

Article

Design and Implementation of Verifiable Credentials Management System in Heterogeneous Cross-Chain Environments

Qibing Zhou, Tenghang Li, Yile Jiang and Datian Zhou * 

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China; 23211060@bjtu.edu.cn (Q.Z.)

* Correspondence: dtzhou@bjtu.edu.cn

Abstract

Heterogeneous cross-domain networks are plagued by fragmented trust, data silos, and privacy risks, while conventional single-chain architectures fail to reconcile cross-chain interoperability, regulatory compliance, and commercial privacy. To address these limitations, we present an *Entity–Data–Asset* triple-verification architecture that integrates three key components: Decentralized Identifiers (DIDs) for identity, Verifiable Credentials (VCs) for credentials, and Oracles for cross-chain coordination. Specifically, this architecture enables trustworthy collaboration through three core mechanisms: (1) a **cross-chain identity binding mechanism** based on DIDs that replaces traditional address binding to construct an “**identity-as-access**” trust model; (2) a **collaborative verification paradigm** leveraging VCs and Verifiable Presentations (VPs) to cryptographically link **off-chain verification with on-chain execution**; and (3) an event-driven **Oracle coordination matrix** designed for complex business semantics, supporting **automated and privacy-preserving state synchronization** across asymmetric domains. Experimental evaluation of a prototype integrating Hyperledger Indy and Besu demonstrates a peak Verifiable Presentation batch verification throughput of 0.96 batches/s at $C = 20$, though throughput degrades noticeably under high concurrency due to middleware contention, and an average cross-chain transfer latency of 13.31 s under single-process conditions (mean over 10 iterations). Furthermore, by leveraging this asymmetric design, our architectural optimization strategy—anchoring only cryptographic hashes rather than full credential payloads—reduces the regulatory chain’s storage and gas overhead by approximately 85% compared to traditional full-payload schemes. These results validate the architecture’s feasibility, security, and cost-efficiency for facilitating trustworthy collaboration in complex, heterogeneous ecosystems.

Keywords: verifiable credentials; decentralized identity; cross-chain interoperability; asymmetric topology; Oracle middleware; privacy preservation



Academic Editor: Hamed Taherdoost

Received: 23 March 2026

Revised: 23 June 2026

Accepted: 1 July 2026

Published: 7 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Blockchain and smart contracts are transitioning multi-party collaboration toward automated, traceable execution [1]. However, trusted automation across heterogeneous boundaries faces two primary hurdles: the lack of a unified cross-entity identity framework, and the inherent tension between data privacy and verifiability. Sensitive business data must drive on-chain logic without public exposure. Resolving these issues is vital for the secure circulation of digital credentials in complex scenarios like cross-border trade and supply chain regulation [2,3].

This work presents a systems-level architecture addressing the integration of decentralized identity, selective-disclosure credentials, and cross-chain coordination within regulated environments. Rather than proposing new cryptographic primitives, we focus on orchestrating existing technologies—such as DIDs, VCs, Zero-Knowledge Proofs (ZKPs), and Oracle-based bridges. Synthesizing these components into a unified framework resolves the design tensions among data privacy, auditability, and regulatory compliance.

While conventional blockchains enhance traceability, single-chain architectures suffer from cross-domain interoperability bottlenecks and the rigid coupling of identity management with business logic. This tight coupling means that any modification to access control rules often necessitates complex smart contract upgrades, exacerbating the risk of privacy leaks. Furthermore, inherently pseudonymous addresses fail to capture the real-world semantics—such as legal entity status, organizational roles, and operational qualifications—that are critical for regulatory auditing and Anti-Money Laundering (AML) compliance [4].

To bridge this gap, our system aligns with the European Digital Identity (EUDI) Reference Framework [5] and the eIDAS 2.0 Regulation by decoupling these concerns. We integrate heterogeneous cross-chain architectures with DIDs and VCs [6,7] to distribute identity sovereignty and business execution across purpose-built ledgers. Specifically, a dedicated identity ledger handles DID resolution and credential revocation, while an independent business ledger processes high-throughput transaction logic. By relying on off-chain trusted issuance and lightweight cross-domain verification, the system ensures that only cryptographic commitments or zero-knowledge proofs are anchored on-chain. This workflow strictly enforces the principle of data minimization, effectively mitigating the inherent trade-off between system scalability and the risk of data overexposure.

We validate this architectural design through a functional cross-border trade prototype. By orchestrating Hyperledger Indy and Hyperledger Besu via an event-driven Oracle matrix [8,9], we establish a dual-layer architecture. This approach decouples identity authentication from on-chain settlement logic, providing empirical evidence of its practical utility and data minimization capabilities in multi-ledger environments.

The primary contributions of this paper are as follows:

1. **Cross-Chain Identity Binding Architecture.** We establish a unified Entity-Data-Asset verification workflow mapping real-world entities to identity networks (Hyperledger Indy) and business chains (Hyperledger Besu). Decoupling identity sovereignty from on-chain asset control enables fine-grained, decentralized access management.
2. **Asymmetric Relay Verification Paradigm.** We design and implement a framework that cryptographically links off-chain credential verification with on-chain state execution. By implementing a hash anchoring strategy and asymmetric proof relay, the system shifts intensive cryptographic workloads off-chain while maintaining verifiable state transitions. This approach substantially reduces on-chain storage and gas overhead compared to full-payload schemes, as quantified in Section 4.3.3.
3. **Event-Driven Integration Matrix.** We develop an Oracle-based middleware matrix that coordinates heterogeneous interactions between identity and business layers. This matrix automates on-chain event monitoring, off-chain credential processing, and privacy-preserving state synchronization across asymmetric domains. A cross-border trade prototype integrating Hyperledger Indy and Besu demonstrate the functional correctness and practical feasibility of this coordination mechanism for multi-agent collaboration in complex ecosystems.

2. Background and Related Work

2.1. Blockchain in Supply Chain and Cross-Border Trade

Blockchain enhances supply chain collaboration by providing transparency, traceability, and automated reconciliation [10]. However, complex security and data integrity requirements force architectures to evolve [11]. Today, blockchain forms the critical infrastructure for supply chain finance, logistics, and regulatory auditing.

Hyperledger Fabric suits closed consortiums well. However, its CA/MSP identity model restricts identity portability. Its channel mechanisms strictly target homogeneous networks, severely impeding cross-chain interoperability and credential circulation. Combined with ordering bottlenecks under high concurrency [12], Fabric struggles to support multi-platform scenarios across independent trust domains.

TradeLens exemplifies the pitfalls of single-platform governance. Although it successfully facilitated maritime data sharing, centralized governance and rigid access mechanisms ultimately stifled its expansion [13]. This outcome suggests that a single consortium dominated by a few entities often struggles to provide a neutral, scalable trust infrastructure for cross-domain collaboration.

In summary, single-chain solutions face significant limitations due to functional coupling and the difficulty of decoupling real-world identities from on-chain business states. Relying exclusively on centralized trust can therefore hinder broader cross-domain collaboration. Instead, cross-border trade requires a layered, heterogeneous architecture that isolates identity infrastructure from automated business settlement to ensure privacy and cross-domain interoperability.

2.2. Cross-Chain Interoperability and Oracle Mechanisms

As the blockchain ecosystem fragments, cross-chain interoperability has become essential for multi-chain coordination. Mainstream solutions include atomic swaps, sidechains, relay chains, and standardized messaging protocols [14]. While Hashed TimeLock Contracts (HTLCs) facilitate basic asset exchanges, networks like Polkadot and Cosmos IBC handle broader state synchronization and message passing.

Although these frameworks excel at inter-chain communication, they focus exclusively on the mechanics of data transfer. They lack the unified identity semantics and credential verification required to resolve access authorization, validate qualifications, and execute target-chain logic privately. Consequently, communication mechanisms alone cannot establish a complete trust loop for complex business workflows.

Cross-chain interactions in global supply chains extend beyond simple asset migrations; they demand the rigorous verification of off-chain qualifications, regulatory results, and business credentials. On-chain settlements frequently necessitate prerequisites such as customs clearance or quality inspections. Traditional asset bridges primarily focus on transferring state and value, often lacking the native mechanisms required to process these complex business semantics.

While Oracles traditionally bridge on-chain contracts with off-chain data, mainstream solutions target decentralized finance by processing standardized quantitative feeds [15]. In contrast, integrating non-standardized trade documents requires Oracles to cryptographically verify data provenance, signatures, and entity associations. Consequently, enterprise cross-chain systems require dedicated Oracle infrastructure to coordinate identity verification, credential processing, and on-chain write-backs. The Oracle matrix proposed in this paper is engineered to address this complex requirement.

2.3. Decentralized Identifier and Verifiable Credentials

The Self-Sovereign Identity (SSI) model, underpinned by W3C DIDs and VCs, is increasingly codified within frameworks like **eIDAS 2** to enforce data minimization via **selective disclosure**. While **SD-JWT** provides a mature solution for static attribute sharing, its credential size scales linearly with the number of claims. Furthermore, revealing the exact claim count exposes owners to inference attacks [16]. To address this, **Compact and Selective Disclosure for JWTs (CSD-JWT)** uses a cryptographic accumulator to encode claims into a fixed-length format. This minimizes overhead for resource-constrained devices, conceals the claim count, and enhances privacy [16].

Despite optimizing static attribute sharing, SD-JWT and CSD-JWT lack the expressiveness required for **dynamic predicate** evaluation. For instance, they cannot easily prove a threshold condition without revealing the underlying attribute value. In contrast, the **Camenisch-Lysyanskaya (CL) signature scheme** [17] enables non-interactive zero-knowledge proofs (ZKPs) for complex business logic. This approach shields raw credential attributes, offering superior privacy-preserving adaptability across regulated environments.

To ground our design in established cryptography, the system builds upon the **Direct Anonymous Attestation (DAA)** framework [18]. DAA originally provided anonymous yet authenticatable attestation for hardware modules. It leverages CL signatures to enforce non-revocable anonymity and granular linkability control. The robustness of this approach is well-documented in diverse real-world applications, including privacy-preserving **COVID-19 vaccination certification** [19] and secure **Vehicle-to-Grid (V2G) communications** [20].

We acknowledge that CL-based zero-knowledge proofs are more computationally expensive and produce larger proofs than modern elliptic-curve alternatives. Nevertheless, under the cross-ledger and regulatory constraints considered in this work, CL signatures remain a mature and well-studied foundation based on the strong RSA assumption. Accordingly, our system adapts the DAA-style verification logic to heterogeneous cross-chain coordination using the CL signature scheme that underlies Hyperledger Indy and Anon-Creds. This choice supports selective disclosure and predicate-based proofs required in cross-border trade.

However, current DID and VC research predominantly focuses on isolated identity authentication. Limited attention has been given to linking off-chain verification results with on-chain business execution across heterogeneous ledgers. In cross-border trade, automated workflows require identity credentials and business documents to be cryptographically bound to on-chain asset states. Consequently, standalone identity frameworks alone cannot support the ownership transfer and lifecycle management required by high-value digital assets [21].

Decentralized identity frameworks must also comply with prevailing regulatory requirements. In highly regulated sectors, such as trade finance, Know Your Customer (KYC) and Anti-Money Laundering (AML), compliance is mandatory. Without being anchored to real-world legal entities and recognized auditors, DID credentials cannot serve as valid compliance evidence for customs authorities or financial institutions [22]. Therefore, SSI systems must integrate real-world authentication, off-chain verification, and on-chain execution.

In summary, existing approaches address only parts of the problem considered in this work. Single-chain consortium solutions (e.g., Hyperledger Fabric) tightly couple identity management with platform-specific execution logic and offer limited portability across heterogeneous ledgers. Generic cross-chain interoperability protocols (e.g., Cosmos IBC, Polkadot, and Chainlink-style oracles) support message and state relay but lack unified identity semantics and credential-level verification, and therefore cannot bind off-chain

verification results to on-chain business execution [14]. Standalone DID/VC frameworks (e.g., Sovrin/AnonCreds, did:ethr-based frameworks) provide decentralized identity and selective disclosure, yet they generally stop at the credential layer and do not explicitly connect verifiable presentation verification to cross-chain state transitions in regulated environments.

In contrast, the proposed architecture integrates DID-based cross-chain identity binding, cryptographic linkage of off-chain VP verification to on-chain state, and event-driven Oracle coordination within a single regulated workflow. This systems-level integration—rather than a new cryptographic primitive—constitutes the main contribution of this work. Table 1 further compares these technical routes from the perspectives of trust root, identity binding, interaction mechanism, verification locus, and on-chain cost.

Table 1. Analytical comparison of technological routes in identity expression and cross-domain synergy.

Dimension	Traditional	Consortium	Cross-Chain	DID/VC	Proposed
Trust Root	Central authority	CA/MSP governance	Protocol proofs	Crypto trust	Hybrid: Indy + Besu
Identity	Licenses	X.509 + accounts	Pseudonymous keys	W3C DIDs	DID-address mapping via DIDVerifier
Interaction	Manual	Intra-consortium	Message relay	Off-chain exchange	Oracle matrix: event-driven state sync
Verification	Offline	On-chain endorse	On-chain validate	Off-chain VP	Off-chain check → on-chain trigger
On-Chain Cost	None	High (coupled)	Low (roots)	Minimal (schemas)	Hash anchor only; ~85% Gas reduction
Verification–Execution Link	None	Implicit	None	None	VP receipt triggers on-chain state transition
Business-Semantic Awareness	Manual	Channel-scoped	Generic messages	Credential-level	Credential-aware cross-chain coordination

As shown in Table 1, the novelty of the proposed architecture lies in its explicit integration of identity, credential, and execution layers. Unlike consortium-chain identity models, it decouples legal-entity identity from business-chain accounts through DID-to-address binding. Unlike generic cross-chain relays, its Oracle matrix is credential-aware and can translate VP verification results into executable cross-chain state updates. Unlike standalone DID/VC systems, it closes the verification–execution loop by linking off-chain credential verification to on-chain contract execution across heterogeneous ledgers.

3. Proposed Cross-Chain Verifiable Credential Architecture

Before detailing the architecture, we summarize the integration challenges motivating its design. These challenges arise from integrating independently developed components—DIDs, VCs, ZKPs, and Oracle-based coordination—into a unified system for regulated cross-border trade:

1. **Bridging trust domains:** Connecting heterogeneous networks (Hyperledger Indy and Besu) with incompatible identity models and cryptographic primitives.
2. **Linking verification to execution:** Cryptographically linking off-chain ZKP verification to on-chain state transitions without compromising selective disclosure.
3. **Preventing replay attacks:** Preventing cross-cycle credential reuse in multi-stage workflows.
4. **Ensuring state consistency:** Maintaining cross-chain state consistency under strict privacy constraints.
5. **Reconciling access control:** Aligning DID-based self-sovereignty with EVM-based access control mechanisms.

The following subsections describe how the architecture addresses these challenges.

3.1. Overall Architecture Design

To mitigate trust fragmentation and privacy risks, we propose a cross-chain architecture that decouples identity from business logic via DIDs, VCs, and VPs. The framework pairs Hyperledger Besu for business execution with Hyperledger Indy for identity sovereignty, enforcing an “identity-as-access” paradigm at the identity-verification layer. Cross-domain state coordination, however, currently relies on a consortium trust assumption for Oracle execution, as detailed in Section 3.6.2. A DIDVerifier contract governs on-chain authorization, while off-chain Oracle matrices and ACA-Py agents orchestrate credentials and synchronize cross-network states. As illustrated in Figure 1, the physical topology adopts an asymmetric dual-chain design: Chain A executes business and logistics logic, whereas Chain B handles regulatory oversight. Logically, the architecture comprises six modular layers connecting heterogeneous ledgers to application interfaces.

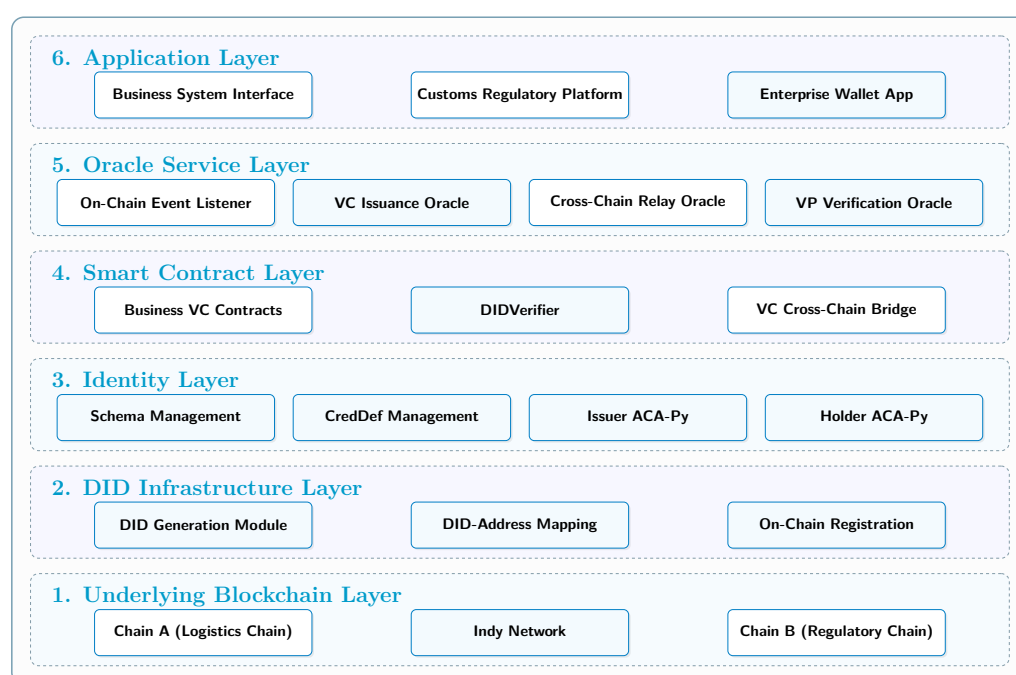


Figure 1. Overall dual-domain layered architecture of the proposed system.

Detailed Description of the Six-Layer Logical Architecture

Based on the aforementioned overall design, the system logic is further abstracted into six decoupled functional layers from bottom to top. Each layer collaborates efficiently through standardized interfaces to jointly support the lifecycle management of Verifiable Credentials, from generation and transmission to cross-chain verification.

Layer 1: Underlying Blockchain Layer. As the system’s operational foundation, this layer integrates two specialized execution chains with the Hyperledger Indy network. Chain A is dedicated to business and logistics execution, including credential anchoring. Chain B acts as the regulatory and compliance hub, which receives proofs/receipts and triggers release or settlement logic. The Hyperledger Indy network underpins the entire ecosystem by maintaining a consistent root of trust for DIDs, schemas, and credential definitions.

Layer 2: DID Infrastructure Layer. This layer aims to construct a self-sovereign identity framework and maintain the strong binding relationship between DIDs and on-chain accounts. It derives DIDs and their associated public keys through a deterministic generation mechanism and registers the mapping logic into the DIDVerifier contract on the business

chain, thereby achieving the separation of identity authentication semantics and asset control logic at the architectural level.

Layer 3: Identity Layer. Composed of the ACA-Py agent cluster and credential management components, this layer provides full-lifecycle services for Verifiable Credentials. By deploying Issuer and Holder agent nodes, the system can abstract real-world trade documents—such as quality inspection and insurance certificates—into programmable, structured digital credentials based on predefined Schemas and credential definitions.

Layer 4: Smart Contract Layer. This layer is responsible for the on-chain formal expression and execution of business rules, implementing an asymmetric deployment strategy. Chain A deploys a complete suite of business Verifiable Credential contracts (including quality inspection, insurance, certificate of origin, and bill of lading contracts) to achieve multi-source data anchoring. In contrast, Chain B only retains the cross-chain bridge contract and the DIDVerifier contract, focusing exclusively on cross-chain message reception and access authentication.

Layer 5: Oracle Service Layer. This layer operationalizes the event-driven coordination model described in Section 3.4, bridging on-chain contracts and off-chain identity agents. Detailed interaction flows are provided later and are not duplicated here.

Layer 6: Application Layer. This layer directly interfaces with end-users and external ecosystems. It specifically includes customized pre-entry API interfaces for injecting business data, enterprise wallet applications for credential management and Verifiable Presentation submission, and customs regulatory platforms for processing compliance directives. User requests are distributed through this layer, driving the underlying modules to complete complex identity verification and cross-chain state migrations.

This six-layer architecture decouples blockchain networks, identity infrastructure, smart contracts, and middleware services. It provides credibility at the base, identity and credential management in the middle, and automated business collaboration at the top. The result is a scalable cross-chain VC management framework.

3.2. Cross-Chain Identity Binding and Access Control Mechanism

To address the lack of real-world legal entity semantics in pseudonymous blockchain addresses, we design a cross-chain identity binding mechanism. This mechanism establishes a verifiable triple-mapping among real-world entities, identity chain identifiers, and business chain accounts, laying the foundation for an identity-as-access trust model.

3.2.1. Deterministic Identity Anchoring and Address Binding

In permissioned chains, pseudonymous addresses lack direct mapping to legal entities. To ensure regulatory traceability, the system implements identity anchoring through deterministic generation and on-chain access control.

Deterministic Identity Generation: An Ed25519 key pair controls the DID in the Indy identity layer, while a secp256k1 ECDSA key pair signs Besu/EVM transactions. A DID-to-address binding is registered on-chain to link both identities.

Off-Chain Mapping and On-Chain Access Control: An off-chain Oracle maintains DID-to-address mappings, while the on-chain DIDVerifier contract authenticates initiators before execution. This dual mechanism mitigates Sybil attacks by restricting cross-chain operations to certified entities, balancing efficiency with compliance.

3.2.2. Dual-Layer Access Control Based on DID and Role-Based Access Control

To address the limitations of traditional address-based whitelisting, which lacks identity semantics, we implement a dual-layer access control framework combining W3C DIDs with Role-Based Access Control.

Layer 1: Identity Admission and Holder Consistency. Smart contract modifiers verify that the caller is bound to a valid DID by checking whether `msg.sender` maps to a registered DID in the DIDVerifier registry. For credential-specific operations, the caller's DID hash is compared with the authorized holder's DID hash. Any mismatch reverts the transaction, restricting state changes to the legitimate holder.

Layer 2: Role Control for Privileged Operations. Administrative functions such as cross-chain relaying and proof submission are governed by RBAC. Critical operations, including verification receipt submission and credential digest anchoring, are restricted to authorized Oracle nodes with designated roles (e.g., `VPVerifier_Role`).

Together, these two layers enforce identity binding and role authorization directly at the contract level. By requiring verifiable decentralized identities for contract interaction, the design improves resistance to Sybil-style misuse and supports auditable coordination among multiple agents in heterogeneous environments. However, overall system security still depends on the user's secure off-chain management of cryptographic credentials.

3.3. Collaborative Model: Off-Chain Verification and On-Chain Execution

We link off-chain verification with on-chain execution to resolve computational bottlenecks and privacy leakage. **Verifiable Credentials** and **Verifiable Presentations** serve as trust carriers, shifting cryptographic workloads off-chain while maintaining verifiable state transitions on-chain.

3.3.1. Dual-Domain Trust and Triple Decoupling

The architecture distributes operations across the identity sovereignty domain (anchored by Indy) and the business execution domain (managed by Besu). Under a consortium trust model, on-chain hashes anchor credential and transaction integrity, while cross-domain state synchronization depends on honest Oracle execution.

- **Identity Sovereignty Domain:** Hyperledger Indy network maintains the identity trust root. Participants register DIDs and obtain **Verifiable Credentials** issued under predefined Schemas and Credential Definitions. This domain establishes identity and compliance prerequisites for business execution.
- **Business Execution Domain:** Hyperledger Besu manages on-chain state migrations, including asset locking, credential digest anchoring, and cross-chain synchronization. It ensures traceable state migrations and auditable cross-chain results.

Instead of sharing full business data, the two domains coordinate via credential digests, on-chain events, and DID bindings. State migration thus depends on the consistency between off-chain verification and on-chain anchoring, eliminating redundant identity data. To meet the performance, privacy, and compliance demands of cross-border trade, the system implements a triple-decoupling architecture:

- **Identity and Assets:** Besu addresses are used only for transaction signing and asset operations, while business entities are represented via DIDs. The DIDVerifier contract maintains this address-to-DID mapping.
- **Verification and Execution:** To eliminate on-chain verification costs for cryptographic proofs [17], the architecture offloads these processes to ACA-Py agents and a VP verification Oracle. Consequently, the blockchain records only verification receipts and credential digests. As demonstrated in Section 4, the Oracle matrix manages

concurrent workloads of up to 160 proofs per cycle. This approach substantially reduces on-chain storage and gas consumption compared to full-payload anchoring, as quantified in Section 4.3.3.

- **Privacy and Auditing:** Sensitive identity attributes and trade documents remain off-chain. Zero-knowledge proofs enable compliance verification without revealing full credential content [5,23].

3.3.2. Asymmetric VC Anchoring and Zero-Knowledge Proof Computation Offloading

We adopt an asymmetric verification strategy to prevent data leakage and state bloat. This strategy combines off-chain issuance, source-chain anchoring, and lightweight destination-chain verification. Notably, “lightweight” refers to minimal on-chain storage, not reduced cryptographic security. The system anchors only VC hashes on the source chain and relays compact proof fragments to the destination. This eliminates plaintext exposure and minimizes storage overhead. As formalized in Algorithm 1, this approach enables stateless verification at the destination bridge. A reverse compensation mechanism preserves eventual consistency. Upon verification failure, a rejection receipt triggers an automated rollback on the source chain.

Algorithm 1 Asymmetric Anchoring with Lightweight Cross-Domain Verification and Rollback

Require: R : Cross-domain request, $\mathcal{A}_I$: Issuer Agent, $\mathcal{O}$: Oracle Middleware

Require: $\mathcal{C}_S$: Source chain contracts, $\mathcal{C}_T$: Target chain bridge

Ensure: S : Final execution status of the transaction

```

1: // Phase 1: Asynchronous Delivery & Source-Chain Anchoring
2:  $id_{vc} \leftarrow \text{GenerateUUID}()$  {Inject tracking identifier into attrs}
3:  $credExId \leftarrow \mathcal{A}_I.\text{OfferCredential}(\text{holder DID}, \text{attrs} \cup \{id_{vc}\})$ 
4: wait until  $\text{VerifyDelivery}(\mathcal{O}, \text{holder DID}, id_{vc})$  {Ensure secure off-chain storage}
5:  $h_{vc} \leftarrow \text{Keccak256}(\mathcal{A}_I.\text{GetCredential}(credExId))$ 
6:  $\mathcal{C}_S.\text{AnchorMetadata}(h_{vc}, meta)$ 
7: // Phase 2: Asymmetric Relay & Lightweight Verification
8: loop
9:    $e \leftarrow \text{Listen}(\mathcal{C}_S, \text{CrossChainEvent})$ 
10:  if  $e.h_{vc} == h_{vc}$  then
11:     $proof \leftarrow \text{ParseAndSign}(e, DID_{\mathcal{O}})$  {Generate lightweight proof fragment}
12:    // Target chain RBAC-based authorization
13:    if  $\mathcal{C}_T.\text{VerifyRBAC}(DID_{\mathcal{O}}, proof)$  is Valid then
14:       $\mathcal{C}_T.\text{Execute}(\pi_{proof}, h_{vc})$ 
15:    else
16:       $\mathcal{C}_S.\text{Rollback}(h_{vc})$  {Trigger reverse compensation}
17:    end if
18:    break
19:  end if
20: end loop

```

To implement the protocol in Algorithm 1, the architecture adopts an *off-chain computation offloading* paradigm. This approach shifts resource-intensive cryptographic evaluations away from the Ethereum Virtual Machine (EVM). The implementation translates the protocol into a deterministic, multi-stage reactive workflow. First, the system monitors events and resolves metadata asynchronously to extract credential digests and participant identifiers. Second, a subsequent *Entity–Data–Asset* verification validates cryptographic integrity, schema compliance, and identity-to-address binding off-chain. Finally, the workflow culminates in the generation of the Oracle signature σ_{Oracle} and the submission of a lightweight boolean receipt to the destination chain. This process adheres to the “off-chain computation, on-chain receipt” model described in Section 3.3.1.

Formal Definition of the Proof Relay. During state synchronisation (Algorithm 1, Line 11), the Oracle relays a proof fragment

$$proof = \{h_{vc}, meta, N, \sigma_{Oracle}\},$$

where h_{vc} and $meta$ are extracted from the source-chain event e , and N is a nonce for freshness and replay resistance. The Oracle signs

$$\sigma_{Oracle} = \text{Sign}_{sk_{Oracle}}(\mathcal{H}(h_{vc} || meta || N)).$$

This signature binds the relayed metadata to the Oracle and provides integrity and provenance.

Event-Driven Verification Workflow. For each incoming Verifiable Presentation, the Oracle matrix performs:

1. **Entity verification:** checks the presenter's DID signature and registration status.
2. **Data verification:** verifies credential validity, schema/issuer constraints, and cryptographic integrity via the Indy stack.
3. **Asset/state verification:** evaluates whether disclosed attributes satisfy the destination contract conditions.

After verification, authorized Oracle nodes submit a boolean receipt to the destination bridge contract to trigger the corresponding state transition.

3.3.3. Security Responsibility Boundaries: Cryptographic vs. Architectural Guarantees

We distinguish between security properties that are enforced by cryptographic primitives—and thus hold independently of system architecture—and those that depend on correct architectural execution under consortium-level trust assumptions. To define the system's trust boundaries, we partition security responsibilities between the cryptographic and architectural layers. This prevents overestimating the guarantees provided solely by the zero-knowledge proof protocol.

Cryptographic Layer: VC and VP Guarantees. Rooted in the Camenisch-Lysyanskaya signature scheme, a Verifiable Credential provides two native guarantees: *unforgeability* (the issuer's signature covers all attributes and the `cred_def_id`, invalidating any modification) and *holder binding* (the VC is cryptographically tied to the holder's DID). A Verifiable Presentation extends these guarantees via an AnonCreds non-interactive ZKP, proving: (i) possession of a valid VC, (ii) the authenticity of the selectively disclosed attributes, and (iii) binding to a single-use challenge nonce to prevent replay attacks. These cryptographic properties hold independently of the system architecture.

Architectural Layer: Oracle, DID, and Contract Enforcement. Conversely, several security properties cannot be derived from the ZKP protocol alone and depend entirely on the correct execution of the system architecture.

- **UUID-based credential binding.** A valid VP alone does not confirm that the credential belongs to the current business cycle. Each VC therefore embeds a UUID as the `contractName` attribute, anchored on-chain at issuance (Algorithm 1, Line 6). During VP verification, the Oracle compares the disclosed `contractName` against the `expected_uuid` from the source-chain contract; any mismatch triggers immediate rejection.
- **On-chain digest anchoring.** As formalized in Section 3.3.2, only the credential hash (h_{vc}) and minimal metadata are stored on-chain, while all plaintext attributes remain off-chain.

- Credential format as a security prerequisite. The above guarantees require credentials to conform to pre-registered Indy schemas. Specifically, `contractName` must match a live on-chain UUID record, `cred_def_id` must reference an authorized credential definition, and the issuer DID must appear in the trust registry. Non-conforming presentations are rejected at the Oracle's proof request gate before any ZKP evaluation.
- Access control and role enforcement. The `DIDVerifier` contract restricts state-modifying operations to addresses with bound DIDs, while the `VPVerifier_Role` RBAC mechanism limits Oracle write-backs to authorized nodes.

Table 2 summarizes this partitioning, mapping each security property to its responsible layer and concrete enforcement mechanism.

Table 2. Security responsibility partition: cryptographic layer versus architectural layer.

Security Property	Guaranteed By	Mechanism
VC attribute unforgeability	Cryptographic	CL signature (Strong RSA Assumption)
Selective disclosure/ZK	Cryptographic	AnonCreds ZKP protocol
Cryptographic anti-replay	Cryptographic	Single-use challenge nonce in VP
Holder binding	Crypto + Architectural	ZKP linkage + <code>DIDVerifier</code> contract
UUID-credential binding	Architectural	Oracle <code>contractName</code> matching
On-chain data minimization	Architectural	Hash-only anchoring (h_{vc})
Access control	Architectural	<code>DIDVerifier</code> + RBAC modifiers
Cross-chain state integrity	Architectural	Oracle signature σ_{Oracle}

Note: Properties above the first divider are mathematically guaranteed by cryptographic primitives under the Strong RSA Assumption, and hold regardless of Oracle behavior or consortium governance. Properties below the divider are contingent on correct architectural execution and the consortium-level trust assumptions defined in Section 3.6.2 (Classes C–D); they do not follow from the ZKP protocol alone.

3.4. Event-Driven Oracle Coordination Mechanism for Complex Business Semantics

Bridging the identity domain and the business execution domain requires a robust middleware capable of understanding complex trade semantics. We deploy an event-driven Oracle matrix working alongside ACA-Py agents to coordinate on-chain event monitoring, off-chain credential issuance, and privacy-preserving cross-chain synchronization.

3.4.1. Identity Agent and Oracle Matrix Architecture

To resolve the “Oracle problem”—where on-chain contracts lack direct access to off-chain data—a middleware layer comprising an Oracle matrix and ACA-Py agents bridges Hyperledger Besu and Indy. Event-driven decoupling automates this collaboration: on-chain states trigger off-chain credential services, while verification results govern on-chain asset circulation. As illustrated in Figure 2, Hyperledger Indy network maintains the global trust root, and ACA-Py agents manage credential interactions. Simultaneously, the Oracle monitors on-chain events and synchronizes cross-chain states, mapping DIDs to blockchain accounts.

Identity Agent Design: The system introduces Aries Cloud Agent Python(ACA-Py) as the execution engine for the Verifiable Credential protocol and constructs a dual-agent division model to achieve duty isolation:

- **Issuer Agent:** Operating as the system-level trust anchor, it represents authoritative institutions to maintain credential definitions and issue W3C-compliant Verifiable Credentials. It executes credential issuance only when business conditions triggered by the Oracle are met.
- **Holder Agent:** Simulating the digital identity wallet, it is responsible for securely receiving, verifying, and storing Verifiable Credentials. It incorporates an Auto-Respond

mechanism to ensure efficient end-to-end credential delivery in asynchronous HTTP environments and generates Verifiable Presentations during the verification phase.

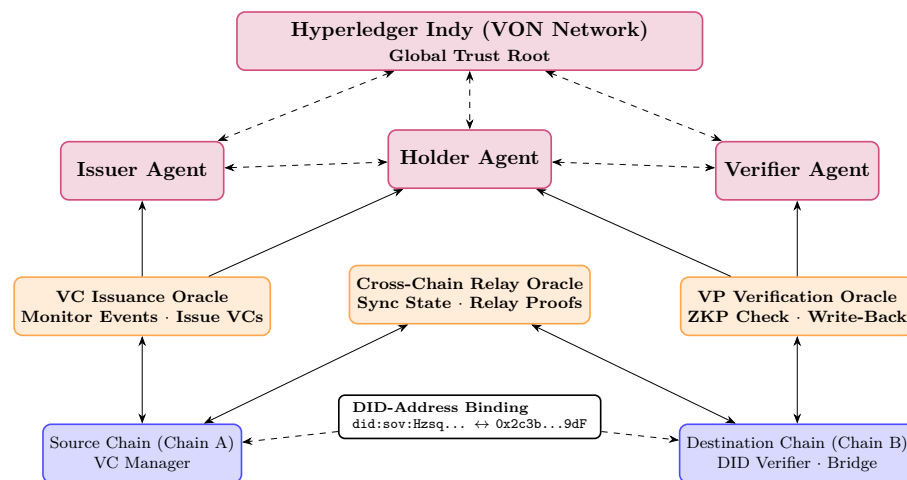


Figure 2. Oracle and ACA-Py Collaborative Architecture with DID-to-Address Binding.

Oracle Service Matrix: This matrix employs a parallel-processing architecture in which multiple authorized nodes handle requests concurrently. It consists of three core components:

- **VC Issuance Oracle:** Monitors off-chain requests, triggers Issuer Agents to generate **Verifiable Credentials**, and anchors their digests to the source-chain VCManager contract.
- **Cross-Chain Relay Oracle:** Synchronizes metadata across heterogeneous networks by extracting anchored Verifiable Credential digests and relaying them to destination bridge contracts via lightweight cryptographic proofs.
- **VP Verification Oracle:** Validates **Verifiable Presentations** against anchored digests. It uses zero-knowledge proofs to verify attributes privately and submits verification receipts to destination bridge contracts to authorize state transitions.

3.4.2. Cross-Domain Dynamic Interaction and Anti-Replay Mechanism

Based on the Oracle matrix and dual-agent architecture, the cross-chain business process in the system can be abstracted into a four-phase dynamic interaction lifecycle, as illustrated in Figure 3.

1. **Phase 1: Identity Preparation and Binding.** Traders register DIDs and obtain qualification credentials via the Issuer Agent. The system then maps these DIDs to blockchain addresses within the DIDVerifier contract to authorize on-chain access.
2. **Phase 2: Business Request, Credential Issuance, and Delivery.** A source-chain business request triggers an on-chain event. The VC Issuance Oracle captures this event, directing the Issuer Agent to initiate issuance. Upon confirmed delivery to the Holder's wallet, the Oracle hashes the credential and anchors the digest to the source chain.
3. **Phase 3: Cross-Chain Relay and State Synchronization.** The Cross-Chain Oracle synchronizes the anchored VC digest from the source to the destination chain. It transmits only lightweight cryptographic proofs rather than full credential plaintext.
4. **Phase 4: Presentation Verification and State Confirmation.** The user submits a Verifiable Presentation for off-chain Oracle validation. Upon successful verification, the

Oracle submits a receipt to the destination bridge contract, completing the compliance clearance and state update.

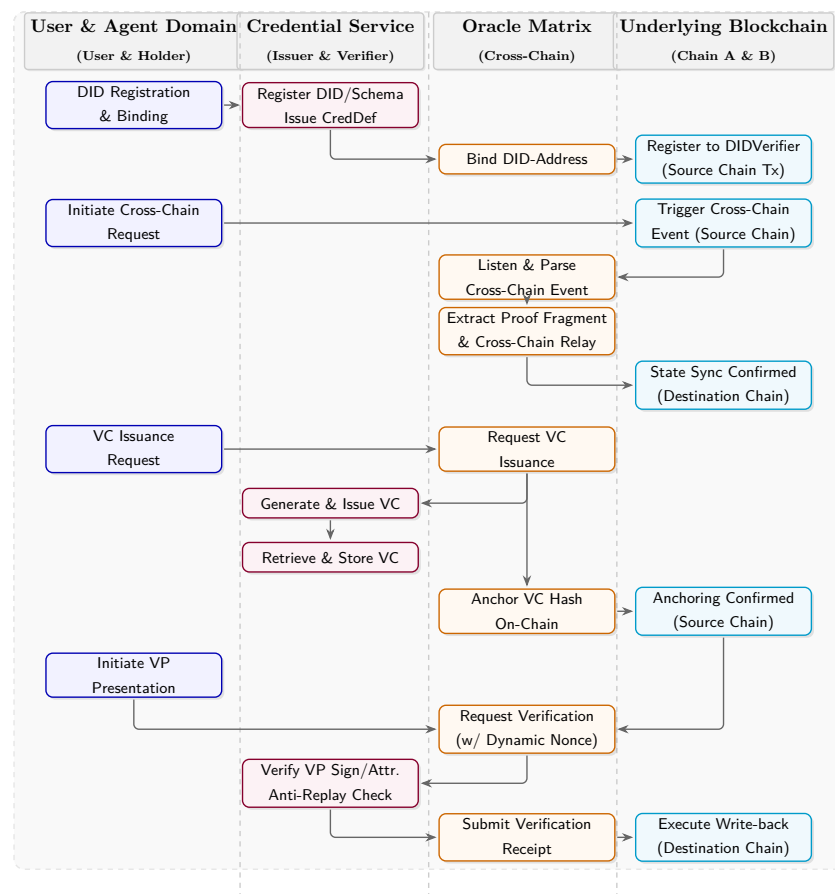


Figure 3. Dynamic interaction lifecycle and cross-chain data flow across multiple domains.

To achieve precise attribute verification without exposing plaintext and to mitigate interception risks during these dynamic interactions across open networks, the system employs a dynamic Challenge-Response model featuring two core security mechanisms:

- **Cryptographic Primitives and Security Assumptions:** Our prototype employs Anon-Creds (based on Camenisch–Lysyanskaya signatures) under the **Strong RSA Assumption**. While proof size typically scales linearly (2–4 KB) and standalone verification takes tens of milliseconds, the observed batch latencies (2.37–3.72 s) are dominated by ACA-Py middleware overhead—including HTTP handling, event-loop scheduling, and database contention—rather than the cryptographic verification process itself.
- **Dynamic Nonce-Based Anti-Replay:** To prevent replay attacks during cross-domain transmission, each destination-chain proof request includes a fresh challenge nonce and an expiry timestamp. The submitted **Verifiable Presentation** is cryptographically bound to this nonce, ensuring single-use validity within a limited time window. Upon receipt, the VP Verification Oracle checks nonce freshness, timestamp validity, and nonce uniqueness before accepting the proof and writing back the verification receipt on-chain. The detailed procedure is formalized in Algorithm 2.

Algorithm 2 Anti-Replay Verification Protocol for Cross-Chain VP Presentation**Input:** Holder agent $\mathcal{H}$, VP Verification Oracle $\mathcal{V}$, target chain bridge $\mathcal{C}_T$ **Input:** *UsedNonces*: set of consumed nonces maintained by $\mathcal{V}$ **Output:** *accept* / *reject***Phase 1: Challenge Generation (Verifier side)**

1. $c_{\text{nonce}} \leftarrow \text{GenerateRandomNonce}()$
2. $t_{\text{expire}} \leftarrow \text{CurrentTimestamp}() + \Delta_{\text{max}}$
3. $\text{proofReq} \leftarrow \{\text{requested_attributes}, \text{restrictions:}\{\text{cred_def_id}, \text{issuer_did}\}, \text{nonce: } c_{\text{nonce}}, \text{expiry: } t_{\text{expire}}\}$
4. Send *proofReq* to $\mathcal{H}$

Phase 2: Presentation Generation (Holder side)

5. $vp \leftarrow \text{AnonCreds.CreatePresentation}(\text{credential}, \text{proofReq}, \text{link_secret})$
6. Send *vp* to $\mathcal{V}$

Phase 3: Anti-Replay Verification (Oracle side)

7. **if** $\text{CurrentTimestamp}() > t_{\text{expire}}$ **then return reject**
8. **if** $c_{\text{nonce}} \in \text{UsedNonces}$ **then return reject**
9. **if** $\text{AnonCreds.VerifyPresentation}(vp, \text{proofReq}) = \text{false}$ **then return reject**
10. $\text{UsedNonces} \leftarrow \text{UsedNonces} \cup \{c_{\text{nonce}}\}$
11. $\text{receipt} \leftarrow \{v_{\text{bool}}: \text{true}, h_{vc}, \text{UUID}, t_{\text{verify}}, \sigma_{\text{Oracle}}\}$
12. $\mathcal{C}_T.\text{SubmitReceipt}(\text{receipt})$
13. **return accept**

3.5. Prototype System Implementation**3.5.1. Infrastructure Deployment**

The prototype integrates a Hyperledger Besu network and a Hyperledger Indy network as mentioned in the previous section. At the application layer, ACA-Py agent nodes facilitate trusted cross-chain interactions between smart contracts and the identity network via an event-driven Oracle matrix.

3.5.2. Use of GenAI Tools in Manuscript Preparation

DeepSeek V3 (DeepSeek, 2025) and Gemini 3.1 Pro (Google, 2025) were used during the preparation and revision of this manuscript solely for translation assistance, English polishing, grammar checking, and reducing repetitive expressions. These tools were used only to refine author-written text for readability and clarity, and were not involved in the research design, architecture development, algorithm design, smart contract implementation, experiments, data analysis, or scientific conclusions. All technical content was independently developed and verified by the authors, who reviewed and edited all GenAI-assisted text and took full responsibility for the content of this publication.

3.5.3. Engineering Interfaces of the Oracle Matrix

The Oracle matrix serves as the core middleware bridging on-chain states with off-chain verification. It implements the three logical roles described in Section 3.4.1 (issuance, relay, and verification) via modular Python services. Each service executes a unified, asynchronous pipeline:

Event Listening $\rightarrow$ Off-chain Execution $\rightarrow$ On-chain Write-back

To ensure interoperability between standard Ethereum and Hyperledger Besu environments, the Oracle matrix utilizes the `web3.py geth_poa_middleware`. This middleware normalizes Besu's extended block-header format—specifically the validator-related `extraData` field—at the client-side connection layer. This adaptation allows Oracle services to reliably subscribe to contract events and perform cross-chain transactions across heterogeneous EVM-compatible ledgers, without requiring architectural assumptions regarding the underlying consensus mechanism.

3.5.4. Data Models and Schemas for Verifiable Credentials and Presentations

We define four Verifiable Credential schemas on the Hyperledger Indy network to support cross-border customs clearance: (i) Inspection Report (exporter, importer, product-Batch, etc.), (ii) Insurance Contract (policyNumber, coverage, etc.), (iii) Certificate of Origin (origin, manufacturer, etc.), and (iv) Bill of Lading (shipper, consignee, shipmentID, etc.). Each schema specifies plaintext attributes required for business operations.

On-chain, only minimal metadata is recorded per credential: a 32-byte keccak256 digest h_{vc} , holder DID, and UUID. Plaintext attributes remain off-chain, and attribute-level verification is delegated to the VP Verification Oracle via AnonCreds. The UUID acts as a cryptographic link between off-chain verification and on-chain state, as detailed in Section 3.3.3.

3.6. Security and Privacy Analysis

To systematically evaluate the security of the proposed heterogeneous cross-chain credential management system, this study defines the multi-party interaction environment and potential threats based on the security evaluation framework for decentralized identity proposed by prior studies [24].

3.6.1. Formal System Model

To formalize the heterogeneous cross-chain environment and ground our security analysis in the actual implementation, we define the system model as follows:

Definition 1 (Heterogeneous Blockchain Network): Let the system comprise two heterogeneous ledgers, B_1 (Chain A) and B_2 (Chain B). Each ledger is defined as a tuple $B_i = (\mathcal{L}_i, \mathcal{C}_i, \mathcal{S}_i, \mathbb{C}_i)$, where $\mathcal{L}_i$ is the ledger state (e.g., cross-chain account mappings), $\mathcal{C}_i$ represents the consensus mechanism (e.g., IBFT 2.0 with a 12-second block time), $\mathcal{S}_i$ denotes the global state, and $\mathbb{C}_i$ is the set of deployed smart contracts. To handle ledger heterogeneity, cross-chain interactions with B_i are encapsulated via an injected consensus middleware.

Definition 2 (Cross-Chain Entity & Metadata): A Verifiable Credential metadata anchored on-chain is formalized as $vc = (h, \text{name}, \text{endpoint}, \text{did}, \text{addr}, t_{exp}, \text{exist})$, directly mapping to the on-chain data structure. Here, $h \in \{0, 1\}^{256}$ is the keccak256 digest ensuring cross-chain data tamper-resistance, and $\text{addr} \in \{0, 1\}^{160}$ represents the 20-byte EVM address tightly bound to the W3C standard *did*.

3.6.2. Adversary Model and Explicit Trust Assumptions

To systematically evaluate the security of the proposed system, we explicitly categorize the trust assumptions underlying our architecture and define the corresponding threat model. This approach clarifies the boundaries between cryptographically guaranteed security and architecture-dependent trust.

Explicit Trust Assumptions and Limitations. We categorize the system's operational and architectural trust assumptions into four distinct classes:

1. **A-Class: Cryptographic Assumptions (Standard).** We assume the collision resistance of the keccak256 hash function and the existential unforgeability of the Ed25519 and Camenisch-Lysyanskaya signature schemes.
2. **B-Class: System Component Assumptions (Verified).** We assume the DIDVerifier smart contract correctly maintains the bidirectional mapping between DIDs and EVM addresses, which is enforced programmatically via on-chain logic.
3. **C-Class: Operational Assumptions (Infrastructure).** We assume that the private keys of the Oracle nodes (sk_{Oracle}) are securely stored, configuration files are protected by OS-level access controls, and network communication is secured via TLS.
4. **D-Class: Design Limitations (Consortium Trust).** The current prototype lacks a formal Byzantine Fault Tolerance (BFT) consensus layer. Instead, the Oracle matrix relies on a consortium trust assumption. It uses individual digital signatures, σ_{Oracle} , computed over relayed metadata:

$$\text{Sign}_{sk_{Oracle}}(\mathcal{H}(h_{vc} \parallel meta \parallel N))$$

This design assumes that authorized nodes act honestly. In permissionless environments, this introduces significant collusion risks. Because of this single-Oracle design, cross-chain state integrity and UUID-based credential binding are not cryptographically provable. While the architecture enforces these properties, it lacks formal mathematical guarantees. In contrast, cryptographic primitives independently ensure VC unforgeability and zero-knowledge privacy. Without a BFT consensus layer, Byzantine Oracles can compromise the system: they might submit false verification receipts, censor cross-chain requests, or inject conflicting state transitions across the asymmetric dual-chain topology.

The Trust Transition Bottleneck: While off-chain VP verification provides strict cryptographic guarantees (Properties 1–3, Table 2), the destination chain does not verify the ZKP directly, relying instead on the Oracle's signature σ_{Oracle} . This marks a critical transition from “mathematically proven” (ZKP domain) to “socially trusted” (Oracle domain). A malicious Oracle could authorize fraudulent state transitions (e.g., $v_bool: true$) without on-chain detection, reducing end-to-end security to Oracle honesty—a single point of failure in open networks. This trade-off is **intentional**: in regulated cross-border trade, entry is gated and participants are legally accountable. For future public deployments, BFT-based consensus or TEE-assisted verification will be essential to resolve this bottleneck.

Attacker Types and Attack Vectors. Building upon the established trust model, we formalize the attack vectors by categorizing potential adversaries into three distinct profiles. These profiles reflect both network-level vulnerabilities and architectural limitations in the heterogeneous execution environment:

1. **Network-Level External Adversaries:** Entities attempting to intercept, tamper with, or replay cross-domain verification fragments during asynchronous off-chain transmission.
2. **Malicious Credential Holders:** Internal participants attempting to exploit the asynchronous state synchronization window (e.g., the empirically measured 13.31-second cross-chain latency) to bypass credential revocation or execute unauthorized cross-cycle authorizations.
3. **Semi-Honest Infrastructure Operators:** Oracle nodes that strictly follow the protocol logic but passively attempt to infer privacy-sensitive attributes from the selective-disclosure process, or theoretically collude to inject conflicting state receipts.

To bridge the gap between theoretical security modeling and engineering implementation, Table 3 delineates the correspondence among the identified threats, their architectural mitigation strategies, the concrete code-level enforcement logic, and the current validation status.

Table 3. Mapping of Threat Models to Mitigation Strategies and Implementation Status.

Threat ID & Description	Mitigation Strategy	Code-Level Enforcement	Validation Status
T1: Cross-chain Replay	Cryptographic nonce & Cache	Oracle deduplication cache and state-bound c_{nonce}	Empirically Verified
T2: Unauthorized Access	On-chain DID-based RBAC	onlyAllowedOracleDID modifiers in bridge contracts	Empirically Verified
T3: Data Tampering	Hash-only State Anchoring	32-byte keccak256 digest integrity evaluation	Empirically Verified
T4: Reentrancy Attack	Secure Execution Pattern	Checks-Effects-Interactions design in Solidity	Code Audited
T5: Man-in-the-Middle	Transport Layer Security	TLS encryption over HTTP/JSON-RPC interfaces	Env-Dependent
T6: Oracle Collusion	Multi-Oracle Consensus	Threshold signatures or BFT-based voting mechanisms	<i>Under Experimentation</i>
T7: Sybil Attack	Deterministic Admission	On-chain DID-to-address mapping via smart contracts	Empirically Verified

3.6.3. Security Properties and Analysis

These security properties span two responsibility layers. Properties 1 and 3 rely on the underlying cryptographic primitives, operating entirely independent of the system architecture. Property 2 combines cryptographic binding with smart contract execution. Property 4 relies jointly on dynamic nonces (cryptographic layer) and Oracle UUID matching (architectural layer). Table 2 summarizes these attributions.

Table 4 details the concrete proof mechanisms in our prototype. Credential authenticity relies on the issuer's CL signature, verified via AnonCreds. Issuer authorization is achieved by binding the proof request to a specific `schema_id`, `cred_def_id`, and `issuer_did` from the Indy ledger. Holder binding follows the AnonCreds Link Secret model: assuming the Link Secret remains confidential, only the legitimate holder can generate a valid presentation. The verification policy for each VC type is instantiated as $P_T = (A_{\text{reveal}}, R_T, c_{\text{nonce}})$, where A_{reveal} is the revealed attribute set, R_T denotes issuer restrictions, and c_{nonce} is the single-use nonce for anti-replay. Predicate and range proofs supported by AnonCreds are reserved for future implementation. Credential revocation remains a known limitation, discussed in Section 3.6.4.

Table 4. Specific cryptographic proof mechanisms for each security property.

Security Property	Specific Mechanism	Status
Credential authenticity	CL signature over all attributes and <code>cred_def_id</code>	Implemented
Issuer authorization	<code>schema_id</code> + <code>cred_def_id</code> + <code>issuer_did</code> restrictions verified against Indy ledger	Implemented
Holder binding	Link Secret (m_1) in CL signature; knowledge proven via ZKP	Implemented
Equality proof	<code>contractName</code> UUID matching via proof request filter	Implemented
Predicate / range proof	Supported by AnonCreds <code>requested_predicates</code> ; reserved for future extension	Planned
Revocation proof	Accumulator-based non-membership proof; identified as a known limitation and left for future work	In progress
Anti-replay	Single-use c_{nonce} bound into Fiat-Shamir transform per session	Implemented
Anchor membership	Direct on-chain mapping $O(1)$; Merkle proof not required by design	By design
Oracle receipt	$\{v_{bool}, h_{vc}, \text{UUID}_{disclosed}, t_{verify}, \sigma_{Oracle}\}$	Implemented

1. **Unforgeability:** The unforgeability of Verifiable Credentials is rooted in the cryptographic signatures produced by the authorized Issuer. Because any modification to the credential attributes or the underlying credential definition would invalidate the CL signature, it is computationally infeasible for an adversary to forge a valid credential or construct a deceptive Verifiable Presentation without access to the Issuer's

private key. This guarantee is therefore purely **cryptographic**, rooted in the Strong RSA Assumption, and holds independently of the Oracle and smart-contract layers.

2. **Holder Binding and Non-transferability:** To prevent credential-sharing attacks in which a malicious holder transfers their credentials to unauthorized parties, the system implements a deterministic identity-anchoring mechanism. Each VC is cryptographically bound to the holder's Decentralized Identifier, which is in turn mapped to a designated blockchain address via the DIDVerifier contract. This binding ensures that only the entity in possession of the private key corresponding to the registered DID can successfully generate a valid VP and trigger state transitions on the business chain. This property, therefore, combines a **cryptographic** component—namely, the ZKP-based linkage between the VP and the holder's DID—with an **architectural** component enforced by the DIDVerifier contract.
3. **Privacy and Selective Disclosure:** Privacy preservation relies on the selective disclosure guarantees of the AnonCreds ZKP protocol, as discussed in Sections 2.3 and 3.3.1. No plaintext trade documents are stored on-chain, and verification consumes only proof artifacts.
4. **Integrity and Anti-Replay:** Cross-domain integrity follows the nonce-binding and UUID-matching mechanisms defined in Section 3.4.2. Replay attempts are rejected at the Oracle layer before any state transition occurs.

3.6.4. Challenges and Limitations in Credential Revocation

A critical limitation of the current prototype is the absence of a complete, automated credential revocation mechanism, which remains a core challenge in heterogeneous cross-chain environments. The specific hurdles are analyzed as follows:

1. **The Problem of State Inconsistency:** As highlighted in prior research [24,25], a key challenge in heterogeneous ledger systems is state inconsistency between the identity sovereignty domain and the business execution domain. While a credential may be revoked on the identity chain, the corresponding hash (h_{vc}) anchored on the business chain remains static. Without real-time synchronization, an adversary could exploit this inconsistency to execute unauthorized transactions with a revoked credential.
2. **Architectural Hurdles for Cross-Chain Synchronization:** Efficient revocation in this asymmetric dual-chain topology introduces notable architectural trade-offs:
 - *Synchronization Latency:* Our experiments report an average cross-chain latency of 13.31 s. In high-frequency scenarios, this delay creates a vulnerability window during which a revoked credential may still be accepted by the destination contract.
 - *Verification Overhead:* Actively querying revocation status (e.g., via Revocation Status Lists or accumulators) for each cross-chain interaction increases computational overhead on the Oracle matrix and ACA-Py instances, which are already identified as system bottlenecks.
3. **Privacy and Linkability Risks:** Beyond consistency and performance, credential revocation introduces severe privacy challenges, primarily through linkability vulnerabilities. When revocation proofs are stored in public registries such as DLTs, sensitive information can be leaked. If revocation registries use public keys or unique identifiers, third parties and verifiers can correlate the revoked credential back to the holder's real identity [26,27]. Furthermore, traditional revocation checks can introduce the "Phone Home" problem, where issuers learn the association between holders and verifiers, thereby enabling unauthorized tracking of user behavior across domains [26].
4. **Future Considerations:** To mitigate these limitations, future iterations should integrate privacy-preserving revocation mechanisms. While short-lived credentials [25] can re-

duce reliance on explicit revocation, advanced cryptographic techniques are required to thwart linkability. For instance, replacing conventional digital signatures with Verifiable Random Functions (VRFs) can decouple the credential from the public key, utilizing unpredictable hash values to resist third-party correlation [27]. Alternatively, mechanisms like Prevoke, which synergize Bloom Filters with Merkle Tree Accumulators, enable holders to selectively disclose revocation status via zero-knowledge proofs without exposing precise identifiers or exposing organizational revocation rates [26]. In addition, a decentralized Oracle-driven invalidation signal that propagates revocation events from Indy to Besu is necessary for a production-ready zero-trust framework [24].

4. Experiments, Results, and Discussion

This section validates the functional integrity of the proposed architecture and evaluates the performance of its core components.

4.1. Experimental Design and Setup

4.1.1. Configuration

Experiments were conducted in an isolated network. Blockchain nodes, ACA-Py agents, and Oracle services were deployed on separate hosts with fixed configurations. Table 5 details the hardware and software parameters. The Besu chain uses IBFT consensus and a 3M EVM gas limit.

Table 5. Experimental environment configurations.

Component	Configuration Details
Hardware & OS	AMD Ryzen 7 255H (8-core), 8 GB RAM, Ubuntu 20.04/22.04 LTS
Blockchain Layer	Hyperledger Besu (IBFT 2.0, Block Period: 2 s, Gas Limit: 3 M)
Identity Layer	ACA-Py (Present Proof, Auto-respond Holder, Verifier Service)
Test Workloads	VP Verification Oracle; 4 VC Schemas (Inspection, Insurance, Origin, Bill of Lading)
Testing Tool	Custom multi-process script (5 concurrency levels, batch/VC-level metrics export)

The `config.json` file configures the Oracle matrix by mapping DIDs to specific Besu addresses. The blockchain layer uses a 3M Gas Limit and a 10^9 wei Gas Price. This setup ensures predictable execution costs during batch testing.

Experiments were conducted in a controlled laboratory environment using a small-scale node cluster under ideal network conditions. This study does not represent a real-world distributed deployment involving multiple independent organizations or heterogeneous networks.

4.1.2. Workload Models and Datasets

Evaluation targets VP batch verification and VC cross-chain transfer. To ensure reproducibility and eliminate ambiguity, the experimental workload is parameterized across these two core scenarios:

- **VP Verification:** We pre-generated a dataset with four VC types in equal proportions (1:1:1:1). To ensure accurate baseline measurement, no credential issuance occurs during the performance evaluation window. A logical batch is defined as one task containing $N_{vp} = 4$ requests (one each for inspection report, insurance contract, certificate of origin, and bill of lading).

- **Concurrency Model:** A multi-process model evaluates concurrency $C \in \{2, 5, 10, 20, 40\}$. Each process independently executes $I = 10$ iterations of batch verification.
- **Total Sample Size:** At $C = 40$, the sample size reaches $B_{total} = C \times I = 400$ batches (1600 verifications) to ensure statistical significance.
- **Proof Request Parameters:** The experimental workload uses standardized *Proof Requests* generated via the ACA-Py REST API. This approach ensures reproducibility. Each request includes a dynamic challenge nonce (c_{nonce}) valid for 300 s. It specifies required attributes, such as *inspectionPassed* and *productBatch*. Furthermore, the *restrictions* field enforces cryptographic integrity. This binds the presentation to a designated *cred_def_id* and a UUID-based business identifier (*contractName*). This value-level constraint ensures that each Verifiable Presentation is uniquely associated with a specific trade event. This prevents cross-cycle credential reuse and supports the “Entity–Data–Asset” verification loop described in Section 3.
- **Cross-Chain Transfer:** We simulate multi-terminal initiation of cross-chain transfers to evaluate the success rate and stability of Oracle write-back transactions. The test compares single-process and concurrent conditions, specifically focusing on EVM transaction nonce conflicts in concurrent transmission scenarios.
 - **Operational Chain:** A single-process baseline is established over 10 iterations to measure the intrinsic latency of the “Initiate-Relay-Confirm” loop.
 - **Oracle Processing Conditions:** To mitigate the aforementioned EVM nonce serialization conflicts under high concurrency, the Oracle matrix employs a “Multi-account Pool” with a round-robin scheduling algorithm to disperse write-back transactions.
 - **Reporting Convention.** Unless otherwise stated, throughput is calculated as $B_{total} / T_{elapsed}$ over all completed tasks, aggregated across $C \times I$ iterations. Latency values represent arithmetic means over the measured samples. Percentile values (P50/P90/P99) are explicitly labeled where applicable.

All configuration parameters, including API endpoints and hardware mapping, are synchronized with the `config.json` file in our open-source repository.

4.2. Functional Validation

Table 6 summarizes the four-phase workflow validation results, confirming consistent state transitions from identity binding to customs clearance.

Table 6. Functional Validation of the Four-Phase Workflow.

Phase	Action	Checkpoint
1	Identity setup	DID registered on Indy; DID–address binding stored in DIDVerifier
2	VC issuance & anchoring	Bill of Lading event triggers VC issuance; digest h_{vc} anchored on Chain A
3	Cross-chain relay	Relay h_{vc} + metadata to Chain B; bridge state updated
4	VP verification & settlement	VP verified off-chain; receipt submitted to Chain B; clearance completed

4.2.1. End-to-End Cross-Border Workflow Validation

To validate the functional integrity of the architecture, we simulated a complete trade cycle executing the four-phase cross-domain workflow defined in Section 3. As summarized in Table 6, the execution logs and observed checkpoints across all phases confirm that the end-to-end state migrations—from initial identity binding to final customs clearance—align with the theoretical design.

4.2.2. Access Control Validation: Decentralized Identifier and Role-Based Access Control

We simulated unauthorized and anomalous invocation scenarios to verify the dual-layer access control. Table 7 summarizes the results. Evaluation indicates that accounts without bound Decentralized Identifiers or unmatched roles are blocked at the EVM level, strictly enforcing the “Identity-as-Access” paradigm.

Table 7. Validation results of DID and RBAC access control.

Scenario	Expected Behavior	Result
Anonymous contract invocation	Blocked (No valid DID)	✓ Blocked
Non-holder cross-chain transfer	Blocked (Holder only)	✓ Blocked
Unauthorized Oracle write-back	Blocked (Invalid role)	✓ Blocked
Legitimate cross-chain transfer	Allowed (Tx confirmed)	✓ Allowed

4.2.3. Verifiable Credentials Anchoring Consistency and Verifiable Presentation Anti-Replay Validation

To validate anti-replay security, we intercepted a legitimate cross-chain **Verifiable Presentation** and replayed it in a subsequent request. The system utilizes a dynamic challenge-response mechanism where destination-chain Oracles generate unique nonces containing timestamps and random identifiers for each interaction. The replayed Verifiable Presentation failed verification because its embedded Zero-Knowledge Proof signature mismatched the current nonce. This confirms the architecture’s anti-replay robustness in open cross-domain networks.

4.3. Performance Evaluation

4.3.1. Performance Analysis of Logical VP Batching

The processing efficiency of Verifiable Presentation parsing and Zero-Knowledge Proof validation is a primary factor influencing the throughput of off-chain components. In this study, we define a **logical batch** as a set of $N_{vp} = 4$ distinct VPs—comprising an inspection report, insurance contract, certificate of origin, and bill of lading—which are required for a single trade clearance cycle. It is important to note that this batching is purely an application-level parallelism strategy: each VP is independently verified using the standard AnonCreds/CL signature protocol, and the system does not currently employ cryptographic batch optimization techniques such as multi-exponentiation or aggregate signatures. Integrating such techniques to further improve verification throughput is identified as a direction for future work. All benchmarks were conducted in a standardized, automated environment to ensure rigor and reproducibility. Latency and throughput are reported as P_{50} (median), P_{90} , and P_{99} percentiles to fully characterize tail behavior.

Throughput at concurrency C is defined as

$$\text{Throughput} = \frac{B_{\text{total}}}{T_{\text{elapsed}}}$$

where B_{total} is the total number of completed batches across all processes, and T_{elapsed} is the wall-clock time from first submission to final off-chain verification.

Mean throughput scales from 0.53 batch/s at $C = 2$ to a peak of 0.96 batches/s (3.84 VP/s) at $C = 20$, then drops to 0.77 batch/s at $C = 40$ with average latency rising to 57.5 s, marking the practical processing limit.

However, as illustrated in Figure 4, throughput begins to deviate from ideal linear scaling at higher concurrency levels ($C = 40$), where average batch latency increases and tail latencies, particularly P_{90} and P_{99} , grow more noticeably. Log analysis suggests that this degradation is driven primarily by contention within the single Aries Cloud Agent Python

instance during peak ZKP processing, including event-loop blocking and storage-lock contention, rather than by the computational complexity of the proofs themselves. To improve benchmarking consistency across heterogeneous infrastructures, we standardized the thread-pooling mechanism and request intervals.

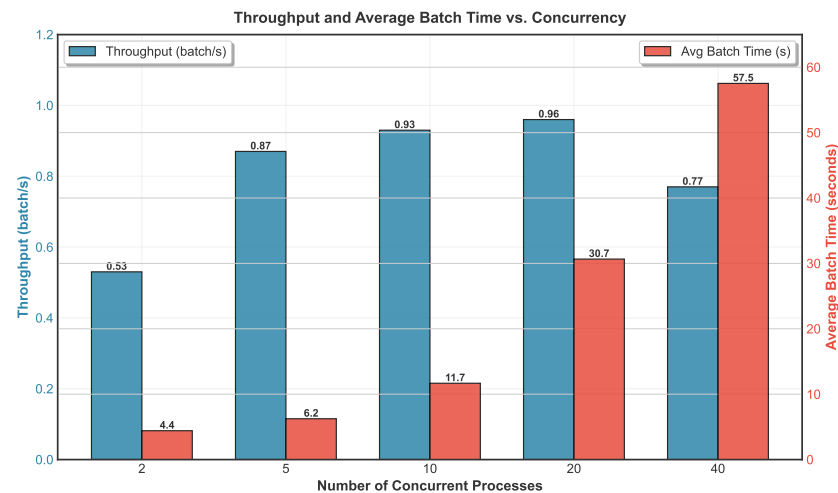


Figure 4. VP verification throughput and average batch time versus the number of concurrent processes.

Bottleneck and Scaling: The log analysis identifies the single ACA-Py instance as the dominant bottleneck under high ZKP verification load. In production settings, this limitation can be mitigated through horizontal scaling, where a load balancer distributes tasks across an ACA-Py cluster to improve throughput and reduce contention.

Latency Breakdown: Figure 5 details verification latency across Verifiable Credential types. Under a high success rate, “Bill of Lading” and “Certificate of Origin”—characterized by high attribute density—exhibit latencies of 2.89 s and 2.68 s, respectively. “Inspection Report” requires the longest verification time (3.72 s), consistent with the structural complexity defined in Table 8.

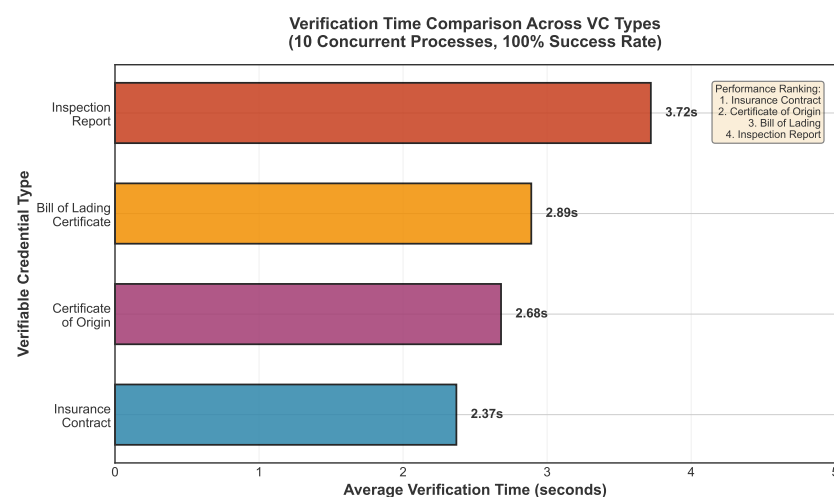


Figure 5. Comparison of average verification time across different Verifiable Credential (VC) types; (C = 10, 100% success rate).

Table 8. Core VC data models and field distribution in the cross-chain system.

Schema	Fields	Primary Use	Anchoring Format
InspectionReport	8	Quality & compliance	$h_{vc} + \text{UUID} + \text{result tag}$
InsuranceContract	18	Risk coverage	$h_{vc} + \text{UUID} + \text{policy summary}$
CertificateOfOrigin	12	Origin traceability	$h_{vc} + \text{UUID} + \text{issuer reference}$
BillOfLadingCertificate	15	Title transfer & delivery	$h_{vc} + \text{UUID} + \text{circulation state}$

4.3.2. VC Cross-Chain Transfer Performance

To rigorously evaluate the efficiency of state migration in the proposed system, we formally define the end-to-end **Latency** of a cross-chain transfer operation as:

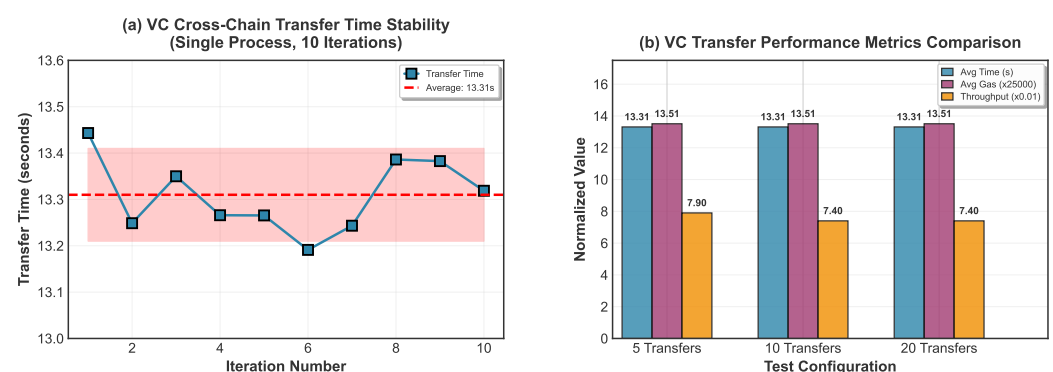
$$\text{Latency} = T_{\text{destination_confirmed}} - T_{\text{source_initiated}} \quad (1)$$

where $T_{\text{source_initiated}}$ denotes the timestamp at which the transfer transaction is submitted to the Logistics Chain (Chain A), and $T_{\text{destination_confirmed}}$ denotes the timestamp at which the corresponding state transition is permanently committed on the Regulatory Chain (Chain B).

As shown in Figure 6a, single-process iterations exhibit an average end-to-end latency of **13.31 s**. This measured interval strictly comprises the following three components:

1. **On-chain Confirmation Delay (T_{conf}):** Two sequential block confirmation windows (2 s per block) on both the source and destination chains, contributing a minimum mandatory finality wait of 4 s.
2. **Middleware Processing Overhead (T_{mid}):** The time consumed by the Oracle matrix for on-chain event monitoring, credential metadata resolution, and proof-relay generation.
3. **Network Transit Delay (T_{net}):** The asynchronous communication overhead incurred by HTTP/JSON-RPC interactions between the Oracle nodes and the Hyperledger Besu peers.

Figure 6b presents a normalized comparison of three key performance indicators—average latency, Gas consumption (expressed in equivalent Fabric resource units), and throughput—across transfer batches of varying sizes. The results indicate that, as the batch size increases, the per-transfer latency decreases significantly while the overall throughput improves, confirming the effectiveness of batch proof aggregation in amortizing the fixed overhead of cross-chain relays.

**Figure 6.** Performance and stability evaluation of VC cross-chain transfer.

Concurrent requests induced EVM nonce conflicts and transaction pool drops. A middleware “Multi-account Pool” disperses write-backs via round-robin scheduling across independent accounts, maintaining high success rates while eliminating single-account nonce serialization jitter.

4.3.3. Cost Analysis and Baseline Comparison

To rigorously evaluate the economic efficiency and architectural advantages of the proposed hash-anchoring and off-chain VP verification paradigm, we conducted a controlled comparison against a representative baseline architecture. This baseline implements a **full-payload on-chain scheme**, in which complete Verifiable Credential business attributes—such as exporter, productBatch, inspectionPassed, and other domain-specific fields—are directly persisted in the smart contract’s state structure and consumed by on-chain business logic during verification and settlement. In contrast, the proposed architecture anchors only the 32-byte credential hash (h_{vc}), holder DID, and UUID on-chain, while delegating attribute-level cryptographic verification to the off-chain Oracle matrix and ACA-Py agent stack.

Before presenting the quantitative comparison, we clarify that the full-payload on-chain scheme adopted here is intended as an upper-bound reference rather than a state-of-the-art alternative. We acknowledge that off-chain storage combined with on-chain hash anchoring (e.g., IPFS-based schemes) is widely used in practice, and our architecture aligns with this design philosophy. The full-payload baseline is selected because it allows us to measure, in an unambiguous way, the on-chain cost overhead that our architecture is able to eliminate. A broader architectural comparison with IPFS-style schemes is provided in Section 4.4.

Experimental setup for fair comparison. To ensure comparability, both the baseline and the proposed architecture were evaluated under identical conditions: the same Hyperledger Besu IBFT 2.0 network (4 validator nodes per chain), the same consensus parameters (block period: 2 s, Gas limit: 3M), the same Oracle-triggered business workflow, and the same test dataset comprising 100 VCs per credential type (Inspection Report, Insurance Contract, Certificate of Origin, Bill of Lading). The only architectural variable was the credential storage and verification strategy. This controlled setup ensures that the observed differences in Gas consumption, storage footprint, and privacy exposure can be attributed primarily to the credential handling paradigm rather than to platform heterogeneity or workload variance.

Quantitative comparison results. Table 9 summarizes the comparative evaluation across four key dimensions: on-chain storage footprint, Gas consumption, privacy exposure, and verification architecture. The baseline scheme incurs an average cost of 300,000 Gas and 1248 bytes per VC due to the linear overhead ($O(N)$) of storing multiple plaintext business fields. In contrast, the proposed architecture anchors only the essential 32-byte VC hash, DID, and timestamp on-chain, reducing the cost to 48,000 Gas and 192 bytes per VC. This represents an optimization of approximately **84% in Gas consumption** and **84.6% in storage footprint**. Furthermore, by maintaining sensitive business attributes off-chain and performing cryptographic verification via Zero-Knowledge Proofs, the proposed architecture substantially reduces on-chain privacy exposure—a critical requirement for regulatory compliance and commercial confidentiality in cross-border trade.

Table 9. Comparison between proposed architecture and full-payload baseline.

Metric	Baseline	Proposed	Improvement
Storage per VC	1248 B	192 B	84.6% reduction
Gas per VC	300,000	48,000	84.0% reduction
Privacy exposure	Full on-chain	Hash only	Strongly reduced
Verification	On-chain	Off-chain + receipt	Decoupled

Architectural trade-offs. While the proposed architecture achieves significant gains in cost efficiency and privacy preservation, it introduces additional latency due to the

asynchronous Oracle-driven coordination and off-chain VP verification workflow. As reported in the previous section, the average end-to-end cross-chain transfer latency is 13.31 s, which encompasses source-chain block confirmation, Oracle event monitoring, off-chain cryptographic verification, and destination-chain write-back confirmation. In contrast, a direct on-chain verification model (if privacy were not a concern) could potentially reduce this latency by eliminating the Oracle coordination overhead. However, such a model would sacrifice the core privacy-preserving and data-minimization properties that are indispensable for regulated cross-border trade scenarios. Therefore, the proposed architecture represents a deliberate design choice: **trading modest asynchronous latency for substantial gains in Gas efficiency, storage footprint, and privacy compliance**. For high-frequency, real-time settlement scenarios, this latency can be further mitigated through horizontal scaling of the Oracle matrix and ACA-Py agent clusters, as discussed in Section 5.

4.3.4. Comprehensive Performance Comparison

To globally evaluate the stability and execution efficiency of the system, we comprehensively compared the core metrics of VP verification (under different concurrency levels) and VC cross-chain transfer. The results are shown in Figure 7.

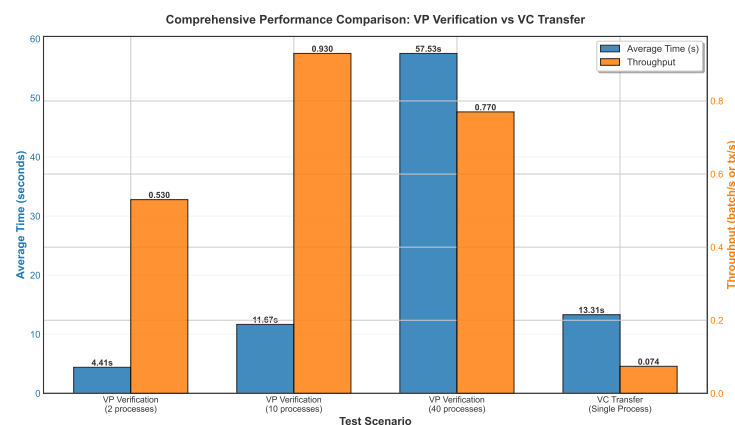


Figure 7. Comprehensive performance comparison: VP verification vs. VC transfer.

Single-process and low-concurrency scenarios ($C = 2$ to 10) maintain high success rates with throughput scaling alongside concurrency. Under high concurrency ($C = 40$, 400 samples), average Verifiable Presentation verification latency increases sharply to 57.53 s, representing an approximately 87% increase over the $C = 20$ case and indicating a significant scalability constraint in the current single-instance deployment. These results suggest that the prototype is not well-suited for high-concurrency production scenarios without further architectural restructuring.

4.4. Discussion

The architecture demonstrates feasibility and privacy-preserving capabilities; however, performance evaluation reveals system-level bottlenecks and limitations that warrant further discussion.

System Bottlenecks and Scalability: As illustrated in Figures 4 and 8, the system's throughput deviates significantly from ideal linear scaling under high concurrency ($C = 40$), with average batch latency increasing by approximately 87% compared to the $C = 20$ case. To better understand whether this degradation reflects a structural limitation of our architecture or an implementation characteristic of its components, we conducted a diagnostic analysis correlating our experimental logs with the documented internal mechanisms of the underlying libindy wallet backend used by the ACA-Py agent. This analysis suggests

that the observed bottleneck is primarily attributable to specific implementation properties of the wallet layer, rather than to the proposed Oracle matrix design itself.

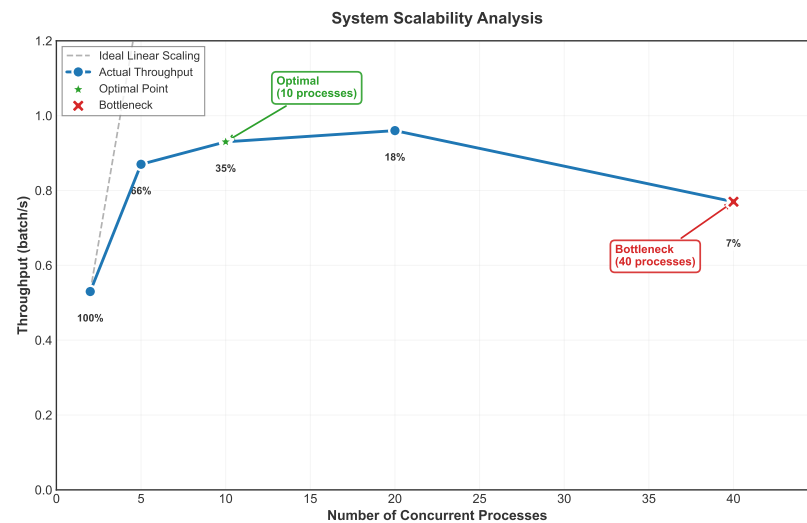


Figure 8. System scalability analysis: actual throughput versus ideal linear scaling under varying concurrent processes.

The first contributing factor is **libindy wallet lock contention**. The **libindy** library employs a global write-lock to ensure consistency of wallet state during credential operations. During VP verification, successful proof presentations trigger updates to the wallet (e.g., recording verification metadata), which require acquiring this exclusive lock. Under high concurrency, multiple verification processes contend for the same lock, leading to request serialization and a noticeable increase in wait times. This is consistent with the latency growth pattern observed in our measurements.

A second contributing factor is **libindy's fixed-size cryptographic thread pool**. This internal pool, typically configured with a small number of worker threads, processes CPU-intensive cryptographic operations including ZKP verification. When the number of concurrent requests exceeds the thread pool capacity, additional tasks are queued, which limits the system's ability to fully utilize multi-core CPU resources and caps the achievable parallelism.

Importantly, both factors are localized within the wallet and cryptographic backend of a single ACA-Py instance. The root cause is two-fold: (1) **libindy** enforces a global Rust-level mutex over wallet credential operations, serializing concurrent verification requests regardless of available CPU cores; (2) ACA-Py's **asyncio** event loop is effectively blocked by synchronous FFI calls into **libindy**, compounded by Python's Global Interpreter Lock (GIL), preventing true parallelism under elevated concurrency.

Based on this analysis, a natural and well-established mitigation strategy is **horizontal scaling**: deploying multiple stateless ACA-Py verification instances, each with an independent wallet, behind a load balancer. Such a deployment would distribute verification requests across instances, thereby reducing per-instance lock contention and increasing aggregate cryptographic throughput. We acknowledge that a full empirical evaluation of such a clustered deployment is beyond the scope of the current study and is identified as an important direction for future work. Nevertheless, the diagnostic analysis above suggests that the proposed architecture is compatible with this scaling pattern, and that the observed bottleneck reflects a known characteristic of the chosen wallet backend rather than a structural limitation of the design itself.

Architectural Limitations: Beyond performance, the prototype exhibits limitations in governance and consensus coordination. The decentralized Oracle matrix currently

operates under a *consortium trust assumption* without a formal **Byzantine Fault Tolerance** consensus or majority voting layer. This lack of a decentralized agreement protocol for conflicting Oracle outputs poses a risk of malicious collusion or state inconsistency in adversarial environments. Furthermore, the multi-component integration of Hyperledger Besu, Indy, and ACA-Py significantly increases the operational complexity of the deployment.

Potential BFT-based Oracle Coordination: To mitigate the *consortium trust limitation*, the Oracle matrix can be upgraded to a quorum-based decentralized Oracle committee of $3f + 1$ nodes, requiring at least $2f + 1$ consistent verification results before the destination bridge contract accepts a receipt. The final receipt can be extended from $\{v_{\text{bool}}, h_{\text{vc}}, \text{UUID}, t_{\text{verify}}, \sigma_{\text{Oracle}}\}$ to $\{v_{\text{bool}}, h_{\text{vc}}, \text{UUID}, t_{\text{verify}}, \text{QC}\}$, where QC denotes a *quorum certificate*. Two concrete implementations are possible: (1) lightweight **BLS threshold signature aggregation**, which compresses multi-Oracle signatures into a single constant-size proof; and (2) an off-chain **BFT** voting protocol such as **Tendermint**, providing stronger agreement under adversarial conditions. However, both options introduce additional communication overhead that may further increase T_{mid} . We leave their quantitative evaluation as future work. Notably, this architectural reliance on the Oracle introduces a trust asymmetry: while the source chain enforces strict on-chain accountability, the destination chain outsources verification to an off-chain entity, making the Oracle the system's weakest link. Although this trust concentration cannot scale to permissionless environments without BFT or TEE integration, it remains a deliberate, pragmatic choice for consortium-based supply chains where nodes are pre-authorized and legally liable.

Comparison with IPFS Architectures: Our architecture is often compared to IPFS-based systems. These systems store payloads in distributed storage and anchor hashes on-chain. Both approaches share a high-level principle: off-chain data retention combined with on-chain integrity anchoring.

Our primary contribution is not a new storage backend. Instead, we provide a unified execution framework. This framework integrates credential verification, DID-based access control, UUID-based business-cycle binding, and Oracle-mediated cross-chain state synchronization. Furthermore, the performance results reported in this paper were obtained in a controlled lab setting and may not fully reflect the latency, reliability, and governance challenges encountered in actual cross-organizational deployments.

A direct quantitative comparison with IPFS-based schemes would mainly highlight differences in storage middleware and networking overhead. It would not accurately reflect differences in identity binding, proof execution workflow, or cross-chain consistency logic. However, an empirical evaluation of these deployment trade-offs remains an important direction for future research.

The above comparison also clarifies the intended deployment scope of the proposed architecture. Although our experiments were conducted in a controlled laboratory setting, the design is applicable to regulated, document-driven multi-organization scenarios such as cross-border trade, customs clearance, and supply-chain compliance, where asynchronous finality at the scale of seconds is generally acceptable. For larger-scale deployment, the main scalability pressure lies in the off-chain ACA-Py/Oracle layer rather than in the hash-anchoring contracts; therefore, verifier agents and Oracle services can be horizontally scaled behind load balancers and operated by independent consortium members. Nevertheless, production deployment still requires further validation under WAN latency, heterogeneous governance policies, and large-scale credential revocation management.

5. Conclusions

This research implements a cross-chain Verifiable Credential management system utilizing an *Entity–Data–Asset* triple-verification mechanism. Our core contributions are

threefold. First, an asymmetric dual-domain architecture decouples identity sovereignty from on-chain execution via Decentralized Identifiers, enforcing zero-trust verification. Second, a dynamic interaction model using Verifiable Credentials and Verifiable Presentations prevents replay attacks and preserves privacy without plaintext exposure. Third, an optimized execution loop offloads Zero-Knowledge Proof computations and anchors lightweight hashes. This reduces EVM storage and Gas costs by approximately 85% compared to a full-payload baseline.

However, this asynchronous design introduces coordination overhead. System throughput degrades noticeably under high concurrency, requiring further optimization prior to high-frequency production deployment.

Limitations and Future Directions. While our prototype demonstrates feasibility and privacy, several limitations remain regarding single-instance bottlenecks and the consortium trust assumptions explicitly defined in our trust model (Section 3.6.2). To address these architectural limitations and advance the system towards production readiness, our future work will focus on three primary directions:

- **Performance Scaling and Privacy Optimization:** To mitigate off-chain verification bottlenecks under concurrent workloads, we will evaluate the horizontal scaling of an ACA-Py verification cluster and explore migrating to Hyperledger Aries Askar and AnonCreds-Rust to eliminate underlying libindy wallet lock contention. Additionally, we aim to integrate specialized Zero-Knowledge Proof circuits, hardware acceleration, and constant-size credentials (e.g., Coconut [28]) to achieve $\mathcal{O}(1)$ verification complexity and significantly improve system throughput [29].
- **Oracle Decentralization and Governance:** We plan to replace the current consortium-trusted Oracle matrix with a quorum-certified decentralized Oracle committee of $3f + 1$ nodes to mitigate Byzantine behavior. The destination bridge contract will require at least $2f + 1$ consistent signatures, supported by BLS threshold signature aggregation and off-chain BFT voting protocols like PBFT or Tendermint. Furthermore, we will explore Decentralized Autonomous Organization (DAO) governance and staking mechanisms to provide economic disincentives for malicious node collusion.
- **Ecosystem Interoperability and Adversarial Validation:** We will integrate Universal Resolvers to support diverse DID methods (e.g., did:key, did:ethr) and conduct quantitative architectural comparisons with IPFS-based hash anchoring schemes. Finally, to move beyond controlled laboratory conditions, we will execute geographically distributed adversarial benchmarks—including fault injections, Sybil attacks, and WAN latency testing—to validate the architecture’s resilience and consistency guarantees in large-scale, real-world trade scenarios.

Author Contributions: Conceptualization, Q.Z.; Methodology, Q.Z. and Y.J.; Software, Q.Z. and T.L.; Validation, Q.Z.; Investigation, Q.Z.; Resources, Q.Z.; Data curation, Q.Z.; Writing—review & editing, D.Z.; Visualization, Q.Z. and D.Z.; Supervision, D.Z.; Project administration, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: To enable full reproducibility and foster future research, the complete source code of the prototype system has been made publicly available. The repository contains: (i) the foundational smart contracts for the asymmetric dual-chain architecture; (ii) the event-driven Oracle middleware services; (iii) the decentralized identity infrastructure configurations; and (iv) the comprehensive benchmarking suite used to generate all reported workloads. All artifacts, implementation details, and supporting documentation are accessible at: <https://github.com/mmjd2019/cross-chain> (accessed on 30 June 2026).

Acknowledgments: During the preparation and revision of this manuscript, the authors used DeepSeek V3 and Gemini 3.1 Pro for the purposes of language polishing, grammar refinement, and streamlining repetitive expressions to improve overall readability. The AI tools were not used for generating research content, designing experiments, analyzing data, or producing scientific conclusions. The authors have reviewed and edited all output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACA-Py	Aries Cloud Agent Python
AML	Anti-Money Laundering
ARF	Architecture and Reference Framework
BFT	Byzantine Fault Tolerance
CA	Certificate Authority
CL	Camenisch-Lysyanskaya
DAO	Decentralized Autonomous Organization
DAA	Direct Anonymous Attestation
DID	Decentralized Identifier
ECDSA	Elliptic Curve Digital Signature Algorithm
eIDAS	electronic IDentification, Authentication, and trust Services
EUDI	European Digital Identity
EVM	Ethereum Virtual Machine
HTLC	Hashed TimeLock Contract
IBFT	Istanbul Byzantine Fault Tolerance
IBC	Inter-Blockchain Communication
JSON	JavaScript Object Notation
KYC	Know Your Customer
RBAC	Role-Based Access Control
REST	Representational State Transfer
RPC	Remote Procedure Call
SD-JWT	Selective Disclosure JSON Web Token
SSI	Self-Sovereign Identity
UUID	Universally Unique Identifier
V2G	Vehicle-to-Grid
VC	Verifiable Credential
VP	Verifiable Presentation
ZKP	Zero-Knowledge Proof

References

1. Dekker, P.; Andrikopoulos, V. Automating Bulk Commodity Trading Using Smart Contracts. In Proceedings of the 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), Oxford, UK, 3–6 August 2020; pp. 52–60.
2. Zeng, F.; Chen, A.; Xu, S.; Chan, H.K.; Li, Y. Digitalization in the Maritime Logistics Industry: A Systematic Literature Review of Enablers and Barriers. *J. Mar. Sci. Eng.* **2025**, *13*, 797.
3. Christou, T.A.; Taylor, J.L. *Blueprint Paper on Digital Trade and the UNCITRAL Model Law on Electronic Transferable Records (MLETR)*; European Bank for Reconstruction and Development (EBRD): London, UK, 2023.
4. Micheal, L. *Decentralized Identity and KYC: Strengthening Trust in Global Trade Finance Networks*; ResearchGate Publication: Berlin, Germany, 2025.
5. Flamini, A.; Sciarretta, G.; Scuro, M.; Sharif, A.; Tomasi, A.; Ranise, S. On cryptographic mechanisms for the selective disclosure of verifiable credentials. *J. Inf. Secur. Appl.* **2024**, *83*, 103789.

6. Mazzocca, C.; Acar, A.; Uluagac, S.; Montanari, R.; Bellavista, P.; Conti, M. A Survey on Decentralized Identifiers and Verifiable Credentials. *IEEE Commun. Surv. Tutorials* **2025**, *27*, 3641–3671. <https://doi.org/10.1109/COMST.2025.3543197>.
7. Camenisch, J.; Drijvers, M.; Lehmann, A. Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited. In *International Conference on Trust and Trustworthy Computing*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 1–20.
8. Hyperledger Foundation. Hyperledger Indy Node Official Documentation. Available online: <https://hyperledger-indy.readthedocs.io/> (accessed on 30 June 2026).
9. Hyperledger Foundation. Hyperledger Besu Ethereum Client. Available online: <https://besu.hyperledger.org/> (accessed on 30 June 2026).
10. Sharabati, A.A.; Jreisat, E.R. Blockchain Technology Implementation in Supply Chain Management: A Literature Review. *Sustainability* **2024**, *16*, 2823.
11. Karaduman, Ö.; Gülhas, G. Blockchain-Enabled Supply Chain Management: A Review of Security, Traceability, and Data Integrity Amid the Evolving Systemic Demand. *Appl. Sci.* **2025**, *15*, 5168.
12. Hasan, H.; Al-Rikabi, M. Performance Comparison of Blockchain Platforms for Modeling Financial Transactions: A Case Study of Ethereum and Hyperledger Fabric. *Int. J. Finance Econ. Bus.* **2025**, *4*, 50–58.
13. Najati, I. Exploring the Failure Factors of Blockchain Adopting Projects: A Case Study of TradeLens Through the Lens of Commons Theory. *Front. Blockchain* **2025**, *8*, 1503595.
14. Belchior, R.; Vasconcelos, A.; Guerreiro, S.; Correia, M. A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *arXiv* **2024**, arXiv:2005.14282.
15. Al-Breiki, H.; Rehman, M.H.U.; Salah, K.; Svetinovic, D. Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges. *IEEE Access* **2020**, *8*, 85675–85685.
16. Buldini, A.; Mazzocca, C.; Montanari, R.; Uluagac, S. Compact and Selective Disclosure for Verifiable Credentials. *arXiv* **2025**, arXiv:2506.00262v2.
17. Camenisch, J.; Lysyanskaya, A. A signature scheme with efficient protocols for anonymous credentials. In *Proceedings of the 4th International Conference on Security in Communication Networks (SCN 2002)*, Amalfi, Italy, 8–10 September 2004; Springer: Berlin/Heidelberg, Germany, 2002; pp. 123–142.
18. Brickell, E.; Camenisch, J.; Chen, L. Direct Anonymous Attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*, Washington, DC, USA, 25–29 October 2004; Association for Computing Machinery: New York, NY, USA, 2004. <https://doi.org/10.1145/1030083.1030103>.
19. Eisenstadt, M.; Ramachandran, M.; Chowdhury, N.; Third, A.; Domingue, J. COVID-19 Antibody Test/Vaccination Certification: There's an App for That. *IEEE Open J. Eng. Med. Biol.* **2020**, *1*, 148–155. <https://doi.org/10.1109/OJEMB.2020.2999214>.
20. Parameswarath, R.P.; Gope, P.; Sikdar, B. A Privacy-Preserving Authenticated Key Exchange Protocol for V2G Communications Using SSI. *IEEE Trans. Veh. Technol.* **2023**, *72*, 14771–14785. <https://doi.org/10.1109/TVT.2023.3281371>.
21. International Trade Centre. *Expediting Trade Through Electronic Bills of Lading*; International Trade Centre (ITC): Geneva, Switzerland, 2025. Available online: <https://www.intracen.org/file/20250423itcexpediting-tradewebpagespdf> (accessed on 30 June 2026).
22. Adamyk, B.; Benson, V.; Adamyk, O.; Liashenko, O. Risk Management in DeFi: Analyses of the Innovative Tools and Platforms for Tracking DeFi Transactions. *J. Risk Finance Manag.* **2025**, *18*, 38.
23. Buldini, A.; Mazzocca, C.; Montanari, R.; Uluagac, S. Benchmarking Selective Disclosure Mechanisms for Verifiable Credentials: A Systematic Comparison for Security and Privacy. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 13205–13220.
24. Mazzocca, C.; Acar, A.; Uluagac, S.; Montanari, R. EVOKE: Efficient Revocation of Verifiable Credentials in IoT Networks. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24)*, Philadelphia, PA, USA, 14–16 August 2024; Association for Computing Machinery: New York, NY, USA, 2024; pp. 1279–1295.
25. Lee, Y.; Shin, H.; Choi, D. A Survey on Credential Revocation and DID Deactivation in Self-Sovereign Identity Systems. *IEEE Access* **2026**, *14*, 16089–16115.
26. Manimaran, P.; da Conceição, A.F.; Garrett, T.; Raikwar, M.; Vitenberg, R. Prevoke: Privacy-Preserving Configurable Method for Revoking Verifiable Credentials. In *Proceedings of the 2024 IEEE International Conference on Blockchain (Blockchain)*, Copenhagen, Denmark, 19–22 August 2024; IEEE: New York, NY, USA, 2024; pp. 354–361. <https://doi.org/10.1109/Blockchain62396.2024.00053>.
27. Papathanasiou, A.M.; Polyzos, G. C. Privacy-preserving Revocation of Verifiable Credentials with Verifiable Random Functions. In *2024 International Conference on Information Networking (ICOIN)*, Ho Chi Minh City, Vietnam, 17–19 January 2024; IEEE: New York, NY, USA, 2024; pp. 391–394. <https://doi.org/10.1109/ICOIN59985.2024.10572151>.

28. Sonnino, A.; Al-Bassam, M.; Bano, S.; Meiklejohn, S.; Danezis, G. Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium 2019, San Diego, CA, USA, 24–27 February 2019*; The Internet Society: Reston, VA, USA, 2019. <https://doi.org/10.14722/ndss.2019.23272>.
29. Flamini, A.; Ranise, S.; Sciarretta, G.; Scuro, M.; Sharif, A.; Tomasi, A. A First Appraisal of Cryptographic Mechanisms for the Selective Disclosure of Verifiable Credentials. In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023), Rome, Italy, 10–12 July 2023*; SciTePress: Setúbal, Portugal, 2023; pp. 123–134. <https://doi.org/10.5220/0012084000003555>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.