INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Explore Self-Sovereign Identity Management with Hyperledger Aries

JORGE FILIPE CARVALHO FERREIRA
Junho de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Explore Self-Sovereign Identity Management with Hyperledger Aries

Jorge Filipe Carvalho Ferreira

Dissertation

Master's in Informatics Engineering

Specialization in Software Engineering

Supervisor: Isabel Azevedo

Porto, June 2025

Statement of Integrity

I hereby declare that I have conducted this academic work with integrity.

I have not plagiarized or applied any form of undue use of information or falsification of results during the process leading to its elaboration.

Therefore, the work presented in this document is original and authored by me, having not previously been used for any other end. The exceptions are explicitly recognized in the section “Ethical considerations” of the first chapter. This section also states how AI tools were used and for what purpose.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P. PORTO.

ISEP, Porto, 29 June 2025

Jorge Filipe Carvalho Ferreira

Abstract

Self-sovereign identity is a decentralized means of possessing a digital identity in which the individual is in control and owns the information. Individuals can interact in secure, private transactions where they are in control of the data, choose when to share it, using decentralized identifiers and verifiable credentials. Some problems need to be addressed to a degree, such as the problem of how to establish interoperability between different identity systems, the security ramifications of managing decentralized identifiers, and the absence of research into the revocation of verifiable credentials without compromising privacy.

A snapshot of trends, innovation, and concerns of self-sovereign identity is given in this report. The focus of this project was to keep these aspects in mind while ideating, designing, and prototyping a solution to a problem around identity. The research also touched on using one self-sovereign technology in an ongoing project. Investigating distributed identity and working with reliable verifiable credentials were part of this to make various systems communicate efficiently with each other.

Keywords: self-sovereign identity, decentralized identifiers, verifiable credentials, revocation, interoperability, security.

Resumo

Identidade auto-soberana é uma forma descentralizada de possuir uma identidade digital, na qual o indivíduo tem controlo e é proprietário da informação. Indivíduos podem interagir através de transações seguras e privadas, mantendo o controlo sobre os seus dados e escolhendo quando os partilhar, através de identificadores descentralizados e credenciais verificáveis. Ainda existem desafios por resolver, como estabelecer a interoperabilidade entre diferentes sistemas de identidade, as preocupações relacionadas com a segurança na gestão de identificadores descentralizados e abstinência de pesquisa sobre a revocação de credenciais verificáveis sem comprometer a privacidade.

Uma visão geral das tendências, inovações e preocupações da identidade auto-soberana é apresentada neste relatório. O foco deste projeto foi manter esses aspetos em mente ao idealizar, conceber e criar um protótipo de uma solução para um problema relacionado com a identidade. A integração desta tecnologia num projeto em curso também foi abordada pela pesquisa. A exploração da identidade distribuída e o trabalho com credenciais verificáveis foram parte integrante deste processo, com vista a permitir uma comunicação eficiente entre vários sistemas.

Keywords: identidade auto-soberana, identificadores descentralizados, credenciais verificáveis, revocação, interoperabilidade, segurança.

Table of Contents

List of Figures

List of Tables

List of Code Snippets

Abbreviations

AnonCreds	Anonymous Credentials
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
GDPR	General Data Protection Regulation
GQM	Goal Question Metric
JSON-LD	JSON for Linked Data
OpenID4VC	OpenID for Verifiable Credentials
JWT	JSON Web Token
SSI	Self-Sovereign Identity
UML	Unified Modelling Language
URI	Uniform Resource Identifier
VC	Verifiable Credential
VP	Verifiable Presentation
W3C	World Wide Web Consortium
WBS	Work Breakdown Structure
ZKP	Zero-Knowledge Proofs

1 Introduction

This section sets out the following elements: the context and description of the problem, the research methods and objectives, ethical considerations, and finally, an overview of the structure of this document.

1.1 Context and Problem

Self-sovereign identity (SSI) is a decentralized identity approach that empowers users with complete control over their data. It employs decentralized identifiers (DIDs) to establish unique profiles and verifiable credentials (VCs) to validate personal information [1]. Hyperledger Aries is a versatile toolkit for developing decentralized identity solutions that enable the issuance, storage, and sharing of verifiable credentials with robust privacy controls. Supporting various protocols, credential formats, and ledgers, it gives users control over what information they share regarding their identity, and with whom [2]. SSI technologies have the potential to build more secure and decentralized digital identity systems, making a remarkable contribution to strengthening the security of communications that typically involve many distributed participants [1].

However, some questions remain, as several challenges related to SSI have been identified:

- DID management presents security risks, such as a potential impersonation of the DID owner by third parties, requiring robust authorization and auditing mechanisms to prevent unauthorized changes [1].
- The lack of comprehensive research on verifiable credential revocation limits effective approaches, and existing mechanisms face challenges in balancing privacy and manageability depending on the size of the revocation list [1].
- Achieve an architecture approach for web authentication and authorization that ensures semantic interoperability across different SSI approaches [3].

- The longer average response time for credential verification compared to credential issuance poses a performance challenge that could delay critical processes, affecting user experience and security [4].

Failing to address these challenges could seriously undermine the reliability of self-sovereign identity management. Failing to address the risk of someone impersonating a DID owner could lead to security breaches and a loss of public trust, particularly in sensitive areas such as healthcare and finance. If verification transaction processing is slow, this could cause problems in situations where urgent decisions need to be made, such as in emergency services [5].

This last point is particularly important when considering the incorporation of SSI into voting systems, where trust and accessibility are established through the secure and instantaneous verification of identity. Public controversy and distrust surrounding vote counting in recent elections [6] demonstrate the need for a transparent digital identity infrastructure. SSI systems that address these challenges could form part of the solution by offering citizens a secure and verifiable way to participate in these processes.

1.2 Objectives and Approach

The main objective of this work was to explore the integration of self-sovereign identity technology into systems, focusing on key challenges such as DID management, verifiable credential revocation, and interoperability. This was supported by the development and evaluation of a prototype that involved adapting an existing application to include SSI functionalities with Hyperledger Aries.

The research questions guiding this line of work were as follows:

- RQ1: How can self-sovereign identity technology be integrated to mitigate security risks associated with decentralized identifier management in systems that demand secure and scalable identity solutions?
- RQ2: How can verifiable credential issuance and revocation be designed into systems integrating self-sovereign identity technology?
- RQ3: How can one self-sovereign identity technology be integrated into systems to enable interoperability with other self-sovereign identity approaches?

The State of the Art chapter discusses them, and several relevant studies had been analyzed. The research examined three primary aspects of self-sovereign identity integration within systems, including how to mitigate security risks associated with decentralized identifiers, the issuance and revocation of verifiable credentials, and the challenges involved in achieving interoperability with other self-sovereign identity approaches.

These issues were addressed through an experimental research methodology [7], which involved a controlled experiment focused on the development of a prototype. To this end, a base was selected in the form of an open-source voting application licensed for reuse.

Additional components were then integrated from this starting point using Hyperledger Aries SSI technology, implemented through its TypeScript framework. This incorporates the lifecycle and functionalities of self-sovereign identity management, such as issuing and revoking verifiable credentials. Finally, the solution was evaluated against pre-defined metrics using the Goal-Question-Metric (GQM) approach, with the results detailed in Chapter 7. With respect to the voting work area, this prototype is primarily applied in small-scale elections with emphasis on auditability and verification. It is not inclusive of user identification evidence, proof of voter eligibility, or integration into formal electoral registries.

The project's contribution is to demonstrate a prototype example of how SSI technology can be integrated into an existing application (voting system), overcoming significant technical challenges and offering insightful information on interoperability, security and performance trade-offs. All work, including code and documentation, is available in a public repository [8] to enable other researchers and developers to build upon these results.

1.3 Ethical Considerations

As a student of the Instituto Politécnico do Porto, I subscribe to the principles and standards set out in the institution's Code of Ethics and Conduct [9]. Article 6 deals with the specific obligations of the student community, with point 2.8 highlighting the prohibition of using ideas, phrases, paragraphs, or complete texts from others without proper citation. Point 2.9 emphasizes the importance of originality in submitted work. Point 2.10 refers to the prohibition of unauthorized collaboration. Point 2.11 specifies the importance of not presenting work that has been falsified or misleadingly interpreted. Point 2.12 prohibits the unauthorized alteration of another person's work. Finally, point 2.13 concerns the buying or selling of academic work, in whole or in part, for personal gain during the assessment process.

I am aware of the Statement of Commitment in Article 8, which upholds the integrity of academic work through its accuracy, originality, and authority. I am also aware that the Statement of Commitment implies compliance with the Code of Ethical Conduct. Article 10 calls for the research community to uphold the highest standards of scientific integrity in its conduct.

As a student of Software Engineering at the Instituto Politécnico do Porto, I also commit myself to the IEEE Code of Conduct [10] in this work of academic excellence. This statement is an expression of my individual beliefs as a student, emphasizing the importance of respecting others, being fair, avoiding vindictiveness, complying with applicable laws, and avoiding unethical behavior such as bribery, conflict of interest, and misuse of intellectual property.

For project selection, I chose a project on GitHub with an openly declared open-source license, such as the Apache-2.0 license, which allows legal use, modification, and redistribution. I also ensure that whatever was created is original, genuine, and follows the above ethics.

This work included planning and competency management, specified in Chapter 2, by the principles mentioned above. The required technical expertise was applied based on prior

learning experience and training in applicable technologies. Planning for the project was aimed at delivering high-quality results within a realistic timeframe, while at the same time recording the decisions made.

Artificial intelligence tools were used to assist with paraphrasing, refining the technical writing, and providing code improvements. This use was strictly supportive and subject to the author's critical review and did not replace the author's intellectual work or critical analysis.

1.4 Document Structure

This paper consists of nine main chapters, with each chapter serving a particular function to aid the general purpose of the document:

- Introduction: The Introduction chapter introduces the fundamental problem, emphasizes its importance, and sets the stage for the subsequent chapters.
- Planning: The second chapter contains planning and skill management activities, including timelines, critical actions, and needs. It also has milestones and potential risks to having a clear and well-organized project.
- Background: Chapter three provides the fundamental information needed to understand the topic, including definitions, explanations of central concepts, and background history, to give a solid foundation to work with the subject.
- State of the Art: The fourth chapter reviews current advances in research and technology relevant to the topic. It analyses recent developments, emerging trends, and existing gaps, laying the groundwork for further analysis and research.
- Project Selection and Analysis: The fifth chapter focuses on the analysis and design of the developed implementation. It explains the process of integrating self-sovereign identity into an existing system.
- Solution Construction: The sixth chapter describes the implementation process of the solution with the inclusion of self-sovereign identity management into the chosen project.
- Results: The seventh chapter clarifies the process of evaluation performed, focusing on challenges identified in this work.
- Conclusion: Chapter eight summarizes the main findings and reviews/reflects on the project’s outcomes.
- Appendixes: It contains supplementary materials to support the work's main content.

2 Skills Management and Planning

This section outlines the following key elements required for the project: skills management, project planning activities (including a description of the mandatory tasks needed to complete the work described in this document), and a comprehensive risk identification plan.

2.1 Skills Management

A skills questionnaire assessed their impact on the dissertation work. This process involved evaluating key competencies based on their relevance to the project, with self-assessment and expert assessment from someone familiar with the questionnaire proponent's knowledge. After completing the assessment, the most developed skills identified were time management, decision-making, and leadership. The first two skills are crucial for producing a high-quality dissertation, and leadership, although well-developed, is less relevant since the project was carried out independently, with guidance from the supervisor.

Three skills identified as areas for improvement: technical skills in the specific area of knowledge, planning and organization, and oral communication. The approach was to list them along with information that could be taken to improve each one.

Table 1 presents an evaluation of required technical skills for this project, showing current proficiency levels, identifying existing gaps, and listing the following columns:

- Skill: The technical or soft skill relevant to the development and evaluation of the prototype.
- Required Proficiency: The estimated proficiency level needed on a scale from 1 (low) to 5 (expert) to complete the work effectively.
- Current Proficiency: The current level of knowledge or experience with the skill, also on a 1–5 scale.
- Gap: Indicates the size of the gap between current and required proficiency (Low, Medium, or High).

- Comments: A brief explanation of the gap and the actions or observations related to each skill.

Skill	Required Proficiency	Current Proficiency	Gap	Comments
Self-Sovereign Identity	5	1	High	No prior experience; Requires study of core SSI concepts and standards.
Hyperledger Aries	4	3	Medium	Some familiarity exists due to prior courses performed before project selection.
Typescript	4	3	Medium	Comfortable with basics; Needs refinement in advanced use within frameworks.
UML	4	3	Medium	Academic experience: Needs improvement for clear architectural documentation.
GitHub	3	3	-	No gap; sufficient proficiency for version control and collaboration.
Testing & Evaluation	4	3	Medium	Basic testing skills are in place; Improvement needed for structured experiments.
Project Management	4	3	Low	Experience managing tasks; Slight improvement needed in planning/monitoring
Technical Writing	4	2	High	Able to write clearly; Needs refinement for structured technical documents.

Table 1 - Technical skills evaluation for the project.

For better planning and organization, the strategy was to record and manage tasks systematically to anticipate deadlines and implement regular project reviews to track progress and make necessary adjustments. Regarding oral communication, the focus was on dedicating time to preparing and refining the final dissertation presentation and organizing, gather feedback about it and refine communication skills.

While some of the domains were included through project planning activities, others had to be added as planned tasks. For example, the LSF172x - Introduction to Hyperledger Self-Sovereign Identity Blockchain Solutions course, which was studied at the beginning of the project, was a plus for improving technical skills.

2.2 Project Planning

The project planning is an experimental research methodology with six general phases: Planning, Preparation, Design, Implementation, Evaluation, and Conclusion. These phases are further divided into tasks to allow a systematic and sequential development in the project. The phases are described in a Work Breakdown Structure (WBS) with concrete deliverables at the end of each, to address the project aspects, as shown in Figure 1.

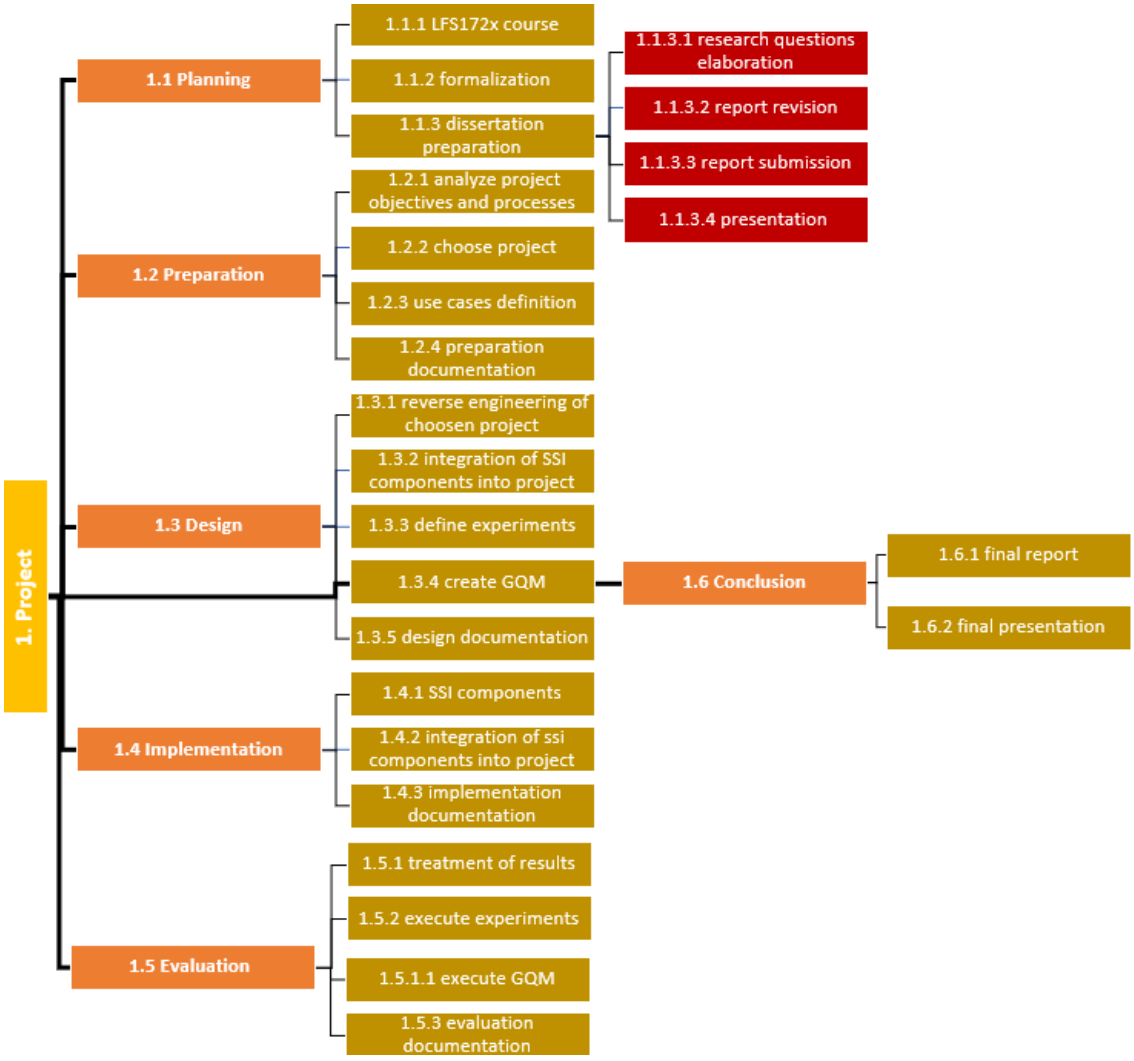

Figure 1 - Project WBS

The distribution of phases and tasks between the phases was designed to facilitate a natural progression from one phase to the next, ensuring that each phase builds on the previous one and that deliverables are met on time, allowing for timely evaluation and adjustment if necessary. Figure 2 illustrates the high-level timeline of the project phases. The blank spaces between each phase represent the excluded weekends from the schedule to reflect non-working time. Each phase also incorporates the tasks and deliverables outlined in the WBS to achieve all project goals and milestones in a structured and timely manner. The goal was to

provide a clear overview of the project's progress and highlight key deadlines for each phase's deliverables.

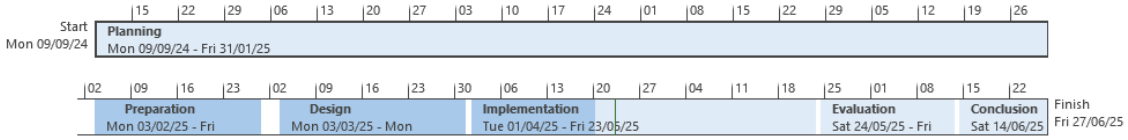

Figure 2 - High-level timeline of the project

Table 2 shows the division of the planning phase, which included an initial course designed to enhance skills in a specific knowledge area of this project. This marked the initial tasks of the project and included activities related to the formalization and preparation of the dissertation, as well as revisions and submissions coordinated with the project advisor.

WBS	Task Name	WBS Level	Duration	Start	Finish
1.1	**Planning**	**Phase**	**105 days**	**09/09/24**	**31/01/25**
1.1.1	LFS172x course	Task	28 days	09/09/24	16/10/24
1.1.2	Formalization	Deliverable	28 days	09/09/24	16/10/24
1.1.3	dissertation preparation	Deliverable	86 days	16/09/24	13/01/25
1.1.3.1	research questions elaboration	Task	19 days	16/10/24	11/11/24
1.1.3.2	report revision	Task	42 days	16/10/24	12/12/24
1.1.3.3	report submission	Task	58 days	16/10/24	04/01/25
1.1.3.4	Presentation	Task	23 days	12/12/24	13/01/25

Table 2 - Detailed Planning WBS Phase

For the next phases, each is divided into specific timeframes that align with the project's goals and strategic objectives. In particular, the implementation phase is broken down into critical tasks that focus on tangible results and maintaining alignment with the defined project timeline. These structured durations, shown in Table 3, ensure timely delivery, efficient use of resources, and flexibility to meet evolving project needs.

WBS	Task Name	WBS Level	Duration	Start	Finish
1.2	Preparation	Phase	20 days	03/02/25	28/02/25
1.3	Design	Phase	21 days	03/03/25	31/03/25
1.4	Implementation	Phase	39 days	01/04/25	23/05/25
1.5	Evaluation	Phase	16 days	24/05/25	13/06/25
1.6	Conclusion	Phase	11 days	14/06/25	27/06/25

Table 3 - Detailed WBS Phases duration

While cost estimation does not pertain to this project, monitoring and control procedures have been implemented to ensure alignment with project objectives. At the end of each phase, the information gathered was documented, enabling a systematic track of progress and ensuring tasks and milestones are completed as planned. This information was prepared for potential review by the supervisor, facilitating feedback and validation to confirm that deliverables meet the required standards before advancing to the next phase, with the content being incorporated into the full dissertation report.

2.2.1 Risk Management

During the project planning process, several risks were identified and assessed to ensure effective risk management. It is part of the control procedures implemented for this project. These risks were structured and prioritized using the Probability and Impact (PI) score metrics to assess their potential severity and likelihood (Table 4).

Description	Cause	Effect	Risk Owner	Probability (1-5)	Impact (1-5)	PI Score	Expected Result, No Action	Risk Response Type	Response description
Failure to meet the deadline for the intermediate stages	Lack of rigor in following the chronogram	Cumulative impact on final deliveries	Jorge Ferreira	3	5	15	The project will be delivered with delays	Avoid	Set intermediate deadlines and meet them strictly.
Limitation of specific knowledge	Lack of experience with certain parts of the project	Slow development or difficulties in achieving objectives	Jorge Ferreira	2	3	6	Need to research or learn, which can delay the project	Avoid	Continuous and early learning, and seeking tutorials or online support for critical topics.
Unexpected technical problems	Chosen tools or frameworks have errors or failures	Interruptions and delays during implementation	Jorge Ferreira	2	2	4	Rework or delays in solving problems	Mitigate	Carry out initial tests on components before integrating and preparing alternative plans.
Lack of iteration between stages	Ignoring improvements during intermediate stages	Accumulated problems to solve in the final stages	Jorge Ferreira	2	3	6	Final quality may be jeopardized	Mitigate	Review each stage before moving on to the next.
Presentation or final report problems	Difficulties in organizing and communicating project results	Poorly structured reports or presentations that impact evaluation	Jorge Ferreira	3	4	12	The perception of the project can be damaged by a poor presentation	Avoid	Allow enough time to revise and practice the presentation and report.

Table 4 - Identified risks for the project

It attempted to find and mitigate risks beforehand in the development of projects and ensure absolutely that the possible effects of each risk could be predetermined, with appropriate measures available to limit disruption, keep momentum going, and ultimately improve what the project can deliver.

3 Background

This chapter discusses key issues in the context of self-sovereign identity and provides a technical introduction to prepare for understanding the topic. It defines decentralized identifiers and verifiable credentials, their lifecycle including issuance, verification, revocation, and the underlying cryptographic building blocks that enable them. Furthermore, it brushes on the technologies, challenges and issues relating to the utilization of self-sovereign identity, mentioning its benefits over the potential hurdles that may accompany it.

3.1 Self-Sovereign Identity

Self-Sovereign Identity (SSI) is a distributed identity management solution that allows people to be entirely in control of their data. It is different from traditional identity systems, which, however they may be subdivided, are still centrally controlled systems that allow users to manage, store, and share their identity. This advantage is achieved by using decentralized components such as decentralized identifiers (DIDs) and verifiable credentials (VCs), whose usage is increasing to facilitate privacy, security, and trust for digital transactions [1].

DIDs provide users with the authority to create and own their own digital identities independent of centralized systems. These identifiers are globally unique and cryptographically secure. In identity and credential management, DIDs link VCs to an applicant's identity, ensuring identity is authentic and under the individual's control [11].

The structure of DIDs (Figure 2) has been standardized by the World Wide Web Consortium (W3C) and consists of three distinct parts. First, the DID URI scheme identifier, always preceded by "did:", indicates that the identifier is a DID. Second, the string contains the DID method identifier, which specifies the decentralized network or system used to create and resolve the DID. This method identifier is most often an indicator of some specific blockchain or decentralized directory, i.e., a "key". Finally, the DID contains the method-specific identifier, a unique string referencing the specific entity within the chosen DID method. This identifier is perhaps a cryptographic hash or public key, depending on the method used [12].

Figure 2 - A simple example of a decentralized identifier (DID) structure

The example DID resolves to a DID document, which contains information linked to the DID, including methods for cryptographically authenticating the DID controller (Code Snippet 1).

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

Code 1 - A simple DID document example

VCs, in contrast, enhance DIDs by providing a secure and tamper-resistant method for creating and verifying claims about an individual. VCs allow an organization to assert a claim about a subject, such as an individual, and to substantiate that claim, probably without depending on centralized databases. The issuer cryptographically signs them, so they cannot be forged without detection. VCs also enable the credential holder to be in control of their data, revealing only data that is relevant to the context of a particular situation, such as job confirmation or academic qualifications. VCs also enable the credential holder to be in control of their data, revealing only data that is relevant to the context of a particular situation, such as job confirmation or academic qualifications.

For example (Code Snippet 2), a person applying for a job can show a verifiable document from a school to support his or her academic credentials or from an accrediting agency to verify his or her competency, and the employer can independently verify such presentations. This cryptographic and decentralized method streamlines the process, saves time, and enhances trust in hiring by providing employers with confidence in the accuracy of candidates' claims.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/1872",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuer": "https://example.edu/issuers/565049",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": [{
        "value": "Example University",
        "lang": "en"
      }, {
        "value": "Exemple d'Université",
        "lang": "fr"
      }]
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://example.edu/issuers/565049#key-1",
    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5X
      sITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-pQy7UJiN5mgRxD-WUc
      X16dUEMGlv50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-kronKb78cPN25DGlcTwLtj
      PAYuNzVBAh4vGHSrQyHUdBBPM"
  }
}
```

Code 2 - Verifiable credential schema example

3.1.1 Lifecycle

The Self-Sovereign Identity (SSI) lifecycle describes the stages of creating, managing, and using an individual's digital identity within a decentralized identity management system. An entity or subject is either a person or an organization that performs a role in a situation [4]. In the context of digital identities, as shown in Figure 3, the entities can perform the following roles [13]:

- The issuer is a role fulfilled by an entity that asserts claims about one or more subjects, creates a verifiable credential from those claims, and transmits the credential to the holder.

- Holder is a role that an entity can play by possessing one or more verifiable credentials and generating verifiable presentations from them.

- Verifier is another role played by an entity that receives one or more verifiable credentials, optionally within a verifiable presentation, for processing.

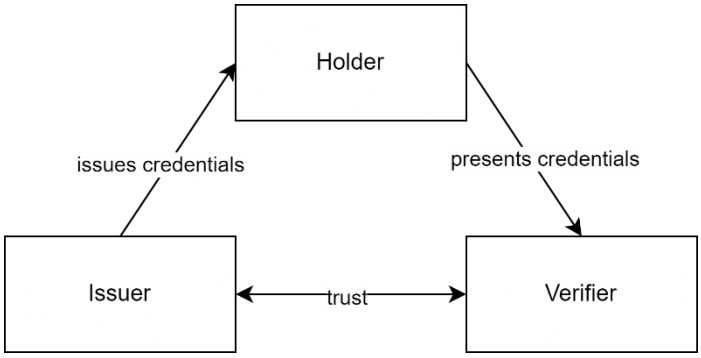

Figure 3 - Different entity roles interaction in the SSI lifecycle

Each entity in the SSI system is assigned a unique decentralized identifier to distinguish it from others in a digital context. The lifecycle begins with DID creation, followed by the issuance of verifiable credentials by an issuer making claims about the subject's identity [4]. The holder is charged with managing their verifiable credentials, which verify their identity or identity attributes. When the holder wants to share credentials with a verifier, they only share the minimum data required. The verifier then evaluates this new information, ensuring it's not something they already possess, to confirm its authenticity. This process usually involves cryptographic validation of the DID to check the legitimacy of the information provided.

3.1.2 Principles

Self-sovereign identity is a transformative way for people to control and own their identity on the internet. Traditional identity systems follow a 1 tiered model where someone's identity is governed by a centralized authority. However, SSI is about individuals, and their ability to manage their identity data through trust, privacy, and interoperability while being able to fully self-manage & control it. SSI is a trust framework consisting of 10 concepts [14] [15]:

- **Existence**: Users have an inner, unique identity that can never fully be captured online. A self-sovereign identity just makes parts of this identity public and accessible.
- **Control**: Users should have complete control over their identity, including the ability to update, hide, or share it. They should decide whether to be private or public, but they can't control what others claim about them.
- **Access**: Users should always be able to see and retrieve their identity information without hidden data or intermediaries. They may not modify all information, but they should be aware of it.
- **Transparency**: The systems managing identities must be open and clear about how they work. The algorithms should be free, open-source, and accessible for anyone to examine.
- **Persistence**: Identities should last if a user wants, and changes like key updates are acceptable. A user can dispose of their identity or update claims over time.

- **Portability**: Identity information should not be controlled by a single organization. Users should have the ability to take their identity wherever they go, ensuring they remain in control.
- **Interoperability**: Identities should work across various systems and countries. The goal is for identities to be usable globally while still allowing the user to maintain control.
- **Consent**: Users must agree to share their identity, and any information linked to it. No data should be shared without the user's consent, even if others make claims about them.
- **Minimalization**: When sharing information, only the minimum necessary data should be disclosed. For example, only sharing someone's age, not their exact birth date.
- **Protection**: The user's rights should always come first. If there's a conflict, the system should prioritize individual freedoms and use secure, decentralized methods to authenticate identities.

It should be noted that such principles, as proposed in article [14] and referenced in relevant literature, are not binding, but a guideline for building identity systems that respect personal freedom. The intention of embracing self-sovereign identity through these principles is to create a more equitable digital identity space to reduce dependency on centralized identity providers and preserve personal privacy and data sovereignty.

It is equally important to comply with global regulations such as the General Data Protection Regulation (GDPR), which sets out the rules for processing personal data to respect the privacy and rights of individuals. It considers principles such as fairness, transparency, data minimization, accuracy, security, and accountability to ensure that personal data is collected and used responsibly and only for the stated purpose [16]. The ten principles of self-sovereign identity align with the GDPR principles by giving individuals full ownership of their data, allowing them to control what is shared and for what purpose, keeping data collection to a minimum, and ensuring security through decentralization and encryption.

3.1.3 Hyperledger Aries

Self-Sovereign Identity technologies empower individuals to manage and control their digital identities in a decentralized, secure, and privacy-preserving manner. Founded in 2019, Hyperledger Aries is a comprehensive toolkit designed for decentralized identity solutions. It enables the issuance, storage, and presentation of verifiable credentials (VCs) while preserving privacy and facilitating secure, confidential communication between parties, and its key features include [2] [17]:

- **Privacy Preservation**: Aries provides owners with full control over their data. It supports selective disclosure, allowing individuals to share only the minimum information required, and zero-knowledge proofs (ZKPs), allowing users to prove certain claims (e.g., age) without revealing sensitive data (e.g., date of birth).

- **Confidential, ongoing connections**: Aries supports encrypted, peer-to-peer messaging, facilitating secure, ongoing relationships between parties. It also enables proactive notification of credential revocation or updates.
- **Ease of deployment**: Aries provides out-of-the-box agent frameworks in multiple programming languages, including the credo [18] TypeScript framework.
- **Flexibility**: Aries is highly customizable, supporting various credential types such as Hyperledger AnonCreds and JSON-LD W3C VCs, providing high-level APIs for easy future modifications.

In addition to Hyperledger Aries and credo-ts, several other SSI technologies are major contributors to the decentralized identity landscape, including Veramo [19], Sovrin [20], and Microsoft's ION [21]. These technologies are distributed identity management solutions and have different features and levels of integration, such as user control over personal data and the use of verifiable credentials. OpenID4VC [47] was considered because it is a specification that uses the Aries framework to create verifiable credentials in various formats.

Aries is designed to be ledger-agnostic, meaning that it can work with various distributed ledger technologies (DLTs) or blockchains for identity management. While Aries itself doesn't inherently enforce a specific ledger, it's often used in conjunction with Hyperledger Indy [22], which provides a DLT-based infrastructure for DID management and credential definitions [23]. Both projects are part of the Hyperledger ecosystem [24], so it makes sense to use both together.

A comparative study of decentralized identity solutions [25] considered five critical factors: scalability, reliability, security, adaptability, and cost. Hyperledger Indy emerged as a leading solution, particularly for enterprise use, due to its strengths in these areas:

- **Security and Privacy**: The Scalability value of Hyperledger Indy is greater in comparison to other decentralized systems because it uses Redundant Byzantine Fault Tolerance (RBFT). It enables handling up to 33% faulty nodes while processing around 300 transactions per second (TPS).
- **Interoperability**: Indy offers strong reliability through its fault-tolerant consensus, allowing the network to run even if a third of its nodes fail or act maliciously.
- **Security**: The security of Indy relies on cryptographic methods, such as zero-knowledge proofs, which allow users to verify their identity without revealing personal details. Its blockchain network avoids the security issues seen in public chains and the weaknesses found in smart contract platforms like Ethereum.
- **Adaptability**: Compared to other decentralized platforms, Indy is more adaptable for enterprises because it supports open standards and allows integration with existing IT systems via custom APIs or plugins.
- **Cost**: Although the initial investment is higher with Indy due to the need to set up validator nodes and expertise, the unpredictability of Ethereum's transaction fees is not an issue with Indy. Stable costs are an advantage because they make Indy more predictable and sustainable for organizations and businesses.

3.1.4 Practical Applications

Self-sovereign identity is transforming the digital landscape by creating a secure and trusted environment where individuals, organizations, and institutions can seamlessly interact. In this system, individuals (holders), organizations (issuers), and verifiers use decentralized cryptographic mechanisms to verify identities and credentials. SSI offers a wide range of practical applications across various sectors, enhancing privacy, security, and efficiency [26][27]:

- **Education:** Digital credentials are securely stored in a personal wallet and shared with educational institutions during application processes. Academic documents such as transcripts or diplomas are stored and verified by SSI to validate authenticity and prevent fraud.
- **Finance:** For online transactions and account access, verifiable credentials offer a secure and seamless way to authenticate identities. Financial institutions leverage SSI for efficient customer onboarding, streamlining the process by eliminating the need for extensive identification checks.
- **Healthcare:** When accessing healthcare services, individuals can share only the relevant medical details, protecting their privacy. SSI enables selective data sharing, ensuring that only necessary information is disclosed while maintaining security and trust.
- **Voting:** SSI provides secure and transparent voting through authentication without invading privacy. Voting is done electronically by the voter, with the voter's identity hidden, reducing the likelihood of fraud. The application adds credibility to e-voting, simplifies the voting process, and ensures that the process remains secure.

SSI is not limited to one application but has the potential to revolutionize different aspects of society [26]. Figure 4 illustrates how the three SSI actors interact in some of these areas.

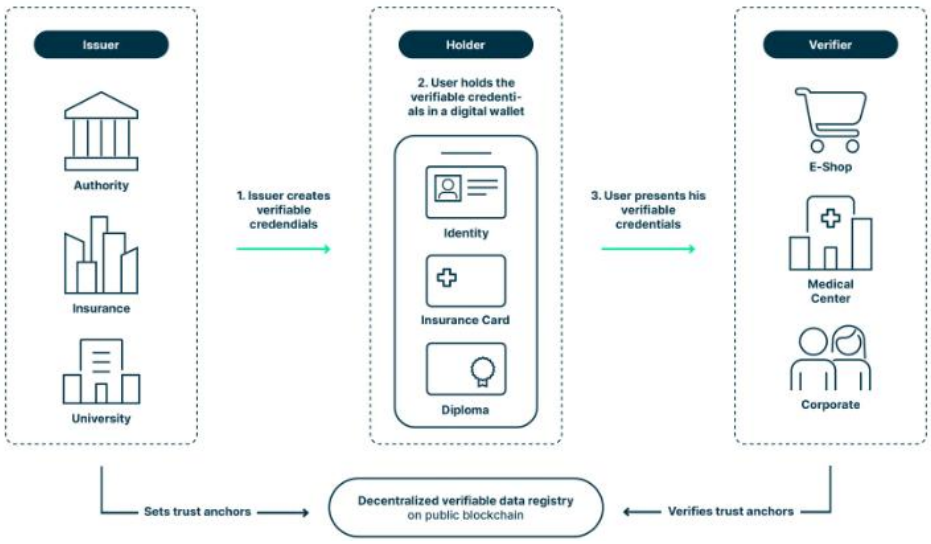

Figure 4 - Interaction of SSI actors in practical applications [26]

For instance, a government agency may issue an ID card, an insurance company could offer a digital insurance certificate, and a university might grant a diploma. These credentials are safely stored in a digital wallet, giving individuals full control over their data and removing the need for centralized databases. When authentication is necessary, users can present their credentials for validation. For example, a company can verify a candidate's diploma directly from the wallet during the job application process. Similarly, an online pharmacy can confirm a user's identity before a purchase, and a healthcare provider can authenticate an insurance card before delivering treatment, all with real-time verification [26].

In systems where voting is involved, it is essential to meet several key requirements. These include ensuring the certainty and legal validity of the voter's identity, providing public verifiability, and safeguarding both security and secrecy [27].

3.2 Identity Management Systems

Identity management concerns a framework of policies and technologies designed to ensure that access to specific resources is granted only to authorized individuals within a defined context. However, as society becomes increasingly interconnected and digitalized, accompanied by a substantial rise in the variety and number of identities and systems requiring management, it is essential to reconsider traditional approaches [7]. These systems are designed to handle the identity lifecycle across many platforms and applications so that users can access the right resources at the right time. Beyond authentication and authorization, their capacity allows responsibilities, permissions, and other custom data to be managed creatively.

Attempts have been made to propose identity management solutions that allow users to manage their own identity [7], demonstrating that identity trust is an area of interest that is not covered by current standards and codes. Credential verification is a challenge that occurs in many industries, like the medical field, social media and job platforms, as security risks and unethical behaviours are compounded by the growing availability of false credentials. When using an external identity provider, users outsource all management processes to the service provider, such as the privacy of their data and the accuracy of their information. The user is only required to perform a few simple actions to facilitate the exchange of information. One of the challenges of SSI is not only making its use possible but also ensuring that it is convenient [14], which includes both the creation and management of multiple digital identities for an individual and the management of the credentials that maintain the validity of those identities.

In addition, mechanisms are needed to enable and record events involving the transfer of rights and permissions to other people and services. The challenges are particularly pronounced when attempting to minimize reliance on trusted third parties. These measures are critical to mitigate the risk of third parties impersonating the DID owner, which requires strong authorization and auditing systems to prevent unauthorized changes [1].

4 State of the Art

This chapter reviews the project's main aspects and provides clear guidelines and selection procedures. It is divided into two main sections: a systematic literature review and a discussion of research questions. It also discusses variations in data analysis procedures for systematic literature studies.

4.1 Systematic Literature Review

A systematic literature review (SLR) is a methodical approach used to identify, analyze, and interpret existing evidence related to specific research questions in an objective and, to some extent, reproducible manner [29]. The purpose of an SLR is to provide the reader with an up-to-date overview of the current literature on a particular topic. Its primary aim is to assess the current knowledge on research question topics, thereby highlighting areas that warrant further investigation [30].

4.1.1 Research Goals and Research Questions

To systematically gather information on the project's area of interest, the following main research objectives were identified for this review:

- Investigate the potential security issues associated with decentralized identifiers within self-sovereign identity frameworks.
- Analyze the impact of verifiable credential issuance and revocation processes on system trust and user autonomy.
- Explore the challenges and solutions for ensuring seamless interoperability between different self-sovereign identity approaches.

Based on the objectives defined above, the research questions [31] guiding this line of work are as follows:

- RQ1: How can self-sovereign identity technology be integrated to mitigate security risks associated with decentralized identifier management in systems that demand secure and scalable identity solutions?

This is a specific and brief research question, as it addresses the contribution of the given integration of self-sovereign identity technology to addressing the security issues of decentralized identifiers. It addresses two primary issues, integration and security, in systems that require effective identity management solutions. The scope is pertinent and broad enough to support different integration methods and their implications. The question is neither too simple nor too complex, as it requires a detailed study of integration methods and their effectiveness, but within the limits of available time and resources. It is researchable because there is a wealth of academic literature, case studies, and technical documentation on SSI and decentralized identity security. In addition, this question encourages an analytical approach by requiring an evaluation of integration and risk mitigation strategies, moving beyond mere description to a more comprehensive examination.

- RQ2: How can verifiable credential issuance and revocation be designed into systems integrating self-sovereign identity technology?

This question focuses on the design of verifiable credential management, specifically the issuance and revocation processes, in systems that incorporate SSI technology. The scope is appropriate, focusing on a well-defined aspect of SSI while allowing exploration of multiple dimensions such as scalability, trust models, and technical constraints. It is researchable due to the availability of substantial academic and technical resources on SSI and VC. By focusing on design, the question naturally leads to an analytical examination of challenges, potential gaps, and areas for improvement, rather than simply describing the operational aspects of verifiable credentials.

- RQ3: How can one self-sovereign identity technology be integrated into systems to enable interoperability with other self-sovereign identity approaches?

This research question focuses on the design of integration strategies to achieve interoperability between different SSI approaches, addressing a critical technical challenge in the field. Its scope is well balanced, as interoperability is a substantial problem, with emphasis on design strategies narrowing the focus enough to make the question manageable within time and resource constraints. The question avoids being too simple by requiring the exploration of technical solutions, but is not overly simplistic as it is consistent with current discussions in SSI research and practice. This question encourages an analytical approach, requiring a critical evaluation of how systems can be designed to overcome interoperability challenges and a synthesis of potential solutions, rather than simply describing existing approaches.

4.1.2 Data Source and Initial Search

The search was conducted through the continued use of the trail-and-error method, an approach that uses metasearch engines to refine and test queries. It starts with initial keywords or partial phrases as candidates to systematically evaluate whether the query produces meaningful results, whether the keywords or combinations produce any hits, and whether the query is precise enough to ensure that results are relevant to the specific domain of interest [29]. The B-on search engine served as the primary source. Additionally, Google Scholar alerts were set up to capture newly added papers, to be manually incorporated into the collection for research topics.

For the initial search, the term "self-sovereign identity" was identified as the primary word of the initial filter to narrow down the search results. A total of five additional words were selected for each research question. The initial words to be searched in the full text of the articles and the number of B-on results corresponding to all combined words separated by commas were then determined for each research question (Table 5).

Research Question	Words	B-on initial search
RQ1	self-sovereign identity integration decentralized identifier security risks issues	1094 results
RQ2	self-sovereign identity verifiable credential revocation issuance limitations results	229 results
RQ3	self-sovereign identity integration systems interoperability approaches design	1279 results

Table 5 - Initial trail-and-error Search Results

The purpose was to achieve a deeper understanding of the available results and provide the necessary information to develop inclusion and exclusion criteria for a more focused search, thereby narrowing the results.

4.1.3 Search Terms

The keywords most pertinent for the problem were self-sovereign identity, integration, security, decentralized identifier, verifiable credentials, issuance, revocation, and interoperability. These terms highlight the fundamental concepts and challenges of creating a secure and user-focused identity management framework.

- **Self-Sovereign Identity (SSI):** Refers to a user-controlled digital identity model that enables individuals to manage and own their identity without reliance on centralized authorities.
- **Integration:** Highlights the importance of ensuring seamless functionality among various systems and platforms to support SSI.
- **Security:** Emphasizes the protection of identity data and credentials from unauthorized access, misuse, and breaches.
- **Decentralized Identifier (DID):** A foundational element of SSI, representing unique identifiers that are not tied to centralized systems but are instead anchored in decentralized infrastructures like blockchains.
- **Verifiable Credentials (VC):** Digital statements issued by trusted entities that can be cryptographically verified to prove attributes or claims about an individual or entity.
- **Issuance:** Creating and providing verifiable credentials to users in a secure and trustworthy manner.
- **Revocation:** The mechanism to invalidate or withdraw verifiable credentials when they are no longer valid or secure.
- **Interoperability:** The ability of diverse systems, organizations, and technologies to work together effectively, ensuring that SSI solutions can be universally adopted and function across different ecosystems.

The following query (Code 3) was created using these keywords on the B-on database:

```
(
    "Self-sovereign identity" OR "SSI"
) AND (
    ("Integration" OR "System integration") AND
    ("Security" OR "Data security" OR "Identity security")
) AND (
    ("Decentralized identifier" OR "DID") AND
    ("Verifiable credential" OR "VC")
) AND (
    ("Issuance" OR "Credential issuance") OR
    ("Revocation" OR "Credential revocation")
) AND (
    "Interoperability" OR "User-centric identity" OR
    "Identity management framework" OR "Digital identity"
)
```

Code 3 - Search Query

4.1.4 Inclusion and Exclusion Criteria

Selection criteria were then established to minimize bias and increase objectivity. The purpose of defining inclusion and exclusion criteria is to refine the data set, select relevant studies, and narrow the results by ensuring that the procedures and steps presented are rigorous and reproducible [29]. The inclusion (I) and exclusion (E) criteria were designed to address the research questions. The following inclusion and exclusion criteria were chosen:

- **I1:** The paper explores the use of self-sovereign identity.
- **I2:** The full text of the source is available and has been the subject of peer review.
- **E1:** The paper was published before 2019 because Hyperledger Aries SSI technology was founded this year.
- **E2:** The paper is not freely or institutionally accessible.
- **E3:** The paper is not published in English.

4.1.5 Data Collection and Results

For data collection purposes, the PRISMA [32] process, as shown in Figure 5, is used to retrieve relevant articles. The process begins by recording the total number of articles found. It also ensures transparency in the selection process by carefully documenting the decisions made at each stage of the systematic review. Articles are also counted at each stage.

The search begins with a review of the total number of records retrieved from various sources, such as the databases used in this research, such as b-on. The process further involves the number of records that were excluded according to specified parameters, e.g., date of publication, language, or other specified criteria, thus detailing an open and reproducible searching process.

The second stage is screening, during which the titles and abstracts of the remaining records are examined. This stage eliminates irrelevant studies, leaving only those that may be relevant. A detailed manual check confirms that the trials meet the criteria for the next stage(s).

After screening, the eligibility stage involves reviewing full-text papers for compliance with the inclusion criteria set for the review. Studies are examined for relevance, methodological quality, and outcomes. Articles that do not meet the criteria are excluded and recorded.

In the final phase, studies that meet all criteria are recorded and included in the final analysis. These studies are synthesized to contribute to the systematic review content and provide the evidence base for the conclusions drawn.

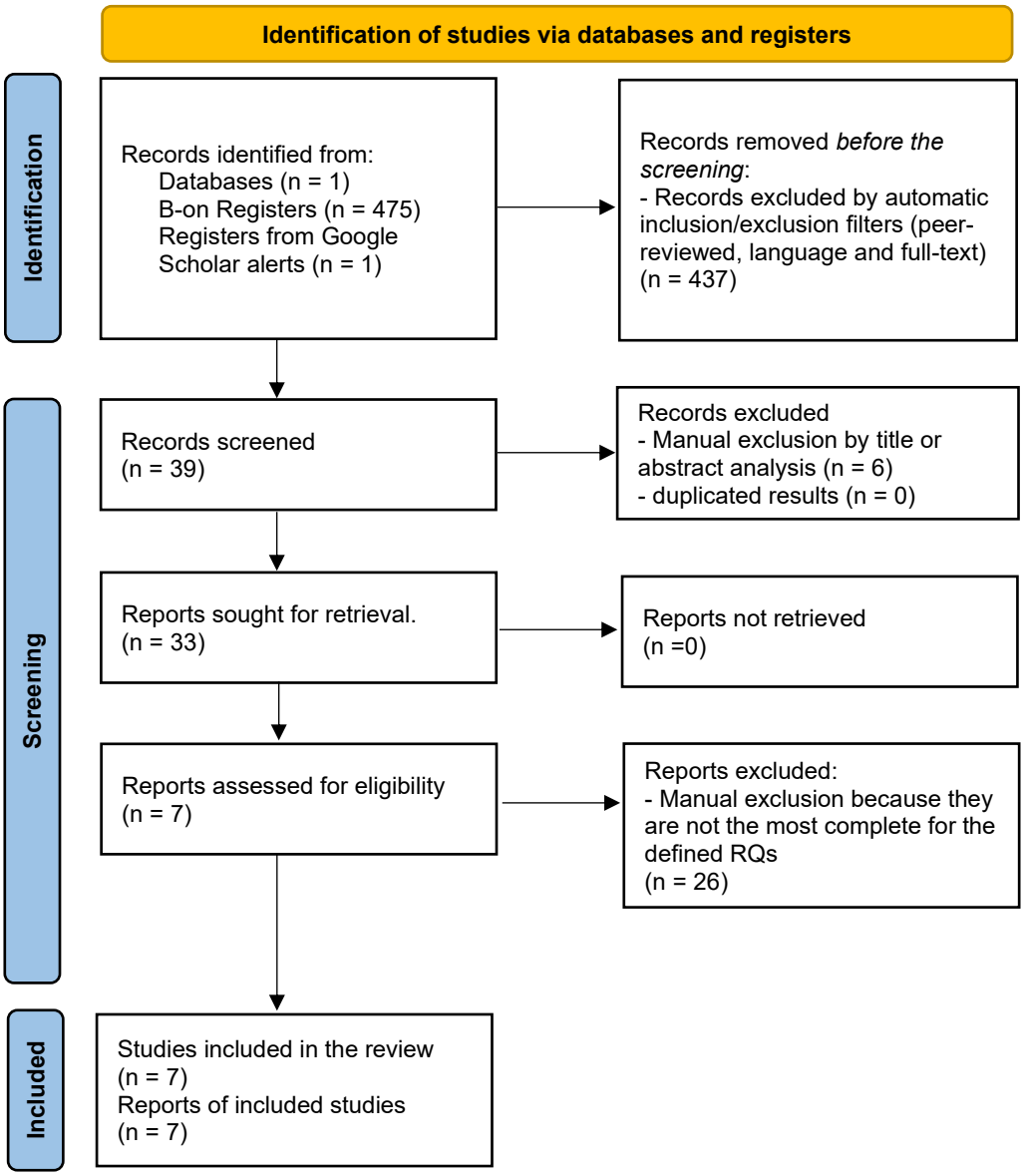

Figure 5 - PRISMA process executed

Upon completion of this comprehensive process, a thorough review resulted in the identification of the following studies, which are detailed in Table 6 and separated by research question.

Research Questions	Studies
RQ1	[33][34][35]
RQ2	[36][37]
RQ3	[38][39][35]

Table 6 - Study results for each research question

4.2 RQ1 - Decentralized Identifier Management

The study selected [33] for RQ1 presents a blockchain-based framework that utilizes decentralized identifiers for patient authentication and consent management in Electronic Health Record access through verifiable credentials. It outlines the procedures for generating DIDs, setting up authentication credentials, and establishing workflows for issuing and verifying credentials within the Electronic Health Record (EHR) ecosystem. The framework utilizes the Hyperledger Indy blockchain and the Aries library for implementation. This research evaluates the effectiveness of these workflows alongside a security analysis.

The main objective of the present research is to evaluate the security features of decentralized identifiers (DIDs) that are used in patient authentication and consent management processes. Dynamic key rotation is facilitated by these identifiers through the ability to create new DID documents without compromising the trust established in self-sovereign identity, and at the same time, following cryptographic best practices. Combining privacy-enhancing credential sharing and verification using zero-knowledge proofs serves to further strengthen the security of such identifiers by concealing the identity of the credential holder from the verifier.

This study identifies common security problems related to solutions that incorporate blockchain, focusing on the ones that include smart contracts. Smart contracts bring with them issues associated with misbehaviour, potential coding bugs, or relying on uncompiled external data, and when mixed with identity solutions, these can compromise user identity autonomy. The proposed framework avoids these vulnerabilities by not using them for identity management, providing users with a more independent and secure identity solution [33].

The selected paper proposes a system based on Self Sovereign Identity (SSI) to enable Know Your Customer (KYC) activities using smart contracts [34]. In the proposed system, users receive a token from a metaverse service provider based on a decentralized identifier credential issued during KYC verification. Avatar identities are verified using smart contracts that ensure user privacy and security. The platform provides tools to generate Zero-Knowledge Proof (ZKP), making it easy for non-technical users to use ZKP. An integrated wallet is also available for efficient management of DID credentials and tokens.

An analysis of privacy and security concerns in DID management highlighted several key issues. Traditional systems centralize user information with service providers, posing significant risks in the event of hacking. In the proposed system [34]:

- Only the DID issuer, who may also perform auditing, holds user information.
- Service providers verify derived proofs without accessing personal data, ensuring user privacy.
- Suspicious activities can be traced by querying the DID issuer using the credential number stored in a token's external link.
- Token revocation is supported to protect privacy further. To address risks of identity inference from multiple SBTs in a wallet, the system employs standards that allow metadata updates upon user request, effectively breaking identity links.

The other selected study [35] presents a framework for data exchange and identity verification, which uses decentralized identifiers in conjunction with blockchain technology. Data is stored in the decentralized Interplanetary File System (IPFS), while the hash value is securely and automatically stored in the decentralized networks to maximize storage without sacrificing information integrity.

This W3C-compliant solution, presented in the selected paper, is designed to provide interoperability between different SSI approaches, enabling auditability of the operations performed and eliminating single points of failure and security vulnerabilities associated with centralized approaches.

The study evaluates the framework's effectiveness in terms of security and availability. It highlights the following key benefits [35]:

- **Security and Privacy Self-Sovereignty**: Decentralized identity gives users control of their information, fewer dependencies on central authorities, and fewer risks of data breaches and abuse.
- **Verifiability and Transparency**: Once a DID has been registered on the blockchain, it is transparent and verifiable to all, with the blockchain's immutability guaranteeing the correctness and integrity of information.
- **Scalability and Compatibility**: DID is designed as per W3C standards, providing cross-system and cross-platform identity verification. It enables a user to use one identity on multiple systems, making it easier to manage and improving the user experience.
- **Persistence and Availability**: Blockchain ensures identities are persistent and unalterable once they are stored, ensuring long-term dependability, and the decentralized nature of IPFS ensures data availability even when nodes crash, providing consistent service and system resiliency enhancement.

The selected RQ1 studies demonstrate that self-sovereign identity technology is effective in mitigating the security issues associated with decentralized identity management and provides secure and scalable identity solutions. Drawing on blockchain and decentralization techniques, such approaches target inherent issues like impersonation fraud, credential compromise, and unauthorized data exposure. The first paper provides DIDs together with Electronic Health Record systems, with adaptive key rotation, zero-knowledge proofs, and selective disclosure. Also introduces the security advantage of smart contracts, utilizing them without revealing their weaknesses, in a manner intended to allow for independent and secure management of identity. The second trial streamlines know-your-customer procedures through SSI utilizing intelligent contracts, with user anonymity sustained through the separation of personal information from verification through zero-knowledge proving, token revocation, and metadata updating to evade identity linking. The third test uses decentralized storage to ensure data integrity, availability, and scalability while complying with W3C standards for cross-platform identity proofing. Taken together, these platforms demonstrate how SSI puts people in control of their

identity, minimizes reliance on centralized systems, and introduces scalable, secure, and privacy-preserving identity solutions.

4.3 RQ2 - Verifiable Credentials Issuance and Revocation

The primary objective of the first study [36] was to evaluate and implement self-sovereign identity (SSI) features using open and closed-source frameworks within a building access management scenario. This scenario required compliance with health risk and security questionnaires during the COVID-19 pandemic. The research evaluated whether user privacy could be protected through verifiable credential issuance and revocation and examined the potential need to move away from centralized storage of personal data.

The developed SSI system includes a credential management application for issuers and a mobile application for users, designed to store and exchange verifiable proof of credential presentation. This system enables seamless interaction with business processes via QR (Quick Response) codes. To increase efficiency and demonstrate the viability of SSI, third-party frameworks were integrated into the solution to automate access verification for public buildings while maintaining user confidentiality. The system uses zero-knowledge proof techniques to validate credential schemas and ensure the trustworthiness of credential sources without exposing sensitive user data.

A key feature of this system is the integration of business rule validation into the credential issuance and verification process. Credentials are securely exchanged via a user's mobile wallet and are autonomously verified based on pre-defined business rules stored in the verifier's database. These rules are derived from a maintenance table within the database, eliminating the need for additional source code development by enabling configuration-based updates [36].

The mobile wallet interface provides users with a comprehensive view of their stored credentials. A card detail widget offers three core functions: checking revocation status, emailing a credential proof, and sharing a full credential proof via QR code. An additional credential is accessible through intuitive gestures: a single tap reveals all fields and values associated with the credential, while a long press displays the credential in plain text JSON-LD format, enabling manual distribution as an alternative method of sharing.

The second study [37] proposes a privacy-enhanced distributed identity management scheme solution with sequential aggregate issuance, threshold traceability, and revocability in the setting of multiple credential issuers and regulators. Efficient revocation is defined as a significant challenge in VC schemes. The solution, shown in Figure 6, adheres to the W3C standards for identity management systems, incorporating decentralized identifiers and verifiable credentials to enable hierarchical and user-centric identity authentication under the SSI paradigm. A dynamic accumulator and a smart contract allow hierarchical and efficient revocation of credentials, while zero-knowledge proofs validate without much exposure of private information.

A comprehensive security evaluation identifies that the proposed scheme satisfies critical security needs, with comparisons with existing solutions demonstrating their robustness. In addition, theoretical and experimental analysis confirms the effectiveness and practicability of the scheme in industrial Internet environments.

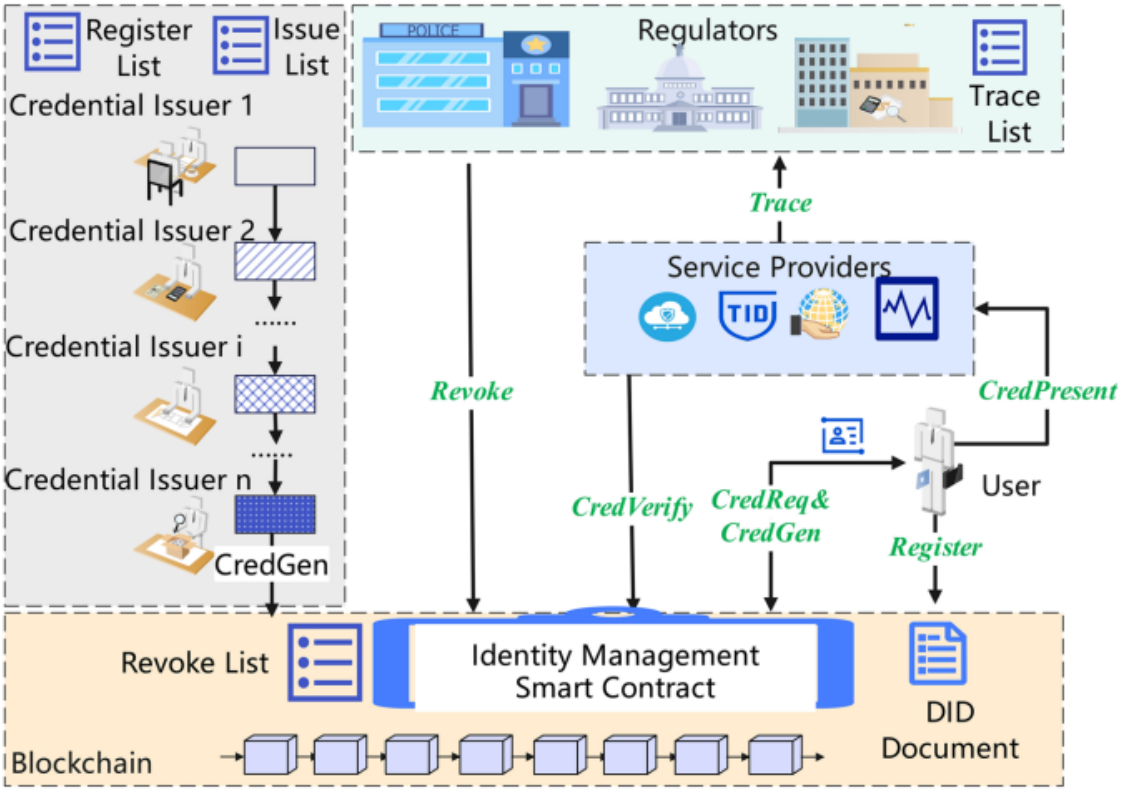

Figure 6 - Identity management system model [37]

The system consists of five major entities: credential issuers (CIs), users (U), regulators (RGs), service providers (SPs), and the blockchain (BC).

The blockchain operates an identity management infrastructure called Identity Management Smart Contract (IMSC), which acts as a reliable ledger for recording all identity management operations conducted by the entities. Credential issuers, made up of various issuing authorities, can sequentially offer credentials [37]. The solution developed includes the revocation process to ensure that credentials can be revoked, thus preventing invalid users from accessing the service and increasing overall identity management security procedures.

4.4 RQ3 - Self-sovereign Identity Interoperability

Interoperability within self-sovereign identity (SSI) systems ensures that verifiable credentials can be utilized widely across different platforms and domains [38]. This initial study for RQ1 shows a novel framework for online document verification using SSI technology that focuses on addressing interoperability problems in digital document verification systems. By adhering to

SSI standards, the framework eliminates intermediaries and promotes direct trust between ecosystem participants, improving system efficiency, interoperability, and privacy.

Most existing SSI solutions rely on two widely adopted open standards established by the W3C. The proposed framework aligns with these standards and incorporates vendor-neutral technology components, maximizing interoperability and transitive trust while reducing the risk of vendor lock-in. A key to the proposed framework is a JSON-LD schema definition recorded on a public blockchain. It allows verifiers to access and assess these schemas, ensuring semantic interoperability - the ability of systems to exchange data with shared and clearly defined meanings. Semantic interoperability goes beyond syntax to include metadata, linking data elements to a controlled and shared vocabulary to ensure data and its meaning are transmitted accurately [38].

W3C standards and JSON-LD create an efficient system that ensures verifiable credentials (VCs) can be widely and securely adopted while maintaining user control. These standards contribute to interoperability, and diverse SSI technologies to seamlessly utilize VCs and decentralized identifiers (DIDs). By adhering to W3C specifications, the system achieves technical and semantic interoperability, providing consistent data exchange and common understanding across platforms and domains [35].

Interoperability is one of the main drivers for success for self-sovereign identity (SSI) systems, as it facilitates seamless data exchange and integration with different platforms and industries. Interoperability for SSI has to be attained through a design process that achieves both technical and semantic alignment [39] and involves using widely adopted standards, ensuring compatibility between different ecosystems, and also facilitating collaboration between stakeholders [39]:

- **Adherence to W3C Standards**: W3C standards, such as DIDs and VCs, serve as the cornerstone for interoperability by providing a standardized technique for establishing, managing, and verifying identities and credentials, allowing disparate systems to communicate seamlessly.
- **Semantic Interoperability through Credential Schemas**: For effective data exchange, the semantics of VCs must be standardized. Credential schemas, defined using JSON-LD, describe the structure and meaning of data in a universally understandable way.
- **Adaptable and Transparent Schema management:** It is essential that systems are built with robust schema management to facilitate long-term interoperability, and schema update and versioning mechanisms are needed to adapt to new requirements while maintaining backward compatibility.
- **Consensus Building**: Involving trusted organizations in the schema creation and validation to ensure credibility and widespread adoption.
- **Decentralized Data Transfer**: In SSI systems, data is transferred by the identity holder via their digital wallet, eliminating direct communication between service providers. This decentralization reduces integration complexity but necessitates consensus on the data attribute semantics encoded in VCs.

- **Integration with Existing Ecosystems**: To achieve interoperability in systems where SSI is integrated, systems should incorporate elements of existing ecosystems. Gradually introducing SSI components alongside traditional systems allows for smoother integration and builds user confidence.

In conclusion, interoperability among self-sovereign identity systems needs to be a careful and deliberate process that balances strict compliance with well-known standards, e.g., those published by the W3C, with good schema management and decentralized data transfer procedures. With widely accepted models like DIDs and VCs and providing semantic interoperability in the form of JSON-LD schemas, SSI allows secure and user-managed data exchange between domains and platforms. The incorporation of SSI technology into current environments, along with cooperation between organisations that are most trusted, is fundamental to making certain that systems become flexible, clear, and efficient in sustaining compatibility over time.

5 Project Selection and Analysis

This chapter focuses on the analysis and design of the developed implementation. As mentioned in Section 1.2, the prototype started from an open-source application, focused on voting, one of the practical applications identified in Section 3.1.4. After, self-sovereign identity components were integrated into this project to explore and incorporate the lifecycle and functionalities of self-sovereign identity management, such as issuing and revoking verifiable credentials. The chapter starts by explaining the design of the initial project, mentioning the established architecture. It then explains the process for integrating the self-sovereign identity components into the system.

5.1 Initial Project

As stated in Section 1.2, a project was selected to integrate lifecycle and functionalities of self-sovereign identity management, including the issuing and revocation of verifiable credentials. The objective is to explore this identity management approach using the Hyperledger Aries, evaluate the solution based on predefined metrics, and conduct relevant experiments. When selecting the project, some criteria were considered. As such, the project must have:

- The code is publicly accessible and comes with a license that permits its use.
- Actively maintained, with commits made within the past year.
- Include both functional and architectural documentation.
- Topic identified as a relevant practical application of self-sovereign identity.
- Conducted exploratory tests to assess its functionalities.

The chosen project was example-voting-app, a Docker-based example of a distributed application for voting procedures and result consultation [40]. It serves as a simple demonstration of the different components and programming languages applied in the voting domain.

The characteristics of the project are the following:

- The repository is licensed under Apache-2.0, actively maintained, with commits in the last five months and over 5,000 stars on GitHub.
- Documentation includes a logical view of the architecture and high-quality usage instructions to ensure a clear understanding and effective implementation.
- Unit and functional tests are available for the application.
- The theme is Voting, which was identified in the Background chapter as a usual application of the self-sovereign identity approach.
- Voting and results are managed in different components, which simplifies self-sovereign identity integration.
- Exploratory tests confirm that the project functions as documented.

Since the initial project is licensed under Apache-2.0, all documentation and implementation must be created and distributed under the same terms [41]. The project repository preserves the original license [8].

In the base system, vote identities are hashed under a minimal anonymization layer with no storage of any metadata for the users. Only the vote selection is stored, not the identity or any voter identification on which the vote was cast. The anonymity of the voters from the first project was ensured, and this feature is maintained in the refreshed version to keep on supporting privacy and following the SSI principles.

Auditability and verifiability are essential for voting systems, but this feature was not included in the initial implementation. It was added in the second iteration to ensure that votes can be verified independently, without compromising voter anonymity, while aligning with the principles of SSI by keeping the voter in control and maintaining transparency and trust in the system.

5.1.1 Structure

As its name implies, the example-voting-app is a system that allows clients to submit votes and check the results of a poll with two options. While the application enables clients to submit multiple votes, it does not restrict or track individual voters. This design makes it more appropriate for demonstration and educational purposes than for use in secure elections. It serves as a basic example to showcase how various components and programming languages can be integrated and managed using Docker. Figure 7 shows the use cases established for this initial version of the project, with two main pages: the page where users enter their vote and the results page, which shows the count in real time.

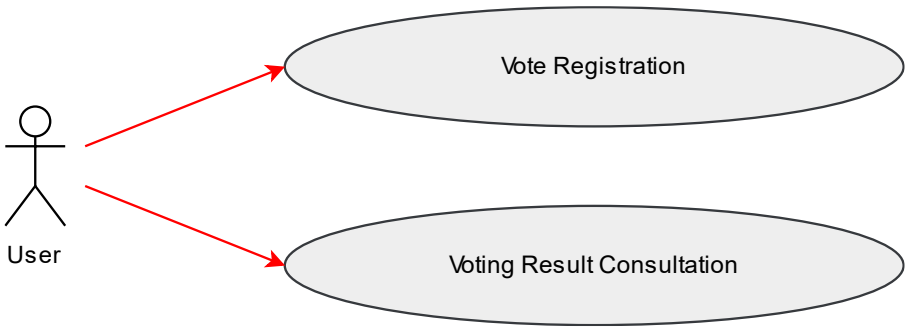

Figure 7 - Use case diagram of the initial project

Table 7 provides a brief overview of the five components that comprise the architecture of this voting system project. It details their functions, the technologies employed, and their interactions within the system.

Component/DB name	Description
vote	Front-end web application, developed in Node.js, that allows users to choose between two options. It ensures that each client may only vote once.
result	Real-time web application, developed in C#, that displays voting results as they are received.
worker	.NET-based background worker service that consumes votes from a Redis queue and stores them in the PostgreSQL database.
redis	In-memory data structure store that functions as a message broker, gathering fresh votes for processing.
postgres	The database that stores the final voting results, backed by a Docker volume to ensure data persistence across container restarts.

Table 7 - Components description of the initial project

The initial project is deployed using Docker Compose for containerization and Kubernetes for managing each component listed in Table 7. For the scope of this work, Docker usage is not a primary focus as it does not play a significant role in self-sovereign identity integration and achieving defined objectives.

5.2 Integration Process

To explore self-sovereign identity management, new components have been developed to explore the SSI lifecycle, focusing on its three actors: issuer, holder, and verifier. The base project's functionalities were maintained and included an Apache 2.0 license that allows reuse. This process follows a structured approach with three main steps:

- Strategy: The approach for integrating self-sovereign identity within the system.
- Requirements: Identification and specification of relevant scenarios where SSI is applied.
- Final architecture: Design of the overall system structure to ensure seamless integration of SSI within the initial project.

5.2.1 Strategy

The initial project focuses on the voting and results domains, which are accessible through a web browser via two primary routes: /vote and /results. It involves testing and documentation. The major goal is to incorporate the self-sovereign identity (SSI) lifecycle and investigate its use in this setting. To do this, more complexity has been added to accommodate and explore decentralized identifiers and verifiable credentials usage.

At its core, the final project is built as an express application using TypeScript. This back-end system is responsible for defining entities and objects. The three key actors in the SSI framework are represented within this system via the voter, the voting authority, and the system. The backend handles verifiable credential definitions, issuance and revocation processes, proof requests, and all necessary cryptographic hashing that underpins SSI systems. It functions as a server, providing API endpoints for interaction. A web application is also being developed to provide a visual representation of how the system works and it is the client of the Express server. Attention was focused on the diminishing SSI on the server side. Each of the three actors have a special page on the web interface where it is possible to visualize and participate in the SSI-based voting experience.

Technically, the front-end part was developed using React with a simple routing mechanism. The server only communicates with the front-end's IP using the axios library, by HTTP/S requests for secure and efficient information transfer.

5.2.2 Additional Requirements

In this work, to apply self-sovereign identity (SSI) technology to a voting system successfully, it is essential to resolve the identified crucial challenges, such as decentralized identifier management, issuance, and revocation of verifiable credentials, and test the performance in practical scenarios. Before this solution was implemented, it was required that the principal

actors in the system and their duties be clearly defined. In the proposed solution, three key actors assume the responsibility:

- Holder (Voter): The individual who votes. The voter holds a decentralized identifier and is presented with a verifiable credential by the Electoral Authority (issuer), which they prove as evidence showing that they can vote. The voter then uses this credential to authenticate to the system and safely cast their vote.
- Issuer (Voting Authority): The issuing entity of verifiable credentials to eligible voters. It verifies the identity of the voters, verifies compliance with voting rules, and can revoke credentials if necessary. It plays a key role in maintaining the integrity of the voting process.
- Verifier (Voting System): The system employed for the verification of the voting credentials of the voter before allowing him to vote. It permits only valid and unrevoked credentials and maintains track of votes in a safe and decentralized environment.

In the context of Hyperledger Aries and its Credo TypeScript implementation, these individuals are addressed as an Agent, a root concept within the framework. An agent functions as a go-between who conducts identity-related activities on behalf of a single identity owner, such as a user, organization, or machine. It communicates with other players via protocols to create relationships, issue credentials, and authenticate identities in a secure and decentralized way [42]. Five use cases were defined to explore and integrate self-sovereign identity into this voting system, with a focus on solving the problems identified in the context and problems Section 1.1. The UML Use Case Diagram in Figure 8 identifies the functional requirements through the definition of use cases. Table 8 describes each identified use case.

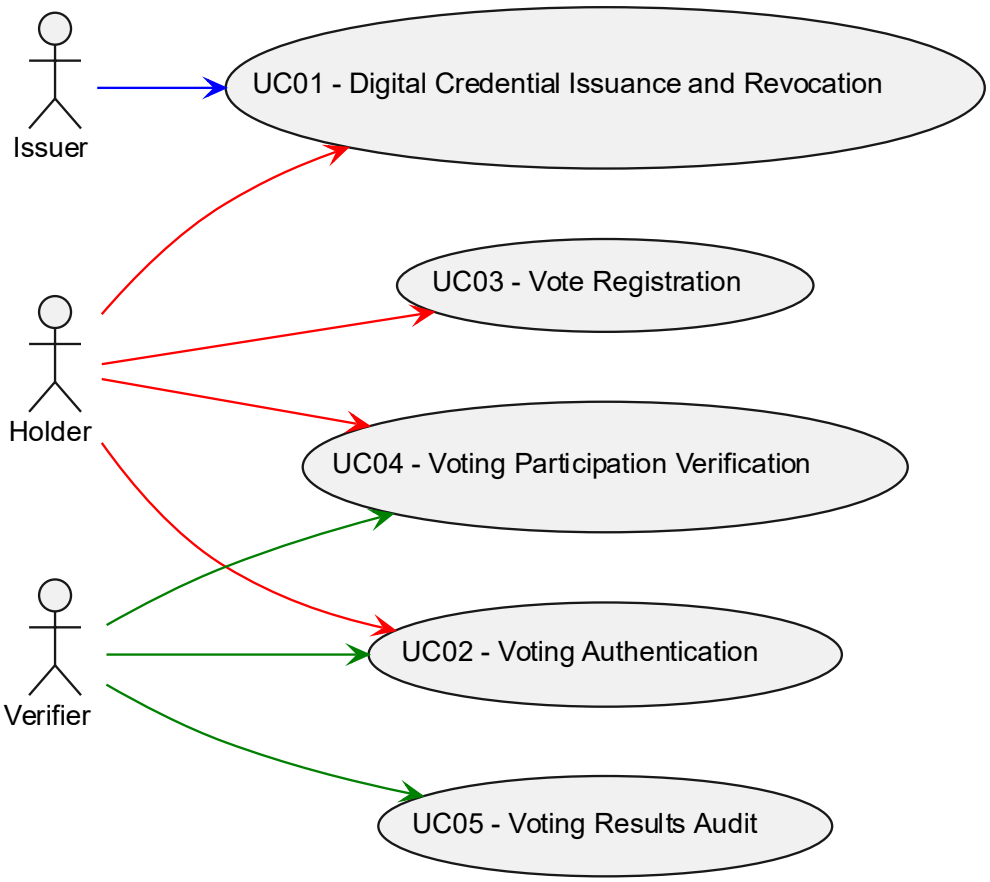

Figure 8 - Final prototype use cases diagram

Use Case Identification	Description
UC01	The initial process in which the Holder obtains a digital credential for voting, issued by the Issuer, that can be revoked later.
UC02	Secure verification of the Holder's identity before the voting process by the Verifier.
UC03	Secure Holder choice registration process, guaranteeing data anonymization and encryption.
UC04	Holder requests and receives a secure confirmation of their participation in the voting poll, issued by the voting system.
UC05	Process of verifying the integrity of the Verifier, ensuring that all votes have been properly counted and that no irregularities have occurred.

Table 8 - Use cases description

5.3 Architecture

After taking a closer look at the initial project structure, this exploration revealed potential problems in bringing the new SSI components together and helped identify ways to strengthen the system. This was a very critical step in looking for potential conflicts, optimizing workflows, and facilitating scalability and performance for future requirements. Figure 9 shows the interaction between the components of the original project.

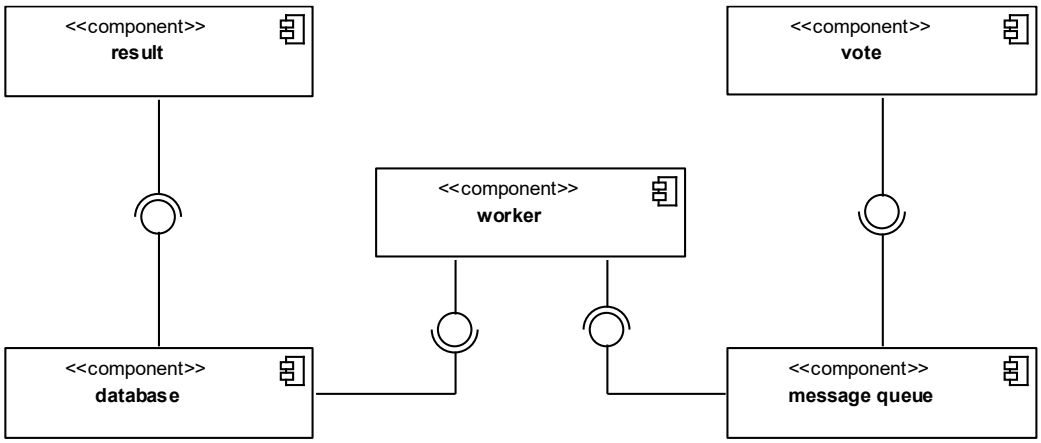

Figure 9 - Components of the initial project

In the new setup, the voting feature is accessible through the new self-sovereign identity system package. However, voting is allowed only if the voter possesses the necessary verifiable credentials to vote. To this, new complexity was introduced by adding a server that holds the operations for this feature. The server acts as issuer, holder, and verifier and ensures that the verifiable credentials are issued properly, stored securely, and verified before going ahead to conduct the vote.

The target should be a streamlined voting mechanism based on a self-sovereign identity management solution that can be achieved by a range of approaches, with the requirements outlined in section 5.2.2. After careful consideration, the decision was made to use an Express server in combination with a web application. This setup allows users to interact with the system in a graphical interface, as opposed to interacting solely through requests.

Given that the initial project is also accessible via REST API requests, it was decided to apply the same approach for the new self-sovereign identity components. The server manages the API endpoints, which are responsible for issuing, holding, and verifying credentials. For the web application, the goal is to allow voting on the initial project and rely on the server for credential issuance and verification, avoiding direct management of these tasks.

This separation of concerns ensures the credentials are supplied by an authority that can be trusted, stored securely, and proven beforehand to unlock activities such as voting. The technology stack upon which one may work in Hyperledger Aries is its TypeScript

implementation, which facilitates the further utilization of this technology for any new components. This TypeScript-based Self-Sovereign Identity Management with RESTful API architectural solution accentuates the following important aspects:

- TypeScript-based: Focusing on the use of TypeScript to build self-sovereign identity management with libraries like credo, credo-core, and express.

- Self-Sovereign Identity Management: Founded on the core activity of handling self-sovereign identities (issuer, holder, and verifier) in the electoral sector.

- RESTful API: Defining the communication method that allows communication between the server, web applications, and other external systems.

With HTTP/S and other protocols, this approach enables end-to-end integration with third-party services while enabling self-sovereign identities of the voting system. Decentralized identity frameworks allow this through the provision of verifiable proof of identity and authentication, eliminating centralized control. Table 9 contains a brief description of each of the components that constitute the solution of the self-sovereign identity management package integrated into the initial project. The UML Component Diagram in Appendix A represents the full prototype components and how they interact.

Component/DB name	Description
web application	Front-end web application, developed with a React framework, that allows users to simulate self-sovereign identity with issuer, holder, and verifier options applied to the voting domain and allows access to the vote web app.
server	A server express that allows the interaction between actors in the SSI framework is represented within this system.
holder	A typescript component representing the individuals who vote in this solution, implemented through the Hyperledger Aries framework.
issuer	Typescript component representing the voting authority in this solution, implemented through the Hyperledger Aries framework.
verifier	A typescript component representing the voting system in this solution, implemented through the Hyperledger Aries framework.

Table 9 - Components description of the SSI management integration package

5.3.1 Issuance

A key architectural consideration in the design of the system involved identifying the best Hyperledger Indy distributed ledger infrastructure to use for the purpose of publishing credential schemas, definitions, and revocation registries. Two options were considered - to use the BCovrin network or deploy a self-hosted Indy pool instance.

BCovrin is an open testnet based in Indy that anyone can use to construct and prototype decentralized identity solutions. The main benefit of using this approach is that users can eliminate the operational overhead of ledger infrastructure and waste less time onboarding and testing credential transactions. Furthermore, BCovrin makes it possible for all distributed participants to participate and communicate publicly [43].

Conversely, a self-hosted Indy pool via Verifiable Organizations Network (VON) entails an in-house deployment of the Indy ledger, managed by an organization or consortium through dedicated validator nodes. It also permits tailored mechanisms and potentially enhanced system availability and reliability, contingent upon robust infrastructure management. This approach can demand substantial technical expertise, significant resource allocation, and continuous maintenance to ensure resilient operation [44].

Table 10 summarizes the trade-offs identified between these architectures, outlining their respective evaluations according to multiple criteria [25][43][44].

Criteria	Public Test Indy Network	Local VON Indy Network
Readiness to Use	Ready-to-use test network with existing nodes	Requires full setup and configuration
Control over the Ledger	No control over ledger state or availability	Full control over network behaviour, configuration, and schemas
Infrastructure Independence	No local infrastructure maintenance	Requires local infrastructure and node setup
Latency and Performance	Potential latency and rate limiting due to shared resources	Reduced latency and better performance in isolated environments
Maintenance Complexity	Low complexity since is managed externally	Higher complexity, requiring setup and ongoing maintenance
Interoperability Testing	Useful for public demos and interoperability tests	Not suitable without external exposure
Stability	May pose stability concerns during critical operations	Stable and reliable in controlled environments

Table 10 - Public Test Indy Network and Local VON Indy Network Comparison

The decision was made to adopt the Public Indy Network (BCovrin) because it is immediately usable and does not require local infrastructure. Deploying a Local VON Indy Network would have required significant resources for setting up and maintaining validator nodes, which was not feasible within the project's constraints. After exploratory testing, the BCovrin Indy Network was favoured over the other approach because of its simplicity and alignment with the project's needs. It was also identified in the literature review [36]. Figure 10 shows a high-level view of the voting system's interaction with the selected ledger infrastructure using the UML language.

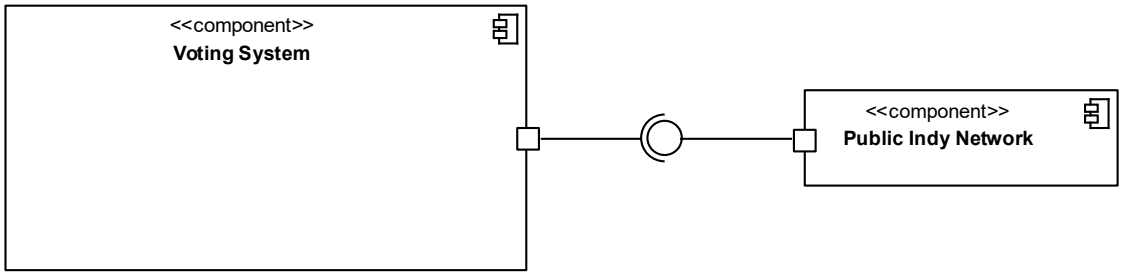

Figure 10 - High-Level view of Public Indy Network interaction

With this choice, to achieve a self-sovereign identity voting prototype, it is necessary to implement a decentralized identity management system where voters receive credentials from an authorized issuer, maintain secure storage of these credentials, and present verification proofs to the voting system before submitting their vote choice.

The end-of-the-end credential lifecycle includes the following steps:

- **Issuance**: Upon verifying voter eligibility, the issuer (voting authority) generates a verifiable credential compliant with the W3C data model and signs it cryptographically.
- **Storage**: The credential is stored by the voter (holder), who retains full control over it.
- **Verification and Presentation**: To vote, the holder presents a verifiable presentation (VP) derived from the original credential. The verifier (voting system) validates the authenticity, issuer signature, and non-revocation status.
- **Authorization to Vote**: If the credential is valid and meets the required criteria, the voter is granted access to cast a vote.

By using verifiable credentials, the system avoids reliance on centralized identity providers or static voter lists, instead enabling real-time, decentralized validation of voter eligibility. The cryptographic foundation ensures authenticity and non-repudiation, while the use of selective disclosure techniques can minimize data exposure, aligning with privacy regulations like GDPR.

Within the broader self-sovereign identity ecosystem, other alternative approaches were studied for implementing each stage of the credential lifecycle:

- **Credential Format**: The JSON-LD/W3C Verifiable Credentials was identified in literature review studies [38] for interoperability and standard compliance. Hyperledger Indy systems use alternative formats like JWT (JSON Web Token) and Anoncreds.

- **Privacy Mechanisms**: For enhanced privacy within the W3C model, algorithm signatures could be used to enable selective disclosure and non-linkable presentations, but within a standard-compliant framework.
- **Transport Layer**: Instead of serving RESTful APIs, which are suitable for prototyping and simplicity, DIDComm protocols can also be used to establish encrypted and peer-to-peer communication between SSI agents (issuer, holder, verifier).

The choice to implement the Anoncreds and JWT models (W3C Verifiable Credentials compliant) with Hyperledger Aries and RESTful APIs was primarily influenced by pragmatic issues related to the integration, developer tooling, and the developing status of standards. These simplifications were applied to the voting prototype to facilitate development and experimentation. The W3C VC data model provides an extensible framework for representing credentials in a way that is both machine and human readable. The goal is to provide a structure that satisfies the requirements of a working system with clear separation of concerns and extensibility for additional, incremental development, such as privacy-preserving credentials, while remaining aligned with the principles of decentralized identity.

5.3.2 Revocation

Following the decision to use the Indy Network as the underlying infrastructure for the revocation mechanism, three architectural approaches were evaluated: Non-cryptographic Revocation Lists, Credential Expiry, and Accumulator-based Revocation with Tails Files.

- **Non-cryptographic Revocation Lists**: Several credentialing systems outside of Indy, or systems only slightly related to it, use a very simple revocation list model. Obviously, with a revocation list model, the simplest means of describing the model is to keep a central (or decentralized) list of revoked credential identifiers. The benefit of this model is that it is simple to implement and reason about. The cost to this model is that user privacy is severely impacted: the revoked credential identifiers are in the clear, and the revoked credential holder metadata is knowable, meaning that some metadata may potentially be linked back to credential holders. This model is, at its core, incompatible with the architectural goals of the project.
- **Credential Expiry**: Instead of using the revocation model, it is possible to issue verifiable credentials that have a limited validity period, attaching an expiration timestamp to each credential. This can allow an easier implementation for some systems in general and guarantee that the verifiable credentials expire in a short period. However, the downside is that this effectively adds operational load since issuers must regularly issue new credentials to maintain continuity.
- **Accumulator-based Revocation with Tails Files**: This approach utilizes cryptographic accumulators to construct non-revealing, dynamic public revocation registries. Each credential issued by an authority has a unique revocation factor. The authority simply maintains a Tails file, which is a big, static, immutable collection of coded, created values representing revocation factors. Each time a new credential is issued, the

authority generates a new revocation factor. This revocation factor is used to calculate the accumulator that encapsulates all valid credentials in an unbroken progression, which indicates the current state of the public revocation registry. The revocation process removes the revocation factor from all future calculations to indicate that the revocation has occurred, effectively indicating that the credential was never issued to the credential holder. This registration method does not generate any identifying information about the credential holder [45].

After reviewing all three methods, Tails Files with Accumulator-based Revocation was determined to be the best form for the project. The rationales for this determination are in line with the privacy guarantees of Indy's proof model, a means for decentralized verifiability without disclosure of credential identifiers, and a proven compatible cryptographic scheme made for the Indy ecosystem. This approach also relies on a revocation list, method identified in the literature review, where the status of each registry is stored. It is stored on the blockchain and is publicly accessible to prevent revoked credentials from performing operations [37].

At a high level, a (cryptographic) accumulator can be thought of as a mathematical object that compresses several inputs down to a single value. Broadly speaking, this is like multiplying several numbers together. If we have inputs *a*, *b*, *c*, and *d*, e could be computed as: $e = a \times b \times c \times d$. The value of *e* is the accumulator in this case. With the accumulator and a subset of the inputs, we can determine if an input was included (or excluded) in the original set. We can do this without revealing all the elements of the accumulator. For instance, knowing the product of *b*, *c*, and *d* (inputs), we can recompute *e*, checking if a was part of the accumulator [45].

For privacy, this approach also uses tails file(s), which are public, non-secret binary files containing a large array of cryptographic factors, which is a massive random number indexed for each credential to make inversion computationally infeasible. The number of entries in a tails file can range from hundreds of thousands to tens of millions, depending on the scale of the credential issuance [46]. Each issued credential is associated with an index in the tails file, as shown in Figure 11.

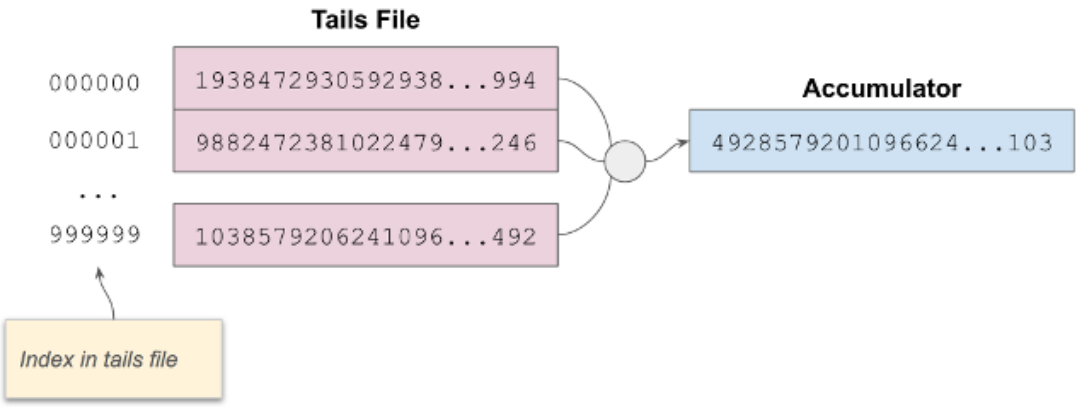

Figure 11 - Index, tails files, and accumulator interaction [46]

Importantly, only unrevoked credentials contribute to the value of the accumulator. Revocation is accomplished by updating the accumulator value to exclude the factor corresponding to the revoked credential's index. To facilitate this, the revocation registry definition includes metadata such as a tails hash (a unique identifier for a tails file) and location (its downloadable URL). These are stored by the application in an in-memory map for quick reference and management of tail file locations. When there is a revocation to process, the system first checks to see if the required tails file already exists locally. If not, it downloads the file from its public URL and stores it locally so that it can be used in cryptographic computations to update the accumulator.

Figure 12 illustrates a sequence diagram that illustrates the interactions between the issuer, the tails files, and the Indy network. It outlines the steps involved in coordinating with the files and updating the accumulator state on the ledger via the Indy network.

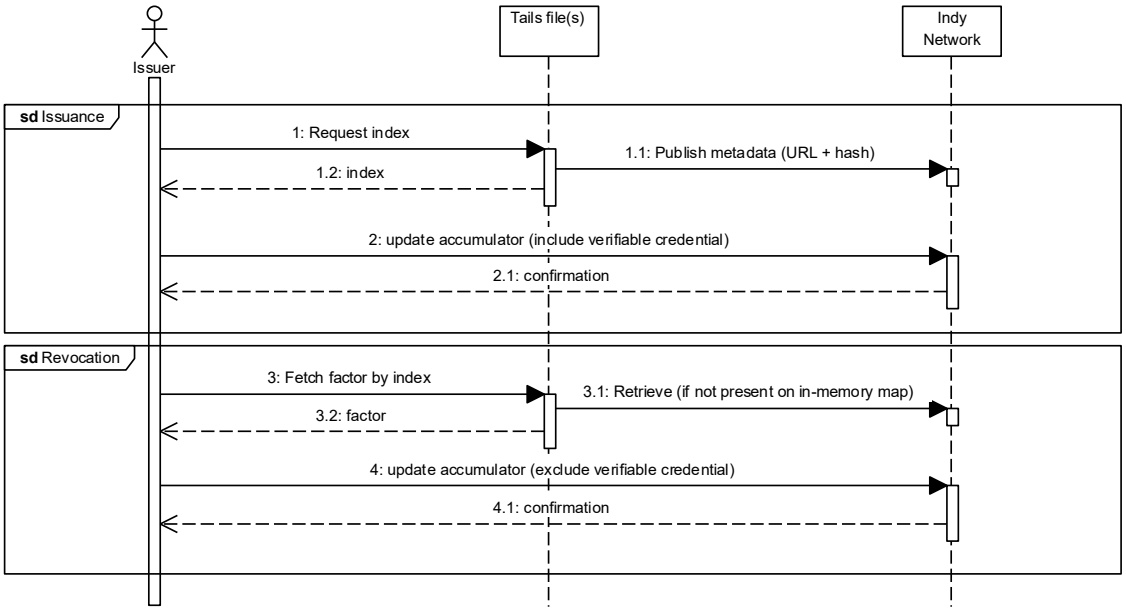

Figure 12 - Interactions between the Issuer, the Tails file(s), and the Indy Network

This development technique promotes both transparency and privacy. It allows anyone to check the revocation status of a credential without accessing private or identifying information. Using cryptographic accumulators and publicly accessible Tails files aligns with the principles of decentralized identity, where privacy is preserved during verification. The public revocation registry, combined with the accumulator's cryptographic properties, protects the confidentiality of credential information

6 Solution Construction

This chapter describes the implementation process of the solution with the inclusion of self-sovereign identity management in the voting domain project. It describes how the integrated solution is implemented, following the procedures described in the Project Selection and Analysis chapter.

6.1 DID Management

Decentralized identifiers are a type of identifier that envelopes a digital identity within a distributed system, and within the scope of this project, can be used to identify the voter, the voting authority, or the voting system. Each of these three roles is instantiated as a Hyperledger Aries Agent, and each of these agents is assigned a corresponding unique DID. The goal is to make every participant in the ecosystem independently verifiable and securely interact with other participants in a trusted, peer-to-peer manner. To accomplish this, a custom class called SSIAgent has been defined to handle the initialization and configuration of an Aries agent to represent each and allow secure interaction using verifiable credentials and decentralized identity protocols.

Code Snippet 4 shows this class's constructor, which accepts by argument a properties object for each agent's name and port, and a modules object for other self-sovereign identity agent configurations, i.e., credential and proof types, connections, ledger support, among others. Each agent is hosted on a local server using the Express framework so that it can be accessed over HTTP. To handle communication in the network, the agent registers the *HttpOutboundTransport* and *HttpInboundTransport* classes. The outbound transport allows the agent to start communication with other agents, and the inbound allows the agent to listen and accept messages on a certain port. This extensible and modular configuration enables the deployment of a secure digital voting platform where each party is an independent and auditable entity in the overall decentralized system.

```
constructor({ modules, properties }: { modules: AgentModules; properties:
Properties }) {
 this.agent = new Agent({
 config: {
label: properties.name,
walletConfig: { id: properties.name, key: properties.name },
endpoints: ['http://localhost:' + properties.port],
},
dependencies: agentDependencies,
modules,
 });
 this.express = express();
 this.properties = properties;
 this.config = this.agent.config;
 this.agent.registerOutboundTransport(new HttpOutboundTransport());
 this.agent.registerInboundTransport(new HttpInboundTransport({ app:
this.express, port: properties.port }));
}
```

Code 4 - SSIAgent class constructor method

With interoperability in mind, the solution supports both OpenID for Verifiable Credentials (OpenID4VC), which enables the issuance and presentation of verifiable credentials based on OAuth 2.0 using the Self-Issued OpenID Provider (SIOPv2) [47], and verifiable credentials issued and registered on a Hyperledger Indy [22] distributed ledger. Both approaches leverage the Hyperledger Aries specifications for secure protocol communication and agent-based identity management.

In terms of DID management, these two approaches require a different method of identifying each agent. The OpenID4VC product requires a generic *did:key* method for its specifications, while Hyperledger Indy registration requires a more complex identifier that also includes the Indy network namespace in use. Figures 13 and 14 show an example of the DID specification for OpenID4VC and Hyperledger Indy usage, respectively. These formats follow the structure defined by the W3C for Decentralized Identifiers, as they start with the DID prefix, followed by a method specification (key or indy), and include method-specific parameters.

Figure 13 - DID structure used in OpenID4VC operations

Figure 14 - DID structure used in Hyperledger Indy operations

Code Snippet 5 shows the methods created to start each of these agents and then set the DID for both specifications. Both methods take a secret private key as an argument, with the second *startWithImportDidIndy* method requiring prior registration on the distributed Indy network used. In this case, since the main objective is to explore self-sovereign identity management, the open BCovrin test Indy network was used [43], which is a public test network (or testnet) based on Hyperledger Indy.

```
  public async startOpenID4VC(secretPrivateKey: string): Promise<void> {
    await this.agent.initialize();
    const didCreateResult = await
this.agent.dids.create<KeyDidCreateOptions>({
      method: 'key',
      options: { keyType: KeyType.Ed25519 },
      secret: { privateKey: TypedArrayEncoder.fromString(secretPrivateKey) },
    });
    this.didKey = DidKey.fromDid((this.did = didCreateResult.didState.did as
string));
    this.kid = `${this.did}#${this.didKey.key.fingerprint}`;
    this.verificationMethod =
didCreateResult.didState.didDocument?.dereferenceKey(this.kid,
['authentication'])!;
  }

  public async startWithImportDidIndy(secretPrivateKey: string): Promise<void>
{
    await this.agent.initialize();
    const unqualifiedIndyDid = 'SDAYCoZ7gZGGoRzBdunAAr';
    const indyDid =
`did:indy:${indyNetworkConfig.indyNamespace}:${unqualifiedIndyDid}`;
    const did = indyDid;
    await this.agent.dids.import({
      did,
      overwrite: true,
      privateKeys: [
        {
          keyType: KeyType.Ed25519,
          privateKey: TypedArrayEncoder.fromString(secretPrivateKey),
        },
      ],
    });
    this.anonCredsIssuerId = did;
  }
```

Code 5 - Start methods used for each specification

To address security issues associated with managing DIDs, particularly impersonation risks, this project provides means to secure each interaction and identity. By using the Hyperledger Aries framework and initiating all agents via the *SSIAgent* class, the system ensures that only authentic individuals can register their DIDs. Every agent is associated with a confidential private key that is used to generate or import its DID, depending on whether the agent uses the *did: key* method for OpenID4VC or the *did:indy* method for Hyperledger Indy. These are cryptographic stewards that prevent unauthorized entities from impersonating an existing

participant. In addition, the modular agent configuration allows for the integration of logging and auditing modules capable of monitoring DID-related events with tamper-proof traceability.

6.2 UC01 - Digital Credential Issuance and Revocation

In this work, credential issuance and revocation can be described as the process in which the voter receives a verifiable credential issued by the voting authority, which can later revoke that credential, rendering the voter ineligible to vote. The voter acts as the holder, and the election authority acts as the issuer in this SSI environment.

For this purpose, an Issuer class was created to handle all the issuance and revocation methods, including the previous and intermediate steps of these two flows. For this, the credential that allows voting is issued using the Hyperledger Indy, which has a purpose-built blockchain and where revocation is possible. As stated for UC04, the credential that validates that the vote has been cast was issued using OpenID4VC, which is a standard for credential issuing/presentation using OpenID Connect flows, and no revocation is possible since once the vote is confirmed, that action cannot be changed or updated. This credential can later be used to integrate verifiable credentials into existing identity systems such as OAuth2/OpenID Connect.

Before issuing a credential, it was necessary to define a credential schema, which includes the credential type, attributes, name, and version of the credential. Code Snippet 6 shows the registration of the developed credential schema in the network to be used. The credential type is the Anoncreds W3C VC-compliant structure [48] used in Hyperledger Indy, and the attributes identified for voting are 'name', 'code' (id number), and 'date' (birthdate), as these are the normal requirements needed before voting.

```
  public async schemaResult(): Promise<RegisterSchemaReturn> {
    const schemaResult = await this.agent.modules.anoncreds.registerSchema({
      schema: {
        attrNames: ['name', 'code', 'date'],
        issuerId: this.anonCredsIssuerId as string,
        name: 'Voting Credential ' + Math.random().toString(),
        version: '1.0.0',
      },
      options: {},
    });
    if (schemaResult.schemaState.state === 'failed') {
      throw new Error(`Error creating schema:
${schemaResult.schemaState.reason}`);
    }
    return schemaResult;
  }
```

Code 6 - Credential schema registration

The credential definition must then be registered using the schema registration ID. The *credentialDefinitionResult* method, shown in Code Snippet 7, registers a credential definition using the Hyperledger Aries agent modules initialized by the *SSIAgent* class. The result is a template used by an issuer to offer credentials based on the previously registered scheme. It

includes the issuer identifier (DID) and has revocation support enabled in the options (*supportRevocation: true*), meaning that credentials issued from this definition can be revoked later.

```
  public async credentialDefinitionResult(registerSchemaReturn:
RegisterSchemaReturn): Promise<RegisterCredentialDefinitionReturn> {
    const credentialDefinitionResult = await
this.agent.modules.anoncreds.registerCredentialDefinition({
      credentialDefinition: {
        tag: 'default',
        issuerId: this.anonCredsIssuerId as string,
        schemaId: registerSchemaReturn.schemaState.schemaId as string,
      },
      options: {
        supportRevocation: true,
      },
    });
    if (credentialDefinitionResult.credentialDefinitionState.state ===
'failed') {
      throw new Error(`Error creating credential definition:
${credentialDefinitionResult.credentialDefinitionState.reason}`);
    }
    return credentialDefinitionResult;
  }
```

Code 7 - Credential definition registration

Following schema registration and credential definition registration, the next step in fully enabling revocation was to implement a revocation registry on the network. This involves creating a registry that can track the revocation status of credentials issued in accordance with a particular credential definition. Code Snippet 8 shows that the *revocationRegistryResult* method creates a revocation registry using the credential definition ID. It includes the issuer ID, a random tag, and the maximum number of credentials it can manage. For development purposes, a maximum of 10 credentials has been set, but this could go to millions of entries if necessary.

```
  public async revocationRegistryResult(credDefId: string) {
    const result = await
this.agent.modules.anoncreds.registerRevocationRegistryDefinition({
      revocationRegistryDefinition: {
        issuerId: this.anonCredsIssuerId as string,
        credentialDefinitionId: credDefId,
        tag: Math.random().toString(),
        maximumCredentialNumber: 10,
      },
      options: {},
    });
    if (result.revocationRegistryDefinitionState.state === 'failed') {
      throw new Error(`Error creating revocation registry:
${result.revocationRegistryDefinitionState.reason}`);
    }
    return result;
  }
```

Code 8 - Revocation definition registration

To keep track of which credentials are active or revoked, a revocation status list has also been published. The *registerRevocationStatusList* method handles this by sending the list to the network and checking for successful completion. All of this is part of the credential issuance method shown in Code Snippet 9. It takes as a parameter a *CredentialAttributes* object, so that it is variable for each credential offering, containing a connection ID corresponding to the specific owner (voter) for whom this credential is being issued. The holder can then accept and present the credential before executing their voting option.

```
public async issueCredentialIndy(attributes: CredentialAttributes):
Promise<CredentialExchangeRecord> {
    const schemaResult = await this.schemaResult();
    const credentialDefinition = await
this.credentialDefinitionResult(schemaResult);
    const revocationRegistry = await
this.revocationRegistryResult(credentialDefinition.credentialDefinitionState.c
redentialDefinitionId!);
    await this.registerRevocationStatusList({
      revocationRegistryDefinitionId:
revocationRegistry.revocationRegistryDefinitionState.revocationRegistryDefinit
ionId as string,
      issuerId: this.anonCredsIssuerId!,
    });
    const indyCredentialExchangeRecord = await
this.agent.credentials.offerCredential({
      protocolVersion: 'v2',
      connectionId: this.connectionId as string,
      credentialFormats: {
        anoncreds: {
          credentialDefinitionId:
credentialDefinition.credentialDefinitionState.credentialDefinitionId as
string,
          revocationRegistryDefinitionId:
revocationRegistry.revocationRegistryDefinitionState.revocationRegistryDefinit
ionId!,
          revocationRegistryIndex: revocationRegistryIndex,
          attributes: [
            { name: 'name', value: attributes.name },
            { name: 'code', value: attributes.code },
            { name: 'date', value: attributes.date },
          ],
        },
      },
    });
    return indyCredentialExchangeRecord;
  }
```

Code 9 - Full issue credential method with revocation support

These options are accessible through a server, so the holder has the possibility to obtain all the credential offers that he has assigned by filtering with his connection ID to the issuer. After obtaining these offers, each one is assigned to a specific identifier with which the holder executes the procedure for accepting the credential using a method that takes the credential offer ID as an argument and returns the credential as a *CredentialExchangeRecord* object, as shown in Code Snippet 10.

```
        {
            "_tags": {
                "connectionId": "c28d1228-ec8f-4535-85f3-72c70f91732f",
                "credentialIds": [
                    "cc1ddd71-e6ac-4fc3-8180-1c670b622880"
                ],
                "role": "holder",
                "state": "done",
                "threadId": "73db3644-321a-42ed-ba06-b85b7c566d0a"
            },
            "metadata": {
                "_anoncreds/credentialRequest": {
                    "link_secret_blinding_data": {
                        "v_prime":
"4007893400024935814295621594068756751312756670137100503119741098016379983000939245878718555707988415045654685829488915352375397790095299278724511415686807486007660608370255864577146474306114121879462873318045603184931203702903606546341025056471635569282756024110220658777147342204897836684185298405511098870081312637478426498728556315809929684879440377884138745250810080126564746481864119741653062900532808634759184299556793528254025656237560540115421032190151223510941455370619185126782296394824873713429048107482418782394710239860160078979827220859281373571079260311365261251014760697605069094601744128978916961163902307919199848898026939",
                        "vr_prime":
"17A8B4C0884919B16824264122AB1FF4B7A5EA91537826377E68D8C1992ADEAA"
                    },
                    "nonce": "77271592827059823451575",
                    "link_secret_name": "a0e0e03b-c286-4092-ba65-3df7135c496b"
                },
                "_anoncreds/credential": {
                    "credentialDefinitionId":
"did:indy:bcovrin:test:SDAYCoZ7gZGGoRzBdunAAr/anoncreds/v0/CLAIM_DEF/2784999/default",
                    "schemaId":
"did:indy:bcovrin:test:SDAYCoZ7gZGGoRzBdunAAr/anoncreds/v0/SCHEMA/Voting Credential 0.3912408620128984/1.0.0"
                }
            },
            "credentials": [
                { "credentialRecordType": "w3c", "credentialRecordId":
"cc1ddd71-e6ac-4fc3-8180-1c670b622880" }
            ],
            "id": "d4d93bb0-4f4e-46b8-a5c9-40f0509b0f01",
            "createdAt": "2025-04-24T13:19:14.029Z",
            "state": "done",
            "role": "holder",
            "connectionId": "c28d1228-ec8f-4535-85f3-72c70f91732f",
            "threadId": "73db3644-321a-42ed-ba06-b85b7c566d0a",
            "protocolVersion": "v2",
            "updatedAt": "2025-04-24T13:21:44.387Z",
            "credentialAttributes": [
                { "mime-type": "text/plain", "name": "name", "value": "Jane
Doe" },
                { "mime-type": "text/plain", "name": "code", "value":
"85746217" },
                { "mime-type": "text/plain", "name": "date", "value":
"01/05/2004" }
            ]
        }
```


Code 10 - *CredentialExchangeRecord* object example

The *revokeCredentialIndy* method is responsible for revoking a credential previously issued by the voting authority using the protocol of this Indy-based identity system. The authority that previously issued the voting credential holds the necessary inputs, specifically the *revocationRegistryId* (identifying the registry of the credential) and the *credentialRevocationId* (index representing the credential within the revocation list). This method (Code Snippet 11) updates the list of revocation statuses by marking the specified index as revoked. On successful execution, it returns a standard success response, and on failure (e.g., network issues, invalid inputs), the function logs it and returns a response with an appropriate status code.

```
  async revokeCredentialIndy(credentialRevocationId:
string,revocationRegistryId: string):
  Promise<ServiceResponse<RegisterRevocationStatusListReturn | null>> {
    try {
      const response = await
issuerIndy.agent.modules.anoncreds.updateRevocationStatusList({
        revocationStatusList: {
          revocationRegistryDefinitionId: revocationRegistryId,
          revokedCredentialIndexes: [Number(credentialRevocationId)],
        },
        options: {},
      });
      return ServiceResponse.success('Credential revoked.', response,
StatusCodes.OK);
    } catch (ex) {
      const errorMessage = `Error revoking credential: ${(ex as
Error).message}`;
      logger.error(errorMessage);
      return ServiceResponse.failure(errorMessage, null,
StatusCodes.INTERNAL_SERVER_ERROR);
    }
  }
```

Code 11 - Method responsible for credential revocation

Each credential issued by an authority is associated with a specific factor in a tails file, which is a large, immutable set of randomly generated numbers. These factors are used in a cryptographic accumulator to compute a public revocation registry value. Only valid credentials contribute their factors to this accumulator. When a credential is revoked, its associated factor is deducted from subsequent accumulator calculations, effectively deleting it without revealing the credential's identity [45].

To facilitate this, the revocation registry definition includes metadata such as a tails hash (a unique identifier for a tails file) and location (its downloadable URL). These are stored by the application in an in-memory map for quick reference and management of tails file locations. When there is a revocation to process, the system first checks to see if the required tails file already exists locally. If not, it downloads the file from its public URL and stores it locally so that it can be used in cryptographic computations to update the accumulator. This technique finds a trade-off between transparency and privacy because anyone can query revocation status without accessing private information.

6.3 UC02 - Voting Authentication

The authentication method uses Self-Sovereign Identity (SSI) principles to authenticate the identity of the voter. The voter has a credential offered by the voting authority. Before voting is conducted, the credential is validated to verify that the voter is allowed to vote. In this use case, the voting system serves as a verifier; it validates the credential while ensuring privacy and maintaining the integrity of the data.

To begin the process, when a voter is ready to vote, they select one of their credentials in the web application interface and indicate they are ready to submit their choice. The application then performs system validations to validate the request integrity. If the system validation is accepted, the normal voting process continues. As part of the next application step, the application checks for credential revocation status by sending a request to the server, which sends requests to external verification agents to validate the revocation status of the credential. The server queries the verifier component for the most up-to-date revocation data in the Indy Network, where credentials and revocation records are stored. The server then returns the revocation information.

The server checks the list for the voter's credentials and checks that the credentials are still valid. A first credential index of zero indicates that it is valid. If so, the server sends a proof request to the voter's device, which acts as the holder. The device responds to the proof request with verifiable proof of having a valid, unrevoked credential. After the server verifies the proof, the server confirms the authentication of the voter and sends a notification to the web application. The voter can then access the voting interface to vote.

This process is depicted as a UML sequence diagram in Appendix B.

6.4 UC03 - Vote Registration

After completing the voting authentication process, the voter is granted access to submit their vote. Due to constraints on the project and limited scope, this prototype implementation allows just one voter. Once a voter has been authenticated and submitted their vote, the system knows that they have finished voting. This is accomplished by marking the voter's identity (which is represented anonymously), and subsequent attempts to vote are checked against the previously recorded voters.

The voter's anonymity is preserved through the design of the authentication process. In this prototype, once the authentication credential is validated, the voter is allowed to vote and is reduced to an anonymous session secret, which is a randomly generated secret stored in a browser cookie. This session secret is the anonymization layer implemented in the vote-handling component, so the system doesn't log personally identifiable information. Yet, it can identify whether a vote has been cast. Code Snippet 12 represents the vote submission logic,

integrating all the above constraints and behaviours preserved from the initially selected project.

```
@app.route("/", methods=['POST','GET'])
def vote():
    voter_id = request.cookies.get('voter_id')
    If not voter_id:
        voter_id = hex(random.getrandbits(64))[2:-1]

    vote = None
    vote_submitted = False

    if request.method == 'POST':
        redis = get_redis()
        vote = request.form['vote']
        data = json.dumps({'voter_id': voter_id, 'vote': vote})
        redis.rpush('votes', data)
        vote_submitted = True

    resp = make_response(render_template(
        'index.html',
        option_a=option_a,
        option_b=option_b,
        hostname=hostname,
        vote=vote,
        vote_submitted=vote_submitted
    ))
    resp.set_cookie('voter_id', voter_id)
    return resp
```

Code 12 - Vote submission logic

There are ways to improve this logic by implementing stronger vote processes that are not reliant on client-based cookies. For example, server-side session storage or token management combined with the use of JWT verifiable credential tokens would offer stronger protection against cookie attack (renaming), cookie clearing, etc. However, the voting authentication approach is a legitimate security method for this prototype.

6.5 UC04 - Voting Participation Verification

The voter can request and receive a secure confirmation of their voting participation, as shown in the voting participation verification sequence diagram in Appendix C. To securely issue and verify digital credentials based on decentralized identity principles, the system makes use of OpenID for Verifiable Credentials, a protocol. The JWT format, which is based on JSON and contains linked data elements, is used to structure the digital credential in this instance. This case aims to demonstrate Hyperledger Aries' interoperability capabilities by demonstrating that the prototype can participate in the issuance, holding, and verification using both OpenID4VCs (voting verification) and W3C verifiable credentials (voting authentication).

When the voter requests confirmation of their voting activity through the web application, the process starts. This causes the web application to start using OpenID4VC to create a credential issuance process. The server then gets in touch with the issuance component and requests the

creation of a Verifiable Credential, which verifies that the voter voted. A structured message alerting the system that a credential is prepared for issuance is called a credential offer, and it is created by the issuer. This offer returns to the web application via the server. Following receipt of the offer, the web application starts a resolution procedure to verify the offer's structure and legitimacy. This involves multiple steps, including interactions with the issuer to analyze and finalize the offer.

After the offer is resolved, the credential is formally accepted, and the process continues with the voter, acting as the holder, receiving the credential. At this point, the credential has been successfully issued to the voter, and it includes attributes and a JWT token, as shown in Code Snippet 13.


```
"credential":"eyJ0eXAiOiJKV1QiLCJhbGciOiJFZERTQSIsImtpZCI6ImRpZDprZXk6ejZNa3RD
QndxQnk5b0dNbVQ2bzQyVkxkSlFiVlI1S3ZzY3ptOXZhN1pjQnkyNndzI3o2TWt0Q0J3cUJ5OW9HTW
1UNm80MlZMZEpRYlZSNUt2c2N6bTl2YTdaY0J5MjZ3cyJ9.eyJ2YyI6eyJAY29udGV4dCI6WyJodHR
wczovL3d3dy53My5vcmcvMjAxOC9jcmVkZW50aWFscy92MSJdLCJ0eXBlIjpbIlZlcmlmaWFibGVDc
mVkZW50aWFsIiwiVm90aW5nQ3JlZGVudGlhbCJdLCJjcmVkZW50aWFsU3ViamVjdCI6e319LCJpc3M
iOiJkaWQ6a2V5Ono2TWt0Q0J3cUJ5OW9HTW1UNm80MlZMZEpRYlZSNUt2c2N6bTl2YTdaY0J5MjZ3c
yIsInN1YiI6ImRpZDprZXk6ejZNa2lZa2l1UHJldjM4bU5ROFA4VVMxZTVjQ1dlUHJiZUJuV2htbzV
YakZYdjcyIiwibmJmIjoxNzQ4NDU0NTYyfQ.o8RkqqnO5hJEY9E38aHKpVNozcvC5S7xU2AMIfGyL5
YHrofvyT7JUfLciqkr6lGnzKPi1BbxS1LbmvK31o0zAg"
```


Code 13 - JWT Verifiable Credential token example

Only the creation of an OpenID verified credential with the token property is supported by this prototype. Though this is outside the scope of this project's accomplishments, it might be linked to OAuth or OIDC systems to safely access protected data, such as accessing voting data over secure endpoints. Since the Issuer takes on the role of a provider for this particular use case, there is no communication with the Test Indy Network.

When users attempt to access secure (HTTPS) endpoints using tokens, the OpenID Connect (OIDC) authentication protocol assists in confirming their identity. OIDC is based on the OAuth protocol and was developed by the OpenID Foundation, which comprises businesses such as Microsoft and Google. OIDC provides an additional layer of security by adding authentication (who the user is), while OAuth manages authorization (what a user can do). Users may be made simpler, and their identities can be managed across systems with this method [49].

6.6 UC05 - Voting Results Audit

A logging and credential verification mechanism has been implemented in the prototype to help ensure the integrity of the voting system and ensure that all votes are counted correctly and without irregularities. The logging mechanism supports a publish-subscribe model, allowing components to listen for and react to specific log messages, thereby making it easier to monitor and debug the voting system. A custom class is the central part of the mechanism and provides historical logging services as shown in Code Snippet 14. The service supports the complete tracking of system events related to auditable voting, including the process of issuing voting

credentials and their subsequent verification. The service replays the historical logs to any present listeners when they are initialized, providing a stable and reliable source of state knowledge throughout the application.

```
type LogListener = (log: string) => void;
class LogService {
  private listeners: LogListener[] = [];
  private readonly STORAGE_KEY = 'logHistory';
  constructor() {
    const storedLogs = JSON.parse(sessionStorage.getItem(this.STORAGE_KEY) ||
'[]');
    storedLogs.forEach((log: string) => {
      this.listeners.forEach((listener) => listener(log));
    });
  }
  log(message: string) {
    const logEntry = `[${new Date().toLocaleString()}] ${message}`;
    console.log(logEntry);
    const currentLogs = JSON.parse(sessionStorage.getItem(this.STORAGE_KEY) ||
'[]');
    const updatedLogs = [logEntry, ...currentLogs].slice(0, 100);
    sessionStorage.setItem(this.STORAGE_KEY, JSON.stringify(updatedLogs));
    this.listeners.forEach((listener) => listener(logEntry));
  }
  onLog(listener: LogListener) {
    this.listeners.push(listener);
    const existingLogs = JSON.parse(sessionStorage.getItem(this.STORAGE_KEY)
|| '[]');
    existingLogs.forEach(listener);
    return () => {
      this.listeners = this.listeners.filter((l) => l !== listener);
    };
  }
}
export default new LogService();
```

Code 14 - Log Service class

Plus, to local application logging, the system logs credential events such as issuance, verification, and presentation on the Indy Test Network. Recording these important identity events in a blockchain adds an extra level of integrity and trust in the system. The credential and status can be independently verified on the ledger, so no vote can ever be cast on an invalid or forged credential. Figure 15 shows two transaction records issued under the Indy network. They are both components of the infrastructure for issuing and revoking credentials and describe the action of issuing and managing a verifiable credential schema and its related revocation mechanisms.

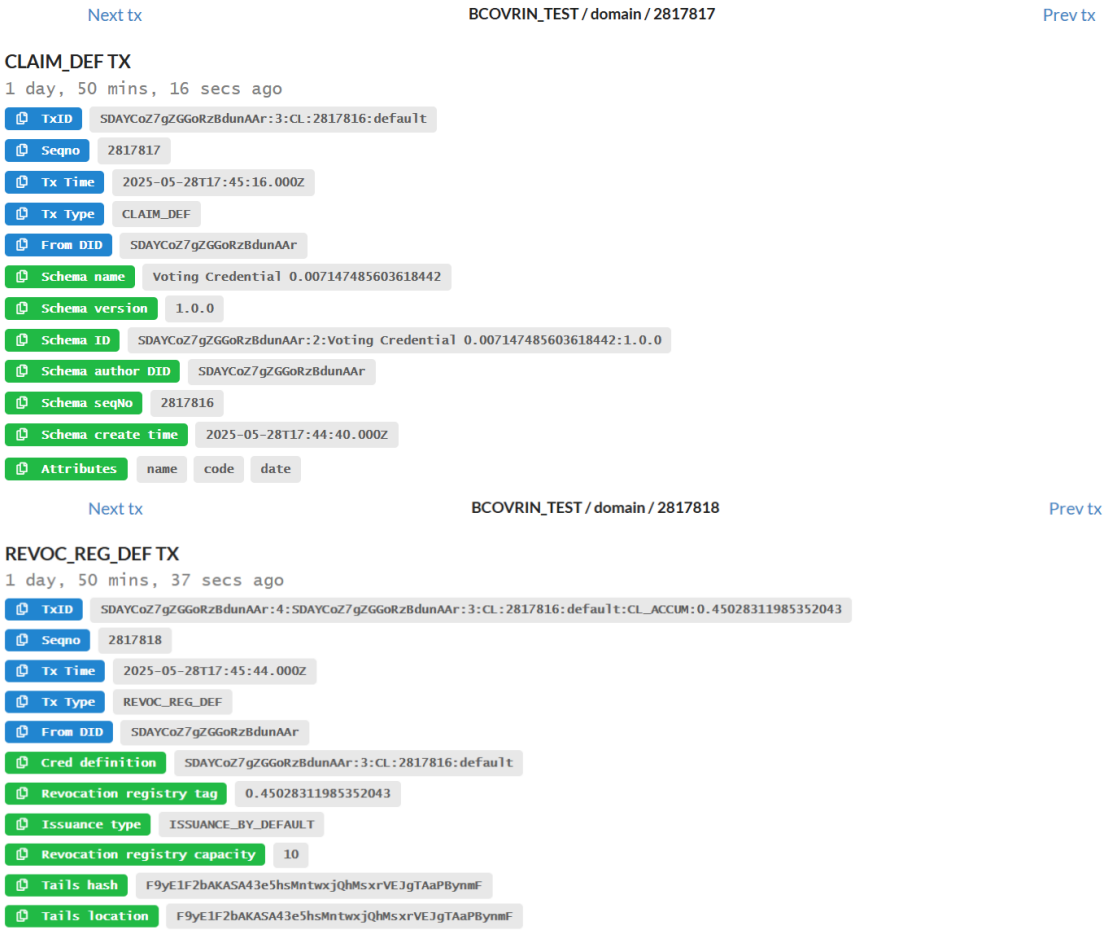

Figure 15 - CLAIM_DEF TX and REVOC_REG_DEF TX records on BCovrin Test Indy Network

The CLAIM_DEF TX is related to the registration of a claim definition, which specifies how a particular type of verifiable credential is issued. In this example, the credential is titled "Voting Credential 0.007..." and applies to schema version 1.0.0. The schema represents the structure of the credential, including its attributes (name, code, and date). It allows issuers and verifiers to operate with a shared and immutable voting credential format. The REVOC_REG_DEF TX defines how credentials issued under the claim definition can be revoked. This includes setting parameters such as the revocation registry capacity, the issuance type (in this case, ISSUANCE_BY_DEFAULT, meaning credentials are valid unless revoked), and metadata related to tails files, which are used in cryptographic proofs of revocation status.

The prototype's vote verification infrastructure is produced by combining two methods: the Test Indy Network and the application-level log system. The ledger ensures the authenticity and verification of identity credentials, and the logs provide an auditable record of voting-related actions within the system.

7 Results

This chapter shows the results and clarifies the process of assessment and experiments conducted. Upon more detailed exploration of self-sovereign identity management and integration challenges, this study's central focal points were attained: security, performance, and interoperability. These aspects present key challenges to SSI integration, requiring seamless interoperability among varied implementations while ensuring efficient processes of verification and issuance of verifiable credentials. The threats to the security of DID management and the revocation restrictions on verifiable credentials were mentioned in the Implementation chapter, where the processes that were carried out with these points in mind were shown and explained throughout the solution development. The experimentation process began during the project selection stage, when several exploratory tests were carried out to assess its viability for use.

The objective of this chapter was to demonstrate the operations executed for achieving interoperability among different self-sovereign identity technologies, to maximize the adoption and usability of decentralized identity systems. Secondly, enhancing credential verification performance aims to make the system more responsive and user-friendly. Based on these guidelines, experiments were designed to verify cross-platform mechanisms and find out the efficiency of credential verification and issuance processes.

The evaluation method chosen for this work was the Goal Question Metric approach. The project was compatible with a list of requirements, such as being open-sourced and being in a domain that is known to be a practical demonstration of self-sovereign identity integration. This undertaking provides facts to aid individuals to help them in self-sovereign identity for various project environments and integrating as well as measuring solutions with a focus on attaining interoperability between various SSI methods, as well as issuance and verification benchmarking.

7.1 Assessment

This project uses the Goal Question Metric (GQM) approach, which states that successful measurement is based on an organization's initial specification of well-articulated goals for itself and its projects. These goals are subsequently monitored against fitting data that operationally define them [28]. This approach provides a framework for interpreting data in the light of stated objectives. It follows a hierarchical structure with an objective (listing the purpose of the measurement, the object to be measured, the problem to be measured, and the viewpoint from which the measure is taken). The objective is then limited to a set of questions, and each question is limited to metrics [51].

The result of applying the GQM approach is the specification of a measurement system for a given set of problems and a set of rules for the interpretation of the data. The resulting measurement model (Figure 16) has three levels [28]:

- Conceptual level: A goal is defined for an object, for numerous reasons, about numerous models of quality, from various viewpoints. Measurement focuses on three areas: products, processes, and resources. Products are the outcomes of a project; processes include tasks such as design, testing, and interviewing; and resources include people, tools, and materials.

- Operational level: A set of questions is used to demarcate the way the evaluation/achievement of the said goal is conducted based on a characterizing model. Questions try to characterize the object of measurement (product, process, resource) against a selected quality issue and to determine its quality from the selected viewpoint.

- Quantitative level: Applied to questions with attached data so that they can be answered with quantifiable data. Data can be objective or subjective. Objective data depends only on the object being measured and not on the point of view from which it is measured. Subjective data depends on both the object being measured and the perspective from which it is measured.

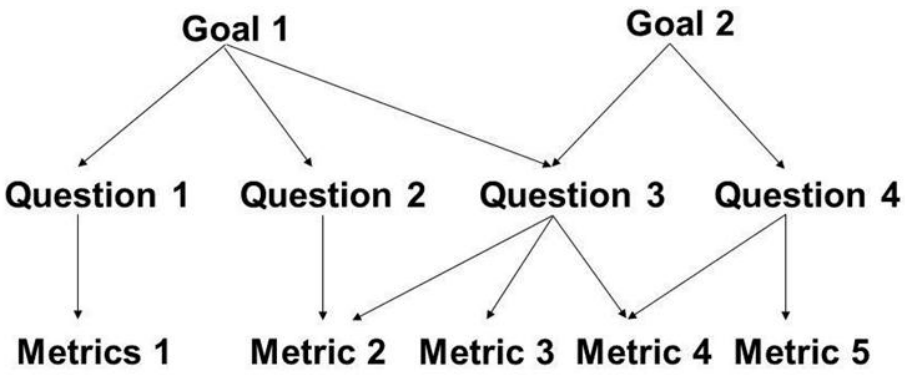

Figure 16 - Goal Question Metric structure [51]

The assessment procedures in this work aim to evaluate the solution SSI components in terms of interoperability and performance. These issues have been identified in Section 1.1 as part of the challenges associated with integrating SSI into existing systems. To achieve a solution that ensures semantic interoperability between different SSI technologies, as raised in Section 2.3, the main object of interest in this solution is the verifiable credentials applied to the voting system. This solution uses Hyperledger Aries, so the aim is to implement the best practices between the presented SSI agents, so that when other technologies adopt or replace them, the solution works while maintaining interoperability. In terms of performance, a longer average response time for credential verification compared to credential issuance has been identified as a process that affects user experience and delays important, secure, critical processes. The objective was to analyze the performance through a set of tests for these two processes in the solution, obtained through the issuer, holder, and verifier components, where a credential is issued by the issuer for the holder, and then this holder presents this credential to a verifier for the verification processes.

The goal is that the obtained results of this assessment process provide lessons to help individuals to assist them in self-sovereign identity for various project environments and integration, as well as measuring solutions with a focus on achieving interoperability between various SSI methods, as well as issuance and verification benchmarking. Figure 17 shows the structure of the implemented GQM, and Table 11 explains each of the objectives, questions, and metrics of the chosen approach.

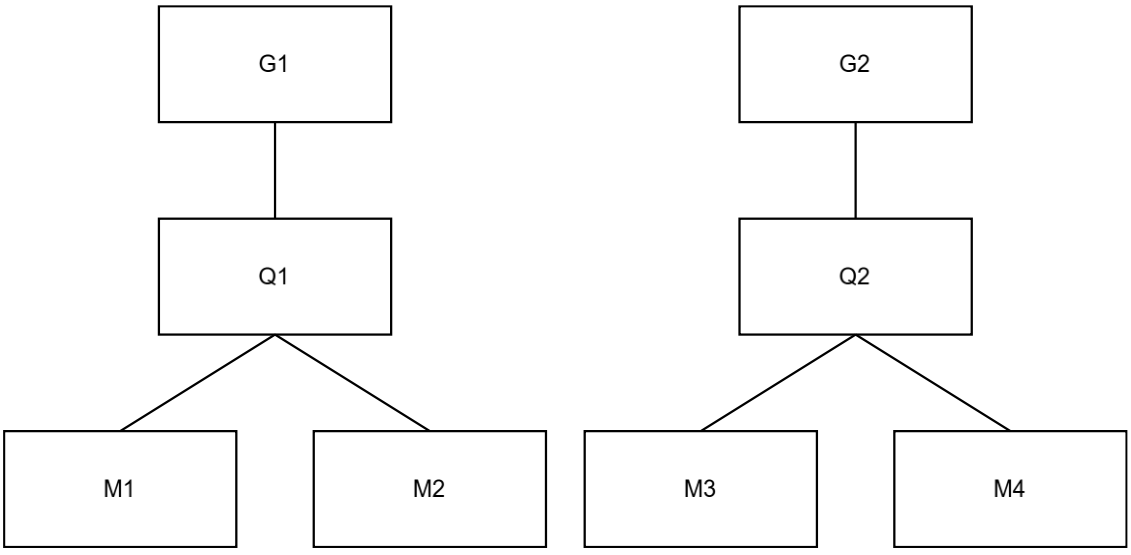

Figure 17 - Goal Question Metric approach implemented

Quality Attribute	Goal	Question	Metric
Interoperability	G1 - Assess the interoperability of verifiable credentials in the solution across different SSI technologies.	Q1 - What verifiable credential formats does the prototype support for interoperability?	M1 - Successful Issuance of W3C and JWT VCs.
		Q2 - How do voting confirmation VCs from the prototype adhere to standard structures for interoperability?	M2 - Successful JWT verifiable credential decoding.
Performance	G2 - Evaluate the performance of credential issuance and verification in SSI integration into existing systems.	Q3 - How efficient is the credential verification process?	M3 - Verification time must be at least 10% less than issuance time.
		Q4 - What is the impact of credential verification performance on user experience?	M4 - A minimum of 90% of verification attempts are successful.

Table 11 - Goal, Questions, and Metrics for each Quality Attribute

Execution metrics are collected manually and automatically to quantify the interoperability of the solution with an existing authentication system and to quantify the performance of significant processes in SSI. Testing guarantees (M1) W3C Verifiable Credentials and JWT format [13] compliance for standardization and interoperability with heterogeneous identity systems and protocols. It also certifies (M2) efficient JWT credential issuance as per the OpenID4VCI spec and protocol [47] to enable simple authentication and authorization across various platforms. It also tests performance through average response time comparison of credential issuance and verification, identifying possible bottlenecks and system optimization (M3 & M4).

The metric collection's end is to find out whether the solution capability for achieving interoperability and performance objectives. Compliance testing, credential exchange, and system response specify how a better job can be done at it, and check whether the solution capability is on par with SSI levels and works best with various authentication conditions.

7.2 Data Collection

For each quality attribute, Performance and Interoperability, a series of processes and experiments were performed to evaluate system behaviour under different operations. They were designed to evaluate specific challenges identified during the initial phase of the project, to understand how the system meets them. With these evaluations, insights were gained into performance limitations and the degree of interoperability achieved, providing a clearer picture of the system's strengths and areas for improvement.

7.2.1 Interoperability

The prototype can issue Verifiable Credentials in both the W3C Verifiable Credential data model and the JWT-based format. When it comes to voting authentication, the voting authority is the issuer and offers a credential to the voter, which is anchored on the Test Indy Network and used by the voter for authentication before voting. The issuer also runs a self-hosted OpenID for Verifiable Credential Issuance endpoint, and after the voter submits their vote, they can request a second credential through this OpenID4VCI flow, which issues a JWT-based credential, which is then issued to the voter as a secure token. Although this JWT could later be integrated into an existing authorization or identity management system, such integration falls outside the scope of the current prototype.

- Q1 - What verifiable credential formats does the prototype support for interoperability?

The prototype demonstrates interoperability by successfully issuing Verifiable Credentials in two widely adopted formats: the W3C Verifiable Credential data model and the JWT-based format. This dual implementation illustrates the system's flexibility in supporting different self-sovereign identity ecosystems. Verifiable credentials can function effectively across different implementations, each with distinct protocols and data models. Figure 18 illustrates a credential issued after a vote is submitted, attesting that the vote was successfully submitted, and Figure 19 presents a credential containing voter information, used prior to voting, where verification is performed before the vote is cast. These illustrate the structural differences between the two formats used for this prototype.


```
{
  "_tags": {
    "expandedTypes": []
  },
  "metadata": {},
  "id": "f4054107-f547-42a5-b665-f42221529d6e",
  "createdAt": "2025-05-31T14:34:12.576Z",
  "credential":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJFZERTQSIsImtpZCI6ImRpZDprZXk6ejZNa3RDQndxQnk5b0dNbVQ2bzQyVkxkS1FiV1I1S3ZzY3ptOXZhN1pjQnkyNndzI3o
2TWt0Q0J3cUJ5OW9HTW1UNm80M1ZMZEpRY1ZSNUt2c2N6bTl2YTdaY0J5MjZ3cyJ9.eyJ2YyI6eyJAY29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC
9jcmVkZW50aWFscy92MSJdLCJ0eXBlIjpbI1ZlcmlmaWFibGVDcmVkZW50aWFsIiwiVm90aW5nQ3J1ZGVudG1hbCJdLCJjcmVkZW50aWFsU3ViamVjdCI6e319LC
Jpc3MiOiJkaWQ6a2V5Ono2TWt0Q0J3cUJ5OW9HTW1UNm80M1ZMZEpRY1ZSNUt2c2N6bTl2YTdaY0J5MjZ3cyIsInN1YiI6ImRpZDprZXk6ejZNa2lZa211UHJ1dj
M4bU5ROFA4VVMxZTVjQ1d1UHJiZUJuV2htbzVYakZYdjcyIiwibmJmIjoxNzQ4NzAyMDM4fQ.Y37H_KIrc0mes7Daq5fCuVbXHXzt_4MZ7jIOXN8Ula-
xAkqpjIarefD7LmU4OqK0O8v2rsW9z1K3HD3Bb3khAQ",
  "updatedAt": "2025-05-31T14:34:12.577Z"
}
```


Figure 18 – Verifiable Credential JWT format object

```
{
  "_tags": {
    "anonCredsCredentialRevocationId": "8",
    "anonCredsRevocationRegistryId": "did:indy:bcovrin:test:SDAYCoZ7gZGGoRzBdunAAr/anoncreds/v0/REV_REG_DEF/2821877/default/0.4097029058782251",
    "connectionId": "1596ef69-0d12-41a2-8556-caf9210fb2ae",
    "credentialIds": [
      "c646dc0f-6934-4ca2-a1c4-abf3f3429054"
    ],
    "role": "holder",
    "state": "done",
    "threadId": "650659c7-a715-43b8-a03a-fe73b9d9c20b"
  },
  "metadata": {
    "_anoncreds/credentialRequest": {
      "link_secret_blinding_data": {
        "v_prime": "154503333798683674536800019156943938958428140190361764134686092886943566115405600006849172764653145613358395437164580940003 5",
        "vr_prime": "1BDE90D80FB589A4430F16EAD004915197F04255526F9285D0944CF4D092D167"
      },
      "nonce": "75160214340399578154938 8",
      "link_secret_name": "d4842d90-a6a9-4008-8500-5ffab5e0ec59"
    },
    "_anoncreds/credential": {
      "credentialDefinitionId": "did:indy:bcovrin:test:SDAYCoZ7gZGGoRzBdunAAr/anoncreds/v0/CLAIM_DEF/2821877/default",
      "schemaId": "did:indy:bcovrin:test:SDAYCoZ7gZGGoRzBdunAAr/anoncreds/v0/SCHEMA/Voting Credential 0.8236820990801503/1.0.0",
      "credentialRevocationId": "8",
      "revocationRegistryId": "did:indy:bcovrin:test:SDAYCoZ7gZGGoRzBdunAAr/anoncreds/v0/REV_REG_DEF/2821877/default/0.4097029058782251"
    }
  },
  "credentials": [
    {
      "credentialRecordType": "w3c",
      "credentialRecordId": "c646dc0f-6934-4ca2-a1c4-abf3f3429054"
    }
  ],
  "id": "168f936a-495b-4f41-8ff8-3258da39777a",
  "createdAt": "2025-05-31T12:30:46.200Z",
  "state": "done",
  "role": "holder",
  "connectionId": "1596ef69-0d12-41a2-8556-caf9210fb2ae",
  "threadId": "650659c7-a715-43b8-a03a-fe73b9d9c20b",
  "protocolVersion": "v2",
  "updatedAt": "2025-05-31T12:35:03.671Z",
  "credentialAttributes": [
    {
      "mime-type": "text/plain",
      "name": "name",
      "value": "George Phillips"
    },
    {
      "mime-type": "text/plain",
      "name": "code",
      "value": "19738246"
    },
    {
      "mime-type": "text/plain",
      "name": "date",
      "value": "01/05/2001"
    }
  ]
}
```


Figure 19 - Verifiable Credential W3C format object

These two verifiable credential formats issued in this prototype - JWT and W3C Verifiable Credentials - successfully answer M1 metric, which required that the prototype could issue these two types. This metric definition started at the start of the project, so there was no guarantee that it would be achieved. The prototype was able to respond to this metric successfully at the end of this work.

- Q2 - How do voting confirmation VCs from the prototype adhere to standard structures for interoperability?

Figure 18 shows a JWT-based credential issued via OpenID for the Verifiable Credential Issuance endpoint, which could be integrated with the OpenID Connect authentication protocol to confirm their identity and allow requests [49]. To validate the content of the JWT issued, a JWT decoder was used to validate its content, as shown in Figure 20. After exploratory testing with other tools, Online JWT Decoder [50] was chosen because it makes token decoding on the client part (within the browser), without making any POST or GET requests that pass the JWT to external servers.

A signed JWT contains these 3 parts [50]:

- Header: Contains metadata, including information about the key used to sign the JWT.
- Body: JSON object with a reliable payload; the keys of this JSON object are usually named as claims.
- Signature: Built by performing a cryptographic operation over the header and the body.

Figure 20 - Verifiable Credential JWT content decoded

This credential is considered valid for the following reasons: It follows the W3C Verifiable Credentials data model because it includes essential fields such as context and type. The JWT is digitally signed using a private key linked to a decentralized identifier, with the corresponding public key referenced via the key ID, to allow recipients to verify both the authenticity and integrity of the verifiable credential. Also, the utilization of the DID:KEY method to identify both the issuer and the subject aligns with decentralized, self-sovereign identity principles. The adoption of JWT allows compatibility with existing identity systems and protocols, facilitating seamless integration and verification between different platforms [52] [49].

- G1 - Assess the interoperability of verifiable credentials in the solution across different SSI technologies.

The G1 goal, which focused on interoperability and compliance with standards, was achieved because the prototype demonstrated the ability to issue verifiable credentials in both W3C and JWT formats successfully. Decoding the JWT token revealed that its schema aligns with W3C standards, indicating that it can interoperate with other systems that adhere to the same rules.

7.2.2 Performance

The prototype supports a revocation list with 10 entries, so the same number of verifications was executed to evaluate the performance of the credential verification process. This experiment focused on measuring the average time taken to complete the verification requests, which were executed against the chosen Test Indy Network. Apache JMeter was used as the testing tool to execute the two scenarios related to the Performance quality attribute. The test plan involved executing all HTTP requests defined in the Thread Group multiple times according to the configured settings. The configuration applied (Figure 21) for addressing Questions 3 and 4 of the GQM approach was as follows:

- A single virtual user (issuer) executed the issuance process 10 consecutive times.
- A single virtual user (verifier) executed the verification process 10 consecutive times.

Since the prototype is designed to support only one voter, the experiments were done with a single virtual user. This user performed the issuance and then verification process repeatedly, matching the system's capacity for these two processes.

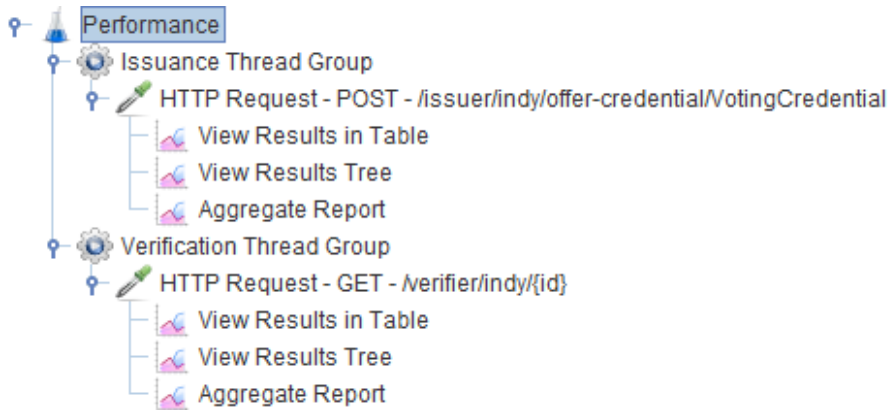

Figure 21 - JMeter test plan

The performance metrics acquired for different issuance and verification process configurations are shown in Table 12 and Table 13. The tests were performed on a computer that has Windows 11 with an AMD Ryzen 7 7435HS CPU and 24 GB of RAM. To reduce the impact of any possible anomalies or inconsistencies, each configuration was performed three times.

File Name	Samples	Min (ms)	Max (ms)	Median (ms)	Average Response Time (ms)	Error (%)
Issuance-01	10	34.788	40.985	40.177	39.852	0,00
Issuance-02	10	35.399	41.737	40.784	40.469	0,00
Issuance-03	10	34.464	41.509	40.238	40.217	0,00
Total	30	34.788	41.737	40.642	40.179	0,00

Table 12 – Average Response Time for Issuance process

File Name	Samples	Min (ms)	Max (ms)	Median (ms)	Average Response Time (ms)	Error (%)
Verification-04	10	24.342	38.704	35.979	34.906	0,00
Verification-05	10	23.754	30.140	29.968	29.354	0,00
Verification-06	10	32.241	42.390	32.322	33.803	0,00
Total	30	23.754	42.390	32.241	32.688	0,00

Table 13 - Average Response Time for the Verification process

The records of all executions are available in the project repository as an XML file, which also includes additional metrics such as throughput. No errors were encountered during any of the six executions.

- Q3 - How efficient is the credential verification process?

The results presented in Table 12 indicate average response times (ms) of 34.906, 29.354, and 33.803 across three runs, resulting in an average of approximately 32.688 ms per verification. This value represents the time needed to complete the verification of a voting credential within the current prototype. The 10% margin seeks to confirm whether the prototype indeed resolves one of the stated problems, which was that the longer average response time for credential verification than credential issuance raises a performance issue that can slow down important processes, which affects user experience and security [4].

To check if the verification time meets the M3 metric - verification time must be at least 10% less than issuance time - the following equation was evaluated:

$$\text{Verification Time} \leq \text{Issuance Time} \times 0.90$$

The joint mean issuance time was used as the basis of measurement because it was very close to the median and was a more reliable measure than the more variable minimum and maximum values. The threshold of issuance was calculated as $40.179 \times 0.9 = 36.161$ ms. Table 14 shows if each verification run passes or not the M3 metric.

File Name	Avg Verification Time (ms)	Passes M3?	Compared to 36.161 ms
Verification-04	34.906	Yes	34.906 < 36.161
Verification-05	29.354	Yes	29.354 < 36.161
Verification-06	33.803	Yes	33.803 < 36.161
Total	32.688	Yes	32.688 < 36.161

Table 14 - Verification Time Compliance with M3 Metric

The findings indicate that all verification times met the M3 metric, confirming that the verification process was consistently faster than issuance by the needed margin of 10%. The average response time is relatively high, although this was influenced by the Test Indy Network, which is an external element of the prototype. Nevertheless, the system still exceeds the performance expectations for the prototype with a view to limiting potential user delays.

- Q4 - What is the impact of credential verification performance on user experience?

To answer the Q4 question, the metric states that there must be a minimum of 90% successful verification attempts to define an acceptable user experience (i.e., most of them). To analyze this, verification requests were executed and logged through multiple test runs. There were no errors in any of the verification attempts in Table 13, so the success rate of verification was 100%. The prototype is in accordance with the M4 metric and shows a consistent success in credential verification operation.

- G2 - Evaluate the performance of credential issuance and verification in SSI integration into existing systems.

This goal was completed with success based on the results derived from the M3 and M4 performance metrics. The credential verification process consistently met the M3 requirement, achieving average response times below the 36.161 ms threshold (10% less than the average issuance time of 40.179 ms). All verification operations passed, confirming that the prototype has an efficient credential verification process that outperforms issuance and addresses prior concerns over verification delays.

Additionally, credential verification demonstrated full reliability, with a 100% success rate across all test attempts, thus meeting the M4 metric. This indicates a highly trusted system in terms of the user, as it suggests low risk of a verification failure impacting usability or trust.

In sum overall performance of the prototype for both issuance and verification suggests successful resolution of the performance bottlenecks found in the self-sovereign identity approaches, while also showing consideration for the user experience concerning usability and security requirements.

8 Conclusion

This chapter summarizes the dissertation's main achievements. It includes the research's main achievements, limitations, and possible improvements for the future. The work and prototype are also generally appreciated.

8.1 Achievements

The project started with the introduction of self-sovereign identity and the identification of several challenges inherent to this decentralized identity approach, particularly when implementing using Hyperledger Aries technology. Comprehensive planning of the work to be done was completed, including crucial phases such as selecting a project in which SSI was integrated. The project was chosen by analyzing the current practical applications of SSI, with the area of voting being selected. The work includes the reverse engineering of the initial project, and the integration approach used to improve sovereignty, based on a literature review performed.

Next, the solution approach for the prototype was presented, along with the additional requirements designed to successfully integrate SSI into the project. Then, based on the challenges identified at the beginning of this project, the prototype was evaluated, and the results were collected to validate the work performed in terms of interoperability and performance attributes. The document's outputs aim to understand how self-sovereign identity could be integrated into an existing project in the voting area, what was done to achieve it, the problems overcome, and the evaluation.

To enable other researchers and developers to build upon these results, all work, including code and documentation, is available in a public repository [8].

8.2 Limitations and Future Work

Despite the successful integration of self-sovereign identity and the general accomplishment of the purpose of the work, some limitations of the work and areas for improvement were identified.

The solution approach presented and implemented during the work includes simplifications in the integration for the voting project, such as the utilization of a RESTful API to communicate between different components. No search was done to determine which is the most reliable and secure method because the usage of this communication in the prototype was to facilitate development and experimentation. It would be interesting to include an investigation of more concrete options and the advantages and disadvantages of each. Adding person identification for voting, such as submitting documents or evaluating alternative options, would improve the usability of the prototype. However, this was not part of the work's scope.

The prototype includes a front-end component created to enable future workers to interact graphically with the solution. The focus was on the server part that includes the SSI operations such as issuance, verification, and revocation. The web application component created only supports one voter, and it would be good to improve this by adding the option for multiple people, which would require more security and authentication methods.

A promising direction for the assessment of the solution would be to do a security analysis of all components. Hyperledger Indy addressed some of the security points indirectly, with the usage of decentralized identifiers and cryptographic credentials generation. While these features address some issues, a more formal and thorough security analysis with potential vulnerabilities, threat models, and attack surfaces would allow the assessment of general security of the system to be stronger. This test would validate the robustness of the current prototype and provide valuable feedback regarding how to enhance its resilience and credibility.

8.3 Final Considerations

The significant contribution of the project was producing a working prototype that integrated SSI technology into an existing voting system. Lessons on the performance, interoperability and security of decentralized identity solutions were described. The identified challenges were resolved to have the integration with success as result and to distribute information on how this can be done to other projects.

The entire project, along with the detailed description of the prototype that was created, is an effective vehicle for sharing knowledge about the use of self-sovereign identity in voting systems and other areas.

The project held personal interest for the author, giving first-hand experience with self-sovereign identity and helping to create practical skills for possible future careers.

References

[1] - C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, P. Bellavista, and M. Conti, “A survey on decentralized identifiers and verifiable credentials,” arXiv (Cornell University), Feb. 2024, doi: https://doi.org/10.48550/arxiv.2402.02455.
[2] - N. George, “Hyperledger Aries - LF decentralized trust,” Atlassian.net, 2024. https://lf-hyperledger.atlassian.net/wiki/spaces/ARIES/overview (accessed Oct. 2024).
[3] - C. H.-J. Braun, V. Papanchev, and T. Käfer, “SISSI: An architecture for semantic interoperable self-sovereign identity-based access control on the web,” WWW ’23: Proceedings of the ACM Web Conference 2023, pp. 3011–3021, Apr. 2023, doi: https://doi.org/10.1145/3543507.3583409.
[4] - A. Siqueira, A. F. Da Conceição, and V. Rocha, “Performance evaluation of self-sovereign identity use cases,” IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), Athens, Greece, 2023, pp. 135–144, Jul. 2023, doi: https://doi.org/10.1109/dapps57946.2023.00026.
[5] - W. Chan, K. Gai, J. Yu, and L. Zhu, “Blockchain-Assisted Self-Sovereign Identities on Education: A Survey,” Blockchains, vol. 3, no. 1, p. 3, Feb. 2025, doi: https://doi.org/10.3390/blockchains3010003.
[6] - T. Titcombe, “What is e-voting, and which problems could it help solve? – OpenMined,” OpenMined, Nov. 05, 2020. https://openmined.org/blog/what-is-e-voting-and-which-problems-could-it-help-solve/ (accessed May 06, 2025).
[7] - B. J. Oates, Researching information systems and computing. Los Angeles: Sage, 2006. Available: https://dokumen.pub/researching-information-systems-and-computing-978-1-4129-0223-6.html.
[8] - jorgefcferreira, “GitHub - jorgefcferreira/example-voting-app,” GitHub, 2025. https://github.com/jorgefcferreira/example-voting-app (accessed May 19, 2025).
[9] - Instituto Politécnico do Porto, “Código de Boas Práticas e de Conduta do Instituto Politécnico do Porto,” Iscap.ipp.pt. https://www.iscap.ipp.pt/regulamentos/CodigoboaspraticasedecondutaIPP.pdf (accessed Nov. 15, 2024).
[10] - “IEEE Code of Conduct,” Ieee.org, 2024. https://www.ieee.org/about/corporate/governance/code-of-conduct.html (accessed Nov. 15, 2024).
[11] - P. Herbke, S. Lamichhane, and A. Sapkota, “Lifecycle Management of Resumés with Decentralized Identifiers and Verifiable Credentials,” arXiv (Cornell University), pp. 1–3, Oct. 2024, doi: https://doi.org/10.1109/brains63024.2024.10732402.
[12] - “Decentralized Identifiers (DIDs) v1.0,” W3.org, Jul. 19, 2022. https://www.w3.org/TR/2022/REC-did-core-20220719/ (accessed Nov. 12, 2024).
[13] - “Verifiable Credentials Data Model v1.1,” W3.org, Mar. 03, 2022. https://www.w3.org/TR/2022/REC-vc-data-model-20220303/ (accessed Nov. 12, 2024).
[14] - U. Der, S. Jähnichen, and J. Sürmeli, “Self-sovereign Identity - Opportunities and Challenges for the Digital Revolution,” *arXiv.org*, 2017. https://arxiv.org/pdf/1712.01767 (accessed Nov. 08, 2024).
[15] - C. Allen, “The Path to Self-Sovereign Identity,” www.LifeWithAlacrity.com, Apr. 26, 2016. https://www.lifewithalacrity.com/article/the-path-to-self-soverereign-identity/ (accessed Nov. 29, 2024).
[16] - European Commission, “What data can we process and under which conditions?” commission.europa.eu. https://commission.europa.eu/law/law-topic/data-protection/reform/rules-business-and-organisations/principles-gdpr/overview-principles/what-data-can-we-process-and-under-which-conditions_en (accessed Nov. 29, 2024).
[17] - Hyperledger Foundation, “Aries,” Lfdecentralizedtrust.org, Jul. 31, 2023. https://www.lfdecentralizedtrust.org/projects/aries (accessed Dec. 23, 2024).
[18] - openwallet-foundation, “GitHub - openwallet-foundation/credo-ts-docs: Documentation for Credo,” GitHub, 2022. https://github.com/openwallet-foundation/credo-ts-docs (accessed Dec. 23, 2024).
[19] - “Veramo - A JavaScript Framework for Verifiable Data | Performant and modular APIs for Verifiable Data and SSI,” veramo.io. https://veramo.io/ (accessed Dec. 23, 2024).
[20] - “Sovrin Governance Framework,” Sovrin. https://sovrin.org/library/sovrin-governance-framework/ (accessed Dec. 23, 2024).

[21] - “ION - an open, public, permissionless decentralized identifier network,” identity.foundation. https://identity.foundation/ion/ (accessed Dec. 23, 2024).
[22] - Hyperledger Foundation, “Indy,” Lfdecentralizedtrust.org, Jul. 31, 2023. https://www.lfdecentralizedtrust.org/projects/hyperledger-indy (accessed Apr. 24, 2025).
[23] - C. Kukreja, “Verified Credential Ecosystem - Chuni Lal Kukreja - Medium,” Medium, Sep. 05, 2024. https://medium.com/@chunilalkukreja/verified-credential-ecosystem-0bf76a0234a0 (accessed May 18, 2025).
[24] - “LF Decentralized Trust - The open-source foundation for decentralized technologies,” Lfdecentralizedtrust.org, 2025. https://www.lfdecentralizedtrust.org/ (accessed May 18, 2025).
[25] - A. Goel and Y. Rahulamathavan, “A Comparative Survey of Centralised and Decentralised Identity Management Systems: Analysing Scalability, Security, and Feasibility,” Future Internet, vol. 17, no. 1, pp. 1–1, Dec. 2024, doi: https://doi.org/10.3390/fi17010001.
[26] - R. Zoun, “Self-Sovereign Identity: How it Works and How it Impacts our Lives,” www.adnovum.com. https://www.adnovum.com/blog/self-sovereign-identity-ssi-switzerland (accessed Mar. 01, 2025).
[27] - R. Centonze and R. Reale, “Self-sovereign identity as a tool for digital democracy,” arXiv.org, 2021. https://arxiv.org/abs/2106.11714 (accessed Mar. 01, 2025).
[28] - V. Basili, G. Caldiera, and R. H. Dieter, “The goal question metric approach,” 1994. https://www.ecs.csun.edu/~rlingard/COMP587/gqm.pdf.
[29] - M. Kuhrmann, D. Méndez Fernández, and M. Daneva, “On the Pragmatic Design of Literature Studies in Software Engineering: An Experience-based Guideline,” arXiv.org, 2016. https://arxiv.org/abs/1612.03583 (accessed Nov. 08, 2024).
[30] - A. Carrera-Rivera, W. Ochoa, F. Larrinaga, and G. Lasa, "How-to Conduct a Systematic Literature Review: a Quick Guide for Computer Science Research," MethodsX, vol. 9, no. 1, p. 101895, 2022, doi: https://doi.org/10.1016/j.mex.2022.101895.
[31] - Monash University, “Developing research questions,” Monash University, 2023. https://www.monash.edu/library/help/assignments-research/developing-research-questions (accessed Nov. 08, 2024).
[32] - MacQuarie University, “Subject and Research Guides: Systematic Reviews: Step 6: PRISMA Flow Diagram & Screen,” Mq.edu.au, 2020. https://libguides.mq.edu.au/systematic_reviews/prisma_screen (accessed Nov. 08, 2024).
[33] - Manoj T, M. Krishnamoorthi, and Narendra, “A Blockchain Based Decentralized Identifiers for Entity Authentication in Electronic Health Records,” Cogent Engineering, vol. 9, no. 1, Mar. 2022, doi: https://doi.org/10.1080/23311916.2022.2035134.
[34] - G. Kim and J.-C. Ryou, “Digital Authentication System in Avatar Using DID and SBT,” Mathematics, vol. 11, no. 20, pp. 4387–4387, Oct. 2023, doi: https://doi.org/10.3390/math11204387.
[35] - I.-C. Lin, I-Ling. Yeh, C.-C. Chang, J.-C. Liu, and C.-C. Chang, “Designing a Secure and Scalable Data Sharing Mechanism Using Decentralized Identifiers (DID),” Computer Modeling in Engineering & Sciences, vol. 0, no. 0, pp. 1–10, Jan. 2024, doi: https://doi.org/10.32604/cmes.2024.051612.
[36] - D. Naicker and M. Moodley, “Challenges of user data privacy in self-sovereign identity verifiable credentials for autonomous building access during the COVID-19 pandemic,” Frontiers in Blockchain, vol. 7, Mar. 2024, doi: https://doi.org/10.3389/fbloc.2024.1374655.
[37] - J. Fang, T. Feng, X. Guo, and X. Wang, “Privacy-enhanced distributed revocable identity management scheme based self-sovereign identity,” Journal of Cloud Computing Advances Systems and Applications, vol. 13, no. 1, Nov. 2024, doi: https://doi.org/10.1186/s13677-024-00715-8.
[38] - A. Satybaldy, A. Subedi, and M. Nowostawski, “A Framework for Online Document Verification Using Self-Sovereign Identity Technology,” Sensors, vol. 22, no. 21, p. 8408, Nov. 2022, doi: https://doi.org/10.3390/s22218408.
[39] - D. Richter and J. Anke, “Exploring Potential Impacts of Self-Sovereign Identity on Smart Service Systems,” Business Information Systems, pp. 105–116, Jul. 2021, doi: https://doi.org/10.52825/bis.v1i.68.
[40] - dockersamples, “GitHub - dockersamples/example-voting-app: Example distributed app composed of multiple containers for Docker, Compose, Swarm, and Kubernetes,” GitHub, 2015. https://github.com/dockersamples/example-voting-app (accessed Mar. 25, 2025).

[41] - Apache, “APACHE LICENSE, VERSION 2.0,” Apache.org, 2019. https://www.apache.org/licenses/LICENSE-2.0 (accessed Mar. 01, 2025).
[42] - “Agents | Credo,” https://credo.js.org/, 2025. https://credo.js.org/guides/concepts/agents (accessed Mar. 01, 2025).
[43]- “BCovrin Test Indy Network” Vonx.io, 2025. http://test.bcovrin.vonx.io/ (accessed Apr. 01, 2025).
[44] - Government of British Columbia, “VON Network,” GitHub, Jan. 11, 2023. https://github.com/bcgov/von-network (accessed Feb. 15, 2025).
[45] - Hyperledger Indy, “How Credential Revocation Works — Hyperledger Indy SDK documentation,” https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html (accessed Apr. 24, 2025).
[46] - hyperledger, “indy-hipe/text/0011-cred-revocation at master · hyperledger/indy-hipe,” GitHub, 2018. https://github.com/hyperledger/indy-hipe/tree/master/text/0011-cred-revocation (accessed May 15, 2025).
[47] - OpenID Foundation, “OpenID for Verifiable Credentials - OpenID Foundation,” OpenID Foundation - Helping people assert their identity wherever they choose, Apr. 19, 2023. https://openid.net/sg/openid4vc/ (accessed Apr. 24, 2025).
[48] - “AnonCreds Specification,” hyperledger.github.io. https://hyperledger.github.io/anoncreds-spec/ (accessed Apr. 24, 2025).
[49] - “What Is OpenID Connect (OIDC) and How Does It Work?” Frontegg, Aug. 20, 2024. https://frontegg.com/guides/oidc-authentication (accessed May 15, 2025).
[50] - “Online JWT Decoder,” FusionAuth, 2025. https://fusionauth.io/dev-tools/jwt-decoder (accessed Jun. 01, 2025).
[51] - Westfall Team, “12 Steps to Useful Software Metrics,” Softwareexcellenceacademy.com, 2025. https://www.softwareexcellenceacademy.com/12_Steps_To_Useful_Software_Metrics (accessed Mar. 15, 2025).
[52] - “Verifiable Credentials: The Ultimate Guide 2025,” www.dock.io, Apr. 29, 2025. https://www.dock.io/post/verifiable-credentials (accessed May 31, 2025).

Appendix A

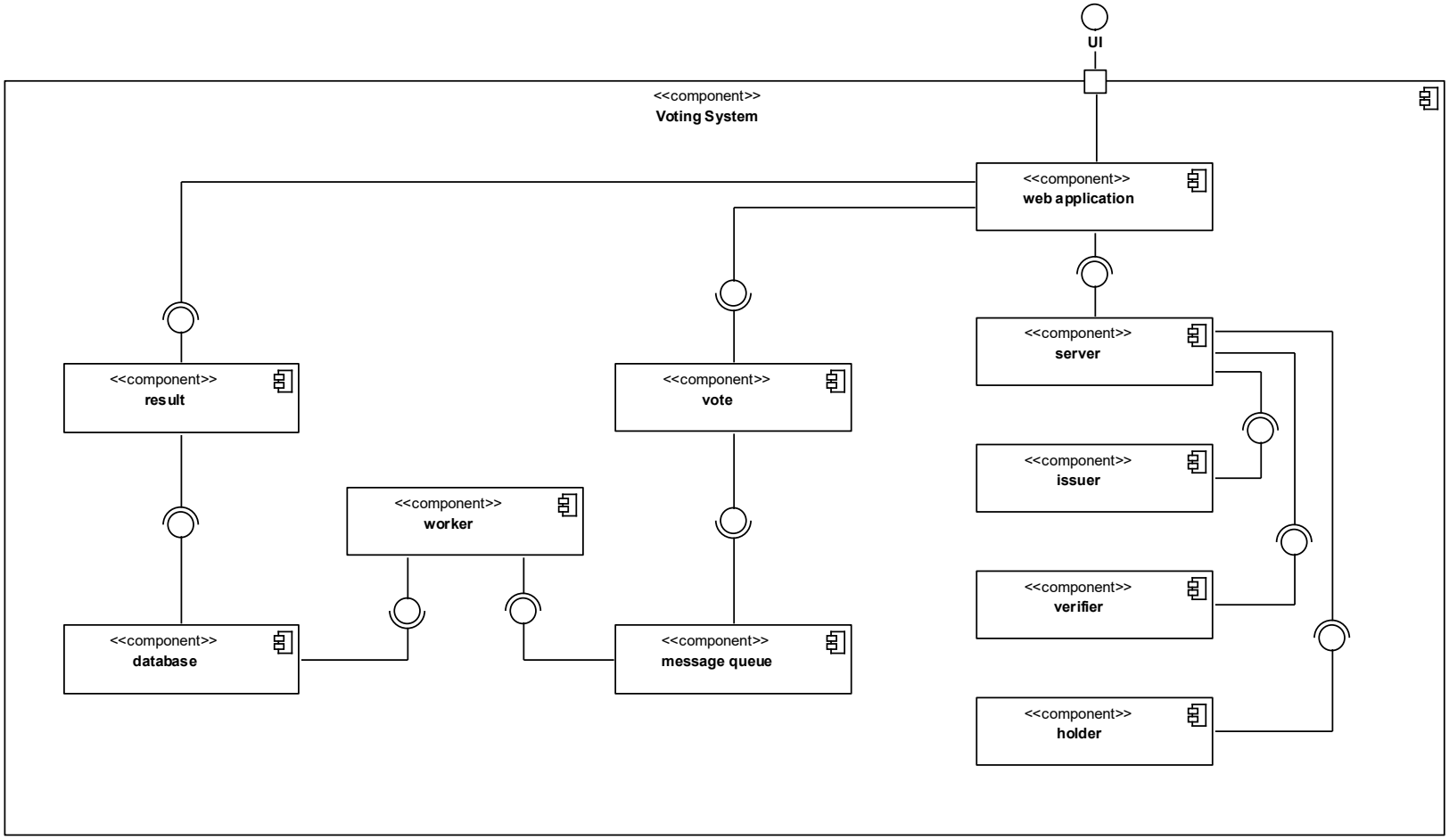

Figure 22 - Full prototype components interaction

Appendix B

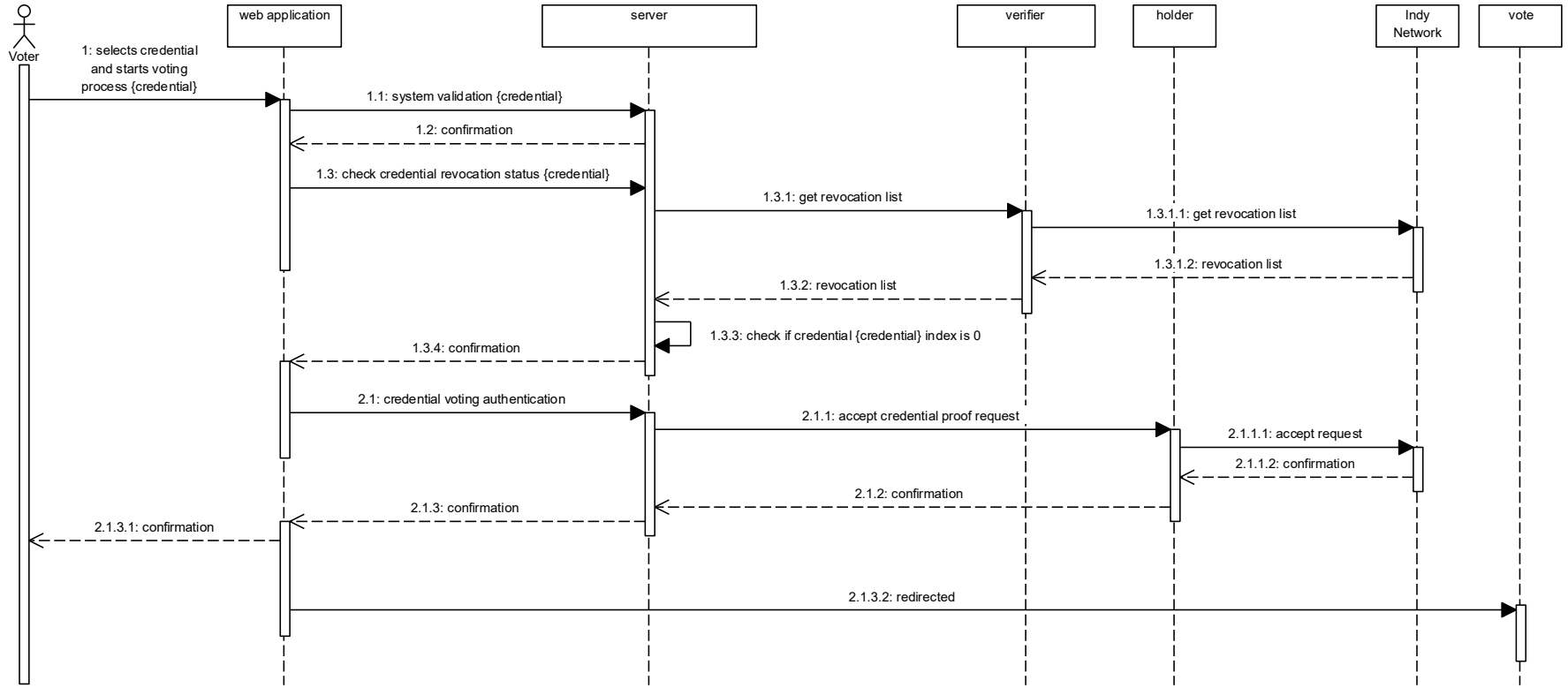

Figure 23 - Voting Authentication UML Sequence Diagram

Appendix C

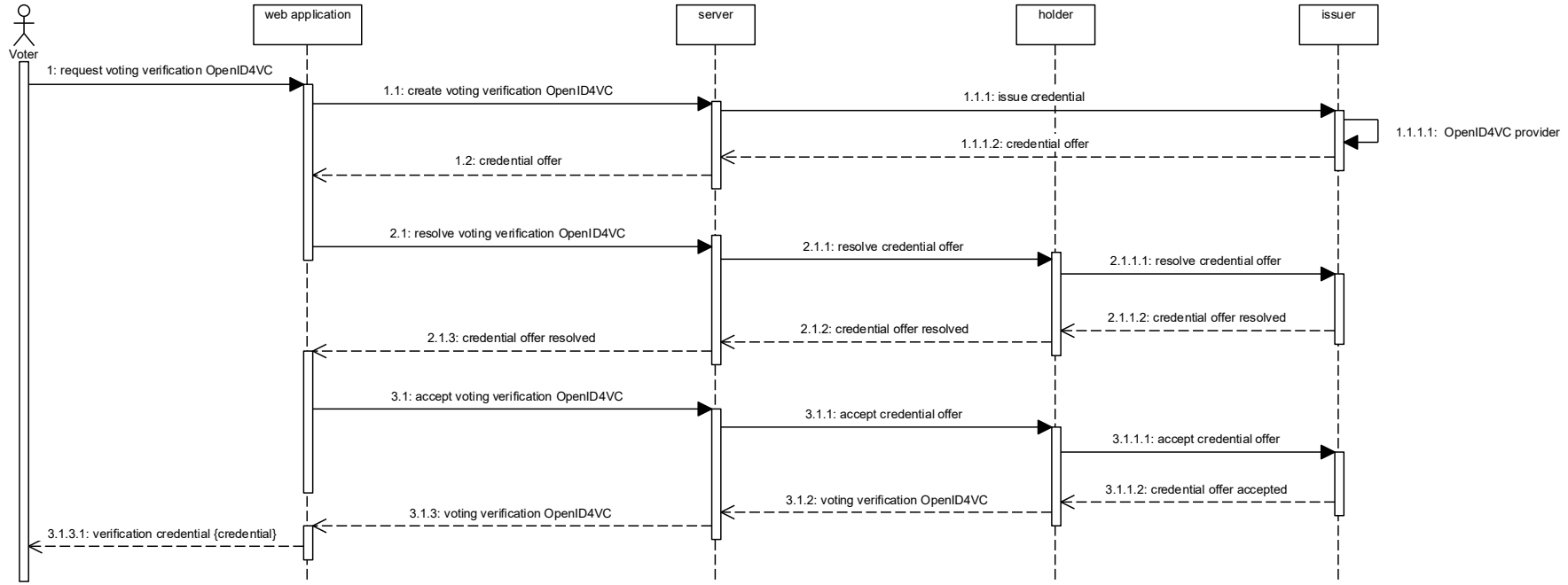

Figure 24 - Voting Verification credential request UML Sequence Diagram