

XIM: The XDC Interledger Messaging Protocol

A Verification-Agnostic Messaging and Settlement Fabric for Heterogeneous Ledgers and Financial Rails

Atul Khekade Ritesh Kakkad Wanwiset Peerapatanapokin
Behnam Mohammadkhani

XDC Network Research and Engineering

September 2026

Abstract

Distributed ledgers, privacy-preserving institutional networks, and conventional payment systems increasingly need to exchange authenticated messages and settle assets across heterogeneous trust domains. Existing interoperability systems typically optimize for one of three concerns: application-level abstraction, cross-chain message transport, or synchronized execution within a related ledger family. This paper proposes XIM, the XDC Interledger Messaging Protocol, a chain-agnostic protocol for transporting canonical messages across heterogeneous networks while allowing each communication lane to select an explicit verification policy. XIM separates message semantics from transport, verification, execution, routing, asset identity, and compliance metadata. Its cryptographic state is represented by deterministic message identifiers and commitment roots, while an append-only transition log provides auditability and replay protection. XIM introduces a Universal Asset Identifier (UAID), a pluggable adapter interface, lane-scoped security policies, and an optional policy-aware route graph for multi-hop settlement. XDC Network can serve as a coordination and settlement domain without requiring every XIM message or route to transact through XDC. We specify the protocol model, state machine, message encoding, commitment structure, verification modes, failure semantics, security assumptions, threat model, implementation architecture, and an incremental deployment plan. The design targets public blockchains, permissioned ledgers, institutional networks, and authenticated financial-system gateways, with particular attention to stablecoins, tokenized assets, trade finance, and ISO 20022-compatible payment workflows.

Keywords: blockchain interoperability; enterprise middleware; cross-chain messaging; interledger; XDC Network; Merkle commitments; light clients; zero-knowledge proofs; stablecoins; tokenized assets; ISO 20022; institutional settlement; routing; distributed ledgers.

1 Introduction

Financial infrastructure is becoming multi-ledger rather than converging on a single execution environment. Public smart-contract networks [16, 17], permissioned ledgers, privacy-preserving institutional networks, tokenized deposit platforms, stablecoin networks, and conventional payment rails have different execution models, finality assumptions, privacy constraints, and governance. Point-to-point bridges do not scale organizationally or technically as the number of networks grows, while a single universal consensus system would require heterogeneous networks to abandon their native trust and privacy models.

XIM addresses this problem by treating interoperability as authenticated message exchange between independently governed state machines. The protocol defines a canonical message, deterministic identifiers, explicit lifecycle states, cryptographic commitments, replay protection,

verification-policy negotiation, destination execution, and acknowledgements. It intentionally does not mandate one consensus algorithm or one cross-chain proof mechanism. Instead, each source–destination lane binds to a verification policy appropriate to the value, finality, and trust characteristics of the connected systems.

The central design principle is separation of concerns. Transport moves bytes; verification establishes that a source event satisfies a lane’s trust policy; routing selects an eligible path; execution applies a verified action at a destination; and settlement concerns the transfer or transformation of economic value. A compromise in one component should not automatically grant authority over all other components.

1.1 Contributions

- A canonical, deterministic interledger message envelope independent of source-chain transaction format.
- Lane-scoped verification policies supporting native proofs, light clients, zero-knowledge proofs, threshold attestations, trusted-hardware attestations, and hybrid verification.
- Commitment-oriented state based on message hashes and Merkle roots rather than global replication of foreign-chain state.
- An explicit cross-domain message state machine with acknowledgement, timeout, refund, and terminal failure semantics.
- A Universal Asset Identifier (UAID) separating economic asset identity from chain-specific token contract addresses.
- A pluggable network adapter interface for public chains, permissioned ledgers, institutional networks, and authenticated legacy gateways.
- An optional policy-aware route graph for multi-hop settlement based on cost, latency, liquidity, security, privacy, jurisdiction, and finality.
- Separation of routing and execution infrastructure from an independent lane risk-monitoring function capable of rate-limiting or pausing a compromised lane.

1.2 Non-goals

- XIM is not a new general-purpose Layer-1 blockchain.
- XIM does not require global replication of all connected-ledger transactions.
- XIM does not claim trustless interoperability where the connected source cannot produce independently verifiable evidence.
- XIM does not define universal legal finality; legal settlement finality remains dependent on the relevant asset, institution, jurisdiction, and network rules.
- XIM does not require XDC to be an intermediate asset or mandatory hop for every route.

1.3 Conventions

The key words MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY are to be interpreted as described in BCP 14 [23, 24] when, and only when, they appear in all capitals. $H(\cdot)$ denotes a collision-resistant hash function and $\parallel$ denotes byte-string concatenation.

2 Related Work and Design Context

Interoperability systems can be viewed along several architectural axes. API-gateway approaches abstract network-specific interfaces behind standardized application APIs [1, 2]. Cross-chain messaging protocols authenticate source events and cause destination execution [3]. Light-client protocols verify counterparty consensus state and packet commitments [4, 5]. Privacy-preserving institutional networks coordinate selectively disclosed transactions among authorized participants [7, 8, 9]. XIM adopts concepts from each family but defines a distinct protocol boundary: a canonical message and state-transition system with pluggable verification and execution. Table 1 summarizes the lessons drawn from representative systems.

Table 1: Representative interoperability systems and the lessons XIM draws from them.

System / family	Primary abstraction	Relevant lesson for XIM
Quant Overledger / API gateways [1, 2]	Unified API over heterogeneous DLT and legacy systems	Application abstraction and connector modularity.
Chainlink CCIP [3]	Cross-chain messages and tokens with separate commit/execute security components	Separation of routing, verification, execution, and independent risk controls.
IBC [4, 5]	Clients, connections/channels, packet commitments, acknowledgements, and timeouts	Explicit counterparty verification and packet lifecycle semantics.
Circle CCTP [6]	Issuer-mediated burn/mint with signed attestation	Prefer canonical native asset movement over wrapped representations when issuer support exists.
Canton [7, 8, 9]	Privacy-aware synchronization across participant nodes and applications	Interoperability can preserve selective disclosure rather than forcing global data replication.

Beyond production systems, the academic literature provides systematizations of cross-ledger communication. Zamyatin et al. [10] show that correct cross-chain communication cannot be achieved without a trusted third party or additional assumptions, which motivates XIM’s requirement that every lane name its trust assumption explicitly. Belchior et al. [11] survey the broader design space of blockchain interoperability. Herlihy’s atomic cross-chain swaps [12] underpin the hash- and time-conditioned settlement discussed in Section 20, and zkBridge [13] demonstrates practical succinct verification of source-chain consensus of the kind assumed by the ZK_PROOF verification mode. Empirical analyses of cross-chain bridge exploits [14] motivate the lane-scoped limits and circuit breakers of Section 21. Finally, the term *interledger* is also associated with the Interledger Protocol (ILP) [15], which routes payments across ledgers through connectors using conditional transfers. XIM differs in scope: it carries general authenticated messages rather than only payments, and it binds an explicit, versioned verification policy to each lane rather than relying on connector-held liquidity and hashlock conditions.

3 System Model

Let $\mathcal{N} = \{N_1, N_2, \dots, N_n\}$ be a set of independently governed networks. A network may be a public blockchain, permissioned DLT, institutional ledger, or authenticated financial-system gateway. XIM defines directed communication lanes L_{ij} from source network N_i to destination network N_j . A lane is identified by its source network, destination network, protocol version, adapter versions, and verification policy.

A XIM deployment may include Observers, Verifiers, Routers, Executors, Risk Monitors, and Gateways (Table 2). Roles may be combined in an early deployment but are logically distinct.

Table 2: Logical roles in a XIM deployment.

Role	Responsibility
Observer	Detects finalized or policy-eligible source events and constructs candidate XIM messages.
Verifier	Evaluates source evidence against the lane verification policy.
Router	Selects direct or multi-hop paths that satisfy policy constraints.
Executor	Submits a verified destination action and reports the resulting destination transaction.
Risk Monitor	Independently detects anomalies and may throttle or pause a lane according to governance rules.
Gateway	Maps non-DLT financial messages or authenticated APIs into and out of XIM semantics.

3.1 Trust Domains

XIM does not collapse connected networks into a single trust domain. The source network remains responsible for its own consensus and finality. The destination remains responsible for destination execution. The lane verification policy defines the evidence required for the destination to accept a source assertion. Consequently, security claims **MUST** be expressed per lane, not globally.

4 Network and Asset Identifiers

4.1 Network Identifier

$$\text{NetworkID} := H(\text{namespace} \parallel \text{canonical_network_name} \parallel \text{genesis_or_domain_identifier}) \quad (1)$$

A NetworkID **MUST** be globally stable for a specific trust domain. Forked or re-genesis networks **SHOULD** receive a new identifier when their security identity changes materially.

4.2 Universal Asset Identifier (UAID)

A UAID identifies an economic asset independently from its representation on a particular ledger. A UAID record is governance-controlled metadata and **MUST NOT** imply that two representations are economically equivalent unless the registry’s issuer and conversion policy establishes that equivalence.

```

UAIDRecord {
  uuid
  issuer_id
  asset_type
  denomination
  jurisdiction
  representations[]
  issuance_policy_hash
  redemption_policy_hash
  compliance_policy_hash
}

```

For example, native USDC on multiple supported networks may map to one issuer-level asset identity, while a third-party wrapped token **SHOULD** use a distinct representation classification unless redemption equivalence is explicitly verified.

5 Canonical XIM Message

XIM messages MUST use deterministic serialization. The reference implementation SHOULD use deterministic Protocol Buffers or an equivalent canonical binary encoding. JSON MAY be exposed at API boundaries but MUST NOT be used directly for consensus-critical hashing unless canonical JSON rules are specified.

```
XIMMessage {
  uint32  version;
  bytes32 source_network;
  uint64  source_height;
  bytes   source_tx_id;
  bytes   sender;

  bytes32 destination_network;
  bytes   receiver;

  uint32  action_type;
  bytes32 asset_id;
  uint256 amount;
  bytes32 payload_hash;

  uint64  nonce;
  uint64  created_at;
  uint64  expiry;

  uint32  finality_policy;
  uint32  proof_type;
  bytes32 proof_commitment;

  bytes32 compliance_metadata_hash;
  bytes32 privacy_policy_hash;
  bytes32 route_policy_hash;
}
```

Fixed-width types such as `bytes32` and `uint256`, which have no native Protocol Buffers equivalent, are carried as `bytes` in the wire encoding (Appendix A); a conforming decoder MUST reject values whose length differs from the declared width, and `uint256` values MUST be encoded as 32-byte big-endian integers.

5.1 Message Identifier

$$\text{message_id} = H(\text{domain_separator} \parallel \text{canonical_encode}(\text{XIMMessage})) \quad (2)$$

The domain separator MUST bind the protocol name and major version. A destination MUST reject a previously consumed `message_id`. This provides protocol-level replay protection independent of destination-chain nonce behavior.

5.2 Action Types

Table 3 lists the action types defined by this version of the protocol.

6 Message Lifecycle and State Machine

Figure 1 shows the nominal lifecycle of a XIM message together with its exceptional terminal states. State transitions MUST be monotonic except where an explicit compensating transaction creates a new message. XIM SHOULD avoid pretending that irreversible execution can be rolled back across heterogeneous ledgers. Instead, failures after irreversible execution are handled

Table 3: XIM action types.

Action	Meaning
MESSAGE	Deliver an application payload commitment.
TRANSFER	Transfer or release an asset representation.
MINT_BURN	Coordinate canonical issuer-mediated burn/mint.
CALL	Invoke a destination contract or application endpoint.
ATOMIC_LEG	Represent one leg of a multi-leg conditional settlement.
ACK	Acknowledge a terminal or intermediate state.
CANCEL / REFUND	Request policy-authorized cancellation or compensation.

by acknowledgements and compensating actions. Table 4 lists the evidence required for each nominal transition.

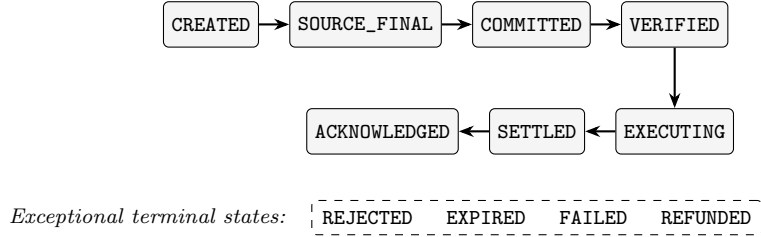


Figure 1: Nominal XIM message lifecycle (solid) and exceptional terminal states (dashed). The order along the solid path defines the lifecycle order $\prec$ used in Section 26.

Table 4: Evidence required for each nominal state transition.

Transition	Required evidence
CREATED $\rightarrow$ SOURCE_FINAL	Source finality evidence satisfying lane policy.
SOURCE_FINAL $\rightarrow$ COMMITTED	Inclusion of the message commitment in a recognized commitment batch/root.
COMMITTED $\rightarrow$ VERIFIED	Successful lane verifier result.
VERIFIED $\rightarrow$ EXECUTING	Executor authorization and destination submission.
EXECUTING $\rightarrow$ SETTLED	Destination execution receipt/finality satisfying destination policy.
SETTLED $\rightarrow$ ACKNOWLEDGED	Acknowledgement commitment returned or recorded.

7 Commitment Structure

XIM uses commitment roots to summarize message state without replicating complete foreign-chain data. The baseline structure is a Sparse Merkle Tree (SMT) [18, 19] keyed by `message_id`. Implementations MAY use a binary Merkle tree, Verkle commitment [20], or other authenticated dictionary if they preserve equivalent inclusion/non-inclusion and update-verification properties and are identified by `commitment_type`.

$$\text{key} = \text{message_id} \quad (3)$$

$$\text{value} = H(\text{state} \parallel \text{source_finality_commitment} \parallel \text{verification_result} \parallel \text{destination_tx_commitment} \parallel \text{acknowledgement_commitment}) \quad (4)$$

$$\text{leaf} = H(\text{key} \parallel \text{value}) \quad (5)$$

Batch roots can be checkpointed to XDC or another coordination ledger. Checkpointing to XDC provides a common audit and dispute anchor but is not required for the correctness of every possible XIM deployment.

8 Append-only Transition Log

```
XIMEvent {  
  message_id  
  sequence  
  event_type  
  network_id  
  ledger_position  
  timestamp  
  previous_state_hash  
  new_state_hash  
  evidence_hash  
  actor_id  
}
```

For each message, sequence numbers **MUST** be strictly increasing and contiguous (invariant I6 in Section 26). The transition log enables deterministic reconstruction of the XIM state, institutional audit trails, incident analysis, and reconciliation. Sensitive source documents **SHOULD** remain off-chain and be referenced only by cryptographic commitments where appropriate.

9 Verification Framework

Verification is the central security boundary. Each lane binds to a **VerificationPolicy** object. No message can be executed solely because a Router asserts that it exists.

```
VerificationPolicy {  
  policy_id  
  source_network  
  destination_network  
  proof_type  
  min_finality  
  verifier_set_or_contract  
  threshold  
  max_message_value  
  rate_limit  
  challenge_period  
  policy_version  
}
```

Table 5 lists the supported proof modes. Verification policies **SHOULD** be value-aware: a low-value informational message may accept a different policy from a high-value asset transfer. Governance **MUST** define how policies are upgraded and how in-flight messages are treated during upgrades.

Table 5: Verification modes and their trust bases.

Proof mode	Typical use	Trust basis
NATIVE_PROOF	Networks with destination-verifiable state/finality proofs	Source protocol cryptography.
LIGHT_CLIENT	Consensus-verifiable chains	Correct light-client implementation and source consensus.
ZK_PROOF	Succinct verification of source state/execution	Proof-system soundness, circuit correctness, source consensus inputs.
THRESHOLD_ATTESTATION	Networks without practical on-chain light clients	Honest threshold, key security, and governance.
TEE_ATTESTATION	Specialized gateways	Hardware and attestation trust assumptions.
HYBRID	High-value institutional lanes	A combination, e.g. a light-client proof plus an independent risk quorum.

10 Adapter Interface

```

interface IXIMAdapter {
  networkID()           -> bytes32
  observe(filter, cursor) -> Event[]
  finality(event)       -> FinalityEvidence
  buildProof(event)      -> Proof
  verifyProof(message, proof) -> bool
  buildExecution(message) -> DestinationTransaction
  submit(transaction)    -> TxID
  executionStatus(txID)  -> ExecutionReceipt
  estimateFee(message)   -> FeeQuote
}

```

Adapters **MUST NOT** silently reinterpret XIM fields. Network-specific mappings **SHOULD** be explicit, versioned, and testable. An adapter upgrade that changes address interpretation, finality, or asset semantics **MUST** create a new adapter version and **SHOULD** require lane governance approval.

11 Routing

Direct lanes are sufficient for basic interoperability. XIM additionally defines an optional route-selection layer modeled as a directed graph $G = (V, E)$, where V represents networks or settlement domains and E represents enabled XIM lanes or financial conversion edges. For an edge e and settlement intent ι ,

$$\begin{aligned} \text{EdgeWeight}(e, \iota) = & w_{\text{fee}} \cdot \text{fee}(e) + w_{\text{lat}} \cdot \text{latency}(e) + w_{\text{risk}} \cdot \text{risk}(e) \\ & + w_{\text{liq}} \cdot \text{liquidity_penalty}(e) + \text{policy_penalties}(e, \iota), \end{aligned} \quad (6)$$

where fee and latency are normalized to a common scale.

Routing **MUST** be constrained before it is optimized. An edge that fails a hard requirement for jurisdiction, asset support, privacy, verification strength, maximum value, or deadline is removed from the candidate graph. Only then may the router optimize cost or latency. This prevents a cheaper but policy-ineligible path from being selected.

Routers are advisory. A destination verifier **MUST** independently enforce lane and message policy even if a Router proposes a route.

12 Intent Layer

```
SettlementIntent {  
  intent_id  
  payer  
  beneficiary  
  source_asset  
  destination_asset  
  amount  
  max_fee  
  deadline  
  required_finality  
  privacy_constraints  
  jurisdiction_constraints  
  compliance_policy_hash  
}
```

An intent describes the desired economic outcome without binding the requester to a particular network path. One intent may compile into one or more XIM messages. The intent layer is optional and does not alter the security validity of the underlying messages.

13 Stablecoin and Tokenized-Asset Semantics

XIM distinguishes canonical issuer-mediated movement from bridge-issued representations. Where an issuer provides native burn/mint or equivalent cross-domain issuance [6], XIM **SHOULD** prefer that mechanism over lock-and-mint wrapping because it avoids accumulating bridge custody risk. The proof policy for such a lane may incorporate issuer attestations in addition to source-chain evidence.

Tokenized securities, deposits, trade-finance instruments, and real-world assets may require transfer restrictions. XIM does not encode legal compliance into a universal global rule; instead, **compliance_policy_hash** commits to the applicable policy, and destination execution **MUST** enforce the rules required by the asset representation and participating institutions.

14 Privacy Model

XIM is commitment-oriented and does not require public disclosure of full business payloads. A message may carry **payload_hash**, **compliance_metadata_hash**, and **privacy_policy_hash** while encrypted or selectively disclosed data moves through authorized channels. This design is compatible with privacy-preserving institutional systems in which only entitled participants receive transaction details.

Privacy is not provided merely by hashing low-entropy data. Sensitive metadata **MUST** be encrypted, salted where appropriate, or represented by commitments that resist dictionary attacks. Implementations **SHOULD** minimize public metadata and **SHOULD** consider traffic-analysis leakage at the routing layer.

15 ISO 20022 and Legacy Financial Gateways

XIM gateways may translate authenticated financial instructions into XIM intents and messages (Figure 2). For example, an ISO 20022 [21] **pacs.008** instruction can be mapped into a payment

intent while preserving a cryptographic reference to the original message. The gateway is a trust boundary: XIM can authenticate the gateway’s assertion but cannot independently prove facts that exist only inside a bank’s private system unless the bank exposes verifiable evidence.



Figure 2: Mapping a legacy financial instruction into XIM. The gateway is an explicit trust boundary.

Gateway deployments SHOULD use HSM-backed signing keys, explicit institution identifiers, key rotation, message deduplication, sanctions and compliance controls where legally required, and auditable mapping rules.

16 XDC Coordination and Settlement Role

XDC Network, whose XDPoS 2.0 consensus provides deterministic finality through a HotStuff-based BFT protocol [22], can act as a neutral checkpoint, coordination, fee, collateral, and settlement domain for XIM deployments. However, XIM’s interoperability semantics do not require every message to transit XDC. This distinction prevents unnecessary token hops while allowing XDC to be used where its execution cost, finality, liquidity, and institutional integrations make it an efficient settlement domain.

Potential XDC-native components include `XIMRouter`, `XIMMessageRegistry`, `XIMVerifierRegistry`, `XIMAssetRegistry`, `XIMAdapterRegistry`, and `XIMRiskManager`. Optional modules include liquidity, fee, and compliance registries.

17 Reference Smart-Contract Interfaces

```

interface IXIMMessageRegistry {
    function commit(bytes32 messageId, bytes32 messageCommitment) external;
    function markVerified(bytes32 messageId, bytes32 proofHash) external;
    function markSettled(bytes32 messageId, bytes32 destinationTxHash) external;
    function acknowledge(bytes32 messageId, bytes32 ackHash) external;
    function status(bytes32 messageId) external view returns (uint8);
}

interface IXIMVerifierRegistry {
    function verifier(bytes32 laneId, uint32 policyVersion)
        external view returns (address);
    function verify(bytes32 laneId, bytes calldata message, bytes calldata proof)
        external view returns (bool);
}

interface IXIMRiskManager {
    function laneStatus(bytes32 laneId) external view returns (uint8);
    function valueLimit(bytes32 laneId) external view returns (uint256);
}
  
```

18 Security Properties

Subject to the explicit assumptions of the selected lane verification policy, XIM aims to provide the properties in Table 6.

Table 6: Target security properties (informal).

Property	Informal requirement
Authenticity	A destination accepts a source assertion only if the required verification evidence validates.
Integrity	Any mutation of a canonical message changes its <code>message_id</code> and commitment.
Replay resistance	A consumed <code>message_id</code> cannot execute twice on the same destination security domain.
Ordering where required	Per-channel or application sequence constraints can be enforced.
Timeout safety	Expired messages cannot newly execute unless the application explicitly permits late settlement.
Auditability	State transitions are reconstructible from authenticated commitments and event records.
Lane isolation	A compromise or anomaly on one lane can be paused without globally halting unrelated lanes.
Policy explicitness	Security assumptions are bound to versioned lane policies rather than hidden in relayer software.

19 Threat Model

Table 7 summarizes the principal threats, their mitigations, and residual limitations.

Table 7: Threats, mitigations, and limitations.

Threat	Mitigation / limitation
Malicious router	Router output is advisory; the destination verifies proof and policy independently.
Compromised executor	The executor cannot create valid source evidence; the destination contract enforces verification and replay protection.
Source-chain reorganization	Minimum finality policy; probabilistic-finality safety margins; delayed high-value execution.
Verifier quorum compromise	Threshold diversification, stake/bonding where appropriate, key rotation, independent risk monitor, lane value limits.
Smart-contract bug	Small modular contracts, audits, formal verification of critical state transitions, staged limits.
Replay across chains or protocol versions	Domain-separated message hash including source/destination identifiers and protocol version.
Proof-system vulnerability	Versioned verifier registry, emergency lane pause, multiple proof modes or hybrid verification.
Gateway key compromise	HSMs, rate/value limits, multi-party authorization, anomaly detection, rapid key revocation.
Liquidity manipulation	Hard slippage/liquidity constraints and independent quote validation.
Privacy leakage	Commitment-only public fields, encryption, metadata minimization; traffic analysis remains a residual risk.

19.1 Safety versus Liveness

XIM prioritizes safety for asset-moving lanes. If finality, verifier availability, or risk status is uncertain, the protocol SHOULD halt or delay execution rather than guess. Liveness therefore depends on source availability, verifier availability, destination availability, and governance responsiveness.

20 Atomicity and Multi-leg Settlement

True atomicity across unrelated consensus systems is not always achievable without shared locking or a common synchronizer [10]. XIM therefore distinguishes (i) native atomic execution within one domain, (ii) conditional cross-domain settlement using escrow, hash, or time conditions or application locks [12], and (iii) coordinated but non-atomic settlement with compensating actions. Implementations **MUST** accurately advertise which model applies.

For institutional delivery-versus-payment, a deployment may place both legs under a common conditional settlement contract or domain, or use synchronized application-level locks with bounded timeouts. XIM provides message coordination but does not claim that messaging alone eliminates principal risk.

21 Risk Management and Circuit Breakers

An independent Risk Monitor **SHOULD** evaluate source finality anomalies, unusual mint/burn volumes, verifier disagreement, rapid value spikes, proof failures, adapter inconsistencies, and chain incidents [14]. Risk controls **SHOULD** be lane-scoped and may include:

- per-message value caps;
- rolling hourly and daily lane limits;
- velocity limits per asset and participant;
- emergency pause for new executions while allowing safe acknowledgement and refund paths;
- verifier-set diversity requirements;
- delayed settlement for unusually large transfers; and
- automatic downgrade to stricter verification when risk thresholds are exceeded.

22 Governance and Upgradeability

Governance controls verifier policies, adapter versions, asset registry entries, lane activation, limits, and emergency actions. Governance itself is part of the threat model. Production deployments **SHOULD** use timelocks for non-emergency upgrades, multi-party authorization, transparent change records, and explicit emergency powers with post-event review.

Protocol versioning **MUST** preserve deterministic interpretation of historical messages. A major version change **SHOULD** use a new domain separator.

23 Performance Model

End-to-end latency is bounded by the slowest mandatory stage rather than by XDC block time alone:

$$T_{\text{total}} = T_{\text{source_finality}} + T_{\text{observation}} + T_{\text{proof_generation}} + T_{\text{verification}} \\ + T_{\text{routing}} + T_{\text{destination_submission}} + T_{\text{destination_finality}} + T_{\text{ack}}. \quad (7)$$

For threshold-attested lanes, proof generation may be short but carries stronger trust assumptions. For light-client or ZK lanes, cryptographic verification may reduce trust but increase engineering complexity or proof latency. XIM makes this trade-off explicit at the lane-policy level.

Throughput can be improved through message batching. If k messages are committed under one Merkle root, checkpoint cost is amortized across k messages. Executors MAY batch destination submissions where destination semantics permit.

24 Reference Node Architecture

Figure 3 shows the reference node architecture. A reference implementation can use PostgreSQL for durable message and application metadata, Redis for ephemeral route and quote caches, RocksDB for authenticated state, and a graph engine or in-memory adjacency representation for routing. The cryptographic commitment root, not the database engine, defines protocol state integrity.

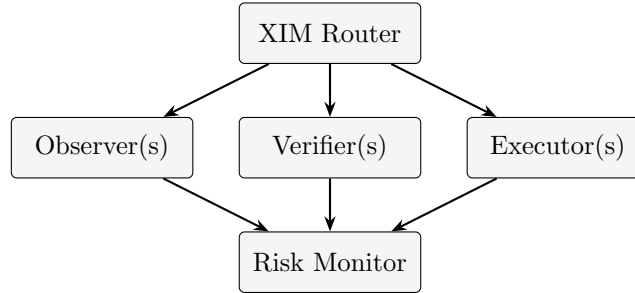


Figure 3: Reference node architecture. The Risk Monitor observes all execution components independently of the Router.

25 Illustrative Workflows

25.1 Ethereum to XDC Asset Transfer

1. A user submits a transfer intent specifying source asset, XDC destination, amount, maximum fee, and verification requirement.
2. The Ethereum adapter observes the source event after the required finality threshold.
3. The Observer constructs a canonical `XIMMessage` and deterministic `message_id`.
4. The proof builder generates the lane-required evidence (e.g., a light-client or state proof, a ZK proof, or an approved attestation).
5. The message commitment enters a batch and is checkpointed according to deployment policy.
6. The XDC verifier contract validates the evidence and checks lane status, expiry, limits, and replay state.
7. The Executor calls the destination asset or settlement contract.
8. The destination receipt is recorded as **SETTLED**.
9. An acknowledgement is generated and optionally returned to the source application.

25.2 Canton-connected Institutional Workflow

A Canton-connected gateway or adapter observes an authorized event available to the relevant participant. It constructs a XIM commitment without publishing confidential transaction contents. Depending on available Canton integration primitives, the lane may verify a cryptographic

proof, an authorized participant or synchronizer attestation, or a hybrid policy. XIM then routes the verified instruction to XDC or another destination. Sensitive documents and participant-specific details remain within entitled systems; public XIM state contains only the minimum commitments needed for integrity, settlement, and audit.

This workflow deliberately avoids assuming that Canton exposes Ethereum-like public state proofs. The concrete adapter and proof policy must follow the capabilities and security model of the deployed Canton integration [7, 8].

25.3 ISO 20022 to XDC Settlement

1. An institution receives or originates an ISO 20022 payment instruction.
2. An authenticated XIM Gateway validates schema, authorization, duplication status, and institution policy.
3. The gateway creates a `SettlementIntent` and commits to the source financial message hash.
4. The policy engine selects an eligible settlement route, for example native USDC on XDC or another approved rail.
5. Gateway signature and HSM evidence is verified under the bank-to-XIM lane policy.
6. Destination execution occurs and the resulting transaction identifier is bound to the XIM message.
7. Reconciliation output maps XIM settlement status back to the institution’s payment workflow.

26 Formal Invariants

A conforming XIM implementation SHOULD preserve at least the following invariants, where m denotes a message, d a destination security domain, and $\prec$ the lifecycle order of Figure 1.

- I1** *At-most-once destination execution:* $\text{executed}[d, m.\text{message_id}] \leq 1$.
- I2** *Verification before execution:* $\text{EXECUTING}(m) \Rightarrow \text{VERIFIED}(m)$.
- I3** *Expiry:* $\text{now} > m.\text{expiry} \wedge \text{state}(m) \prec \text{EXECUTING} \Rightarrow$ no new execution of m .
- I4** *Commitment integrity:* $\text{accepted}(m) \Rightarrow H(\text{domain_separator} \parallel \text{canonical_encode}(m)) = m.\text{message_id}$.
- I5** *Lane authorization:* $\text{execution}(m) \Rightarrow \text{active}(\text{lane_id}, \text{policy_version})$.
- I6** *Monotonic sequence:* $\text{event}[n+1].\text{sequence} = \text{event}[n].\text{sequence} + 1$.
- I7** *Asset mapping:* $\text{transfer}(m, r) \Rightarrow r$ is authorized for $m.\text{asset_id}$ on the destination.

27 Implementation Plan and Research Roadmap

27.1 Staged Minimum Viable Implementation

Table 8 outlines a staged twelve-week plan for a minimum viable implementation. No performance or security claims in this paper are yet backed by measurements from this implementation; the pilot phase is intended to produce a reproducible benchmark report.

Table 8: Staged twelve-week implementation plan.

Phase	Weeks	Deliverables
Protocol freeze	1–2	XIM v0.1 schema; NetworkID/UAID format; state machine; lane policy model; test vectors.
XDC contracts	2–5	MessageRegistry, VerifierRegistry, RiskManager, replay protection, event model.
First adapters	3–7	XDC and Ethereum observers/executors; deterministic encoding; threshold-attestation verifier as bootstrap.
Commitments and SDK	5–8	SMT library, batch roots, TypeScript/Go SDKs, REST/gRPC API.
Security and operations	7–10	Rate limits, lane pause, key rotation, monitoring, chaos/failure tests.
Pilot	10–12	Testnet Ethereum $\leftrightarrow$ XDC message transfer and one asset-transfer demonstration; benchmark report.

27.2 Research Roadmap

- Replace or supplement bootstrap attestations with light-client or ZK verification on high-value public-chain lanes.
- Formal specification in TLA+ [25] or an equivalent formalism for message lifecycle, replay resistance, timeout, and acknowledgement invariants.
- Formal verification of critical Solidity contracts.
- A Canton adapter feasibility study with an explicit privacy and proof model.
- An ISO 20022 gateway reference mapping and institutional HSM profile.
- A multi-route liquidity and policy engine.
- Cross-domain conditional settlement and delivery-versus-payment research.
- Verkle or alternative vector commitments if proof-size or update characteristics justify migration from SMTs.

28 Security Limitations

XIM cannot make a compromised source ledger truthful. If a source consensus finalizes an invalid economic state according to its own governance, a correct light client will faithfully prove that state. Similarly, attestation-based gateways inherit signer and key-management risk. Cross-chain systems also inherit software risk from adapters, verifiers, destination contracts, and asset issuers. XIM’s objective is therefore not to eliminate trust, but to make trust assumptions explicit, modular, measurable, and isolatable.

Multi-hop routes compound risk. A route is no stronger than the weakest economically relevant edge unless the application uses additional end-to-end verification. High-value routes SHOULD minimize trust-domain transitions.

29 Discussion

XIM defines interoperability around canonical authenticated messages rather than a shared execution environment. This allows heterogeneous networks to retain their native consensus, privacy, governance, and execution models while participating in a common message lifecycle and verification framework. Non-blockchain systems may also participate through explicitly trusted and auditable gateways.

Within the XDC ecosystem, XDC Subnets provide application-specific or permissioned execution environments with configurable governance and privacy. Connectivity between a subnet and the XDC mainnet can be provided through XDC Zero, which supports cross-domain message and asset transfer using relayers, validation components, endpoints, and mainnet checkpointing.

XDC may therefore serve as an optional coordination, checkpointing, and settlement domain for XIM deployments without being a mandatory routing layer. This separation preserves protocol neutrality while allowing applications to use XDC where its settlement characteristics, infrastructure, or ecosystem integrations satisfy the requirements of a particular workflow.

30 Conclusion

XIM proposes a modular interledger messaging protocol for a financial world composed of many independent state machines. The protocol separates canonical message semantics, cryptographic commitments, verification, routing, execution, asset identity, privacy metadata, and risk controls. It supports a spectrum of verification models instead of assuming that one bridge-security model fits every network. By making security assumptions explicit per lane and by minimizing foreign-state replication, XIM aims to provide a practical foundation for public-chain interoperability, institutional tokenized assets, stablecoin settlement, privacy-preserving ledgers, and authenticated connections to traditional financial rails.

The next stage of the work is implementation and measurement. Claims about throughput, latency, security, and operational resilience should be validated through public test vectors, reproducible benchmarks, adversarial testing, audits, and formal analysis before production use.

References

- [1] Quant Network. How Overledger delivers enterprise-grade DLT interoperability for financial services, 2026. <https://quant.network/perspectives/how-overledger-delivers-enterprise-grade-dlt-interoperability-for-financial-services/>.
- [2] Quant Network. DLT interoperability: What are the options? <https://quant.network/research/dlt-interoperability-what-are-the-options/>.
- [3] Chainlink. Cross-Chain Interoperability Protocol (CCIP): Documentation and architecture. <https://docs.chain.link/ccip>.
- [4] Inter-Blockchain Communication Protocol. IBC specifications and documentation. <https://ibcprotocol.dev/>.
- [5] C. Goes. The Interblockchain Communication Protocol: An overview. arXiv:2006.15918, 2020.
- [6] Circle. Cross-Chain Transfer Protocol (CCTP) and Attestation API: Developer documentation. <https://developers.circle.com/>.
- [7] Digital Asset. The Canton blockchain protocol. <https://www.canton.network/protocol>.
- [8] Digital Asset Canton Team. Canton: A Daml based ledger interoperability protocol, 2020.
- [9] Canton Network. Canton Network whitepapers. <https://www.canton.network/whitepapers>.
- [10] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt. SoK: Communication across distributed ledgers. In *Financial Cryptography and Data Security (FC)*, 2021.
- [11] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia. A survey on blockchain interoperability: Past, present, and future trends. *ACM Computing Surveys*, 54(8), 2021.

- [12] M. Herlihy. Atomic cross-chain swaps. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2018.
- [13] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song. zkBridge: Trustless cross-chain bridges made practical. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2022.
- [14] S.-S. Lee, A. Murashkin, M. Derka, and J. Gorzny. SoK: Not quite water under the bridge: Review of cross-chain bridge hacks. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023.
- [15] S. Thomas and E. Schwartz. A protocol for interledger payments, 2015. <https://interledger.org/interledger.pdf>.
- [16] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
- [17] G. Wood. Ethereum: A secure decentralised generalised transaction ledger (Yellow Paper). <https://ethereum.github.io/yellowpaper/paper.pdf>.
- [18] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology – CRYPTO ’87*, 1987.
- [19] R. Dahlberg, T. Pulls, and R. Peeters. Efficient sparse Merkle trees: Caching strategies and secure (non-)membership proofs. In *Nordic Conference on Secure IT Systems (NordSec)*, 2016.
- [20] J. Kuszmaul. Verkle trees, 2018.
- [21] ISO 20022. Universal financial industry message scheme. <https://www.iso20022.org/>.
- [22] G. Wang, J. Wang, L. Lai, and F. Yu. Accountability and forensics in blockchains: XDC consensus engine DPoS 2.0. arXiv:2108.01420, 2021.
- [23] S. Bradner. Key words for use in RFCs to indicate requirement levels. RFC 2119, 1997.
- [24] B. Leiba. Ambiguity of uppercase vs lowercase in RFC 2119 key words. RFC 8174, 2017.
- [25] L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

A Proposed XIM Protobuf Skeleton

```
syntax = "proto3";
package xim.v1;

message XIMMessage {
    uint32 version                = 1;
    bytes  source_network         = 2;
    uint64 source_height          = 3;
    bytes  source_tx_id           = 4;
    bytes  sender                 = 5;
    bytes  destination_network    = 6;
    bytes  receiver               = 7;
    uint32 action_type            = 8;
    bytes  asset_id               = 9;
    bytes  amount                 = 10;
    bytes  payload_hash           = 11;
    uint64 nonce                  = 12;
    uint64 created_at             = 13;
    uint64 expiry                 = 14;
    uint32 finality_policy        = 15;
    uint32 proof_type              = 16;
    bytes  proof_commitment       = 17;
    bytes  compliance_metadata_hash = 18;
    bytes  privacy_policy_hash    = 19;
    bytes  route_policy_hash      = 20;
}
```

B API Sketch

```
POST /v1/intents
{
  "source_network": "ethereum",
  "destination_network": "xdc",
  "source_asset": "UAID:...",
  "destination_asset": "UAID:...",
  "amount": "10000000",
  "beneficiary": "xdc:...",
  "constraints": {
    "max_fee": "500",
    "deadline": 1780000000,
    "verification_class": "institutional"
  }
}

GET /v1/messages/{message_id}
GET /v1/routes/{intent_id}
GET /v1/proofs/{message_id}
GET /v1/assets/{uaid}
```

C Test Vector Requirements

A conforming implementation SHOULD publish test vectors covering:

- the canonical encoding byte sequence for each protocol version;
- the expected `message_id` for each canonical message;
- Merkle leaf, path, and root vectors;

- replay-rejection tests;
- expiry boundary tests;
- policy-version upgrade tests;
- source reorganization and finality tests;
- verifier disagreement and lane-pause tests;
- destination execution failure and compensation tests; and
- cross-language SDK conformance tests.