**EMERGING DIGITAL FORENSICS**

# Inside the Black Box

## Attestation-Bound Forensic Readiness for Confidential Virtual Machines

**THE FORENSIC ATTESTATION ENVELOPE**

**James Allister Odd, PhD**
j.a.odd@ntrklnk.net
Affiliation: NTRLNK, Oregon, USA
ORCID: 0009-0008-2976-6313

**DOCUMENT TYPE:** Research Paper
**PUBLICATION DATE:** July 2026
**VERSION:** 1.0


**NTRLNK RELATIONSHIP**

This paper was authored by or in association with NTRLNK. The author is responsible for the research methods, analysis, findings, and conclusions presented.

# Abstract

Confidential computing changes a foundational assumption of digital forensics: that an authorized investigator or infrastructure operator can acquire the memory and internal state of a running system. Confidential virtual machines and trusted execution environments protect data in use from privileged software, including a compromised hypervisor or cloud administrator. That protection is valuable, but it restricts conventional memory acquisition, host-based introspection, and retrospective reconstruction. Remote attestation offers partial assistance by allowing a verifier to assess claims about a protected environment. However, attestation ordinarily establishes properties of a platform or workload at a point in time; it does not, by itself, provide a complete, ordered, and independently preserved history of relevant execution.

This paper proposes the **Forensic Attestation Envelope (FAE)**, an attestation-bound evidence architecture for confidential workloads. The FAE links measured launch state, workload identity, policy, cryptographically chained runtime events, protected time evidence, key-lifecycle records, and externally witnessed checkpoints. Its purpose is not to defeat trusted execution environment protections or recreate unrestricted memory access. Instead, it creates forensic continuity: a demonstrable relationship between an attested execution context and a selectively disclosed history of security-relevant events. The paper defines the FAE trust model, evidence schema, capture and verification procedure, threat model, privacy controls, and failure semantics. It also presents an experimental research agenda comparing conventional virtual machines, confidential virtual machines with ordinary logging, and confidential virtual machines equipped with the FAE. The proposal reframes forensic readiness as a design property of confidential computing systems.


**Keywords:** confidential computing; digital forensics; trusted execution environment; remote attestation; cloud forensics; forensic readiness; evidence provenance; confidential virtual machine

# 1. Introduction

Digital forensic practice has historically depended on privileged access. Investigators image storage, capture memory, inspect operating-system artifacts, acquire logs, and correlate evidence across layers. In cloud environments, those activities are already constrained by multitenancy, provider control, distributed storage, jurisdiction, time synchronization, and uneven log availability [1]. Confidential computing introduces a deeper constraint. It protects data while in use by placing workloads inside hardware-backed trusted execution environments (TEEs), reducing the ability of host software, hypervisors, and infrastructure administrators to inspect protected state [2].

The security objective is appropriate: a tenant should not have to trust every privileged component of a cloud platform with plaintext data. Yet the same boundary can frustrate legitimate incident response and forensic investigation. A provider may be unable to inspect guest memory even when a tenant requests

assistance. A tenant may discover that the artifacts needed after an intrusion were never exported from the protected boundary. An investigator may receive logs but lack a defensible way to establish that they originated from the attested workload, were complete for the relevant interval, and were not rewritten before collection.

Remote attestation is often treated as the natural solution. In the Remote ATtestation procedureS architecture, an attester produces evidence, a verifier evaluates it against appraisal policy and reference values, and a relying party uses the resulting attestation result [3]. Entity Attestation Tokens provide a standardized claims format that may convey evidence or attestation results [4]. Contemporary confidential-computing platforms similarly expose signed reports or quotes describing measured state and associated data [5,6]. These mechanisms are essential, but their evidentiary role is narrower than it first appears. An attestation report can support a claim that a particular environment possessed specified properties at a verification event. It does not necessarily establish what the workload did before or after that event, whether relevant events were omitted, or whether exported logs represent a continuous history.

This paper therefore separates two questions:

1. **Execution integrity:** Was the protected environment in an acceptable measured state when it was attested?
2. **Evidential continuity:** Can relevant events be linked, in order and without undetected rewriting, to that attested environment across the period under investigation?

The proposed FAE joins these questions without dissolving their distinction. It binds an attested workload identity and policy to cryptographically chained event commitments, periodic checkpoints witnessed outside the confidential workload, and an evidence package that records both verified claims and known gaps. The approach does not require continuous export of sensitive plaintext or universal surveillance. It aims to give system owners a defensible minimum evidence set while preserving the confidentiality goals that motivated adoption of TEEs.

This contribution is conceptual rather than empirical. No performance or accuracy results are claimed. The paper identifies the gap between attestation and historical reconstruction, specifies an evidence architecture, describes an acquisition and verification workflow, analyzes failure modes, and proposes a reproducible evaluation program.

# 2. Background and Research Gap

## 2.1 Confidential computing and protected execution

Confidential computing protects data in use through a hardware-based, attested TEE [2]. Confidential virtual-machine technologies extend that protection to an entire guest virtual machine. Intel Trust Domain Extensions isolate trust-domain memory and processor state from the host virtual-machine monitor and other platform software [5]. AMD Secure Encrypted Virtualization with Secure Nested Paging similarly

provides memory-confidentiality and integrity protections and supports signed attestation reports requested by a protected guest [6].

These systems change the allocation of trust. Infrastructure operators retain responsibility for physical hardware, availability, orchestration, and portions of the input/output path, but they may no longer be able to inspect a tenant's protected memory. Security monitoring therefore becomes a distributed exercise involving the guest, TEE firmware, attestation services, workload instrumentation, cloud control-plane records, network observations, and tenant-controlled evidence repositories.

## 2.2 Why conventional acquisition becomes insufficient

Traditional forensic guidance emphasizes collection, examination, analysis, reporting, and preservation of integrity [7]. In a confidential workload, several familiar techniques become unavailable or lose semantic precision:

- Hypervisor-based memory acquisition may yield encrypted or inaccessible pages.
- Host introspection may observe scheduling and encrypted input/output without seeing guest semantics.
- Provider snapshots may preserve ciphertext but not the keys or execution context needed for interpretation.
- Guest logs may remain obtainable, but a compromised guest can suppress, alter, or selectively disclose them.
- Emergency debugging interfaces may weaken the TEE security model and create unacceptable standing access.
- Volatile state can disappear before the tenant recognizes an incident or authorizes collection.

Cloud-forensic research has long advocated forensic readiness and logging across multiple architectural layers [1,8]. Confidential computing makes such readiness indispensable. If a protected workload does not emit suitable evidence during execution, later privileged access may not recover it.

## 2.3 Attestation is not a historical transcript

Attestation provides evidence about a target environment for appraisal. Its strength depends on measurement scope, endorsement and reference-value quality, freshness, verifier policy, key protection, and interpretation of claims [3,4]. Even a valid result leaves historical questions unresolved. A launch measurement does not prove that all later runtime activity was benign. A signed token containing a digest does not prove that the underlying events are complete unless the capture path and gap semantics are also established. A verifier's approval does not identify every action taken by an application.

Recent research proposes stronger chains of trust for confidential virtual machines, including virtual trusted platform modules, runtime measurements, and commitments binding workloads to platform context [9-11]. These efforts demonstrate momentum toward richer trust proofs. The remaining forensic problem is to

convert trustworthy claims into a preservable evidence history with explicit custody, disclosure, and failure semantics.

# 3. Research Questions and Propositions

The proposed research program addresses four questions:

**RQ1:** What minimum evidence is necessary to reconstruct security-relevant activity in a confidential virtual machine without exposing unrestricted protected memory?

**RQ2:** How can runtime evidence be cryptographically and semantically bound to an attested workload, policy, and time interval?

**RQ3:** Which omissions, rollbacks, substitutions, and witness failures can be detected, and which must remain explicit limitations?

**RQ4:** How do evidence utility, confidentiality risk, runtime overhead, and investigative effort change under different capture policies?

The framework advances four testable propositions:

- **P1 - Binding:** An attestation-bound checkpoint will provide stronger origin and configuration assurance than guest-signed logs without verified workload identity.
- **P2 - Continuity:** Hash chaining plus independent checkpoint witnessing will improve detection of truncation, reordering, rollback, and retrospective rewriting.
- **P3 - Corroboration:** Reconstruction will be more reliable when protected guest evidence is correlated with independent control-plane, network, key-management, and verifier records.
- **P4 - Proportionality:** Risk-tiered capture and selective disclosure can preserve investigative value with less sensitive-data exposure than indiscriminate content logging.

# 4. The Forensic Attestation Envelope

## 4.1 Definition and design principles

The FAE is a portable evidence package and supporting runtime protocol. It records verifiable claims about a confidential workload, commitments to its security-relevant events, and external observations needed to evaluate those claims. An envelope may cover a session, incident window, checkpoint interval, or workload lifetime.

The design follows six principles:

1. **Attestation-bound identity.** Evidence is bound to a measured workload and verifier decision, not merely a hostname or guest-controlled key.

2. **Continuity with declared gaps.** Ordered commitments make rewriting detectable, while overflow, restart, and loss conditions are represented as evidence rather than hidden.
3. **Independent witnessing.** Checkpoints are time-stamped and preserved beyond the protected guest so that a compromised workload cannot silently rewrite all history.
4. **Layered corroboration.** Guest events are linked to control-plane, network, identity, and key-management records with distinct trust origins.
5. **Data minimization.** Content, metadata, and commitments are separated so that investigators can verify integrity before obtaining sensitive plaintext.
6. **Crypto-agility and interpretability.** Algorithms, schemas, measurement types, verifier policies, and software versions are explicit and replaceable.

## 4.2 Evidence schema

| Evidence domain | Representative fields | Primary forensic purpose | Principal limitation |
|---|---|---|---|
| Envelope identity | Version, workload instance, tenant case, interval, schema | Distinguish and manage evidence packages | Identifiers may reveal tenancy or activity patterns |
| Platform attestation | Quote or report, endorsements, nonce, verifier result, verification time | Bind evidence to a TEE and evaluated configuration | Validity is limited to measured claims and verifier policy |
| Workload identity | Launch digest, image manifest, signer, configuration digest, runtime measurements | Identify code and configuration expected to emit evidence | Dynamic components require additional measurement |
| Capture policy | Profile, disclosure rules, retention, redaction, approved destinations | Explain what should and should not have been recorded | A compliant policy can remain forensically insufficient |
| Event sequence | Type, sequence, time evidence, actor, object, outcome, payload commitment | Reconstruct security-relevant activity | Guest compromise may suppress events before commitment |
| Checkpoint | Prior digest, interval root, event count, gap flags, nonce, signature metadata | Detect reordering, rollback, truncation, and rewriting | Detection depends on timely external preservation |
| External witness | Receipt time, witness identity, checkpoint digest, log position, signature | Establish independent existence and ordering | Witness compromise or outage reduces assurance |
| Key lifecycle | Key identifier, release, rotation, revocation, destruction, correlation ID | Explain signing and decryption authority | Records may not prove how plaintext was used after release |
| Control plane | Creation, policy change, migration, restart, snapshot, administrator action | Relate protected execution to infrastructure events | Provider logs may be incomplete or unavailable |
| Disclosure record | Requested fields, authorization, decrypting party, redactions, export hashes | Preserve custody and proportional access | Cannot prevent misuse after authorized disclosure |

## 4.3 Chained events and attested checkpoints

Each canonical event record, e(i), is committed into an ordered digest chain:

$$d(i) = H(\text{domain} \,||\, \text{sequence}(i) \,||\, d(i-1) \,||\, \text{canonicalize}(e(i)))$$

The domain separator prevents confusion with another protocol. The sequence exposes duplication or reordering. Canonical serialization avoids ambiguous hashes caused by equivalent encodings. Sensitive payloads can be encrypted separately; the chain commits to ciphertext, metadata, or a salted content digest according to policy.

At interval k, the protected evidence service produces a checkpoint:

$$C(k) = \text{Sign}(\text{workload ID} \,||\, \text{policy digest} \,||\, \text{attestation nonce} \,||\, \text{interval} \,||\, \text{event count} \,||\, \text{root digest} \,||\, \text{prior checkpoint} \,||\, \text{gap flags})$$

The evidence key is released only after an approved attestation result and is scoped to workload identity and capture policy. A checkpoint is sent to at least one external witness. The witness returns a signed receipt containing the checkpoint digest, receipt time, and append-only log position or equivalent ordering proof. The envelope preserves the attestation evidence, verifier output, checkpoint, witness receipt, schema, and algorithm identifiers.

This construction does not prove that an uninstrumented action generated an event. It does make later alteration, reordering, cross-workload substitution, and rollback to an earlier witnessed state detectable. Completeness remains a property of instrumentation coverage, secure event delivery, and visible gap reporting.

## 4.4 Trust topology

- **Protected workload:** performs application activity and emits semantic events.
- **Protected evidence service:** canonicalizes events, maintains the chain, records gaps, and creates checkpoints inside the TEE boundary.
- **Attestation verifier:** evaluates platform and workload evidence against reference values and policy.
- **Key-management service:** releases scoped evidence keys only after an acceptable verifier result.
- **External witness:** preserves checkpoint existence and ordering independently of the workload.
- **Evidence custodian:** retains encrypted records, attestation artifacts, receipts, schemas, and custody metadata.
- **Investigator:** validates the envelope, documents assurance and gaps, obtains authorized disclosures, and correlates independent sources.

No single actor is assumed to be infallible. The TEE constrains the host; the witness constrains retrospective guest rewriting; the verifier constrains key release; provider records corroborate tenant claims; and explicit limitations constrain investigative conclusions.

## 4.5 Risk-tiered capture

- **Baseline profile:** launch and runtime measurements, authentication decisions, privilege changes, policy updates, checkpoint status, evidence-service health, restarts, and key events.

- **Elevated profile:** workload lifecycle, administrative commands, protected resource access, network-flow metadata, model or data-set identifiers, and selected application transactions.
- **Incident profile:** short-lived high-resolution telemetry, targeted payload capture, additional checkpoints, and volatile-state summaries enabled under authorized policy.

Transitions between profiles are themselves committed events. A profile change should identify the requesting authority, reason, duration, and expected privacy impact. The incident profile is not a back door to unrestricted memory; it remains bounded by implemented evidence interfaces and applicable authorization.

# 5. Acquisition and Verification Workflow

## Step 1: Preserve the received package

Record acquisition time, source, transfer method, package hash, authorization, and provider export metadata. Preserve the original package before parsing or decryption.

## Step 2: Resolve schemas and trust material

Identify the envelope version, canonicalization rules, event dictionaries, cryptographic algorithms, endorsements, reference values, certificate status, verifier policy, and time sources. Missing historical reference values must be reported; current values should not silently replace those applicable during the incident.

## Step 3: Re-evaluate attestation

Where feasible, validate the original evidence and verifier result. Check nonce binding, freshness assumptions, platform identity, workload measurements, debug state, policy digest, and key scope. Distinguish cryptographic validity from appraisal: a correctly signed report can still describe an unacceptable configuration.

## Step 4: Verify continuity

Recompute event commitments and interval roots, follow prior-checkpoint links, compare event counts, and validate witness receipts. Identify the earliest and latest independently witnessed states. Treat sequence gaps, buffer overflow, evidence-service restart, clock uncertainty, witness outage, and key rotation as first-class findings.

### Step 5: Authorize selective disclosure

Begin with metadata and commitments. Decrypt only fields necessary for the investigative question and permitted by law, contract, and policy. Record each disclosure and hash derived views so later analysis can be related to the preserved package.

### Step 6: Correlate independent evidence

Compare FAE events with identity-provider records, cloud control-plane activity, network telemetry, key-management logs, application clients, orchestration systems, and witness receipts. Agreement increases confidence; disagreement is analytically valuable and must not be normalized away.

### Step 7: Report claim strength and boundaries

For each material conclusion, state whether it is supported by attested configuration, committed guest evidence, external witnessing, independent corroboration, or inference. Report unobserved intervals before the first checkpoint, after the last checkpoint, and during any declared or inferred gap.

## 6. Threat Model and Failure Semantics

### 6.1 Adversaries in scope

The framework considers a malicious or compromised guest application, privileged guest administrator, cloud host or hypervisor, evidence custodian, network intermediary, or external witness. It also considers limited collusion. Physical attacks, hardware-design flaws, side channels, cryptographic breaks, and supply-chain compromise are acknowledged but not fully solved by the FAE.

### 6.2 Manipulations the FAE is intended to expose

- Modification of committed events.
- Reordering or duplication within a chain.
- Rollback to a checkpoint older than the latest independent receipt.
- Substitution of evidence from a different attested workload or policy.
- Removal of a witnessed interval without an observable chain discontinuity.
- Undeclared changes to capture profile or event schema.
- Use of an evidence key outside its attested scope, assuming correct key-service enforcement.

### 6.3 Residual risks

The FAE cannot guarantee semantic completeness when the monitored application bypasses instrumentation, the evidence service never receives an event, or all relevant actors collude. A compromised workload may generate formally valid but misleading events. A malicious host may deny service, delay

checkpoint delivery, or suppress packets. A witness can misstate time unless its clock and ordering service are independently trustworthy. Attestation keys and endorsement infrastructure can fail or be revoked. Vulnerabilities in TEE implementations may invalidate prior assumptions.

Every envelope should distinguish **verified**, **unverified**, **missing**, **not captured by policy**, **capture failure**, and **cryptographically inconsistent** states. "No recorded event" must never be treated automatically as "the event did not occur."

### 6.4 Time and migration

Protected workloads may lack a perfectly trustworthy real-time clock. The FAE therefore separates guest event time, checkpoint sequence, verifier time, witness receipt time, and provider control-plane time. Investigators can establish partial ordering even when absolute time is uncertain. Migration and restart must create boundary events linked to prior and successor workload identities. A platform unable to produce a verifiable transition should begin a new envelope and declare a continuity break.

## 7. Proposed Evaluation Methodology

### 7.1 Experimental conditions

Three configurations should run equivalent services:

1. A conventional virtual machine using standard guest and provider logging.
2. A confidential virtual machine using the same ordinary logging.
3. A confidential virtual machine using the FAE with baseline, elevated, and incident profiles.

At least two TEE implementations should be included to test portability. Workloads, policies, verifier configurations, schemas, attack scripts, and analysis procedures should be versioned and released where licensing permits.

### 7.2 Incident scenarios

The test corpus should include credential theft, privilege escalation, malicious configuration change, data exfiltration, log deletion, guest rollback, evidence-buffer exhaustion, verifier-policy error, witness outage, key rotation, live migration, and compromised application instrumentation. Benign high-load and maintenance activity should be included to test whether investigators confuse operational anomalies with attacks.

### 7.3 Outcome measures

| Dimension | Example measure |
|---|---|
| Reconstruction utility | Proportion of ground-truth actions correctly ordered and attributed |
| Tamper detection | Detection of deletion, modification, reordering, substitution, and rollback |
| Claim calibration | Ability to distinguish verified facts, corroborated claims, gaps, and inference |
| Time resolution | Width of the defensible event-time or ordering interval |
| Investigative effort | Analyst time, external queries, and steps to a supported conclusion |
| Runtime cost | Throughput, latency, CPU, protected-memory use, and checkpoint overhead |
| Storage and transfer | Evidence volume by capture profile and checkpoint frequency |
| Confidentiality exposure | Sensitive fields collected, disclosed, or inferable from metadata |
| Resilience | Evidence retained during outage, restart, or migration |
| Interoperability | Successful verification across platforms, verifiers, and custodians |

Ground truth should be established by a controlled orchestration layer independent of the evidence under evaluation. Analysts should be blinded to incident labels where practical. Multiple analysts should code findings using a shared claim-strength rubric, and inter-rater agreement should be reported.

### 7.4 Ablation and adversarial testing

Ablation studies should remove one component at a time: attestation binding, external witnessing, control-plane corroboration, gap flags, or selective disclosure. Adversarial testing should attempt valid-signature replay, stale attestation use, checkpoint withholding, forked histories, schema ambiguity, clock drift, compromised reference values, and collusion between guest and custodian. Negative results must be reported. An undetectable class of omission should become a stated limitation or deployment requirement.

## 8. Privacy, Legal, and Ethical Considerations

Attestation and forensic telemetry can expose sensitive information. RFC 9334 notes that attestation evidence may include uniquely identifying or personal information [3]. Workload measurements can reveal software versions; event metadata can reveal relationships and behavior; immutable logs can preserve material beyond its legitimate purpose.

The FAE therefore separates integrity verification from content access. Checkpoints may be available to authorized verifiers while event payloads remain tenant-encrypted. Field-level encryption, pseudonymous

identifiers, short retention periods, purpose-bound keys, and role-separated authorization can reduce exposure. Salting or keyed commitments may be necessary where low-entropy values are vulnerable to dictionary attacks.

Technical integrity does not establish legal admissibility. Investigators must document authority, jurisdiction, custody, provider roles, retention obligations, and the meaning of automated attestation decisions. Reference values and verifier policies may be supplied by vendors with interests in the outcome; independent validation and disclosure are important. Human-participant studies involving analyst performance require institutional ethics review before recruitment. The framework should not be used to justify secret expansion of surveillance within protected workloads.

# 9. Discussion and Standardization Priorities

The FAE reframes forensic readiness as a continuity problem. Confidential computing does not eliminate evidence; it changes when evidence must be created, who can observe it, and what claims can be defended. No cryptographic envelope can manufacture events that were never captured.

Interoperability will require more than a common token format. Standards and profiles should define:

- Stable identifiers for workload instance, measured image, evidence schema, and capture policy.
- Canonical event serialization and domain-separated commitment algorithms.
- Checkpoint and witness-receipt formats, including gap and restart semantics.
- Historical reference-value and verifier-policy preservation.
- Migration and successor-envelope linkage.
- Claim-strength vocabulary for forensic reporting.
- Selective-disclosure and custody records.
- Test vectors containing valid, incomplete, rolled-back, forked, and malformed envelopes.

Procurement requirements for confidential-computing services should address forensic readiness before deployment. Incident-response exercises should test loss of the witness, verifier, key service, and network separately. Service-level objectives should specify checkpoint delay and evidence availability, not merely application uptime.

# 10. Limitations

This paper presents a conceptual architecture, not a validated implementation. The cryptographic sketches require formal protocol analysis, careful key-management design, and platform-specific engineering. TEE implementations differ in measurement scope, migration support, debug behavior, endorsement chains, and vulnerability history. The proposed evidence service expands the trusted computing base and may become a performance bottleneck or attack target.

External witnessing strengthens rollback detection but introduces availability, governance, and metadata risks. Multiple witnesses may reduce reliance on one party while increasing complexity. Selective disclosure protects content but can make semantic analysis harder. The framework cannot resolve every admissibility question across jurisdictions and does not replace ordinary evidence from endpoints, identities, networks, providers, or people.

# 11. Conclusion

Confidential virtual machines protect data in use by limiting the privileged access on which many forensic techniques depend. The resulting tension should not be resolved by weakening TEEs or assuming remote attestation is a complete forensic record. Attestation describes evaluated properties; investigation requires a history.

The FAE links those needs through attested workload identity, policy-bound runtime commitments, explicit gap semantics, independent checkpoint witnessing, layered corroboration, and selective disclosure. Its central objective is forensic continuity: the ability to relate an ordered evidence history to a protected execution context while stating precisely where assurance ends. The next step is implementation and adversarial evaluation across multiple confidential-computing platforms.

# References

1. Herman, M., et al. (2020). *NIST IR 8006: NIST Cloud Computing Forensic Science Challenges*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8006

2. National Institute of Standards and Technology. (2026). *NIST IR 8320E ipd: Hardware-Enabled Security: Confidential Computing of Data in Cloud Workloads*. https://doi.org/10.6028/NIST.IR.8320E.ipd

3. Birkholz, H., et al. (2023). *RFC 9334: Remote ATtestation procedureS (RATS) Architecture*. Internet Engineering Task Force. https://www.ietf.org/rfc/rfc9334.html

4. Lundblade, L., Mandyam, G., & O'Donoghue, C. (2025). *RFC 9711: The Entity Attestation Token (EAT)*. Internet Engineering Task Force. https://www.ietf.org/rfc/rfc9711.html

5. Intel Corporation. (n.d.). *Intel Trust Domain Extensions documentation*. https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/documentation.html

6. Advanced Micro Devices. (2020). *SEV-SNP: Strengthening VM isolation with integrity protection and more*. https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

7. Kent, K., Chevalier, S., Grance, T., & Dang, H. (2006). *NIST SP 800-86: Guide to Integrating Forensic Techniques into Incident Response*. https://doi.org/10.6028/NIST.SP.800-86

8. Simou, S., Kalloniatis, C., Kavakli, E., & Gritzalis, S. (2017). Identifying evidence for implementing a cloud forensic analysis framework. In *Advances in Digital Forensics XIII*. https://doi.org/10.1007/978-3-319-67208-3_7

9. Nassar, M., Bassil, R., Chehab, A., & Elhajj, I. H. (2024). *Towards trust proof for secure confidential virtual machines*. arXiv. https://arxiv.org/abs/2405.01030

10. Intel Corporation. (2024). *Runtime integrity measurement and attestation for a trust domain*. https://www.intel.com/content/www/us/en/developer/articles/community/runtime-integrity-measure-and-attest-trust-domain.html

11. Li, M., et al. (2025). *Proof of Cloud: Data center execution assurance for confidential VMs*. arXiv. https://arxiv.org/abs/2510.12469

# Declarations

**Funding:** This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

**Conflicts of interest:** The author declares no conflict of interest.

**Data availability:** No empirical data were generated for this conceptual paper. Evaluation materials, configurations, synthetic incident scenarios, and analysis code should be released with any subsequent experimental study, subject to security and licensing constraints.

**Ethics statement:** This conceptual paper did not involve human participants, personal data, or animal subjects. Any future human-participant evaluation of analyst performance will require appropriate institutional ethics review before recruitment.

**Use of generative AI:** The author used generative AI as an accessibility aid to address writing and proofreading barriers associated with dyslexia. The tool assisted with spelling, grammar, sentence structure, organization, and clarity. The research questions, analysis, interpretations, arguments, and conclusions are the author's own. The author reviewed and revised the content, independently verified sources and factual claims, and accepts full responsibility for the final manuscript. The generative AI tool is not listed as an author. This statement should be adapted to the disclosure policy of the target journal.