10.5(b); share release is ordered after the PreCommit QC by CP3(c)'s state-machine gate and by the scheduling-tag timing-tightness property of Proposition 10.4(iv). Share release and φ_3 -vote are concurrent within the gate and share the same network round.

Round 5 (Decide, m_5 , gate G_5). After share combination yields the plaintext, each validator recomputes R_{executed} from the decrypted transaction sequence and compares to $R_{\text{committed}}$ (CP4). Upon match, each validator broadcasts a φ_4 -vote (the DECIDE vote, m_5) that binds the executed state root. The block-commit event at any honest validator is causally after m_5 . By within-round causality $m_4 \rightarrow m_5$ in $G(\sigma)$.

Combining, we have a causal chain $m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow m_4 \rightarrow m_5 \rightarrow \text{commit}$ in $G(\sigma)$ with five message-round edges, establishing that the critical path has length ≥ 5 . Multiplying by ℓ_{med} gives the consensus-latency lower bound.

Tightness of the five-round count (no protocol in Π_{CEP} can save a round). We dispose of the four candidate shortcuts:

Shortcut A: execute before decrypt. Execution without plaintext access violates CP4. Formally, suppose π executes before decrypt and R_{executed} is a polynomial-time function f of the committed ciphertext set $\{\tilde{c}_1, \dots, \tilde{c}_n\}$. Then any polynomial-time observer running the same f on the same ciphertexts obtains the same R_{executed} as honest validators. For CP4 to hold ($R_{\text{executed}} = R_{\text{committed}}$), f must recover the plaintext-dependent execution result of each transaction in the sequence. Either (i) f recovers the plaintext orderings of at least one ciphertext with non-negligible probability, in which case f constitutes a polynomial-time IND-CPA distinguisher against threshold hashed-ElGamal in G_1 (violating CP2 under CDH-with-KDF-in-ROM / DDH in G_1); or (ii) f is independent of the underlying plaintexts and produces $R_{\text{executed}} \neq R_{\text{committed}}$ with non-negligible probability (violating CP4). In both cases the shortcut is ruled out by reduction to CP2's IND-CPA assumption.

Shortcut B: decrypt before commit. If the decrypt step is scheduled before the φ_3 -QC (equivalently, if CP3's state-machine gate of Definition 10.5(c) is weakened), the adversary can induce t validators to release shares and attestations a_i with tags bound to a stale or forged $R'_{\text{committed}} \neq R_{\text{committed}}$, extracting plaintext prior to φ_1 -QC finalization. Of the t validators, at most f are Byzantine under the composed adversary model of §16.4; the remaining $t - f \geq f + 1$ are honest. Each honest validator releasing such a share has either (i) violated the state-machine gate (Definition 10.5(c)), contradicting specification-conforming behaviour, or (ii) signed an attestation a_i over a root not attested by any QC_{Prepare}, generating slashable evidence (Definition 10.5(d)). In the pure-cryptographic regime, the f -bounded Byzantine coalition cannot assemble t valid decryption shares $ds_i = sk_i \cdot C_1$ on the ciphertext's ephemeral element $C_1 \in G_1$ without crossing the IND-CPA threshold of threshold hashed-ElGamal ([17, 18]). Relaxing CP3's state-machine gate therefore either relies on honest validators violating their specification (impossible in the honest-majority model) or on Byzantine validators assembling $\geq t$ shares (a cryptographic impossibility); in both cases the shortcut is ruled out at the consensus layer.

Shortcut C: commit without propose. A commit event in a HotStuff-2-class protocol requires a chain φ_1 -QC- φ_2 -QC- φ_3 -QC over a block whose payload includes $R_{\text{committed}}$. Without a prior φ_1 -QC there is no signed attestation of the ordering root to which subsequent phases reference; the φ_2 and φ_3 quorums would then have nothing to commit on.

Shortcut D: merge two rounds into one via aggregation. Aggregation (e.g., threshold-signing φ_2 and φ_3 votes into a single message) does not reduce the causal-path length. The φ_3 -vote is semantically a function of the φ_2 -QC, and a validator can only sign the φ_3 -QC portion after it has received enough φ_2 -votes to reconstruct the φ_2 -QC, which itself is a $(2f + 1)$ -threshold of messages requiring at least one round of dispatch from voters to the aggregator. In the standard model any such pre-aggregation requires a pre-commitment, which is itself a message round.

The four shortcuts are exhaustive. We conclude that $5 \cdot \ell_{\text{med}}$ is a tight lower bound on the worst-case consensus latency of any $\pi \in \Pi_{\text{CEP}}$. \square

Remark 3.9 (Relation to Theorem 10.17). Theorems 3.8 and 10.17 both deliver the numeric claim $r(\Pi) \geq 5$ via the identical five-gate enumeration G_1, \dots, G_5 but illuminate different adversary classes. Theorem 3.8 is a causal-DAG argument over a single protocol execution: it rules out four named shortcuts within one run of a fixed protocol $\pi \in \Pi_{\text{CEP}}$. Theorem 10.17 is the primary statement of the bound over the class $\mathcal{P}_{\text{CP,BFT,EA}}$: it proceeds by four pairwise-merge impossibility lemmas and rules out any round merging across the entire class. Both adopt the convention that Event-Collect is fused with Propose at gate G_1 (Remark 10.18); neither counts attestation broadcast as an additional round. Theorem 10.17 is the class-level result of record; Theorem 3.8 is retained as a complementary execution-level argument because the causal-DAG framing makes the cryptographic-reduction content of the four shortcut classes (execute-before-decrypt, decrypt-before-commit, commit-without-propose, aggregation-merge) explicit in a form the merge-lemma proof does not.

Definition 3.10 (Consensus latency lower bound, restated as a corollary). For MoxieBFT, which satisfies Event-completeness (Definition 3.7(a)), HotStuff-2 three-phase safety (Definition 3.7(b)), and CP1-CP4 commit-reveal-execute (Definition 3.7(c)), the worst-case consensus latency is at least $5 \cdot \ell_{\text{med}}$, corresponding to the five gates G_1, \dots, G_5 (Propose with Event-Collect fused, Prepare, PreCommit, Commit-decrypt, Decide). This is an immediate consequence of Theorem 3.8.

3.2 Consensus state

Definition 3.11 (Validator state). Each validator v_i maintains a state tuple:

$$\sigma_i = (\text{view}_i, h_i, \text{phase}_i, \text{lockedQC}_i, \text{highQC}_i, P_i, V_i, A_i)$$

where:

- $\text{view}_i \in \mathbb{N}$: current view number.
- $h_i \in \mathbb{N}$: next height to commit.
- $\text{phase}_i \in \Phi$: current consensus phase.
- $\text{lockedQC}_i \in \text{QC} \cup \{\perp\}$: the quorum certificate at which the validator is locked (set upon PRECOMMIT quorum).
- $\text{highQC}_i \in \text{QC}$: the highest quorum certificate known to the validator.
- $P_i \in \text{Proposal} \cup \{\perp\}$: buffered proposal for the current round.

- $V_i : \Phi_{\text{vote}} \rightarrow 2^{\mathcal{V}}$: per-phase set of validators from whom votes have been received.
- $A_i : \text{EventId} \rightarrow \text{AttestationAccumulator}$: pending oracle attestations per event.

Definition 3.12 (Quorum certificate). A *quorum certificate* (QC) is a tuple:

$$\text{QC} = (\varphi, B, d, h, r, S, \sigma_{\text{agg}})$$

where $\varphi \in \Phi_{\text{vote}}$ is the vote phase, B is the block hash, d is the certified digest (equals B for pre-reveal phases; binds the final state root for DECIDE), h is the height, r is the view, $S \subseteq \mathcal{V}$ is the set of signers with $|S| \geq 2f + 1$, and σ_{agg} is the aggregated BLS signature.

Definition 3.13 (QC rank). The *rank* of a quorum certificate is the lexicographic triple:

$$\text{rank}(\text{QC}) = (h, \varphi, r)$$

where the ordering on Φ_{vote} is $\text{PREPARE} < \text{PRECOMMIT} < \text{COMMIT} < \text{DECIDE}$. A QC q_1 is *newer* than q_2 if $\text{rank}(q_1) > \text{rank}(q_2)$.

3.3 Transition function

The protocol is specified as a deterministic transition function:

$$\delta : (\Sigma, \mathcal{M}, \mathbb{R}_{\geq 0}) \rightarrow (\Sigma, \mathcal{A}^*)$$

where Σ is the state space, \mathcal{M} is the message space, and \mathcal{A}^* is a sequence of output actions (broadcast vote, commit block, start timer, etc.). The function is pure: it has no side effects beyond the returned state and action list.

Definition 3.14 (Messages). The message space is:

$$\mathcal{M} = \{\text{PROPOSAL}, \text{EVENTATTESTATION}, \text{VOTE}, \text{VIEWCHANGE}, \text{NEWVIEW}\}$$

where:

- $\text{PROPOSAL}(v, B, h, \text{QC}_{\text{just}}, \sigma)$: block proposal from leader v with justification QC.
- $\text{EVENTATTESTATION}(e, v, r, t, d, \sigma)$: oracle attestation for event e from validator v at view r .
- $\text{VOTE}(v, \varphi, B, d, h, r, \sigma)$: vote from validator v in phase φ .
- $\text{VIEWCHANGE}(v, r', \text{QC}_{\text{high}}, \sigma)$: view-change request to view r' .
- $\text{NEWVIEW}(v, r', \text{QC}_{\text{high}}, \{vc_1, \dots\}, \sigma)$: new-view announcement from the designated leader for view r' , carrying the quorum of view-change messages.

The symbol σ is generic at the message-grammar level; the concrete instantiation follows Section 2.2: votes and certificates use BLS12-381, while individually authored pacemaker and proposal messages may be cross-signed by an Ed25519-ML-DSA pair.

3.4 Proposal handling

Upon receiving a $\text{PROPOSAL}(v, B, h, \text{QC}_{\text{just}}, \sigma)$, an honest validator v_i performs the following checks:

1. **Membership:** $v \in \mathcal{V}$.
2. **Height:** $h \geq h_i$.
3. **View:** the proposal's view matches the validator's current view.
4. **Leader:** v is the expected proposer for the current view (see §3.8).
5. **Block commitments:** the block hash, transaction root, event resolution root, and attestation certificates all verify against deterministic recomputation.
6. **Phase validity:** the proposed payload satisfies Definition 3.5, so the block respects the consensus-enforced ordering Phase 1 (event resolution) \rightarrow Phase 2 (CPCAM clearing) \rightarrow Phase 3 (atomic settlement).
7. **Justification QC:** QC_{just} has valid signers from \mathcal{V} meeting quorum, with a valid aggregate BLS signature.
8. **Locked QC compatibility:** if $\text{lockedQC}_i \neq \perp$, then either $B.\text{parent} = \text{lockedQC}_i.\text{hash}$ (the proposal extends the locked chain) or $\text{QC}_{\text{just}}.\text{view} > \text{lockedQC}_i.\text{view}$ (the justification supersedes the lock).
9. **VRF proof:** if the block carries a VRF output and proof, verify the ECVRF proof against the proposer's registered public key and the canonical VRF input $\text{SHA-256}(B.\text{parent} || r || \text{"MOXIEBFT"})$.
10. **No equivocation:** the validator has not seen a different proposal from the same proposer at the same (h, r) .

If all checks pass, the validator buffers the proposal and enters `EVENTCOLLECTING` with a deadline timer.

3.5 Vote handling

Upon receiving a $\text{VOTE}(v, \varphi, B, d, h, r, \sigma)$:

1. Verify $v \in \mathcal{V}$, $r = \text{view}_i$, $B = P_i.\text{hash}$, $h = P_i.\text{height}$.
2. Verify the BLS signature σ .
3. Detect equivocation: if v previously voted with a different (B, d) at the same (h, r, φ) , emit evidence and reject.
4. Accumulate the vote: $V_i[\varphi] \leftarrow V_i[\varphi] \cup \{v\}$.
5. If $|V_i[\varphi]| \geq 2f + 1$ for a single certified digest d , form a QC and advance:
 - `PREPARE` \rightarrow `PRECOMMIT`: set highQC_i , broadcast `PRECOMMIT` vote.
 - `PRECOMMIT` \rightarrow `COMMIT`: lock. Set lockedQC_i to the new QC and broadcast `COMMIT` vote.
 - `COMMIT` \rightarrow `DECIDE`: begin block execution and threshold decryption.
 - `DECIDE` \rightarrow `IDLE`: commit. Record the block, update lockedQC_i , advance $h_i \leftarrow h + 1$ and $\text{view}_i \leftarrow r + 1$, clear state.

3.6 Event-Collect phase

Definition 3.15 (Event resolution). An *event resolution* for event e is a pair (e, d) where d is the outcome hash admitted by the event’s tier-specific evidence rule. The object signed into the block is always a validator quorum certificate $QC_{EC}(e, d, tier, H_{evidence})$. Primary evidence may originate outside validator execution. At least $2f + 1$ validators attest to the tier-specific evidence hash before inclusion. Tier-specific evidence may be a deterministic program output, brand attestation, attestor quorum, optimistic bond record, or Schelling vote result.

Upon the `EVENTCOLLECT` timeout (or early transition when all pending events reach quorum), the validator processes pending attestations through a tiered oracle resolution mechanism:

- **Tier 0 (Programmatic)**: deterministic on-chain query. Requires $f + 1$ matching attestations to prevent a single Byzantine validator from controlling the outcome.
- **Tier 1 (Brand-Attested)**: single attestation from the affiliated brand entity, verified by the latest observation.
- **Tier 2 (Aggregated)**: quorum of $2f + 1$ matching validator attestations. This is the default tier for most events.
- **Tier 3 (Optimistic)**: proposer posts bond and outcome; challenged via dispute game with committee vote.
- **Tier 4 (Schelling Point)**: stake-weighted plurality vote requiring $> n/2$ of the *full validator set*. This full-set denominator prevents abstention attacks.

Events that fail to resolve are deferred with a defer counter. After repeated deferrals, events escalate to a higher tier: $0 \rightarrow 2, 1 \rightarrow 2, 2 \rightarrow 3, 4 \rightarrow 3, 3 \rightarrow 3$ (saturates).

The early-quorum optimization allows the `EVENTCOLLECTING` phase to terminate before the timeout if every pending event has achieved resolution. This reduces latency when attestations arrive quickly.

3.7 View change

View changes follow a HotStuff-family pacemaker with explicit current-target-view synchronization. When a timeout fires (`PROPOSALTIMEOUT` or `VOTETIMEOUT`) at local target view r_i :

1. The validator increments its target view: $view_i \leftarrow view_i + 1$.
2. It clears the proposal buffer and vote accumulators.
3. It broadcasts `VIEWCHANGE($v_i, view_i, highQC_i$)`.

An isolated `VIEWCHANGE` does not by itself justify another jump. A validator advances only when it holds a quorum for its *current target view*. Formally, upon accumulating $q(n)$ `VIEWCHANGE` messages for a target view $r' = view_i$:

1. The validator advances: $view_i \leftarrow r'$.
2. It adopts the highest QC from the accumulated messages.
3. If the highest QC is a `DECIDE QC`, it fast-forwards its height.

4. If the highest QC is at PRECOMMIT or higher, it updates its lock.
5. The designated leader for r' broadcasts a NEWVIEW message carrying the quorum of view-change messages, so replicas that missed some of the underlying VIEWCHANGE traffic can synchronize on the same current target view and highest QC.

A DoS protection bound limits view-change acceptance to a bounded lookahead window $W_{vc} \in \mathbb{N}_{>0}$ around the current view: a validator at target view r ignores VIEWCHANGE messages for any view $r' > r + W_{vc}$. The parameter W_{vc} is a deployment configuration choice; safety and liveness hold for every $W_{vc} \geq 1$, and the bound is orthogonal to the liveness argument of §7.

Remark 3.16 (WAN pacemaker discipline). The liveness-critical point is that the pacemaker does *not* ratchet on arbitrary future-view gossip: it advances on a current-target-view quorum or on the quorum-backed NEWVIEW proof derived from that quorum. In the five-validator WAN case this means four matching VIEWCHANGE messages. A stale committed-certificate rebroadcast path lets lagging replicas import a DECIDE QC even if they missed the original commit, preventing the classic wedge in which part of the network carries a commit certificate while the rest repeatedly times out on an older proposal. The reference WAN parameterization uses a pacemaker timeout of 15 s so that one timer window dominates event collection, four voting rounds, decryption-share propagation, and one certificate rebroadcast.

3.8 Proposer selection

Definition 3.17 (Proposer selection). The proposer for view r in state σ is determined by:

$$\text{proposer}(\sigma, r) = \begin{cases} \mathcal{V}[r \bmod n] & \text{if highQC.height} = 0 \text{ (genesis)} \\ \mathcal{V}[\text{winner}(\text{seed}(\text{highQC.hash}, r), n)] & \text{otherwise} \end{cases}$$

where $\text{seed}(B, r) = \text{SHA-256}(B \| r \| \text{"MOXIE_PROPOSER"})$ and winner uses rejection sampling (see §5).

Remark 3.18 (Leader fairness and DoS surface). Rejection sampling removes modulo bias from leader selection, and the weighted score construction makes proposer probability monotone in stake. The residual DoS surface is pacemaker churn. The protocol limits admissible view-change targets to a bounded lookahead window and requires a current-target-view quorum before advancing, so a Byzantine leader can waste at most one timeout window before the committee converges on the next leader.

4 Committee Selection

To reduce $O(n^2)$ message complexity, MoxieBFT selects a per-round committee of size $k \in [k_{\min}, k_{\max}]$ with $k_{\min} = 4$ (the BFT floor at $f = 1$) via deterministic stake-weighted sortition. The upper bound k_{\max} is a deployment-configuration parameter outside the scope of this paper; safety and liveness hold for every $k \geq k_{\min}$.

Definition 4.1 (Sortition score). For validator v with stake s_v , the sortition score for view r under randomness seed ζ is

$$\text{score}(v, r, \zeta) = \frac{H_{128}(\zeta \| r \| v \| \text{"MOXIE_COMMITTEE"})}{s_v},$$

where H_{128} extracts the first 128 bits of SHA-256 as a big-endian unsigned integer and the division is integer division. Lower scores win; higher stake produces lower weighted scores.

Proposition 4.2 (Determinism). *Given identical inputs $(\mathcal{V}, \{s_v\}, \zeta, r)$, all validators compute identical committees. This follows from: (i) SHA-256 is deterministic, (ii) integer division is deterministic, (iii) the composite key $(score, v)$ is sorted via a `BTreeMap` (no hash-map nondeterminism), (iv) no floating-point arithmetic.*

Proposition 4.3 (Stake monotonicity). *Selection probability is monotonically increasing in stake weight: for $s_v > s_w$, validator v has a lower (better) weighted score than w for any fixed hash output.*

Proposition 4.4 (Zero-stake exclusion). *Validators with $s_v = 0$ receive score $2^{128} - 1$ (maximum), ensuring they are never selected when sufficient staked validators exist.*

5 VRF and Randomness

5.1 Construction

MoxieBFT uses ECVRF-EDWARDS25519-SHA512-TAI per RFC 9381 [10] for verifiable random function evaluation. The seed chain derives per-purpose randomness:

$$\text{block_seed}(o, \omega) = \text{SHA-256}(o \parallel \omega) \quad (1)$$

$$\text{proposer_seed}(B, r) = \text{SHA-256}(B \parallel r \parallel \text{"MOXIE_PROPOSER"}) \quad (2)$$

$$\text{drop_seed}(o, d) = \text{SHA-256}(o \parallel d \parallel \text{"MOXIE_DROP"}) \quad (3)$$

$$\text{market_rand}(o, m) = \text{SHA-256}(o \parallel m \parallel \text{"MOXIE_MARKET"}) \quad (4)$$

where o is the VRF output, ω is the epoch randomness, B is the parent block hash, r is the view number, d is the drop identifier, and m is the market identifier. Domain-separation tags prevent cross-purpose collisions.

5.2 RANDAO commit-reveal

Epoch randomness is produced by RANDAO aggregation over validator reveals:

Definition 5.1 (RANDAO epoch). An epoch spans E_{blocks} blocks, where $E_{\text{blocks}} \in \mathbb{N}_{>0}$ is a deployment-configurable constant. Safety and secrecy are stated in terms of block heights and are independent of the absolute epoch length. The epoch lifecycle is:

1. **Commit:** each validator v_i publishes $c_i = \text{SHA-256}(r_i)$ for a secret 32-byte reveal r_i .
2. **Reveal:** each validator publishes r_i . Validators verify $\text{SHA-256}(r_i) = c_i$.
3. **Aggregate:** if at least $q = 2f + 1$ reveals are collected (under $n = 3f + 1$), $\omega = r_1 \oplus r_2 \oplus \dots \oplus r_k$ (XOR aggregation).
4. **Fallback:** if fewer than q reveals arrive, $\omega = \text{SHA-256}(\omega_{\text{prev}} \parallel (e + 1))$ (deterministic but predictable).

Proposition 5.2 (Bias resistance). *XOR aggregation ensures that any single honest validator's reveal makes the output unpredictable to an adversary who controls all other reveals. Formally: let $r_H \in \{0, 1\}^{256}$ be the reveal of a single honest validator, and let $r_{-H} = \bigoplus_{i \neq H} r_i \in \{0, 1\}^{256}$ be the XOR of all other reveals. Under the REVEAL-phase specification, r_H is committed at COMMIT phase before any other reveal is observed, so an adversary controlling $\{r_i\}_{i \neq H}$ fixes r_{-H} without knowledge of r_H . The output $\omega = r_H \oplus r_{-H}$ is then uniformly distributed over $\{0, 1\}^{256}$ conditional on the adversary's view, since r_H is drawn uniformly and XOR is a bijection on $\{0, 1\}^{256}$ in each argument. As a corollary, the residual bias induced by a last-reveal-withholding*

adversary who decides whether to publish $r_H^{(A)}$ based on the outcome $r_H^{(A)} \oplus r_{-H}$ is bounded by the single-bit conditioning advantage: the adversary controls one bit of decision (publish vs withhold), giving residual bias $\leq 1/2$ on one predicate; spread across n possible leader identities, the per-identity residual bias is $O(1/n)$.

Proof. r_H is sampled uniformly at COMMIT and kept secret until REVEAL. The XOR $\omega = r_H \oplus r_{-H}$ is uniform over $\{0, 1\}^{256}$ when conditioned on any fixed r_{-H} , since $x \mapsto x \oplus r_{-H}$ is a bijection on $\{0, 1\}^{256}$. The 1-bit withholding-adversary corollary follows from the fact that the adversary's decision to publish or withhold $r_H^{(A)}$ admits two possible final seeds per attempted withholding slot; the 256-bit range of ω maps to n validator slots via the rejection sampling of Definition 5.3, yielding at most $O(1/n)$ bias on leader-identity predicates. \square

Reveal withholding is penalized by slashing at per-offense rate $r_{\text{wh}} > 0$. The cost for k colluding validators to explore the $2^k - 1$ candidate seeds available by selective withholding scales as $k \cdot s \cdot r_{\text{wh}}$, where s is the per-validator stake, subject to the cumulative-slash cap $r_{\text{slash}} \cdot s$. Beyond the cap, deterrence is carried by validator ejection under the slashing-dilution floor of Proposition 14.7.

5.3 Winner selection

Definition 5.3 (Rejection sampling). To select a winner index from $[0, n)$ without modulo bias:

1. For iteration $j = 0, 1, \dots, j_{\text{max}} - 1$:
 - (a) Compute $u = H_{64}(\zeta || j)$ (first 8 bytes of SHA-256 as big-endian u64).
 - (b) Set $L = \lfloor 2^{64} / n \rfloor \cdot n$.
 - (c) If $u < L$, return $u \bmod n$.
2. Fallback: return $H_{64}(\zeta) \bmod n$ (bias $< 2^{-192}$ for $j_{\text{max}} = 3$ and reasonable n).

6 Safety

We state and prove the safety properties of MoxieBFT. Pen-and-paper proofs are given here; the accompanying TLA⁺ models check depth-bounded abstractions under the exclusions documented in §15.

6.1 Agreement

The central safety argument requires the HotStuff-2 *unlocking lemma* [7], which we restate in this paper's notation before using it in the Agreement proof. The lemma isolates the non-trivial inductive content and is proved by well-founded induction on the (h, r) lexicographic order, independently of the Agreement statement.

Lemma 6.1 (Unlocking lemma, adapted from HotStuff-2). *Fix a height h and let v^* be an honest validator that holds $\text{lockedQC}_{v^*} = \text{QC}(\text{PRECOMMIT}, B, h, r)$. Then for every view $r' \leq r$ and every block $B' \neq B$, no PREPARE QC for B' at height h and view r' can be formed. Equivalently: once an honest validator locks on B at height h view r , no other block B' can obtain a $2f + 1$ PREPARE QC at height h with view $\leq r$.*

Proof. The PRECOMMIT QC held by v^* at view r required $2f + 1$ PREPARE votes for B at (h, r) , hence the PREPARE QC for B at (h, r) exists.

Well-founded induction on (h, r') , $r' \leq r$. Suppose for contradiction that a PREPARE QC q' for $B' \neq B$ at (h, r') exists for some $r' \leq r$. By Proposition 2.6 applied to the PREPARE quorum of B at (h, r) and the PREPARE quorum of B' at (h, r') , some honest validator u voted PREPARE for both.

Honest validators vote PREPARE for a proposal at (h, r') only if the proposal satisfies the locked-QC compatibility check against lockedQC_u at that point: either $B'.\text{parent} = \text{lockedQC}_u.\text{hash}$ (extension) or $\text{QC}_{\text{just}}(B').\text{view} > \text{lockedQC}_u.\text{view}$ (override). In the extension case, B' is at height $h + 1$, contradicting B' 's height being h . In the override case, the justification QC lies at height $h - 1$ and view strictly greater than the lock's view, whose existence at a strictly smaller $(h - 1, \cdot)$ or smaller (h, r'') is ruled out by the inductive hypothesis applied to $(h - 1, \cdot)$ or to (h, r'') with $r'' < r'$. The base case is explicit: for $r' = 0$, no PRECOMMIT QC at any view < 0 exists, so no prior lock constrains u 's PREPARE vote at $r' = 0$. However, a $B \neq B'$ PREPARE QC at $(h, 0)$ would require two honest quorums at the same $(h, 0)$, contradicted by Proposition 2.6 and the single-vote-per- (h, r, φ) rule. For $h = 0$, genesis is fixed. The inductive step closes. This contradicts the existence of q' , so $B = B'$. \square

Corollary 6.2 (Lock-advancement). *Fix honest validator v^* holding $\text{lockedQC}_{v^*} = \text{QC}(\text{PRECOMMIT}, B, h, r)$ at time t_1 . Suppose at time $t_2 > t_1$, v^* observes a PREPARE QC q^* for block B^* at (h, r^*) with $r^* > r$. Then exactly one of the following holds:*

- (i) $B^* = B$ (the same block as the lock); or
- (ii) q^* derives from a VIEWCHANGE quorum whose highest-QC evidence included a PRECOMMIT QC at height h and view r^\dagger with $r < r^\dagger \leq r^*$, and that PRECOMMIT QC is for B (i.e., v^* 's lock advances from (B, r) to (B, r^\dagger) , still on B); or
- (iii) at least $f + 1$ Byzantine validators exist, contradicting $|\mathcal{H}| \geq 2f + 1$.

Proof. Suppose $B^* \neq B$. The PREPARE quorum for B^* at (h, r^*) contains $2f + 1$ validators, and the PRECOMMIT quorum for B at (h, r) also contains $2f + 1$ validators. By Proposition 2.6 their intersection contains at least $f + 1$ validators; if all are honest, case (iii) is excluded. Let u be an honest validator in the intersection. u voted PRECOMMIT for B at (h, r) , setting $\text{lockedQC}_u.\text{view} \geq r$ at that moment. By locked-view monotonicity (Theorem 6.8), at the time u voted PREPARE for B^* at (h, r^*) with $r^* > r$, u 's lock was at some view r^\ddagger with $r \leq r^\ddagger$. The compatibility check required $\text{QC}_{\text{just}}(B^*).\text{view} > r^\ddagger \geq r$ (override branch, since the extension branch would put B^* at height $h + 1$). Hence B^* 's justification QC has view strictly greater than r , and by Lemma 6.1 applied at (h, r) , no PREPARE QC for $B' \neq B$ at height h and view $\leq r$ exists; so the override justification traces to a PRECOMMIT QC at (h, r^\dagger) with $r < r^\dagger \leq r^*$ whose block is B (the unique block with a PRECOMMIT QC at (h, \cdot) up to view r is B , and subsequent PRECOMMIT QCs at (h, \cdot) must extend the B -locked line, again by Lemma 6.1 applied at the highest such view). This is case (ii): the view-change evidence that enables u 's override necessarily advances the lock from (B, r) to (B, r^\dagger) , still on B , contradicting $B^* \neq B$ at height h . \square

Theorem 6.3 (Agreement). *No two honest validators commit different blocks at the same height. Formally: for all honest $v_i, v_j \in \mathcal{H}$ and committed records (B_1, h) at v_i and (B_2, h) at v_j , we have $B_1 = B_2$.*

Proof. Suppose for contradiction that honest validators v_i and v_j commit different blocks $B_1 \neq B_2$ at height h .

A block is committed only upon collecting a DECIDE QC, which requires a prior COMMIT QC, which requires a prior PRECOMMIT QC. The PRECOMMIT QC is where the lock is set: upon forming a PRECOMMIT QC for block B at view r , every honest validator in the quorum sets $\text{lockedQC} \leftarrow \text{QC}(\text{PRECOMMIT}, B, h, r)$.

Let Q_1 and Q_2 be the PRECOMMIT quorums for B_1 (at view r_1) and B_2 (at view r_2), respectively. By Proposition 2.6, there exists an honest validator $v^* \in Q_1 \cap Q_2 \cap \mathcal{H}$.

Case 1: $r_1 = r_2$. Validator v^* voted PRECOMMIT for both B_1 and B_2 at the same view. But honest validators only vote once per (h, r, φ) : the equivocation detection mechanism rejects any second vote with a different block hash. Contradiction.

Case 2: $r_1 < r_2$ (**WLOG**). After voting PRECOMMIT for B_1 at view r_1 , validator v^* set lockedQC_{v^*} to the PRECOMMIT QC for B_1 at (h, r_1) . We apply Lemma 6.1: no PREPARE QC for any $B' \neq B_1$ at height h and view $\leq r_1$ can exist. In particular, this rules out B_2 's PREPARE QC from being at height h and view $\leq r_1$.

So B_2 's chain of QCs at height h must start at some view strictly greater than r_1 . In particular, v^* (which holds lockedQC_{v^*} on B_1 at view r_1) must at some later time have voted PRECOMMIT for B_2 at view $r_2 > r_1$, and thus must have observed a PREPARE QC for B_2 at (h, r_2) . Apply Corollary 6.2 to v^* with the observed PREPARE QC q^* for B_2 at (h, r_2) , $r_2 > r_1$:

- Case (i): $B_2 = B_1$, contradicting $B_1 \neq B_2$.
- Case (ii): the view-change evidence enabling the observation carries a PRECOMMIT QC at (h, r^\dagger) , $r_1 < r^\dagger \leq r_2$, whose block is B_1 . Hence v^* advances its lock to (B_1, r^\dagger) , and the PREPARE QC for B_2 must override this higher B_1 -lock. Iterating the corollary drives r^\dagger strictly upward within a bounded range $(r_1, r_2]$, so the iteration terminates at a fixed point where the only PREPARE QC admissible at (h, \cdot) above the lock is for B_1 , again contradicting $B_1 \neq B_2$ at height h .
- Case (iii): $f + 1$ Byzantine validators, contradicting $|\mathcal{H}| \geq 2f + 1$.

In all cases, we reach a contradiction. Therefore $B_1 = B_2$. \square

6.2 Validity

Theorem 6.4 (Validity). *Every committed block was proposed by the expected leader for its view. Formally: if honest validator v_i commits block B at height h from view r , then $B.\text{proposer} = \text{proposer}(\sigma_i, r)$.*

Proof. An honest validator only buffers a proposal if $\text{proposer}(\sigma_i, \text{view}_i) = m.\text{from}$ (the leader check in proposal handling, §3.4). A block is committed only if it was previously buffered. Since the proposer field is part of the block's committed hash, tampering is detectable. \square

6.3 View monotonicity

Theorem 6.5 (View monotonicity). *For any honest validator v_i , the view number never decreases across state transitions:*

$$\forall t_1 < t_2 : \text{view}_i(t_1) \leq \text{view}_i(t_2)$$

Proof. We verify by exhaustive case analysis over the transition function. Every action that modifies view_i either:

1. Increments it by 1 (DECIDE quorum: $\text{view}_i \leftarrow r + 1$; PROPOSALTIMEOUT: $\text{view}_i \leftarrow \text{view}_i + 1$; VOTETIMEOUT: $\text{view}_i \leftarrow \text{view}_i + 1$).
2. Sets it to a target view $r' > \text{view}_i$ (VIEWCHANGE quorum: the guard $r' > \text{view}_i$ is checked before advancing; NEWVIEW: the guard $\text{new_view.view} > \text{view}_i$ is checked).

No action sets view_i to a value less than or equal to the current view. The transition function of Section 3.3 emits a safety violation VIEWDECREASED on any violation of $\text{view}' \geq \text{view}$. \square

6.4 Height monotonicity

Theorem 6.6 (Height monotonicity). *For any honest validator v_i , the committed height never decreases:*

$$\forall t_1 < t_2 : h_i(t_1) \leq h_i(t_2)$$

Proof. Height is advanced in exactly three places:

1. DECIDE quorum: $h_i \leftarrow B.\text{height} + 1$, where $B.\text{height} \geq h_i$ (guaranteed by the proposal height check).
2. Height fast-forward in proposal handling: $h_i \leftarrow B.\text{height}$ only if $B.\text{height} > h_i$.
3. Height fast-forward in view-change quorum / NEWVIEW: $h_i \leftarrow \text{QC}.\text{height} + 1$ only if $\text{QC}.\text{height} + 1 > h_i$.

All three cases produce $h'_i \geq h_i$. The transition function of Section 3.3 emits a safety violation HEIGHTDECREASED on any violation of $h' \geq h$. \square

6.5 Locked QC safety

Invariant 6.7 (Locked QC safety). *An honest validator only votes for proposals compatible with its locked QC. Formally: if $\text{lockedQC}_i \neq \perp$ and $P_i \neq \perp$, then $\text{ProposalCompatible}(\text{lockedQC}_i, P_i.\text{QC}_{\text{just}})$ holds, i.e.:*

$$P_i.B.\text{parent} = \text{lockedQC}_i.\text{hash} \quad \vee \quad P_i.\text{QC}_{\text{just}}.\text{view} > \text{lockedQC}_i.\text{view}$$

Proof. The invariant is a per-validator, per-vote conjunction: at the moment honest validator v_i emits a PREPARE vote for proposal P_i , we must have $\text{ProposalCompatible}(\text{lockedQC}_i, P_i.\text{QC}_{\text{just}})$. By inspection of the transition function (Section 3.3), P_i is buffered only if the compatibility predicate (extension or override, per Invariant 6.7) evaluates to \top against the *current* lock state at buffering time, and the PREPARE vote is emitted in the same transition without intervening lock update. The invariant holds at emission time by construction.

The longitudinal property, that the lock's view field is monotone non-decreasing along the validator's local timeline, is the content of Theorem 6.8 below. \square

Theorem 6.8 (Locked-view monotonicity). *For any honest validator v_i with $\text{lockedQC}_i(t_1) \neq \perp$ and any $t_2 > t_1$:*

$$\text{lockedQC}_i(t_2).\text{view} \geq \text{lockedQC}_i(t_1).\text{view}.$$

Proof. The lock field is mutated by exactly two transition rules (Section 3.3, Section 3.5); we prove each preserves the view bound.

Rule L1 (PRECOMMIT QC formation at view r). On observing a PRECOMMIT QC for block B at (h, r) , the validator sets $\text{lockedQC}_i \leftarrow \text{QC}(\text{PRECOMMIT}, B, h, r)$. We must show $r \geq \text{lockedQC}_i^{\text{prev}}.\text{view}$, where $\text{lockedQC}_i^{\text{prev}}$ is the lock held at the instant preceding the update.

Honest validator v_i emits a PRECOMMIT vote at view r only after first emitting a PREPARE vote at view r . By the compatibility predicate (Invariant 6.7) applied at the PREPARE vote, the proposal's justification QC either extends $\text{lockedQC}_i^{\text{prev}}$ (so $\text{lockedQC}_i^{\text{prev}}.\text{view} \leq r$ since all QCs at a view reference justifications at strictly smaller views, and the PRECOMMIT here is at view r) or overrides it with $\text{QC}_{\text{just}}.\text{view} > \text{lockedQC}_i^{\text{prev}}.\text{view}$. In the extension branch, the proposal extends the $\text{lockedQC}_i^{\text{prev}}$ -chain at height h , and the PRECOMMIT QC forms at view $r \geq \text{view}_i$; by view-monotonicity (Theorem 6.5) applied at the validator's prior views, $\text{view}_i \geq \text{lockedQC}_i^{\text{prev}}.\text{view}$ (since the validator

entered its current view only by advancing through its history, and every lock acquisition sets the lock's view to a view the validator has entered). In the override branch, $QC_{\text{just}}.\text{view} > \text{locked}QC_i^{\text{prev}}.\text{view}$ and QC_{just} justifies entering the proposal's view r , so $r > QC_{\text{just}}.\text{view} > \text{locked}QC_i^{\text{prev}}.\text{view}$. In either branch, $r \geq \text{locked}QC_i^{\text{prev}}.\text{view}$.

Rule L2 (lock adoption at VIEWCHANGE / NEWVIEW quorum). On receiving a NEWVIEW message for target view r' carrying a quorum Q_{vc} of $2f + 1$ VIEWCHANGE signatures, each signature carrying a highQC field, validator v_i adopts the QC

$$q^* = \arg \max_{q \in \{\text{highQC}(m) : m \in Q_{\text{vc}}\}} \text{rank}(q),$$

updating $\text{locked}QC_i$ only if $q^*.\text{rank} \geq \text{locked}QC_i^{\text{prev}}.\text{rank}$ and q^* is at least a PRECOMMIT QC (by the lock-update condition of the transition function). We claim $q^*.\text{view} \geq \text{locked}QC_i^{\text{prev}}.\text{view}$.

Let $r_0 = \text{locked}QC_i^{\text{prev}}.\text{view}$. The lock at view r_0 was set (either at L1 above or inductively at a prior L2 application) only because a PRECOMMIT QC at view r_0 formed, which required $2f + 1$ validators' PRECOMMIT votes at view r_0 . Call this quorum Q_{r_0} and let $Q_{r_0}^{\mathcal{H}} = Q_{r_0} \cap \mathcal{H}$, with $|Q_{r_0}^{\mathcal{H}}| \geq f + 1$ (since $|\mathcal{B}| \leq f$).

Consider the VIEWCHANGE quorum Q_{vc} with $|Q_{\text{vc}}| \geq 2f + 1$. By quorum intersection (Proposition 2.6), $|Q_{\text{vc}} \cap Q_{r_0}^{\mathcal{H}}| \geq 1$: some honest validator u contributed both a PRECOMMIT vote at view r_0 (hence set its own lock to view $\geq r_0$ at that moment) and a VIEWCHANGE message to Q_{vc} . By the validator's local locked-view monotonicity up to the moment of sending the VIEWCHANGE (proved inductively for times strictly earlier than the current rule application; the induction is on t_2 , and here we invoke the IH at u 's view-change-sending time which is earlier than v_i 's NEWVIEW-processing time), u 's highQC field at view-change-sending carries a QC of view $\geq r_0$. Formally, honest validators populate the highQC field of their VIEWCHANGE from their local highQC_u , which is updated upon any QC observation with rank \geq the current $\text{highQC}_u.\text{rank}$ (highest-QC rule, Section 3.3); since u observed its own PREPARE QC at (h, r_0) before voting PRECOMMIT at r_0 , $\text{highQC}_u.\text{view} \geq r_0$ at view-change-sending and thereafter.

Therefore $q^*.\text{view} = \max\{q.\text{view} : q \in \{\text{highQC}(m) : m \in Q_{\text{vc}}\}\} \geq \text{highQC}_u.\text{view} \geq r_0 = \text{locked}QC_i^{\text{prev}}.\text{view}$, establishing the claim for Rule L2.

Finally, the state-machine update in Rule L2 sets $\text{locked}QC_i \leftarrow q^*$ only if $q^*.\text{rank} \geq \text{locked}QC_i^{\text{prev}}.\text{rank}$ (the lock-update condition). Combined with $q^*.\text{view} \geq r_0$ just proved, the updated lock has view $\geq r_0$, and the non-decreasing property holds at this step.

Induction closure. The overall statement is by well-founded induction on t_2 : each of rules L1, L2 applied at time t_2 preserves non-decrease provided the IH holds at all earlier times (used in L2 to invoke u 's prior highQC monotonicity). At the base $t_2 = t_1$, the statement is trivial. For times when the lock is not mutated, the lock is unchanged. This exhausts all cases.

Byzantine validators' VIEWCHANGE signatures cannot subvert this: a Byzantine validator may sign a VIEWCHANGE with a fabricated lower highQC field, but the highest-QC selection rule in L2 picks the *maximum* across the quorum, and the honest validator u contributing a VIEWCHANGE with $\text{highQC}_u.\text{view} \geq r_0$ ensures $q^* \geq \text{highQC}_u$. A Byzantine validator submitting a fabricated *higher* highQC is rejected: each highQC field must carry a valid BLS-aggregate signature of a $2f + 1$ quorum, and forging such a signature violates BLS unforgeability (which is assumed throughout). \square

Remark 6.9 (Relation to HotStuff-2 unlocking). Theorem 6.8 is the lock-advancement half of the standard HotStuff-2 unlocking-lemma machinery [7]: the unlocking lemma (Lemma 6.1) bounds which blocks can form PREPARE QCs at views below the lock, and the lock-advancement corollary (Corollary 6.2) shows lock advances remain on the same committed block. Locked-view monotonicity is the per-validator timeline counterpart

that feeds into both: it rules out a validator “unlocking downward” and voting for a stale proposal under a lower-view lock. An adversary cannot induce the regression even under the composed model of Section 16.4 (Byzantine VIEWCHANGE messages do not subvert the highest-QC rule; forged higher QCs violate BLS unforgeability).

Locked-view monotonicity does *not* entail that every proposal compatible with an earlier lock remains compatible with a later lock: a proposal whose justification view lies strictly between the old and new lock views may become incompatible. That is the intended behavior of the locking rule: as the lock advances, the admissible proposal set contracts in view.

6.6 Vote bound

Invariant 6.10 (Vote bound). *For any honest validator v_i and vote phase φ :*

$$|V_i[\varphi]| \leq n$$

Proof. Votes are accumulated per unique validator identity. An honest validator only inserts v into $V_i[\varphi]$ if $v \in \mathcal{V}$, and duplicate votes from the same validator are rejected (equivocation detection). Since $|\mathcal{V}| = n$, the bound holds. \square

7 Liveness

Theorem 7.1 (Liveness). *Assume (a) partial synchrony with unknown GST and known Δ ; (b) honest stake fraction $S_{\mathcal{H}}/S_{\text{total}} \geq 2/3$; (c) the Event-Collect tier-escalation schedule produces QC_{EC} within $c\Delta$ post-GST rounds for some bounded $c \geq 0$; (d) the pacemaker advances only on a current-target-view quorum $q(n) = \lceil 2n/3 \rceil$ of VIEWCHANGE messages or on the quorum-backed NEWVIEW proof derived from that quorum, and the timeout parameter T_{to} dominates one Event-Collect window, the four voting rounds, decryption-share propagation, and one stale-certificate rebroadcast; (e) weak fairness on honest validator actions. Against $\mathcal{A}_{\text{MBFT}}$ (§16.4), the expected latency of committing a block after GST is bounded by*

$$\mathbb{E}[t_{\text{commit}} - \text{GST}] \leq \Delta_{\text{sync}} + t_{\text{ec}} + 5\Delta + c\Delta + \frac{p_{\mathcal{B}}}{1 - p_{\mathcal{B}}} T_{\text{to}}, \quad (5)$$

where $S_{\mathcal{B}} = S_{\text{total}} - S_{\mathcal{H}}$, $p_{\mathcal{B}} = S_{\mathcal{B}}/S_{\text{total}}$, and $\Delta_{\text{sync}} \leq \Delta$ is the post-GST view-synchronization delay.

Proof. We proceed via three lemmas (L1)-(L3), each contributing a term to the bound.

(L1) View synchronization. After GST, all messages between honest validators are delivered within Δ . If honest validators are split across target views, the pacemaker of assumption (d) does not keep ratcheting on isolated future-view gossip. Instead the first honest timeout places each lagging validator at some target view $r + 1$, and within one post-GST round the honest VIEWCHANGE messages for that same target view accumulate to a quorum $q(n)$. The designated leader then broadcasts a quorum-backed NEWVIEW carrying the highest QC among the supporting messages. Any honest replica that missed some of the underlying VIEWCHANGE traffic receives the NEWVIEW proof within Δ and synchronizes to the same view and highest QC. If the highest carried QC is already a DECIDE certificate, stale-certificate rebroadcast imports that committed prefix directly. The contribution of (L1) is at most Δ .

(L2) Honest leader. At each view the probability that the designated leader lies in \mathcal{B} is $p_{\mathcal{B}} := S_{\mathcal{B}}/S_{\text{total}}$ under stake-weighted leader selection (Proposition 4.3). Under assumption (b) ($S_{\mathcal{H}}/S_{\text{total}} \geq 2/3$) we have $p_{\mathcal{B}} \leq 1/3$, so $1 - p_{\mathcal{B}} \geq 2/3$. The number of consecutive views before an honest leader is selected is geometrically distributed with success probability $1 - p_{\mathcal{B}}$, and the expected number of views is $1/(1 - p_{\mathcal{B}}) \leq 3/2$. Each failed

Byzantine-leader view contributes at most the configured timeout envelope T_{to} . The expected number of failed Byzantine leaders before the first honest leader is $p_B/(1 - p_B)$, giving an additive expected term $(p_B/(1 - p_B))T_{\text{to}}$.

(L3) Quorum completion. Once an honest leader proposes in view r^* , the proposal - whose header fuses the Event-Collect quorum QC_{EC} with the Merkle-root commitment $R_{\text{committed}}$ per Remark 10.18 - reaches every honest validator within Δ . Each honest validator enters `EVENTCOLLECTING` and advances on early quorum if all pending events have already resolved; in the general case (c) contributes at most $c\Delta$. Once QC_{EC} is available, the four voting rounds `PREPARE`, `PRECOMMIT`, `COMMIT`, `DECIDE` each complete within Δ . With the fused gate G_1 this is five rounds of Δ , contributing $5\Delta + c\Delta$. Assumption (d) ensures that the pacemaker timeout does not expire inside this healthy round: the WAN timer budget dominates the message, decryption, and rebroadcast envelope of one successful execution.

Summing (L1)-(L3) and the Event-Collect budget t_{ec} yields the bound (5).

Fairness. The proof requires weak fairness on every honest validator action: a continuously enabled action eventually fires. This is modeled in the TLA^+ safety automaton and in the separate pacemaker-liveness model by `WF_vars` declarations on the honest action set (§15).

Conditional dependence on Π_{EC} . The bound is conditional on assumption (c). Tier-escalation liveness for the Π_{EC} component is recorded explicitly in assumption (c) of Theorem 16.2 and is outside the scope of the present proof; see Open Problem 7.3. \square

Remark 7.2. Liveness does not hold before GST: during the asynchronous period messages may be delayed indefinitely, and no consensus protocol can guarantee termination in a fully asynchronous network with even one Byzantine fault (FLP impossibility [5]).

Open Problem 7.3 (Tier-escalation liveness for Π_{EC}). Assumption (c) of Theorem 7.1 posits that the Event-Collect tier-escalation schedule terminates in at most $c\Delta$ post-GST rounds for some bounded c . A formal proof exhibiting an explicit bound on c as a function of the oracle-tier schedule is outside the scope of this paper.

8 Consensus Pipelining

MoxieBFT supports HotStuff-2-inspired pipelining with depth 1:

Definition 8.1 (Pipeline overlap). While block N is in its `COMMIT` or `DECIDE` phase, the proposer for block $N + 1$ may begin `EVENT-COLLECT` (attestation collection) for the next block.

Proposition 8.2 (Pipeline safety). *Pipelining is safe because:*

1. Oracle attestations for block $N + 1$ observe external events, not on-chain state from block N . They are state-independent.
2. The proposer for block $N + 1$ cannot construct a full `PREPARE` message until block N reaches `DECIDE` (it needs the parent hash), so pipeline depth > 1 is not useful for voting phases.
3. Speculative attestations carry no state commitment and are discarded if block N fails (timeout/view-change).

The asymptotic savings from pipelining are one attestation-collection round per block on the happy path: for a deployment with per-round budget ℓ_{\max} , the effective block interval is reduced by up to t_{ec} by overlapping next-block attestation collection with the current block’s commit round, subject to the constraint $t_{ec} \leq t_{com} + t_{dec}$. The numerical savings are deployment-configurable and outside the scope of this paper.

9 MEV Resistance

9.1 Threshold encryption

Definition 9.1 (Encrypted mempool). User-submitted transactions destined for inclusion in the encrypted mempool are encrypted under an epoch-wide threshold public key before submission. The key parameters are:

- Scheme: threshold ElGamal over the BLS12-381 group G_1 (no pairing or hash-to- G_2 operation is used on the encryption/decryption path; the pairing surface is confined to BLS signatures on consensus messages and to Boneh-Drijvers-Neven batch verification of Definition 12.1). Encryption derives a random scalar r , computes the ephemeral group element $C_1 = r \cdot G_1$ and the shared secret $S = r \cdot \text{epoch_pk}(E)$; the payload is encrypted under a symmetric key derived from S via a key-derivation function (KDF) modelled as a random oracle, with a one-time tag binding plaintext and ephemeral key (the standard *hashed-ElGamal* DEM; IND-CPA reduces to the Computational Diffie-Hellman problem in G_1 with KDF in the random-oracle model, or equivalently to DDH in G_1 without ROM). The domain-separation label $\text{id}_{\text{enc}}(h, E)$ of Definition 10.3 is an input to the KDF, not a pairing identity; it binds each ciphertext to the target height and epoch.
- Threshold: $t = q(n) = \lceil 2n/3 \rceil$. In the canonical $n = 3f + 1$ regime this reduces to $t = 2f + 1$, matching the BFT quorum; in a five-validator committee, $t = q(5) = 4$. When n takes the intermediate forms $3f + 2$ or $3f + 3$, the decryption threshold remains the same supermajority rule $q(n)$, so the corruption cap f governs safety while the decryption cap $t - 1$ governs secrecy as independent parameters.
- DKG protocol: Pedersen distributed key generation [18, 19] yielding a degree- $(t-1)$ secret polynomial whose constant term is the threshold secret; every validator holds an additive share sk_i derived from the polynomial. (The public-parameter generator is also interoperable with the non-interactive resharing protocol of [16] at epoch boundaries.) Verification vectors commit each share’s evaluation to G_1 points enabling public share-validity checks.
- Epoch public key: the DKG of each epoch produces an aggregate threshold public key $\text{epoch_pk}(E) \in G_1$ that serves as the encryption public key across every height of the epoch and as an input to the CP3 domain separator (Definition 10.3). The same $\text{epoch_pk}(E)$ is used for every block h with $E = \lfloor h/E_{\text{blocks}} \rfloor$, so ciphertexts submitted at any view of any such h remain decryptable across intra-epoch view changes without re-encryption.
- Key rotation: every E_{rot} epochs, measured in block height. At each rotation boundary $\text{epoch_pk}(E + 1)$ replaces $\text{epoch_pk}(E)$, and users re-encrypt any still-pending transactions under the new epoch key; the per-rotation re-encryption cost is bounded by the mempool depth observed at the boundary. The rotation period E_{rot} is a deployment-configuration parameter and does not enter any safety or secrecy claim below.
- Ciphertext padding to a fixed byte boundary, to prevent size-based traffic analysis.

Primitive choice. The scheme specified above is *threshold hashed-ElGamal*. Decryption shares are $ds_i = sk_i \cdot C_1 \in G_1$, combined by Lagrange interpolation in G_1 ; no pairing or hash-to- G_2 is invoked on the encryption or decryption path. The security guarantee is IND-CPA under CDH-with-KDF-in-ROM (equivalently DDH without ROM) in G_1 , not IND-ID-CPA. An optional ML-KEM-768 wrapper may transport the DEM key or its re-encryption material across committee members, but it does not change the consensus proof obligations: early plaintext recovery still requires crossing the threshold decryption barrier or violating the CP3 gate. A strengthening via Boneh-Franklin IBE, adding identity-bound key extraction and pairing-based share verification, is a natural variant and is not assumed anywhere in the proofs below. The consensus-layer binding of share release to the authoritative $R_{\text{committed}}$ is structural (the BLS-signed attestation of Definition 10.5(b) together with the state-machine gate and slashing of Definition 10.5(c)-(d)) and is independent of whether the underlying cryptosystem is Boneh-Franklin IBE or threshold hashed-ElGamal.

Definition 9.2 (Decryption budget). The threshold decryption budget is the sum of the four component costs:

$$t_{\text{decrypt}} = t_{\text{share_verify}} + t_{\text{share_gossip}} + t_{\text{aggregate}} + t_{\text{final_verify}}.$$

Absolute wall-clock values are deployment-configuration choices.

9.2 Ordering commitment protocol

The MEV resistance mechanism is a three-phase ordering commitment protocol that binds the execution ordering before transaction contents are revealed.

Definition 9.3 (Ordering commitment). The three phases are:

1. **Commit** (during PREPARE): the proposer commits the Merkle root $R_{\text{committed}}$ of the encrypted transaction sequence. This root is computed as a binary SHA-256 Merkle tree over the ciphertext hashes.
2. **Reveal** (during DECIDE): threshold decryption reveals transaction contents. Each validator releases its decryption share.
3. **Enforce** (post-decryption): validators verify $R_{\text{executed}} = R_{\text{committed}}$. A mismatch is provable equivocation, slashable at rate r_{cp4} (Table 2).

Proposition 9.4 (Ordering integrity). *Post-decryption reordering is detectable: if the proposer alters the transaction ordering after threshold decryption reveals the contents, the Merkle root changes, producing $R_{\text{executed}} \neq R_{\text{committed}}$, which every honest validator can independently verify and prove.*

9.3 Attack taxonomy and mitigation

Vector	Mitigation	Residual surface
Event frontrun	Threshold encryption (CP2)	None
Sandwich	Commit-before-decrypt (CP1-CP4)	None
Anticipatory	Phase separation	Timing side-channels
Cross-market arbitrage	Unified clearing	Cross-block repositioning
Backrun	Encrypted conditional orders	Execution-delay leakage
Oracle manipulation	Validator slashing	Governance attack

Table 1: Attack vectors within the adversary model of Definition 2.7, their consensus-layer mitigation, and the residual surface. Formal frontrun-freeness against the composed adversary $\mathcal{A}_{\text{MBFT}}$ is Theorem 10.9; the cross-block residual rate has the lower-bound witness of Proposition 9.8.

9.4 Formal MEV model

Definition 9.5 (MEV extraction cost). For an attack strategy A , the MEV extraction cost is:

$$E(A) \geq \min(\text{threshold_decryption_bribe}, \text{slash_penalty_for_equivocation})$$

Theorem 9.6 (Threshold bribe cost). Let $n \geq 3f + 1$ be the number of validators, each staking at least s_{\min} (in abstract stake units), and let $t := q(n) = \lceil 2n/3 \rceil$ be the decryption threshold. Then corrupting the threshold decryption scheme requires bribing at least t key holders, yielding the lower bound

$$C_{\text{decrypt}} \geq t \cdot s_{\min}$$

Separately, the Byzantine corruption cap $f + 1$, the smallest attack set that a Byzantine quorum can assemble, yields a safety-attack lower bound

$$C_{\text{safety}} \geq (f + 1) \cdot s_{\min}.$$

The two thresholds are distinct attack objectives. Any deployment wishing to bound the single-block MEV extraction value MEV_1 below the attack cost must therefore size s_{\min} such that $t \cdot s_{\min} \geq \text{MEV}_1 \cdot \gamma$ for an economic safety margin $\gamma > 1$; numerical calibration of s_{\min} and γ is a deployment concern outside the scope of this paper.

Proof. The threshold decryption scheme requires $t = q(n) = \lceil 2n/3 \rceil$ shares to reconstruct any plaintext. Each share is held by a distinct validator with at minimum s_{\min} stake. To obtain t shares, an attacker must corrupt at least t validators. The minimum bribe must at least match each validator's stake at risk from slashing plus expected future rewards, giving $C_{\text{decrypt}} \geq t \cdot s_{\min}$. The safety-attack lower bound $C_{\text{safety}} \geq (f + 1) \cdot s_{\min}$ follows identically from the Byzantine corruption cap. \square

Remark 9.7 (Non-canonical n). For non-canonical $n \in \{3f + 2, 3f + 3\}$ admitted by Definition 9.1, the threshold $t = q(n) = \lceil 2n/3 \rceil$ exceeds the canonical shorthand $2f + 1$: at $n = 3f + 2$, $t = 2f + 2$; at $n = 3f + 3$, $t = 2f + 2$. The decryption-bribe lower bound tightens to $C_{\text{decrypt}} \geq t \cdot s_{\min}$ while the safety-attack bound $C_{\text{safety}} \geq (f + 1) \cdot s_{\min}$ is unchanged since the Byzantine cap f is a function of n via $f = \lfloor (n - 1)/3 \rfloor$, which equals the same value in each intermediate form relative to the canonical $n = 3f + 1$.

9.5 Residual MEV: cross-block repositioning

The CP1-CP4 discipline eliminates intra-block frontrunning and sandwich attacks against the encrypted transaction set (Theorem 10.9). It does not close the *cross-block* informational channel: once block b decrypts and its ordering is public, a rational adversary may submit follow-on transactions in block $b + 1$ whose size is optimized against the realized price gap. We state a regime-specific lower bound on this residual rate; its constant requires empirical calibration and is not determined by the consensus layer.

Proposition 9.8 (Residual MEV lower bound). *Let σ_V denote the per-block mid-price volatility of the aggregate trading book, and let $\Delta_{\text{rev}} \geq 1$ denote the number of blocks between ordering commitment and decrypted-state publication. A rational adversary controlling fewer than t threshold-decryption shares can extract residual value at rate*

$$\rho_{\text{adv}} = \Omega\left(\sigma_V \cdot \sqrt{\Delta_{\text{rev}}}\right). \quad (6)$$

Proof sketch. The adversary cannot observe intra-block contents by CP2. After decryption at block b the realized ordering and its price impact become public. The adversary submits a follow-on transaction in block $b + 1$ whose optimal size is governed by the realized-versus-expected price gap. Over Δ_{rev} blocks the expected price-gap magnitude grows as $\sigma_V \sqrt{\Delta_{\text{rev}}}$; Kelly-optimal sizing yields the stated rate. The attack respects CP1-CP4 and exploits the cross-block information gap that the encrypted mempool does not close (cf. Theorem 10.9). \square

Open Problem 9.9 (Derivation and calibration of the constant). A rigorous derivation replaces the sketch above with (i) a stochastic model of the mid-price process, (ii) a specification of the adversary’s information set at block-commit time, and (iii) a reduction from the extraction rate to $\sigma_V \sqrt{\Delta_{\text{rev}}}$ with explicit tracking of the constant. Empirical calibration of that constant against market-microstructure data is a separate deployment task.

Redistribution policies for captured residual MEV are deployment-level mechanism choices outside the scope of this paper.

10 Commit-Precondition Properties CP1-CP4

The ordering-commitment protocol (Definition 9.3) must satisfy four properties for the MEV-resistance guarantees to hold. We state them as named *Commit-Precondition* properties CP1-CP4 and show how each is enforced by the protocol’s state machine (Section 3.3) and the phase-ordering discipline of Algorithm 1 (Section 3.1).

Namespace note. The sibling paper `pwamm-coupling.tex` defines an unrelated property set PC1-PC4 capturing *proposer constraints* (pre-commitment, opacity, censorship resistance, self-binding). Those PC1-PC4 live in the `pwamm-coupling` namespace and are semantically distinct from the CP1-CP4 defined here, which are *cryptographic commit-reveal properties* of the threshold-encrypted mempool. Readers citing CP_i should reference this paper; `pwamm-coupling` owns the PC_i namespace. We cross-reference but never alias the two sets.

Definition 10.1 (CP1: Binding commitment). At the end of the PREPARE phase, the proposer has broadcast a Merkle root $R_{\text{committed}}$ over the ordered ciphertext sequence $(\tilde{c}_1, \dots, \tilde{c}_n)$. A proposer is *CP1-bound* if, conditional on having broadcast $R_{\text{committed}}$, it cannot later produce a distinct ordered sequence $(\tilde{c}'_1, \dots, \tilde{c}'_n)$ with the same Merkle root and different content-for-position mapping without finding a SHA-256 collision.

Definition 10.2 (CP2: Hiding commitment until reveal). Until the threshold of t decryption shares is assembled, no validator or external observer can recover the plaintext of any ciphertext \tilde{c}_i . Formally, for any adversary \mathcal{A} controlling fewer than t validators, the advantage in distinguishing encryptions of any two equal-length messages is negligible in the security parameter λ .

The f -bounded $\mathcal{A}_{\text{MBFT}}$ of §16.4 is strictly weaker than the t -bounded adversary of this definition; CP2 holds *a fortiori* under the composed adversary. The two bounds address distinct attack objectives: f for consensus-safety, $t - 1$ for secrecy. An adversary of size $f^* \in [f + 1, t - 1]$ exceeds MoxieBFT's consensus-safety threat model of Definition 2.7 but preserves CP2 hiding; such an adversary is outside $\mathcal{A}_{\text{MBFT}}$ and its defeat is a primitive-level guarantee, not a consensus-level one.

Definition 10.3 (CP3 identity schedule). The CP3 scheme uses *two distinct objects*, fixed at different times in the protocol: a *domain-separation label* $\text{id}_{\text{enc}}(h, E)$ bound into the encryption KDF (used by the sender at encryption time and by the combiner at decryption time to derive the per-ciphertext symmetric key), and a *scheduling tag* used by the consensus state machine to gate share release. These objects together avoid the circular dependency that would arise if the domain-separation label were made a function of the ciphertext Merkle root, and avoid the view-change liveness pathology that would arise if the label were view-indexed.

The *encryption domain-separation label* is a function of publicly known pre-commitment data that is stable across view changes within an epoch:

$$\text{id}_{\text{enc}}(h, E) := H_{\text{dom}}(\text{"MOXIE_CP3_ENC"} \parallel \text{epoch_pk}(E) \parallel h) \in \{0, 1\}^{256},$$

where $E = \text{epoch}(h) := \lfloor h/E_{\text{blocks}} \rfloor$ is the epoch index containing height h (with epoch length E_{blocks} per Definition 5.1), $\text{epoch_pk}(E) \in G_1$ is the epoch-wide threshold-EIGamal aggregate public key produced by the DKG of Definition 9.1 and re-shared at most every E_{rot} epochs, $H_{\text{dom}} : \{0, 1\}^* \rightarrow \{0, 1\}^{256}$ is a domain-separated collision-resistant hash modelled as a random oracle (any instantiation such as keyed BLAKE3 [20] or SHA-256 suffices), and \parallel denotes concatenation. The label is a byte string, not a group element, and is passed as an input to the KDF of the hashed-EIGamal DEM of Definition 9.1; it is *not* a Boneh-Franklin IBE identity and is not mapped to G_2 at any point. A user submitting an encrypted transaction targeted for block-height h computes $\text{id}_{\text{enc}}(h, E)$ from $(h, E, \text{epoch_pk}(E))$ alone: all three quantities are public and finalized before any proposal at height h is broadcast, and none of them changes when the view advances from r to $r + 1$ within epoch E . Ciphertexts whose KDF inputs include $\text{id}_{\text{enc}}(h, E)$ therefore survive view changes within the epoch; re-encryption is required only at the epoch boundary (every E_{blocks} blocks, measured in height).

The *scheduling tag* is the deterministic function

$$\text{sched} : \mathbb{N} \times \{0, 1\}^{256} \times \mathcal{V} \times \mathbb{N} \times \{0, 1\}^{256} \rightarrow \{0, 1\}^{512},$$

$$\text{sched}(h, \tau, v_i, r, R_{\text{committed}}) := H_{\text{dom}}(\text{"MOXIE_CP3_SCHED"} \parallel h \parallel \tau \parallel r \parallel R_{\text{committed}}) \parallel H_{\text{dom}}(\text{"MOXIE_CP3_VID"} \parallel h \parallel v_i \parallel R_{\text{committed}})$$

which assigns to each block height h , transaction hash τ , validator v_i , view r , and committed Merkle root $R_{\text{committed}}$ a tag that v_i publishes alongside its decryption share. The tag is not an input to the threshold-decryption primitive: threshold hashed-EIGamal (Definition 9.1) takes only (sk_i, C_1) as inputs to share production, where $C_1 \in G_1$ is the ephemeral group element of the ciphertext. It is instead a *consensus-layer* artifact: the honest share-release protocol of Section 3.3 requires the validator to broadcast the share together with its sched-tag and an attestation that the tag was derived from a $(h, \tau, r, R_{\text{committed}})$ tuple fixed at $\text{QC}_{\text{Prepare}}$ formation. Shares broadcast with a tag that does not match the authoritative $R_{\text{committed}}$ of the current QC are rejected by honest recipients *prior to threshold combination* and trigger slashing against the signing validator; see Definition 10.5(c)-(d). The

validator-identity component of the tag ensures per-validator uniqueness for accountability.

Time separation. The encryption domain label id_{enc} is fixed at user-submission time from $(h, E, \text{epoch_pk}(E))$ alone (no dependency on view, proposer, ciphertext contents, or Merkle root); the scheduling tag sched is fixed at $\text{QC}_{\text{Prepare}}$ formation from the block-authoritative $R_{\text{committed}}$ and the current view r . The decryption binding to $R_{\text{committed}}$ is *structural*, enforced by the state-machine gate of Definition 10.5(c) and the slashable tag-attestation rule of Definition 10.5(d), and is not a property of the underlying threshold-ElGamal primitive.

Proposition 10.4 (Schedule properties). *The identity schedule of Definition 10.3 satisfies:*

- (i) **Encryption-time computability and view-change robustness.** *The encryption domain label $\text{id}_{\text{enc}}(h, E)$ is a function only of $(h, E, \text{epoch_pk}(E))$; no input depends on the view r , the proposer $\text{pk}_{\text{prop}}(h, r)$, or the ciphertext set encrypted under epoch key $\text{epoch_pk}(E)$ and domain label $\text{id}_{\text{enc}}(h, E)$. A user submitting a transaction to the mempool for inclusion at height h computes $\text{id}_{\text{enc}}(h, E)$ once at submission time; the resulting ciphertext remains decryptable under any view $r, r + 1, r + 2, \dots$ within epoch E , so view changes within the epoch do not require re-encryption.*
- (ii) **Determinism.** *Every honest validator computes the same scheduling tag $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$ and the same encryption domain label $\text{id}_{\text{enc}}(h, E)$ from inputs observable pre-block-construction (for id_{enc}) or committed into the block header and $\text{QC}_{\text{Prepare}}$ (for sched); no validator-local or private state is used.*
- (iii) **Collision-freeness.** *For any two distinct tuples $(h, \tau, v_i, r, R) \neq (h', \tau', v_j, r', R')$, the scheduling tags collide only with probability at most 2^{-256} under the random-oracle model on H_{dom} . For the encryption domain label, collisions at fixed h across distinct epochs (or at fixed epoch across distinct heights) reduce to H_{dom} pre-image collisions on the input byte string `"MOXIE_CP3_ENC" || epoch_pk(E) || h`: negligible under the random-oracle model on H_{dom} with probability at most 2^{-256} per collision event. The claim reduces to H_{dom} hash-collisions and does not depend on the proposer public key.*
- (iv) **Timing-tightness of share-derivation.** *The scheduling tag's input $R_{\text{committed}}$ becomes authoritatively binding at $\text{QC}_{\text{Prepare}}$ formation: prior to $\text{QC}_{\text{Prepare}}$, the proposer may unilaterally propose candidate Merkle roots, but no $2f + 1$ -quorum signature exists to commit one; after $\text{QC}_{\text{Prepare}}$, any deviation is a signed-equivocation slashable under Invariant 6.7.*
- (v) **Consistency across validators.** *If two honest validators v_i, v_j both observe $\text{QC}_{\text{Prepare}}(B, h, r)$, they derive the same $\text{id}_{\text{enc}}(h, E)$ (from the publicly known epoch $E = \lfloor h/E_{\text{blocks}} \rfloor$ and the DKG-produced $\text{epoch_pk}(E)$), and each computes its own scheduling tag $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$, $\text{sched}(h, \tau, v_j, r, R_{\text{committed}})$; their respective decryption shares, each produced as $\text{ds}_i = \text{sk}_i \cdot C_1 \in G_1$ on the ciphertext's ephemeral element C_1 shared by every party, combine by Lagrange interpolation in G_1 to yield $\sum_{i \in S} \lambda_{i,S} \text{ds}_i = \text{sk} \cdot C_1$, reproducing the hashed-ElGamal shared secret (see Definition 9.1).*
- (vi) **Adversary-resistance.** *An adversary \mathcal{A} controlling fewer than t validators cannot, for any ciphertext \tilde{c} at height h (encrypted under epoch key $\text{epoch_pk}(E)$ with domain label $\text{id}_{\text{enc}}(h, E)$) whose t honest shares have not yet been released, compute $\text{sk} \cdot C_1$ except with probability negligible in λ . This follows from the IND-CPA security of threshold hashed-ElGamal in G_1 , which reduces to Computational Diffie-Hellman in G_1 under the random-oracle model on the KDF (equivalently, to Decisional Diffie-Hellman in G_1 without the ROM on the KDF); controlling $< t$ shares of the degree- $(t-1)$ secret polynomial yields no information on the threshold secret beyond the public commitments [18, 17]. (The scheduling tag sched plays no role in this primitive-level argument; its role is state-machine timing, not cryptographic identity-matching; see Definition 10.5(b)-(d).)*

Proof. (i) is immediate from Definition 10.3: $\text{id}_{\text{enc}}(h, E)$'s inputs are the target height h (consensus-layer constant), the epoch $E = \lfloor h/E_{\text{blocks}} \rfloor$ (deterministic function of h), and

the epoch threshold public key $\text{epoch_pk}(E)$ (finalized at the start of epoch E by the DKG of Definition 9.1); no input is a function of the view r , the proposer identity pk_{prop} , or any ciphertext contents, so the domain label is stable across view changes within the epoch. The circular dependency that would arise from including $R_{\text{committed}}$ (a Merkle root of ciphertexts whose KDF inputs include id_{enc}) is avoided by construction. (ii) and (v) are immediate from the definition: sched and id_{enc} are hash-function applications on values that either appear in the block header, are fixed constants, or are epoch-level DKG outputs; any validator processing the same $\text{QC}_{\text{prepare}}$ and the same (h, E) computes the same outputs. (iii) is a standard random-oracle argument on H_{dom} with domain-separated inputs; the framing is pre-image/output-collision on a random oracle, not a discrete-log statement. (iv) follows from CP1 (Definition 10.1) applied to $R_{\text{committed}}$ and the $\text{QC}_{\text{prepare}}$ formation rule: the tuple $(R_{\text{committed}}, h, r)$ is attested by $2f + 1$ validator signatures, so forging a different authoritative value requires forging $2f + 1 - |\mathcal{B}| \geq f + 1$ honest signatures, which is unforgeable under BLS by assumption. (vi) is the standard threshold hashed-ElGamal IND-CPA reduction: a distinguisher gaining non-negligible advantage on an encryption under $(\text{epoch_pk}(E), \text{id}_{\text{enc}}(h, E))$ with fewer than t compromised shares yields a CDH (resp. DDH) distinguisher on G_1 via the standard hybrid argument over the KDF (resp. directly), per [17]. The domain label's role is KDF separation, not identity-based key extraction. \square

Definition 10.5 (CP3: structurally-bound commit-before-decrypt). CP3 is a *structural* (not cryptographic) constraint on the decryption-share release. Against the composed adversary $\mathcal{A}_{\text{MBFT}}$ of §16.4 with $|\mathcal{A}_{\text{MBFT}}| \leq f < t$, CP3 guarantees that no adversary extracts plaintext before $\text{QC}_{\text{prepare}}$ forms (a cryptographic impossibility for $< t$ -share adversaries, case (i) below); against stronger adversaries of size $[f + 1, t - 1]$, CP3 preserves the IND-CPA security of the underlying primitive while specification-conforming gating fails. The binding to the authoritative $R_{\text{committed}}$ is delivered by a combination of (a) an epoch-stable encryption domain label, (b) a structural scheduling tag with consensus-layer BLS-signed attestation, (c) a state-machine gate, and (d) slashing, rather than by a cryptographic modification of the underlying threshold-decryption primitive; the primitive itself remains the unmodified threshold hashed-ElGamal of Definition 9.1. A cryptographically-tight tag-binding variant via Boneh-Franklin IBE is tracked as future work.

- (a) **Epoch-stable domain label (cryptographic)**. Users encrypt transactions under the epoch threshold public key $\text{epoch_pk}(E) \in G_1$, binding each ciphertext to the domain label $\text{id}_{\text{enc}}(h, E)$ of Definition 10.3 (passed into the KDF of Definition 9.1); the label is a deterministic function of the target height h , its epoch $E = \lfloor h / E_{\text{blocks}} \rfloor$, and $\text{epoch_pk}(E)$: all quantities fixed at submission time and stable across view changes within epoch E . Each validator v_i 's decryption share is produced by the standard two-argument threshold-ElGamal share function $\text{Dec_share}(\text{sk}_i, C_1) = \text{sk}_i \cdot C_1 \in G_1$, where $C_1 \in G_1$ is the ciphertext's ephemeral element, and sk_i is the Lagrange share of the threshold secret key produced by the DKG of Definition 9.1. Lagrange combination of any t such shares at coefficients $\{\lambda_{i,S}\}$ yields $\sum_{i \in S} \lambda_{i,S} \text{ds}_i = \text{sk} \cdot C_1$, which reproduces the shared secret $\text{sk} \cdot C_1 = r \cdot \text{epoch_pk}(E)$ used by the hashed-ElGamal DEM to derive the symmetric key and recover the plaintext of any ciphertext \tilde{c} encrypted under $(\text{epoch_pk}(E), \text{id}_{\text{enc}}(h, E))$; the combination function takes only $(\tilde{c}, \{\text{ds}_i\}_{i \in S})$ as inputs, with $|S| = t$. No scheduling tag is an argument to either Dec_share or Combine ; the underlying cryptographic primitive is the unmodified threshold hashed-ElGamal of Definition 9.1.
- (b) **Structural share-release binding (consensus-layer)**. Before broadcasting its decryption share ds_i , each validator v_i computes the scheduling tag $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$

of Definition 10.3 and signs the share-release attestation

$$a_i := \text{Sig}_{\text{sk}_i^{\text{BLS}}}(\text{"MOXIE_CP3_ATTEST"} \parallel h \parallel \tau \parallel r \parallel R_{\text{committed}} \parallel v_i),$$

where $R_{\text{committed}}$ is the Merkle root bound by the $\text{QC}_{\text{prepare}}$ for block (h, r) . The share-release message broadcast by v_i is the triple $(\text{ds}_i, \text{sched}(h, \tau, v_i, r, R_{\text{committed}}), a_i)$. Honest recipients reject any share whose accompanying attestation a_i does not verify under v_i 's BLS public key, does not bind the view r and root $R_{\text{committed}}$ attested by the current $\text{QC}_{\text{prepare}}$, or whose tag is inconsistent with the signed tuple; rejected shares do not enter threshold combination. The binding of share release to the authoritative $R_{\text{committed}}$ is therefore a *consensus-layer* property, enforced by honest-recipient filtering plus the signed-equivocation evidence generated when a validator releases a_i under one root and then a different attestation under a stale or speculative root. It is not a cryptographic property of the combine primitive.

- (c) **State-machine gate.** By the transition function of Section 3.3, the share-release transition is reachable only from the post- $\text{QC}_{\text{prepare}}$ state, so a specification-conforming validator cannot invoke Dec_share or emit the attestation a_i before observing $\text{QC}_{\text{prepare}}$. The gate removes the honest-validator fraction from the pool of possible early-release sources.
- (d) **Slashing enforcement.** A validator whose share (or whose attestation a_i) surfaces on the network with a tag bound to any $R' \neq R_{\text{committed}}$ attested by the current $\text{QC}_{\text{prepare}}$, or absent any accompanying $\text{QC}_{\text{prepare}}$ evidence at the current view, is slashable under the ‘‘Double signing’’ offense line of the slashing schedule (Table 2, $r_{\text{cp3}} = 0.05$). The attestation a_i provides cryptographically-indisputable signed-equivocation evidence: the slashed validator has endorsed two distinct (h, τ, r, R) tuples with the same (h, τ, r) prefix. Slashing disciplines rational validators who might attempt to circumvent the state-machine gate.

What CP3 guarantees. Under (a)-(d), any adversary \mathcal{A} controlling fewer than t validators that obtains a plaintext of \tilde{c} before $\text{QC}_{\text{prepare}}$ forms must either (i) compute $\text{sk} \cdot C_1 \in G_1$ on the ciphertext's ephemeral element C_1 without controlling $\geq t$ key shares, which requires distinguishing a hashed-ElGamal ciphertext from random and therefore solving CDH in G_1 (with KDF modelled as a random oracle) or DDH in G_1 directly, violating the IND-CPA security of threshold hashed-ElGamal [17, 18]; or (ii) induce at least $f + 1$ honest validators to release signed attestations a_i before they observe $\text{QC}_{\text{prepare}}$, contradicting specification-conforming behaviour (the state-machine gate (c)); or (iii) rely on $\geq f + 1$ Byzantine validators releasing their shares out-of-gate, each of whom generates slashable signed evidence (the attestation (d)) making the attack economically detectable and punishable. Case (i) is a cryptographic impossibility; cases (ii)-(iii) are structural, not cryptographic, and are discharged by the consensus-layer rules rather than by a modification of the encryption primitive. This is a strictly weaker guarantee than a cryptographic tag-binding in the combine primitive would provide, but it is the guarantee that the consensus layer can honestly deliver and it suffices for Theorem 10.9 provided the adversarial corruption $\text{cap } f < t$ is honoured.

Primitive choice. An earlier approach would cryptographically enforce the tag sched via a modified four-argument Dec_share primitive whose combination rule rejects mismatched tags, with Boneh-Franklin IBE as the underlying cryptosystem. Such a four-argument share function does not exist in any standard threshold cryptosystem and Boneh-Franklin IBE adds pairing-based share extraction not needed for the consensus argument. We therefore specify threshold *hashed-ElGamal* over G_1 (no pairing, no hash-to- G_2 on the encryption/decryption path) with the unmodified two-argument $\text{Dec_share}(\text{sk}_i, C_1) = \text{sk}_i \cdot C_1$, and route the $R_{\text{committed}}$ binding through the consensus-layer attestation a_i of item (b). Primitive security is stated uniformly as IND-CPA under CDH-with-KDF-in-

ROM (equivalently DDH without ROM) in G_1 . The binding that *survives this framing* is weaker than full cryptographic immunity to out-of-gate share release. Specifically, *beyond the $\mathcal{A}_{\text{MBFT}}$ threat model of §16.4* (which caps $|\mathcal{A}_{\text{MBFT}}| \leq f$), an adversary controlling $f^* \in [f + 1, t - 1]$ validators (i.e., more than the Byzantine cap but fewer than the decryption threshold) who ignores slashing can still inject out-of-gate shares, but cannot decrypt without crossing the IND-CPA threshold, and every such injection generates signed-equivocation evidence. This extended analysis is offered as a consolation against a stronger-than-modelled adversary, not as a guarantee required by MoxieBFT’s stated security model: an adversary of size $> f$ is outside $\mathcal{A}_{\text{MBFT}}$ and other safety guarantees (Agreement, Liveness) may already fail against it. Migration to a Boneh-Franklin IBE layer (which would strengthen the primitive-level guarantee from IND-CPA to IND-ID-CPA and enable pairing-based share verification by public keys) is tracked as future work and is not assumed in any proof of the present paper.

Proposition 10.6 (CP3 requires identity-schedule consistency). *CP3 enforcement depends on every honest validator computing the same $\text{id}_{\text{enc}}(h, E)$ and attesting consistently to the same $R_{\text{committed}}$ under the observed $\text{QC}_{\text{prepare}}(B, h, r)$. Formally: let $v_i, v_j \in \mathcal{H}$ each observe a quorum certificate $\text{QC}_{\text{prepare}}$ for block B at (h, r) . Then (I) v_i and v_j derive identical encryption domain labels $\text{id}_{\text{enc}}(h, E)$, so their respective shares ds_i, ds_j are combinable by Lagrange interpolation in G_1 on the ciphertext’s ephemeral element C_1 (Definition 9.1); and (II) v_i and v_j sign attestations a_i, a_j (per Definition 10.5(b)) that bind the identical $R_{\text{committed}}$ extracted from B , so neither share is rejected by an honest recipient.*

Proof. (I) By Proposition 10.4(ii) and (v): determinism plus consistency-across-validators. The encryption domain label $\text{id}_{\text{enc}}(h, E)$ is a function of $(h, E, \text{epoch_pk}(E))$ alone, all three of which are determined by public consensus-layer state observable well before block h is constructed and *independent of the view r* (which may advance via view changes without perturbing the label); both honest validators compute the same value. Their respective shares $\text{ds}_i = \text{sk}_i \cdot C_1$ (on the ciphertext’s ephemeral element $C_1 \in G_1$, shared by every party) therefore satisfy the correctness property of threshold hashed-ElGamal: any t of them, combined via Lagrange interpolation in G_1 , yield $\text{sk} \cdot C_1 = r \cdot \text{epoch_pk}(E)$, which fed into the KDF (with domain label $\text{id}_{\text{enc}}(h, E)$) recovers the plaintext of any ciphertext encrypted under that epoch key and domain label.

(II) $R_{\text{committed}}$ is a field of B , uniquely determined by the observed $\text{QC}_{\text{prepare}}$ (which signs over B); the height h and view r are also uniquely determined by the QC. Hence every honest validator computes the same $(h, \tau, r, R_{\text{committed}})$ tuple for its scheduling tag, signs the attestation a_i of Definition 10.5(b) over that common tuple, and accepts the corresponding attestations from other honest validators. The attestations a_i, a_j differ only in the signing validator’s identity; both bind the same $R_{\text{committed}}$. The tags $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$ and $\text{sched}(h, \tau, v_j, r, R_{\text{committed}})$ likewise differ only in their validator-identity second component, supporting per-validator accountability without affecting the combinability of their shares (which depends only on $\text{id}_{\text{enc}}(h, E)$).

If any honest validator computed a different id'_{enc} (via a different (h, E) or a different epoch_pk), its decrypted symmetric key would mismatch that of honest peers after Lagrange combination and KDF-evaluation, so no plaintext would be recovered; if it signed an attestation under a different $R'_{\text{committed}}$, honest recipients would reject the share at the consensus layer. Hence consistency of the identity schedule across \mathcal{H} , both the cryptographic domain label id_{enc} and the consensus-layer attestation a_i , is necessary for CP3 enforcement; it is supplied by the determinism of Definition 10.3, the epoch-level agreement on $\text{epoch_pk}(E)$, and the agreement of all honest validators on the contents of any observed quorum certificate (Theorem 6.3). \square

Definition 10.7 (CP4: Execution-commitment consistency). After decryption reveals plaintexts, the executed ordering must produce $R_{\text{executed}} = R_{\text{committed}}$. Any validator that commits to a block where $R_{\text{executed}} \neq R_{\text{committed}}$ is slashable for equivocation at rate $r_{\text{cp4}} = 0.05$.

Remark 10.8 (CP4 cumulative-cap interaction). The per-offense slash rate r_{cp4} combined with the cumulative cap r_{slash} of Table 2 bounds the number of individually-penalized equivocations at a single validator by $\lfloor r_{\text{slash}}/r_{\text{cp4}} \rfloor$; further offenses saturate the cap and carry zero marginal slash. The deterrent past the cap is the slashing-dilution floor of §14.1: once the cap is reached the validator is ejected, and consensus halts explicitly if ejection breaches the invariant $S(h) \geq \lceil \frac{2}{3}S_0 \rceil$.

Theorem 10.9 (Frontrun-freeness under the MBFT adversary). *Under the standard cryptographic assumptions (collision-resistant SHA-256 and IND-CPA threshold hashed-ElGamal on BLS12-381 G_1 , reducing to CDH-with-KDF-in-ROM (equivalently DDH without ROM) in G_1), a protocol satisfying CP1-CP4 admits no frontrunning attack against transactions in the encrypted mempool under the composed adversary $\mathcal{A}_{\text{MBFT}}$ of §16.4. Formally, against $\mathcal{A}_{\text{MBFT}}$ with corruption cap $|\mathcal{A}_{\text{MBFT}}| \leq f < t$, no adversary can insert, reorder, or condition a transaction on the plaintext of another transaction within the same block, except with negligible probability in λ . The bound $|\mathcal{A}_{\text{MBFT}}| \leq f$ is essential: CP3's binding of share release to $R_{\text{committed}}$ is structural (BLS-signed attestations, state-machine gate, slashing) rather than cryptographic, so the statement does not extend to the intermediate regime $|\mathcal{A}| \in [f + 1, t - 1]$.*

Proof. Fix a target ciphertext \tilde{c}_{tgt} in the committed sequence. To frontrun, $\mathcal{A}_{\text{MBFT}}$ must either (a) learn plaintext(\tilde{c}_{tgt}) before $R_{\text{committed}}$ is finalized, or (b) reorder after learning it.

Case (a) violates CP2: learning the plaintext before decryption shares are released requires either breaking IND-CPA security (negligible by assumption) or obtaining t shares (impossible since $\mathcal{A}_{\text{MBFT}}$ controls at most $f < t$ validators).

Case (b) violates CP4: after decryption, the executed ordering must match $R_{\text{committed}}$; any departure is Merkle-root-verifiable and triggers slashing under CP4. Detection is deterministic and each honest validator computes R_{executed} independently.

CP1 ensures $R_{\text{committed}}$ itself cannot be silently substituted after broadcast, and CP3 ensures decryption cannot begin before the commit is finalized in a QC: Definition 10.5(b)-(d) routes the binding through BLS-signed attestations, the state-machine gate, and slashing on out-of-gate release, under the $f < t$ corruption cap. The combination rules out both case (a) and case (b). \square

Proposition 10.10 (Enforcement correspondence). *CP1-CP4 correspond to the following enforcement points in the specified state machine:*

- **CP1** is enforced by the PREPARE-phase Merkle-root commitment step (Section 3.1): $R_{\text{committed}}$ is bound into the block header and signed into $\text{QC}_{\text{Prepare}}$. Its binding property reduces to SHA-256 collision resistance.
- **CP2** is enforced by the threshold hashed-ElGamal scheme of Definition 9.1 with threshold $t = q(n) = \lceil 2n/3 \rceil$; the share-release gate is the COMMIT-phase transition of Section 3.1.
- **CP3** is enforced structurally rather than cryptographically: the share-release transition is reachable only from the post- $\text{QC}_{\text{Prepare}}$ state (Section 3.3), and each share is broadcast together with the scheduling tag $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$ and a BLS-signed attestation a_i (Definition 10.5(b)) binding the authoritative $R_{\text{committed}}$ of the current $\text{QC}_{\text{Prepare}}$. Honest recipients drop any share whose attestation does not match the current QC's $R_{\text{committed}}$ before combination; misbehaving validators leave signed-equivocation evidence usable for slashing (Definition 10.5(d)). The threshold-ElGamal combine primitive itself is the unmodified Lagrange-interpolation combiner of Definition 9.1 and takes only $(\tilde{c}, \{ds_i\}_{i \in S})$ as inputs: no scheduling tag is an argument to the cryptographic combine.

- **CP4** is enforced by the DECIDE-phase check that recomputes R_{executed} from the decrypted transaction sequence and compares to $R_{\text{committed}}$; mismatch aborts the DECIDE vote and emits a slashing event against the proposer.

10.1 Five-Round Message-Complexity Lower Bound

Definition 3.10 asserts that the consensus latency of MoxieBFT is bounded below by $5 \cdot \ell_{\text{med}}$. We now prove that the five rounds form a *lower bound* for the class of protocols satisfying CP1-CP4 simultaneously with BFT-agreement and event-atomic settlement. No lower bound for this composite setting appears in the prior literature on BFT round complexity surveyed ([1, 3, 7, 15]); the argument below establishes the first such bound for the surveyed class of protocols.

Model. Let Π be a deterministic message-passing protocol run by $n = 3f + 1$ parties in the partial-synchrony model of [12], with the post-GST round structure standard in HotStuff-style pedagogy: a *message round* is one broadcast step of all honest parties followed by adversary-controlled delivery. We write $r(\Pi)$ for the maximum number of message rounds any honest party executes before finalizing a block in an execution with no view changes after GST (the “steady-state” or “happy-path” cost). The lower bound is on $r(\Pi)$ in the steady state; view-change overhead is additive and standard.

Definition 10.11 (Steady-state round sequence). A steady-state execution of Π yields a finite sequence of message rounds $\rho_1, \rho_2, \dots, \rho_{r(\Pi)}$ such that: (i) each ρ_k is either a *leader-to-all* broadcast or an *all-to-all* or *all-to-leader* aggregation; (ii) every honest party’s decision value is a function of the messages it has received by the end of $\rho_{r(\Pi)}$; (iii) no honest party sends a value in ρ_k that depends on messages it receives during ρ_k (causality within a round). Two round indices ρ_k and ρ_{k+1} are *distinct rounds* if at least one honest party’s message in ρ_{k+1} causally depends on a message it received in ρ_k .

Definition 10.12 (Protocol class $\mathcal{P}_{\text{CP,BFT,EA}}$). A protocol Π belongs to $\mathcal{P}_{\text{CP,BFT,EA}}$ iff it satisfies:

- (C1) CP1-CP4 (Definitions 10.1-10.7) under IND-CPA threshold hashed-ElGamal in G_1 (reducing to CDH-with-KDF-in-ROM or DDH in G_1) and collision-resistant hashing;
- (C2) Byzantine fault tolerance at $f < t$ where $t = q(n) = \lceil 2n/3 \rceil$ is the decryption threshold, in the intersection-adversary model (§16.4);
- (C3) BFT-agreement under post-GST partial synchrony: any two honest parties that decide, decide on the same value;
- (C4) event-atomic settlement: at no point in the execution is a partial state visible to a read-only observer that violates observer opacity (Theorem 13.3).

The four round-merger impossibility lemmas below each forbid collapsing a pair of adjacent rounds in a protocol from $\mathcal{P}_{\text{CP,BFT,EA}}$. The first four rounds correspond to the four cryptographic / consensus *gates* that must fire in sequence: ordering-commit, lock-formation, decrypt-gate, and execution-commit. The fifth round carries the agreement QC over the executed ordering.

Lemma 10.13 (Propose and Prepare cannot merge). *Let $\Pi \in \mathcal{P}_{\text{CP,BFT,EA}}$. Then the round in which the leader broadcasts the ordering-commitment $R_{\text{committed}}$ and the round in which $2f + 1$ validators vote on that commitment must be distinct rounds.*

Proof. Suppose for contradiction that Π merges these into a single round ρ_k . By the within-round causality rule (Definition 10.11(iii)), no validator’s ρ_k -vote can causally depend on the leader’s ρ_k -broadcast. Hence every validator must commit its vote without having verified $R_{\text{committed}}$: it votes “blind”.

Two cases: either (a) the vote is a pre-committed unconditional signature (a “pre-broadcast” vote over the yet-unobserved root), or (b) the vote is post-hoc verifiable but the validator is bound by the vote regardless.

Case (a) requires validators to broadcast signatures over $R_{\text{committed}}$ before $R_{\text{committed}}$ is defined. This either (i) fixes $R_{\text{committed}}$ deterministically by some pre-image available to all parties, which by CP1 must be collision-binding and therefore fixed before ρ_k , violating the assumption that ρ_k is the broadcast round; or (ii) makes every validator’s signature meaningless for CP1-binding purposes, i.e., the signature does not bind the ordering to the committed root, so CP1 fails. Contradiction.

Case (b) is the causal-gate obstruction for single-round commit-vote. With $n = 3f + 1$, two conflicting $2f + 1$ certificates cannot both avoid honest overlap; quorum intersection would contain at least $f + 1$ validators. The point is instead that a vote fixed before observing the leader’s root is not evidence of having validated that root. It is therefore either a signature on a value determined before the broadcast (case (a)) or a non-validating certificate that fails the CP1 binding requirement. In both subcases the merged round does not implement the required commit gate. \square

Lemma 10.14 (Prepare and PreCommit cannot merge). *Let $\Pi \in \mathcal{P}_{\text{CP,BFT,EA}}$. Then the PREPARE-vote round (forming $\text{QC}_{\text{Prepare}}$) and the PRECOMMIT-vote round (forming the lockedQC) must be distinct rounds.*

Proof. A merge would require validators to simultaneously (i) cast a PREPARE vote over $R_{\text{committed}}$ and (ii) lock on $\text{QC}_{\text{Prepare}}$. By within-round causality, no validator’s lock in ρ_k can causally depend on a $\text{QC}_{\text{Prepare}}$ formed in ρ_k : the QC does not yet exist when the lock is cast.

Hence validators must lock on a *prior* QC, say $\text{QC}_{\text{Prepare}}^{(k-1)}$ from a prior view. By Definition 10.5(b)-(d), the consensus-layer binding of share release to the authoritative $R_{\text{committed}}$ requires each validator to sign an attestation a_i over the $(h, \tau, r, R_{\text{committed}})$ tuple extracted from the current view’s $\text{QC}_{\text{Prepare}}$. The scheduling tag $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$ is a function of $R_{\text{committed}}$ at *this* view, not at $(h, r-1)$. Locking on $\text{QC}_{\text{Prepare}}^{(k-1)}$ therefore does not bind the share release to the current view’s commitment; an adversary can cause parties to release shares attested under a stale root. Concretely, a proposer in view $r-1$ can choose $R_{\text{committed}}^{(r-1)}$ adversarially before $R_{\text{committed}}^{(r)}$ is known, so attestations over $R_{\text{committed}}^{(r-1)}$ are not bound to the current block’s ordering; honest recipients reject such shares before combination (Definition 10.5(b)), and the protocol either stalls (no t attestations bound to $R_{\text{committed}}^{(r)}$) or accepts mismatched attestations (violating the consensus-layer rule and producing slashable evidence).

Alternatively, a $\text{QC}_{\text{Prepare}}$ could form “simultaneously” with the lock vote via a single-round multi-step composition (e.g. synchronous gossip within a round). But such composition makes the subround a new round in the sense of Definition 10.11, reintroducing the count. Contradiction. \square

Lemma 10.15 (PreCommit and Commit-decrypt cannot merge). *Let $\Pi \in \mathcal{P}_{\text{CP,BFT,EA}}$. Then the PRECOMMIT-vote round (forming lockedQC) and the COMMIT-phase decryption-share release round must be distinct rounds.*

Proof. Definition 10.5(b)-(d) states that a valid decryption share ds_i must be broadcast together with a scheduling tag $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$ and a signed attestation a_i over the authoritative $R_{\text{committed}}$, which by Proposition 10.4(iv) is *authoritatively finalized* only at $\text{QC}_{\text{Prepare}}$ formation. The PreCommit lock certifies that $\text{QC}_{\text{Prepare}}$ is definitive (not a speculative value the leader could later replace). A merge would require validators to release shares in round ρ_k while also forming the PreCommit QC in ρ_k . By within-round

causality, the share release cannot causally depend on the PreCommit QC formed in the same round.

Two cases. (a) Shares are released before the PreCommit QC. This releases shares before the commit-gate fires. By CP2 (Definition 10.2), the plaintext becomes recoverable as soon as t honest shares are released on the common ciphertext's ephemeral element C_1 (the domain label $\text{id}_{\text{enc}}(h, E)$ is a function only of (h, E) and the epoch public key, so all ciphertexts for height h share the same KDF input regardless of view). The scheduling tag and attestation are consensus-layer artifacts that do *not* affect whether Lagrange combination in G_1 succeeds: any t syntactically-valid shares $ds_i = sk_i \cdot C_1$ combine by Lagrange interpolation to recover $sk \cdot C_1$ and hence the plaintext. If $2f + 1$ validators release shares in ρ_k , let t be the decryption threshold. In the canonical $n = 3f + 1$ regime (Definition 2.1) we have $t = 2f + 1$, so an adversary who obtains the $2f + 1$ released shares decrypts *before* the PreCommit QC exists; the plaintext is revealed before the ordering is locked, and a Byzantine leader can substitute a different ordering at a new view (CP1 does not yet bind because no QC has witnessed it). This violates CP3's timing-binding. For non-canonical $n \in \{3f + 2, 3f + 3\}$ we have $t > 2f + 1$; decryption then requires Byzantine contribution (honest $2f + 1$ plus Byzantine $t - (2f + 1) \geq 1$ to reach the threshold), and the same merge attack succeeds via at most f Byzantine validators contributing their shares alongside the $2f + 1$ honest ones ($n \geq t$ in all three canonical-adjacent regimes). In every case the attack decrypts before the PreCommit QC forms. Contradiction.

(b) Shares are released under a *speculative* attestation (the leader's proposed but un-certified $R_{\text{committed}}$). By Definition 10.5(b), honest recipients reject shares whose attestation a_i does not bind the $R_{\text{committed}}$ of the current $\text{QC}_{\text{Prepare}}$; any honest validator who has already observed $\text{QC}_{\text{Prepare}}$ and sees a mismatched attestation discards the share before combination. If the protocol specification instead accepts speculative attestations, it violates the consensus-layer binding rule of Definition 10.5(b) and the slashable-attestation rule of Definition 10.5(d); if it rejects them, the round makes no progress until $\text{QC}_{\text{Prepare}}$ forms, reintroducing a distinct round. Contradiction. \square

Lemma 10.16 (Commit-decrypt and Execute cannot merge). *Let $\Pi \in \mathcal{P}_{\text{CP,BFT,EA}}$. Then the round in which threshold decryption shares are combined (yielding the plaintext ordering) and the round in which the executed ordering is committed (via $\text{QC}_{\text{Decide}}$ binding R_{executed}) must be distinct rounds.*

Proof. A merge would require validators to cast DECIDE votes in the same round as share combination. By within-round causality, no validator's DECIDE vote can depend on a plaintext decrypted in the same round.

Two cases. (a) Validators cast DECIDE votes unconditionally, before verifying that $R_{\text{executed}} = R_{\text{committed}}$. Then the CP4 consistency check never fires pre-vote; a Byzantine leader can propose a block whose plaintext sequence hashes to a root distinct from $R_{\text{committed}}$, and honest validators will sign the $\text{QC}_{\text{Decide}}$ over the mismatched root. CP4 then either (i) is never enforced, violating (C1), or (ii) is enforced post-hoc via slashing, but the block has already been finalized, and the event-atomic settlement property (13.3) fails because an observer sees a committed ordering inconsistent with the advertised commitment, a violation of observer opacity (C4). Contradiction.

(b) Validators defer the DECIDE vote until after verifying R_{executed} : but that verification requires the decrypted plaintext, which is only available after share combination completes in the same round. The DECIDE vote therefore causally depends on an intra-round message, violating Definition 10.11(iii). Equivalently, the proposed merge secretly contains two sub-rounds and is a two-round segment in the sense of the round metric. Contradiction. \square

Theorem 10.17 (Five-round lower bound for CP1-CP4 + BFT + event-atomic). *Let $\Pi \in \mathcal{P}_{\text{CP,BFT,EA}}$ be any deterministic protocol satisfying CP1-CP4, $f < t$ Byzantine fault tolerance, BFT-agreement, and event-atomic settlement (Definition 10.12). Then every steady-state execution of Π requires at least five distinct message rounds:*

$$r(\Pi) \geq 5.$$

In particular, the minimum consensus latency of Π is bounded below by $5 \cdot \ell_{\text{med}}$, where ℓ_{med} is the median post-GST round-trip latency.

Proof. Every protocol in $\mathcal{P}_{\text{CP,BFT,EA}}$ must execute five logical gates in causal order:

- (G1) *Propose:* a leader broadcasts $R_{\text{committed}}$ (CP1).
- (G2) *Prepare:* $\geq 2f + 1$ validators vote on $R_{\text{committed}}$, forming $\text{QC}_{\text{Prepare}}$, which authoritatively fixes the scheduling-tag input $R_{\text{committed}}$ used by the CP3 structural-binding clause (Definition 10.5(b)). The encryption domain label $\text{id}_{\text{enc}}(h, E)$ is fixed earlier, at epoch start; but the consensus-layer attestation binding share release to the authoritative $R_{\text{committed}}$ becomes finalizable only after $\text{QC}_{\text{Prepare}}$.
- (G3) *PreCommit:* $\geq 2f + 1$ validators lock on $\text{QC}_{\text{Prepare}}$, certifying that the commitment is durable across view changes (HotStuff-2 lock).
- (G4) *Commit-decrypt:* $\geq t$ validators release decryption shares $\text{ds}_i = \text{sk}_i \cdot C_1 \in G_1$ on each ciphertext's ephemeral element C_1 (the KDF inputs bind each share to the domain label $\text{id}_{\text{enc}}(h, E)$), each accompanied by the scheduling tag and attestation a_i of Definition 10.5(b); Lagrange combination yields the plaintext ordering (CP2 opens, CP3 structural binding enforced by honest-recipient filtering and slashing).
- (G5) *Decide:* $\geq 2f + 1$ validators verify $R_{\text{executed}} = R_{\text{committed}}$ (CP4) and form $\text{QC}_{\text{Decide}}$, the agreement QC over the executed ordering.

By Lemmas 10.13-10.16, none of the four adjacent pairs (G1,G2), (G2,G3), (G3,G4), (G4,G5) can be merged into a single round without violating at least one of CP1-CP4, BFT-agreement, or event-atomic settlement. Therefore the five gates occupy five distinct rounds:

$$r(\Pi) \geq 5.$$

Each round requires at least one round-trip (leader-to-all broadcast and all-to-leader aggregation in the HotStuff variant, or a single broadcast step in the mempool variant); the median latency cost per round is ℓ_{med} , giving total latency at least $5 \cdot \ell_{\text{med}}$. \square

Remark 10.18 (Matching upper bound: tightness). MoxieBFT (Section 3.1) is a protocol in $\mathcal{P}_{\text{CP,BFT,EA}}$ with exactly five steady-state rounds (EVENT-COLLECT/PROPOSE, PREPARE, PRECOMMIT, COMMIT-decrypt, DECIDE). The bound is therefore *tight*: $r_{\min}(\mathcal{P}_{\text{CP,BFT,EA}}) = 5$. The EVENT-COLLECT step in MoxieBFT overlaps the PROPOSE gate (G1): the leader's block broadcast in round 1 includes both the oracle attestation set and the Merkle-root commitment $R_{\text{committed}}$, so a single round suffices for both, and does not increase the bound.

Scope of tightness. The fusion of Event-Collect with Propose at G_1 is a design choice specific to MoxieBFT. A deployment that separates the two into distinct rounds (for instance, because attestation propagation requires a dedicated broadcast completed before the proposal is constructed) lies in $\mathcal{P}_{\text{CP,BFT,EA}}$ with six rounds; the five-round lower bound of Theorem 10.17 remains valid but is not tight for such a variant. Tightness at five is a property of MoxieBFT's specific implementation (fused G_1), not a property of every protocol in the class.

Remark 10.19 (What the bound does *not* cover). Theorem 10.17 concerns the steady-state happy path. View changes, leader timeouts, and recovery from partial crashes add rounds additively; these are bounded by the standard HotStuff-2 view-change analysis

([3]) and are not the subject of this theorem. The bound is also stated for deterministic protocols; probabilistic protocols (e.g. those using common coins) may trade expected rounds for worst-case rounds but cannot drop below five in expectation under *full* event-atomic + CP1-CP4 by the same causality argument applied in expectation.

Corollary 10.20 (Relaxations that unlock fewer rounds). *Dropping any single condition from Definition 10.12 strictly reduces the round count:*

- *Dropping (C1) (CP1-CP4) recovers the standard HotStuff-2 round complexity, which is 3 rounds in the two-chain lock regime [3].*
- *Dropping (C4) (event-atomic settlement) allows the DECIDE round to merge with COMMIT-decrypt under a post-hoc slashing regime, yielding a 4-round protocol with eventual CP4 enforcement.*
- *Dropping (C3) (BFT-agreement) allows PREPARE and PRECOMMIT to merge under crash-fault tolerance only, yielding a 4-round protocol without Byzantine safety.*

Each relaxation drops a condition of the stated threat model; the five-round count is the exact price of the composed invariant set.

Proof. (C1 relaxation.) Without CP1-CP4 the encryption/commit pipeline is absent, and the protocol reduces to HotStuff-2’s three-phase linear pipeline (Propose, Prepare-vote producing lock, Commit producing decision), which runs in 3 rounds [3].

(C4 relaxation.) Without observer opacity, validators may sign QC_{Decide} over the pre-execution state and defer the CP4 check to post-finalization slashing. The COMMIT-decrypt and DECIDE rounds then fuse: Lemma 10.16’s case (a) no longer contradicts event-atomicity, yielding a 4-round protocol with CP1-CP3 + eventual CP4.

(C3 relaxation.) Under crash-fault tolerance only, equivocation is ruled out by the fault model, so Lemma 10.13’s case (b) disappears. Validators can sign the Prepare and PreCommit votes in a single broadcast (a “Paxos-style” two-phase commit suffices), reducing the total to 4 rounds. \square

Remark 10.21 (Position in the prior-art lower-bound landscape). Classical BFT round-complexity lower bounds establish an $(f + 1)$ -round lower bound for consensus under partial synchrony [12]; an $(f + 2)$ -round lower bound for crash-fault consensus with a single faulty process [14]; and communication-optimal authenticated Byzantine agreement at sub-quadratic complexity [15]. HotStuff-style protocols in partial synchrony achieve three-phase commit [1, 3, 7], which is a protocol upper bound rather than a lower bound; HotStuff-2 collapses this to two phases in the optimistic-responsive regime. The five-round bound of Theorem 10.17 extends this landscape by adding the composite CP1-CP4 invariant: the extra rounds beyond the HotStuff-2 baseline are the cost of binding threshold decryption to a prior-locked ordering before executing and agreeing on the plaintext. No prior bound for the specific class {HotStuff-style BFT + threshold-encrypted mempool + commit-before-decrypt + event-atomic settlement} appears in the literature surveyed; the bound above is established for this class.

11 Parallel Decryption

The threshold decryption of n_{tx} ciphertexts in a block is the most expensive cryptographic step in the pipeline. A naive sequential implementation requires $n_{\text{tx}} \cdot t_{\text{decrypt}}^{\text{single}}$ time. Parallel decryption across P worker threads reduces this wall-clock cost to at most $\lceil n_{\text{tx}}/P \rceil \cdot t_{\text{decrypt}}^{\text{single}} + t_{\text{sync}}(P)$, preserving the committed ordering because each ciphertext is still decrypted only after the consensus gate has fixed its position. The formal bound is stated in Theorem 11.2.

Definition 11.1 (Ciphertext-independence assumption). Ciphertexts in the encrypted mempool are said to be *decryption-independent* if the decryption operation $\text{Dec}_{\text{tk}} : \tilde{c} \mapsto c$ depends only on the ciphertext \tilde{c} and the threshold key tk , with no shared mutable state across decryptions.

Theorem 11.2 (Parallel decryption speedup). *Under Definition 11.1, decrypting n_{tx} ciphertexts on P parallel workers achieves wall-clock time:*

$$t_{\text{decrypt}}^{\text{par}}(P) \leq \left\lceil \frac{n_{\text{tx}}}{P} \right\rceil \cdot t_{\text{decrypt}}^{\text{single}} + t_{\text{sync}}(P),$$

where $t_{\text{sync}}(P) = O(\log P)$ accounts for the final share-gather and commit synchronization. The decrypted plaintexts produced by the parallel scheme are identical to those produced by the sequential scheme.

Proof. By ciphertext-independence, the decryption of ciphertext \tilde{c}_i is a pure function of \tilde{c}_i and tk , commuting with the decryption of any \tilde{c}_j for $j \neq i$. Therefore a partition of the ciphertext set across P workers produces a valid decryption when the results are concatenated, with wall-clock time bounded by the slowest partition plus the synchronization overhead t_{sync} . Output identity follows from the determinism of Dec_{tk} . \square

Corollary 11.3 (Admissibility bound for parallel decryption). *For a deployment with decryption budget t_{decrypt} of Definition 9.2 and per-ciphertext threshold-decryption cost $t_{\text{decrypt}}^{\text{single}}$, Theorem 11.2 is admissible at per-block ciphertext count n_{tx} on P workers iff*

$$\left\lceil \frac{n_{\text{tx}}}{P} \right\rceil \cdot t_{\text{decrypt}}^{\text{single}} + t_{\text{sync}}(P) \leq t_{\text{decrypt}}.$$

Rearranging gives the minimum-parallelism bound

$$P \geq \left\lceil \frac{n_{\text{tx}} \cdot t_{\text{decrypt}}^{\text{single}}}{t_{\text{decrypt}} - t_{\text{sync}}(P)} \right\rceil,$$

valid whenever $t_{\text{decrypt}} > t_{\text{sync}}(P)$; otherwise no per-worker share-release cost admits the budget and either $t_{\text{decrypt}}^{\text{single}}$ must be reduced (e.g., by hardware acceleration of BLS operations) or n_{tx} must be bounded. The admissibility bound is stated purely in terms of deployment-configurable budgets and is independent of any specific validator count, hardware configuration, or implementation choice.

Remark 11.4 (CP3-CP4 binding of the parallel scheme). The parallel decryption scheme preserves the CP3 timing binding structurally, per the same consensus-layer mechanism as the serial scheme. Each worker computes its share via the unmodified two-argument $\text{Dec_share}(\text{sk}_i, C_1) = \text{sk}_i \cdot C_1 \in G_1$ of Definition 10.5(a), where $C_1 \in G_1$ is the ciphertext's ephemeral element; the epoch-stable encryption domain label $\text{id}_{\text{enc}}(h, E)$ is a KDF input at encryption time and at final symmetric-key derivation, not a share-function argument. The scheduling tag $\text{sched}(h, \tau, v_i, r, R_{\text{committed}})$ and BLS-signed attestation a_i (Definition 10.5(b)) are produced once per transaction-validator pair after the worker consumes the block's $\text{QC}_{\text{prepare}}$, and are bundled with the share on broadcast. The state-machine gate of Definition 10.5(c) is enforced at the worker-dispatch boundary (the orchestrator does not release shares to any worker until $\text{QC}_{\text{prepare}}$ is observed), so no worker produces a share before the QC forms. The post-parallel synchronization step validates the global Merkle root against $R_{\text{committed}}$ for CP4 enforcement, aborting the block if any partition reports a mismatch. Ciphertext independence (Definition 11.1) is a property of the threshold hashed-ElGamal scheme (the decryption function is stateless in the ciphertext), not of the parallel orchestration, and CP1-CP4 hold independently of the degree of parallelism P .

12 Batch BLS Aggregation

BLS signature verification is used at multiple points in the pipeline: vote aggregation for quorum certificates, decryption-share verification, and block-proposal authentication. A single BLS verification requires two pairings on BLS12-381. For n signatures over the same or distinct messages, naive verification is $2n$ pairings. The batched multi-pairing check reduces this to a single multi-pairing computation.

Definition 12.1 (Batched BLS verification). Given signatures $\{\sigma_i\}_{i=1}^n$ on pairwise-distinct messages $\{m_i\}_{i=1}^n$ under public keys $\{\text{pk}_i\}_{i=1}^n$ (the multi-message setting of [8]; the equal-message setting is *not* covered by this definition and requires a distinct rogue-key-resistant multi-signature construction such as BDN’s proof-of-possession variant), the randomized-linearization batched check is:

$$\text{Verify}_{\text{batch}} : e\left(\sum_{i=1}^n \alpha_i \sigma_i, g\right) \stackrel{?}{=} \prod_{i=1}^n e(\alpha_i H(m_i), \text{pk}_i),$$

computed as a single multi-pairing, where each $\alpha_i \in \mathbb{F}_q$ is a uniformly random scalar drawn from a domain-separated transcript hash of the full (pk_i, m_i) tuple set. (The symbol α_i is chosen to avoid collision with the RANDAO commit-reveal symbol r_i of Definition 5.1, which denotes a validator’s 32-byte VRF-seed reveal, a distinct object.) Implementation uses the Boneh-Drijvers-Neven randomized-linearization batch construction [8] (“Compact Multi-Signatures for Smaller Blockchains,” ASIACRYPT 2018). This is distinct from the Bellare-Neven multi-signature construction (2006), which addresses a different setting (multi-signatures on a common message) and is not used here.

Theorem 12.2 (Batch verification correctness and soundness). *Let $\{(\sigma_i, m_i, \text{pk}_i)\}_{i=1}^n$ be a sequence of BLS signature tuples.*

- (i) (Correctness) *If every individual verification succeeds, that is $e(\sigma_i, g) = e(H(m_i), \text{pk}_i)$ for all i , then $\text{Verify}_{\text{batch}}$ succeeds.*
- (ii) (Soundness under randomization) *Under the randomized-linearization scheme with per-tuple random scalars α_i , the probability that $\text{Verify}_{\text{batch}}$ succeeds when at least one individual tuple $(\sigma_j, m_j, \text{pk}_j)$ is invalid is at most $1/q$, where q is the order of the scalar field (for BLS12-381, $q \approx 2^{255}$).*

Proof. We assume the co-CDH problem is hard in the BLS12-381 pairing groups (G_1, G_2, G_T, e) , which suffices for BLS signature unforgeability and under which the BDN randomized batch verification is sound [8].

Part (i): multi-linearity of the pairing gives $\prod_i e(\sigma_i, g)^{\alpha_i} = e(\sum \alpha_i \sigma_i, g)$ and $\prod_i e(H(m_i), \text{pk}_i)^{\alpha_i} = \prod_i e(\alpha_i H(m_i), \text{pk}_i)$. Under individual validity, both products equal $\prod_i e(H(m_i), \text{pk}_i)^{\alpha_i}$, so equality holds.

Part (ii): if one tuple $(\sigma_j, m_j, \text{pk}_j)$ is invalid, define $\delta_j = e(\sigma_j, g) \cdot e(H(m_j), \text{pk}_j)^{-1} \neq 1_{G_T}$ in the pairing target group. The batched product is $\Pi(\alpha_1, \dots, \alpha_n) = \left(\prod_{i \neq j} e(\sigma_i, g)^{\alpha_i} \cdot e(H(m_i), \text{pk}_i)^{-\alpha_i}\right) \cdot \delta_j^{\alpha_j}$. Viewed as a multilinear polynomial in the discrete logarithms of the α_i over \mathbb{F}_q , the Schwartz-Zippel lemma bounds the probability (over uniform $\alpha_j \in \mathbb{F}_q$, with the other α_i fixed) that $\Pi = 1_{G_T}$ by $\deg(\Pi)/q = 1/q$. Equivalently, $\Pi = 1_{G_T}$ constrains α_j to a single value in \mathbb{F}_q . Since α_j is uniform in \mathbb{F}_q of order $q \approx 2^{255}$ for BLS12-381, the collision probability is $1/q \approx 2^{-255}$, negligible in the security parameter. A co-CDH reduction extracts a forgery against the invalid tuple from any adversary that finds such an α_j non-negligibly often. \square

Corollary 12.3 (Verification complexity). *For a quorum certificate with n signer tuples, randomized batch verification reduces the verifier’s algebraic work from $O(n)$ independent pairing checks to one randomized multi-pairing relation plus linear-time preprocessing of the public inputs.*

Remark 12.4 (Usage sites). The batched verification of Definition 12.1 is the canonical Boneh-Drijvers-Neven randomized-linearization construction [8] over BLS12-381, specialized to its two call sites in the protocol specification: (i) vote aggregation for the formation of each quorum certificate in Section 3.1, and (ii) decryption-share verification at the COMMIT share-release transition. Both call sites satisfy Definition 12.1’s pairwise-distinct-message precondition: each per-validator signed message includes the signing validator’s identity string, so $m_i = (\varphi, B, d, h, r, v_i) \neq (\varphi, B, d, h, r, v_j) = m_j$ whenever $v_i \neq v_j$, even when two validators vote in the same phase φ on the same block hash B at the same (h, r) . The aggregate BLS signature σ_{agg} of Definition 3.12 aggregates these per-validator-distinct messages. Rogue-key resistance is further provided by per-validator proof-of-possession registered at committee induction (BDN’s PoP variant), so adversarial key-registration attacks are ruled out at validator on-boarding. The equal-message same-key-setup attack surface is therefore not present at either site.

13 Event-Atomic Settlement

The Event-Collect phase (§3.6) produces a quorum-backed certificate over event resolutions. Event-conditional trades must settle atomically with those resolutions: either both commit or neither does, with no intermediate state exposed to any external observer.

Definition 13.1 (Event-atomic block). A block \mathcal{B} is *event-atomic* if it contains: (i) an Event-Collect certificate QC_{EC} attesting a set of event resolutions $\{e_k \mapsto \text{outcome}_k\}_{k=1}^K$; (ii) an execution of transactions $\{T_j\}$ whose event-conditional predicates reference only events in QC_{EC} ; (iii) a post-execution state commitment that includes both the event outcomes and the transaction effects, committed as a single hash in the block header.

The substantive claim specific to event-atomic settlement, beyond the Composition Theorem, is *observer opacity under adversarial scheduling*: an adversary controlling message delivery between the QC_{EC} quorum and the DECIDE QC cannot expose a partial application of event outcomes to an external read-view client. We state this as the core theorem; the internal safety and frontrun-freeness claims follow from Theorem 16.2 and are recorded below as explicit corollaries.

Definition 13.2 (Read-view). A *read-view* is a function $\text{ReadView} : \mathbb{N} \rightarrow \Sigma \cup \{\perp\}$ defined for every block height $h \in \mathbb{N}$ by

$$\text{ReadView}(h) = \begin{cases} \sigma_{\text{post}}(\mathcal{B}) & \text{if } \text{QC}_{\text{Decide}}(\mathcal{B}, h) \text{ is committed and } \mathcal{B}.\text{height} = h, \\ \perp & \text{otherwise,} \end{cases}$$

where $\sigma_{\text{post}}(\mathcal{B})$ is the post-execution state root bound into the DECIDE QC of \mathcal{B} . The read-view exposes no other state.

Theorem 13.3 (Observer opacity under adversarial scheduling). *Let \mathcal{B} be a finalized event-atomic block (Definition 13.1) containing QC_{EC} attesting outcomes $\{e_k \mapsto \text{outcome}_k\}_{k=1}^K$. Against $\mathcal{A}_{\text{MBFT}}$ (§16.4), which includes adversarial post-GST network scheduling, every query to the read-view of Definition 13.2 at height $h = \mathcal{B}.\text{height}$ returns exactly one of:*

- (i) the pre-block state root (namely the post-execution state root of $\mathcal{B}.\text{parent}$, exposed by $\text{ReadView}(h - 1)$), or
- (ii) the post-block state root at \mathcal{B} once $\text{QC}_{\text{Decide}}(\mathcal{B}, h)$ is committed.

No query returns an intermediate state root in which QC_{EC} 's event resolutions apply but the dependent transaction set $\{T_j\}$ does not, or vice versa.

Proof. By Definition 13.2 the read-view exposes only states attested by a DECIDE QC; at any time before $\text{QC}_{\text{Decide}}(\mathcal{B}, h)$ forms, $\text{ReadView}(h) = \perp$ while $\text{ReadView}(h - 1)$ returns the parent's post-execution state. Intermediate executor states between QC_{EC} formation and the DECIDE transition are buffered and flushed atomically on DECIDE (Definition 3.1) and are never exposed. If the adversary schedules messages so as to delay $\text{QC}_{\text{Decide}}$, the read-view continues to return the parent's root; once $\text{QC}_{\text{Decide}}$ forms (by Theorem 7.1 under partial synchrony post-GST), the read-view returns \mathcal{B} 's post-execution root. By Agreement (Theorem 6.3) at most one DECIDE QC exists per height, so the read-view is well-defined. \square

Remark 13.4 (Scope of observer opacity). Theorem 13.3 covers the read-view of Definition 13.2 only. Side-channel signals at the operational interface of a deployment (mempool depth, pending-ciphertext counts, per-validator share-release timing on a gossip overlay, diagnostic endpoints) lie outside the theorem's scope; opacity against such signals is a deployment concern. A deployment that exposes intra-block mempool-depth counters to external observers remains within the theorem as stated while enabling out-of-band event-outcome inference at a rate separate from the residual cross-block bound of Proposition 9.8. Read-view opacity against *validator-observers* with gossip-layer access is an open problem.

Corollary 13.5 (Safety of event-atomic commits: no split). *No honest validator commits an intermediate state in which QC_{EC} 's event resolutions apply but the dependent transaction set $\{T_j\}$ does not (or vice versa).*

Proof. Direct consequence of Theorem 16.2 conclusions (i) and (iv): the state root committed into \mathcal{B} .header is computed over the combined result of QC_{EC} -attested outcomes and post-execution transaction effects as a single hash. By Agreement (Theorem 6.3), no two honest validators disagree on \mathcal{B} .header; and the state machine of Section 3.3 admits a single path through the event-then-execute sequence in any execution reaching the DECIDE QC. Hence no honest validator commits a partial-application state. \square

Corollary 13.6 (Event-conditional front-run freeness). *For any transaction T_j in \mathcal{B} whose predicate depends on $\text{outcome}_k \in \text{QC}_{\text{EC}}$, no adversary controlling $< t$ validator key-shares can inject, reorder, or condition any subsequent transaction on outcome_k within \mathcal{B} except with advantage negligible in λ .*

Proof. Direct consequence of Theorem 16.2 conclusion (iii) (front-run-freeness) applied to the Event-Collect phase placement. Post- QC_{EC} transactions in \mathcal{B} are threshold-encrypted (Definition 9.1) under the epoch-stable encryption identity $\text{id}_{\text{enc}}(h, E)$ fixed at user-submission time (Definition 10.3), so the $< t$ -share adversary cannot learn their plaintexts before the COMMIT QC releases shares attested under the authoritative $R_{\text{committed}}$ fixed by $\text{QC}_{\text{Prepare}}$ (Definition 10.5(a)-(b)). Hence the adversary cannot condition a post- QC_{EC} injection on outcome_k : either the injection is pre- $\text{QC}_{\text{Prepare}}$ (in which case outcome_k is not yet observable to the adversary because QC_{EC} has not formed) or it is post- $\text{QC}_{\text{Prepare}}$ (in which case $R_{\text{committed}}$ is already binding by CP1). No reordering window exists. \square

Corollary 13.7 (No cross-block event arbitrage). *A trader cannot observe event outcome outcome_k on chain before the block containing it is finalized, and cannot submit a transaction that profits from this observation within the same block. The expected profit of an event-arbitrage strategy that straddles the Event-Collect phase is bounded by the residual cross-block rate of Proposition 9.8.*

Proof. Event outcomes are revealed only when QC_{EC} is assembled, which occurs *inside* the Event-Collect phase of the block producing it. Post-EC transactions in that block are threshold-encrypted (Definition 9.1). By Theorem 10.9, no adversary with fewer than t shares can inject a transaction conditional on outcome $_k$ into the block after observing it. Cross-block arbitrage remains possible but is bounded by the residual MEV model. \square

Remark 13.8 (Event-atomic sequencing). The event-atomic pipeline sequences the phases of Section 3.1 as follows: the EVENT-COLLECT phase produces QC_{EC} and binds its outcome set into the block header; the PREPARE through COMMIT phases establish CP1-CP3; and the DECIDE phase resolves event-conditional predicates using the QC_{EC} -attested outcomes and commits a single post-block state root only after all conditional transactions have either applied or explicitly failed. The state root binds event outcomes and transaction effects together, so the finalized block header commits both atomically under the agreement guarantee of Theorem 6.3.

Remark 13.9 (Cross-chapter correspondence with PW-AMM PC1-PC4). The sibling paper [21] proves an event-atomic settlement theorem over the object of trading strategies adapted to in-block event outcomes, under proposer constraints PC1-PC4 and an honest-attestor-majority assumption. The theorem proved here (Theorem 13.3) proves observer opacity over the object of read-view state roots under the adversary \mathcal{A}_{MBFT} of §16.4. Both are instances of a single notion: computational indistinguishability from a perfect-information oracle on the event outcome, parameterized by an adversary class and an observable interface. We record the correspondence here to fix notation across papers; full proof obligations are open (below).

Open Problem 13.10 (CP1-CP4 vs PC1-PC4 correspondence). Prove or refute the forward direction $CP_i \Rightarrow PC_i$ for $i = 1, \dots, 4$, where PC1-PC4 are the proposer constraints of [21]. Prove or refute the reverse-direction separation: PC1-PC4 strictly weaker than CP1-CP4, witnessed by a protocol that satisfies PC1-PC4 but violates CP1 (the expected witness is an unordered-multiset commitment with post-hoc VRF ordering). The forward direction is conjectured to follow from the validator-role collapse of §16.4 (consensus voters, decryption-share holders, and attestors coincide), which would let the CP invariants realize the PC invariants at the intersection adversary class; but the proof is not in the present paper nor in the companion coupling paper, and this remark records the correspondence only as notational alignment.

14 Security Budget

Definition 14.1 (Security budget). The consensus security budget is

$$B_{\text{sec}} = (f + 1) \cdot s_{\text{min}} \cdot r_{\text{slash}},$$

where f is the maximum number of Byzantine validators, s_{min} is the minimum validator stake, and $r_{\text{slash}} \in (0, 1]$ is the maximum cumulative per-validator slash rate.

Definition 14.2 (Open interest cap). The open interest cap is derived from the security budget:

$$OI_{\text{max}} = \frac{B_{\text{sec}}}{\gamma}$$

where $\gamma \geq 2$ is the safety margin.

Example 14.3 (Parametric form). Under $n = 3f + 1$ and safety margin $\gamma \geq 2$, the budget and cap evaluate to

$$B_{\text{sec}} = (f + 1) \cdot s_{\text{min}} \cdot r_{\text{slash}}, \quad (7)$$

$$OI_{\text{max}} = \frac{(f + 1) \cdot s_{\text{min}} \cdot r_{\text{slash}}}{\gamma}. \quad (8)$$

Both quantities scale linearly in s_{min} and $(f+1)$; no numerical calibration is claimed by this paper.

Theorem 14.4 (Structural consistency of the security budget and OI cap). *The security budget B_{sec} and the open-interest cap OI_{max} satisfy the structural inequality $B_{\text{sec}} > OI_{\text{max}}$ whenever:*

1. $s_{\text{min}} > 0$ and $r_{\text{slash}} > 0$ (minimum-stake and slashing-rate conditions);
2. $f + 1 \geq 1$ (at least one Byzantine validator is tolerated);
3. the open-interest cap is enforced by a per-block admission check.

Scope of the claim. *This theorem establishes only the definitional inequality between the security-budget figure B_{sec} and the open-interest cap OI_{max} . It does not establish economic sustainability in the sense of “attacker expected value $\leq B_{\text{sec}}$ for all rational attackers and all attack strategies,” which would require a quantified model of attacker payoff across the MEV, consensus-safety, and oracle-manipulation surfaces. Such a quantified sustainability bound is beyond the scope of this paper; the structural consistency proved here is a necessary and insufficient condition for any stronger sustainability claim.*

Proof. Under Definitions 14.1 and 14.2, $B_{\text{sec}} = (f+1) \cdot s_{\text{min}} \cdot r_{\text{slash}} > 0$ under conditions (1)-(2), and $OI_{\text{max}} = B_{\text{sec}}/\gamma$ with $\gamma \geq 2$ gives $OI_{\text{max}} \leq B_{\text{sec}}/2 < B_{\text{sec}}$. The inequality follows directly from $\gamma \geq 2 > 1$ in Definition 14.2; it is structural to the definitions. Condition (3) enforces the bound at the protocol layer: every block is rejected whose cumulative open interest across markets would exceed OI_{max} . Hence $B_{\text{sec}} > OI_{\text{max}}$ holds at every committed state. \square

A deployment may further enforce per-block economic-safety admission checks that compare the security budget B_{sec} against the maximum single-block attack expected value derived from open interest and payout skew. The specific predicates and rejection rules are deployment policy and are outside the scope of this paper; the structural-consistency inequality $B_{\text{sec}} > OI_{\text{max}}$ proved above is necessary for any such admission discipline.

The slashing schedule calibrates penalties across seven offense classes (Table 2):

Offense	Detection
Double signing / attributable invalid-phase emission (includes CP3 and CP4 violations)	Automatic
Extended downtime (> 24h)	Automatic
Persistent downtime (> 7d)	Automatic
Oracle manipulation	Governance 2/3
Event censorship	Automatic
VRF reveal withholding	Automatic
Key share compromise	Governance 5-0

Table 2: Illustrative slashing schedule. Absolute rates are deployment-configuration choices; the consensus proofs depend only on the cumulative cap $r_{\text{slash}} \in (0, 1]$ being bounded below one, so that some residual stake remains after any sequence of offenses. The “Double signing / attributable invalid-phase emission” row carries the rate $r_{\text{cp3}} = r_{\text{cp4}}$ referenced by CP3 (Definition 10.5(d)) and CP4 (Definition 10.7): releasing a decryption share under a tag not derived from a valid $\text{QC}_{\text{Prepare}}$, signing a DECIDE QC over an $R_{\text{executed}} \neq R_{\text{committed}}$ mismatch, or endorsing a proposal whose payload violates the phase-validity predicate of Definition 3.5 are all attributable violations with cryptographic evidence. Each is rate-equivalent to equivocation because the validator has authenticated a state transition that the protocol rules make provably invalid.

14.1 Slashing dilution floor

The BFT safety arguments of Sections 3-3.3 assume a *fixed* validator set of size $n = 3f + 1$ with Byzantine bound $|\mathcal{B}| \leq f$. Slashing, however, shrinks the active set at runtime: each offense of Table 2 removes one validator from the committee. If repeated slashing shrinks the set below the honest-mass threshold of the quorum-intersection argument (Proposition 2.6), the BFT bound $|\mathcal{H}| \geq 2f + 1$ silently fails and the safety proof no longer applies.

The *slashing-dilution invariant* makes this failure mode explicit: rather than allowing consensus to degrade silently as the active set shrinks, we require the active set at every block height to carry a supermajority of the *genesis* stake, measured against the immutable genesis snapshot, not the live stake; or else consensus halts with an explicit error.

Definition 14.5 (Genesis stake snapshot). The *genesis stake snapshot* is the tuple $(V_0, s_0, S_0, n_{\min})$ where

- $V_0 \subseteq \mathcal{V}$ is the set of validators present at chain genesis;
- $s_0 : V_0 \rightarrow \mathbb{R}_{>0}$ is the per-validator stake weight at genesis;
- $S_0 = \sum_{v \in V_0} s_0(v)$ is the total genesis stake;
- $n_{\min} = |V_0| = 3f + 1$ is the hard floor on the active set size.

The snapshot is recorded at block height 0 and never mutated.

Definition 14.6 (Active genesis stake). Let $V(h) \subseteq \mathcal{V}$ denote the active validator set at block height h . The *active genesis stake* at height h is

$$S(h) = \sum_{v \in V(h) \cap V_0} s_0(v).$$

Validators joining the set after genesis do *not* contribute to $S(h)$: the denominator for the dilution floor is locked at genesis.

Proposition 14.7 (Slashing dilution floor). *At every block height $h \geq 0$, one of the following holds:*

1. (Invariant holds.) $|V(h)| \geq n_{\min}$ and $S(h) \geq \lceil \frac{2}{3}S_0 \rceil$; or
2. (Halt.) Consensus halts at height h with explicit error `DilutionFloorBreached` and produces no further blocks until governance intervention restores condition (1).

Silent continuation, producing blocks with $S(h) < \lceil \frac{2}{3}S_0 \rceil$ or $|V(h)| < n_{\min}$, is forbidden.

Proof sketch. The invariant is enforced at two points in the transition function (Section 3.3): (i) inside the slashing transition, immediately after a validator is removed from $V(h)$, and (ii) inside the generic invariant check that runs on every state transition, including view-changes and block commits. At each call site, the checker computes $S(h)$ and $|V(h)|$ from the current state and compares against $\lceil \frac{2}{3}S_0 \rceil$ and n_{\min} . If either floor is breached, the transition emits a terminal `DILUTIONFLOORBREACHED` safety violation; no subsequent consensus message is accepted. Since every state mutation passes through either the slashing transition or the invariant check, and both terminate on breach, no committed block at height $h + 1$ can exist unless the invariant held at height h . \square

Remark 14.8 (Relation to the BFT safety bound). The floor $S(h) \geq \lceil \frac{2}{3}S_0 \rceil$ is the stake-weighted analogue of the count-based constraint $|V(h)| \geq 2f + 1$ used in Proposition 2.6. Under equal weights $s_0 \equiv 1$, the two coincide: $S_0 = n = 3f + 1$ and $\lceil \frac{2}{3}S_0 \rceil = 2f + 1$. When stakes are non-uniform, the stake-weighted floor is the correct condition: the quorum-intersection argument rests on honest-stake-mass, not raw validator count, once stake weights enter the leader-selection and QC-weighting mechanisms.

Corollary 14.9 (Bounded tolerance). *Under equal weights and $n = 3f + 1$, the dilution-floor invariant tolerates at most f slashes before halting: after k slashes, $S(h) = (3f + 1) - k$, and $\lceil \frac{2}{3}(3f + 1) \rceil = 2f + 1$, so the invariant holds for $k \leq f$ and fails at $k = f + 1$. The floor coincides exactly with the BFT f -bound: a single additional slash beyond the adversary budget forces an explicit halt rather than silent degradation.*

Remark 14.10 (Economic-security consequence for slashing). Proposition 14.7 establishes an adversary disincentive beyond the direct slash penalty. For an adversary to extract value from a coordinated cascade of $\geq f + 1$ slashes, the adversary must first absorb $f + 1$ direct slash penalties (Table 2, at least $0.33 \sum s_0(v_i)$ cumulatively under the maximum-offense schedule), and the result of the cascade is a halt, not a fork or a double-spend: no value accrues to the adversary from the halted state. The expected value of the cascade is therefore strictly negative under any positive slash rate, so a rational adversary does not execute it.

15 TLA⁺ Formal Grounding

MoxieBFT is accompanied by two TLA⁺ models in the sense of Lamport [13]. The first is a safety automaton for the six-phase state machine of Sections 3.1-3.3, tracking phase, view, height, lockedQC, highQC, proposal buffers, vote accumulators, and view-change accumulators. The second is a focused pacemaker-liveness model for the WAN setting, whose state abstracts the timeout, quorum-formation, and synchronization rules used to eliminate the view-change wedge.

The safety model checks the invariants that carry the paper’s BFT argument: agreement, locked-QC compatibility, vote boundedness, and view/height monotonicity. The pacemaker model encodes the WAN-pacemaker discipline of Remark 3.16: validators time out into a new target view, advance only on a quorum for that current target view

or on the quorum-backed NEWVIEW proof derived from it, and import stronger DECIDE certificates rather than continuing to vote on stale rounds. Together these models provide executable specifications for the exact state predicates used in Sections 6 and 7.

These TLA⁺ models do not replace the proofs of the paper’s cryptographic or economic claims. They abstract BLS verification, threshold ciphertexts, ML-KEM wrappers, the full semantics of the Phase 1/2/3 executor, and slashing economics. Consequently, CP2 secrecy, CP3 share-release binding, CP4 execution-commitment consistency, and the slashing-dilution floor rest on the mathematical arguments of Sections 10 and 14.1, not on model checking alone.

The formal point is therefore modest but load-bearing: agreement, monotonicity, lock adoption, and current-view pacemaker synchronization are all stated in a machine-checkable language, and the WAN-liveness pacemaker discipline is grounded in a dedicated temporal model rather than prose alone.

16 Related Work

MoxieBFT sits inside an active line of research on partially-synchronous Byzantine consensus, data-availability-based DAG ordering, and MEV-resistant mempool design. We position our contribution against the specific prior works in each of these three strands.

16.1 Classical and pipelined BFT consensus

PBFT (Castro & Liskov, 1999). PBFT [1] introduced practical Byzantine fault tolerance with $O(n^2)$ message complexity and view-change recovery. MoxieBFT inherits PBFT’s $f < n/3$ resilience bound and its quorum-intersection (Proposition 2.6) argument, and reduces the per-round message count via BLS aggregation (§12) and committee selection (§4). What MoxieBFT adds over PBFT is (i) a prepended oracle-attestation phase producing QC_{EC} committed into the block header, and (ii) a threshold-encrypted mempool with CP1-CP4 commit-reveal-execute discipline.

Tendermint (Buchman, 2016). Tendermint [2] achieves partial-synchronous BFT in two vote rounds without pipelining. MoxieBFT’s four voting rounds plus Event-Collect plus decryption-share release gate add rounds beyond Tendermint, but (a) the added rounds are provably unavoidable for the composite class Π_{CEP} (Theorem 10.17), and (b) the HotStuff-style lock admits a cleaner liveness argument than Tendermint’s locked-proof-of-value.

HotStuff (Yin et al., 2019) and HotStuff-2 (Abraham et al., 2023). HotStuff [3] introduced the three-phase chained BFT pipeline with linear per-phase message complexity and view-change pipelining. HotStuff-2 [7] collapses the three phases to two in the optimistic-responsive regime (view change completes in a single additional round when the leader is honest and GST has passed). MoxieBFT instantiates HotStuff-2 as its BFT kernel: the Prepare-PreCommit-Commit-Decide pipeline is HotStuff-2’s, as is the locked-QC mechanism underlying Lemma 6.1. Our extension is structural (adding QC_{EC} and the CP1-CP4-gated share-release path), not a modification of the HotStuff-2 voting protocol itself.

Jolteon and Ditto (Gelashvili et al., 2021). Jolteon [22] is a HotStuff variant that restores the two-phase responsive-commit property via a linear view-change sub-protocol; Ditto [23] couples Jolteon with asynchronous fallback for graceful degradation past GST.

MoxieBFT’s optimistic commit latency is comparable to Jolteon’s (both recover responsiveness when the leader is honest), but MoxieBFT’s composed latency is determined by CP3’s decryption-share-release ordering, which is a cost neither Jolteon nor Ditto bear.

16.2 DAG-based mempool and consensus

Narwhal-Tusk (Danezis et al., 2022). Narwhal-Tusk [4] separates data dissemination (the Narwhal DAG mempool) from consensus ordering (the Tusk consensus overlay). The separation amortizes the cost of block transmission across many blocks and achieves orders-of-magnitude higher throughput than linear HotStuff-style pipelines. MoxieBFT does not inherit Narwhal: our target regime is single-block financial-settlement latency rather than mempool throughput, and threshold-decryption on a Narwhal DAG raises concurrency questions that the linear pipeline avoids. The Narwhal certificate structure is, however, a plausible replacement for §12’s vote aggregation should a higher-throughput MoxieBFT variant be needed; the CP1-CP4 discipline composes with Narwhal DAG ordering modulo a restatement of Round 4’s share-release gate against DAG certificate references.

Bullshark (Spiegelman et al., 2022) and Narwhal-Bullshark (Kokoris-Kogias et al., 2024). Bullshark [24] is a commit overlay on Narwhal that achieves partially-synchronous finality without a separate Tusk consensus step; the combined stack (Narwhal for mempool, Bullshark for commit) has become the canonical high-throughput Byzantine-consensus baseline. MoxieBFT and Narwhal-Bullshark solve different problems: Bullshark optimizes mempool-limited throughput at the expense of per-block latency (typically $\geq 2s$ end-to-end on WAN configurations), while MoxieBFT optimizes single-block latency under the composite CP1-CP4 constraint. The $5 \cdot \ell_{\text{med}}$ lower bound of Theorem 3.8 applies to Narwhal-Bullshark once it is augmented with a threshold-encrypted mempool and event-atomic settlement.

DiemBFT v4 (Diem Association, 2021). DiemBFT v4 is a HotStuff derivative with two-chain commits, timestamp-based pacemaker, and an explicit epoch mechanism. The pacemaker design (voting timeouts gated on per-round deadlines) is compatible with MoxieBFT’s deployment-configurable t_{block} of Definition 3.4; the epoch-change mechanism of DiemBFT v4 is a precedent for our E_{blocks} -height epoch scheme (Definition 5.1). MoxieBFT diverges from DiemBFT v4 in (a) the prepended Event-Collect phase and (b) the CP3-gated share-release path for threshold-decryption.

Mir-BFT (Stathakopoulou et al., 2022). Mir-BFT [25] generalizes multi-leader PBFT into a parallel-pipeline form with per-leader sequence segments, achieving horizontal throughput scaling without abandoning linear-ordering guarantees. MoxieBFT is single-leader per view (with VRF rotation across views) and does not inherit Mir-BFT’s horizontal-partitioning structure; a multi-leader MoxieBFT variant would require extending the CP3 scheduling tag with per-leader segment indices and is not pursued here.

Hashgraph (Baird, 2016). Hashgraph [27] is a leaderless gossip-based consensus protocol whose asynchronous BFT property relies on virtual voting over a gossip-DAG. Its latency is data-dependent (mean-case bounded in the number of gossip hops), and it does not expose the discrete message-round abstraction against which Theorem 10.17 is stated. The five-round bound does not directly apply to Hashgraph; a gossip-hop-count analog would require restating CP1-CP4 against Hashgraph’s virtual-voting semantics, which we do not undertake.

16.3 MEV-resistant mempool architectures

Flashbots SUAVE. Flashbots SUAVE [6] addresses MEV through a proposer-builder separation architecture with trusted-execution-environment builders. MoxieBFT achieves MEV resistance without trusted parties: the encrypted mempool of Definition 9.1 ensures no single validator (or builder) can observe transaction contents before ordering is committed at QC_{Prepare} . MoxieBFT’s trust assumption is the threshold-decryption corruption cap $t - 1$; SUAVE’s is the TEE-attestation chain for the builder runtime.

F3B (Zhang et al., 2022). F3B [9] introduces the commit-before-decrypt pattern at the consensus layer, using Boneh-Franklin IBE as its threshold primitive. MoxieBFT inherits the consensus-layer pattern of F3B and explicitly does *not* inherit F3B’s IBE instantiation: we specify the simpler threshold hashed-ElGamal over G_1 (no pairings on the encryption/decryption path) and discharge the $R_{\text{committed}}$ -binding property *structurally* through CP3(b)-(d) rather than cryptographically through a modified share primitive. This is both a primitive-level simplification (one group operation per share rather than a pairing-based derivation) and a stronger honest-framing of the timing binding (cryptographic impossibility for $< t$ -share adversaries, plus slashable evidence for $[f + 1, t - 1]$ -controlled adversaries).

Shutter Network and timed-commitment mempools. Shutter [28] is a concurrent commit-before-decrypt mempool design built on an independent Shutter-validator committee distinct from the base-layer consensus. MoxieBFT unifies these two committees (the Byzantine-consensus validator set and the decryption-share-holder set are identical), which tightens the intersection adversary model (§16.4) by precluding cross-committee adversary arbitrage: the corruption cap f binds simultaneously on both roles.

Position of MoxieBFT. Relative to the above works, the contribution of the present paper is the composition: HotStuff-2 as BFT kernel, Event-Collect as oracle-atomic extension, threshold hashed-ElGamal as commit-before-decrypt mempool, under the CP1-CP4 invariant set bound together by Theorem 16.2. The individual components are prior art. The novel elements are (i) the composed adversary $\mathcal{A}_{\text{MBFT}}$ of §16.4 under which the components co-exist without mutual weakening, (ii) the five-round message-complexity lower bound for the composite class (Theorem 10.17), and (iii) the observer-opacity theorem under adversarial scheduling (Theorem 13.3), which has no direct counterpart in the works cited above.

16.4 Composition Theorem

The claim that MoxieBFT’s end-to-end guarantees follow from the guarantees of its three constituent components is what makes the construction a genuine systems contribution rather than a re-labelling. We state the composition theorem formally.

Composed adversary model. The three components Π_{HS2} , Π_{F3B} , and Π_{EC} are analyzed in their native literatures against different adversary types. To compose them, we fix a *single* adversary $\mathcal{A}_{\text{MBFT}}$ against MoxieBFT and state, explicitly, that the composition inherits the *intersection* of the component adversary capabilities: the binding constraint is the tightest bound among the three.

- **Corruption cap (binding).** $\mathcal{A}_{\text{MBFT}}$ statically controls at most $f = \lfloor (n - 1)/3 \rfloor$ validators at epoch start, shared across the consensus role, the decryption-share role, and the attester role. Since a MoxieBFT validator holds all three roles simultaneously, the HotStuff-2 bound f is strictly tighter than the threshold-decryption

secrecy bound $t - 1 = \lceil 2n/3 \rceil - 1$. Threshold-hashed-ElGamal secrecy therefore holds *a fortiori* against $\mathcal{A}_{\text{MBFT}}$ since $f < t$ for all $n \geq 4$. The role-collapse is an implementation requirement: validator operators must hold the consensus-signing key, the threshold-decryption share-key, and the BLS attestation-signing key on equivalent trust boundaries. Outsourcing any single role (e.g., delegating decryption-share custody to a third-party HSM service distinct from the consensus operator) weakens the composed adversary back to component-wise bounds by reintroducing cross-role adversary arbitrage: a Byzantine consensus operator paired with an honest third-party share custodian (or vice versa) expands the effective corruption surface beyond f . MoxieBFT’s canonical deployment assumes role-unified validator operation.

- **Corruption timing.** We adopt the *static-corruption* model as the composed bound: $\mathcal{A}_{\text{MBFT}}$ commits to its corruption set at epoch start. This matches the canonical HotStuff-2 and threshold-decryption secrecy statements. VRF-weighted leader rotation exposes a target-the-next-leader operational risk, but no adaptive threshold-encryption theorem is claimed here. Extending CP2 to adaptive corruptions would require additional assumptions such as no new share-key exposure between proposal and share release, key erasure or proactive refresh, and an HSM compromise-delay lower bound.
- **Network scheduling.** Partial synchrony with adaptive message scheduling after GST, matching the canonical HotStuff-2 statement and strictly stronger than the classical threshold-decryption network model (which is typically synchronous during share release).

Under this intersection model, each component’s canonical guarantee applies without weakening: (a) inherits verbatim; (b) holds against the weaker f -bounded adversary (trivially implied by its $(t - 1)$ -bounded statement); (c) inherits as stated.

Proposition 16.1 (CP1-CP4 preservation under $\mathcal{A}_{\text{MBFT}}$). *Each of CP1-CP4 (Definitions 10.1-10.7) is preserved against $\mathcal{A}_{\text{MBFT}}$, the single adversary of §16.4 with static corruption cap $f < t$ and post-GST partial synchrony.*

Proof. CP1 (*binding commitment*). CP1 reduces to SHA-256 collision resistance, which is independent of the adversary’s corruption capabilities or timing. Any \mathcal{A} (including $\mathcal{A}_{\text{MBFT}}$) that produces a collision against $R_{\text{committed}}$ does so only with probability negligible in the security parameter.

CP2 (*hiding commitment until reveal*). CP2 is the threshold hashed-ElGamal IND-CPA property of Π_{F3B} (Definition 9.1; reducing to CDH-with-KDF-in-ROM or DDH in G_1). The composed corruption cap $|\mathcal{A}_{\text{MBFT}}| \leq f$ is strictly below the threshold $t - 1 = \lceil 2n/3 \rceil - 1$ for all $n \geq 4$ (since $f \leq (n - 1)/3 < 2n/3$); hence $\mathcal{A}_{\text{MBFT}}$ controls $< t$ decryption-share keys and CP2’s IND-CPA statement holds *a fortiori* under the static-corruption model [17]. No adaptive-corruption transfer is used.

CP3 (*timing-bound commit-before-decrypt*). CP3 combines (a) the cryptographic epoch-stable domain label $\text{id}_{\text{enc}}(h, E)$ (Definition 10.3), under the IND-CPA assumption used already for CP2, with (b)-(d) structural consensus-layer gates enforced by state-machine transitions, honest-recipient filtering of mis-attested shares, and BLS-signed attestations that generate slashable evidence on out-of-gate release (Definition 10.5(b)-(d)). BLS unforgeability holds against any adversary controlling fewer than $f + 1$ honest signers, which $\mathcal{A}_{\text{MBFT}}$ does not: $|\mathcal{A}_{\text{MBFT}}| \leq f$ means at least $2f + 1 - f = f + 1$ honest signers are required to forge a $2f + 1$ -quorum signature, which $\mathcal{A}_{\text{MBFT}}$ cannot do under the static-corruption assumption. Combined with the $f < t$ corruption cap, this is exactly the honest guarantee summarized in “What CP3 guarantees” within Definition 10.5: the $\mathcal{A}_{\text{MBFT}}$ attacker faces a cryptographic impossibility at the encryption primitive layer (case (i)) or a consensus-layer impossibility at the attestation layer (cases (ii)-(iii)).

CP4 (execution-commitment consistency). CP4 is a local check performed by each honest validator at DECIDE: it compares R_{executed} (computed from the revealed plaintexts) to $R_{\text{committed}}$ (committed in the block header at $\text{QC}_{\text{prepare}}$). Honest validators hold correct plaintexts by threshold-decryption correctness, recompute R_{executed} deterministically, and refuse to issue a DECIDE vote on mismatch. The property does not depend on network scheduling: even under adversarial scheduling after GST, no honest validator votes DECIDE on a block with $R_{\text{executed}} \neq R_{\text{committed}}$, so no DECIDE QC can form (by Theorem 6.3 applied to the $2f + 1$ honest majority) on such a block.

In all four cases, the adversary model under which the CP_i is proved is the same $\mathcal{A}_{\text{MBFT}}$ used in the composition statement. \square

Theorem 16.2 (MoxieBFT Composition). *Let Π_{HS2} denote the HotStuff-2 BFT protocol [7], Π_{F3B} denote the threshold-encrypted mempool construction of Definition 9.1 (threshold hashed-ElGamal over BLS12-381 G_1 ; the class name references the consensus-layer pattern of F3B [9], from which the commit-before-decrypt discipline is inherited, not the Boneh-Franklin IBE cryptosystem of that work), and Π_{EC} denote the Event-Collect oracle-attestation phase (§3.6). Assume:*

- (a) Π_{HS2} satisfies agreement and view-synchronous liveness under partial synchrony with $n \geq 3f + 1$ honest-majority validators [7];
- (b) Π_{F3B} satisfies IND-CPA ciphertext secrecy until a threshold $t = q(n) = \lceil 2n/3 \rceil$ of decryption shares is assembled, under the CDH-with-KDF-in-ROM assumption (equivalently the DDH assumption without ROM) in G_1 of BLS12-381 [17, 18];
- (c) Π_{EC} satisfies attestor-quorum liveness: under the tier-escalation schedule, Tier ≤ 3 eventually produces a resolution QC QC_{EC} after at most $c\Delta$ post-GST rounds;
- (d) the composed protocol satisfies CP1-CP4 (Definitions 10.1-10.7).

Then MoxieBFT ($= \Pi_{\text{HS2}} \circ \Pi_{\text{EC}} \circ \Pi_{\text{F3B}}$ under the CP1-CP4 binding) satisfies, against the composed adversary $\mathcal{A}_{\text{MBFT}}$ of §16.4: (i) Agreement (Theorem 6.3); (ii) Liveness with the bound of Theorem 7.1; (iii) Frontrun-freeness (Theorem 10.9); (iv) Event-atomic safety (Corollary 13.5) and observer opacity under adversarial scheduling (Theorem 13.3).

Proof. Under the composed adversary model $\mathcal{A}_{\text{MBFT}}$ above, each of CP1-CP4 is preserved by Proposition 16.1, and each component’s native guarantee inherits as noted. We verify the four conclusions in turn.

Safety (i). Agreement for MoxieBFT is Theorem 6.3, proved via Lemma 6.1 on the (PREPARE, PRECOMMIT, COMMIT, DECIDE) vote graph. We must show that Π_{EC} and Π_{F3B} do not alter the premises of Lemma 6.1. Π_{EC} inserts a message round *before* PREPARE: it does not emit PREPARE / PRECOMMIT / COMMIT / DECIDE votes and does not modify the locked-QC update rule or the quorum-intersection property (Proposition 2.6); its output QC_{EC} is a committed field of the block, participating in safety only through the block hash. Π_{F3B} alters payload *contents* (ciphertext \tilde{c} in place of plaintext), while payload *ordering* and *vote authority* remain fixed: the PREPARE / PRECOMMIT / COMMIT / DECIDE vote signatures sign block hashes that include $R_{\text{committed}}$ (Definition 10.1), and the vote state machine is unchanged. Hence Lemma 6.1 applies verbatim, and Theorem 6.3 composes to the composite protocol: for all honest v_i, v_j and committed records $(B_1, h), (B_2, h), B_1 = B_2$.

Liveness (ii). The additive decomposition is the one proved in Theorem 7.1. View synchronization contributes Δ_{sync} ; each failed Byzantine-leader view contributes at most T_{to} and the expected count is $p_B / (1 - p_B)$; by (c), Π_{EC} produces QC_{EC} within $c\Delta$ post-GST rounds. Once QC_{EC} is available, the proposer invokes Π_{HS2} ’s PREPARE phase with the block containing QC_{EC} and $R_{\text{committed}}$. By (a), Π_{HS2} commits within four voting rounds plus one DECIDE-propagation round, totalling 5Δ . Π_{F3B} ’s share-release is concurrent with COMMIT (Definition 10.5(c)), adding no net latency.

Frontrun-freeness (iii). By (b) Π_{F3B} satisfies IND-CPA ciphertext secrecy until t decryption shares are assembled. By Theorem 10.9, this plus CP1-CP4 yields frontrun-freeness (no adversary with $< t$ shares can insert, reorder, or condition a transaction on another's plaintext within the same block, except with negligible advantage in λ).

Atomicity (iv) splits into two substatements: (a) internal event-atomic safety (the no-split property of Corollary 13.5), which follows from (i) (Agreement) applied to the block header's single state-root commitment; and (b) observer opacity under adversarial scheduling, which is Theorem 13.3 and is the substantive content novel to the event-atomic setting (the read-view never exposes an intermediate state to external observers, even under adversarially delayed DECIDE QCs).

Atomicity (iv). Theorem 13.3 proves joint event-transaction atomicity from (i), (c)'s committed QC_{EC} , and the execution layer's single post-block state-root commitment. The observer-opacity clause uses that the state-root read-view exposes only states attested by a DECIDE QC, which by (i) is unique per height.

Combining the four items, MoxieBFT satisfies Agreement, Liveness, Frontrun-freeness, and Event-atomic settlement as a single system guarantee against \mathcal{A}_{MBFT} . \square

Proposition 16.3 (CP1-CP4 are jointly necessary). *Under any strict relaxation of CP1-CP4, at least one of the four conclusions (i)-(iv) of Theorem 16.2 fails.*

Proof. • Weakening CP1 (ordering-Merkle commitment) permits post-broadcast substitution: the adversary may replace the committed sequence after observing votes, violating the ordering premise of (iii).

- Weakening CP2 (hiding until t shares) permits pre-reveal plaintext leakage via fewer than t shares, directly violating (iii).
- Weakening CP3 from the structural attestation binding of Definition 10.5(a)-(d) to a post-hoc-slashing-only variant (removing the per-share attestation a_i of (b) and the state-machine gate of (c), retaining only the slashing rule of (d)) permits race-condition plaintext leakage: a validator releases its share before $QC_{Prepare}$ is observed and the slashing penalty is paid after the fact, not preventing the leakage. In the present formulation the attestation a_i of Definition 10.5(b) combined with the state-machine gate of Definition 10.5(c) prevent specification-conforming validators from releasing shares early and generate slashable signed evidence against any validator that does.
- Weakening CP4 (execution-commitment consistency) permits post-reveal reordering: the executed order may diverge from $R_{committed}$ without a Merkle-root check, violating (iii).

Each relaxation produces a concrete attack; CP1-CP4 are jointly necessary. \square

17 Open Problems

The following are the principal open problems left by the present work.

- OP1. Exhaustive TLA⁺ verification.** Complete exploration at $N = 4, F = 1$ and extension to larger N (§15).
- OP2. Tier-escalation liveness for Π_{EC} .** A formal proof with an explicit bound on c (Open Problem 7.3).
- OP3. Derivation and calibration of the residual-MEV constant.** A rigorous stochastic derivation of the cross-block bound of Proposition 9.8 and empirical calibration of its constant (Open Problem 9.9).

- OP4. CP1-CP4 vs PC1-PC4 correspondence.** Prove or refute $CP_i \Rightarrow PC_i$ for $i = 1, \dots, 4$ (Open Problem 13.10).
- OP5. Boneh-Franklin IBE tight-binding variant.** Strengthen CP3’s structural binding to a cryptographic tag-binding via Boneh-Franklin IBE, achieving IND-ID-CPA at the primitive layer.
- OP6. Game-based cryptographic reduction of the Composition Theorem.** The present argument is a structural composition proof at the system-property level; a tight reduction in the cryptographic-security-games framework is open.
- OP7. Adaptive corruption beyond one view.** Safety under a fully adaptive adversary, lifting the bounded-window model of Remark 2.8.
- OP8. Read-view opacity against validator-observers.** Side-channel-aware observer opacity covering gossip-layer and share-release timing leakage to validator-observers.

18 Conclusion

We have presented MoxieBFT as a systems-composition result addressing the event-completeness and frontrunning problems under partial synchrony. The protocol augments the HotStuff-2 pipeline [7] with a prepended Event-Collect phase and a threshold-encrypted mempool adopting the consensus-layer pattern of F3B [9] under threshold hashed-ElGamal. The invariants CP1-CP4 bind the components, while the block-validity predicate makes the execution order Phase 1 \rightarrow Phase 2 \rightarrow Phase 3 a consensus rule rather than a post-consensus convention. Theorem 16.2 discharges agreement, liveness, frontrun-freeness, and event-atomic safety under the composed adversary $\mathcal{A}_{\text{MBFT}}$.

The safety properties (agreement, validity, view monotonicity, height monotonicity, locked-QC safety, vote boundedness) are proved formally. Liveness is proved under partial synchrony with quorum $q(n) = \lceil 2n/3 \rceil$, current-target-view pacemaker synchronization, stale-certificate rebroadcast, and a timeout envelope large enough to cover the WAN message and decryption budget. The accompanying TLA⁺ safety and pacemaker models provide formal grounding for these invariants without substituting for the cryptographic arguments.

Frontrun-freeness against the composed adversary $\mathcal{A}_{\text{MBFT}}$ is established by Theorem 10.9. Cross-block residual MEV is not eliminated; Proposition 9.8 gives a regime-specific lower bound $\Omega(\sigma_V \sqrt{\Delta_{\text{rev}}})$, not an upper risk bound. The threshold-encryption bribe cost scales as $C_{\text{decrypt}} \geq t \cdot s_{\text{min}}$ with $t = q(n) = \lceil 2n/3 \rceil$ (Theorem 9.6) and binds separately from the consensus-safety bribe cost $C_{\text{safety}} \geq (f+1) \cdot s_{\text{min}}$; both scale linearly in t or f respectively and in s_{min} .

The protocol is specified as a deterministic, pure-function state machine that checks every safety invariant on every transition. Block-interval admissibility follows the closed-form bound of Definition 3.4; absolute wall-clock block times, validator counts, and concrete cryptographic parameterizations are deployment-configuration choices outside the scope of this paper.

References

- [1] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173-186, 1999.
- [2] E. Buchman. Tendermint: Byzantine fault tolerance in the age of blockchains. Master’s thesis, University of Guelph, 2016.

- [3] M. Yin, D. Malkhi, M. K. Reiter, G. Golan Gueta, and I. Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 347-356, 2019.
- [4] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman. Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems (EuroSys)*, pages 34-50, 2022.
- [5] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374-382, 1985.
- [6] Flashbots. SUAVE: Single unifying auction for value expression. Technical report, Flashbots Research, 2023.
- [7] I. Abraham, D. Malkhi, K. Nayak, and L. Ren. HotStuff-2: Optimal two-phase responsive BFT. Cryptology ePrint Archive, Report 2023/397, 2023. (Originally distributed 2022.)
- [8] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *Advances in Cryptology - ASIACRYPT 2018*, pages 435-464, Springer, 2018.
- [9] H. Zhang, L.-H. Merino, V. Estrada-Galiñanes, and B. Ford. F3B: A low-overhead blockchain architecture with per-transaction front-running protection. Cryptology ePrint Archive, Report 2022/1523, 2022. Conference version in *Proceedings of the 5th Conference on Advances in Financial Technologies (AFT 2023)*, LIPIcs 282, pages 3:1-3:23, 2023. (Bibkey year refers to the original preprint.)
- [10] S. Goldberg, L. Reyzin, D. Papadopoulos, and J. Vrček. Verifiable Random Functions (VRFs). RFC 9381, Internet Engineering Task Force, 2023.
- [11] A. Faz-Hernández, S. Scott, N. Sullivan, R. S. Wahby, and C. A. Wood. Hashing to Elliptic Curves. RFC 9380, Internet Engineering Task Force, 2023.
- [12] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288-323, 1988.
- [13] L. Lamport. *Specifying Systems: The TLA⁺ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.
- [14] I. Keidar and S. Rajsbaum. On the cost of fault-tolerant consensus when there are no faults: a tutorial. *SIGACT News*, 32(2):45-63, 2001.
- [15] A. Momose and L. Ren. Optimal communication complexity of authenticated Byzantine agreement. In *35th International Symposium on Distributed Computing (DISC)*, LIPIcs 209, pages 32:1-32:16, 2021.
- [16] J. Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339, 2021.
- [17] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology - EUROCRYPT 1998*, Lecture Notes in Computer Science vol. 1403, pages 1-16, Springer, 1998. Standard treatment of threshold ElGamal, including IND-CPA and IND-CCA variants; the mempool primitive specified here is the IND-CPA threshold hashed-ElGamal construction of this line of work. (Distinct from Boneh-Franklin identity-based encryption.)

- [18] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology - EUROCRYPT 1991*, Lecture Notes in Computer Science vol. 547, pages 522-526, Springer, 1991. Original Pedersen distributed key generation protocol used (with Gennaro-Rabin-style secure variants [19]) to produce the (t, n) -threshold ElGamal key of Definition 9.1.
- [19] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology - EUROCRYPT 1999*, Lecture Notes in Computer Science vol. 1592, pages 295-310, Springer, 1999. Hardens Pedersen DKG against rushing adversaries; the standard reference for secure threshold DKG over cyclic groups.
- [20] J. O'Connor, J.-P. Aumasson, S. Neves, and Z. Wilcox-O'Hearn. BLAKE3: one function, fast everywhere. Specification, 2020. Keyed BLAKE3 is one of several domain-separated, collision-resistant hashes that may instantiate H_{dom} and the KDF in the hashed-ElGamal DEM of Definition 9.1; the security reduction models it as a random oracle.
- [21] R. Lorgat. Prediction-weighted automated market making: event-atomic settlement and spot coupling. Technical paper, April 2025.
- [22] R. Gelashvili, L. Kokoris-Kogias, A. Sonnino, A. Spiegelman, and Z. Xiang. Jolteon and Ditto: Network-adaptive efficient consensus with asynchronous fallback. In *Financial Cryptography and Data Security (FC)*, 2022.
- [23] A. Spiegelman, B. Aurnia, and L. Alistarh. Ditto: A responsive commit-optimistic BFT protocol under partial synchrony with asynchronous fallback. Extended version of [22], 2022.
- [24] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias. Bullshark: DAG BFT protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2705-2718, 2022.
- [25] C. Stathakopoulou, M. Pavlovic, and M. Vukolić. State machine replication scalability made simple (Mir-BFT). In *Proceedings of the Seventeenth European Conference on Computer Systems (EuroSys)*, pages 17-33, 2022.
- [26] The Diem Team. DiemBFT v4: State machine replication in the Diem blockchain. Diem Association technical report, 2021.
- [27] L. Baird. The Swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance. Swirlds Tech Report SWIRLDS-TR-2016-01, 2016.
- [28] Shutter Network Team. Shutter Network: Threshold-encrypted mempool for Ethereum rollups. Technical report, 2022.