algorithms

IMPACT FACTOR 2.1

CITESCORE 4.5

Article

A Bidirectional Bridge for Cross-Chain Revocation of Verifiable Credentials in Segregated Blockchains

Matei Sofronie, Andrei Brînzea, Alexandru Bratu, Iulian Aciobăniței and Florin Pop

Special Issue
Algorithms for Cyber Defense: From Cryptography to Behavioral Analysis

Edited by
Dr. Paul A. Gagniuc

MDPI

https://doi.org/10.3390/a18120734

Article

A Bidirectional Bridge for Cross-Chain Revocation of Verifiable Credentials in Segregated Blockchains

Matei Sofronie [1,†], Andrei Brînzea [1,2,†], Alexandru Bratu [3,†], Iulian Aciobăniței [1,*,†] and Florin Pop [2,†]

1 Faculty of Information Systems and Cyber Security, Military Technical Academy "Ferdinand I", 050141 Bucharest, Romania; matei.sofronie@mta.ro (M.S.)
2 Faculty of Automatic Control and Computers, National University of Science and Technology POLITEHNICA, 060042 Bucharest, Romania
3 Special Telecommunication Service, 060044 Bucharest, Romania
* Correspondence: iulian.aciobanitei@mta.ro
† These authors contributed equally to this work.

Abstract

Verifiable Credentials (VCs) are a core component of decentralized identity systems, enabling individuals to prove claims without centralized intermediaries. However, managing VC revocation across segregated blockchain networks remains a key interoperability challenge. In this paper, we present a bidirectional blockchain bridge that enables the cross-chain verification of VCs between two Ethereum-compatible private blockchain networks: Geth and Besu. The system allows credentials issued and revoked on one chain to be validated from another without duplicating infrastructure or compromising security. Our architecture combines on-chain smart contracts with an off-chain relay, ensuring auditable, low-latency credential checks across chains. Our proposal is validated through an open-source working prototype. It is particularly relevant for domains where independent organizations must validate shared credentials across segregated blockchain infrastructures, including education, healthcare, and governmental identity services.

Keywords: verifiable credentials; blockchain; blockchain bridge; attribute attestations

check for updates

Academic Editors: Paul A. Gagniuc and Arslan Munir

Received: 28 October 2025
Revised: 14 November 2025
Accepted: 19 November 2025
Published: 21 November 2025

Citation: Sofronie, M.; Brînzea, A.; Bratu, A.; Aciobăniței, I.; Pop, F. A Bidirectional Bridge for Cross-Chain Revocation of Verifiable Credentials in Segregated Blockchains. *Algorithms* **2025**, *18*, 734. https://doi.org/10.3390/a18120734

Verifiable Credentials (VCs) have emerged as a foundational element in decentralized identity systems, enabling secure and privacy-preserving attribute verification. However, a persistent challenge in real-world deployments is ensuring compatibility and interoperability between distinct blockchain infrastructures. Different organizations may adopt diverse Ethereum-compatible technologies, such as Hyperledger Besu or Go-Ethereum (Geth), each operating in logically and physically isolated environments. Although these systems follow similar protocol standards, they maintain independent smart contract deployments and do not natively support credential validation across chains.

To address this limitation, the implementation presented in this work introduces a bidirectional bridge mechanism that enables secure, automated, and auditable VC verification between separate private blockchain networks. For instance, a credential issued on a Besu network can be verified from a Geth-based environment without duplicating the original infrastructure or compromising architectural separation. This approach allows each participating organization to maintain full control over its blockchain configuration while achieving functional interoperability through off-chain synchronization and on-chain logging. This mechanism can be applied in scenarios such as verifying academic certificates across universities, checking professional licenses issued by regulatory bodies, or validating medical records between hospitals operating on distinct private networks. As a result, the system enables the development of decentralized identity ecosystems in which credential validation occurs transparently and securely across independent blockchains.

Self-Sovereign Identity (SSI) is an innovative approach to digital identity management that transfers control over personal data from centralized authorities to the individuals. Within this paradigm, users can own, manage, and selectively share their identity attributes. The core of SSI are VCs, cryptographically signed attestations issued by trusted entities that confirm the validity of specific attributes. The trust model, called trustful SSI, presented in the article [1], defines the basic principles and minimal responsibilities of each component within an SSI system. It identifies key threats, such as risks to credential integrity and external attacks, and outlines strategies to mitigate them. In a wider context, these models illustrate the varying levels of trust assumed across different SSI implementations, highlighting that trust is complex and varies depending on the context. The catalog of SSI components and design requirements serves as a practical guide for practitioners and a reference framework for researchers to support future work. Although VCs provide numerous benefits and a high level of security, they may still be exposed to certain vulnerabilities [1]. The study [2] discusses mitigation mechanisms, such as requiring each credential to be digitally signed by the issuer, which prevents adversaries from forging VCs without the issuer's private key. Every credential must include a unique issuer identifier, verifying that a trusted and recognized authority has validated the claims. The use of VCs and digital identities has been analyzed for their compatibility and potential integration in numerous workflows and use cases in various industries, including automotive, healthcare, energy distribution, supply chain management, education, and more. Their studies have shown that these technologies can be successfully incorporated into existing systems to enhance security, interoperability, and user control over personal data.

Despite ongoing standardization efforts around VCs , real-world deployments remain fragmented. Organizations often adopt distinct blockchain technologies and credential implementations, leading to network isolation and the inability to perform interoperable revocation checks. This gap becomes critical in ecosystems where credentials must be verified across institutional or infrastructural boundaries—such as education, healthcare, or governmental digital identity systems. In this work, we address this limitation by proposing a bidirectional bridge architecture that enables secure, auditable verification of the VC revocation status between two private Ethereum-compatible blockchains.

To reduce the complexity of verifier-side integration, our architecture delegates all cross-chain communication to the bridge layer. In traditional setups, verifiers would need to integrate independently with each issuer's blockchain infrastructure, increasing development overhead and fragmenting credential validation logic. In contrast, our system allows verifiers to interact solely with their local blockchain environment through a unified smart contract interface. The bridge handles the remote communication, credential lookup, and verification flow across chains. This separation of concerns simplifies deployment for relying parties and encourages adoption by minimizing integration complexity. The implementation is publicly available on Github [3].

The rest of the paper is organized as follows. Section 1 presents a comprehensive context of both VCs and blockchain bridges, mainly focused on VC revocation using blockchain technologies. It also touches the main security and interoperability requirements from working standards. The proposed architecture is presented in detail in Section 2. This section also presents the implementation details, main limitations, and performance results. Section 3 provides detailed implementation information of the proposed system, including smart contracts, the bridge, and network configuration. Section 4 presents the experimental results, including functional testing, cross-chain verification, and performance evaluation. Section 5 discusses the broader applicability of the system, highlighting its main benefits, limitations, and directions for future work. In the end, Section 6 is reserved for conclusions and future work.

1. Technological Context

This section presents the technological foundations relevant to our work, focusing on two key domains: blockchain bridging mechanisms and the secure revocation of VCs. It provides an overview of the applicable standards, existing architectures, and related academic literature that contextualize our proposed solution.

1.1. Working Standards

VCs represent a fundamental shift in digital identity management, enabling users to receive, store, and selectively disclose [4] digitally signed attestations about their attributes. These credentials, defined under the W3C Verifiable Credentials Data Model 2.0 [5], are central to the European Digital Identity Wallet (EUDI Wallet) and are reinforced by regulatory and architectural frameworks such as eIDAS 2.0 [6] and the Architecture and Reference Framework (ARF) [7].

The Verifiable Credentials Data Model 2.0 decouples the data model from specific syntaxes and introduces format profiles for interoperability across diverse ecosystems. This ensures compatibility with privacy-preserving technologies such as selective disclosure and zero-knowledge proofs (ZKPs) [8]. Within the ARF context, VCs are structured, issued, and verified using standards such as OpenID4VCI and OpenID4VP, and support lifecycle operations like revocation [9].

Revocation of VCs presents unique challenges, especially in preserving user privacy. Traditional methods such as Certificate Revocation Lists (CRLs) or Online Certificate Status Protocols (OCSPs) introduce linkability risks and allow issuers and verifiers to track credential usage [10]. ARF recommends using a validation strategy where status lists (e.g., Attestation Status Lists or Attestation Revocation Lists) are decoupled from the issuer and are locally downloaded by the relying party to avoid privacy leaks [11].

1.2. EBSI

A particularly relevant initiative is the European Blockchain Services Infrastructure (EBSI) [12], which aims to support cross-border digital public services through a network of interconnected, permissioned blockchains managed by EU member states. One of its key use cases is the issuance and verification of VCs. EBSI adheres to the W3C VCs standard and emphasizes interoperability between independent blockchain environments. However, revocation mechanisms across multiple EBSI-conformant nodes remain an ongoing challenge. Our proposed bidirectional bridge architecture aligns with the EBSI goals by providing a practical, decentralized solution for credential verification and revocation between logically and physically separate blockchain networks.

1.3. Relevant Research Literature

Revocation remains one of the most technically and ethically complex aspects of decentralized identity systems. Traditional methods such as CRLs or the OCSP rely on centralized infrastructures and introduce latency, availability, and trust issues. In contrast, blockchain-based revocation mechanisms aim to address these problems by offering immutable, auditable, and decentralized registries.

A prominent work in this area is [13], where the authors present a blockchain-based revocation system for X.509 certificates using Bloom filters on Namecoin. Their implementation supports efficient and decentralized revocation status verification, demonstrating notable performance improvements over conventional approaches.

The literature has since evolved toward more sophisticated schemes targeting VCs, which require both decentralization and privacy. Xu et al. [14] introduce a tamper-evident, privacy-preserving revocation mechanism that combines cryptographic accumulators, blockchain-based status lists, and ZKPs. This allows verifiers to confirm the revocation status of a credential without learning any extraneous metadata or compromising the subject's identity. The system is implemented in Rust and validated through end-to-end experiments, showing its viability in real-world applications.

Complementing this, Huynh et al. [15] provide a broad survey of blockchain-based VC revocation strategies. Their work categorizes current architectures into on-chain registries, accumulators, and privacy-enhancing designs, emphasizing the importance of balancing auditability with privacy. The survey also identifies open challenges, such as maintaining unlinkability during repeated credential checks, and the limitations of immutable revocation models.

These privacy concerns are especially relevant in ARF-aligned designs like Revocation List 2020 [16], where static revocation indices can lead to correlation attacks across sessions. To mitigate this, recent research proposes unlinkable revocation techniques using randomized indices and rotating cascaded Bloom filters [17–19]. These approaches reduce metadata leakage by ensuring that each revocation proof is independently verifiable but unlinkable to past or future credential usage. One of our recent contributions [19] addresses this challenge by introducing a decentralized revocation mechanism based on cascaded Bloom filters that are refreshed daily. The system avoids the use of static identifiers entirely and generates ZKPs that allow a holder to prove non-revocation without exposing a static revocation ID for the VC. This approach enhances privacy and scalability while remaining compatible with ARF's cryptographic structure. Techniques validated in large-scale deployments like CRLite [18] further support the viability of such probabilistic, high-performance revocation systems.

Parallel to this, research on blockchain interoperability—popularly referred to as blockchain bridges—has gained momentum. These systems aim to enable the transfer of data and value across heterogeneous blockchain networks. Polkadot [20] introduced a multi-chain framework where parachains communicate via a shared Relay Chain, providing native message-passing and shared security. Similarly, Cosmos [21] offers an open-standard protocol called Inter-Blockchain Communication (IBC), supporting cross-chain messaging in a hub-and-spoke topology.

Wormhole [22] extends this concept with a decentralized validator model that enables cross-chain messaging over more than 20 blockchains, including both EVM-compatible and non-EVM environments. Its architecture facilitates use cases such as token bridging, NFT transfer, and even cross-chain contract calls. However, these systems typically focus on asset transfer or general-purpose messaging and are not specialized for credential management or revocation.

Few systems have attempted to combine decentralized revocation mechanisms with a blockchain bridge tailored for identity use cases. Our proposed solution fills this gap by enabling verifiers on one blockchain to validate the revocation status of credentials issued on another, without compromising privacy or requiring infrastructure replication. Through event-driven off-chain relays and deterministic smart contract logic, the system achieves cross-chain interoperability in credential validation—advancing the decentralized identity stack with scalable, secure, and auditable revocation across networks.

Recent work has explored efficient revocation mechanisms designed for IoT environments, where resource constraints make traditional approaches impractical. One notable contribution is EVOKE [23], which introduces a lightweight revocation framework designed specifically for Decentralized Identifiers (DIDs) and VCs. The proposed system uses elliptic curve cryptography (ECC) to build a cryptographic accumulator that represents large sets of revocation data as a single constant size value.

As a result, EVOKE achieves scalability while preserving security and minimizing overhead in constrained environments. On the other hand, the proposed scheme achieves high scalability by representing large revocation sets with a fixed-size cryptographic accumulator and performing verification with minimal computational overhead. This design not only enables the efficient revocation of many credentials simultaneously but also supports offline updates. The approach presented in EVOKE provides a foundation for addressing these challenges and highlights potential directions for future research on scalable, privacy-preserving revocation mechanisms in IoT networks [23].

Recent research has advanced decentralized identity through a variety of blockchain-based mechanisms, including SSI-oriented frameworks for access control and credential management [24], relay-based cross-chain authentication models [25], and Web 3.0 credential verification schemes relying on VCs as core identity primitives [26]. Broader analyses of digital identity ecosystems further highlight the architectural, usability, and interoperability challenges that persist across DID and VC infrastructures [27]. Complementary work has also explored privacy-enhanced and revocable identity management using decentralized techniques, demonstrating the need for secure lifecycle operations and scalable revocation in SSI environments [28]. These studies collectively underline the importance of interoperable, secure, and Verifiable Credential workflows—an area in which cross-chain revocation remains insufficiently addressed.

Existing research on VCs spans standardized data models, privacy-preserving revocation schemes, and large interoperability frameworks. However, these solutions typically operate within a single blockchain ecosystem and do not offer practical support for cross-chain revocation checks. Existing bridges focus on asset transfer rather than identity workflows, while VC revocation mechanisms assume a single-chain trust model and overlook interoperability constraints. As a result, verifiers on one blockchain cannot reliably obtain revocation status from another without duplicating infrastructure or compromising security. Our work addresses this gap by introducing a bidirectional bridge tailored for VC revocation, enabling secure and auditable cross-chain verification between isolated blockchain networks.

2. Proposed Solution

2.1. System Requirements

The proposed framework is motivated by the need to enable decentralized, secure, and VC management across isolated blockchain networks. The solution must achieve the following functional requirements:

- It must allow only authorized issuers to issue and register VCs on-chain;
- Enable users to securely and independently store and manage their issued credentials;
- Enable verifiers to request validation and obtain trusted responses, even when the credential was originally issued on a different blockchain;
- Ensure the entire process is auditable, cryptographically safe, and interoperable with traditional identity models (e.g., W3C VC Data Model).

The system architecture is designed to enable cross-chain interoperability through decentralized services and trust-free third-party operations. This implies that the application must coordinate both on-chain operations through smart contracts and off-chain services, such as signature management, transaction routing, and event-based logic propagation, via a bidirectional bridge. For practical feasibility, the system needs to satisfy a list of performance requirements:

- **Responsiveness:** Credential verification and issuance operations must be visible in the user interface within seconds of confirmation in the blockchain, supported by the real-time monitoring of events.
- **Bridge latency:** The bridge between the chains must respond and process verification events within an upper limit of polling time (for example, 10 s) to ensure usability and prevent asynchronous inconsistencies.
- **Throughput:** The solution must be able to perform multiple credential operations in parallel without decreasing node responsiveness or UI response, even in multi-user system usage situations.
- **Security and isolation:** Each blockchain account used for issuing, verification, or bridging must be processed locally (e.g., by the web3 wallet or isolated key stores) to preserve private key segregation. The bridge must be able to operate without requiring access to any user secrets.

2.2. System Actors

User. A user initiates a VC issuance request to an authorized issuer by completing a credential form via the dApp and signing it using a web3 wallet. After approval and registration, the credential is stored in the user's wallet and can be exported in JSON format. Users maintain complete control of their credentials, with the ability to verify or share them without relying on centralized infrastructure.

Issuer. Issuers are trusted authorities such as universities or employers responsible for validating and signing credentials. Each blockchain includes one privileged issuer (the owner) and several authorized issuers. Authorized issuers can issue, query, and revoke credentials, but only the owner has the authority to manage access control and modify the issuer list. While authorized issuers can issue, query, and revoke credentials, only the owner issuer has the administrative rights to manage access control and update the issuer list. As a correspondence to the ARF, those two issuer types can be associated (although not fully) to the Provider and the Authentic Source roles.

Verifier. Verifiers are authorized parties that request credential validation. To ensure cross-chain functionality, a verifier is always located on a different blockchain from the one on which the VC was issued. Verifiers do not alter credential state nor interact directly with the issuer. Instead, they submit on-chain requests handled asynchronously by the bridge, which retrieves the data from the issuing chain and logs the result back into the verifier's chain.

The lifecycle of a VC and the interactions between the involved actors can be observed in Figure 1.

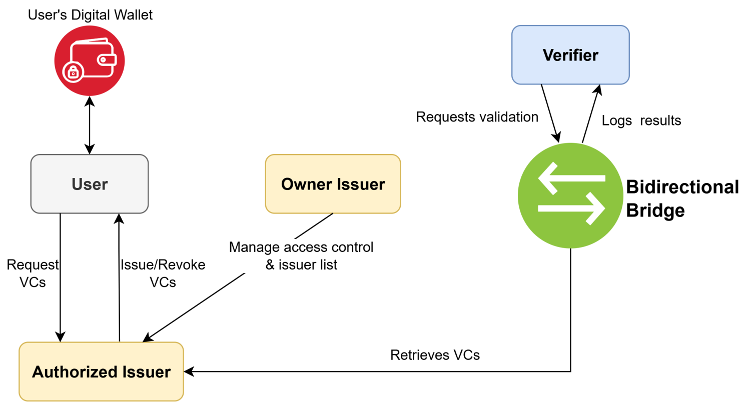

Figure 1. VCs lifecycle and actor responsibilities.

2.3. Proposed Architecture

Figure 2 illustrates the main components of an application responsible for issuing and checking VCs. It highlights user interaction, the web interface of the decentralized application (dApp), web3 wallet, smart contracts, and a bidirectional bridge that connects two private blockchain networks.

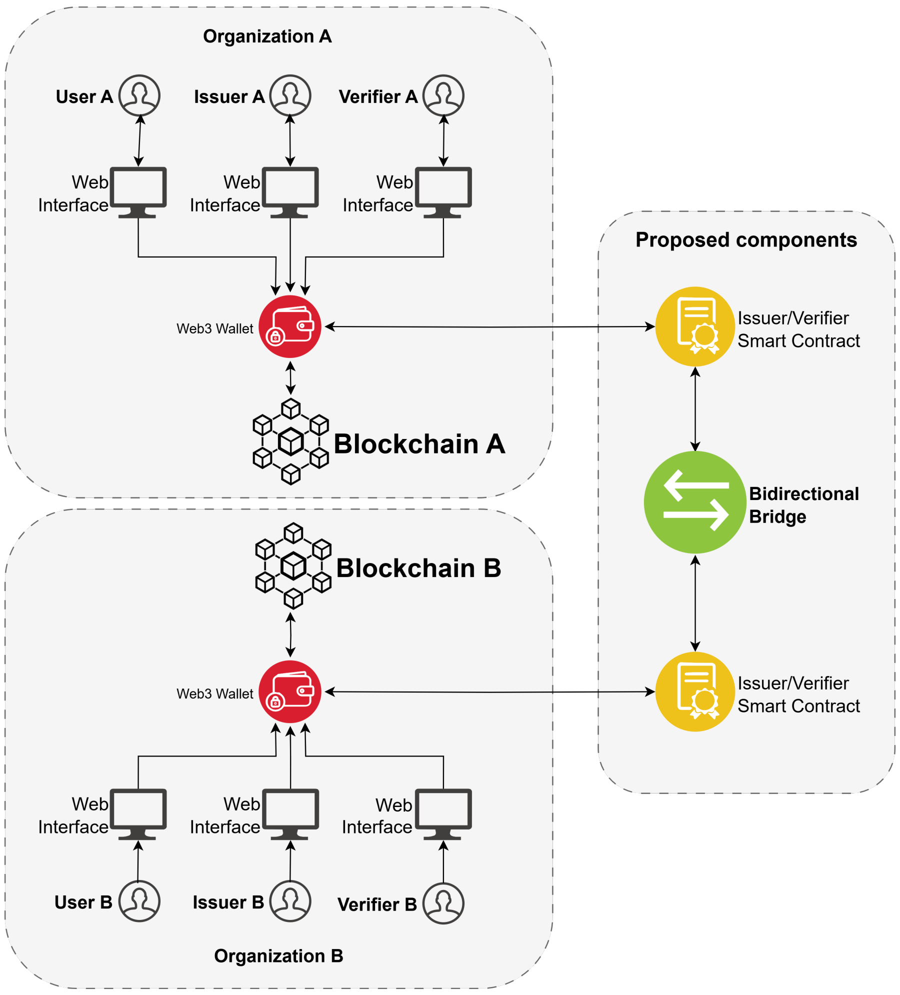

Figure 2. Overall system architecture.

Our proposal is a distributed credential management architecture that enables secure data transmission between isolated networks. The implementation of a two-way bridge ensures uninterrupted interoperability, allowing a VC created on one blockchain to be verified on another without compromising decentralization. This model showcases how cross-chain systems can be applied across diverse sectors such as education, identity verification, and professional certification.

The system comprises a set of core components that support the full credential lifecycle: issuance, registration, verification, and display of credentials across multiple blockchains.

2.3.1. Decentralized Application (dApp)

The dApp acts as the main access point for users, providing a web interface. This interface facilitates secure communication with smart contracts on both blockchain networks. The dApp does not make use of a central backend and relies on the blockchain for state management and business logic and adapts as per the user role and the network.

It employs a web3 wallet to offer off-chain and on-chain interaction, allowing users to sign data and authorize transactions without giving up control of their assets. The dApp listens to smart contract events and refreshes the interface dynamically in real time. Prior to submission, the credential data stored on-chain undergoes Keccak-256 hashing and ABI encoding to maintain consistency and integrity. Lastly, dApp is the native middleman between blockchain and the user and a mere relay and visualization layer for transactions without internal business logic.

2.3.2. Smart Contracts

Smart contracts are the on-chain structure for implementing the management of VC. From an interoperability perspective, both the Issuer Contract and the Verifier Contract smart contracts are implemented symmetrically across the two blockchains.

The Issuer Contract manages credential registration, tracks their status (valid, revoked, and expired), and handles revocation. The registration of new credentials can be performed only by trusted issuers, and there is a list of issuers stored by an owner that has been appointed specially. At the time of registering or revoking a credential, the contract triggers events for off-chain modules like dApps and bridges.

The Verifier Contract handles cross-chain verification. It records requests and triggers an event. In the scenario in which the bridge takes data from the issuing network, it posts results on-chain via a secure function so that they are available forever for auditing. Most importantly, none of the contracts on the two networks communicate directly with each other; all interactions regarding state changes occur exclusively through the bridge. This provides modularity and security and enables efficient cross-chain coordination.

2.3.3. Web3 Wallet

The web3 wallet part of the system is the secure interface of the users to blockchain networks. The web3 wallet gives users control over their blockchain identities by exposing their public addresses to the dApp and storing private keys locally in a secure manner. It plays the vital role of facilitating operations initiated through dApp by allowing users to sign off-chain messages through `personal_sign` and on-chain transactions through `eth_sendTransaction` without divulging sensitive credentials to the application itself.

The web3 wallet is required in this configuration to allow user-initiated interaction with the two networks. Every private network is configured by users manually in the web3 wallet by entering information such as the RPC endpoint and chain ID. Actor-specific accounts (users, verifiers, and issuers) are imported together with private keys and linked to their respective roles within the system.

2.3.4. Private Blockchain Networks

Our solution involves two isolated environments, each running a blockchain network. They both employ the same smart contracts to issue and authenticate credentials for functional symmetry and interoperability support.

Each network possesses its own set of nodes, user accounts, and RPC endpoints, which are configured manually. Web3 wallets are linked to these networks through their parameters (RPC URL and chain ID) so that the users may call them directly from the dApp. Both networks can play the role of issuer or verifier depending upon the transaction flow. This flexibility, as clearly seen in the architecture diagram, indicates the capacity of the system to achieve full bidirectional interoperability between the two blockchains without losing their infrastructural and logical independence.

2.3.5. Bridge

The bridge module is a critical off-chain service that facilitates secure and independent interoperability between the two private blockchain networks. The bridge module runs continuously in the background, periodically querying each network for specific on-chain events triggered by Verifier Contracts. It does not act autonomously and behaves deterministically, responding solely to blockchain inputs. Upon receiving a verification request, the bridge identifies the source network, switches context to the target blockchain, and calls the corresponding Issuer Contract functions to retrieve the status and metadata of the requested credential.

Using the collected data, the bridge constructs a transaction and submits the verification result back to the originating network via a dedicated function. It is signed with a special bridge account, and access to the contract is restricted by onlyBridge modifiers so that no third party can access the verification results. The bridge is a passive relay with access to no private key or user data and does not impose its own logic; its behavior is entirely dictated by the smart contracts that have been put on the two networks.

Cross-organization verification is a key feature of the proposed system. A VC issued to a user belonging to Organization A can be verified by a verifier associated with Organization B, even if they operate on different blockchain networks. The bridge facilitates this interoperability by asynchronously retrieving credential information from the issuing blockchain and transmitting the verification results to the verifier's blockchain, enabling smooth and secure validation across chains while preserving decentralization and data integrity.

2.3.6. Conclusion

The system architecture consists of several specialized modules that collaborate to enable the decentralized, secure, and interoperable issuance and verification of credentials. Each module performs a distinct role, collaborating seamlessly to provide comprehensive functionality within diverse blockchain networks. Modular architecture promotes scalability, transparency, and trust, thereby providing a robust platform for an effective cross-chain credential management solution.

2.4. Smart Contracts Architecture

Figure 3 illustrates the structure of the smart contract architecture and its interaction with the off-chain bridge, detailing how verification requests, event emissions, and on-chain state updates flow across the issuer and verifier networks.

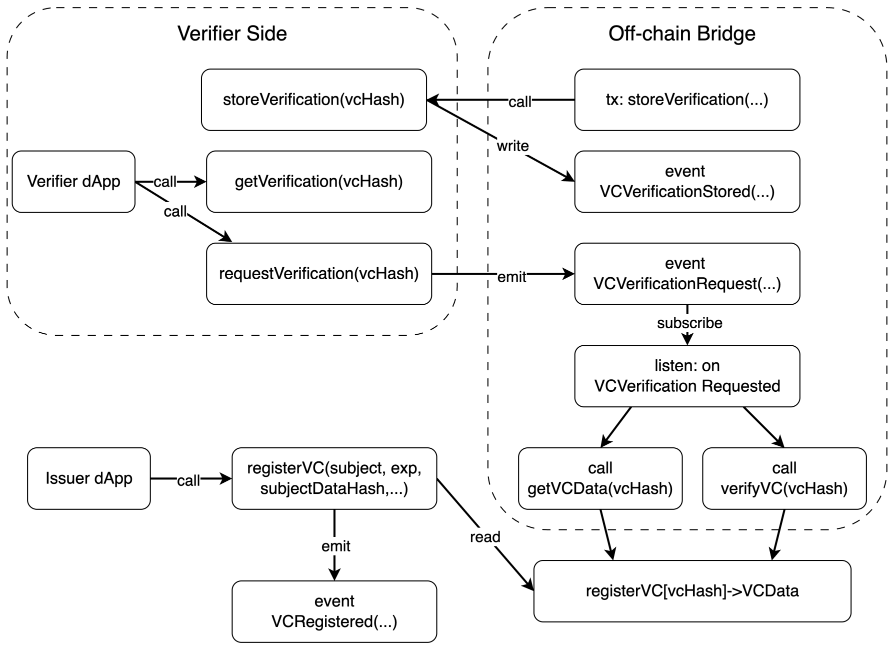
Verifier Side
Off-chain Bridge
storeVerification(vcHash)
call
tx: storeVerification(...)
write
Verifier dApp
call
getVerification(vcHash)
call
event VCVerificationStored(...)
requestVerification(vcHash)
emit
event VCVerificationRequest(...)
subscribe
listen: on VCVerification Requested
Issuer dApp
call
registerVC(subject, exp, subjectDataHash,...)
call getVCData(vcHash)
call verifyVC(vcHash)
emit
read
event VCRegistered(...)
registerVC[vcHash]->VCData

Figure 3. Smart contract architecture.

2.5. Proposed Component Interaction

The runtime system behavior and the sequence of activities performed by each component are depicted in Figure 4. It covers both off-chain and on-chain operations in a rational and consistent order. Moreover, it illustrates how bidirectional bridging facilitates requests and responses between the two chains without them directly talking to each other inter-chain.

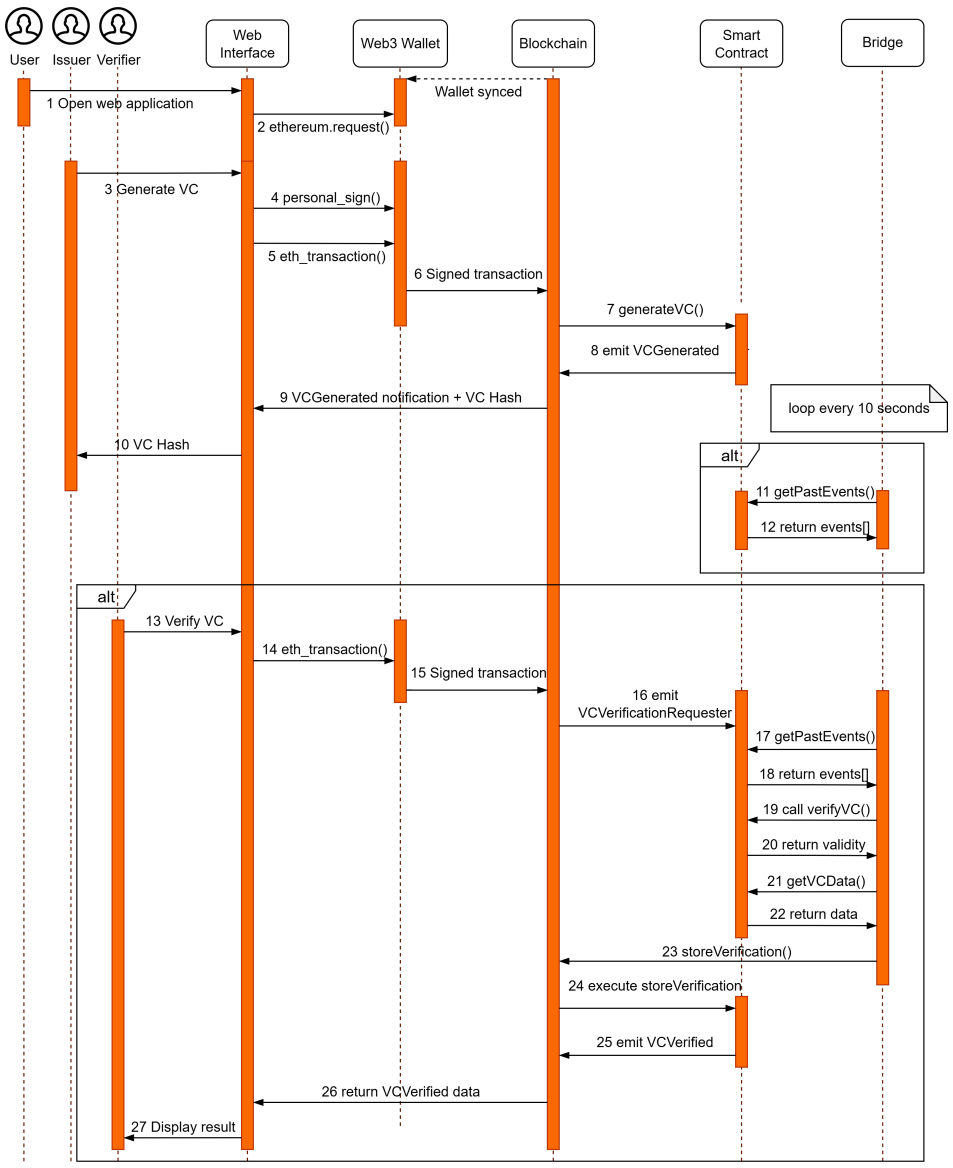

Figure 4. Sequence diagram illustrating full VC issuance and cross-chain verification.

2.5.1. Steps 1–2: Initialization and Wallet Connection

The application begins by launching the web interface, which prompts the web3 wallet to request wallet access following standard web3 method: `method:'eth_requestAccounts'`. Once the request has been granted, the web3 wallet displays the user's public address along with the network they are currently using.

Based on the detected chainId, the dApp dynamically loads network-specific logic. This is significant to establish a secure and context-based connection between users and the blockchain, ensuring that users retain full control over their identities and private keys.

2.5.2. Steps 3–10: Credential Issuance

The issuer initiates this process upon receipt of a request from a user for issuance of the credentials. After the request has been evaluated and eligibility confirmed, the issuer populates the necessary fields using the W3C VC data model. The credential is then hashed using Keccak-256 to obtain a unique identifier, known as the `vcHash`.

To secure the contents and maintain secrecy with the explicit permission of the issuer, the hash is locally signed using the `personal_sign()` function via the web3 wallet. Signing occurs off-chain and then is sent along as part of the transaction payload but not as a separate one, thus maintaining secrecy for the original message.

The decentralized application then builds a complete transaction that includes both the VC fields and the issuer signature and sends it on-chain using `eth_sendTransaction()`. The transaction signing and broadcasting are taken care of by the web3 wallet. The transaction is sent to the participating blockchain node (either one) and added to a new block. At the time of commit on the transaction, the Issuer Contract calls the `generateVC()` method, which places the credential in a mapping using the `vcHash` as a unique key. All that is required, along with the signature of the issuer, is stored, officially documenting the credential on-chain. The contract invokes a VCGenerated event on receiving it, stored by the dApp and used to populate the interface with the VC hash and its associated metadata. The credential can now be publicly verified via the contract.

2.5.3. Steps 11–12: Passive Network Monitoring by the Bridge

The bridge is an off-chain, independent service that scans both networks every 10 s. It applies the `getPastEvents()` function in an effort to find only `VCVerificationRequested` events generated recently. If it cannot find such events, it returns an empty result and waits until the next poll cycle. The bridge operates autonomously, without human intervention.

2.5.4. Steps 13–22: Handling of a Verification Request by the Bridge

The verifier, associated with either of the two networks, initiates a verification request for a VC through the dApp by providing the vcHash offered by the user. The dApp then invokes the `requestVerification(vcHash)` method on the Verifier Contract using the web3 wallet through `eth_sendTransaction()`, which requests verification on-chain. After the transaction is mined, the contract triggers a `VCVerificationRequested` event. It sends the notification to the bridge, which recognizes it in its next poll cycle and starts the cross-chain verification process.

When the bridge detects this event, it sends it to the other network where the credential was first issued. It invokes first the `verifyVC(vcHash)` function on the Issuer Contract, which is a view function, to check whether the VC is registered, revoked, and not expired. It returns a Boolean value (true if valid, false otherwise). The bridge records this along with a narrative message and justification.

Next, the bridge calls the `getVCData(vcHash)` method to retrieve all of the metadata for the credential. The returned structure contains the issuer and subject addresses, issuance and expiration timestamps, the hash of the original subject data, the schema URI, the verification method, the proof purpose, and the issuer's original signature. All this information is essential to ensure transparency and auditability during the validation process.

2.5.5. Steps 23–25: Result Confirmation

When the bridge receives the validation result and corresponding VC metadata, it returns to the source network where the request was initiated. The bridge then initiates an authenticated transaction that calls the `storeVerification()` method in the Verifier Contract. The contract verifies the authenticity of the sender (the bridge address), saves the verification result along with a timestamp, and fires a `VCVerified` event. This event marks the conclusion of the verification process and is currently on-chain for future use. This method is guaranteed to execute state transitions across chains deterministically and securely without requiring explicit interoperability between blockchains.

2.5.6. Steps 26–27: Displaying the Verification Result

When the `VCVerified` event is detected, the dApp notifies the verifier's interface of the verification outcome and its related metadata. The result (valid, revoked, expired) is displayed along with contextual information, thus signifying the successful execution of the process.

2.6. Smart Contract Interaction

Figure 5 presents the complete runtime sequence of a cross-chain verification request, showing how the verifier dApp, the verifier smart contract, the off-chain bridge service, and the issuer smart contract interact to produce an auditable verification result. The diagram illustrates the event-driven workflow triggered by a request on the verifier's blockchain, the bridge's finality-aware handling of emitted events, and the retrieval of credential status from the issuing chain through view calls. It also highlights the decision logic applied by the bridge to classify a credential as valid, revoked, expired, not found, or unreachable, followed by the on-chain storage of the verification outcome for auditability.

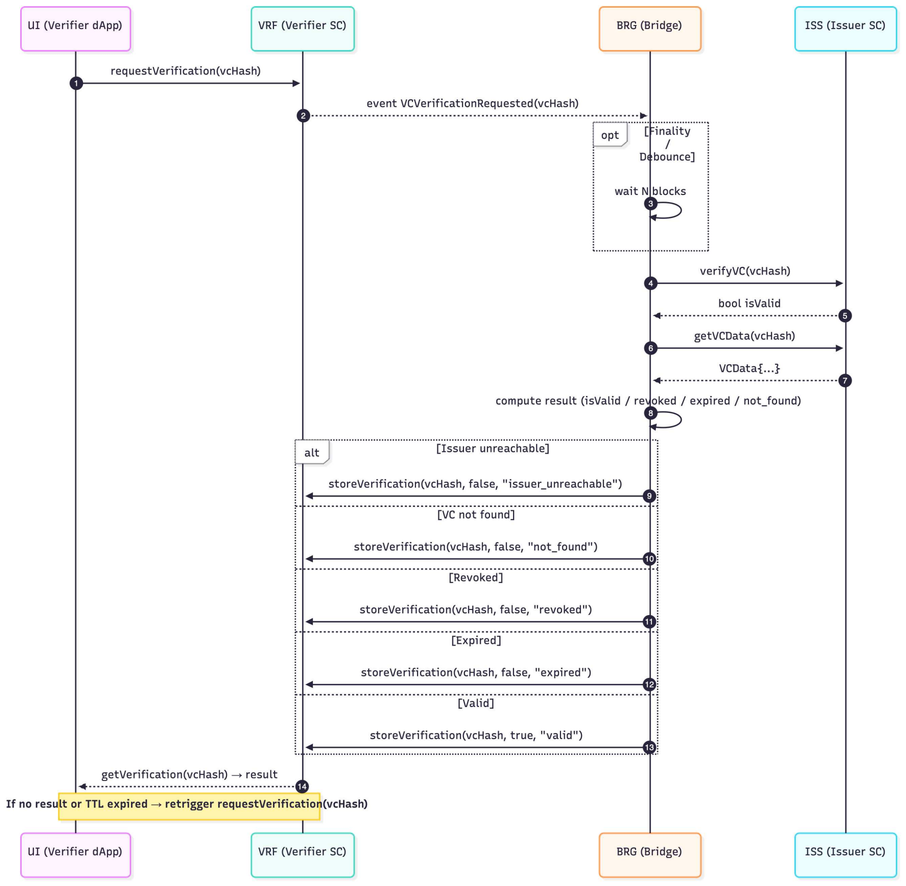
UI (Verifier dApp)
VRF (Verifier SC)
BRG (Bridge)
ISS (Issuer SC)
requestVerification(vcHash)
event VCVerificationRequested(vcHash)
opt
[Finality / Debounce]
wait N blocks
verifyVC(vcHash)
bool isValid
getVCData(vcHash)
VCData{...}
compute result (isValid / revoked / expired / not_found)
alt
[Issuer unreachable]
storeVerification(vcHash, false, "issuer_unreachable")
[VC not found]
storeVerification(vcHash, false, "not_found")
[Revoked]
storeVerification(vcHash, false, "revoked")
[Expired]
storeVerification(vcHash, false, "expired")
[Valid]
storeVerification(vcHash, true, "valid")
getVerification(vcHash) → result
If no result or TTL expired → retrigger requestVerification(vcHash)

Figure 5. Smart contracts interaction.

2.7. Threat Model and Security Analysis of the Bidirectional Bridge Architecture

As the bidirectional blockchain bridge represents the central component of our proposed architecture, we centered our threat model and evaluated the security of its design. We adopted a classical Dolev–Yao adversary model [29], where adversaries are assumed to have full control over the communication channel: they can observe, intercept, modify, inject, or replay network traffic between the bridge components and blockchain nodes. While cryptographic primitives are considered secure and cannot be broken, malicious actors may still attempt to compromise the system by impersonating the bridge, submitting fake verification requests, or trying to alter verification results.

To strengthen the security guarantees of our proposed solution, several mitigation mechanisms have been considered and implemented:

- The bridge operates off-chain as a passive, deterministic service that monitors blockchain events at regular intervals;
- Each credential is hashed using Keccak-256 (vcHash) and stored on-chain, providing immutable and publicly verifiable records;
- Critical smart contract functions can only be called by the bridge through onlyBridge modifiers, preventing unauthorized access;
- Only authorized issuers can revoke credentials on-chain, and verifiers can check the current status (valid, revoked, expired);
- Credential issuance and verification generate blockchain events (VCGenerated, VCVerified), ensuring complete transparency and auditability;
- VCs are digitally signed by the issuer using digital wallet, ensuring data integrity;
- User private keys remain in their digital wallets, ensuring that users maintain complete control over their credentials.

These implemented mechanisms collectively ensure the secure, auditable, and privacy-preserving management of VCs across independent blockchain networks. They uphold decentralization, protect user data, and enable cross-chain interoperability without relying on external intermediaries.

2.8. Bridge Processing Logic

To support reproducibility and offer a clearer understanding of how cross-chain verification is performed, we provide formal pseudocode describing the core logic executed by the bidirectional bridge. This routine represents the central mechanism responsible for detecting verification requests on the verifier chain, securely querying the issuer chain for credential status, and posting deterministic results back on-chain. The algorithm ensures that all operations remain event-driven, auditable, and consistent with chain finality constraints, while preventing duplicate processing and maintaining strict access control. The pseudocode presented in Algorithm 1 summarizes the complete processing loop executed by the bridge.

Algorithm 1: Bridge main loop

```
while true do
    sleep(POLL_INTERVAL_MS)
    evts ← B.logs(Verifier, "VCVerificationRequested", lastBlockB+1,
      B.latestBlock())
    foreach e in evts do
        id ← (e.blockNumber, e.logIndex)
        if id ∈ seen then
            continue
        end
        seen.add(id)
        hA ← A.latestBlock()
        if hA < FINALITY_N then
            continue
        end
        target ← hA - FINALITY_N
        (status, ok) ← call A.verifyVC(e.vcHash) atBlock ≥ target
        if not ok then
            result.isValid ← false
            result.issuer ← 0x0
            result.subject ← 0x0
            result.reason ← "issuer_unreachable"
        end
        else if status.issuedAt = 0 then
            result.isValid ← false
            result.issuer ← status.issuer
            result.subject ← status.subject
            result.reason ← "vc_not_found"
        end
        else
            result.isValid ← status.valid AND (NOT status.revoked) AND (NOT
              status.expired)
            result.issuer ← status.issuer
            result.subject ← status.subject
            result.reason ← status.reason
        end
        result.checkedAt ← now()
        result.validUntil ← result.checkedAt + TTL_SECONDS
        send B.storeVerification(e.vcHash, result) signed_by BRIDGE_KEY
        lastBlockB ← max(lastBlockB, e.blockNumber)
    end
end
```

3. Implementation Details

This section presents a concrete instantiation of the technology-agnostic architecture described above, used to validate feasibility and measure performance in a realistic setting.

3.1. Wallet and Client Stack

The prototype uses a web3-compatible wallet (MetaMask in our implementation) to manage private keys locally and to authorize both off-chain message signing (`personal_sign`) and on-chain transactions (`eth_sendTransaction`). The two blockchain environments used for testing are permissioned, Ethereum-compatible networks: Hyperledger Besu [30] configured with QBFT, and Go-Ethereum [31] configured with Clique. The design itself is not tied to any vendor-specific features; any EIP-1193-compatible wallet and any EVM-compatible client can be used as a drop-in replacement without requiring changes to the smart contract interfaces.

3.2. Network Isolation and Configuration

Each private network runs independently with its own RPC endpoint, chain ID, and validator set. The verifier and issuer roles are deployed on separate chains to emulate cross-organization boundaries. The verifier chain has no direct RPC access to the issuer chain; all cross-chain effects occur via the off-chain bridge. RPC endpoints are access controlled at the transport layer (IP allowlists) and application layer (per-account rate limiting).

3.3. Smart Contracts

Two contracts are deployed symmetrically on both chains: (i) an *Issuer Contract* that registers credentials, maintains status (valid, revoked, expired), and emits lifecycle events; and (ii) a *Verifier Contract* that accepts verification requests and persists bridge-posted results. Access control follows an owner/authorized-issuer model. A dedicated `onlyBridge` modifier restricts result-posting functions to a configured bridge address. Events include `VCGenerated`, `VCRevoked`, `VCVerificationRequested`, and `VCVerified` to support observability and audit.

3.4. Credential Representation

A credential identifier `vcHash` is computed off-chain by Keccak-256 over a canonical encoding of selected VC fields (subject data hash, schema URI, issuance/expiry, issuer DID/address, verification method, and proof purpose). On registration, `vcHash` and metadata are stored on-chain. This approach avoids publishing raw PII while enabling deterministic lookup and verifiable status checks.

3.5. Bridge Service

The bridge is an off-chain worker that (i) monitors `VCVerificationRequested` events on the verifier chain, (ii) queries the issuer chain's *view* functions (`verifyVC(vcHash)` and `getVCData(vcHash)`), and (iii) posts results back to the verifier chain via `storeVerification(...)`. In our prototype, the worker uses a polling interval of 10 s; it records the last processed block per chain to ensure idempotent handling and safe restarts. Transactions sent by the bridge are signed with a dedicated bridge account; no user secrets are ever accessible to the bridge. The bridge maintains an allowlist of chain IDs and contract addresses to prevent misrouting.

3.6. Reliability and Reorg Handling

To tolerate transient faults and brief chain reorganizations, the bridge employs bounded retries with exponential backoff and a configurable confirmation depth before persisting results. Event processing is idempotent by design (duplicate requests for the same `vcHash` do not alter prior results except to append a fresh timestamped record when required by policy).

3.7. Portability

Because the contracts expose standard ABIs and the bridge consumes only generic JSON-RPC/WebSocket interfaces, the implementation is portable to other web3 wallets (e.g., WalletConnect-compatible clients) and other EVM-compatible nodes (e.g., Nethermind and Erigon). Replacing components requires only endpoint and address reconfiguration; no code changes to contract logic are necessary.

3.8. Build and Deployment Notes

Contracts were compiled and deployed with standard EVM tooling; the dApp is a stateless web client that subscribes to contract events to update the UI in real time. Environment configuration (RPC URLs, chain IDs, contract addresses, and bridge key material) is externalized and versioned separately from code to facilitate reproducible experiments. The public code repository and deployment artifacts are referenced in the paper's bibliography.

4. Experimental Results

4.1. Experimental Environment

All experiments were conducted on an HP ZBook Power 16" G11 A Mobile Workstation equipped with an AMD Ryzen 7 PRO 8845HS processor (16 cores, 3.8 GHz base frequency) and 32 GB of RAM, running Windows 11 Pro 64-bit (Build 26100). To ensure isolation and reproducibility, the entire testing environment was deployed inside a VMware virtual machine configured with Debian 12 (Bookworm) as the guest operating system, where all services and blockchain clients were executed natively.

The prototype system was instantiated using two independent Ethereum-compatible private networks. The issuer chain was implemented using Go-Ethereum (Geth) v1.13.15 configured with the Clique Proof-of-Authority consensus mechanism, while the verifier chain was deployed using Hyperledger Besu v23.4.0 operating under the QBFT consensus protocol. Both chains were run locally and exposed HTTP-based RPC endpoints for interaction with external components.

The bidirectional bridge responsible for cross-chain communication was developed in Node.js v18 and connected simultaneously to both RPC interfaces. This setup enabled the deterministic monitoring of events on the verifier chain and consistent retrieval of credential status from the issuer chain.

To evaluate verification latency, a series of 100 independent cross-chain verification requests were issued from the verifier (Besu) network toward the issuer (Geth) network. For each request, the elapsed time was measured between the emission of the `VCVerificationRequested` event on Besu and the final confirmation of the corresponding `storeVerification` transaction on Geth. The values reported in Section 4.4 represent the mean latency computed across all 100 executions, excluding initialization overhead and network setup time.

4.2. Testing the Main Functionalities

To verify the correctness, integrity, and cross-chain operability of the developed system, a comprehensive set of acceptance tests was conducted in realistic settings. The primary aim of the tests was to confirm that each functional component functions as required and that the entire system facilitates the decentralized issuance and verification of credentials between different blockchain networks.

The main objectives of the acceptance tests were as follows:

- To verify that credentials can be successfully issued and revoked on both Geth and Besu;

- To ensure that the bridge can detect events, perform credential verification, and relay the result to the original network;
- To confirm that MetaMask and the dApp maintain consistent behavior across multiple roles and networks;
- To validate that credential status (valid, revoked, expired) is correctly interpreted on both blockchains, regardless of the source network.

4.3. Cross-Chain Revocation Verification (Test Case)

A typical test case involved credential issuance on Besu, revoking it, and then verifying that its status is appropriately identified on Geth:

1. A VC was issued on Besu and confirmed via the `VCRegistered` event.
2. The issuer then revoked the credential using `revokeVC(vcHash)`, and a `VCRevoked` event was emitted on-chain.
3. A verifier connected to Geth initiated a verification request on Geth.
4. The bridge detected the verification request on Geth, queried Besu via `verifyVC(vcHash)`, and received a negative result (revoked).
5. The result was stored back on Geth by calling `storeVerification`, and a `VCVerified` event was triggered.
6. The verifier's dApp successfully displayed that the VC was revoked despite being issued and revoked on a different chain.

This confirmed not only the correct propagation of state across networks but also bridge logic correctness, event tracking, and UI consistency.

This test case exemplifies the broader representative use case described in Section 5.3, where credentials issued and revoked on one chain (e.g., Besu) can be consistently validated on another (e.g., Geth), illustrating the practical value of the bridge in real-world interoperability scenarios.

4.4. Performance Testing

In addition to functional testing, the system also underwent performance testing, where latency, scalability, and resource efficiency were the prime concerns. The primary objectives of the performance test included the following:

- To measure the average time between the creation of a VC and its appearance in the UI;
- To assess the delay introduced by the bridge during cross-chain verification;
- To determine how scalable the system remains with an increasing number of concurrent operations;
- To verify that on-chain functions execute within acceptable gas limits for private PoA networks.

Tests were conducted in a local environment with private Geth and Besu networks using real RPC endpoints. The results are summarized in Table 1.

Table 1. Execution time analysis of core operations.

Operation	Avg (s)	Min (s)	Max (s)
VC generation (Geth)	5.671	5.001	6.670
VC generation (Besu)	3.002	2.604	3.995
VC revocation (Geth)	4.058	3.750	4.641
VC revocation (Besu)	2.497	2.396	2.661
Cross verification (Geth $\rightarrow$ Besu)	5.639	3.973	8.410
Cross verification (Besu $\rightarrow$ Geth)	6.151	4.220	8.957

The results presented in Table 1 confirm that most operations are processed in real time and that cross-chain verification completes reliably within the 10 s polling window used by the bridge. The slightly longer execution times observed during cross verification reflect the overhead involved in coordinating between different blockchain networks. Nevertheless, these timings demonstrate the system's efficiency and feasibility for practical deployment in scenarios requiring timely credential validation and revocations.

A detailed description of each operation listed in Table 1 is provided below:

- **VC generation (Geth)** : This refers to the credential issuance process executed on the Geth blockchain. It involves hashing the credential data (vcHash), signing it via the issuer's web3 wallet, and calling the generateVC() function in the Issuer Contract to register the VC on-chain.
- **VC generation (Besu)**: This is the same issuance process as above, executed on the Besu network. The slightly lower execution time is due to Besu's QBFT consensus and lower block confirmation delay.
- **VC revocation (Geth)**: This represents the on-chain execution of the revokeVC(vcHash) function, which marks a previously issued credential as revoked and emits a VCRevoked event. The status change is stored permanently on Geth network.
- **VC revocation (Besu)**: This is the equivalent revocation process on Besu. Authorized issuers invoke the revokeVC(vcHash) method to update the credential's revocation state in the on-chain registry.
- **Cross verification (Geth $\rightarrow$ Besu)**: A verifier on Geth requests validation for a credential originally issued on Besu. The bridge detects the VC verification event on Geth, queries the Besu chain using verifyVC(vcHash) and getVCData(vcHash), and posts the result back to Geth. The measured time includes bridge polling, remote query, and result submission.
- **Cross verification (Besu $\rightarrow$ Geth)**: This is the reverse process, initiated by a verifier on Besu for a credential issued on Geth. The bridge performs the same request–query–response sequence in the opposite direction. The slightly higher average latency observed in this direction is mainly caused by the longer block confirmation interval of the consensus algorithm used by Geth and by the bridge's polling cycle alignment during cross-chain message relay.

Table 2 illustrates the gas used in system operations. The unmarked operations in the table are `call()` type operations that do not alter storage, and thus, they are free from any gas. One can observe that the registration of credentials and storage of new verification results consume the most gas. In contrast, the revocation, issuer management, and issuing verification request operations demonstrate much greater efficiency. The usage differences between Geth and Besu are typically minimal, except for the Remove Issuer action, which is due to the presence of a for-loop over the list. Besu performs internal optimizations to read storage in the same call. In contrast, Geth adheres more closely to each EVM instruction, thereby accounting for the differences observed across the two networks.

A detailed description of each operation listed in Table 2 is provided below:

- **RegisterVC**: This executes the generateVC() function in the Issuer Contract to register a new VC on-chain. The operation stores the credential's metadata and hash (vcHash), resulting in the highest gas consumption among standard lifecycle functions.
- **RevokeVC**: This invokes the revokeVC(vcHash) function to mark a credential as revoked. This updates the on-chain status flag and emits a VCRevoked event, consuming relatively little gas since it modifies a single storage field.
- Add Issuer: This executes the administrative function that adds a new authorized issuer to the contract's internal list. This modifies access control mappings and requires authorized permissions.

- **Remove Issuer**: This removes an existing issuer from the authorization list. In Geth, this operation consumes slightly more gas due to an explicit loop over stored addresses, whereas Besu optimizes storage reads internally.
- Emit Verify to Bridge: This represents the on-chain request initiated by the Verifier Contract to trigger the bridge via the VC verification event. It prepares verification data for off-chain retrieval but does not alter storage, thus consuming minimal gas.
- **Store Verification** (already verified): This is called by the bridge when posting a repeated verification result for a credential that has already been processed. It appends a timestamp without overwriting existing data, resulting in moderate gas use.
- **Store Verification** (first time, valid): This records the outcome of a first-time verification for a valid credential. It writes a new verification record and associated metadata, explaining the relatively high gas consumption.
- **Store Verification** (first time, non-valid): This is similar to the previous case but records a negative verification result (revoked or expired). There is slightly lower gas usage due to the reduced data written to storage.

Table 2. Gas consumption for main operations.

Operation	BESU (Gas)	GETH (Gas)
RegisterVC	427,858	427,860
RevokeVC	28,992	28,992
Add Issuer	75,161	75,161
Remove Issuer	23,947	36,008
Emit Verify to Bridge	23,564	23,564
Store Verification (already verified)	54,045	54,045
Store Verification (first time, valid)	190,545	190,545
Store Verification (first time, non-valid)	150,529	150,529

5. Discussion

This section discusses the broader applicability of our proposed system, highlighting its relevance through a representative use-case, the key benefits it offers, and the limitations that can inform future work.

5.1. Fulfillment of Security Requirements

The proposed system was designed in accordance with the functional and performance requirements defined in Section 2.1. Additionally, Tables 1 and 2, and the experimental results presented in Section 4, demonstrate that these requirements were effectively achieved through the proposed solution.

5.1.1. Only Authorized Issuers Allowed

Authorized issuance of VCs: The Issuer Contract restricts credential registration to authorized addresses only, managed through an owner/authorized issuer model. This design guarantees credential authenticity and prevents unauthorized issuance.

5.1.2. Users Can Independently Store and Manage Their VCs

Secure and independent credential management by users: Users store their credentials locally in their web3 wallet, which handles private key protection and signing through standard EIP-1193 methods. No private data or secret key material is ever exposed to the dApp or bridge, ensuring user autonomy and cryptographic security.

5.1.3. Verifiers Can Validate Credentials Issued on Other Blockchain

Verifiers initiate a validation request through the Verifier Contract on their local network. The off-chain bridge listens for VCVerificationRequested events, retrieves credential data from the remote Issuer Contract, and posts results back on-chain. This mechanism enables verifiers to confirm the revocation status of a VC issued on another blockchain, assuring the interoperability objective.

5.1.4. Process Is Auditable, Safe and Interoperable

The proposed system ensures auditability, cryptographic safety, and interoperability through the use of cryptographic hashing, digital signatures, structured event logging, and standards-aligned data models. The main mechanisms are summarized below.

Cryptographic guarantees are enforced at each stage of the credential lifecycle. The key mechanisms include the following:

- Keccak-256 credential hashing: Only the credential hash (vcHash) is stored on-chain. The credential is standardized and hashed locally with Keccak-256 to ensure integrity and prevent disclosure of sensitive information.
- Wallet-based transaction signing: All issuance, revocation, and verification operations are signed locally by the user's web3 wallet. Private keys remain isolated, as the bridge and the smart contracts never access or handle user secrets.
- Immutable on-chain state: Credential status (issued, revoked, verified) is stored in contract storage, and changes require a valid signed transaction, ensuring tamper resistance and end-to-end integrity.

Auditability is achieved through transparent and immutable event logging across the entire credential lifecycle:

- Lifecycle event logging: Smart contracts emit specific events for credential generation, revocation, and verification, creating a permanent and timestamped trace of all actions.
- Traceable cross-chain verification flow: Verification requests, remote lookups, and final responses are all logged on their respective chains, enabling deterministic reconstruction of the full verification process.
- Public verifiability: Any participant with access to the chain can inspect logs, verify issuer identities, and confirm that operations followed the expected sequence.

Interoperability is supported through standards-aligned structures and blockchain-independent communication:

- W3C VC structure alignment: Credential data follow the W3C Verifiable Credentials Data Model. Hashing is performed on a stable JSON representation of the credential, ensuring compatibility with existing SSI ecosystems.
- Chain-independent representation: Only the hashed credential is stored on-chain, allowing any EVM-compatible network (Geth, Besu, etc.) to verify and interpret credential status without structural modifications.
- Standard JSON-RPC communication: The bridge operates exclusively through standard Ethereum RPC calls (eth_call, eth_getLogs, eth_sendRawTransaction), enabling cross-chain interoperability without custom protocols.

5.2. Fulfillment of Performance Requirements

The framework meets all performance requirements specified in Section 2.1. Each requirement and the corresponding implementation mechanism are described below:

- Responsiveness: As shown in Table 1, credential generation and revocation operations complete within 2–6 s, enabling near real-time user feedback through the dApp interface. Event-driven updates via web3 subscriptions ensure that UI changes occur immediately after transaction confirmation.

- Bridge latency: The bridge employs a 10 s polling interval to detect and process cross-chain verification events. The average execution time for cross-verification (5.6–6.1 s) remains within this limit, confirming the bridge's responsiveness and reliability.
- Throughput: The system supports parallel issuance and verification operations without performance decline. The bridge's event handling and transaction queuing mechanism allow multiple verifications to execute in parallel, maintaining consistent responsiveness across users and nodes.
- Security and isolation: Each blockchain network operates independently with its own validator set, RPC endpoint, and credential registry. The bridge interacts only through a dedicated account protected by the *onlyBridge* modifier, ensuring that no user private key or credential data leaves its originating environment. This architecture enforces a strong separation of trust and cryptographic isolation between the two networks.

5.3. Representative Use-Case

A representative use case for our system involves two established organizations operating independent private Ethereum-compatible networks. Consider a governmental agency responsible for issuing professional licenses on a Besu-based blockchain network. These licenses are modeled as VCs and may later be revoked due to expiration, disciplinary action, or invalid data.

On the other side, a university—operating its own private Geth-based network—needs to verify the authenticity and revocation status of a professional's credential before granting access to certain academic programs. Due to regulatory and operational boundaries, the two networks cannot be merged or centrally governed.

Our bidirectional bridge allows the verifier to request credential status across chains securely and automatically, without shared infrastructure or direct trust assumptions. This ensures that the revocation information is synchronized and verifiable across institutions, while maintaining full blockchain separation and data sovereignty.

This scenario has also been validated experimentally (see Section 4.3), where a credential issued and revoked on Besu was successfully verified as revoked from Geth through the bridge mechanism, confirming the operational significance of cross-chain interoperability.

5.4. Main Benefits

The proposed solution brings several practical advantages:

- **Simplified verifier integration:** Verifiers interact only with their local blockchain network via a single smart contract interface, while the bridge manages cross-chain communication transparently. This eliminates the need for verifiers to implement and maintain separate integrations with each issuer's infrastructure, significantly reducing adoption barriers.
- **Interoperability:** Credentials can be verified across different Ethereum-compatible networks without modifying their internal architecture.
- **Decentralization:** Each chain retains its autonomy, with no need for shared databases or central coordination.
- **Auditability:** All verification results are logged on-chain, enabling transparent, tamper-proof traceability.
- **Low impact on existing systems:** The bridge integrates with standard web3 wallets (in our implementation MetaMask) and uses widely adopted smart contract patterns.

5.5. Main Limitations

While functional, the current implementation relies on a polling mechanism to detect new verification requests. The bridge periodically queries both blockchain networks for emitted events, which introduces a small delay and consumes unnecessary resources.

In future work, we aim to replace this polling approach with a callback-based event subscription model using technologies such as WebSockets or blockchain event listeners. This change is highly likely to improve responsiveness and efficiency while maintaining the system's decentralized architecture.

5.6. Future Work

There are multiple directions in which this system can evolve. A natural extension would be to generalize the bridge to support additional Ethereum-compatible clients such as Nethermind or Erigon. This would allow broader adoption across public and private ecosystems.

Furthermore, future iterations may incorporate privacy-preserving verification techniques such as ZKPs to enhance user confidentiality during credential validation. Another possible direction is adapting the architecture to support non-EVM blockchains through standardized messaging formats, enabling credential validation between Ethereum-based networks and alternative infrastructures like Hyperledger Fabric or Cosmos-based systems.

5.7. Comparison with General-Purpose Bridges

Unlike general-purpose blockchain bridges such as Polkadot [20] or Wormhole [22], which focus on asset transfer or arbitrary message passing, our system is tailored specifically for identity and credential workflows. This specialization allows for tighter control over access rights, data structures, and logging requirements relevant to VCs.

While general-purpose bridges offer flexibility, they often introduce higher attack surfaces or require consensus-layer changes. In contrast, our implementation operates at the application layer and remains agnostic to underlying consensus mechanisms, which enhances deployability in enterprise and institutional contexts.

5.8. Comparison with Federal PKI Bridge

Our proposed solution challenge resembles, at a conceptual level, the problem faced in the early development of the U.S. Federal PKI. There, each relying party would otherwise have needed separate cross-certifications with each certification authority, leading to an 'n-to-n' trust management problem. The Federal Bridge was introduced to streamline this by acting as an intermediary trust enabler. Similarly, our blockchain bridge addresses the analogous 'n-to-n' integration problem but in the context of technical interoperability between isolated blockchains rather than trust policy alignment.

6. Conclusions

This paper introduces a practical and modular system for the cross-chain revocation verification of VCs using a bidirectional blockchain bridge. The proposed architecture supports interoperability between two Ethereum-compatible private blockchain networks—Geth and Besu—without compromising their logical or physical separation.

Our approach does not rely on shared infrastructure or central coordination. This allows for maintaining the security, privacy, and autonomy of each chain. Through a fully functional prototype, we demonstrated the ability to detect and confirm revocation status across chains with acceptable latency. The implemented solution is designed to solve a real-world problem: difficult integration between each VC verifier with multiple blockchain solutions.

Our fully functional prototype demonstrates that the architecture can meet the performance, security, and usability demands of real-world decentralized identity deployments.

The results validate the system's potential to accelerate the adoption of VCs by offering cross-chain interoperability in regulated and enterprise environments.

The proposed bridge enables the cross-chain verification of credential revocation without requiring shared infrastructure or trust assumptions. This makes it suitable for use cases such as verifying academic certificates between universities, checking professional licenses across agencies, or validating medical records exchanged between autonomous healthcare networks. It can also support governmental digital identity systems aligned with eIDAS and the EUDI Wallet. By providing a lightweight, auditable, and standards-compatible method for synchronizing credential status across isolated blockchains, the system offers a practical pathway toward interoperable SSI deployments in both public-sector and enterprise environments.

Future work will focus on extending bridge compatibility across heterogeneous DLT platforms, optimizing resource usage, and integrating privacy-preserving verification algorithms.

Author Contributions: Conceptualization, F.P. and I.A.; methodology, I.A.; software, M.S., A.B. (Alexandru Bratu) and A.B. (Andrei Brînzea); validation, F.P.; formal analysis, I.A.; investigation, M.S., and I.A.; writing—original draft preparation, I.A., M.S., A.B. (Alexandru Bratu), A.B. (Andrei Brînzea) and F.P.; writing—review and editing, I.A., and A.B. (Andrei Brînzea); visualization, M.S. and I.A.; supervision, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Executive Unit for Financing Higher Education, Research, Development and Innovation (UEFISCDI), Romania, grant number 20PTE/2025.

Informed Consent Statement: Not applicable.

Data Availability Statement: Derived data supporting the findings of this study are available from the corresponding author on request. The Github repository of the project can be accessed at the following url https://github.com/matei2803/Blockchain_Bridge, accessed on 10 November 2025.

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

Abbreviations

The following abbreviations are used in this manuscript:

ABI	Application Binary Interface
ARF	Architecture and Reference Framework
CRL	Certificate Revocation List
CRSet	Non-Interactive Verifiable Credential Revocation Set
CRLite	Certificate Revocation Lite
DID	Decentralized Identifier
dApp	Decentralized Application
DLT	Distributed Ledger Technology
EBSI	European Blockchain Services Infrastructure
ECC	Elliptic Curve Cryptography
EIDAS	Electronic IDentification, Authentication, and trust Services
EUDI	European Digital Identity
EUDIW	European Digital Identity Wallet
EVM	Ethereum Virtual Machine
Geth	Go Ethereum (client implementation)
OCSP	Online Certificate Status Protocol
PKI	Public Key Infrastructure
PoA	Proof of Authority
QBFT	Quorum Byzantine Fault Tolerance
RPC	Remote Procedure Call
SSI	Self-Sovereign Identity

VC	Verifiable Credential
VCI	Verifiable Credential Issuance (OpenID4VCI)
VP	Verifiable Presentation
ZKP	Zero-Knowledge Proof

References

1. Ernstberger, J.; Lauinger, J.; Elsheimy, F.; Zhou, L.; Steinhorst, S.; Canetti, R.; Miller, A.; Gervais, A.; Song, D. SoK: Data Sovereignty. In Proceedings of the 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), Delft, The Netherlands, 3–7 July 2023; pp. 122–143. [CrossRef]
2. Mazzocca, C.; Acar, A.; Uluagac, S.; Montanari, R.; Bellavista, P.; Conti, M. A Survey on Decentralized Identifiers and Verifiable Credentials. *IEEE Commun. Surv. Tutor.* **2025** . [CrossRef]
3. Github Repository of the Project. Available online: https://github.com/matei2803/Blockchain_Bridge (accessed on 23 June 2025).
4. Decentralized Identity, Verifiable Credentials and Self Sovereign Identity Web Directory. Available online: https://decentralized-id.com/web-standards/w3c/verifiable-credentials/data-integrity-bbs+/ (accessed on 24 May 2025).
5. World Wide Web Consortium. Verifiable Credentials Data Model v2.0. W3C Proposed Recommendation. 2025. Available online: https://www.w3.org/TR/vc-data-model-2.0/ (accessed on 10 June 2025).
6. European Digital Identity Regulation (Regulation (EU) 2024/1183). Available online: https://www.european-digital-identity-regulation.com (accessed on 10 June 2025).
7. Architecture and Reference Framework v. 1.7.1 (ARF). Available online: https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/1.7.1/architecture-and-reference-framework-main/ (accessed on 10 May 2025).
8. Preukschat, A.; Reed, D. *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*; Manning Publications: Shelter Island, NY, USA, 2021.
9. OpenID Foundation. OpenID for Verifiable Credential Issuance (OpenID4VCI) and Presentations (OpenID4VP), Draft 16. Available online: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html (accessed on 10 July 2025).
10. Kocher, P. Analysis of Privacy Risks in OCSP Protocols. In Proceedings of the Privacy Enhancing Technologies Symposium (PETS), Stockholm, Sweden, 16–20 July 2019.
11. ANNEX 2—High-Level Requirements. Available online: https://eudi.dev/1.4.0/annexes/annex-2/annex-2-high-level-requirements/ (accessed on 29 June 2025).
12. European Commission; European Blockchain Services Infrastructure (EBSI). *Digital Strategy*, European Union. Available online: https://digital-strategy.ec.europa.eu/en/policies/european-blockchain-services-infrastructure (accessed on 19 July 2025).
13. Adja, Y.C.E.; Hammi, B.; Serhrouchni, A.; Zeadally, S. A blockchain-based certificate revocation management and status verification system. *Comput. Secur.* **2021**, *104*, 102209. [CrossRef]
14. Xu, L.; Li, T.; Erkin, Z. Verifiable Credentials with Privacy-Preserving Tamper-Evident Revocation Mechanism. In Proceedings of the 2023 Fifth International Conference on Blockchain Computing and Applications (BCCA), Kuwait, Kuwait, 24–26 October 2023. [CrossRef]
15. Huynh, P.; Pham, K.; Tan-Vo, K.; Nguyen, T.; Nguyen-Hoang, T.A.; Nguyen, T.; Dinh, N.T. Beyond Immutable: The Landscape of Blockchain Credential Revocation Solutions. In *International Conference on Intelligent Systems Design and Applications*; Springer Nature: Cham, Switzerland, 2023; pp. 321–330. Available online: https://ouci.dntb.gov.ua/en/works/98aq1O27/ (accessed on 19 July 2025).
16. World Wide Web Consortium. W3C Recommendation, Bistring Status List v1.0 - Privacy-Preserving Status Information for Verifiable Credentials, 2025. Available online: https://www.w3.org/TR/vc-bitstring-status-list/ (accessed on 10 June 2025)
17. Felix, H.; Gebele, J.; Matthes, F. CRSet: Non-Interactive Verifiable Credential Revocation. *arXiv* **2025**. [CrossRef]
18. Larisch, J.; Choffnes, D.; Levin, D.; Maggs, B.M.; Mislove, A.; Wilson, C. CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 539–556. [CrossRef]
19. Bureacă, E.; Leancă, R.-A.; Ciobanu, I.; Brînzea, A.; Aciobăniței, I. Unlinkable Revocation Lists for Qualified Electronic Attestations: A Blockchain-Based Framework. *Electronics* **2025**, *14*, 2795. [CrossRef]
20. Wood, G. Polkadot: Vision for a heterogeneous multi-chain framework. *White Pap.* **2016**, *21*, 4662. Available online: https://polkadot.com/papers/Polkadot-whitepaper.pdf (accessed on 19 July 2025).
21. Kwon, J.; Buchman, E. Cosmos: A Network of Distributed Ledgers. Available online: https://cosmos.network/whitepaper (accessed on 19 July 2025).
22. Murdock, M. Wormhole: A Cross-Chain Messaging Protocol, Technical Paper. Available online: https://wormhole.com/docs/protocol/introduction/ (accessed on 19 July 2025).

23. 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; pp. 1279–1295. Available online: https://www.usenix.org/conference/usenixsecurity24/presentation/mazzocca (accessed on 19 July 2025).
24. Ou, H.-H.; Chen, G.-Y.; Lin, I.-C. A Self-Sovereign Identity Blockchain Framework for Access Control and Transparency in Financial Institutions. *Cryptography* **2025**, *9*, 9. [CrossRef]
25. Huang, Q.; Tan, M.; Tian, W. Cross-Chain Identity Authentication Method Based on Relay Chain. *Information* **2025**, *16*, 27. [CrossRef]
26. Nita, S.L.; Mihailescu, M.I. A Novel Authentication Scheme Based on Verifiable Credentials Using Digital Identity in the Context of Web 3.0. *Electronics* **2024**, *13*, 1137. [CrossRef]
27. Sedlmeir, J.; Smethurst, R.; Rieger, A.; Fridgen, G. Digital identities and verifiable credentials. *Bus Inf. Syst. Eng.* **2021**, *63*, 603–613. [CrossRef]
28. Fang, J.; Feng, T.; Guo, X.; Wang, X. Privacy-enhanced distributed revocable identity management scheme based self-sovereign identity. *J. Cloud. Comp.* **2024**, *13*, 154. [CrossRef]
29. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **2003**, *29*, 198–208. [CrossRef]
30. Besu Hyperledger Documentation. Available online: https://besu.hyperledger.org/ (accessed on 10 June 2025).
31. Geth (Go Ethereum) Documentation. Available online: https://geth.ethereum.org/docs (accessed on 10 June 2025).