Forensic Analysis of Cyber-Physical Systems

Chander Shekhar[1], Vikas Sihag[2*], Gaurav Choudhary[3]

[1]Kumaraguru College of Technology, India

[2]Sardar Patel University of Police, Security and Criminal Justice, India

[3]DTU Compute, Technical University of Denmark (DTU), Denmark

Received: July 25, 2026; Revised: September 30, 2026; Accepted: October 31, 2026; Published: November 15, 2026

Abstract

Cyber-physical incident reconstruction requires evidence linking a physical observation, its transmission and the responses of independently administered services. Heterogeneous records, ambiguous event identifiers, volatile storage and separate trust domains can break this link even when individual messages are delivered successfully. This paper develops an evidence-oriented analysis of an existing Node-RED smartwatch-alert workflow, in which an aggregator, watch company, hospital and law-enforcement agency exchange messages and assemble event records. The architecture combines MQTT communication with proposed participant signatures, hash-linked updates and proof-of-past-log publication derived from established forensic logging research. The contribution is a structured integration of acquisition requirements, event provenance, threat-to-control reasoning, verification test specifications and explicit research-claim boundaries. Six analytical tables connect these elements to the retained implementation artefacts. Inspection supports the presence of configured message paths and illustrative aggregate output; it does not establish cryptographic correctness, distributed consensus, complete evidence capture or measured performance. The analysis shows why retained source records, authentic historical keys and independently preserved references are necessary for meaningful verification, and why even these cannot certify sensor truth. The resulting workflow and verification requirements provide a basis for reproducible evaluation, subject to unresolved implementation, retention, timing and governance constraints.

Keywords: Cyber Physical Systems, Digital Forensics, SCADA, MQTT.

1 Introduction

Cyber-physical systems integrate sensing, computation, communication and physical actuation. Their components exchange operational and sometimes sensitive data to coordinate processes in domains such as healthcare, transportation and critical infrastructure. The coupling of physical and cyber components introduces security challenges that cannot be understood through either domain alone [1]. An incident may affect both information and the physical process, so a forensic investigation must relate messages and system actions to the conditions under which they occurred.

Evidence acquisition and chain-of-custody preservation are difficult when records reside on heterogeneous devices, controllers and organizational servers. A log from one component may explain only part of an event. Changes in device membership, transient data and unavailable services can further complicate reconstruction. Industrial control systems (ICS), including supervisory control and data acquisition (SCADA) installations, also use specialized interfaces that can limit conventional forensic collection. Incidents such as Stuxnet

illustrate the importance of examining interactions between digital instructions and physical operations [1].

The underlying problem is a discontinuity between operational coordination and evidential accountability. Device identifiers can be reused across incidents; clocks and schemas differ across services; acknowledgements can be mistaken for completed actions; and temporary records can disappear before an investigator acquires them. Encryption protects selected content but does not resolve these associations. Likewise, hashing records after their collection cannot reveal an observation that was never recorded. The research objective is therefore to specify what must accompany the existing cross-organizational workflow for its event history to support bounded, reproducible forensic conclusions.

1.1 Scope and Contribution

This article develops an earlier logging prototype study. It retains the reported Node-RED scenario and examines its evidence-handling assumptions against the literature. The original contribution of this article is an evidence-oriented synthesis around that workflow:

(i) a mapping from heterogeneous sources to acquisition requirements;

(ii) an explicit event lifecycle and threat-to-control analysis; and

(iii) a verification specification tied to the evidence actually available.

The six analytical tables operationalize this synthesis. CLASS supplies the PPL concept, Block4Forensic informs multi-party evidence management, and MQTT supplies transport semantics; this article introduces none of those mechanisms. Its contribution is the linkage between those roles, the smartwatch scenario and the conditions under which an investigator may draw a conclusion. No new sensor observations, patient data, workloads or benchmark results are introduced.

The review first considers device, network and cloud evidence. The resulting gap motivates the problem statement and architecture; the implementation account then separates the reported workflow from safeguards needed for deployment. Results are assessed from the preserved artefacts, followed by limitations and future research directions.

2 Literature Review and Research Gap

The review is organized around the evidence lifecycle addressed by the prototype study: acquisition from devices, reconstruction of communications, and preservation across administrative boundaries. It is a focused narrative synthesis, not a systematic review with an exhaustive search protocol. Standards complement research papers by identifying implementation details that the prototype leaves open; citing a standard does not establish implementation compliance.

2.1 Analytical Method and Evidence Basis

The evidence corpus comprises the prototype documentation and the scenario descriptions, diagrams and screenshots retained in this article. The unit of analysis is an event and its transitions between participating services. The analysis first identifies each source and trust boundary, then maps threats to controls and residual uncertainty, and finally specifies verification inputs and expected outcomes. Research papers and standards support the interpretation of mechanisms, rather than substitute for missing implementation evidence. Fig. 1 summarizes this analytical sequence and the evidence categories used throughout.

Four evidence categories are kept distinct. A documented artefact is a visible diagram, node path or output excerpt; a reported mechanism is described in the prototype documentation but not independently executed

here; an analytical deduction follows from stated assumptions; and a proposed requirement or test remains future implementation work. No exported flow, raw execution dataset or rerun is available for this analysis. Accordingly, the assessment is a qualitative design and artefact analysis, not an experimental validation or systematic literature review.

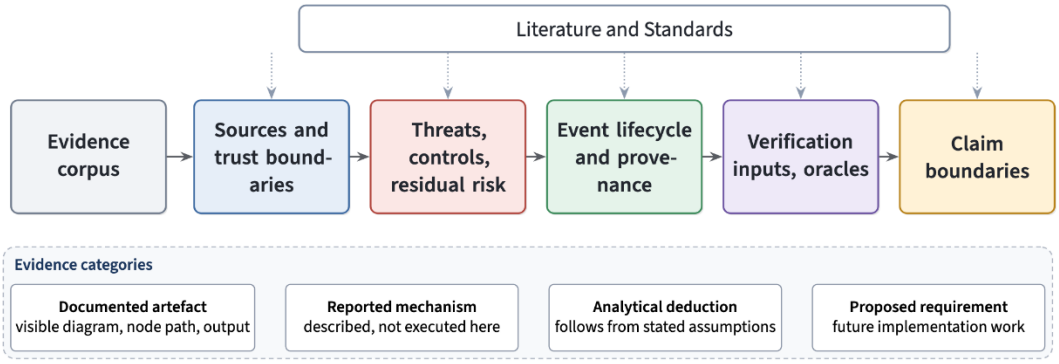


Figure 1. Analytical method. Each stage of the evidence-oriented analysis is operationalized by one analytical table; the unit of analysis is an event and its transitions between participating services. Every statement is classified into one of four evidence categories.

2.2 Control Loops and Heterogeneity

CPS components differ in computing capability, interfaces and operational roles. Closed-loop control describes feedback between sensing and actuation; it does not necessarily mean that a system is disconnected from external networks. A forensic view must account for both feedback paths and communications with external services. MQTT is a publish/subscribe messaging protocol [2]. In this design it transports observations and responses. Application logic performs threshold checks and validation; MQTT itself is not an anomaly detector or a guarantee of real-time deadlines.

2.3 SCADA, PLC and Protocol-Specific Evidence

Yadav and Paul [3] review SCADA architectures, communication protocols and security needs as systems become increasingly interconnected. Their analysis supports treating the controller, communication path and supervisory service as distinct evidence sources. An application-level alert cannot, by itself, establish which controller instructions executed or whether the physical process followed them.

Yau and Chow [4] investigate PLC forensics through control-program logic change detection. Senthivel et al. [5] show how PCCC traffic can expose programmable-logic updates and controller configuration. Kleinmann and Wool [6] model the periodic structure of Siemens S7 communications for intrusion detection and forensic interpretation. These studies address device- or protocol-specific visibility. Their findings motivate retention of controller state and network context alongside application logs; the smartwatch demonstration does not implement their acquisition techniques.

The distinction matters for generalization. An MQTT record can document that a service received an observation, while a PLC acquisition may reveal the program responsible for an actuation. Linking the two requires an explicit association between device identity, program version and event context. NIST guidance on operational technology emphasizes safety, reliability and performance constraints [7]. Therefore, adding forensic collection to a working control loop requires deployment-specific assessment rather than assuming that an IoT messaging demonstration is equivalent to an industrial testbed.

2.4 Membership and Ledger-Based Forensics

Adding or removing a device changes the set of possible message sources and actions. Recording authorized

membership updates can help establish which entities were present during an event. Block4Forensic proposes a permissioned-blockchain framework for connected-vehicle forensics, including membership management and shared and fragmented ledgers [8]. Here, its multi-party evidence-management ideas inform a CPS logging design. The present simulation is a centrally coordinated, hash-linked log workflow; it does not demonstrate the consensus protocol or complete security properties of Block4Forensic. Low overhead is a design objective requiring measurement, rather than an established result of this prototype.

2.5 Cloud Forensics and CLASS

Cloud forensic studies describe difficulties in evidence access, collection and preservation across distributed resources [9–11]. Preparation, identification, collection, analysis and reporting each depend on access to distributed records, preservation of volatile data and a documented custody process. These issues also arise when CPS evidence is retained on cloud or organizational servers.

The Cloud Log Assuring Soundness and Secrecy (CLASS) scheme secures cloud logs using user-public-key encryption and a PPL constructed with Rabin fingerprinting and a Bloom filter [12]. Its relevant ideas are content protection, retained log availability and verification against published proof. The original study evaluates CLASS in OpenStack; those results are not results of the Node-RED simulation reported here. CPS use requires defined identities, cryptographic implementation, retention and permitted key access for user or investigator verification.

2.6 Distributed Logging and Secure Audit Trails

DistLog aggregates, compresses and encrypts IoT logs before authenticating and distributing fragments using a modified information dispersal algorithm [13]. Its contribution concerns evidence availability and integrity when storage nodes or records are threatened. This addresses a different requirement from publishing an integrity reference: a surviving digest cannot reconstruct a destroyed log. The present design adopts no demonstrated DistLog dispersal mechanism, so it cannot inherit that scheme's recovery properties.

Schneier and Kelsey [14] examine secure audit logs on machines that may later be compromised. Their work highlights the need to protect earlier entries even when a logger becomes untrustworthy. Signed Syslog [15] similarly provides a reference for message authentication, sequencing and detection of missing messages. These approaches make clear why a hash chain, signatures, retained content and externally preserved references play different roles. A chain recomputed by a compromised coordinator is not an independent witness to its own history.

2.7 IoT and Network Evidence

Logs and captured communications can help trace an incident and relate it to system behaviour. Packet analysis is an established network-forensic technique [16]. Forensic analysis does not itself prevent an IoT attack, but findings may inform countermeasures. This work focuses on the event records and participant relationships needed for such analysis. NIST forensic guidance [17] separates collection, examination, analysis and reporting; log-management guidance [18] addresses the infrastructure and processes that make records available. Together, they motivate preserving acquisition provenance as well as application messages. NIST's cloud-forensic analysis [19] also places distributed access and organizational dependencies within the investigation problem.

2.8 Synthesis and Remaining Gap

Table 1 compares the roles of the principal research strands. The unresolved task in the prototype study is to connect heterogeneous participant records into an interpretable event history while retaining enough material for later verification. This is an integration and evidence-preservation problem; it is not a claim that existing cryptographic primitives or forensic frameworks are inadequate in every setting.

Table 1. Comparative assessment of existing forensic approaches.

Approach	Established focus	Boundary for this CPS scenario	Role in this paper
PLC/PCCC/S7 studies [4–6]	Controller logic and protocol semantics.	Device/protocol evidence does not itself correlate all organizational responses.	Complementary acquisition context; not implemented here.
Cloud logging [10] and CLASS [12]	Usable cloud logs; CLASS encrypts logs and publishes PPL.	Cloud-log assurance does not by itself validate sensor truth or physical action.	PPL and protected-log concepts are inherited, not new primitives.
Block4Forensic [8]	Permissioned vehicular evidence sharing and membership.	Its ledger and trust design cannot be inferred from a central Node-RED flow.	Informs organizational boundaries and membership requirements.
DistLog [13]	Distributed protection and recovery of IoT log fragments.	Recovery depends on retained fragments and scheme assumptions.	Highlights availability; no dispersal implementation is claimed.
Secure audit logs; Signed Syslog [14, 15]	Authentication, sequence integrity and compromise-aware logging.	A protected digital history cannot establish an unobserved physical fact.	Basis for integrity and omission-analysis requirements.
MQTT [2]	Publish/subscribe delivery with selectable quality of service.	Transport delivery is not organizational completion or archival custody.	Message carrier only; application correlation remains necessary.
Present study	Cross-party smartwatch event workflow and evidence analysis.	Screenshots and illustrative output; no independent execution or benchmarks.	Integrates acquisition, provenance, risks, test oracles and claim boundaries.

3 Problem Statement and Design Objectives

The proposed workflow periodically collects and relates CPS logs to support reconstruction after an incident or partial shutdown. Hash linking and participant signatures are intended to make unauthorized changes detectable when a trustworthy reference and verification credentials remain available. They do not establish the truth of the original sensor reading or prevent every attack.

Three architectural objectives guide the design. First, MQTT carries messages between CPS services, including the programmable logic controller (PLC) layer where applicable. Second, device and service membership changes are recorded through a ledger-oriented approach informed by Block4Forensic [8]. Third, virtual machines associated with distributed services can provide isolation boundaries intended to restrict incident propagation. Isolation effectiveness depends on the deployment; the smartwatch simulation does not evaluate PLC timing or virtual-machine containment.

3.1 Research Question and Evidence Requirements

Research question. How can the reported CPS workflow associate an observation with the responses of participating organizations and preserve a history that an authorized investigator can subsequently examine?

A useful solution needs event correlation, attributable updates, detectable changes relative to a trusted reference, recoverable evidence, and documented handling of incomplete events. These requirements follow from the review; the present artefacts establish only a subset of them.

The scope excludes clinical diagnosis, automated attribution of wrongdoing and proof of sensor accuracy. A recorded confirmation is evidence of a reported response, not proof that an ambulance arrived or that a

physical action succeeded. Likewise, a missing confirmation may reflect interruption, delay or collection failure. Reconstruction must retain these distinctions instead of silently completing an event.

Table 2 translates the reconstruction objective into collection obligations. It distinguishes source content from the provenance required to interpret that content: retaining an aggregate alone loses the independent participant context needed to test omissions or disagreement.

Table 2. Forensic evidence sources and acquisition requirements.

Source	Potential evidence	Acquisition and preservation requirement	Interpretive limit
Watch / sensor	Health payload, device identity, local observation time and cache.	Preserve original bytes, clock context and device state; collect volatile records before cleanup.	A stored reading does not certify physiological accuracy.
Aggregator / MQTT broker	Topics, receipt events, session state and delivery context.	Retain broker configuration, QoS/session settings and capture scope; record collection time.	Missing broker evidence cannot establish that no alert was sent.
Watch-company service	Event versions, append order, response associations and completion decision.	Acquire database versions and application configuration with integrity checks and collector identity.	Coordinator-controlled records are not an independent witness.
Hospital and LEA	Requests, reported confirmations, response bytes and signatures.	Acquire each side independently where possible; retain key identifiers and request-response bindings.	A confirmation asserts a response; physical completion needs corroboration.
Epoch publication / keys	PPL, signature, epoch metadata and historical verification material.	Preserve exact proof bytes, retrieval provenance, authenticated keys and membership history.	A reference is neither a backup nor an independent sensor timestamp.
PLC / network / host, if deployed	Controller logic, packet traces, host and virtual-machine logs.	Use safety-compatible acquisition; preserve program version, capture gaps and configuration.	These sources extend the architecture; absent from the smartwatch evidence.

3.2 Threat Model and Trust Boundaries

The relevant threats are unauthorized alteration of stored records, loss of evidence, replay or duplication of updates, disclosure of sensitive fields, and compromise of a participating service. The coordinator controls aggregation and publication, so a verifier must not rely solely on a current copy supplied by that same coordinator. Participant-held records or independently retained epoch references provide comparison points, subject to their own integrity and availability.

The design assumes authentic verification keys, correctly implemented cryptography and availability of the records being checked. It does not resist collusion among all evidence holders, guarantee logging before a compromised endpoint suppresses an observation, or certify a dishonest sensor’s output. These boundaries are essential when interpreting the conditional properties discussed later.

Table 3 links each failure mode to a verification dependency. The central distinction is between detecting a change relative to a protected reference and establishing that the collected history is complete. The former does not entail the latter.

Table 3. Threat-to-control and residual-risk mapping.

Threat / failure	Control required or proposed	Evidence needed to assess control	Residual risk
Alteration after collection	Sign exact response bytes; hash-link updates; retain an external checkpoint.	Original records, authentic keys and independently held reference.	Compromised signing keys or pre-signing falsehoods remain possible.
Coordinator rewrites history	Compare participant copies and previously witnessed epoch commitments.	Independent receipt records and proof-publication history.	A rewrite before witnessing, or collusion, may escape detection.
Omission / early cleanup	Archive content before deletion; record expected participant set and missing epochs.	Durable archive receipt, inventory and continuity checks.	Never-recorded observations cannot be recovered from hashes.
Replay / concurrent alerts	Bind signed responses to an event and request; make processing idempotent.	Unique event binding, message identity and duplicate decisions.	The prototype uses device indexing; a tested incident-key policy is absent.
Clock disagreement / reordering	Retain source and receipt times, local sequence and explicit append order.	Clock provenance and request-response relations.	A total append order is not necessarily physical causal order.
Disclosure of health/location	Minimize fields; authenticated encryption, access control and controlled key access.	Mode/nonce configuration, authorization records and key policy.	Metadata, authorized misuse and endpoint compromise remain.
Broker / service interruption	Persist pending events; record timeout and recovery; preserve incomplete histories.	Recovery logs, durable state and fault-injection results.	Availability and deadline compliance are unmeasured.
Dishonest or damaged sensor	Corroborate with independent physical or device evidence.	Calibration/state records and independent observations.	An intact digital chain can preserve a false observation.

4 Proposed Methodology and Architecture

The architecture in Fig. 2 connects CPS instances to distributed services and organizational participants, followed by event-log processing and PPL publication. The diagram represents the proposed general structure. Insurance and other servers are optional participants in that structure and are not part of the demonstrated four-party scenario.

The design represents hardware and software components as nodes and their signals as messages. Deployment-specific paths need not form a fully connected network. It assumes correct cryptographic operation, authentic public keys and appropriate verification credentials.

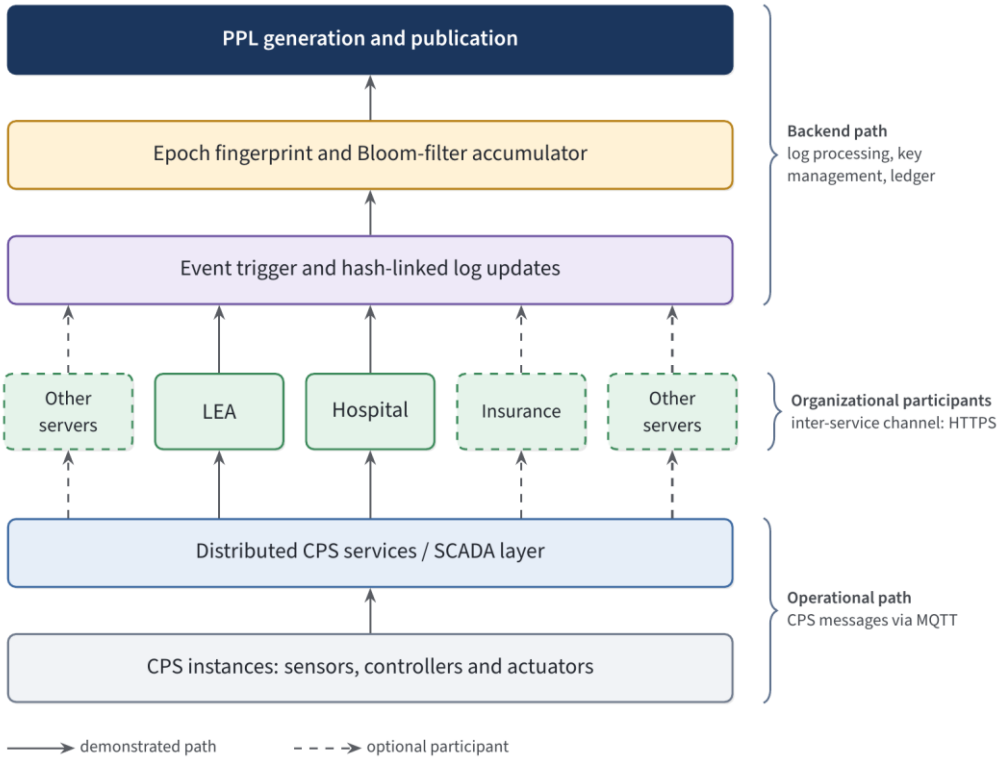


Figure 2. Generalized CPS logging architecture. Dashed paths denote optional participants that are not part of the demonstrated four-party scenario. PPL denotes proof of past log; LEA denotes law-enforcement agency.

Two processing paths are distinguished. The operational path uses MQTT to carry CPS messages. The backend path processes logs, manages keys and coordinates the ledger. HTTPS is the proposed channel for communication between organizational services. This separation aims to decouple operational actions from proof generation; its latency has not been measured.

The backend stores JSON event records and signed hashes at the participating servers. Partial encryption protects selected content only; fields left in plaintext, including identifiers or locations, require separate access controls. A centrally managed private record links the watch company to the hospital and LEA in the simulated scenario. Hash chaining in this record must be distinguished from a distributed blockchain with independently verified consensus.

5 Implementation

Node-RED, a flow-based programming tool [20], is used to illustrate a SCADA-like message workflow. The hypothetical event begins when a smartwatch detects an elevated heartbeat for a configured interval. It sends health information, an International Mobile Equipment Identity (IMEI) and location information through an aggregator to the watch company's server. The company requests support from a hospital and LEA, collects their responses and updates the event record. This is the illustrative scenario already reported in the prototype study, not a new patient case or a validated medical decision system. No new threshold, location, identifier or physiological value is assigned here.

The scenario uses a heartbeat trigger, while its example packet includes a blood-pressure field. The prototype documentation does not explain this mismatch or establish that both measurements were acquired. They are therefore treated as unspecified illustrative health data, without inferring a clinical transformation or a second sensor. No device specification, workload or repeat-count measurement is reported.

5.1 Cryptographic Primitives and Key Distribution

The prototype documentation reports RSA-2048 for inter-organizational exchange and AES-192 for symmetric protection of payloads carried through MQTT. The central node creates and distributes the symmetric key associated with a temporary log. It names alternative key-sharing approaches; Shamir's construction provides a published basis for threshold sharing [21]. Such sharing is considered for distributing sensitive key material. The design does not specify a threshold, shareholder policy or tested recovery procedure, so secret sharing is treated as a design option rather than a verified implementation result. The design also proposes sharing the smartwatch private key. Reconstruction of that signing key by other parties would weaken attribution to the watch alone. A deployment must distinguish decryption-key recovery from signing authority and identify any reconstruction as a custody event.

The available description also omits encryption modes, nonce handling, signature padding, certificate validation and revocation rules. Consequently, specifying key lengths alone does not demonstrate secure encryption or signing. These details must be fixed and tested before deployment. AES is specified in FIPS 197 [22]; RSA encoding and signature schemes are distinguished in PKCS #1 [23]. Neither reference establishes which mode or padding the original nodes used.

For a future implementation, authenticated encryption such as GCM [24] would require explicit nonce management and authentication-tag verification. Signing and encryption keys should have distinct purposes, documented lifetimes and compromise procedures consistent with key-management guidance [25]. TLS [26] can protect a configured transport channel, but does not replace signatures on records retained after that channel closes. These are implementation requirements, not retroactive claims about the simulation.

5.2 Emergency Message Workflow

Fig. 3 summarizes the operational scenario. The aggregator encrypts a packet containing timestamp, imei, blood_pressure and location. The company receives that packet together with a danger or signal code. It forwards location and device identity to the LEA, and the relevant health information, location, identity and code to the hospital.

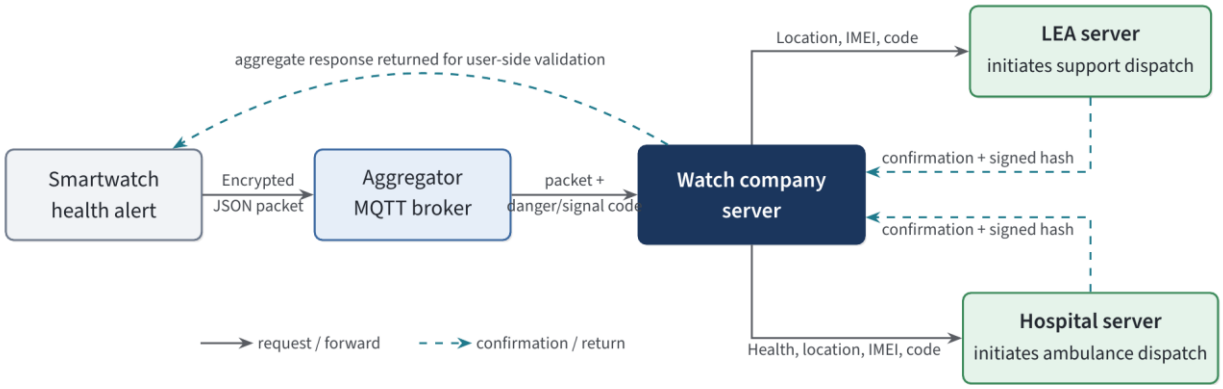


Figure 3. Smartwatch emergency-response workflow. Solid arrows show the forward request path; dashed arrows show participant confirmations returning to the watch company through the backend path in Fig. 4.

Response fields are initially empty. The hospital initiates an ambulance request, while the LEA initiates the corresponding support action. Each organization returns a confirmation code and its signed response hash. The company updates the JSON record and returns the aggregate response to the smartwatch for user-side validation. For the proposed normalized schema, response fields use snake_case names and JSON null denotes an unset value; this is not a claim about the original serialized bytes.

5.3 Backend Log Processing

The processing relationships are shown in Fig. 4. First, the smartwatch stores a temporary packet. The company creates a corresponding record indexed by the watch's IMEI, retaining the health observations and response fields. Second, the hospital and LEA return their signed response hashes. Each confirmed update is appended to the event history, and the updated aggregate is sent to the watch cache. The reported workflow treats the event as completed when the required confirmations have arrived; the exact required set and timeout policy are not specified.

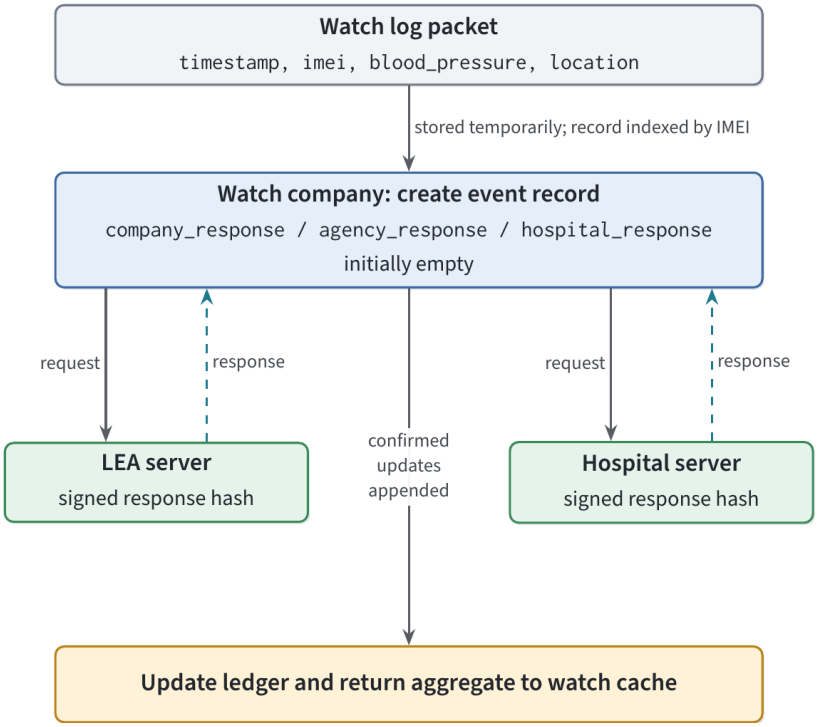


Figure 4. Backend record creation and participant response updates. The central event record coordinates the hospital, LEA and smartwatch cache.

The completed aggregate contains start and end timestamps and the company, agency and hospital response hashes. The original workflow clears temporary frontend records after aggregation. In an evidence-preserving deployment, cleanup must occur only after durable retention and verification of the required records; hash publication alone cannot recover deleted evidence.

Each subsequent update incorporates the previous log digest. A compact expression of the described hash-linking operation is

$$h_i = H(h_{i-1} \parallel r_i) \tag{1}$$

where r_i denotes the serialized update, H the selected hash function and $\parallel$ concatenation. The initial entry uses the chosen initialization convention. The implementation description does not specify canonical JSON serialization or digest initialization, so these must be defined for reproducible cross-party verification. This equation abstracts the intended chaining relationship; the prototype description alternates between a previous log entry and concatenated hashes and does not define an executable encoding. It cannot be used to reproduce the displayed strings without additional specification. Hash linking is reported to continue until event completion or the configured epoch boundary.

5.4 Event Lifecycle and Provenance Requirements

Table 4 separates the application event history from the subsequent custody history. As a proposed completion rule, let Re be the declared set of required responders for event e, and Ve the set whose responses have passed the configured identity, event-binding and signature checks. Completion requires

$$R_e \subseteq V_e \tag{2}$$

and no unresolved conflicting response. The rule must record which policy version defined R_e; it is a specification for follow-up implementation, not a reconstruction of unavailable Node-RED logic.

A timeout closes a waiting interval, not the evidential gap. A late response should create a new version linked to the previous state. Similarly, an epoch can close while an event is still incomplete; its continuation must retain the same event binding across epochs. This prevents operational completion, archival preservation and proof publication from being treated as one interchangeable status.

Table 4. Event lifecycle and evidence provenance.

Stage	Record and transition	Provenance to retain	Failure interpretation
Observation	Watch creates an alert payload.	Source identity, payload bytes, local time and acquisition context.	Sensor truth remains outside signature assurance.
Receipt / routing	Aggregator and company receive and forward the request.	Event/request binding, destination, receipt time and message identity.	Transport acknowledgement is not an action confirmation.
Pending response	Hospital and LEA return separate responses.	Responder, signed content, key identifier and request reference.	Absent response remains pending or timed out, not successful.
Aggregation	Company appends updates and decides completion.	Append index, prior digest, required participant set and decision basis.	Conflicting or late updates must be retained, not overwritten.
Epoch closure	Publisher constructs and signs the PPL.	Epoch scope, algorithm parameters, record inventory and publication receipt.	Unsigned, missing or unwitnessed proof limits the claim.
Preservation / cleanup	Archive evidence before deleting temporary copies.	Content manifest, storage acknowledgement, custody transfer and retention rule.	Hash-only retention cannot support content recovery.
Investigation	Collector exports records; verifier evaluates claims.	Original/working-copy distinction, tool/configuration details and check outcomes.	Missing inputs yield an inconclusive affected claim.

5.5 Record Interpretation and Ordering

The recorded schema contains timestamps, a device identifier, health information, location and participant responses. Field names in the journal are normalized for readability; the original prototype documentation includes spelling inconsistencies and mixes empty-string and null-like response descriptions. A reproducible implementation must specify one representation for an absent response and retain the original bytes used for signing. JSON Canonicalization Scheme (JCS) [27] is a possible serialization basis, but the supplied flows do not establish its use. Likewise, the hash function must be named; FIPS 180-4 [28] supplies standardized algorithms, not evidence of the prototype's algorithm selection.

An IMEI identifies a device in the described scenario, not a unique incident. Repeated alerts therefore require an additional event-correlation rule before deployment. No fabricated event identifiers are inserted into the reported results. A later implementation should also record both observation and receipt times and distinguish a participant's local sequence from the coordinator's append order. Lamport's treatment of distributed event ordering [29] explains why local timestamps alone do not establish a globally reliable causal

order.

MQTT acknowledgements and organizational confirmations must remain distinct. A delivery acknowledgement does not mean that the receiving organization completed its action. Duplicate delivery must not create a second action or erase the first response. The MQTT specification [2] defines delivery semantics; the application still needs event correlation and duplicate handling. The supplied record does not report a tested policy for late, repeated or conflicting confirmations.

5.6 Proof-of-Past-Log Generation

The prototype design describes an epoch-end procedure in which the company orders log hashes, applies Rabin fingerprinting and constructs Bloom-filter membership information as an accumulator entry AE, following the PPL concept in CLASS [12]. The proposed published record is

$$\mathrm{PPL} = (A_E,\ T_E,\ \mathrm{Sign}_{sk}(A_E \parallel T_E)) \tag{3}$$

where T_E, denotes the epoch time and *sk* the publishing party's signing key. Publication is proposed through a REST interface or an RSS feed. Authorized verification relates retained logs to the published accumulator and verifies the signature using the authentic public key.

A Bloom filter supports probabilistic membership queries and can produce false positives [30]. Rabin fingerprints and Bloom membership are not, by themselves, a collision-resistant security proof. The scheme therefore requires a defined construction, parameters, protected reference records and signature checks. A positive Bloom-filter lookup indicates possible membership, not a unique authenticated record or its position. Ordering must come from retained sequence information and chain verification. Without the filter size, number of inserted items and hash/fingerprint configuration, no false-positive probability can be assigned to the displayed output. The example output includes a proof-like hexadecimal string beginning ec664e889ed6c1b2...; it is illustrative output, not a tested security claim.

5.7 Verification and Evidence Preservation

A verifier needs more than a displayed proof string. The evidence package should retain the event records, participant responses and signatures, serialization rules, authenticated public keys, relevant membership history and the published epoch reference. The verifier can then check signatures, recompute the ordered chain and evaluate the specified accumulator. Any unsupported or missing component should lead to an inconclusive finding for the affected claim, rather than being interpreted as successful verification. This describes the procedure needed to complete the design; it is not a report of executed tests.

A publisher-supplied epoch time is also not an independent timestamp. An external time-stamping service, such as the mechanism described in RFC 3161 [31], could provide additional evidence that a digest existed by a given time. It would not prove when the sensor observation occurred. Independent retention of published proofs and a policy for missing epochs are therefore separate requirements from chronological sorting at the company server.

Evidence preservation must precede temporary-cache cleanup. Original records should remain distinguishable from working copies, with acquisition time, collector identity, transfer history and integrity checks documented. The distinction between application history and investigative chain of custody follows forensic practice [17]: the first describes the event, while the second describes how evidence was handled. Neither a successful signature check nor an intact chain establishes completeness of collection or legal admissibility by itself.

5.8 Verification Decision and Test Specification

The verifier should report a result for each claim, using consistent, inconsistent or inconclusive. "Consistent"

requires the inputs and checks specified for that claim; "inconsistent" requires a demonstrable mismatch against an authenticated reference or rule; "inconclusive" records missing evidence or an undefined check. An unknown verification key, for example, is not the same finding as a failed signature under an authenticated key. No global success flag should suppress such distinctions. Fig. 5 summarizes this per-claim decision logic.

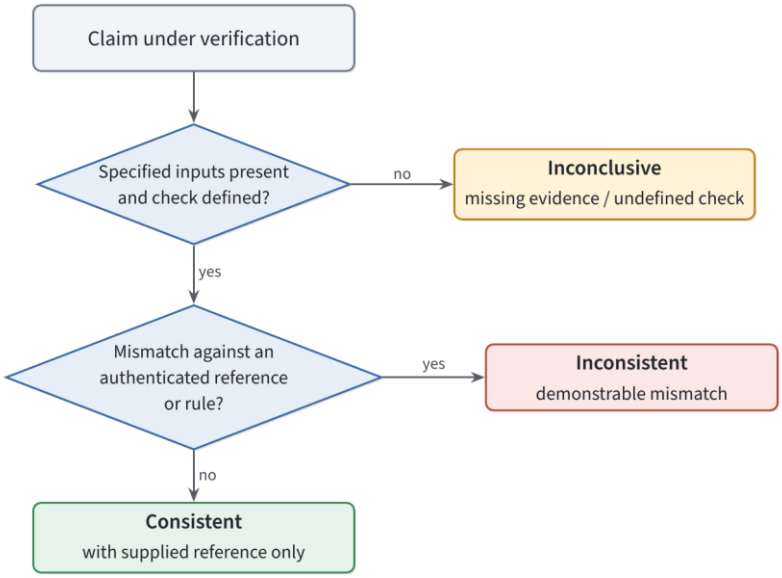


Figure 5. Per-claim verification decision. Each claim receives its own outcome; an unknown verification key leads to inconclusive, whereas a failed signature under an authenticated key leads to inconsistent.

Table 5 specifies controlled manipulations and expected oracles for a future implementation. Each test requires an immutable fixture, preserved configuration, recorded actual outcome and independent review of any discrepancy. Negative tests are essential because a successful-looking output string alone cannot establish that tampering, replay or missing evidence is handled correctly.

Table 5. Forensic verification test cases.

Case	Fixture / manipulation	Expected oracle	Scope of conclusion
V1: intact record	Known original bytes, authentic keys, chain and witnessed checkpoint.	All configured checks succeed; inventory reconciles.	Consistency with the supplied reference only.
V2: altered response	Change a signed field; retain original signature and checkpoint.	Signature or digest check fails for the affected record.	Detects post-signing inconsistency, not its author.
V3: reordered / deleted update	Swap or remove an intermediate entry while holding checkpoint fixed.	Chain or inventory comparison fails.	Requires original ordering and trusted checkpoint.
V4: replay / duplicate	Resubmit a valid response for the same or a different event.	Same-event repeat is recorded without repeating action; wrong-event binding is rejected.	Depends on the proposed event-binding policy.
V5: missing response / epoch	Withhold a required participant response or scheduled proof.	Mark event incomplete or proof unavailable; never return unqualified success.	Does not distinguish attack from outage.
V6: replacement history	Recompute a locally coherent chain after modifying content.	Independent earlier checkpoint or participant comparison exposes divergence.	Without such a reference, outcome is inconclusive.

V7: key / encoding mismatch	Supply unknown key, ambiguous serialization or altered ciphertext tag.	Stop affected check; report missing specification or authentication failure.	Cannot infer historic key validity from current key alone.
V8: archive / recovery	Restart coordinator or clear cache only after an archive acknowledgement.	Retrieve original records and verify manifest before accepting preservation.	Recovery and deletion policy require execution tests.
V9: Bloom membership	Query a controlled member and non-member set under fixed parameters.	Report membership outcomes; do not equate a positive lookup with exact proof.	False-positive behaviour requires measurement.

6 Evidence Assessment and Discussion

The available results are qualitative simulation artefacts from the prototype study. Fig. 6 shows completed-event strings and hash-linked output. Figs. 7–9 provide readable schematics of the Node-RED flows. The original screenshots are retained in Fig. 10, so the schematics are not substituted for execution evidence. The redraws standardize visual labels while the screenshots preserve the original node names. They describe visible processing roles and do not reconstruct unavailable node configurations or a complete runnable flow.

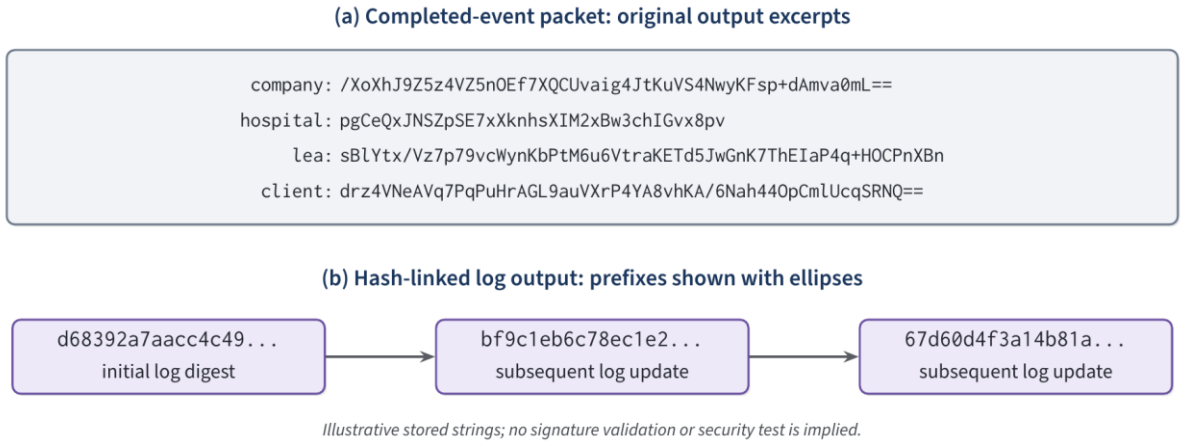


Figure 6. Completed-event output and hash-linked record excerpts from the simulation. Panel (a) retains the original displayed participant strings; panel (b) abbreviates the displayed digest strings for readability.

6.1 What the Artefacts Demonstrate

The screenshots show configured message paths, encryption and decryption node labels, aggregation branches and hash-output routing. They illustrate multi-party event association, without independently verifying cryptographic contents, node correctness or performance. There is no reported baseline comparison, tampering experiment, throughput measurement, verification-time result or scalability test.

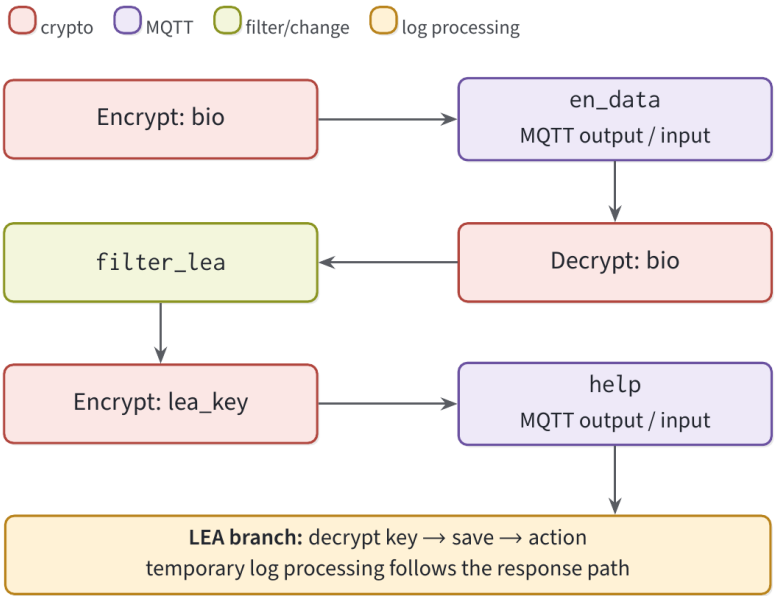


Figure 7. Schematic of the visible CPS message-protection path. Temporary LEA log processing is summarized; the original screenshot is retained in Fig. 10(a).

6.2 Conditional Security and Forensic Properties

Verifiability. An authorized party can compare retained records with a published reference and check participant signatures, provided the keys and reference are authentic. Detecting a changed entry requires preserving the reference outside the adversary's control. A centrally controlled log may otherwise be rewritten together with its hashes.

Confidentiality and privacy. Encryption can restrict access to protected fields when decryption keys are controlled. Hashes do not conceal low-entropy content against guessing, and identifiers or location metadata may remain linkable. The public-key protection described in CLASS [12] should not be assumed to apply automatically to every field in this prototype's partially encrypted JSON record.

Attribution and evidence integrity. Signed response hashes can bind updates to signing keys if keys remain uncompromised. They do not guarantee that the physical event was accurately observed, that every event was logged, or that a record is legally admissible. Evidence acceptance also depends on acquisition procedures, provenance, chain of custody and applicable rules. The artefacts support a logging demonstration, rather than unconditional non-repudiation or court admissibility.

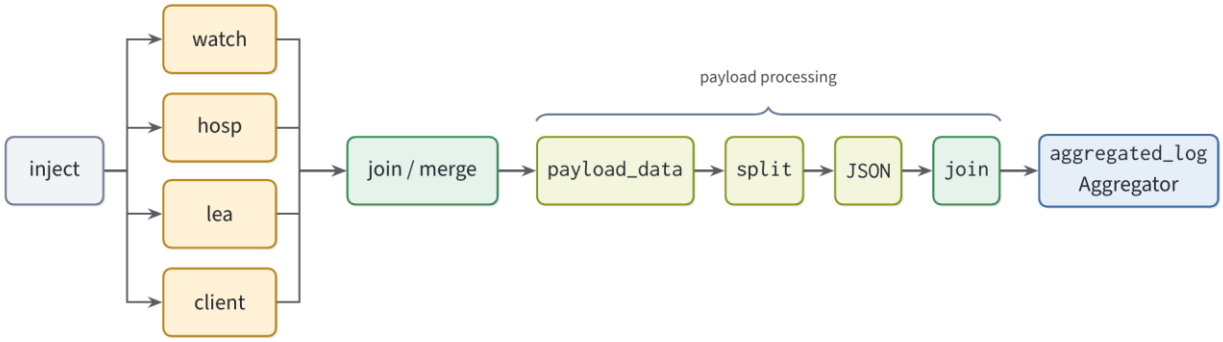


Figure 8. Schematic of participant-log aggregation. The watch, hospital, LEA and client branches feed joining and payload-processing stages.

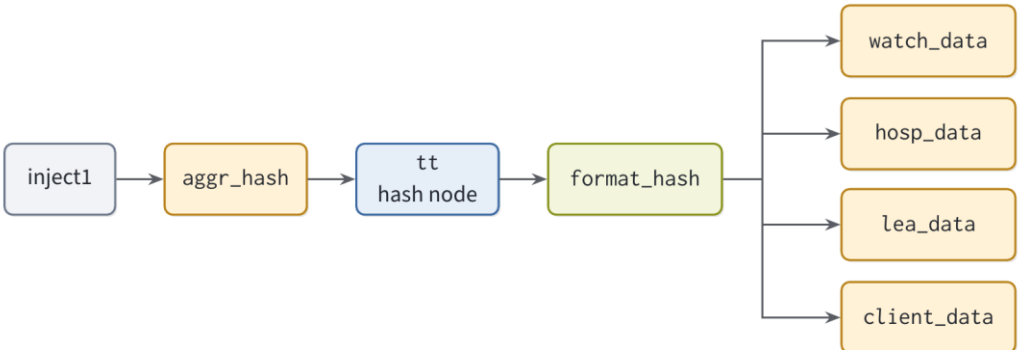
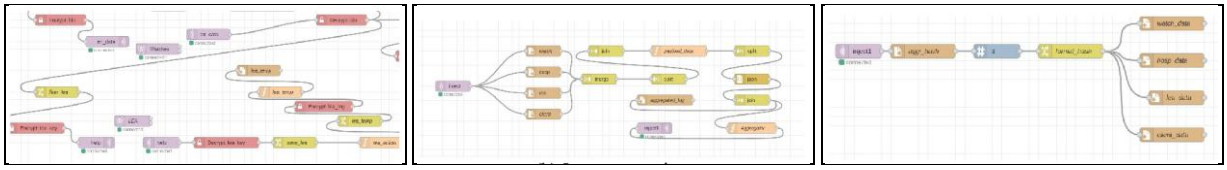


Figure 9. Schematic of aggregate-hash processing and routing to the four participant data outputs. The label tt is retained from the original hash-node screenshot.



(a) CPS message flow (b) Log aggregation (c) Event-log routing

Figure 10. Original Node-RED implementation screenshots from the prototype study, retained in the supplied journal: (a) CPS message flow; (b) log aggregation; (c) event-log routing. Readable companion schematics appear in Figs. 7–9.

6.3 Contribution and Strength of Inference

Table 6 bounds the conclusions at the level supported by the corpus. The original analytical result is the dependency structure across the six tables: a threat can be assessed only when its source evidence survives the lifecycle and the relevant verification inputs exist. For example, a deletion claim links the acquisition inventory in Table 2, the retention control in Table 3, the preservation stage in Table 4 and tests V3, V5 and V8. This is a traceable design argument, not a measured improvement in detection.

The most consequential gap is between aggregation and preservation. A completed organizational exchange can still leave an unverifiable record if the original responses, key history or independent proof are missing. Conversely, a verifiable record may remain operationally incomplete or physically inaccurate. Treating these as separate dimensions is the practical value of the proposed analysis. The workflow offers an integration example; the acquisition and verification specification explains what would be needed to evaluate it scientifically.

Table 6. Research claims versus available evidence.

Claim	Available basis	Defensible interpretation	Evidence still needed
Multi-party event association	Architecture, scenario and aggregation paths in Figs. 2–4 and 8.	Documented workflow and visible configuration.	Flow export and correlated execution trace.
Aggregate / hash-linked output	Illustrative strings in Fig. 6 and prototype description.	Output is displayed; bytes and algorithm cannot be independently reproduced.	Complete records, initialization and encoding.
Authenticity and tamper detection	Signed-hash and PPL mechanism described; no verification trace.	Conditional design property, not an observed security result.	Authenticated keys and tests V1–V7.
Confidentiality	Encryption/decryption node labels and reported key lengths.	Protection intent is documented; mode and	Configuration, key policy and negative tests.

		enforcement remain unknown.	
Evidence completeness / recovery	Temporary-cache cleanup is described.	No proof that all source evidence survives aggregation.	Retention manifest and recovery test V8.
Consensus / membership security	Block4Forensic-inspired architecture; central coordinator.	No demonstrated consensus or replicated membership protocol.	Independent-node implementation and fault tests.
Performance / scalability	No timed workload or baseline data.	No quantitative advantage can be claimed.	Reproducible workload and comparative measurements.
Physical truth / legal acceptance	Illustrative health alert; no physical corroboration or legal assessment.	Neither follows from valid hashes or signatures.	Independent context and case-specific evidence review.

7 Limitations and Future Scope

Jurisdiction. CPS communications and records can cross organizational or national boundaries. A common log schema must accommodate lawful access, retention and privacy requirements.

Physical security. Device damage, environmental conditions and compromised sensors can affect the evidence available to investigators. Context such as terrain or operating conditions may be needed to explain recorded events. Cryptographic integrity of a stored record does not certify the correctness of its physical source.

Scalability and timing. Additional devices increase message volume, storage, key-management effort and proof-generation work. MQTT quality-of-service and retention settings must be chosen alongside operational deadlines; tested settings are not reported. Future evaluation should measure end-to-end delay, storage growth, verification cost and behaviour during service failure. These measurements should separate operational message delivery from backend signing, persistence and proof construction. Device count alone is insufficient: event rate, payload size, participant fan-out and retention period also affect resource demand. No numerical capacity or timing target is inferred from the screenshots.

Reproducibility. The available implementation artefacts include screenshots but no exported Node-RED flow JSON, executable configuration, dependency versions or complete experiment logs. Reproducing the implementation requires those materials, plus a defined event schema, clock policy, cryptographic configuration and controlled tampering tests. These are outstanding evaluation requirements, not results established here.

7.1 Priorities for Follow-up Evaluation

A first priority is a reproducible record format and verification implementation. The preserved flow export, node versions and configuration should identify the exact bytes signed and the construction of the epoch accumulator. Verification should then be tested against altered fields, reordered updates, replayed responses, deleted records and missing epochs. Expected outcomes must distinguish detected inconsistency from unavailable evidence; no detection rate is reported in this article.

A second priority is resilience. Evidence-retention checks should examine broker or participant interruption, coordinator restart and recovery of archived records before any cache deletion. A comparison with local-only logging and with separately retained participant records would help isolate the benefit of aggregation. Such comparisons remain proposed evaluation work, not datasets generated for this revision.

A third priority is privacy and governance. Future implementations should minimize disclosure of location and health information, define who can decrypt each field, and preserve verification material across authorized key rotation. Device-membership changes and revoked credentials must remain interpretable for historical records. The intended use is accountable reconstruction under a defined access policy, rather than unrestricted publication of sensitive CPS data.

8 Conclusion

This paper describes a CPS logging workflow that links operational messages to multi-party response records and periodic proof publication. The smartwatch scenario illustrates the coordination of a watch company, hospital and LEA using Node-RED, while hash-linked records provide a basis for later integrity checking under stated assumptions. The article contributes an explicit acquisition-to-verification argument and six analytical mappings for this scenario. It draws on CLASS and Block4Forensic without establishing new cryptographic primitives, consensus security or comparative superiority. The strongest supported finding is that event correlation, integrity checking and recoverable custody require distinct evidence and controls; none can substitute for the others. Quantitative evaluation, implementation reproducibility and deployment-specific evidence procedures remain necessary, particularly for systems with strict timing, privacy and physical-security requirements.

Abbreviations

Term	Meaning	Term	Meaning
CPS	Cyber-physical system	MQTT	Message Queuing Telemetry Transport
ICS	Industrial control system	QoS	Quality of service
SCADA	Supervisory control and data acquisition	PPL	Proof of past log
PLC	Programmable logic controller	CLASS	Cloud Log Assuring Soundness and Secrecy
OT	Operational technology	JCS	JSON Canonicalization Scheme
LEA	Law-enforcement agency	IMEI	International Mobile Equipment Identity

References

[1] Yaacoub, J. P. A., Salman, O., Noura, H. N., Kaaniche, N., Chehab, A., & Malli, M. (2020). Cyber-physical systems security: Limitations, issues and future trends. Microprocessors and microsystems, 77, 103201.

[2] A. Banks, E. Briggs, K. Borgendale, R. Gupta (Eds.), MQTT Version 5.0, OASIS Standard, 7 March 2019. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

[3] Yadav, G., & Paul, K. (2021). Architecture and security of SCADA systems: A review. International Journal of Critical Infrastructure Protection, 34, 100433.

[4] Yau, K., & Chow, K. P. (2015). PLC forensics based on control program logic change detection. Journal of Digital Forensics, Security and Law, 10(4), 5.

[5] Senthivel, S., Ahmed, I., & Roussev, V. (2017). SCADA network forensics of the PCCC protocol. Digital Investigation, 22, S57-S65.

[6] Kleinmann, A., & Wool, A. (2014). Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics. Journal of Digital Forensics, Security and Law, 9(2), 4.

[7] Stouffer, K., Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., & Thompson, M. (2023). Guide to operational technology (ot) security.

[8] Cebe, M., Erdin, E., Akkaya, K., Aksu, H., & Uluagac, S. (2018). Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles. arXiv preprint arXiv:1802.00561.

[9] Ali, S. A., Memon, S., & Sahito, F. (2018, August). Challenges and solutions in cloud forensics. In Proceedings of the 2018 2nd International Conference on Cloud and Big Data Computing (pp. 6-10).

[10] Pichan, A., Lazarescu, M., & Soh, S. T. (2018). Towards a practical cloud forensics logging framework. Journal of information security and applications, 42, 18-28.
[11] Purnaye, P., & Kulkarni, V. (2022). A comprehensive study of cloud forensics. Archives of Computational Methods in Engineering, 29(1), 33-46.
[12] M. A. M. Ahsan, A. W. A. Wahab, M. Y. I. Idris, S. Khan, E. Bachura, K.-K. R. Choo, CLASS: Cloud log assuring soundness and secrecy scheme for cloud forensics, IEEE Transactions on Sustainable Computing 6 (2) (2021) 184–196.
[13] H. N. Noura, O. Salman, A. Chehab, R. Couturier, DistLog: A distributed logging scheme for IoT forensics, Ad Hoc Networks 98 (2020) 102061.
[14] B. Schneier, J. Kelsey, Secure audit logs to support computer forensics, ACM Transactions on Information and System Security 2 (2) (1999) 159–176.
[15] J. Kelsey, J. Callas, A. Clemm, Signed syslog messages, RFC 5848, May 2010.
[16] L. F. Sikos, Packet analysis for network forensics: A comprehensive survey, Forensic Science International: Digital Investigation 32 (2020) 200892.
[17] K. Kent, S. Chevalier, T. Grance, H. Dang, Guide to Integrating Forensic Techniques into Incident Response, NIST SP 800-86, 2006.
[18] K. Kent, M. Souppaya, Guide to Computer Security Log Management, NIST SP 800-92, 2006.
[19] M. Herman et al., NIST Cloud Computing Forensic Science Challenges, NIST IR 8006, 2020.
[20] OpenJS Foundation, About Node-RED. https://nodered.org/about/ (accessed 23 May 2026).
[21] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.
[22] NIST, Advanced Encryption Standard (AES), FIPS 197, 2001, updated 2023.
[23] K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch, PKCS #1: RSA cryptography specifications version 2.2, RFC 8017, Nov. 2016.
[24] M. Dworkin, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST SP 800-38D, 2007.
[25] E. Barker, Recommendation for Key Management: Part 1 – General, NIST SP 800-57 Part 1 Rev. 5, 2020.
[26] E. Rescorla, The Transport Layer Security (TLS) protocol version 1.3, RFC 8446, Aug. 2018.
[27] A. Rundgren, B. Jordan, S. Erdtman, JSON Canonicalization Scheme (JCS), RFC 8785, June 2020.
[28] NIST, Secure Hash Standard (SHS), FIPS 180-4, Aug. 2015.
[29] Lamport, L. (2019). Time, clocks, and the ordering of events in a distributed system. In Concurrency: the Works of Leslie Lamport (pp. 179-196).
[30] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7) (1970) 422–426.
[31] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, Internet X.509 public key infrastructure time-stamp protocol (TSP), RFC 3161, Aug. 2001.
[32] Singh, P., Choudhary, G., Shandilya, S. K., & Sihag, V. (2022). Enhancing the Security of JSON Web Token Using Signal Protocol and Ratchet System. In Proceedings of Emerging Trends and Technologies on Intelligent Systems: ETTIS 2022 (pp. 387-400). Singapore: Springer Nature Singapore.

Author's Biography



Z Chander Shekhar is an Assistant Professor at Kumaraguru College of Technology, based in Coimbatore, India. He brings 3.8 years of extensive experience in Governance, Risk, and Compliance (GRC), having previously worked at Deloitte Touche Tohmatsu India LLP. His expertise includes security testing and enterprise governance for major clients such as Amazon and Shell. Shekhar holds an M.Tech in Cyber Security from SPUP Jodhpur and a BE in Computer Science and Engineering from Chandigarh University. In his current academic role, he contributes to the field of engineering education and cyber security. His professional background encompasses both industry practice and academic instruction. Relevant topics include information security, enterprise risk management, and governance frameworks. The integration of practical industry experience supports the development of secure and compliant digital systems. He also holds certification as a Master Sound Healing Professional. Chander Shekhar brings a combined perspective of corporate security governance and academic

instruction to his work in cyber security education.



Dr. Vikas Sihag (B.E. ’09, M.Tech ’12, Ph.D. ’22) is an Assistant Professor in the Department of Computer Science Engineering at Sardar Patel University of Police, Security, and Criminal Justice, Jodhpur, India. He earned his Ph.D. from the National Institute of Technology, Raipur, specializing in behavior-based malware detection for Android. Dr Sihag has over a decade of experience in cyber forensics, malware analysis, AI in cybersecurity, and network security. He has published extensively in reputed journals and conferences and has supervised numerous post graduate and doctoral research. Certified as an Ethical Hacker and ISO 27001 Lead Auditor, he actively contributes to academic and professional committees.



Dr. Gaurav Choudhary is an Associate Professor at the Section for Cybersecurity Engineering, Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Copenhagen. He has worked on various projects funded by reputed organizations such as the European Research Funding – Horizon Europe, Industriensfond, National Cluster Digital Technologies and Innovation, DigitalLead, Institute for Information and Communications Technology Promotion (IITP), National Research Foundation of Korea (NRF), and the Air Force Office of Scientific Research (AFOSR), USA. His research interests include Threat Intelligence, IoT and CPS Security, Cyber Security, Vulnerability Assessment, 5G Security, and Drone Security. He has authored and co-authored 90+ peer-reviewed scientific papers in international journals and conference proceedings. He also serves as a Reviewer for various IEEE, ACM, and other journals.