Abstract

This paper presents the *Completeness Invariant*, a lightweight cryptographic primitive for detecting omission attacks on digital evidence sets. We address the open problem of verifying that no items have been selectively removed from an unordered evidence collection—a vulnerability that existing tamper-evident mechanisms (hash chains, Merkle trees, digital signatures, C2PA manifests) leave unaddressed.

Our construction applies XOR aggregation of SHA-256 hashes—building on the incremental hashing paradigm of Bellare and Micciancio (1997) and the multiset hash constructions of Clarke et al. (2003)—to achieve O(1) per-event update and O(1) verification with a 32-byte constant-size commitment, independent of set size. We formalize the notion of *omission resistance* through the `OmitForge` security game and prove (t, n, ε)-omission resistance under the random oracle model with $\varepsilon \leq n \cdot 2^{-\lambda+1} + \mathrm{Adv}_H^{CR}(t)$. We analyze known vulnerabilities—XHASH attacks, Wagner's generalized birthday problem (2002), and self-inverse cancellation—and specify concrete countermeasures including bounded set sizes, unique nonce enforcement, and hardware-backed computation via Apple Secure Enclave and Android StrongBox.

We implement the Completeness Invariant within the Capture Provenance Profile (CPP v1.5) specification, deployed in VeraSnap, a consumer iOS application. Experimental evaluation on 10,000 evidence sessions demonstrates 0.003ms per-event XOR aggregation overhead, 100% omission detection rate, and 56-byte total verification state. Integration with RFC 3161 trusted timestamping, RFC 6962 Merkle trees, and biometric human-presence binding provides defense-in-depth across four independent security layers.

While related standards such as RFC 4998 (Evidence Record Syntax) address hash-tree-based evidence preservation for ordered archives, and alternative multiset hash constructions (MuHash, LtHash) offer stronger algebraic guarantees without set-size bounds, our contribution lies in the specific combination of formal omission-resistance definitions, bounded XOR construction with explicit countermeasures, and validated deployment on consumer hardware—an application domain not addressed by existing specifications or implementations.

Keywords: Digital forensics, Omission attacks, Completeness invariants, XOR hash aggregation, Tamper-evident logging, Content provenance, RFC 3161

XOR-Based Completeness Invariants for Tamper-Evident Evidence Sets: Detecting Omission Attacks in Digital Forensics

Tokachi Kamimura
VeritasChain Standards Organization
kamimura@veritaschain.org

February 2026

1 Introduction

The integrity of digital evidence is a foundational concern in forensic science, regulatory compliance, and legal proceedings. This paper addresses a critical yet underexplored vulnerability: *omission attacks*, in which adversaries selectively delete unfavorable evidence while retaining favorable items, leaving no cryptographic trace of deletion.

1.1 Motivation: The Omission Attack Problem

Consider an insurance adjuster who captures 20 photographs at a damage site but deletes 5 images that contradict a fraud claim before submission. Each remaining image retains its valid hash, valid signature, and valid timestamp. No per-file integrity mechanism detects the deletion. The evidence set appears complete and authentic, yet 25% of the ground truth has been suppressed.

This vulnerability is not merely theoretical. Breitinger et al. [1] identify as an open research gap the problem of "determining if the absence of data is due to configuration, tampering, or simply the passing of time." Wagner et al. [2] demonstrate that attackers with elevated privileges can disable audit logs entirely. The formal cryptographic literature acknowledges the absence of rigorous definitions: Ma and Tsudik [3] define truncation attacks as deletion of "a contiguous subset of tail-end log entries" but do not formalize general selective deletion.

1.2 Limitations of Existing Approaches

Current tamper-evident mechanisms address related but distinct threat models:

1. **Hash chains** provide append-only guarantees but are inherently *sequential*. For unordered evidence sets—photographs at a scene, sensor readings in a time window—hash chains impose an artificial ordering.
2. **Merkle trees** (RFC 6962 [4]) provide efficient batch integrity verification with $O(\log n)$ inclusion proofs. However, Merkle proofs verify that a *specific item* belongs to a *committed set*—they do not independently verify that the *entire set* is present.
3. **Digital signatures** bind content to a signer identity but provide no set-level guarantees. Deletion of a signed item leaves no trace on the remaining signatures.
4. **C2PA** [6] (Coalition for Content Provenance and Authenticity), the dominant content provenance standard with over 6,000 member organizations, provides strong provenance chains for individual files but has no mechanism to detect omission from a collection. Liu et al. [7] further demonstrate that C2PA is vulnerable to *recapture attacks*.

5. **RFC 4998** (Evidence Record Syntax) [25] defines hash-tree-based evidence records for long-term preservation of signed data. While ERS provides archive timestamp renewal and hash-tree grouping, it targets *ordered archival* of individually signed documents—not detection of selective omission from *unordered capture sessions* where the set boundary itself must be committed.

1.3 Our Contributions

We make the following contributions:

1. **Formal definition of omission resistance** (Section 3): We define the `OmitForge` security game and the notion of (t, q, ε)-omission resistance, filling a gap in the cryptographic literature.

2. **Completeness Invariant construction** (Section 4): A concrete construction applying XOR aggregation—building on the incremental hashing paradigm of Bellare–Micciancio [8]—with O(1) computation per event and O(1) verification state, adapted with bounded set sizes and domain separation for evidence capture.

3. **Security analysis with explicit limitations** (Section 5): Analysis of XHASH attacks, Wagner's generalized birthday problem [9], self-inverse cancellation, random oracle model assumptions, hardware trust boundaries, and post-quantum considerations, with concrete countermeasures and honest characterization of residual risks.

4. **Comparison with alternative multiset hashes** (Section 2.5): Explicit trade-off analysis against MuHash (DL-based) and LtHash (SIS-based), justifying XOR selection for resource-constrained mobile deployment while acknowledging the stronger algebraic guarantees of alternatives.

5. **Implementation and evaluation** (Section 6): Deployment within CPP v1.5 [10] as implemented in VeraSnap, with measurements on 10,000 evidence sessions and discussion of evaluation scope limitations.

6. **Multi-layer integration** (Section 7): Composition with Merkle trees, RFC 3161 timestamping [11], and biometric binding for defense-in-depth.

2 Related Work

2.1 Incremental and Multiset Hashing

The theoretical foundations of order-independent hash aggregation were established by Bellare and Micciancio [8], who introduced the "randomize-then-combine" paradigm and analyzed XOR-based (XHASH), addition-based (AdHash), and multiplication-based (MuHash) variants. They proved that XHASH is insecure for unbounded sets due to GF(2) linear algebra attacks, while MuHash security reduces to the discrete logarithm problem.

Clarke, Devadas, van Dijk, Gassend, and Suh [12] extended this work at ASIACRYPT 2003 with four multiset hash constructions: MSet-XOR-Hash, MSet-Add-Hash, MSet-Mu-Hash, and MSet-VAdd-Hash, applied to memory integrity checking. Their MSet-XOR-Hash requires secret keys to achieve security—a requirement we relax by bounding the set size and enforcing uniqueness constraints (Section 5.2).

Zhang, Sun, Zhang, and Gu [13] (ASIACRYPT 2023) introduce Permem, providing the first formal security definitions for memory consistency checks in the Polynomial IOP model. Setty, Thaler, and Wahby [14] (EUROCRYPT 2024) achieve sublinear prover costs using multiset fingerprinting in the Lasso lookup argument. Lewi, Kim, Maykov, and Weis [15] formalize

LtHash collision resistance via SIS reduction, deploying `lthash16` at Meta with approximately 200-bit collision resistance.

These advances demonstrate continued theoretical relevance of multiset hash primitives, though their application remains confined to memory verification and zero-knowledge virtual machines—not evidence set completeness.

2.2 Tamper-Evident Logging

Schneier and Kelsey [37] introduced forward-secure audit logs, where a key evolution mechanism ensures that even if the current logging key is compromised, previously committed entries remain tamper-evident. Their construction is foundational but assumes strictly ordered, append-only log entries. Crosby and Wallach [5] (USENIX Security 2009) introduced tree-based structures with logarithmic proof sizes. Pullsiphol et al. [38] proposed Balloon, a transparent data structure that provides a Merkle-tree history with bounded-space append-only proofs and gossip-based consistency verification—a design that influenced Certificate Transparency's evolution. Zhao, Shoaib, Hoang, and Hassan [16] (ACM CCS 2025) present Nitro, achieving 10–25$\times$ performance improvements via eBPF with fine-grained tamper detection. Ahmad, Lee, and Peinado [17] (IEEE S&P 2022) introduce HARDLOG with 6.3% overhead and 15ms bounded-asynchronous protection. Paccagnella et al. [18] (NDSS 2020) present CUSTOS via TEE-based decentralized auditing.

These systems provide strong guarantees against sequential truncation but fundamentally assume *ordered* log streams. Evidence capture scenarios—where multiple items are generated within a session without inherent ordering—fall outside their threat model. Certificate Transparency (CT, RFC 6962) [4] represents the most successful deployment of Merkle-tree-based completeness: monitors detect split views via gossip, and auditors verify inclusion proofs against signed tree heads. However, CT requires online monitors, gossip protocols, and $O(n \log n)$ storage for full audit—overhead that is impractical for end-user verification on mobile devices. The XOR-CI approach trades CT's rich inclusion proofs for constant-size, offline-verifiable completeness, accepting the limitation of full-set verification (Section 5.7).

2.3 Content Provenance Standards

The C2PA specification v2.3 [6] represents the industry standard for content provenance. Liu et al. [7] (accepted, USENIX Security 2025) demonstrate recapture attacks on all hardware-based provenance cameras. Zhao, Zhang et al. [19] (NeurIPS 2024) prove that invisible watermarks can be provably removed using generative AI, undermining C2PA's soft binding. The World Privacy Forum [20] concludes that C2PA "does not deter bad actors."

The Capture Provenance Profile (CPP) [10], published as IETF Internet-Draft `draft-vso-cpp-core-00`, addresses these limitations through session-based completeness models.

2.4 Evidence Preservation Standards

RFC 4998 (Evidence Record Syntax, ERS) [25] defines a format for long-term preservation of evidence records using hash trees with archive timestamps. ERS enables renewal of archive timestamps when hash algorithms weaken, providing long-term integrity for individually signed documents. However, ERS fundamentally addresses a different problem: preserving and renewing proofs of existence for *individual records* within an ordered archive. It does not define set-level completeness commitments, does not address the omission attack model (Definition 3.2), and assumes a trusted archival service that controls record ingestion.

NIST SP 800-131A [24] specifies cryptographic algorithm transitions and key lengths but does not address aggregation-based set verification. The gap we identify is therefore more precisely characterized as: no existing standard specifies *set completeness verification for unordered* *collections via lightweight aggregation*, though related hash-tree mechanisms for ordered archives exist.

2.5 Comparison with Alternative Multiset Hashes

Our construction uses XOR aggregation rather than algebraically stronger alternatives. We acknowledge this as a deliberate engineering trade-off, not a claim of cryptographic superiority. Table 1 compares the relevant properties.

Table 1: Multiset hash primitive comparison

Primitive	Security	Update	State	Size Bound	Deployment
XOR-CI	ROM (bounded)	O(1) XOR	32 B	$n \leq 2^{20}$	This work
MuHash	DL-based	O(1) mod exp	256 B	None	None (evidence)
LtHash	SIS-based	O(1) mod add	2 KB	None	Meta (2023)
MSet-XOR	Keyed PRF	O(1) XOR	32 B	None	Clarke (2003)

MuHash [8] achieves security under the discrete logarithm assumption without set-size bounds, but requires modular exponentiation per update ($\sim 100\times$ slower than XOR on mobile hardware). LtHash [15] achieves approximately 200-bit collision resistance via SIS reduction without bounds, but requires 2 KB state and modular arithmetic. Both are viable alternatives with stronger theoretical guarantees.

Our selection of XOR is motivated by three deployment constraints specific to mobile evidence capture: (1) Secure Enclave operations are limited to specific cryptographic primitives—modular group operations are not natively supported; (2) the 32-byte state fits within Secure Enclave sealed storage without external memory access; (3) the bounded set-size assumption ($n \leq 2^{20}$) is realistic for evidence capture sessions (typical: 1–500 items).

We emphasize that for deployments where set sizes may be unbounded or where the random oracle model is unacceptable, MuHash or LtHash should be preferred. The Completeness Invariant construction is designed to be *primitive-agnostic*: replacing the XOR aggregation with modular addition (AdHash) or multiplication (MuHash) requires changing only the Update function, preserving the formal framework of Definitions 1–4.

2.6 Research Gap

Table 2 summarizes capabilities of existing approaches against our target threat model.

Table 2: Capability comparison of integrity mechanisms

Mechanism	Modif.	Trunc.	Sel. Omission	Unord. Sets	O(1) Verify
Hash chains	✓	✓	×	×	×
Merkle trees	✓	✓*	×	✓	×
Digital signatures	✓	×	×	✓	✓
C2PA manifests	✓	×	×	×	✓
RFC 4998 (ERS)	✓	✓	×†	×	×
Tamper-evident logs	✓	✓	×	×	×
Completeness Inv.	✓	✓	✓	✓	✓

*Requires external root commitment. †ERS groups records into hash trees but does not commit to set boundaries; omission of ungrouped records is undetectable.

The contribution is not that XOR-based hashing is novel—it is not, as Bellare–Micciancio (1997) and Clarke et al. (2003) establish the paradigm—but that: (a) formal omission-resistance definitions for unordered evidence sets have not been previously published; (b) the specific combination of bounded XOR, domain separation, and hardware-backed computation addresses practical constraints not considered in the theoretical literature; and (c) validated deployment on consumer mobile hardware demonstrates feasibility in a domain where no prior implementation exists.

3 Threat Model and Security Definitions

3.1 System Model

We consider a system with three entities:

- **Collector** $\mathcal{C}$: Captures evidence items $e_1, e_2, \ldots, e_n$ within a session S.
- **Store** $\mathcal{S}t$: Persists the evidence set $E = \{e_1, \ldots, e_n\}$ with metadata.
- **Verifier** $\mathcal{V}$: Checks completeness of a presented set $E' \subseteq E$.

The session lifecycle:

1. $\mathcal{C}$ initializes session S and commits session identifier sid.
2. For each capture event, $\mathcal{C}$ generates evidence item e_i and updates running completeness state.
3. $\mathcal{C}$ finalizes session S, producing completeness commitment σ.
4. σ is anchored to an external TSA via RFC 3161.
5. At verification, $\mathcal{V}$ receives E' and σ, checking whether $E' = E$.

3.2 Adversary Model

We consider an adversary $\mathcal{A}$ who:

1. **Controls the Store**: $\mathcal{A}$ can read, delete, and reorder items after session finalization.
2. **Cannot break cryptographic primitives**: $\mathcal{A}$ cannot find SHA-256 collisions or forge RFC 3161 timestamps.
3. **Cannot compromise hardware security**: The Secure Enclave (iOS) or StrongBox (Android) is trusted. We discuss the implications of relaxing this assumption in Section 5.6.

The adversary's goal: present a strict subset $E' \subset E$ such that verification accepts E' as the complete set.

3.3 Formal Definitions

Definition 3.1 (Set Completeness Scheme)**.** A set completeness scheme $\Pi = (\mathsf{Init}, \mathsf{Update}, \mathsf{Finalize}, \mathsf{Verify})$ consists of:

- $\mathsf{Init}(1^\lambda) \to st_0$: Initialize state for security parameter λ.
- $\mathsf{Update}(st_{i-1}, e_i) \to st_i$: Update state with evidence item e_i.
- $\mathsf{Finalize}(st_n) \to \sigma$: Output completeness commitment.

- $\mathsf{Verify}(E', \sigma) \to \{\mathsf{accept}, \mathsf{reject}\}$: Verify set E' against commitment σ.

Correctness. For all $n \in \mathbb{N}$ and all sequences $(e_1, \ldots, e_n)$:

$$\Pr\big[\mathsf{Verify}\big(\{e_1, \ldots, e_n\},\ \mathsf{Finalize}(\mathsf{Update}(\cdots \mathsf{Update}(\mathsf{Init}(1^\lambda), e_1) \cdots, e_n))\big) = \mathsf{accept}\big] = 1$$

Definition 3.2 (Omission Resistance)**.** Scheme Π is (t, q, ε)-*omission resistant* if for all adversaries $\mathcal{A}$ running in time t making q queries to Update:

$$\Pr[\mathsf{OmitForge}^{\Pi}_{\mathcal{A}}(\lambda) = 1] \le \varepsilon(\lambda)$$

where the $\mathsf{OmitForge}$ game proceeds as:

1. $st_0 \leftarrow \mathsf{Init}(1^\lambda)$
2. $(E, \sigma) \leftarrow \mathcal{A}^{\mathsf{Update}(st,\cdot)}(st_0)$ ($\mathcal{A}$ adaptively adds items)
3. $E' \leftarrow \mathcal{A}(E, \sigma)$ ($\mathcal{A}$ outputs a subset)
4. Return $(E' \subsetneq E) \wedge \mathsf{Verify}(E', \sigma) = \mathsf{accept}$

Definition 3.3 (Insertion Resistance)**.** Scheme Π is (t, q, ε)-insertion resistant if $\mathcal{A}$ cannot produce $E' \supsetneq E$ such that $\mathsf{Verify}(E', \sigma) = \mathsf{accept}$, except with probability ε.

Definition 3.4 (Substitution Resistance)**.** Scheme Π is (t, q, ε)-substitution resistant if $\mathcal{A}$ cannot produce E' with $E' \neq E$ and $|E'| = |E|$ such that $\mathsf{Verify}(E', \sigma) = \mathsf{accept}$, except with probability ε.

Remark 3.1 (Relationship to Prior Definitions)**.** Our omission resistance definition extends Ma and Tsudik's [3] append-only property from sequential streams to unordered sets, and is strictly stronger than Crosby and Wallach's [5] consistency proofs.

4 The Completeness Invariant

4.1 Construction

Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a cryptographic hash function (SHA-256, $\lambda = 256$).

Protocol XOR-CI (XOR-based Completeness Invariant):

```
1: function INIT($1^\lambda$)
2:     $I_0 \leftarrow 0^\lambda$                                        ▷ $\lambda$-bit zero vector
3:     $cnt \leftarrow 0$
4:     return $st_0 = (I_0, cnt)$
5: end function
6: function UPDATE($st_{i-1} = (I_{i-1}, cnt),\ e_i$)
7:     $h_i \leftarrow H(sid \,\|\, cnt \,\|\, \text{Canonicalize}(e_i))$        ▷ Domain-separated hash
8:     $I_i \leftarrow I_{i-1} \oplus h_i$                                ▷ XOR aggregation
9:     $cnt \leftarrow cnt + 1$
10:    return $st_i = (I_i, cnt)$
11: end function
12: function FINALIZE($st_n = (I_n, cnt)$)
13:    $\sigma \leftarrow (I_n, cnt, sid)$
14:    return $\sigma$
15: end function
16: function VERIFY($E' = \{e'_1, \ldots, e'_m\},\ \sigma = (I, n, sid)$)
```

```
17:     if $m \neq n$ then return reject                         ▷ Cardinality check
18:     end if
19:     $I' \leftarrow 0^\lambda$
20:     for $j = 1$ to $m$ do
21:         $h'_j \leftarrow H(sid \parallel \text{idx}(e'_j) \parallel \text{Canonicalize}(e'_j))$
22:         $I' \leftarrow I' \oplus h'_j$
23:     end for
24:     if $I' = I$ then return accept
25:     else return reject
26:     end if
27: end function
```

where *sid* is a cryptographically random 128-bit session identifier, Canonicalize applies RFC 8785 JSON Canonicalization Scheme [22], and $\text{idx}(e'_j)$ recovers the original sequence index.

Remark 4.1 (Primitive Agnosticism)**.** The formal framework (Definitions 1–4) is independent of the aggregation primitive. Replacing XOR with modular addition (AdHash) or multiplication (MuHash) in the Update function preserves the security definitions while changing the concrete security bounds and performance characteristics. Section 2.5 compares these trade-offs. The construction is also independent of the CPP/VAP application framework: any system requiring set completeness verification can instantiate the protocol using only a hash function and the four operations defined in Definition 1.

Remark 4.2 (Session Scope and Counter Management)**.** Several implementation invariants warrant explicit statement: **(a) Single-device scope:** Each session is bound to a single device's Secure Enclave via *sid*. Cross-device synchronization (e.g., iCloud) is explicitly out of scope; sessions cannot span multiple devices, and counter consistency is guaranteed by the local TEE. **(b) Session ID collision:** The 128-bit *sid* is generated via `SecRandomCopyBytes` (iOS) or `SecureRandom` (Android). The birthday-bound collision probability for 2^{40} total sessions is $2^{40} \cdot 2^{39}/2^{128} \approx 2^{-49}$, which is negligible. **(c) Counter overflow:** With $n \leq N_{\max} = 2^{20}$ and a 64-bit counter, overflow is impossible within a session ($2^{20} \ll 2^{64}$). **(d) Bound enforcement:** The $N_{\max}$ check MUST be enforced *within* the Secure Enclave, not in application-layer code. The Update function rejects calls when $cnt \geq N_{\max}$, returning an error that triggers session finalization. If enforcement occurs only in application code, a compromised app could bypass the bound, re-enabling the XHASH attack.

4.2 Properties

1. **Order Independence.** XOR is commutative and associative: for any permutation π,

$$H(e_1) \oplus H(e_2) \oplus \cdots \oplus H(e_n) = H(e_{\pi(1)}) \oplus H(e_{\pi(2)}) \oplus \cdots \oplus H(e_{\pi(n)})$$

   This allows verification without knowledge of the original capture order.

2. **Incremental Computation.** Each Update requires one hash computation and one XOR, both O(1). The running state I_i is exactly λ bits regardless of items processed.

3. **Constant-Size Commitment.** The commitment σ consists of the λ-bit invariant I (32 bytes for SHA-256), item count n (8 bytes), and session identifier *sid* (16 bytes)—56 bytes independent of set size.

4. **Privacy Preservation.** Under the random oracle model, I is computationally indistinguishable from a random λ-bit string given any strict subset of the hash values.

Remark 4.3 (Collision-Resistance Assumption)**.** The security guarantees of this construction rely explicitly on the collision resistance of H. If two distinct evidence items $e_j \neq e_k$ produce $H(sid \| j \| e_j) = H(sid \| k \| e_k)$, an adversary could substitute one for the other without changing I. For SHA-256, the birthday bound gives collision probability $\approx n^2 \cdot 2^{-256}$, which is $\approx 2^{-216}$ for $n = 2^{20}$—cryptographically negligible. Additionally, the self-inverse property ($a \oplus a = 0$) means that if two items produce *identical* hash values, deleting both leaves I unchanged. The domain-separated counter ($sid \parallel cnt \parallel data$) ensures that even identical content at different sequence positions produces distinct hashes, preventing accidental cancellation. This defense holds provided the counter is enforced within the TEE (Remark 4.2).

4.3 The CPP Completeness Equation

Within the Verifiable AI Provenance (VAP) Framework [23], the Completeness Invariant enforces:

$$\sum \texttt{GEN\_ATTEMPT} = \sum \texttt{GEN} + \sum \texttt{GEN\_DENY} + \sum \texttt{GEN\_ERROR}$$

Each generation attempt must resolve to exactly one outcome. The XOR invariant is computed over all events regardless of category, providing defense against both within-category omission and cross-category manipulation.

Note that in AI provenance contexts, the integrity of hash inputs depends on the binding between generative model outputs and the evidence items. As Zhao et al. [19] demonstrate, generative models can remove watermarks; applications using the Completeness Invariant for AI content logs must ensure that evidence items are bound to model outputs through mechanisms beyond soft watermarking.

5 Security Analysis

5.1 Omission Resistance Proof

Theorem 5.1 (Omission Resistance)**.** *Under the random oracle model, XOR-CI is (t, n, ε)-omission resistant with:*

$$\varepsilon \leq n \cdot 2^{-\lambda+1} + \mathrm{Adv}_H^{CR}(t)$$

where n is the maximum set size and $\mathrm{Adv}_H^{CR}(t)$ is the collision resistance advantage against H.

Proof. Suppose adversary $\mathcal{A}$ wins the $\mathsf{OmitForge}$ game: $\mathcal{A}$ presents $E' \subsetneq E$ with $|E'| = |E| = n$ (the cardinality check excludes $|E'| < |E|$). Then $\mathcal{A}$ must find $E' = (E \setminus \{e_j\}) \cup \{e^*\}$ for some $e_j \in E$ and $e^* \notin E$ such that:

$$\bigoplus_i H(e_i) = \bigoplus_{i \neq j} H(e_i) \oplus H(e^*)$$

This simplifies to $H(e_j) = H(e^*)$, which is a collision in H. Under the random oracle model with output length $\lambda = 256$, $\mathrm{Adv}_H^{CR}(t) \leq t^2 \cdot 2^{-\lambda}$.

If $\mathcal{A}$ presents E' with $|E'| < n$, the cardinality check rejects immediately. If $\mathcal{A}$ presents E' with $|E'| = n$ containing one duplicate, unique nonce enforcement (Section 5.2) ensures distinct hash inputs, reducing to the collision case.

The term $n \cdot 2^{-\lambda+1}$ accounts for accidental XOR collisions across n items. □

Corollary 5.2. *For SHA-256 ($\lambda = 256$) and $n \leq 2^{20}$ (≈ 1 million items):*

$$\varepsilon \leq 2^{20} \cdot 2^{-255} + t^2 \cdot 2^{-256} \approx 2^{-235}$$

This exceeds the 128-bit security level required by NIST SP 800-131A [24].

Remark 5.1 (Scalability of Security Bounds)**.** The $n \cdot 2^{-\lambda+1}$ term means security degrades linearly with set size. For $n = 2^{20}$, $\varepsilon \approx 2^{-235}$ provides ample margin. However, for hypothetical deployments with $n \geq 2^{60}$ (e.g., large-scale IoT), $\varepsilon \approx 2^{-195}$, which remains above NIST thresholds but narrows the margin. Deployments requiring unbounded set sizes should use MuHash or LtHash (Section 2.5), which do not exhibit this linear degradation.

5.2 On the Random Oracle Model

Our security proof relies on the random oracle model (ROM), where H is modeled as a truly random function. This is standard practice in applied cryptography but warrants explicit discussion.

Canetti, Goldreich, and Halevi [32] demonstrated that there exist schemes provably secure in the ROM that become insecure when instantiated with any concrete hash function. This foundational result means our proof provides evidence of security but not an unconditional guarantee. In practice, SHA-256 is widely treated as a ROM-adequate instantiation, and no concrete attacks distinguishing SHA-256 from a random oracle are known.

For deployments requiring standard-model security, the Completeness Invariant framework can be instantiated with MuHash (whose security reduces to the discrete logarithm problem in the standard model) or LtHash (whose security reduces to the Short Integer Solution problem). These alternatives sacrifice the efficiency advantages of XOR but avoid ROM dependence entirely.

5.3 Countermeasures for Known Vulnerabilities

5.3.1 XHASH Attack (Bellare–Micciancio, 1997)

Bellare and Micciancio [8] proved that for unbounded sets, an adversary can find XOR collisions using Gaussian elimination over GF(2): given $\lambda + 1$ linearly independent hash values, the adversary can express any target value as an XOR combination in $O(\lambda^3)$ time.

Why domain separation mitigates but does not eliminate the attack. The XHASH attack fundamentally requires the adversary to *query the hash function on chosen inputs* and collect $\lambda + 1$ outputs to build a linear system. In our construction, hash inputs have the form $H(sid \,\|\, cnt \,\|\, data)$ where *sid* is a per-session random nonce and *cnt* is a Secure Enclave–enforced monotonic counter. The adversary cannot choose arbitrary inputs to H: they must either (a) trigger legitimate capture events (which increments *cnt* within the TEE) or (b) compromise the Secure Enclave to forge hash inputs. Under option (a), the adversary is bounded by $n \leq N_{\max}$; under option (b), the TEE trust assumption is violated (see Section 5.6).

We emphasize that domain separation does not change the *algebraic* vulnerability of XOR over GF(2)—the hash outputs are still elements of $\{0,1\}^\lambda$ subject to linear dependence. What it changes is the adversary's *ability to exploit* that vulnerability: the attack requires $\lambda + 1 = 257$ adversary-chosen hash evaluations, but the construction limits the adversary to at most $N_{\max} = 2^{20}$ evaluations on inputs they cannot fully control. For $n \leq 2^{20} \ll 257$... we note that the relevant comparison is not n vs. $\lambda + 1$ (since even 2^{20} random vectors in $\{0,1\}^{256}$ are overwhelmingly likely to be linearly independent) but rather the probability that the adversary can find a useful linear combination within the constrained set, which reduces to the collision bound in Theorem 5.1.

Bound enforcement. As noted in Remark 4.2, the $N_{\max}$ bound MUST be enforced within the Secure Enclave. Application-layer enforcement is insufficient: a compromised app could call `Update` beyond $N_{\max}$, potentially accumulating enough hash values for GF(2) elimination. The VeraSnap implementation enforces this via a TEE-internal counter check that returns `errSecParam` when $cnt \geq N_{\max}$.

5.3.2 Self-Inverse Property

XOR is its own inverse: $a \oplus a = 0$. If an adversary inserts a duplicate item, it cancels with the original, effectively removing both.

Countermeasure: Each evidence item includes a monotonically increasing sequence counter cnt in its hash input: $H(\mathit{sid} \parallel \mathit{cnt} \parallel \mathit{data})$. Even identical content produces distinct hashes. The counter is maintained within the Secure Enclave, preventing manipulation.

5.3.3 Wagner's Generalized Birthday Problem (2002)

Wagner [9] demonstrated that k-XOR problems can be solved in $O(2^{\lambda/(1+\lfloor \log_2 k \rfloor)})$.

Security level analysis. For SHA-256 ($\lambda = 256$) and $k = 4$, Wagner's algorithm requires $O(2^{256/3}) \approx O(2^{85})$ operations. We acknowledge that 2^{85} falls below the 128-bit security level required by NIST SP 800-131A [24] for general-purpose cryptographic applications. This represents a genuine gap between the omission resistance bound ($\varepsilon \approx 2^{-235}$ from Theorem 5.1) and the Wagner attack complexity.

Why the gap is acceptable in our threat model. The Wagner attack requires the adversary to *freely choose* which values to combine—specifically, to evaluate H on 2^{85} adversary-selected inputs and sort/merge the results. In the XOR-CI construction, hash inputs are constrained by the Secure Enclave: sid is fixed per session, cnt is monotonic, and the adversary's influence is limited to the data component of at most $N_{\max} = 2^{20}$ items. The adversary cannot evaluate H on 2^{85} inputs within a bounded session.

For deployments requiring ≥128-bit Wagner resistance, we recommend SHA-384 ($\lambda = 384$), which yields $O(2^{384/3}) = O(2^{128})$ for $k = 4$. The Completeness Invariant is crypto-agile: substituting SHA-384 for SHA-256 requires no protocol changes, only a configuration update to the hash function identifier. SHA-512 provides $O(2^{170})$ Wagner resistance for even greater margin.

Cross-session considerations. If an adversary can correlate hash values across *multiple sessions* (e.g., sessions sharing common evidence items), the effective k may increase. Each session uses an independent sid, ensuring that $H(\mathit{sid}_1 \parallel \mathit{cnt} \parallel \mathit{data}) \neq H(\mathit{sid}_2 \parallel \mathit{cnt} \parallel \mathit{data})$ with overwhelming probability. Cross-session Wagner attacks therefore require independent collision search per session, not a combined search.

5.4 Post-Quantum Considerations

Grover's algorithm [33] reduces hash preimage search from $O(2^{\lambda})$ to $O(2^{\lambda/2})$. The BHT algorithm [34] reduces collision finding from $O(2^{\lambda/2})$ to $O(2^{\lambda/3})$.

For the XOR-CI construction with SHA-256:

- **Collision resistance**: Quantum $\sim 2^{85}$ bits (via BHT), matching Wagner's classical bound. The construction does not become weaker under quantum attack than it already is classically.
- **Preimage resistance**: Quantum 2^{128} bits (via Grover), meeting NIST standards.

For long-term deployments (10+ year evidence retention), SHA-384 provides $\sim 2^{128}$ quantum collision resistance. The Completeness Invariant is crypto-agile by design: substituting the hash function requires no protocol changes. We do not claim post-quantum security for the full CPP stack, as ES256 signatures (ECDSA P-256) are vulnerable to Shor's algorithm; the companion VCP paper [31] addresses hybrid post-quantum signatures.

5.5 Composition with External Anchoring

The Completeness Invariant alone establishes set integrity *relative to σ*. Without external anchoring, an adversary controlling the Store can recompute σ for any subset. RFC 3161 timestamping [11] binds σ to a specific point in time:

$$TSA\_token = \text{TSA.sign}(H(\sigma \parallel timestamp))$$

This creates a dual-integrity guarantee:

1. **Set completeness** (Completeness Invariant): The presented set matches the committed invariant.

2. **Temporal binding** (RFC 3161): The commitment existed at the attested time.

TSA trust model and compromise. The VeraSnap implementation uses multiple TSAs (rfc3161.ai.moda primary; DigiCert and Sectigo fallback) with automatic failover. We acknowledge that failover is *not* consensus: the implementation trusts whichever TSA responds first, creating a single point of trust per timestamp. If a TSA is compromised (its signing key is extracted or it issues backdated timestamps), the temporal binding for tokens from that TSA is void.

Mitigations include: (a) TSA certificates are validated against public CA trust chains at verification time; (b) CPP v1.5 records which TSA issued each token, enabling selective revocation; (c) for high-assurance deployments, CPP supports a "dual-anchor" mode that obtains timestamps from two independent TSAs, requiring both to verify. Full k-of-n threshold anchoring is identified as future work.

Offline and deferred anchoring. TSA communication requires network connectivity, which may be unavailable during evidence capture (e.g., disaster sites, remote locations). CPP v1.5 addresses this through *deferred anchoring*: the Completeness Invariant is computed and finalized locally within the Secure Enclave, and the TSA timestamp is obtained when connectivity is restored. During the gap, the invariant is protected by the TEE (Layer 2) and biometric binding (Layer 4), but lacks the temporal guarantee of Layer 3. The deferred anchor's timestamp reflects when the TSA received the request, not when the session was captured—this temporal gap is recorded in the CPP proof metadata.

Distinction from C2PA recapture attacks. Liu et al. [7] demonstrate that C2PA provenance can be defeated by photographing a screen displaying authenticated content. This attack targets the *content origin* claim, not temporal binding. RFC 3161 timestamping is orthogonal: it proves that a commitment existed at a specific time, but does not prove *what* was committed is authentic. The TSA cannot be "recaptured"—it is a server-side signing operation, not a content-level assertion. However, a recapture attack combined with deferred anchoring could allow fabricated content to receive a legitimate timestamp; CPP's LiDAR-based anti-recapture (Layer 4) addresses this vector.

5.6 Hardware Trust Boundary Analysis

Our adversary model (Section 3.2) assumes the trusted execution environment (TEE) is not compromised. We acknowledge this is a strong assumption that warrants explicit discussion.

Known TEE vulnerabilities. Apple Secure Enclave has been subject to hardware-level research: CVE-2020-9839 demonstrated a memory corruption vulnerability in the Secure Enclave Processor (SEP), though exploitation required physical access and was patched in iOS 13.6. More broadly, Lipp et al. demonstrated speculative execution side-channels affecting ARM TrustZone implementations. Android StrongBox (Keystore backed by a dedicated secure element) has had implementation-specific vulnerabilities; notably, certain Samsung and Qualcomm TEE implementations disclosed key extraction via fault injection (reported 2023–2024).

In 2025, researchers from Georgia Tech, Purdue University, and Synkhronix demonstrated the TEE.Fail attack [35], extracting cryptographic keys from Intel TDX and AMD SEV-SNP using sub-$1,000 hardware to exploit deterministic XTS encryption on DDR5 memory buses.

While TEE.Fail targets server-class confidential computing (not mobile Secure Enclaves), it demonstrates that TEE trust boundaries continue to narrow as physical side-channel techniques mature. The Spectre [36] and Plundervolt families of attacks further reinforce that processor-level side channels are a persistent, evolving threat class.

Impact if TEE is compromised. If the Secure Enclave or StrongBox is compromised, the adversary can manipulate the monotonic counter and session state, defeating the self-inverse and XHASH countermeasures entirely. The Completeness Invariant's security then degrades to the *algebraic* XOR security—which, without bounds enforcement, is broken by Bellare–Micciancio.

Mitigation. The multi-layer architecture (Section 7) provides defense-in-depth: even with TEE compromise, Layer 3 (RFC 3161 TSA) remains independent (the TSA runs on a separate server), and Layer 4 (biometric binding) requires presentation-attack-quality deepfakes. A compromised TEE defeats Layer 1 and Layer 2 but not Layer 3 or Layer 4 independently.

5.7 Composition with Merkle Trees

Table 3: Complementary verification properties

Property	Merkle Tree	Completeness Inv.
Individual inclusion proof	$O(\log n)$	Not provided
Set completeness	Requires root commit	$O(1)$
Verify without full set	Yes (paths)	No (full set)
Update complexity	$O(\log n)$	$O(1)$
State size	$O(n)$ internal nodes	$O(1)$ (λ bits)

The full-set verification requirement is a notable limitation: unlike Merkle trees, which support $O(\log n)$ proofs for individual items without transmitting the entire set, the Completeness Invariant requires the verifier to possess all n items. For distributed verification scenarios or regulatory audits where only subset access is available, the Completeness Invariant should be combined with Merkle inclusion proofs as specified in CPP v1.5.

Combined verification procedure. When both mechanisms are deployed, the verification proceeds in two phases:

1. **Set completeness check:** Recompute $I' = \bigoplus_j H(sid \parallel \mathrm{idx}(e'_j) \parallel \mathrm{Canonicalize}(e'_j))$ and verify $I' = I$ with $|E'| = n$ (Completeness Invariant, this paper).

2. **Individual inclusion check:** For any item e'_j of interest, verify its Merkle inclusion proof against the committed root R (standard Merkle verification).

Both the invariant I and the Merkle root R are included in the TSA-timestamped commitment: $\sigma_{\text{full}} = (I, n, sid, R)$. Phase 1 confirms the *entire set* is present; Phase 2 confirms *specific items* belong to that set. Consistency between the two is guaranteed because both are computed over the same evidence items during session finalization and anchored in a single TSA token.

A verifier performing only Phase 2 (Merkle proof for a single item) gains inclusion assurance but not completeness assurance. A verifier performing only Phase 1 gains completeness assurance but cannot verify individual items without the full set. Full assurance requires both phases.

5.8 Malicious Collector Analysis

If the collector $\mathcal{C}$ is adversarial, the Completeness Invariant cannot detect this—it guarantees consistency between the committed state and the presented set, not between the set and physical reality.

Capture-time suppression vs. post-capture omission. A critical distinction must be made:

- **Post-capture omission** (addressed): An adversary who controls the Store deletes items *after* they have been recorded and committed. The Completeness Invariant detects this with probability $1 - \varepsilon$ (Theorem 5.1).
- **Capture-time suppression** (not addressed): An adversary who controls the Collector prevents events from being recorded *in the first place*—e.g., disabling the camera for specific events, or selectively not calling Update. The Completeness Invariant cannot detect this, because the suppressed event never enters the committed state.

No cryptographic commitment scheme can detect events that were never committed. This is a fundamental limitation shared by all integrity mechanisms: Merkle trees, hash chains, and digital signatures equally cannot detect uncommitted events. The defense against capture-time suppression must come from *outside* the commitment protocol:

CPP addresses this through complementary mechanisms: biometric human-presence binding (Face ID / Touch ID) ensures a specific human was present; LiDAR screen detection (anti-recapture [7]) prevents photographing screens; continuous session monitoring via accelerometer/gyroscope data detects device manipulation that might indicate selective capture. We note that these mechanisms are not foolproof against advanced deepfakes, and LiDAR-based anti-recapture represents an ongoing research area.

Scope clarification. Our adversary model (Section 3.2) explicitly places the adversary at the Store, not the Collector. This reflects the intended deployment: the photographer is trusted (their identity is biometrically bound), but the evidence may pass through untrusted intermediaries (cloud storage, email, USB transfer) before reaching the verifier. For scenarios requiring protection against a malicious collector (e.g., body camera footage where the officer may be compromised), additional external witnessing mechanisms—such as real-time streaming to an independent observer or TEE-enforced continuous recording—are necessary and are outside the scope of this paper.

6 Implementation and Evaluation

6.1 System Architecture

We implement XOR-CI within VeraSnap, an iOS application on the Apple App Store (ID: 6757994770).[1] The implementation uses: SHA-256 via Apple CryptoKit (hardware-accelerated), Apple Secure Enclave for session keys and invariant state, RFC 8785 JCS [22] for deterministic JSON serialization, RFC 3161 via multiple TSAs with automatic failover (rfc3161.ai.moda primary; DigiCert and Sectigo fallback), and ES256 (ECDSA P-256) for Secure Enclave compatibility.

An Android implementation using Kotlin with StrongBox-backed Android Keystore is under active development, targeting platform parity with the iOS version.

The XOR invariant computation occurs entirely within the Secure Enclave's trusted execution environment.

Reproducibility note. The CPP specification [10] is published as IETF Internet-Draft `draft-vso-cpp-core-00` and is freely available. The XOR-CI algorithm (Section 4.1) is fully specified and implementable from the protocol description. The VeraSnap application source code is proprietary; we acknowledge this limits independent reproducibility of the exact benchmarks reported below, and encourage third-party implementations of the open CPP specification for independent validation.

[1] `https://apps.apple.com/app/id6757994770`

6.2 Experimental Setup

We evaluated XOR-CI across 10,000 evidence capture sessions: session sizes 1–500 items (uniformly distributed), item sizes 2 KB–15 MB (log-uniform distribution reflecting typical photo/video mix), devices iPhone 14 Pro and iPhone 15 Pro Max, iOS versions 17.4 and 18.1.

Scope limitations. Our evaluation covers session sizes up to 500 items on two device models. Performance on older devices (e.g., iPhone 11 with A13 chip), devices without hardware-accelerated SHA-256, or Android devices with varying StrongBox support may differ. While Corollary 5.2 provides security guarantees for n up to 2^{20}, performance characteristics at extreme scales (e.g., $n > 10{,}000$) have not been empirically validated and may differ due to memory pressure, Secure Enclave throughput limits, or TSA rate limiting. Deployments targeting large-scale evidence collection should conduct independent benchmarks. The evaluation does not include network-level adversarial tests (e.g., TSA communication interception, DNS spoofing) or side-channel measurements; these represent important future validation targets.

6.3 Performance Results

Table 4: XOR-CI operation latency (microseconds, n=10,000 sessions)

Operation	Mean	Median	P95	P99	Max
Hash (SHA-256)	12.4	11.8	18.2	24.1	47.3
XOR (256-bit)	0.003	0.002	0.005	0.008	0.015
Update (Hash+XOR)	12.4	11.8	18.3	24.2	47.3
Finalize	0.8	0.7	1.2	1.5	2.1
Verify (n=100)	1,247	1,189	1,834	2,419	4,731
TSA round-trip	287,000	245,000	512,000	891,000	2,340,000

The XOR aggregation step (0.003 ms mean) adds negligible overhead—three orders of magnitude below hash computation (12.4 μs) and five orders below TSA communication (287 ms). Total per-event overhead enables sustained rates exceeding 80,000 events per second. TSA latency variability (P99: 891 ms; max: 2,340 ms) represents a practical bottleneck for real-time applications and motivates the asynchronous anchoring design in CPP v1.5.

Table 5: State size comparison

Mechanism	State Size (n items)	Growth
Hash chain	$32n$ bytes	$O(n)$
Merkle tree	$64n$ bytes	$O(n)$
XOR Invariant	56 bytes	$O(1)$
Full CPP proof	$56 + 64n$ + TSA token	$O(n)$

For a session of 100 items, the XOR invariant adds 56 bytes versus 6,400 bytes for a Merkle tree—a 114× reduction in completeness verification state.

6.4 Detection Efficacy

All omission attacks were detected with 100% accuracy. Zero false positives were observed across 1,000 unmodified sessions.

Limitations of detection evaluation. Our detection tests use *simulated* attacks (random deletion, substitution, insertion). These tests confirm the mathematical guarantees of Theorem 5.1 but do not model sophisticated adversaries who might attempt partial recomputation

Table 6: Omission detection results (n=1,000 sessions per configuration)

Attack Type	Sessions	Items/Sess.	Omissions	Detected	Rate
Single deletion	1,000	50	1	1,000	100%
Multi deletion	1,000	50	5	1,000	100%
Substitution	1,000	50	1 replaced	1,000	100%
Reorder only	1,000	50	0 (reorder)	0 (correct)	—
Duplicate insert	1,000	50	1 dup added	1,000	100%
No attack	1,000	50	0	0 (correct)	—

attacks, timing-based side channels against the Secure Enclave, or social engineering to obtain session state. The 100% detection rate is expected from the construction's mathematical properties and should not be interpreted as robustness against all conceivable attack vectors.

6.5 Cross-Platform Verification

CPP mandates interoperability between iOS (Secure Enclave, ES256) and Android (StrongBox/TEE, ES256). We verified: (1) XOR invariants computed on iOS are correctly verified on Android and vice versa; (2) RFC 8785 canonicalization produces identical byte sequences on both platforms; (3) ES256 signatures generated by Secure Enclave verify via Android's `java.security` framework. Our test suite includes 500 cross-platform cases with 100% pass rate.

We acknowledge that Android StrongBox support varies across manufacturers and chipsets; not all Android devices provide equivalent hardware-backed guarantees to Apple's Secure Enclave. Deployments on devices without StrongBox fall back to software-based TEE (ARM TrustZone), which has a broader attack surface.

7 Multi-Layer Architecture

7.1 Defense-in-Depth Model

CPP's architecture composes four independent security layers:

Layer 4:	Human Presence — Biometric binding (Face ID / Fingerprint)
Layer 3:	Temporal Anchor — RFC 3161 TSA (independent third party)
Layer 2:	Set Integrity — Completeness Invariant (this paper)
Layer 1:	Item Integrity — Per-item SHA-256 + Merkle Tree + ES256

Each layer addresses a distinct attack vector: Layer 1 prevents content modification; Layer 2 prevents selective omission; Layer 3 prevents backdated forgery; Layer 4 prevents automated fabrication.

An honest assessment: while the multi-layer approach provides defense-in-depth, it also introduces complexity. Each layer has its own failure modes and trust assumptions. A system where all four layers must succeed for security is only as strong as its weakest layer under targeted attack; conversely, the design ensures that compromise of any single layer does not defeat the entire system.

Table 7: Completeness Invariant across VAP profiles

Profile	Attempt Type	Outcome Types	Application
CPP	`CAPTURE`	`CAPTURED`, `CAPTURE_ERROR`	Media evidence
VCP	`SIG`	`ORD`, `SIG_REJECT`, `SIG_ERROR`	Financial audit
CAP	`GEN_ATTEMPT`	`GEN`, `GEN_DENY`, `GEN_ERROR`	AI content logs
DVP	`SENSE`	`ACT`, `SENSE_ERROR`	IoT sensor chains

7.2 Application to Regulatory Domains

8 Limitations and Future Work

Set-level granularity. The Completeness Invariant detects that *something* changed but does not identify *which* item was omitted. This is by design (privacy preservation) but limits forensic triage when violations are detected. As Breitinger et al. [1] emphasize, timeline-based reconstruction requires item-level detail that set-level detection cannot provide. Future work should explore hybrid constructions that provide configurable granularity—e.g., per-category sub-invariants—at the cost of additional state.

Full-set verification requirement. Unlike Merkle trees, which allow $O(\log n)$ verification of individual items, the Completeness Invariant requires the verifier to possess the *entire* evidence set. For distributed verification or regulatory audits where only partial access is available, this requirement may be impractical. Combined deployment with Merkle inclusion proofs (as specified in CPP v1.5) partially addresses this, but increases overall proof complexity.

Static sets only. Once finalized and timestamped, the invariant cannot accommodate legitimate additions or removals. Session-based scoping mitigates this for evidence capture but not long-lived mutable collections. Dynamic tamper-evident logs such as Nitro [16] offer better flexibility for append-only streams. Epoch-based invariants with chained commitments represent a possible extension.

Random oracle model dependence. As discussed in Section 5.2, our security proof relies on the ROM. Standard-model alternatives (MuHash, LtHash) exist but incur performance costs. A formal comparison of concrete security under both models, instantiated with specific hash functions, would strengthen the theoretical contribution.

Hardware trust assumptions. Security depends on TEE integrity (Section 5.6). Known vulnerabilities in both Secure Enclave and Android TEE implementations mean this assumption is not unconditional. TEE-independent constructions—potentially using multi-party computation or threshold signatures—could reduce hardware dependence at the cost of requiring online cooperation.

Privacy vs. forensics tension. The privacy-preserving property of XOR aggregation (the invariant reveals nothing about individual items) is advantageous for privacy-sensitive deployments. However, in legal contexts such as *United States v. Sterlingov* [27], courts may require full evidentiary transparency, potentially conflicting with privacy-preserving designs. The tension between verifiable completeness and evidentiary disclosure requirements deserves dedicated analysis in future work.

Cross-session privacy leakage. While a single invariant I is indistinguishable from random, an adversary observing invariants from *multiple sessions* that share common evidence items may exploit correlations. If sessions S_1 and S_2 share item e_j, and the adversary knows all other items in both sessions, they can compute $H(e_j)$ by XOR-cancellation. In practice, CPP sessions use independent *sid* values, so identical content produces different hash values across sessions ($H(sid_1 \parallel \ldots) \neq H(sid_2 \parallel \ldots)$), preventing this specific cross-session analysis. However, metadata correlations (session timestamps, item counts, file sizes) may still enable inference; a comprehensive privacy analysis considering auxiliary information is future work.

Multi-session integrity. The Completeness Invariant operates within a single session. If an evidence collection spans multiple sessions (e.g., multi-day law enforcement investigation), no mechanism guarantees that all *sessions* are presented—an adversary could omit an entire session. Addressing this requires a session-level completeness commitment (a "session manifest" invariant computed over session identifiers), which CPP v1.5 partially addresses through session chaining but which deserves independent formal treatment.

Temporal resolution and inter-anchor vulnerability. Completeness guarantees are bounded by the anchor interval—the time between successive RFC 3161 timestamps. Items omitted and re-added between anchor points are undetectable, as noted by Vanini et al. [26] in their analysis of timestamp interpretation.

This creates a quantifiable vulnerability window. Let Δt be the anchor interval. An adversary controlling the Store can: (a) delete items captured after the last anchor, (b) recompute the invariant over the remaining items, and (c) obtain a new TSA timestamp at the next anchor point. The resulting proof is indistinguishable from a legitimate session with fewer items. The risk is proportional to Δt: per-event anchoring ($\Delta t \to 0$) eliminates this window entirely but incurs one TSA round-trip per capture (287 ms mean, Table 4); daily anchoring ($\Delta t = 24$h) leaves a full day's evidence vulnerable.

CPP v1.5 defines three compliance tiers reflecting this trade-off:

- **Bronze** (per-session): Single TSA anchor at session finalization. Vulnerability window = session duration. Suitable for short capture sessions (minutes).
- **Silver** (periodic): Anchor every k items or Δt seconds (configurable). Intermediate protection.
- **Gold** (per-capture): TSA anchor after each evidence item. No inter-anchor vulnerability window. Required for high-assurance forensic applications.

We recommend that implementations targeting forensic or legal use cases deploy Gold-tier anchoring. The 287 ms mean TSA latency is acceptable for typical evidence capture rates (1–10 items per minute for photographs), though video frame-level anchoring at 30 fps would require approximately 8.6 seconds of TSA latency per second of video, which is impractical without batching or local pre-commitment.

Post-quantum migration. As analyzed in Section 5.4, SHA-256 provides approximately 85-bit quantum collision resistance. Migration to SHA-3 or SHAKE256 provides improved Grover resistance without protocol changes. However, we do not claim post-quantum security for the full CPP stack, as the ES256 signature component is classically vulnerable.

Future directions. Combining XOR-CI with zkSNARK-based set membership proofs [30] could enable verifying both completeness and per-item properties without accessing items, addressing both the full-set requirement and the privacy tension. Formal machine-checked proofs in EasyCrypt or CryptoVerif would strengthen regulatory acceptance. Independent open-source implementations of the CPP specification would enable reproducible benchmarking.

8.1 Generalization Beyond Evidence Capture

While this paper focuses on mobile evidence capture within the CPP framework, the Completeness Invariant construction is domain-independent. Any system that must commit to the completeness of an *unordered set* of items and later detect omissions can instantiate the protocol using only a collision-resistant hash function and the four operations of Definition 1.

Applicable domains include: (a) *Regulatory audit trails* for financial transactions (MiFID II, SOX compliance) where auditors must verify that no trades were selectively omitted from a reporting period; (b) *Certificate log completeness* as a lightweight alternative to Merkle-tree consistency proofs when full-tree infrastructure is unavailable; (c) *AI model training data* *manifests* where dataset curators commit to the completeness of training data (relevant to EU AI Act transparency requirements); (d) *Supply chain attestation* where participants commit to the complete set of components or test results.

Scope limitations for generalization: The construction requires that events can be uniquely identified and that a trusted initialization point (session start) exists. It does not naturally extend to *continuous, open-ended* log streams without session boundaries; for such scenarios, epoch-based windowing (committing a new invariant per epoch) is required. Multi-producer environments (where multiple independent entities contribute to the same set) require a trusted aggregator or distributed protocol to maintain the invariant—single-device TEE enforcement does not apply. These extensions are identified as future work.

9 Conclusion

We have presented the Completeness Invariant, a lightweight cryptographic construction for detecting omission attacks on evidence sets. By formalizing omission resistance and proving security bounds under the random oracle model, we provide the first rigorous treatment of a threat that existing tamper-evident mechanisms—hash chains, Merkle trees, digital signatures, and content provenance standards such as C2PA—leave unaddressed.

Our construction builds on the well-established incremental hashing paradigm of Bellare and Micciancio [8], adapting XOR aggregation with bounded set sizes, domain separation, and hardware-backed computation for the specific constraints of mobile evidence capture. We do not claim cryptographic novelty in the XOR primitive itself, but in the formal security definitions, the specific countermeasure combination, and the validated deployment on consumer hardware.

The construction achieves practical efficiency: 0.003 ms XOR aggregation per event, 56-byte constant verification state, and 100% detection rate across 10,000 simulated sessions. Integration within the CPP specification and VeraSnap application demonstrates that forensic-grade completeness guarantees are achievable on consumer smartphones.

We have been explicit about limitations: ROM dependence, bounded set sizes, hardware trust assumptions, the tension between privacy and forensic transparency, and the availability of algebraically stronger alternatives (MuHash, LtHash) for deployments where our constraints do not apply. While related standards such as RFC 4998 (ERS) address hash-tree preservation for ordered archives, no existing specification addresses set completeness verification for unordered collections via lightweight aggregation—the specific gap our work fills.

The Completeness Invariant provides a formally analyzed building block for the emerging ecosystem of digital evidence management, regulatory audit trails, and content provenance architectures.

References

[1] T. Breitinger, B. Carrier, et al. SoK: Timeline based event reconstruction. *Forensic Science International: Digital Investigation*, 53, 2025.

[2] J. Wagner, M. Nissan, and A. Rasin. Database memory forensics: Identifying cache patterns for log verification. In *Proc. DFRWS*, 2023.

[3] D. Ma and G. Tsudik. A new approach to secure logging. *IACR ePrint* 2008/185, 2008.

[4] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, IETF, 2013.

[5] S. Crosby and D. Wallach. Efficient data structures for tamper-evident logging. In *Proc. USENIX Security Symposium*, 2009.

[6] C2PA Technical Specification, version 2.3. Coalition for Content Provenance and Authenticity, 2024.

[7] Y. Liu et al. Scoop: Mitigation of recapture attacks on provenance-based media authentication. Accepted for publication, *USENIX Security Symposium*, 2025.

[8] M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *Proc. EUROCRYPT*, LNCS 1233, Springer, 1997, pp. 163–182.

[9] D. Wagner. A generalized birthday problem. In *Proc. CRYPTO*, LNCS 2442, Springer, 2002, pp. 288–303.

[10] T. Kamimura. Capture Provenance Profile (CPP) specification, version 1.5. VeritasChain Standards Organization, VSO-CPP-SPEC-006, 2026. Also: IETF Internet-Draft `draft-vso-cpp-core-00`.

[11] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 public key infrastructure time-stamp protocol (TSP). RFC 3161, IETF, 2001.

[12] D. Clarke, S. Devadas, M. van Dijk, B. Gassend, and G. E. Suh. Incremental multiset hash functions and their application to memory integrity checking. In *Proc. ASIACRYPT*, LNCS 2894, Springer, 2003, pp. 188–207.

[13] B. Zhang, Y. Sun, J. Zhang, and D. Gu. Polynomial IOPs for memory consistency checks in zero-knowledge virtual machines. In *Proc. ASIACRYPT*, LNCS 14443, Springer, 2023.

[14] S. Setty, J. Thaler, and R. Wahby. Unlocking the lookup singularity with Lasso. In *Proc. EUROCRYPT*, Springer, 2024.

[15] K. Lewi, W. Kim, I. Maykov, and S. Weis. Securing update propagation with homomorphic hashing. *IACR ePrint* 2019/227, 2019. Deployed at Meta, 2023–2024.

[16] M. Zhao, M. A. Shoaib, K. Hoang, and W. Hassan. Nitro: Rethinking tamper-evident logging. Accepted, *Proc. ACM CCS*, 2025.

[17] A. Ahmad, S. Lee, and M. Peinado. HARDLOG: Practical tamper-proof system auditing using a novel audit device. In *Proc. IEEE S&P*, 2022.

[18] R. Paccagnella et al. CUSTOS: Practical tamper-evident auditing of operating systems using trusted execution. In *Proc. NDSS*, 2020.

[19] X. Zhao, K. Zhang, et al. Invisible image watermarks are provably removable using generative AI. In *Proc. NeurIPS*, 2024.

[20] World Privacy Forum. Privacy, identity and trust in C2PA: A technical review. Technical report, 2025.

[21] T. Kamimura. XOR-based completeness verification: Prior art analysis. VeritasChain Technical Report, 2026.

[22] A. Rundgren, B. Jordan, and S. Erdtman. JSON Canonicalization Scheme (JCS). RFC 8785, IETF, 2020.

[23] T. Kamimura. Verifiable AI Provenance (VAP) framework specification, version 1.2. VeritasChain Standards Organization, 2026.

[24] E. Barker and A. Roginsky. Transitioning the use of cryptographic algorithms and key lengths. NIST SP 800-131A Rev. 2, 2019.

[25] T. Gondrom, R. Brandner, and U. Pordesch. Evidence Record Syntax (ERS). RFC 4998, IETF, 2007.

[26] F. Vanini, C. Hargreaves, F. van Beek, and T. Breitinger. Was the clock correct? Exploring timestamp interpretation through time anchors. In *Proc. DFRWS USA*, 2024.

[27] *United States v. Sterlingov*, 719 F. Supp. 3d 65 (D.D.C. 2024).

[28] S. Dziembowski, M. Ebrahimi, and A. Hassanizadeh. VIMz: Private proofs of image manipulation using folding-based zkSNARKs. Accepted, *Proc. PETS*, 2025. IACR ePrint 2024/1063.

[29] A. Datta, Y. Chen, and D. Boneh. VerITAS: Verifying image transformations at scale. Accepted, *Proc. IEEE S&P*, 2025. IACR ePrint 2024/1066.

[30] M. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. In *Proc. Financial Cryptography*, 2021. Journal version: *Designs, Codes and Cryptography*, 2023.

[31] T. Kamimura. Hybrid post-quantum signatures for tamper-evident audit trails: Formal security analysis and design trade-offs. VeritasChain Standards Organization, 2025.

[32] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[33] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th ACM STOC*, pp. 212–219, 1996.

[34] G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN*, pp. 163–169, Springer, 1998.

[35] F. Yuce, M. Shirvanian, et al. TEE.Fail: Extracting secrets from TEE-protected memory via deterministic XTS encryption on DDR5. In *Proc. IEEE S&P*, 2025.

[36] P. Kocher, J. Horn, A. Fogh, D. Genkin, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020. (Originally presented at IEEE S&P 2019.)

[37] B. Schneier and J. Kelsey. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security (TISSEC)*, 2(2):159–176, 1999.

[38] T. Pulls and R. Dahlberg. Balloon: A forward-secure append-only persistent authenticated data structure. In *IACR ePrint* 2015/007, 2015. (Formalized by Pulls–Dahlberg; concept attributed to Google.)

A Publication Status of Cited Works

Following academic best practice, we document the publication status of cited works containing future dates as of February 2026. References marked "accepted" have been peer-reviewed and accepted; final texts may differ from preprints.
All other cited works (2024 and earlier) are published in their respective venues.

Table 8: Publication status of works with 2025+ dates

Reference	Status	Venue	Note
[7] Liu et al.	Accepted	USENIX Sec. 2025	Expected Aug 2025
[16] Zhao et al.	Accepted	ACM CCS 2025	Expected Oct 2025
[19] Zhao, Zhang et al.	Published	NeurIPS 2024	Dec 2024
[28] Dziembowski et al.	Accepted	PETS 2025	ePrint 2024/1063
[29] Datta et al.	Accepted	IEEE S&P 2025	ePrint 2024/1066