The Design and Formal Analysis of the Merkle-Chained Event Ledger

John G. Underhill

Quantum Resistant Cryptographic Solutions Corporation

Abstract. Merkle-Chained Event Ledger (MCEL) is an append-only cryptographic ledger construction designed to provide strong integrity, ordering, and auditability guarantees for event-based systems without relying on consensus mechanisms or global state replication. MCEL organizes events as a sequence of typed records, each cryptographically bound to prior ledger state through hash commitments and Merkle aggregation, enabling efficient verification of inclusion, consistency, and historical correctness. The construction supports explicit anchoring and checkpoint mechanisms, allowing ledger state to be externally committed and authenticated while preserving local append semantics and deterministic verification.

This paper presents a formal definition of the MCEL protocol and a cryptanalytic examination of its security properties under standard hash function and signature assumptions. We analyze the ledger's resistance to record modification, deletion, reordering, and equivocation attacks, and examine the security implications of its anchoring and checkpoint model. The analysis clarifies the threat model, identifies explicit non-goals, and establishes the conditions under which MCEL provides verifiable append-only behavior suitable for audit logs, evidence ledgers, and compliance systems.

1 Introduction

Event logs and audit trails play a central role in security-critical systems, including compliance infrastructures, evidence retention systems, and distributed operational records. In these settings, it is often necessary to provide strong guarantees that recorded events are complete, ordered, and resistant to undetectable modification after the fact. Traditional database logging mechanisms offer limited cryptographic assurance, while blockchain-based systems introduce significant complexity, operational overhead, and trust assumptions that are unnecessary for many audit and accountability use cases.

A recurring requirement in such systems is an append-only ledger that provides verifiable integrity without requiring consensus, proof-of-work, or global state replication. In many deployments, a single logical authority, or a small number of cooperating authorities, is responsible for appending records, while a larger set of external parties require the ability to verify historical correctness. This asymmetry motivates constructions that emphasize deterministic verification and tamper evidence over decentralized agreement.

Merkle-Chained Event Ledger (MCEL) is a cryptographic ledger construction designed to address this class of problems. MCEL represents ledger state as a sequence of typed records, each cryptographically bound to prior state through hash commitments and Merkle aggregation. Rather than batching records into blocks, the ledger evolves record by record, allowing fine-grained auditability while preserving efficient verification. Checkpoints and anchor records provide explicit mechanisms for externally committing ledger state, enabling third-party verification and temporal binding without altering the append-only semantics of the ledger itself.

The design of MCEL deliberately separates cryptographic guarantees from operational policy. The protocol defines how records are committed, chained, and verified, but does not prescribe how authorities are selected, how keys are managed, or how anchoring events are scheduled. This separation allows MCEL to be deployed in a wide range of environments, from local compliance logs to multi-institution audit systems, while maintaining a precise and analyzable security model.

This paper presents a formal definition of the MCEL protocol and a detailed analysis of its correctness and security properties. We specify the ledger state, record structures, and evolution rules, and analyze the construction under a clearly defined threat model. The analysis focuses on resistance to record modification, deletion, reordering, and equivocation, as well as the security implications of anchoring and checkpoint mechanisms. By explicitly identifying assumptions and non-goals, the paper aims to clarify the security guarantees that MCEL provides, and the conditions under which those guarantees hold.

2 Preliminaries and Notation

This section defines the cryptographic primitives, assumptions, and notation used throughout the paper. Unless otherwise stated, all security claims are made under standard computational assumptions.

2.1 Cryptographic Primitives

MCEL relies on cryptographic hash functions and digital signature schemes. Let H denote a cryptographic hash function with fixed output length λ, modeled as collision resistant and second-preimage resistant. Where domain separation is required, distinct labeled invocations of H are assumed to be independent.

Let $\mathsf{Sign}(sk, m)$ and $\mathsf{Verify}(pk, m, \sigma)$ denote a digital signature scheme that is existentially unforgeable under chosen-message attacks (EUF-CMA). Public keys are assumed to be distributed through an authenticated channel external to the protocol.

Merkle tree constructions are used to aggregate multiple hash commitments into a single root value. Given an ordered list of leaf values $(x_1, \ldots, x_n)$, the function $\mathsf{MerkleRoot}(x_1, \ldots, x_n)$ denotes the deterministic Merkle root computed using H as the compression function.

2.2 Ledger Model

An MCEL ledger is modeled as a finite, ordered sequence of records

$$\mathcal{L} = (R_1, R_2, \ldots, R_n),$$

where each record R_i is appended exactly once and never modified thereafter. The index i denotes the logical position of a record within the ledger.
Each record consists of a header and an optional body. Headers contain metadata required for chaining, domain separation, and verification, while bodies contain application-specific data that is opaque to the protocol analysis except where explicitly noted.

2.3 Notation

The following notation is used throughout the paper.

- $\|$ denotes byte-string concatenation.
- $\mathsf{len}(x)$ denotes the length of a byte string x.
- $\mathsf{Commit}(R_i)$ denotes the cryptographic commitment of record R_i, computed as defined in Section 5.
- Root_i denotes the Merkle root or cumulative commitment representing ledger state after record R_i has been appended.
- $\perp$ denotes failure or an invalid value.

All algorithms are deterministic unless explicitly stated otherwise. Randomized algorithms, when used, are assumed to draw randomness from a source that is uniform and unpredictable to the adversary.

2.4 Security Model

Adversaries are modeled as probabilistic polynomial-time algorithms with full read access to the ledger and the ability to observe, delay, or replay records and anchors. The adversary may attempt to modify historical records, suppress records, or present inconsistent ledger views to different verifiers. Key compromise and denial-of-service attacks are considered outside the scope of this analysis unless explicitly stated.
Throughout the paper, negligible functions are defined with respect to the security parameter λ in the standard sense.

3 System and Threat Model

This section defines the system participants, trust assumptions, and adversarial capabilities considered in the analysis of MCEL. The security properties established in later sections hold only within the scope of this model.

3.1 System Participants

The MCEL system consists of the following entities.
Ledger Authority. The ledger authority is responsible for constructing and appending records to the ledger according to the MCEL protocol. The authority maintains the canonical ledger state and controls the private keys used to sign anchor and checkpoint records.
Verifiers. Verifiers are external parties that obtain ledger data and validate its correctness. A verifier may be an auditor, regulator, or automated process that requires assurance that a given sequence of records is complete and untampered.
Anchoring Infrastructure. Anchors bind ledger state to an external reference, such as a timestamping service or external ledger. The anchoring infrastructure is treated as an external system and is not assumed to be trusted beyond the authenticity of anchor outputs.

3.2 Trust Assumptions

MCEL assumes that the ledger authority follows the protocol when generating records and computing commitments, but may be subject to later scrutiny or dispute. The protocol is designed to provide evidence of misbehavior, rather than to prevent all forms of misbehavior at record creation time.
Cryptographic primitives are assumed to satisfy their standard security properties as defined in Section 2. Public verification keys are assumed to be authentic and correctly associated with the ledger authority.
No assumptions are made about network reliability, record delivery order, or record availability. Verifiers may obtain partial or delayed views of the ledger.

3.3 Adversary Capabilities

The adversary is modeled as a probabilistic polynomial-time algorithm with the following capabilities.

- Full read access to all records, checkpoints, and anchors.
- The ability to delay, replay, or suppress the delivery of records to verifiers.
- The ability to present different subsets of the ledger to different verifiers.

The adversary may control the ledger authority after some point in time, including possession of signing keys, but is not assumed to have retroactive control over previously generated anchors or external commitments.

3.4 Out-of-Scope Threats

The following threats are explicitly outside the scope of this analysis.

- Denial-of-service attacks against the ledger authority or verifiers.
- Side-channel attacks against cryptographic implementations.
- Compromise of cryptographic primitives or secure key storage.
- Policy-level failures, including improper key management or unauthorized record creation.

The exclusion of these threats does not imply that they are unimportant, but reflects the focus of this paper on cryptographic correctness and integrity guarantees rather than operational security.

4 MCEL Construction

This section provides an overview of the Merkle-Chained Event Ledger construction. The goal is to describe the structural components and their interaction at a conceptual level, deferring precise algorithms and state transition rules to the formal definition in the following section.

4.1 Ledger Structure

An MCEL ledger is an append-only sequence of records, ordered by insertion time and indexed sequentially. Each record represents a discrete event and is cryptographically bound to prior ledger state. Once appended, records are immutable and cannot be altered or removed without detection by a verifier.

Unlike block-based ledgers, MCEL does not group records into blocks. Instead, each record advances the ledger state directly, enabling fine-grained verification and precise reasoning about record order and inclusion. Ledger state evolves monotonically as records are appended.

4.2 Record Types

MCEL supports multiple record types, each serving a distinct semantic role within the ledger. At a minimum, the construction distinguishes between ordinary event records, checkpoint records, and anchor records.

Event records capture application-specific data and form the primary contents of the ledger. Checkpoint records summarize accumulated ledger state at specific points in time and provide explicit verification boundaries. Anchor records bind checkpoint commitments to external systems, providing temporal or contextual grounding beyond the ledger itself.

All record types share a common header structure to ensure uniform commitment and chaining behavior, while allowing record bodies to vary according to their function.

4.3 Chaining and Commitments

Each record is associated with a cryptographic commitment derived from its header, its body, and a reference to prior ledger state. These commitments form a hash chain that enforces ordering and immutability.

To efficiently aggregate multiple record commitments, MCEL employs Merkle tree constructions. At defined intervals, typically aligned with checkpoint records, the commitments of recent records are combined into a Merkle root that compactly represents ledger state up to that point. This root value becomes part of the checkpoint commitment and serves as the basis for subsequent verification.

4.4 Domains and Separation

MCEL supports logical separation of records through explicit domain identifiers. Domains allow multiple independent logical ledgers to coexist within a shared construction while preventing cross-domain replay or substitution attacks. Domain identifiers are incorporated into record commitments, ensuring that records are cryptographically bound to their intended context.

This separation enables MCEL to support multi-tenant or multi-purpose deployments without requiring separate ledger instances at the cryptographic level.

4.5 Anchors and External Commitments

Anchors provide a mechanism for binding ledger state to external references. An anchor record contains a commitment to a checkpoint value along with evidence of its publication or registration in an external system. The security of anchoring relies on the immutability or timestamping guarantees of the external system, rather than on trust in the ledger authority.
Anchors do not alter the internal append-only semantics of the ledger. Instead, they serve as externally verifiable attestations that a given ledger state existed at or before a particular point in time.

4.6 Verification Model

Verification in MCEL is deterministic and does not require interaction with the ledger authority. A verifier validates record integrity by recomputing commitments, checking Merkle inclusion proofs, and verifying signatures on checkpoint and anchor records. Given sufficient ledger data and associated proofs, a verifier can independently determine whether a presented ledger view is consistent and complete relative to a known checkpoint or anchor.
This verification model supports both online and offline auditing scenarios and allows third parties to assess ledger correctness without relying on trusted intermediaries.

5 Formal Definition of MCEL

This section presents a formal definition of the Merkle-Chained Event Ledger protocol. We define the ledger state, record structure, commitment functions, and ledger evolution rules. All definitions are deterministic unless explicitly stated otherwise.

5.1 Ledger State

The state of an MCEL ledger after i records is denoted by

$$\mathsf{State}_i = (\mathcal{L}_i, C_i),$$

where $\mathcal{L}_i = (R_1, \ldots, R_i)$ is the ordered sequence of records, and C_i is the cumulative commitment representing the ledger state after record R_i.
The initial ledger state State_0 is defined as an empty ledger with a fixed initial commitment C_0, which may be a constant or a domain-specific initialization value.

5.2 Record Structure

Each record R_i consists of a header H_i and an optional body B_i.

$$R_i = (H_i, B_i)$$

The header contains the following fields.

- A record type identifier.
- A domain identifier dom.
- A reference to prior ledger state, typically C_{i-1}.
- Metadata required for verification, including flags and length fields.

The body B_i contains application-specific data and is treated as an opaque byte string by the protocol, except where its inclusion is required for commitment computation.

5.3 Record Commitment

The commitment of a record R_i is computed as

$$\mathsf{Commit}(R_i) = H\big(\mathsf{dom} \| H_i \| B_i \| C_{i-1}\big),$$

where H is the cryptographic hash function defined in Section 2, and $\|$ denotes byte-string concatenation.
The inclusion of the domain identifier and prior commitment ensures that each record is bound to both its logical context and the existing ledger state.

5.4 Merkle Aggregation and Checkpoints

Let $(\mathsf{Commit}(R_j), \ldots, \mathsf{Commit}(R_k))$ denote a contiguous sequence of record commitments. A Merkle root over this sequence is computed as

$$\mathsf{Root}_{j,k} = \mathsf{MerkleRoot}(\mathsf{Commit}(R_j), \ldots, \mathsf{Commit}(R_k)).$$

A checkpoint record R_c contains such a Merkle root summarizing a prefix of the ledger. The commitment of a checkpoint record is computed in the same manner as any other record, with the Merkle root included in its body.

5.5 Anchors

An anchor record binds a checkpoint commitment to an external reference. Formally, an anchor record R_a includes a checkpoint identifier, the corresponding checkpoint commitment, and evidence of external publication or registration.
Anchor records are signed by the ledger authority. Let $\sigma_a = \mathsf{Sign}(sk, m_a)$ denote the signature over the anchor payload m_a. Verification of an anchor record requires validating both the internal commitment structure and the external evidence according to the anchoring system's rules.

5.6 Ledger Evolution Rules

The ledger evolves by appending records according to the following rules.

1. A new record R_i may be appended only if its header references the current commitment C_{i-1}.
2. The commitment $\mathsf{Commit}(R_i)$ is computed deterministically.
3. The new ledger state is updated as $C_i = \mathsf{Commit}(R_i)$.

Any deviation from these rules results in a ledger state that will fail verification.

5.7 Verification

Given a purported ledger prefix $(R_1, \ldots, R_i)$, a verifier recomputes commitments sequentially starting from C_0. Verification succeeds if and only if all record commitments match and any checkpoint or anchor records validate according to their respective rules.
This deterministic verification process allows independent parties to assess ledger correctness without interacting with the ledger authority.

6 Security Analysis

This section proves the main security properties of MCEL under the threat model of Section 3. We model record and checkpoint commitments as deterministic functions over canonical encodings of record components as defined in Section 5. All probabilities are taken over the internal randomness of the adversary and the randomness of the underlying primitives (where applicable). We write (λ) for a negligible function in the security parameter λ.

6.1 Assumptions

MCEL relies on the following standard assumptions.

Collision resistance. A hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ is collision resistant if for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}(x, x') \leftarrow \mathcal{A}(1^\lambda) : \\ x \neq x' \wedge H(x) = H(x')\end{array}\right] \leq (\lambda).$$

EUF-CMA security. A signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is EUF-CMA secure if for every PPT adversary $\mathcal{A}$ with access to a signing oracle $\mathsf{Sign}(sk, \cdot)$,

$$\Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(pk) : \\ \mathsf{Verify}(pk, m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{Q}\end{array}\right] \leq (\lambda),$$

where $\mathcal{Q}$ is the set of messages queried to the signing oracle.

6.2 Canonical Encoding and Commitment Function

All security results require that record encodings are canonical, meaning the serialization map from structured records to byte strings is injective.

Definition 1 (Canonical encoding). Let $\mathsf{Enc}(R)$ denote the byte-string encoding of a record $R = (H, B)$. Encoding is *canonical* if for all records $R \neq R'$, we have $\mathsf{Enc}(R) \neq \mathsf{Enc}(R')$.

We restate the record commitment function using explicit encoding. For record $R_i = (H_i, B_i)$ with domain identifier dom_i and prior cumulative commitment C_{i-1}, define

$$\mathsf{Commit}(R_i, C_{i-1}) := H(\mathsf{dom}_i \| \mathsf{Enc}(H_i) \| \mathsf{Enc}(B_i) \| C_{i-1}) .$$

We write C_i for the cumulative commitment after appending R_i, namely $C_i := \mathsf{Commit}(R_i, C_{i-1})$.

6.3 Security Experiments

We formalize tampering resistance as an inability to produce two distinct valid explanations for the same ledger state.

Experiment $\mathsf{Exp}^{\mathsf{prefix}}_{\mathcal{A}}(\lambda)$. The adversary outputs two record sequences $\mathcal{L} = (R_1, \ldots, R_k)$ and $\mathcal{L}' = (R'_1, \ldots, R'_k)$ such that $\mathcal{L} \neq \mathcal{L}'$. The experiment computes cumulative commitments C_k and C'_k from a common C_0 using the MCEL rules and outputs 1 if $C_k = C'_k$ and both sequences satisfy the structural validity conditions (well-formed headers, correct reference to prior commitment, and canonical encoding).

Intuitively, winning this experiment means producing two different histories with the same resulting state commitment.

6.4 Prefix Uniqueness and Tamper Evidence

Lemma 1 (One-step binding). Assume canonical encoding. For any fixed domain dom and prior commitment C, if

$$\mathsf{Commit}(R, C) = \mathsf{Commit}(R', C),$$

then either $R = R'$ or $\mathcal{B}$ can be constructed that finds a collision in H with probability at least that of this event.

Proof. Let

$$X := \mathsf{dom} \| \mathsf{Enc}(H) \| \mathsf{Enc}(B) \| C, \quad X' := \mathsf{dom} \| \mathsf{Enc}(H') \| \mathsf{Enc}(B') \| C.$$

If $R \neq R'$, canonical encoding implies $X \neq X'$. Equality of commitments yields $H(X) = H(X')$ with $X \neq X'$, which is a collision. A reduction $\mathcal{B}$ outputs (X, X'). □

Lemma 2 (Prefix uniqueness). Assume H is collision resistant and encoding is canonical. For any $k \geq 1$,

$$\Pr\left[\mathsf{Exp}^{\mathsf{prefix}}_{\mathcal{A}}(\lambda) = 1\right] \leq (\lambda).$$

Proof. Suppose $\mathcal{A}$ wins with non-negligible probability, producing distinct $\mathcal{L} \neq \mathcal{L}'$ with $C_k = C'_k$. Let t be the smallest index where $R_t \neq R'_t$. Then $R_i = R'_i$ for all $i < t$, hence $C_{t-1} = C'_{t-1}$. By the definition of t we have

$$\mathsf{Commit}(R_t, C_{t-1}) = C_t = C'_t = \mathsf{Commit}(R'_t, C'_{t-1}),$$

with $R_t \neq R'_t$ and $C_{t-1} = C'_{t-1}$. By Lemma 1 this yields a collision in H with the same non-negligible probability, contradicting collision resistance. □

Corollary 1 (Record modification detectability). Assume H is collision resistant and encoding is canonical. Given a valid prefix $(R_1, \ldots, R_k)$, any change to any record R_t for $t \leq k$ produces a sequence that fails verification except with negligible probability.

Proof. Any change produces a distinct prefix $\mathcal{L}' \neq \mathcal{L}$. If verification were to succeed to the same state commitment, it would contradict Lemma 2. If verification succeeds to a different commitment, the mismatch is detected by the verifier. □

Corollary 2 (Deletion and reordering detectability). Under the same assumptions, removing a record from a valid prefix or reordering records causes verification failure except with negligible probability.

Proof. Deletion or reordering yields a distinct record sequence. If it verified to the same final commitment, Lemma 2 is violated. Otherwise, the verifier detects the commitment mismatch. □

6.5 Merkle Aggregation and Checkpoint Soundness

Let $(x_1, \ldots, x_n)$ be leaf values. We assume a deterministic Merkle root algorithm $\mathsf{MerkleRoot}$ built from H with explicit domain separation between leaf hashing and internal-node hashing, or an equivalent injective encoding of node types.

Lemma 3 (Merkle root binding). Assume H is collision resistant and the Merkle construction uses injective node encodings. Then it is infeasible for a PPT adversary to find two distinct leaf sequences $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_m)$ such that

$$\mathsf{MerkleRoot}(x_1, \ldots, x_n) = \mathsf{MerkleRoot}(x'_1, \ldots, x'_m),$$

except with negligible probability.

Proof. Standard Merkle tree binding arguments apply: equality of roots for different trees implies a collision in some internal node or a collision at the leaf-to-root compression path, yielding a collision in H. A reduction traverses the first point of divergence to extract such a collision. □

Theorem 1 (Checkpoint soundness). Assume H is collision resistant, encoding is canonical, and the Merkle construction satisfies Lemma 3. Then a checkpoint committing to $\mathsf{Root}_{j,k}$ uniquely binds the multiset and order of the covered record commitments $(\mathsf{Commit}(R_j), \ldots, \mathsf{Commit}(R_k))$ except with negligible probability.

Proof. If an adversary could produce a different covered sequence yielding the same checkpoint root, Lemma 3 would be violated. Since record commitments are themselves binding (Lemma 1), substituting different records without changing commitments is infeasible. □

6.6 Forking, Equivocation, and Consistency

Because MCEL is not a consensus protocol, fork prevention is not a goal. The relevant property is *detectable consistency* relative to a shared reference, such as a known checkpoint commitment or an external anchor.

Definition 2 (Fork). Two ledgers $\mathcal{L} = (R_1, \ldots, R_k)$ and $\mathcal{L}' = (R'_1, \ldots, R'_\ell)$ *fork* if there exists a maximal $t \geq 0$ such that $R_i = R'_i$ for all $i \leq t$, and either $t < \min(k, \ell)$ with $R_{t+1} \neq R'_{t+1}$, or $k \neq \ell$ and one ledger extends the other.

Theorem 2 (Prefix consistency at equal commitments). Assume collision resistance and canonical encoding. If two valid ledger prefixes yield the same cumulative commitment value C_t, then the prefixes are identical up to index t, except with negligible probability.

Proof. Directly Lemma 2. If two distinct prefixes produced the same C_t, $\mathcal{A}$ would win $\mathsf{Exp}^{\mathsf{prefix}}$. □

Theorem 3 (Checkpoint-relative fork detectability). Assume Theorem 1 and Theorem 2. Let a verifier possess a valid checkpoint commitment C_c and associated checkpoint record that is internally valid. Then any presented ledger history inconsistent with that checkpoint is rejected, except with negligible probability. Moreover, if a verifier observes two histories both claiming consistency with the same checkpoint commitment, then the histories must share an identical prefix up to the checkpoint boundary.

Proof. If a history claims consistency with checkpoint commitment C_c but yields a different cumulative commitment at the checkpoint boundary, verification fails deterministically. If two distinct histories both yield C_c at the boundary, Theorem 2 implies they are identical up to that point. □

6.7 Anchors and Signature Guarantees

Let an anchor payload be a message m_a that includes, at minimum, a checkpoint identifier and checkpoint commitment value, along with any required external anchoring metadata. The anchor record contains (m_a, σ_a) with $\sigma_a = \mathsf{Sign}(sk, m_a)$.

Theorem 4 (Anchor authenticity). Assuming the signature scheme is EUF-CMA secure, no PPT adversary can produce a valid anchor record (m_a, σ_a) for a new message m_a not previously signed by the ledger authority, except with negligible probability.

Proof. This is a direct EUF-CMA reduction: an adversary that forges such an anchor yields a signature forgery on a new message. □

Theorem 5 (Anchor consistency binding). Assume collision resistance, canonical encoding, and EUF-CMA security. Once an anchor record binding a checkpoint commitment C_c is published and accepted by a verifier, no adversary can later present an inconsistent ledger history as being anchored to that same anchor, except with negligible probability.

Proof. To claim consistency with the anchor, the adversary must present a checkpoint and ledger history whose checkpoint commitment equals C_c. By Theorem 3, any history consistent with C_c shares the same prefix up to the checkpoint boundary. Presenting a different prefix with the same C_c would violate Theorem 2. Producing a different anchor message for a different checkpoint commitment while reusing the same signature would either fail verification or imply a signature forgery. □

6.8 Domain Separation and Cross-Domain Attacks

Theorem 6 (Domain isolation). Assume collision resistance and canonical encoding. If the domain identifier dom is included in the commitment input, then a record committed in one domain cannot be replayed as a valid record in a different domain without detection, except with negligible probability.

Proof. Let R be a record with commitment computed under domain dom, and suppose an adversary attempts to present the same encoded record under $\mathsf{dom}' \neq \mathsf{dom}$ with the same prior commitment. The two commitment inputs differ in the domain prefix, hence equality of resulting commitments implies a collision in H. □

6.9 What MCEL Does Not Prove

The results above establish binding, tamper evidence, and checkpoint and anchor relative detectable consistency. They do not imply any of the following properties.

- *Global agreement.* Without consensus or gossip, verifiers may temporarily hold different views.
- *Liveness.* Records may be delayed or withheld by an adversary controlling the authority.
- *Prevention of equivocation.* The authority may fork histories; the protocol makes this detectable relative to checkpoints or anchors.

6.10 Summary

Under collision resistance of H, canonical encoding, and EUF-CMA security of signatures, MCEL provides: (i) unique prefix commitments, implying tamper-evident append-only behavior, (ii) sound checkpoints via Merkle binding, (iii) detectable consistency relative to shared checkpoint commitments, and (iv) strong authenticity and non-repudiation of anchors via signatures. These properties match the audit and evidence objectives of MCEL in the non-consensus setting.

7 Security Invariants

This section summarizes the core security invariants maintained by MCEL. These invariants hold under the assumptions and threat model defined earlier and provide a concise statement of the guarantees established by the protocol and its analysis.

7.1 Commitment Immutability Invariant

Invariant 1 (Commitment Immutability). Once a record R_i is appended and its commitment $\mathsf{Commit}(R_i)$ is incorporated into the cumulative ledger state, the contents of R_i cannot be altered without detection.

This invariant follows directly from the collision resistance of the hash function and the inclusion of prior cumulative commitment values in each record commitment. Any modification propagates forward and invalidates subsequent verification.

7.2 Prefix Consistency Invariant

Invariant 2 (Prefix Consistency). For any cumulative commitment value C_k, there exists at most one valid record sequence $(R_1, \ldots, R_k)$ that produces C_k.

This invariant captures the uniqueness of ledger prefixes and underlies fork detectability. It ensures that once a verifier observes a specific commitment value, all valid histories consistent with that value share the same prefix.

7.3 Append-Only Invariant

Invariant 3 (Append-Only Behavior). Ledger state evolves strictly by appending records. No valid ledger state can be obtained by removing or reordering previously committed records.
This invariant is enforced by the chaining of commitments and guarantees that historical records, once committed, remain permanently bound to ledger state.

7.4 Domain Separation Invariant

Invariant 4 (Domain Isolation). Records committed under one domain identifier cannot be interpreted as valid records under a different domain.
By incorporating domain identifiers into commitment computation, MCEL ensures cryptographic separation between logical ledgers, preventing replay or substitution across domains.

7.5 Checkpoint Consistency Invariant

Invariant 5 (Checkpoint Soundness). A valid checkpoint uniquely and correctly summarizes the commitments of the records it covers.
This invariant ensures that checkpoints serve as reliable verification boundaries and that inclusion proofs derived from checkpoints are sound.

7.6 Anchor Binding Invariant

Invariant 6 (Anchor Binding). Once a checkpoint commitment is bound to an external anchor, no alternative ledger history inconsistent with that commitment can be presented as valid relative to the anchor.
This invariant captures the role of anchoring as an external consistency reference and establishes the immutability of anchored ledger state.

7.7 Discussion

Taken together, these invariants characterize the security posture of MCEL. They emphasize detectability, determinism, and verifiable history over real-time enforcement or consensus. All correctness and security results presented in this paper can be viewed as consequences of these invariants under the stated assumptions.

8 Correctness Properties

This section establishes the correctness properties of MCEL under honest execution. Correctness refers to the protocol's ability to produce a ledger that is internally consistent, verifiable, and append-only when all participants follow the protocol rules defined in Section 5.

8.1 Ledger Consistency

Proposition 1 (Sequential Consistency). For any ledger prefix $(R_1, \ldots, R_i)$ generated according to the ledger evolution rules, the cumulative commitment C_i is uniquely determined.
Proof. The initial commitment C_0 is fixed. For each $i \geq 1$, the commitment C_i is computed deterministically as $\mathsf{Commit}(R_i)$, which depends only on the record contents and C_{i-1}. By induction, C_i is uniquely determined by the sequence $(R_1, \ldots, R_i)$. □

8.2 Append-Only Behavior

Proposition 2 (Append-Only Property). If a record R_i is included in a valid ledger prefix, then no alternative ledger prefix of the same length can exclude or modify R_i without detection by a verifier.

Proof. Each record commitment incorporates the prior cumulative commitment. Removing or modifying R_i changes $\mathsf{Commit}(R_i)$, which propagates to all subsequent commitments. A verifier recomputing commitments from C_0 will detect any such deviation, causing verification to fail. □

8.3 Checkpoint Correctness

Proposition 3 (Checkpoint Soundness). A valid checkpoint record correctly summarizes the commitments of the records it covers.

Proof. Checkpoint records include a Merkle root computed over an explicit sequence of record commitments. Given the determinism of the Merkle construction, any verifier can recompute the root from the referenced commitments and confirm equality. Any omission, reordering, or modification of records changes the computed root and is detected during verification. □

8.4 Deterministic Verification

Proposition 4 (Deterministic Verification). Given the same ledger data and verification keys, all honest verifiers reach the same verification result.

Proof. Verification consists solely of deterministic computations, including hash evaluations, Merkle root recomputation, and signature verification. No randomized steps or external state are involved. Therefore, verification outcomes are identical for all verifiers operating on the same input. □

8.5 Anchor Binding

Proposition 5 (Anchor Binding Correctness). A valid anchor record binds a specific checkpoint commitment to the referenced external evidence.

Proof. Anchor records include a signed payload that commits to the checkpoint identifier and its commitment. Assuming signature correctness, any verifier can confirm that the anchor payload was authorized by the ledger authority and that it references a unique checkpoint value. □

8.6 Discussion

The properties established above show that MCEL, when executed according to the protocol, produces a ledger that is internally consistent, append-only, and independently verifiable. These correctness guarantees do not rely on assumptions about adversarial behavior or cryptographic hardness beyond determinism and proper implementation. In the next section, we analyze how these properties hold in the presence of adversarial manipulation attempts.

9 Limitations and Non-Goals

This section clarifies the limitations of MCEL and explicitly identifies properties that the protocol does not attempt to provide. These limitations are intentional design choices and reflect the scope of the construction and analysis presented in this paper.

9.1 No Consensus or Liveness Guarantees

MCEL does not provide consensus, leader election, or liveness guarantees. The protocol assumes that records are appended by a designated authority and does not include mechanisms to ensure progress in the presence of authority failure or network disruption. As a result, MCEL is not suitable for environments requiring multi-writer agreement or adversarial fault tolerance at the protocol level.

9.2 Detectability Rather Than Prevention of Equivocation

MCEL does not prevent a malicious ledger authority from creating multiple divergent ledger histories. Instead, it provides cryptographic evidence that allows such equivocation to be detected once ledger views or anchors are compared. The protocol does not guarantee that all verifiers will observe the same ledger state at all times, particularly in the absence of timely anchoring.

9.3 Dependence on External Anchoring

The strength of temporal guarantees in MCEL depends on the properties of the external anchoring mechanism. Delayed or infrequent anchoring reduces the ability to bound when a given ledger state existed. MCEL does not mandate anchoring frequency or enforce anchor publication, as these are considered operational decisions rather than cryptographic properties.

9.4 No Policy or Identity Enforcement

MCEL defines cryptographic mechanisms for record commitment and verification but does not specify policies governing who may append records, how identities are managed, or how authorization decisions are made. These concerns are intentionally left to higher-level systems and deployment-specific policy frameworks.

9.5 Operational and Implementation Risks

The analysis in this paper assumes correct implementation of cryptographic primitives, canonical serialization, and secure key management. Side-channel attacks, implementation bugs, and key compromise are outside the scope of the model. While MCEL can provide evidence of historical inconsistency, it cannot retroactively correct errors or misconfigurations introduced during operation.

10 Implementation Considerations

This section discusses practical considerations for implementing MCEL. The material is non-normative and does not alter the formal protocol definition or security claims established in earlier sections.

10.1 Reference Implementation Mapping

A reference implementation of MCEL follows directly from the formal definitions given in Section 5. Record headers and bodies map to serialized byte structures, and commitment computation is implemented as a deterministic hash over the serialized components. Care must be taken to ensure that serialization is canonical and unambiguous, as any divergence between implementations may lead to verification failures.

The reference implementation separates ledger logic, Merkle aggregation, anchoring, and domain handling into distinct components. This separation reflects the conceptual decomposition of the protocol and simplifies both verification and auditing.

10.2 Hash Function Selection

The security analysis assumes a cryptographic hash function with strong collision and second-preimage resistance. Implementations should use a modern hash function with a well-defined security margin. Domain separation may be implemented either by explicit domain identifiers in record headers or by labeled hash invocations, provided that the separation is unambiguous and consistent.
All hash inputs should include explicit length encoding or structured serialization to prevent ambiguity in concatenation.

10.3 Merkle Tree Construction

Merkle trees should be constructed deterministically, with a fixed leaf ordering and well-defined handling of odd numbers of leaves. Any optimization, such as incremental tree construction or caching of intermediate nodes, must preserve functional equivalence with the abstract Merkle root definition.
Inclusion proofs generated by the implementation must be sufficient for independent verification without requiring access to the full ledger.

10.4 Key Management and Signatures

Private keys used for signing checkpoint and anchor records represent a critical trust component. Implementations should support key rotation and explicit identification of signing keys within record headers to allow verifiers to validate historical records across key changes.
Signature formats and verification rules must be fixed and documented to avoid cross-implementation inconsistencies.

10.5 Anchoring Systems

Anchoring mechanisms depend on external systems and may vary by deployment. Implementations should treat anchor evidence as opaque data, verified according to the rules of the chosen anchoring system. The MCEL protocol does not require anchors to be frequent, but timely anchoring improves detectability of equivocation.
Care should be taken to record sufficient anchor metadata to allow independent verification long after anchor creation.

10.6 Performance Considerations

Verification cost in MCEL grows linearly with the number of records unless checkpoints are used. Implementations should provide efficient access to checkpoint data and inclusion proofs to support scalable verification. Storage requirements are modest, as records are append-only and commitments are compact.
These considerations do not affect the cryptographic guarantees of MCEL but influence its suitability for large-scale or long-lived deployments.

11 Related Work

Cryptographically verifiable logs and append-only data structures have been studied extensively in several contexts, including transparency systems, secure logging, and distributed ledgers. Merkle tree constructions are a common foundation in these systems, providing compact commitments to ordered data sets and efficient inclusion proofs.

Certificate transparency logs and similar append-only log systems employ Merkle trees to provide public auditability and detect equivocation by log operators. These systems typically assume a globally visible log and rely on gossip or monitoring mechanisms to expose inconsistencies. MCEL shares the use of Merkle aggregation but differs in its record-by-record chaining model and explicit support for checkpointing and external anchoring without requiring a globally synchronized log.

Blockchain-based ledgers also provide append-only semantics and tamper resistance, often through consensus protocols and economic incentives. While effective in adversarial multi-writer environments, such systems impose significant complexity and performance costs. MCEL targets a different design space, where append authority is limited and auditability, determinism, and low operational overhead are primary goals.

Secure logging and audit frameworks have explored hash chaining and periodic signing to detect tampering. MCEL extends these ideas by combining per-record chaining with Merkle aggregation, domain separation, and externally verifiable anchoring, enabling flexible deployment across a range of audit and compliance scenarios.

12 Conclusion

This paper presented a formal definition and cryptanalytic analysis of the Merkle-Chained Event Ledger protocol. MCEL is designed to provide verifiable append-only behavior, strong integrity guarantees, and deterministic verification for event-based ledgers without relying on consensus mechanisms or global state replication.

Through a precise specification and analysis under a well-defined threat model, we showed that MCEL resists record modification, deletion, reordering, and undetectable equivocation under standard cryptographic assumptions. The inclusion of checkpoints and external anchoring enables efficient verification and temporal binding, while preserving the core append-only semantics of the ledger.

MCEL does not attempt to solve all problems associated with distributed trust or authority compromise. Instead, it provides a clear and analyzable foundation for audit and evidence systems in environments where accountability and verifiability are required. Future work includes exploring deployment-specific anchoring strategies, automated equivocation detection mechanisms, and extensions to support additional audit and compliance use cases.

References

1. R. C. Merkle, *Protocols for Public Key Cryptosystems*, IEEE Symposium on Security and Privacy, 1980.
https://ieeexplore.ieee.org/document/6233529
2. S. Haber and W. S. Stornetta, *How to Time-Stamp a Digital Document*, Journal of Cryptology, 3(2), 1991.
https://link.springer.com/article/10.1007/BF00196791
3. A. Langley, B. Laurie, E. Kasper, and R. Stradling, *Certificate Transparency*, RFC 6962, Internet Engineering Task Force, 2013.
https://datatracker.ietf.org/doc/html/rfc6962
4. D. Eastlake and P. Jones, *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*, RFC 6234, Internet Engineering Task Force, 2011.
https://datatracker.ietf.org/doc/html/rfc6234
5. M. Bellare and P. Rogaway, *Introduction to Modern Cryptography*, UCSD Lecture Notes, 2005.
https://cseweb.ucsd.edu/~mihir/papers/intro.pdf

6. N. Szabo, *Formalizing and Securing Relationships on Public Networks*, First Monday, 2(9), 1997.
https://firstmonday.org/ojs/index.php/fm/article/view/548

7. J. Underhill, *MCEL Technical Specification*, QRCS Corporation.
https://www.qrcscorp.ca/documents/mcel_specification.pdf