ENHANCING BLOCKCHAIN SCALABILITY IN POWER SYSTEMS THROUGH ULTRA-FAST SECOND-LAYER CHANNEL WITH DETERMINISTIC STATE INTEGRITY

A PREPRINT

Roua Hamila[1]*, **Cesar A. Cazal**[1] , **Md Razaul Haque Subho**[2] , **Rachid Fourati**[3] ,
Ferdinanda Ponci[1] , Thomas Rose[4,5] , Antonello Monti[1,4]

[1]Institute for Automation of Complex Power Systems, RWTH Aachen University, 52074 Aachen, Germany
[2]Advaneo GmbH, Düsseldorf, Germany
[3]EuroSkyPark GmbH, France
[4]Fraunhofer FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
[5]RWTH Aachen University, Templergraben 55, Aachen, Germany

ABSTRACT

Blockchain use in real-time power system automation is still limited because many off-chain protocols improve speed without checking whether exchanged actions remain physically safe for the grid. Some rollup-based approaches also introduce proof-generation cost or dispute delays that are too slow for protection and control tasks. This paper presents a second-layer ultra-fast channel, an off-chain peer-to-peer communication architecture for cyber-physical energy systems. The channel is modeled as a finite-state machine and combines message validation before execution with state-convergence rules after execution to preserve deterministic protocol behavior without waiting for on-chain consensus. To reduce reliance on passive monitoring, the design uses a two-stage dispute process with an internal auditor for local recovery and a smart-contract-based virtual judge for final arbitration. An event-driven prototype separates communication, synchronization, and corrective auditing into parallel execution paths. Experiments on a standard computer platform achieved an average throughput of 546 transactions per second for 128-byte messages, critical-path latency below 4 ms, and a memory requirement of 10.15 KB for standard messages. These results show that safety-aware validation and local recovery can be achieved with low overhead, making the proposed channel a practical candidate for secure real-time grid automation and decentralized energy coordination.

Keywords Blockchain · Power system automation · Second-layer solutions · Deterministic finite automaton (DFA) · Cryptography · Scalability · Semantic gap · Diligence gap

1 Introduction

The digital transformation of modern power systems is driving a shift toward decentralized and highly interactive operating environments in which prosumers, microgrids, storage units, and grid operators exchange data and control signals in real time Aghahadi et al. [2024]. Blockchain technology offers transparency, immutability, and decentralized coordination, which makes it attractive for this setting Nasrinasrabadi et al. [2025]. However, its use in cyber-physical infrastructures remains limited by the blockchain scalability trilemma, which states that decentralization, security, and scalability cannot all be maximized simultaneously Mssassi and Abou El Kalam [2025]. In public blockchain networks, this trade-off appears as confirmation delay and limited throughput, both of which can interfere with time-sensitive grid operation Gajanur et al. [2021]. Recent work has also explored blockchain-based real-time regulation in renewable-energy power systems, which further highlights the importance of timing-aware design in cyber-physical settings Yu

*Corresponding author: roua.hamila@eonerc.rwth-aachen.de

et al. [2024]. In parallel, real-time power-system communication frameworks such as IEC 61850 process bus operate under strict timing requirements Mocanu and Thiriet [2021].

Layer-2 and off-chain mechanisms attempt to reduce these delays by moving transaction execution outside the base chain Gangwal et al. [2023], Alghamdi et al. [2024]. Most state-channel and rollup-based systems focus primarily on syntactic transaction validity, including signature checks, balance constraints, and conditional execution logic Thibault et al. [2022]. In power-system automation, however, cryptographic correctness alone is not sufficient because a message can be properly signed yet still be operationally unsafe, practical protection therefore also requires contextual and physical admissibility checks Mocanu and Thiriet [2021], Alsharif et al. [2025], Sen et al. [2022]. This paper refers to the mismatch between these two requirements as the *semantic gap*.

A second limitation appears in reactive security models such as optimistic rollups, which rely on a dispute period to detect and challenge invalid state transitions Sheng et al. [2024]. Such delayed correction is poorly suited to power systems because unsafe commands may affect the physical process before the dispute is resolved Mocanu and Thiriet [2021]. In this paper, proof of diligence means that validation, traceability, and recovery are enforced inside the channel itself rather than left to passive external observation. Zero-knowledge rollups reduce dependence on fraud challenges, but proving and verification overhead still remain a practical cost in high-speed settings Chaliasos et al. [2024].

Conventional Layer-2 protocols mainly improve off-chain scalability and protocol-level correctness Gangwal et al. [2023]. Secure grid messaging mechanisms mainly protect communication authenticity and integrity Hussain et al. [2023]. In contrast, real-time power-system coordination requires authenticity, temporal freshness, semantic admissibility, and bounded recovery within the same operational loop. This distinction motivates the proposed design, which targets not only fast off-chain exchange, but also accountable validation and reconciliation before unsafe or inconsistent states can affect the protected process.

This paper proposes the Second-Layer Ultra-Fast (SLUF) channel, an off-chain communication architecture for cyber-physical grid automation. The objective is to support millisecond-scale validation and recovery without relying on delayed global consensus. Unlike conventional off-chain designs that mainly improve transaction speed or cryptographic correctness, SLUF is designed to enforce operational safety and bounded recovery within the communication loop itself. To achieve this, the proposed design embeds a finite-state model into the communication logic, applies strict cryptographic, temporal, and semantic validation before execution, and enforces deterministic ledger convergence and dispute resolution through protocol-defined rules. In this way, the SLUF channel addresses both the semantic gap between digital validity and physical admissibility, and the diligence gap between delayed fault detection and the real-time accountability required in power-system operation.

In summary, the main contributions of this work are as follows:

- A safety-aware second-layer channel architecture for cyber-physical grid automation that integrates finite-state control with cryptographic, temporal, and semantic validation, enabling unsafe commands to be rejected before execution.
- A deterministic resilience and accountability framework that combines a formal threat model, state-convergence rules, and a two-tier dispute-resolution mechanism based on an Internal Auditor and a smart-contract-anchored Virtual Judge.
- An event-driven prototype implementation and experimental evaluation demonstrating that the proposed channel achieves millisecond-scale processing, low resource overhead, and effective handling of representative attack scenarios on a standard x86 platform.

2 Theoretical Background

This section summarizes the concepts that directly motivate the proposed channel design for real-time grid automation.

2.1 Limits of Blockchain Scaling for Real-Time Control

Blockchain systems face a well-known trade-off among decentralization, security, and scalability Hafid et al. [2020]. In practice, stronger security and decentralization usually require consensus procedures that add communication and computation delay, which reduces throughput and increases latency Mssassi and Abou El Kalam [2025]. This limitation is especially important in power-system automation, where protection and control functions may operate under very small timing budgets Mocanu and Thiriet [2021]. As a result, direct on-chain execution is generally unsuitable for fast operational decisions Mocanu and Thiriet [2021].

Off-chain and layer-2 mechanisms improve effective throughput, but they do not eliminate this constraint Tortola et al. [2024], Alghamdi et al. [2024]. Instead, they shift part of the coordination burden to external protocols, delayed settlement, or additional trust assumptions Tortola et al. [2024]. For real-time grid applications, the central challenge is therefore not only how to scale transaction processing, but also how to preserve deterministic and operationally safe coordination under strict timing requirements Mocanu and Thiriet [2021].

2.2 Temporary Divergence and Operational Risk

Many distributed systems rely on eventual consistency, where replicas may temporarily diverge and later converge once communication stabilizes Vogels [2009]. This model can improve responsiveness because updates do not always wait for immediate global agreement Junfeng et al. [2022]. However, temporary divergence means that different participants may act on different local views of the system state Vogels [2009].

In cyber-physical environments, this is risky because local decisions may already affect the physical process before reconciliation occurs Nafees et al. [2023]. A particularly important failure mode is equivocation, in which faulty participants present conflicting histories to different replicas Sheng et al. [2021]. In operational settings, this can create split-brain-like divergence before reconciliation completes Vogels [2009], Sheng et al. [2021]. Therefore, eventual convergence alone is not sufficient for real-time grid coordination under strict timing and safety constraints Mocanu and Thiriet [2021], Nafees et al. [2023]. The protocol must also limit unsafe actions during the divergence window Nafees et al. [2023].

2.3 Transport Layer Security (TLS)

Transport Layer Security (TLS) protects peer-to-peer communication by authenticating endpoints, establishing session keys, and encrypting traffic in transit Rescorla [2018]. When properly configured, it reduces the risk of interception, tampering, and man-in-the-middle attacks through authenticated handshakes and certificate-based verification Saint-Andre and Hodges [2011].

However, TLS only secures the communication path Rescorla [2018]. It does not determine whether a received message is semantically correct, temporally fresh in the application context, or consistent with the receiver's local ledger Hussain et al. [2023]. In other words, transport security can confirm that a message arrived through a protected channel, but it cannot confirm that the message is operationally safe to apply Hussain et al. [2023].

2.4 Cyber-Physical Threats and the Semantic Gap

Off-chain channels used in power applications must withstand both external and internal threats, including spoofing, replay, man-in-the-middle behavior, compromised insiders, and unsafe but correctly signed messages Nafees et al. [2023], Usama and Aman [2024], Achaal et al. [2024]. Equivocation can also create conflicting histories across distributed participants Sheng et al. [2021]. More critically, a compromised legitimate node may still send unsafe but cryptographically valid messages Hussain et al. [2023]. False data injection and unsafe command injection attacks illustrate this problem because manipulated measurements or commands may appear authentic even though they are operationally harmful Sen et al. [2022], Usama and Aman [2024].

This creates a semantic gap between cryptographic and operational validity Sen et al. [2022]. A message may be authentic, intact, and correctly delivered, yet still be unsafe for the physical system Sen et al. [2022]. For this reason, a secure channel for real-time grid automation must check not only message origin and integrity, but also whether the message is contextually valid and admissible for the current system state Sen et al. [2022].

3 Proposed System Model and Formalization

This section presents the formal model of the proposed SLUF channel. It defines the architecture, state evolution, validation rules, convergence rules, and dispute-resolution logic in a unified manner. The proposed model is deterministic at the protocol level, meaning that each admissible input yields a uniquely defined validation outcome, state transition, and dispute-resolution path.

3.1 Architectural Overview and Assumptions

The SLUF channel is composed of four elements: the main chain, a smart contract, an off-chain peer-to-peer communication channel, and an arbitration layer. As shown in Fig. 3.1, the main chain provides persistent anchoring, while the smart contract stores participant identities, public keys, session metadata, and dispute parameters. The off-chain channel carries low-latency peer-to-peer data exchange, and the arbitration layer resolves unresolved inconsistencies.

[htbp][width=0.9] Sections/Figures/Second Layer Ultra Fast.png Proposed SLUF architecture

At session establishment, the smart contract binds the session to authenticated peers through identifiers, public keys, wallet references, and network parameters. During runtime, each participant maintains a local ledger that records transmitted requests, acknowledgments, and state changes. These ledgers form the evidentiary basis for synchronization, auditing, and final settlement.

The communication environment is modeled as

$$C = (P, A, K) \tag{1}$$

where P denotes the set of honest peers, A denotes a probabilistic polynomial-time adversary, and K denotes the cryptographic key space.

The model relies on three assumptions. First, transport confidentiality holds such that

$$\Pr[A(E_{\text{trans}}(m)) = 1] \leq \epsilon \tag{2}$$

where $E_{\text{trans}}(\cdot)$ denotes transport-layer encryption, m denotes a transmitted message, and ϵ is negligible. The event $A(E_{\text{trans}}(m)) = 1$ means that the adversary succeeds in extracting useful information from the protected message. Second, the hash function is collision-resistant. Third, each public key is uniquely bound to a registered physical identity. Together, these assumptions ensure that only registered actors can produce admissible protocol events and that tampering becomes detectable.

3.2 Finite-State Channel Model

The SLUF channel is modeled as a finite-state machine

$$M = (Q, \Sigma, \delta, q_0, F) \tag{3}$$

where Q is the finite state set, Σ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 = q_{\text{idle}}$ is the initial state, and $F = \{q_{\text{settle}}\}$ is the accepting state Linz and Rodger [2022]. The lifecycle of this machine is illustrated in Fig. 3.2.

The set of states is defined as

$$Q = \{q_{\text{idle}}, q_{\text{init}}, q_{\text{active}}, q_{\text{sync}}, q_{\text{audit}}, q_{\text{judge}}, q_{\text{settle}}\} \tag{4}$$

where q_{idle} denotes the absence of an active session, q_{init} denotes secure initialization, q_{active} denotes active real-time communication, q_{sync} denotes synchronization, q_{audit} denotes internal dispute resolution, q_{judge} denotes external arbitration, and q_{settle} denotes final settlement.

The accepted input alphabet is

$$\Sigma = \{\sigma_{\text{req}}, \sigma_{\text{init-ack}}, \sigma_{\text{ack}}, \sigma_{\text{data}}, \sigma_{\text{flag}}, \sigma_{\text{match}}, \sigma_{\text{conflict}}\} \tag{5}$$

This restricted alphabet defines the operational language of the channel and excludes malformed or semantically undefined inputs from state progression.

In multi-actor settings, acknowledgments are explicitly modeled as

$$\sigma_{\text{ack}} = (p_{\text{sender}}, h_{\text{ref}}) \tag{6}$$

where p_{sender} identifies the sender of the acknowledgment and h_{ref} is the unique reference to the parent transaction.

The principal state transitions are defined as

$$\delta(q_{\text{idle}}, \sigma_{\text{req}}) = q_{\text{init}} \tag{7}$$
$$\delta(q_{\text{init}}, \sigma_{\text{init-ack}}) = q_{\text{active}} \tag{8}$$
$$\delta(q_{\text{active}}, \sigma_{\text{data}}) = q_{\text{active}} \tag{9}$$
$$\delta(q_{\text{active}}, \sigma_{\text{flag}}) = q_{\text{sync}} \tag{10}$$
$$\delta(q_{\text{sync}}, \sigma_{\text{match}}) = q_{\text{settle}} \tag{11}$$
$$\delta(q_{\text{sync}}, \sigma_{\text{conflict}}) = q_{\text{audit}} \tag{12}$$
$$\delta(q_{\text{audit}}, \sigma_{\text{conflict}}) = q_{\text{judge}} \tag{13}$$

Here, δ maps each admissible state-input pair to exactly one successor state, which is the formal basis of protocol determinism.

[htbp][width=0.9] Sections/Figures/DFA.png SLUF channel deterministic finite automaton (DFA) workflow

3.3 Message Validation and Transaction Integrity

During the active communication phase, every incoming transaction is validated before it is recorded in the local ledger. These message-level checks constitute the *A Priori Rules* of the SLUF channel, because they are enforced before a transaction is recorded or allowed to influence the protected process. In the SLUF model, the A Priori Rules are divided into two classes. The first class, called *Hard Rules*, verifies message authenticity and temporal admissibility. The second class, called *Soft Rules*, verifies semantic admissibility, meaning that the payload must both conform to the restricted command structure and remain operationally safe for the protected process.

Each message is modeled as

$$m = (ID, T, D, SIG) \tag{14}$$

where ID is the sender identity, T is the timestamp, D is the payload, and SIG is the digital signature.

The Hard Rules first enforce cryptographic validity:

$$V_{\text{crypto}}(m) = 1 \iff \text{Verify}(X(m)) \tag{15}$$

where

$$X(m) = (SIG, Z(m), PK_{ID}) \tag{16}$$

and

$$Z(m) = H(ID \parallel T \parallel D \parallel H_{\text{anchor}}) \tag{17}$$

Here, $\text{Verify}(\cdot)$ denotes the signature-verification procedure, $X(m)$ is the verification tuple extracted from message m, $H(\cdot)$ denotes SHA-256, $\parallel$ denotes concatenation, PK_{ID} is the sender public key, and H_{anchor} binds the message to its protocol context. For requests, $H_{\text{anchor}} = H_{\text{meta}}$, where H_{meta} is the cryptographic digest of the session metadata established during channel setup. For replies, $H_{\text{anchor}} = H_{\text{trans}}$, where H_{trans} is the digest of the parent transaction being acknowledged.

The Hard Rules also enforce temporal validity. Messages are rejected if their timestamps are in the future, exceed the allowed time-to-live window, or violate the active session context. Together, these checks ensure that only messages with valid origin, correct protocol anchoring, and admissible timing context can progress through the active state.

The Soft Rules are applied after the Hard Rules are satisfied. First, the payload must belong to the restricted command vocabulary induced by Σ. Second, domain-specific physical constraints are checked before execution. Accordingly, a message may be cryptographically correct and still be rejected if its command structure is not admitted by the protocol or if its physical meaning is unsafe for the system being controlled.

3.4 State Convergence and File Rules

When the end-process flag is raised, the active log file is closed and passed to synchronization, while a new file is opened in parallel to preserve uninterrupted communication. The goal of synchronization is to drive distributed ledgers toward a unique ordered sequence. These synchronization-stage checks constitute the *A Posteriori Rules* of the SLUF channel, because they are applied after active message exchange in order to restore a unique and admissible ledger state. In the SLUF model, the A Posteriori Rules govern file-level convergence through deterministic ordering, causal completeness, and, where required, optional post-aggregation screening for abnormal values.

The first convergence rule is temporal ordering:

$$O(tx_a) < O(tx_b) \iff \Omega(tx_a, tx_b) \tag{18}$$

where

$$\Omega(tx_a, tx_b) = \Omega_T(tx_a, tx_b) \vee \Omega_R(tx_a, tx_b) \tag{19}$$

with

$$\Omega_T(tx_a, tx_b) = (T_a < T_b) \tag{20}$$

and

$$\Omega_R(tx_a, tx_b) = (T_a = T_b) \wedge (\text{Rank}(ID_a) < \text{Rank}(ID_b)) \tag{21}$$

Here, tx_a and tx_b denote two candidate transactions, $O(tx)$ denotes the canonical position of transaction tx in the final ordered ledger, and $\Omega(tx_a, tx_b)$ is the precedence predicate that determines whether tx_a must appear before tx_b. The predicate Ω_T captures timestamp-based precedence, while Ω_R captures tie-breaking precedence when timestamps are equal. The function $\text{Rank}(\cdot)$ is the deterministic ordering rank assigned to each participant at session establishment.

The second convergence rule is causal completeness:

$$S_{\text{final}} = \mathcal{C}(L) \tag{22}$$

where

$$\mathcal{C}(L) = \{(q, r) \in L \times L \mid \Xi(q, r)\} \tag{23}$$

and

$$\Xi(q, r) = (H_{\text{trans}}(q) \subset \text{Payload}(r)) \tag{24}$$

Here, L denotes the local ledger, S_{final} denotes the converged ledger state, $\mathcal{C}(L)$ denotes the causally complete projection of L, q is a parent request, and r is the corresponding reply. The predicate $\Xi(q, r)$ tests whether the parent reference of request q is correctly embedded in the payload of reply r, and $\text{Payload}(r)$ denotes the reply payload field examined during validation.

In multi-actor environments, the parent transaction reference is computed as

$$H_{\text{trans}} = H(Y_{\text{trans}}) \tag{25}$$

where

$$Y_{\text{trans}} = (\text{requestID} \parallel T \parallel D) \tag{26}$$

Here, requestID is the identifier of the parent request, and Y_{trans} is the tuple from which the parent transaction digest is computed. This transaction hash establishes strict lineage and prevents the insertion of orphan acknowledgments or fabricated histories.

For data-centric deployments such as smart meters and charging stations, an additional application-layer screening step may optionally be applied before final file hashing. Given an aggregated dataset D_s, a value x_i is retained only if

$$\mu - k\sigma \leq x_i \leq \mu + k\sigma \tag{27}$$

where μ is the sample mean, σ is the sample standard deviation, and k is the confidence multiplier. Here, x_i denotes the ith observed value in the aggregated dataset D_s. This optional screening step can be understood as an A Posteriori soft filter for measurement-oriented deployments, because it removes clearly abnormal aggregated values before the final ledger digest is computed.

3.5 Dispute Resolution

If synchronization yields identical file hashes across peers, the session proceeds directly to settlement. Otherwise, the protocol triggers a two-tier dispute-resolution mechanism. The overall repair and arbitration logic is summarized in Fig. 3.5.

If the peers compute different digests, that is,

$$H(L_A) \neq H(L_B) \tag{28}$$

the channel enters the audit state. In this expression, L_A and L_B denote the ledgers held by peers A and B, respectively, and $H(L)$ denotes the cryptographic digest of ledger L. In this state, the Internal Auditor merges the candidate ledgers, re-applies the validation and convergence rules, removes duplicates, and identifies the conflicting records.

At this stage, the auditor applies a final category of corrective rules called the *Summary Rules*. Unlike the A Priori Rules, which decide whether an individual message is admissible before execution, and unlike the A Posteriori Rules, which restore file-level convergence after active exchange, the Summary Rules operate only after a conflict has already been detected. Their purpose is to summarize the dispute into a single repaired outcome by isolating the inconsistent subset, preserving the admissible remainder, and producing either a corrected master ledger or an escalation package for external arbitration. In the specific case of equivocation, the Summary Rules are realized through conflict elimination, because the auditor must remove only the mutually inconsistent branch while preserving the rest of the session history.

[htbp][width=0.9] Sections/Figures/Combined auditor and judge.png Overview of the Auditor and Judge processes

In the specific case of equivocation, the repaired state is obtained through conflict elimination:

$$S_{\text{final}} = \mathcal{R}(S_{\text{initial}}, Tx_{\text{conflict}}) \tag{29}$$

where

$$\mathcal{R}(S_{\text{initial}}, Tx_{\text{conflict}}) = S_{\text{initial}} \setminus \mathcal{D}(Tx_{\text{conflict}}) \tag{30}$$

and

$$\mathcal{D}(Tx_{\text{conflict}}) = \{tx \mid tx \in Tx_{\text{conflict}}\} \tag{31}$$

Here, S_{initial} denotes the pre-repair ledger state, Tx_{conflict} denotes the set of conflicting transactions, $\mathcal{R}(\cdot)$ is the repair operator, and $\mathcal{D}(\cdot)$ extracts the inconsistent subset to be removed. Thus, $\mathcal{R}(S_{\text{initial}}, Tx_{\text{conflict}})$ returns the repaired state obtained by removing only the disputed transactions from the initial state. This operation preserves the valid remainder of the session history.

If the Internal Auditor cannot reconstruct a consistent state, the channel escalates to the Virtual Judge. The judge retrieves immutable metadata and rule definitions from the smart contract and evaluates the disputed ledgers using

$$J(L_A, L_B) = \begin{cases} L_A & \text{if Valid}(L_A) \wedge \neg\text{Valid}(L_B) \\ L_B & \text{if } \neg\text{Valid}(L_A) \wedge \text{Valid}(L_B) \\ \emptyset & \text{if } \neg\text{Valid}(L_A) \wedge \neg\text{Valid}(L_B) \end{cases} \tag{32}$$

Here, $J(L_A, L_B)$ is the judge decision function, which maps the two disputed ledgers to a single authoritative outcome. The predicate $\mathrm{Valid}(L)$ returns true only if ledger L satisfies all validation, ordering, and convergence rules. The empty set $\emptyset$ denotes that neither ledger is admissible and that the disputed session must be rejected rather than recovered from either side. The selected ledger becomes the authoritative state for recovery, while the cryptographic evidence can be committed on-chain to trigger penalties against the faulty actor.

3.6 Threat Model, Adversarial Capabilities, and Formal Attack Resistance

To evaluate the security properties of the SLUF channel, the analysis distinguishes among the protected assets, the adversarial capabilities, and the formal success conditions of representative attacks. The protected assets are: (i) authenticity of protocol events, (ii) temporal integrity of the transaction sequence, (iii) convergence of distributed ledgers, and (iv) semantic safety of payloads before execution.

Let the adversary set be

$$\mathcal{A} = \{\mathcal{A}_{\text{ext}}, \mathcal{A}_{\text{peer}}, \mathcal{A}_{\text{ins}}\} \tag{33}$$

where $\mathcal{A}_{\text{ext}}$ denotes an external network adversary, $\mathcal{A}_{\text{peer}}$ denotes a malicious registered peer, and $\mathcal{A}_{\text{ins}}$ denotes a compromised but legitimate internal node.

The adversarial capability function $\Gamma(\cdot)$ is defined as

$$\Gamma(\mathcal{A}_{\text{ext}}) \subseteq \mathcal{G}_{\text{ext}} \tag{34}$$

$$\Gamma(\mathcal{A}_{\text{peer}}) \subseteq \mathcal{G}_{\text{peer}} \tag{35}$$

$$\Gamma(\mathcal{A}_{\text{ins}}) \subseteq \mathcal{G}_{\text{ins}} \tag{36}$$

where

$$\mathcal{G}_{\text{ext}} = \{\text{eavesdrop}, \text{intercept}, \text{relay}, \text{modify}, \text{delay}, \text{replay}, \text{inject}\} \tag{37}$$

$$\mathcal{G}_{\text{peer}} = \{\text{replay}, \text{equivocate}, \text{withhold}, \text{forkHistory}\} \tag{38}$$

$$\mathcal{G}_{\text{ins}} = \{\text{signPayload}, \text{falsifyData}, \text{unsafeCommand}\} \tag{39}$$

Here, $\Gamma(\cdot)$ maps each adversary class to the set of actions it can attempt, while $\mathcal{G}_{\text{ext}}$, $\mathcal{G}_{\text{peer}}$, and $\mathcal{G}_{\text{ins}}$ denote the capability sets of external, malicious-peer, and compromised-insider adversaries, respectively. The terms *intercept*, *relay*, and *modify* capture active man-in-the-middle behavior, in which an external adversary places itself on the communication path and attempts to forward or alter traffic while remaining undetected.

To unify cryptographic, temporal, contextual, and semantic admissibility, the overall acceptance predicate is defined as

$$V_{\text{acc}}(m, L) = \Lambda_1(m, L) \wedge \Lambda_2(m, L) \tag{40}$$

with

$$\Lambda_1(m, L) = V_{\text{crypto}}(m) \wedge V_{\text{time}}(m, L) \tag{41}$$

and

$$\Lambda_2(m, L) = V_{\text{ctx}}(m, L) \wedge V_{\text{sem}}(m) \tag{42}$$

Here, $V_{\text{acc}}(m, L)$ denotes the overall message-acceptance predicate for candidate message m under current ledger state L. The symbol $\Lambda(m, L)$ denotes the same composite acceptance condition, $\Lambda_1(m, L)$ is the cryptographic-temporal admissibility test, and $\Lambda_2(m, L)$ is the contextual-semantic admissibility test. The predicate $V_{\text{time}}(m, L)$ checks freshness and session-time consistency, while $V_{\text{ctx}}(m, L)$ checks whether the message matches the current protocol context, including correct channel phase and parent-child linkage.

The semantic predicate is defined as

$$V_{\text{sem}}(m) = \mathcal{S}_1(D) \wedge \mathcal{S}_2(D) \tag{43}$$

with

$$\mathcal{S}_1(D) = \big(\text{Cmd}(D) \in \Sigma\big) \tag{44}$$

and

$$\mathcal{S}_2(D) = \big(\mathrm{Safe}(D) = 1\big) \tag{45}$$

Here, $V_{\mathrm{sem}}(m)$ denotes the semantic validity of message m, $\mathrm{Cmd}(D)$ extracts the command type encoded in payload D, and $\mathrm{Safe}(D)$ returns 1 only when the payload satisfies the domain-specific physical safety constraints.

For an attack class α, attack success is defined as

$$\mathrm{Succ}_\alpha = 1 \iff \exists(m, L)\ \Upsilon_\alpha(m, L) \tag{46}$$

where

$$\Upsilon_\alpha(m, L) = V_{\mathrm{acc}}(m, L) \wedge \Theta_\alpha(m, L) \tag{47}$$

and

$$\Theta_\alpha(m, L) = \big(\mathrm{Goal}_\alpha(m, L) = 1\big) \tag{48}$$

Here, α denotes an attack class, Succ_α denotes its success event, $\Upsilon_\alpha(m, L)$ is the composite success predicate, and $\Theta_\alpha(m, L)$ captures whether the specific operational objective of attack class α is achieved after the message has passed the acceptance process.

Masquerading and identity spoofing The goal of spoofing is to inject a state-changing message that is accepted as originating from a registered actor. Formally,

$$\mathrm{Goal}_{\mathrm{spoof}}(m, L) = \Phi_{\mathrm{spoof},1}(ID) \wedge \Phi_{\mathrm{spoof},2}(m, L) \tag{49}$$

with

$$\Phi_{\mathrm{spoof},1}(ID) = \big(\mathrm{Unregistered}(ID) = 1\big) \tag{50}$$

and

$$\Phi_{\mathrm{spoof},2}(m, L) = \big(V_{\mathrm{acc}}(m, L) = 1\big) \tag{51}$$

Here, $\mathrm{Goal}_{\mathrm{spoof}}(m, L)$ denotes the spoofing objective, $\Phi_{\mathrm{spoof},1}(ID)$ states that the claimed identity ID is not registered, and $\Phi_{\mathrm{spoof},2}(m, L)$ states that the message is nevertheless accepted under ledger state L. This attack fails because Eq. (15) requires a valid signature under the public key PK_{ID} and the correct protocol anchor. An external actor without the corresponding private key cannot satisfy $V_{\mathrm{crypto}}(m) = 1$.

Replay attacks The goal of replay is to re-submit an old message such that it is accepted as a new transition. Formally,

$$\mathrm{Goal}_{\mathrm{replay}}(m, L) = \Phi_{\mathrm{replay},1}(m) \wedge \Phi_{\mathrm{replay},2}(m, L) \tag{52}$$

with

$$\Phi_{\mathrm{replay},1}(m) = \big(T \le T_{\mathrm{last}}\big) \tag{53}$$

and

$$\Phi_{\mathrm{replay},2}(m, L) = \big(V_{\mathrm{acc}}(m, L) = 1\big) \tag{54}$$

Here, $\mathrm{Goal}_{\mathrm{replay}}(m, L)$ denotes the replay objective, $\Phi_{\mathrm{replay},1}(m)$ states that message m carries a stale timestamp, and T_{last} denotes the timestamp of the most recently accepted event in the current session. The predicate $\Phi_{\mathrm{replay},2}(m, L)$ states that the stale message is nevertheless accepted. This attack fails because temporal admissibility and ordering require monotonic progression of accepted events. By Eq. (18), stale or reordered messages violate the canonical ordering relation and are rejected.

Man-in-the-middle attacks The goal of a man-in-the-middle (MITM) attack is to cause traffic delivered through an attacker-controlled path to be accepted as if it came over the authenticated peer-to-peer channel. Formally,

$$\mathrm{Goal}_{\mathrm{mitm}}(m, L) = \Phi_{\mathrm{mitm},1}(m) \wedge \Phi_{\mathrm{mitm},2}(m, L) \tag{55}$$

with

$$\Phi_{\mathrm{mitm},1}(m) = \big(\mathrm{PathControlled}(m) = 1\big) \tag{56}$$

and

$$\Phi_{\mathrm{mitm},2}(m, L) = \big(V_{\mathrm{acc}}(m, L) = 1\big) \tag{57}$$

Here, $\mathrm{Goal}_{\mathrm{mitm}}(m, L)$ denotes the MITM objective, $\Phi_{\mathrm{mitm},1}(m)$ states that the delivery path of message m is attacker-controlled, and $\Phi_{\mathrm{mitm},2}(m, L)$ states that the message is nevertheless accepted. This attack fails because on-path access alone is not sufficient for acceptance. To succeed, attacker-relayed or attacker-modified traffic must still satisfy the overall acceptance predicate in Eq. (40). In particular, authenticated transport establishment and message-level cryptographic validation prevent an external adversary from being accepted as a registered peer without valid credentials, while any altered message must still satisfy temporal, contextual, and semantic admissibility.

Equivocation and split-brain behavior The goal of equivocation is to make two conflicting histories survive into the committed final state. Let L_A and L_B denote conflicting ledgers produced from the same logical session. Then

$$\mathrm{Goal}_{\mathrm{eq}}(L_A, L_B) = \Phi_{\mathrm{eq}}(L_A, L_B) \tag{58}$$

where

$$\Phi_{\mathrm{eq}}(L_A, L_B) = \Phi_{\mathrm{eq},1}(L_A, L_B) \wedge \Phi_{\mathrm{eq},2}(L_A) \wedge \Phi_{\mathrm{eq},3}(L_B) \tag{59}$$

with

$$\Phi_{\mathrm{eq},1}(L_A, L_B) = \big(H(L_A) \neq H(L_B)\big) \tag{60}$$

$$\Phi_{\mathrm{eq},2}(L_A) = \big(L_A \neq \emptyset\big) \tag{61}$$

$$\Phi_{\mathrm{eq},3}(L_B) = \big(L_B \neq \emptyset\big) \tag{62}$$

Here, $\Phi_{\mathrm{eq},1}(L_A, L_B)$ states that the two candidate ledgers produce different digests, while $\Phi_{\mathrm{eq},2}(L_A)$ and $\Phi_{\mathrm{eq},3}(L_B)$ state that both conflicting histories remain nonempty. This attack cannot persist because the protocol first applies local conflict removal using Eq. (29). If that does not restore convergence, the arbitration rule in Eq. (32) selects the authoritative ledger or rejects both.

False data injection and unsafe command submission The goal of an FDIA-style attack is to inject a message that is cryptographically valid but operationally unsafe. Formally,

$$\mathrm{Goal}_{\mathrm{fdia}}(m, L) = \Phi_{\mathrm{fdia}}(m, L) \tag{63}$$

where

$$\Phi_{\mathrm{fdia}}(m, L) = \Phi_{\mathrm{fdia},1}(m) \wedge \Phi_{\mathrm{fdia},2}(D) \wedge \Phi_{\mathrm{fdia},3}(m, L) \tag{64}$$

with

$$\Phi_{\mathrm{fdia},1}(m) = \big(V_{\mathrm{crypto}}(m) = 1\big) \tag{65}$$

$$\Phi_{\mathrm{fdia},2}(D) = \big(\mathrm{Safe}(D) = 0\big) \tag{66}$$

$$\Phi_{\mathrm{fdia},3}(m, L) = \big(V_{\mathrm{acc}}(m, L) = 1\big) \tag{67}$$

Here, $\Phi_{\mathrm{fdia},1}(m)$ states that the message is cryptographically authentic, $\Phi_{\mathrm{fdia},2}(D)$ states that the payload is operationally unsafe, and $\Phi_{\mathrm{fdia},3}(m, L)$ states that the unsafe message is nevertheless accepted. This attack fails because overall acceptance requires $V_{\mathrm{sem}}(m) = 1$ in Eq. (43). For data-centric deployments that use aggregated measurements, abnormal values can also be screened through Eq. (27) before final hashing, while parent-child consistency is enforced through Eq. (22).

Accordingly, the security claim of the SLUF channel is not merely that attacks are detected, but that representative attacks either fail at acceptance time or are removed before final settlement.

4 Proposed Methodology

This section explains how the formal model is realized in software. The implementation is designed to preserve low latency while enforcing the validation and recovery rules introduced in Section 3.6. The security claims remain defined by the protocol predicates and convergence rules rather than by software structure alone.

4.1 Core Architecture and Session Establishment

The implementation follows the architecture shown in Fig. 1. At initialization, the system processes block source information, participant identifiers, public keys, wallet references, and network parameters such as IP addresses and ports. These credentials are used by the smart contract to generate session metadata, initialize dispute parameters, and establish the secure off-chain session.

Once the session is established, each peer automatically creates and maintains a local ledger that records all transmitted and received events. This provides traceability, complete historical preservation, and policy-compliant auditability throughout the channel lifecycle.

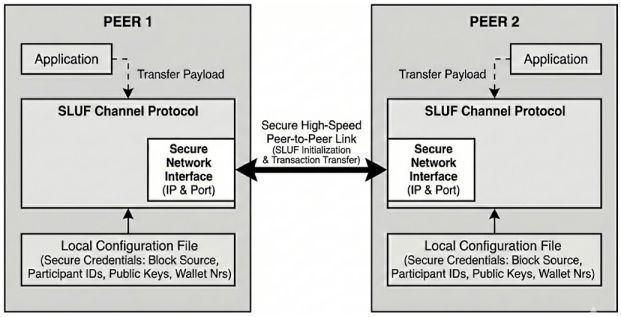

Figure 1: SLUF Channel Peer-to-Peer Initialization and Configuration Topology

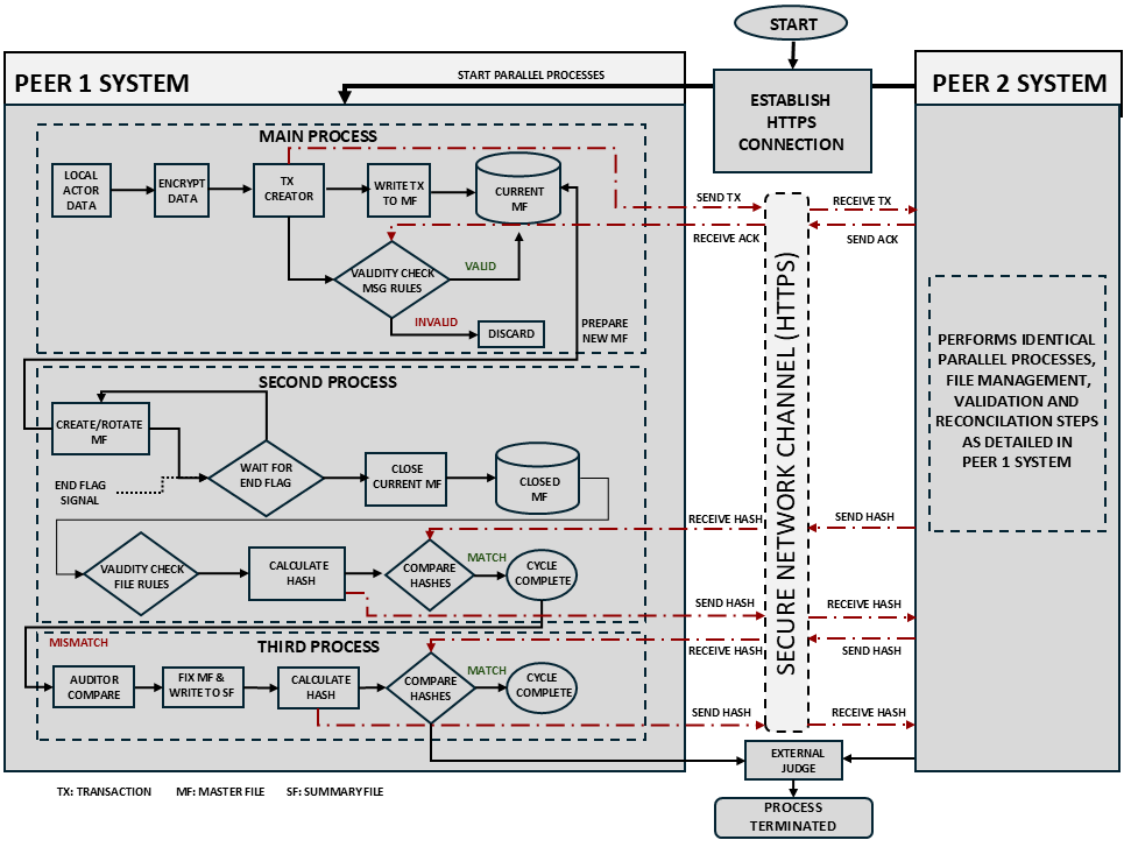

Figure 2: SLUF Architecture and Operational Workflow

4.2 The Execution Model

The prototype is implemented in Node.js as an event-driven design organized around three cooperative execution paths, as illustrated in Fig. 2.

The first path is the *main process*, which realizes the active state q_{active}. It enforces the A Priori Rules defined in Section 3, meaning that it applies the Hard Rules and Soft Rules before any message is admitted to the active ledger. The second path is the *synchronization process*, which realizes q_{sync}. It enforces the A Posteriori Rules defined in Section 3, i.e., it applies file-level convergence logic after active exchange in order to restore a unique ordered ledger state. The third path is the *auditing process*, which realizes q_{audit}. It enforces the Summary Rules defined in Section 3, which means that it resolves detected conflicts by reconstructing a repaired ledger, removing inconsistent subsets, and preparing either the corrected master file or the escalation package for the Virtual Judge.

Operationally, the main process receives local input, validates and signs admissible messages, transmits them through the secure channel, and appends both outgoing and incoming events to the active ledger. When the end-process flag is raised, the synchronization process closes the current log file, opens a fresh file in parallel, applies the file-level consistency rules, and computes the final hash without interrupting active communication. If a hash mismatch is detected, the auditing process receives the candidate ledgers, performs merging and de-duplication, re-validates the records, and produces either a repaired master file or an escalation package for external arbitration.

4.3 Message Structures and Communication Workflow

The implementation distinguishes between two-actor and multi-actor communication patterns, as shown in Fig. 3.

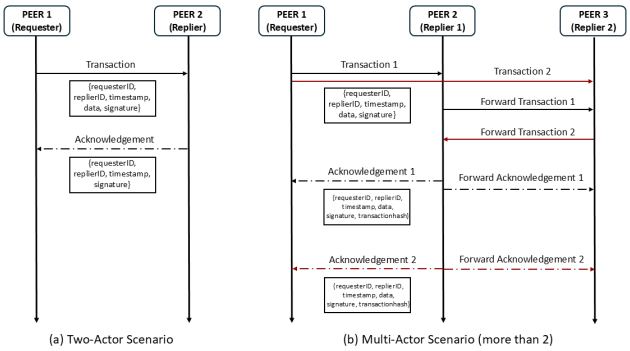

Figure 3: Workflow and data structures of transaction and acknowledgment messages in (a) two-actor and (b) multi-actor scenarios

In a two-actor session, a request contains requester ID, responder ID, timestamp, payload, and signature. The corresponding acknowledgment contains requester ID, responder ID, timestamp, and signature. Because the topology is strictly linear, the parent-child relation can be reconstructed through direct session context and temporal ordering. In contrast, multi-actor communication requires explicit causal references to avoid ambiguity. Therefore, every acknowledgment must include the parent transaction hash H_{trans}, which binds the acknowledgment to its unique parent request. As illustrated in Fig. 4, this creates a strict chain of custody from session metadata to request and from request to reply.

4.4 File Management and Synchronization Workflow

The local ledger is implemented as a rolling log file that stores all messages and responses in real time. This file acts as the immutable evidentiary record of the session. When the end-process flag is triggered, the active file is finalized and passed to the synchronization process, while a new file is opened immediately to preserve continuity. The finalized file is then processed according to the ordering, causal completeness, and semantic filtering rules defined in Section 3. If the resulting hash matches the peer's hash, the file is accepted as the valid session record. Otherwise, it is forwarded to the auditing pipeline for local recovery or external arbitration.

4.5 Security and Cryptographic Compliance

The implementation follows a defense-in-depth strategy that maps directly to the formal assumptions. At the transport layer, peer-to-peer communication is protected through HTTPS/TLS, using asymmetric identity verification and symmetric payload encryption to prevent interception and tampering Rescorla [2018]. At the message layer, digital signatures bind sender identity, timestamp, payload, and protocol anchor into a single verifiable hash. At the protocol

layer, only messages conforming to the restricted command alphabet are admitted for processing. The hierarchical binding used to preserve origin and lineage is summarized in Fig. 4.

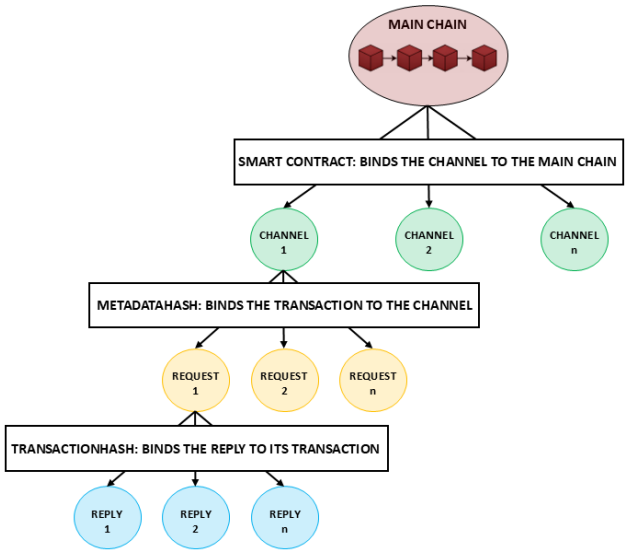

Figure 4: Hierarchical Cryptographic Dependency Graph (Chain of Custody)

This layered design provides three practical protections. First, transport security prevents unauthorized observation and modification during transmission. Second, digital signatures make spoofing and tampering detectable. Third, semantic filtering blocks correctly signed but operationally unsafe commands before execution.

4.6 Arbitration Realization

The arbitration workflow is realized in two stages. The first stage is local and lightweight: the Internal Auditor attempts recovery using only the exchanged ledgers and the predefined convergence rules. The second stage is external and authoritative: if local recovery fails, the Virtual Judge retrieves immutable metadata from the smart contract and applies the decision function to determine the valid ledger. By invoking external arbitration only when necessary, the implementation avoids imposing blockchain latency on normal operation while preserving accountability and recoverability under fault conditions.

5 Results and Discussion

This section evaluates the practical performance of the implemented SLUF channel. The analysis covers communication latency, validation and recovery overhead, resource usage, throughput, and security behavior under controlled adversarial conditions. The goal is to quantify the runtime cost of the prototype and to assess whether its protection mechanisms remain effective within the measured timing bounds.

5.1 Experimental Setup

The experimental testbed was hosted on a machine equipped with an Intel Core i5-7200U CPU (2 cores, 4 threads at 3.10 GHz), 7.64 GB of DDR4 RAM, and a 1 TB SSD, running Ubuntu 24.04.1 LTS. The implementation used Node.js v18.19.1. In addition to performance benchmarking, controlled attack scenarios were executed against the prototype to evaluate protocol behavior under representative adversarial conditions. The detailed platform configuration is summarized in Table 1.

5.2 Communication Latency Analysis

To assess suitability for real-time automation, the communication delay of the SLUF protocol was analyzed over the complete transaction lifecycle. The timing model for the requester–responder exchange is illustrated in Fig. 5. This model separates local protocol processing from network travel time, so that the reported values reflect implementation overhead rather than infrastructure-dependent propagation delay.

Parameter	Details
Operating System	Ubuntu 24.04.1 LTS
Model + Motherboard	ASUSTeK COMPUTER INC. X556UQK
CPU Name	Intel(R) Core(TM) i5-7200U
CPU Topology	1 Processor, 2 Cores, 4 Threads
CPU Frequency	3.10 GHz (base clock)
Geekbench 6 Score relative to the baseline score of 2500	Single-core: 544, Multi-core: 922
RAM	7.64 GB DDR4
Storage	1 TB SSD
Node.js Version	v18.19.1

Table 1: System Specifications

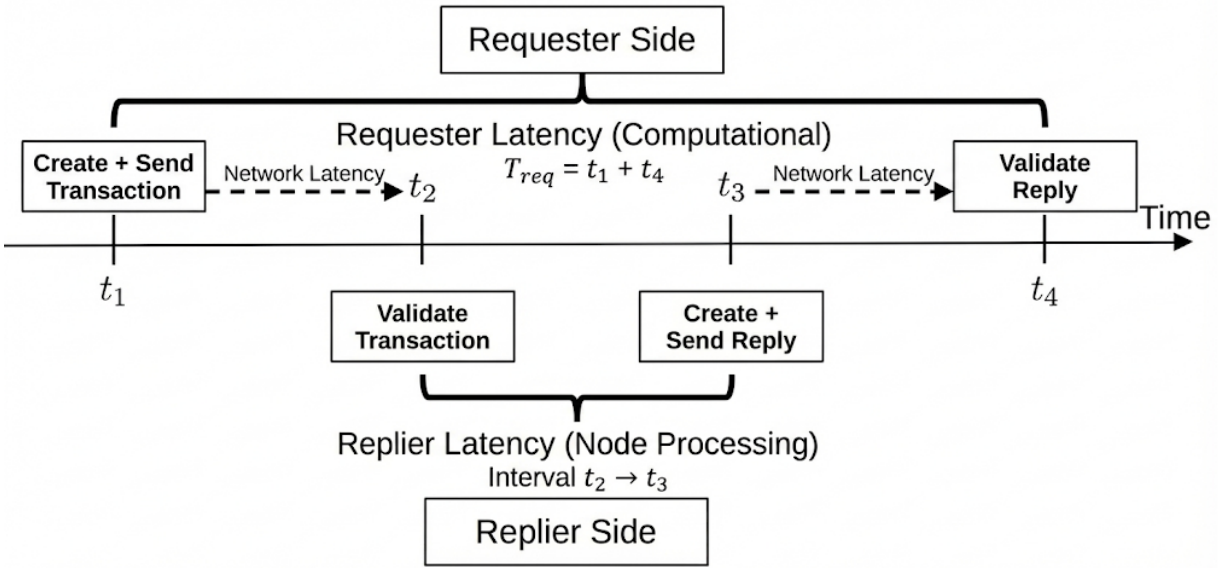

Figure 5: Timing model used to measure requester-side and responder-side computational latency.

The measured latency trends for requester-side and responder-side processing are shown in Fig. 6.

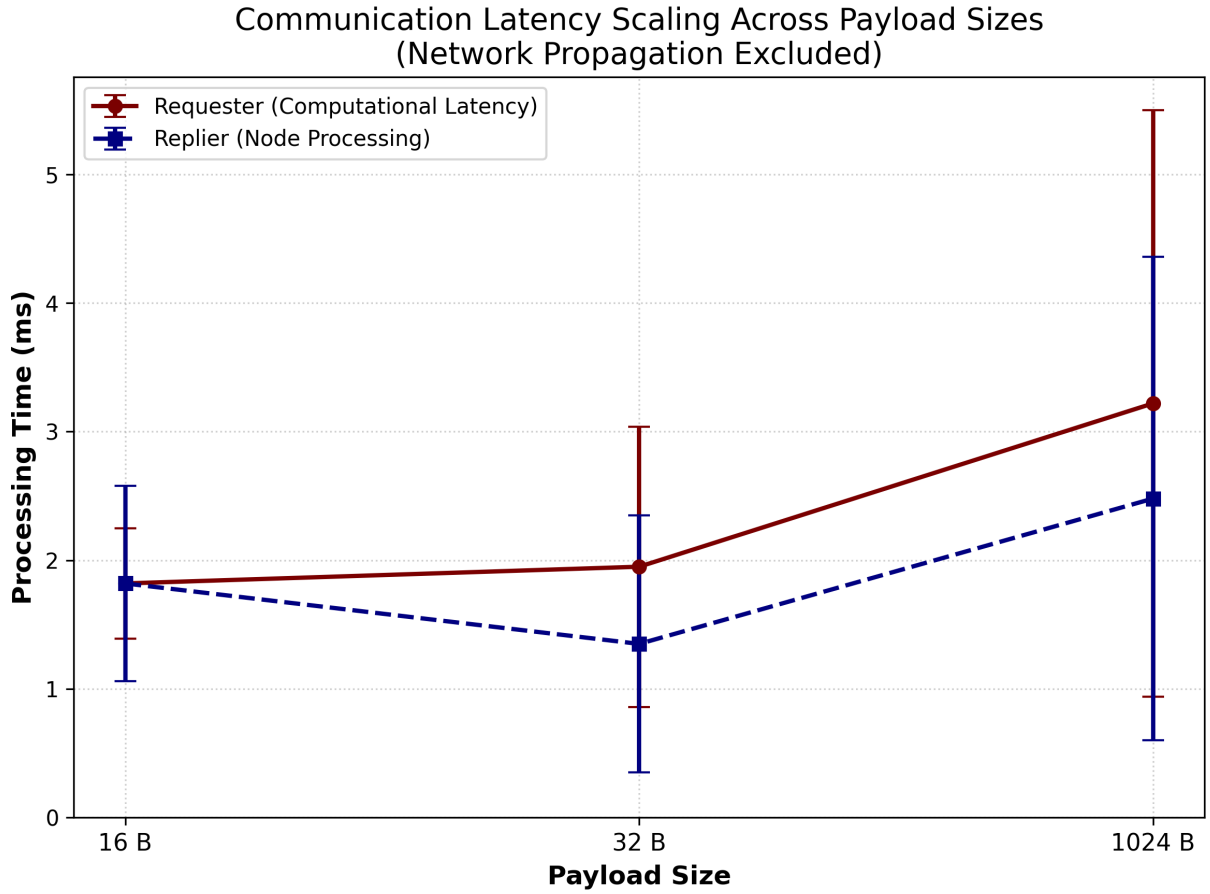

Figure 6: Communication Latency vs. Payload Size: Requester and Responder Processing Overhead

For clarity, let t_1 denote the requester-side instant at which the local request-generation and transmission routine is executed, t_2 the instant at which the responder receives the request, t_3 the instant at which the responder completes processing and transmits the signed acknowledgment, and t_4 the requester-side instant at which the returned acknowledgment has been received and locally validated. Using these reference points, the latency metrics are defined as follows.

The requester-side computational latency is defined as

$$T_{\text{req}} = t_1 + t_4, \tag{68}$$

where T_{req} denotes the total requester-side local processing cost, t_1 captures the local cost of preparing and sending the request, and t_4 captures the local cost of validating the acknowledgment after reception.

The responder-side computational latency is defined as

$$T_{\text{rep}} = t_3 - t_2, \tag{69}$$

where T_{rep} denotes the total responder-side local processing cost, t_2 is the instant at which the responder receives the request, and t_3 is the instant at which the signed acknowledgment is transmitted after local validation and response construction.

These metrics isolate algorithmic latency from network travel time and therefore capture the computational cost of cryptographic verification, semantic checks, state handling, and protocol processing rather than external communication delay between nodes. The results show that the active-path latency remains below the 4 ms threshold relevant to real-time grid protection Mocanu and Thiriet [2021]. This indicates that the required security checks can be performed on the live communication path without exceeding the timing range expected in time-critical cyber-physical applications.

A second observation is that latency growth over the tested payload range remains limited. This suggests that, within the examined operating window, the dominant cost is the fixed per-message control logic, including validation, signing, and acknowledgment handling, rather than payload size alone. In practical terms, the protocol behaves as a lightweight guarded exchange rather than a bandwidth-heavy bulk-transfer mechanism, which is consistent with its intended use in fast coordination and command validation.

5.3 Validation Overhead and Local Recovery

To examine the computational cost of validation and reconciliation, three internal timing metrics were evaluated. The timing windows used for these measurements are illustrated in Fig. 7. The figure separates the active validation phase, the file rollover and synchronization phase, and the internal auditing phase triggered after a hash mismatch.

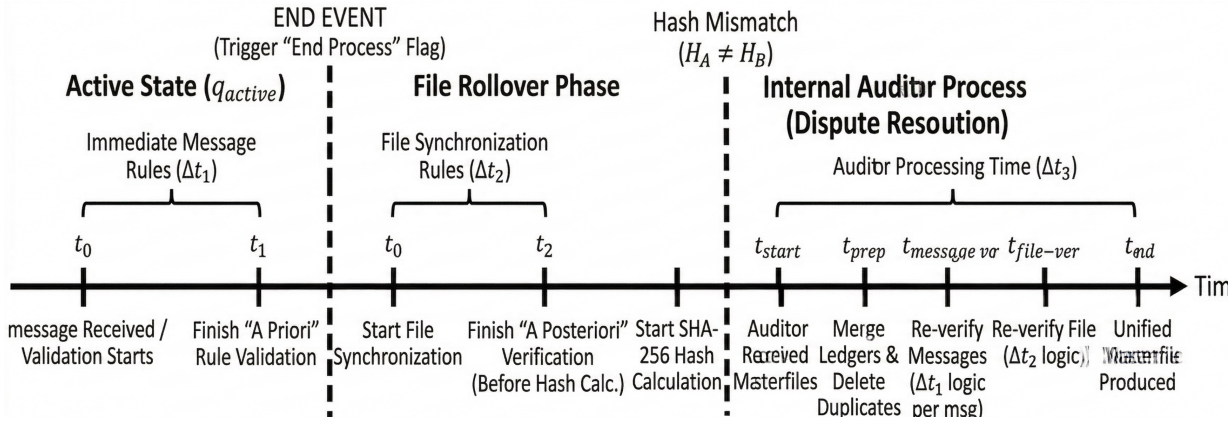

Figure 7: Timing model used to measure active validation, file synchronization, and internal auditing overhead.

The corresponding measured values are shown in Fig. 8.

For clarity, let t_0 denote the instant at which a received message enters the local validation routine. Let t_1 denote the instant at which immediate message-level validation completes, and let t_2 denote the instant at which file-level synchronization processing completes after the rollover event. For the audit path, let t_{start} denote the instant at which the Internal Auditor begins processing the mismatched files, and let t_{end} denote the instant at which the repaired master file or escalation package is produced.

The first metric, τ_{val}, captures the cost of applying immediate message-level rules during active communication:

$$\tau_{\mathrm{val}} = t_1 - t_0. \tag{70}$$

Here, τ_{val} denotes the active-path validation latency, measured from the start of local validation at t_0 to the completion of message-level rule enforcement at t_1.

The second metric, τ_{sync}, measures the overhead of file-level rule application during synchronization:

$$\tau_{\mathrm{sync}} = t_2 - t_0. \tag{71}$$

Here, τ_{sync} denotes the synchronization latency, measured from the same validation entry point at t_0 to the completion of file-level reconciliation and synchronization work at t_2.

The third metric, τ_{audit}, evaluates the processing time of the Internal Auditor during local dispute resolution:

$$\tau_{\mathrm{audit}} = t_{\mathrm{end}} - t_{\mathrm{start}}. \tag{72}$$

Here, τ_{audit} denotes the auditor-side corrective latency, measured from the start of auditor processing at t_{start} to the generation of the repaired master file or escalation package at t_{end}.

For 16-byte messages, immediate validation requires only 0.42 ms, increasing to 1.49 ms for 1024-byte payloads. This confirms that the most frequent protection layer remains sufficiently lightweight for proactive enforcement before execution.

The synchronization metric τ_{sync} ranges from 91.69 ms to 130.95 ms. Although this cost is much larger than the active-state validation cost, it is incurred during file rollover rather than on the primary real-time path. The results therefore indicate that batch consistency checks can be applied without interrupting active communication.

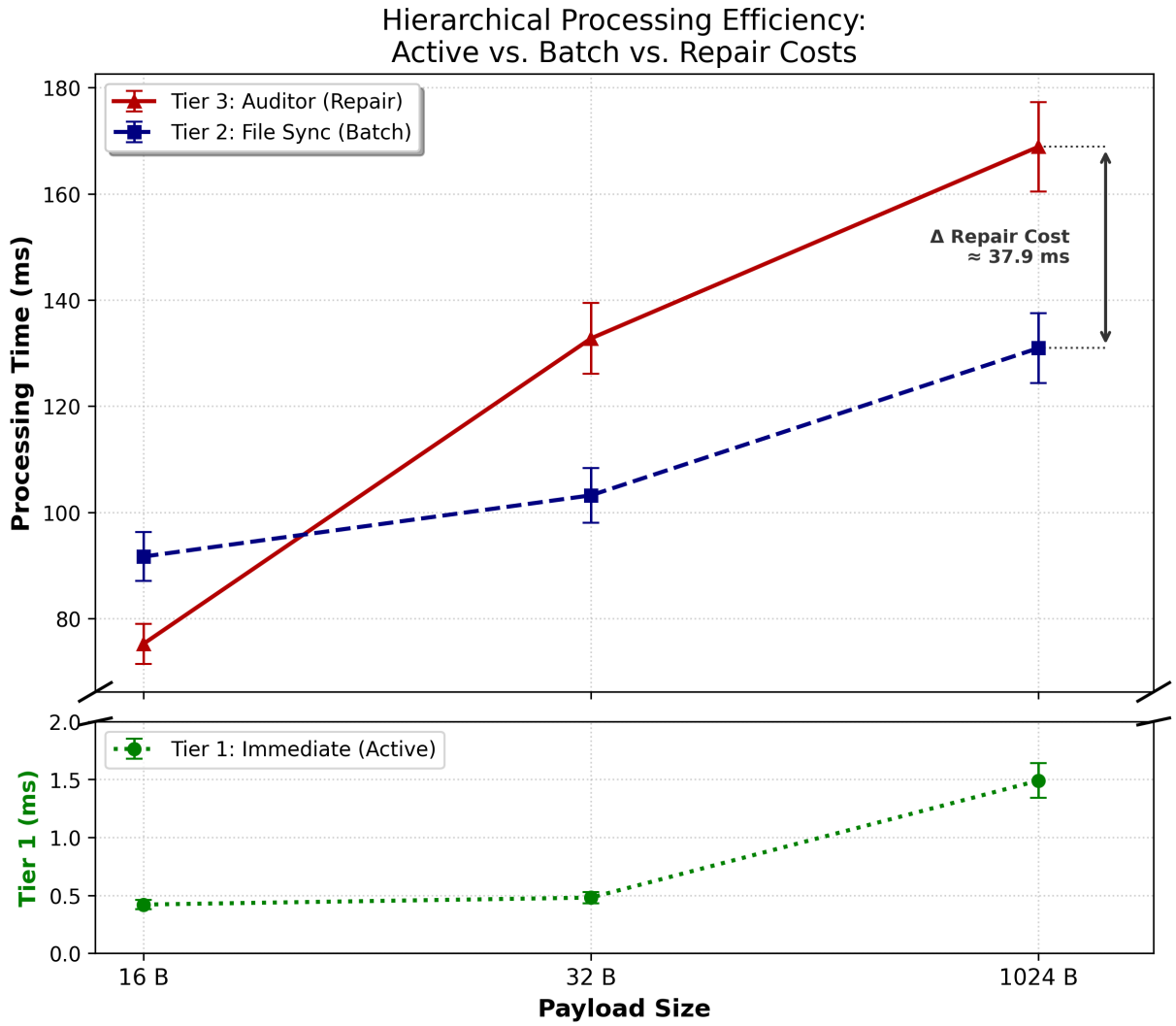

Figure 8: Processing latency comparison across active, batch, and corrective operations for varying payload sizes.

The auditor metric τ_{audit} ranges from 75.26 ms to 168.86 ms. This path includes receiving candidate files, re-verifying messages, re-verifying file-level consistency, removing duplicates, and producing a repaired master file when possible. The measured values show that local recovery is slower than immediate validation, as expected, but still bounded well enough to support practical self-healing before escalation to external arbitration.

Taken together, these results reveal a clear hierarchical efficiency pattern. The fastest and most frequent operations occur on the active path, synchronization is slower but safely offloaded to rollover time, and auditing is the most expensive because it performs corrective reconstruction. This behavior is consistent with the intended design logic of the SLUF channel: fast validation on the critical path, with heavier checking deferred to boundary events or conflict conditions.

5.4 Computational Resources and Storage Analysis

The local memory and storage footprint of the SLUF protocol were evaluated to estimate its suitability for resource-constrained edge environments Rzepka et al. [2024]. The measured RAM usage and storage accumulation are summarized in Fig. 9(a).

For a 16-byte message, the system consumes only 10.15 KB of RAM. Even for a 1024-byte message, the memory requirement is only 32.25 KB. These values indicate that the protocol logic is lightweight and does not require large working memory to maintain validation, signing, and logging operations.

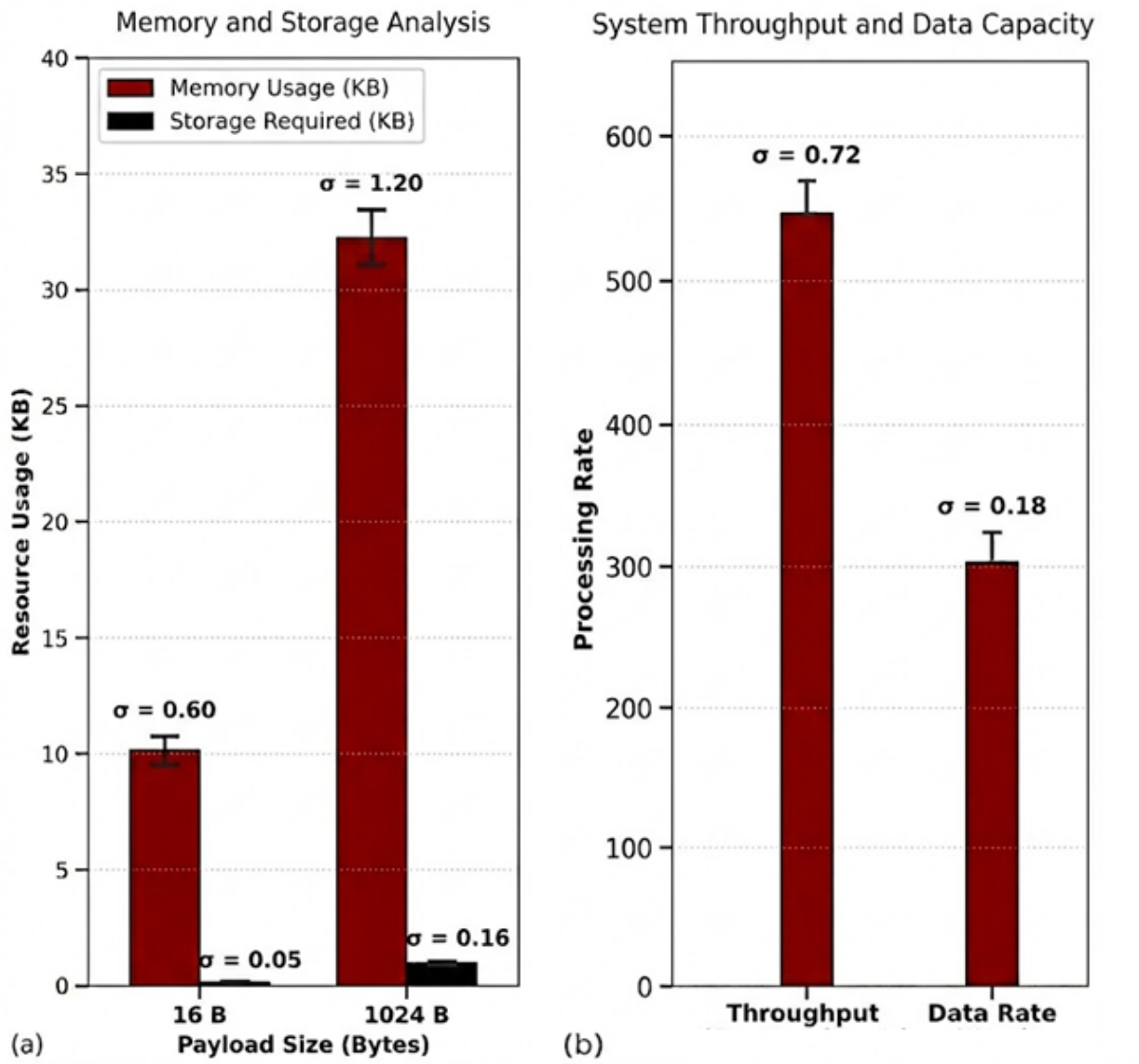

Figure 9: System Performance and Resource Utilization. (a) Memory and Storage Analysis. (b) System Throughput and Data Capacity

In terms of storage, a standard 16-byte message requires only 0.18 KB of disk space, whereas a 1024-byte payload requires 1.16 KB. The growth is modest and remains consistent with append-only local ledger storage. From a deployment perspective, this is useful because the protocol can preserve evidentiary logs for synchronization and auditing without creating an excessive storage burden on the host node.

Overall, the resource profile supports the claim that the SLUF channel is architecturally lightweight. This matters because low latency alone is not sufficient for practical cyber-physical deployment; the implementation must also avoid turning memory and storage into hidden bottlenecks.

5.5 Throughput and Data Handling Capacity

To evaluate the maximum operating capacity of the channel, throughput was measured under saturation conditions using 128-byte payloads over one-second intervals.

As shown in Fig. 9(b), the system achieved an average throughput of 546 transactions per second, with a standard deviation of 0.72. In this context, throughput denotes the number of fully processed protocol transactions completed per second, while the standard deviation quantifies the run-to-run variation observed across repeated one-second measurement windows. In terms of raw data movement, the channel processed 303 KB/s.

These values should be interpreted in light of the validation scope of the protocol. Unlike generic payment-oriented channels, the SLUF channel performs state transitions, signature verification, and semantic checks on every packet before execution Gangwal et al. [2023], Dziembowski et al. [2019]. The measured throughput therefore reflects an intentional trade-off between raw speed and safety-aware processing.

Accordingly, the throughput result should not be interpreted as a simple forwarding limit. Rather, it is the operating rate achieved after the protocol spends computation on admissibility and accountability. In other words, the system is not merely forwarding packets; it is determining whether each packet is allowed to affect the protected state. For the target use case, this is the more meaningful performance metric.

5.6 Experimental Security Evaluation

The SLUF channel was also evaluated under controlled adversarial scenarios representing the main threat classes considered in this work. The purpose was to verify whether representative attacks are rejected during validation or contained during reconciliation, while remaining consistent with the timing bounds measured in the previous subsections.

For clarity, the symbols used in Tables 2 and 3 follow the formal definitions introduced in Section 3. In particular, $V_{\mathrm{crypto}}(m)$ denotes the cryptographic validity predicate for message m, $V_{\mathrm{sem}}(m)$ denotes the semantic admissibility predicate, H_{trans} denotes the hash of the parent transaction used to preserve causal linkage, Eq. (29) denotes the local conflict-removal rule, Eq. (32) denotes the judge decision function for disputed ledgers, Eq. (22) denotes the causal-completeness rule, and Eq. (27) denotes the optional outlier-screening condition for aggregated data streams.

Table 2 summarizes the formal success condition of each representative attack class and the corresponding protocol rule that prevents success. Table 3 summarizes the observed system response under the controlled attack scenarios implemented in the prototype.

Together, the two tables show two complementary protection modes: *acceptance-time rejection* and *post-conflict recovery*. Spoofing, replay, man-in-the-middle manipulation, unsafe signed payloads, and invalid causal references are rejected before they can produce valid state transitions, whereas conflicting histories are routed to reconciliation and, when necessary, arbitration.

These observations are consistent with the performance results reported earlier in this section. Immediate rule enforcement remains in the sub-millisecond to low-millisecond range, which supports proactive validation on the active communication path. Synchronization and auditing are slower, but they occur outside the primary fast path and remain bounded enough to support recovery without forcing normal operation onto the blockchain.

Overall, the results support the main security claim of the paper: the SLUF channel combines low-latency admissibility checks with bounded reconciliation, allowing representative attack classes to be rejected before execution or removed before final settlement.

Attack class	**Adversarial capability**	**Attack success condition**	**Formal reason for failure**	**Practical implication in the results**
Masquerading / identity spoofing	External actor attempts to inject a state-changing message without valid registration credentials	A forged message is accepted as a valid protocol event	Rejected because $V_{\mathrm{crypto}}(m) = 0$ unless the signature verifies under the registered public key and the correct protocol anchor is present. Hence spoofed state transitions fail before acceptance	The cost of immediate validation remains bounded by the active-state rule-enforcement time, so proactive rejection does not violate the latency budget
Replay attack	External actor or malicious peer retransmits an old message	A stale transaction is accepted as a new state transition	Rejected by temporal admissibility and ordering in Eq. (18). Stale messages violate the monotonic sequence required for ledger acceptance	The sub-4 ms critical-path latency shows that replay checks can be enforced before execution in real time
Man-in-the-middle (MITM) attack	External adversary intercepts, relays, or modifies traffic on the communication path	Attacker-relayed or attacker-modified traffic is accepted as a valid protocol event	Rejected because on-path access alone is insufficient for acceptance. Any relayed or modified message must still satisfy the overall acceptance predicate in Eq. (40). Authenticated transport establishment and message-level cryptographic validation prevent the attacker from being accepted as a registered peer without valid credentials	The measured low critical-path latency indicates that transport- and message-level checks can still be enforced in real time without violating the operational timing budget
Equivocation / split-brain	Malicious registered peer signs conflicting histories for the same logical session	Two incompatible histories both survive into the settled state	Conflicting histories are first reduced by Eq. (29). If local repair is insufficient, Eq. (32) selects the authoritative ledger or rejects both, preventing persistent divergence	The measured auditor and synchronization times show that recovery is computationally tractable without moving normal operation on-chain
False data injection / unsafe signed command	Compromised legitimate node uses valid credentials to send unsafe measurements or commands	A cryptographically valid but unsafe payload is accepted and executed	Rejected because overall acceptance requires semantic admissibility in Eq. (43). Aggregated anomalies are additionally constrained by Eq. (27)	The measured rule-application overhead confirms that semantic checks can be applied proactively rather than only after the fact
Orphan acknowledgment / causal forgery	Multi-actor adversary attempts to inject a response without a valid parent request	A response without valid lineage enters the committed ledger	Rejected by causal completeness in Eq. (22), which requires a valid parent-child relation through H_{trans}	File-level synchronization preserves a consistent chain of custody before final hashing

Table 2: Formal threat-resistance analysis of the SLUF channel

Attack scenario	**Injected behavior**	**Observed system response**	**Security implication**
Replay attack	Re-submission of a previously valid message under stale temporal context	Message rejected during validation due to timestamp and ordering constraints	Stale messages cannot be reintroduced as valid new events
Identity spoofing	Forged message submitted without a valid sender signature and protocol anchor	Message rejected during signature and context validation	Unregistered actors cannot inject accepted protocol events
Equivocation / split-brain	Conflicting histories produced for the same logical session	Mismatch detected and routed to reconciliation and, when needed, arbitration	Divergent histories do not persist into final settlement
Unsafe signed payload / FDIA-style command	Payload signed by a valid identity but violating semantic safety constraints	Payload rejected before execution by semantic admissibility checks	Credential compromise alone does not imply operational acceptance
Orphan acknowledgment / causal forgery	Response submitted without a valid parent request reference	Record rejected during lineage verification or excluded during synchronization	Invalid causal chains cannot enter the committed ledger

Table 3: Observed outcomes under controlled attack scenarios

6 Conclusion

This paper addressed three main obstacles to the use of blockchain-based mechanisms in real-time power-system automation: the scalability trilemma, the semantic gap of generic Layer-2 protocols, and the diligence gap of reactive security models. To address these limitations, it introduced the Second-Layer Ultra-Fast (SLUF) channel, an off-chain architecture for secure, low-latency, and accountable cyber-physical communication.

The proposed design models the communication channel as a finite-state machine supported by a formal state-convergence framework. Unlike conventional approaches that emphasize either off-chain scalability or secure transport, the SLUF channel is designed for operational settings in which a message must be authentic, timely, physically admissible, and recoverable under conflict. To support this requirement, the architecture performs validation, reconciliation, and accountability off-chain while preserving transaction integrity, semantic admissibility, and structured dispute handling. By combining message validation, ledger synchronization, conflict elimination, and a two-tier arbitration mechanism, the SLUF channel replaces delayed trust assumptions with protocol-level proof of diligence.

The software realization further shows that this model can be implemented efficiently through an event-driven Node.js architecture that separates active communication, synchronization, and auditing into parallel execution paths. Experimental evaluation showed that the system achieves 546 transactions per second for 128-byte payloads, maintains critical processing latency below 4 ms, and requires only 10.15 KB of RAM for standard messages. These results indicate that the proposed channel can provide bounded processing overhead compatible with real-time cyber-physical operation while avoiding dependence on blockchain confirmation delays during normal execution.

Although the present results demonstrate low latency and low overhead on a standard x86 platform, additional benchmarking on representative embedded and edge devices remains necessary. Future work should also expand the comparative evaluation against other off-chain and secure communication approaches under matched experimental conditions. Nevertheless, the SLUF channel shows that power-system automation can be decoupled from blockchain congestion through a secure, practically efficient, and formally accountable second-layer design.

Acknowledgments

This research was supported by the InterSCADA project (Interoperable, Scalable, and Secure AC-DC Modular Automation System), funded by the European Union's Horizon Europe program under Grant No. 101173007.

References

Morteza Aghahadi, Alessandro Bosisio, Marco Merlo, Alberto Berizzi, Andrea Pegoiani, and Samuele Forciniti. Digitalization processes in distribution grids: a comprehensive review of strategies and challenges. *Applied Sciences*, 14(11):4528, 2024.

M. Nasrinasrabadi, M. A. Hejazi, E. Chaharmahali, and M. Hussein. A comprehensive review of blockchain integration in smart grid with a special focus on internet of things. *Energy Conversion and Management: X*, page 101196, 2025.

Souhail Mssassi and Anas Abou El Kalam. The blockchain trilemma: A formal proof of the inherent trade-offs among decentralization, security, and scalability. *Applied Sciences*, 15(1):19, 2025.

N. Gajanur, M. Greidanus, G. S. Seo, S. K. Mazumder, and M. A. Abbaszada. Impact of blockchain delay on grid-tied solar inverter performance. In *2021 IEEE 12th International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, pages 1–7, 2021.

Yi Yu, Guo-Ping Liu, Yi Huang, Chi Yung Chung, and Yu-Zhong Li. A blockchain consensus mechanism for real-time regulation of renewable energy power systems. *Nature Communications*, 15(1):10620, 2024.

Stéphane Mocanu and Jean-Marc Thiriet. Real-time performance and security of iec 61850 process bus communications. *Journal of Cyber Security and Mobility*, 10(2):305–346, 2021.

Ankit Gangwal, Haripriya Ravali Gangavalli, and Apoorva Thirupathi. A survey of layer-two blockchain protocols. *Journal of Network and Computer Applications*, 209:103539, 2023.

Turki Ali Alghamdi, Rabiya Khalid, and Nadeem Javaid. A survey of blockchain based systems: Scalability issues and solutions, applications and future challenges. *IEEE Access*, 12:79626–79651, 2024.

L. T. Thibault, T. Sarry, and A. S. Hafid. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access*, 10: 93039–93054, 2022.

Ghadeer O. Alsharif, Christos Anagnostopoulos, and Angelos K. Marnerides. Energy market manipulation via false-data injection attacks: A review. *IEEE Access*, 13:42559–42573, 2025. doi:10.1109/ACCESS.2025.3548914.

Ömer Sen, Dennis van der Velde, Maik Lühman, Florian Sprünken, Immanuel Hacker, Andreas Ulbig, Michael Andres, and Martin Henze. On specification-based cyber-attack detection in smart grids. *Energy Informatics*, 5(Suppl 1):23, 2022.

P. Sheng, R. Rana, S. Bala, H. Tyagi, and P. Viswanath. Proof of diligence: Cryptoeconomic security for rollups. *arXiv preprint arXiv:2402.07241*, 2024.

Stefanos Chaliasos, Itamar Reif, Adrià Torralba-Agell, Jens Ernstberger, Assimakis Kattis, and Benjamin Livshits. Analyzing and benchmarking zk-rollups. In *6th Conference on Advances in Financial Technologies (AFT 2024)*, pages 6:1–6:21. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

SM Suhail Hussain, Mohd Asim Aftab, Shaik Mullapathi Farooq, Ikbal Ali, Taha Selim Ustun, and Charalambos Konstantinou. An effective security scheme for attacks on sample value messages in iec 61850 automated substations. *IEEE Open Access Journal of Power and Energy*, 10:304–315, 2023.

Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. Scaling blockchains: A comprehensive survey. *IEEE Access*, 8:125244–125262, 2020.

Domenico Tortola, Andrea Lisi, Paolo Mori, and Laura Ricci. Tethering layer 2 solutions to the blockchain: A survey on proving schemes. *Computer Communications*, 225:289–310, 2024.

W. Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2009.

Tian Junfeng, Bai Wenqing, and Jia Haoyi. Pgce: A distributed storage causal consistency model based on partial geo-replication and cloud-edge collaboration architecture. *Computer Networks*, 212:109065, 2022.

Muhammad Nouman Nafees, Neetesh Saxena, Alvaro Cardenas, Santiago Grijalva, and Pete Burnap. Smart grid cyber-physical situational awareness of complex operational technology attacks: A review. *ACM computing surveys*, 55(10):1–36, 2023.

Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. Bft protocol forensics. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1722–1743, 2021.

Eric Rescorla. The transport layer security (tls) protocol version 1.3. RFC 8446, 2018.

Peter Saint-Andre and Jeff Hodges. Representation and verification of domain-based application service identity within internet public key infrastructure using x.509 (pkix) certificates in the context of transport layer security (tls). RFC 6125, 2011.

Muhammad Usama and Muhammad Naveed Aman. Command injection attacks in smart grids: A survey. *IEEE Open Journal of Industry Applications*, 5:75–85, 2024.

Batoul Achaal, Mehdi Adda, Maxime Berger, Hussein Ibrahim, and Ali Awde. Study of smart grid cyber-security, examining architectures, communication networks, cyber-attacks, countermeasure techniques, and challenges. *Cybersecurity*, 7(1):10, 2024. doi:10.1186/s42400-023-00200-w.

P. Linz and S. H. Rodger. *An Introduction to Formal Languages and Automata*. Jones & Bartlett Learning, 2022.

Karol Rzepka, Przemysław Szary, Krzysztof Cabaj, and Wojciech Mazurczyk. Performance evaluation of raspberry pi 4 and stm32 nucleo boards for security-related operations in iot environments. *Computer Networks*, 242:110252, 2024.

Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 106–123. IEEE, 2019.