University of Zurich^UZH

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Functional Requirements Engineering for Decentralized Identity Applications

Xiong Li
Zurich, Switzerland
Student ID: 21-739-594

Supervisor: Daria Schumm, Katharina Müller, Prof. Dr. Burkhard Stiller
Date of Submission: August 15, 2025

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Declared tools used in this thesis are:

- Grammarly: for the correction of grammar, spelling.
- QuillBot: for polishing writing.
- DeepL: for translation assistance and language refinement
- Google Translate: for basic translation queries
- Connected Papers: for literature discovery and exploring research connections

Zürich, 15.08.2025

[illegible]

Signature of student

Abstract

Self-sovereign identity (SSI) is a user-centric and decentralized paradigm for digital identity, empowering individuals with full control over their personal data. This concept is guided by high-level principles such as privacy, security, and user autonomy, which function as Non-Functional Requirements (NFRs) for system design. However, while NFRs are sometimes complicated for system evaluation and verification, a significant gap exists between these abstract principles and the concrete, verifiable Functional Requirements (FRs) due to the lack of a systematic derivation methodology.

This thesis addresses this challenge by developing a systematic framework to translate the abstract NFRs of SSI into a concrete catalog of verifiable FRs. Combining simplified Softgoal interdependency graph(SIG), SSI use case analysis, and a comprehensive analysis of SSI literature, a structured, traceable process was designed to decompose NFRs and map them to the core architectural components of the SSI model (Issuer, Holder, and Verifier). The framework's practical utility was validated through in-depth case studies on two prominent SSI systems, Sovrin and uPort, demonstrating its effectiveness in providing a structured, evidence-based assessment of system compliance and enabling objective comparisons.

Self-Sovereign Identity (SSI) ist ein nutzerzentriertes und dezentrales Paradigma für digitale Identitäten, das Einzelpersonen die volle Kontrolle über ihre persönlichen Daten ermöglicht. Dieses Konzept orientiert sich an übergeordneten Prinzipien wie Datenschutz, Sicherheit und Nutzerautonomie, die als nichtfunktionale Anforderungen (NFRs) für das Systemdesign fungieren. Obwohl NFRs für die Systemevaluierung und -verifizierung manchmal kompliziert sind, besteht aufgrund des Fehlens einer systematischen Herleitungsmethode eine erhebliche Lücke zwischen diesen abstrakten Prinzipien und den konkreten, überprüfbaren funktionalen Anforderungen (FRs).

Diese Arbeit begegnet dieser Herausforderung durch die Entwicklung eines systematischen Rahmens zur Übersetzung der abstrakten NFRs von SSI in einen konkreten Katalog überprüfbarer FRs. Durch die Kombination eines vereinfachten Softgoal-Interdependenzgraphen (SIG), einer SSI-Anwendungsfallanalyse und einer umfassenden Analyse der SSI-Literatur wurde ein strukturierter, nachvollziehbarer Prozess entwickelt, um NFRs zu zerlegen und sie den zentralen Architekturkomponenten des SSI-Modells (Aussteller, Inhaber und Prüfer) zuzuordnen. Der praktische Nutzen des Frameworks wurde durch ausführliche Fallstudien zu zwei bekannten SSI-Systemen, Sovrin und uPort, bestätigt. Dabei wurde seine Wirksamkeit bei der Bereitstellung einer strukturierten, evidenzbasierten Bewertung der Systemkonformität und der Ermöglichung objektiver Vergleiche nachgewiesen.

Acknowledgments

First and foremost, I am deeply grateful to Prof. Dr. Burkhard Stiller, my supervising professor for granting me this opportunity to conduct this thesis under his group. His invaluable guidance and insights have played a crucial role in shaping this work.

I would also like to express my sincere gratitude to Daria Schumm and Katharina Müller for their unwavering support, constructive suggestions, and profound expertise throughout this thesis.

Finally, I would like to express my heartfelt appreciation to my family and girlfriend, Bingyan Lin, for their continuous support, patience, and understanding during the intensive periods of this thesis work. Their encouragement during challenging moments motivated me to persevere and complete this work.

Contents

Chapter 1

Introduction

1.1 Introduction and Motivation

Self-Sovereign Identity (SSI) represents a paradigm shift in digital identity management, empowering individuals with control over their data [8, 33, 46]. By decentralizing the control of identity, SSI aims to enhance privacy, security, and user autonomy in the digital world [24, 41].

The foundational principles of decentralized identity applications are largely based on the pioneering work of Christopher Allen's "The Path to Self-Sovereign Identity" [1] and Kim Cameron's "Laws of Identity" [3]. These works established a set of ideals and properties that a Self-Sovereign identity system should achieve. More recent research, such as the work by Cucko et al. [7], has extended and systemized this set of properties, which can be viewed as NFRs for system evaluation.

However, a fundamental challenge persists: these properties, much like traditional NFRs, are abstract, qualitative, and often difficult to model, measure, or verify directly. For instance, while a principle like "Privacy and Minimal Disclosure" is a cornerstone of SSI, it does not, on its own, provide a concrete, testable specification for a software developer to implement or a quality assurance engineer to validate. This abstraction layer creates a significant gap between the high-level vision of SSI and the practical engineering of robust, compliant, and verifiable systems.

The current literature reveals very limited attempts to bridge this gap. While some research, such as that presented in [31], provides a list of FRs for SSI, it does not systematically map these requirements back to the foundational NFRs and principles that they are meant to satisfy. Similarly, while industry white papers like the QuarkID paper [34] map SSI principles to high-level design principles, they lack a detailed discussion or analysis of the design principles.

This thesis is motivated by the critical need for a systematic and traceable method to translate the abstract NFRs of SSI into concrete and verifiable FRs. Without such a bridge, it remains extremely difficult to:

- **Objectively evaluate** whether an application truly adheres to the core principles of SSI.
- **Compare** different SSI solutions in a meaningful, structured manner.
- **Guide developers** in implementing systems that are verifiably compliant with SSI ideals.

By developing a framework to derive FRs from NFRs systematically, this work aims to provide the necessary tools for rigorous analysis, development, and evaluation of decentralized identity applications, ultimately contributing to the creation of more trustworthy and effective SSI ecosystems.

Therefore, the primary contribution of this thesis is the structured and actionable NFR-to-FR derivation methodology and the validation of the methodology, which demonstrates its applicability and ability.

1.2 Thesis Goals

The primary goal of this thesis is to develop and validate a systematic framework for deriving and specifying Functional Requirements (FRs) for Self-Sovereign Identity (SSI) systems directly from their guiding principles and Non-Functional Requirements (NFRs). This framework will serve as a bridge between high-level concepts and concrete Functional Requirements (FRs), enabling a more structured and measurable approach to SSI system analysis.

To achieve this main objective, the following specific goals are defined:

1. **Establish a Foundational Understanding:** Conduct a comprehensive literature review to identify and consolidate the core principles, NFRs, architectural components (Issuer, Holder, Verifier), and fundamental processes (e.g., credential issuance, verification) within SSI and decentralized identity systems.
2. **Develop a Systematic NFR-to-FR Mapping Methodology:** Design a clear, repeatable, and traceable methodology for decomposing abstract NFRs and deriving corresponding FRs. This includes mapping NFRs to the specific system components and processes responsible for their fulfillment.
3. **Create a Catalog of Functional Requirements:** Apply the developed methodology to generate a structured catalog of essential FRs for SSI systems. These requirements will be formulated as clear, actionable statements aligned with SSI principles.
4. **Design a Practical Evaluation Framework:** Construct an evaluation framework based on the derived FRs that can be used to systematically assess the functional completeness and compliance of existing decentralized identity applications with core SSI principles.

5. **Validate the Framework through Case Studies:** Apply the evaluation framework to prominent, real-world SSI systems to demonstrate its practical utility, effectiveness, and ability to produce objective and deterministic conclusions.

1.3 Methodology

To achieve the goals outlined above, this thesis adopts a constructive research methodology, which involves the design, demonstration, and evaluation of a new artifact, the NFR-to-FR mapping and evaluation framework. The research process is structured into three distinct phases:

(i) Phase 1: Foundational Research and Analysis: This initial phase focuses on building a comprehensive theoretical basis. It involves an extensive literature review of SSI principles, NFRs in software engineering, and existing decentralized identity architectures. The key NFRs guiding SSI, as identified in the literature [7], will be selected. The core components of the SSI ecosystem (Issuer, Holder, Verifier) and their interaction patterns will be analyzed based on established models like the W3C Verifiable Credentials and trust triangle.

(ii) Phase 2: NFR-to-FR Derivation and Framework Design: This phase constitutes the core constructive work of the thesis. A systematic methodology, inspired by Softgoal Interdependency Graph (SIG) introduced in [6] and use case analysis in literature [31], will be designed to decompose high-level NFRs, the methodology identify the use cases and concrete steps for SSI, afterwards each NFR will be mapped to relevant use case, system processes and components. From this mapping, concrete FRs will be derived and formulated using standardized language (e.g., RFC 2119 [2]) to ensure clarity and testability. The collection of these derived FRs will form a catalog, which will then serve as the foundation for the design of the evaluation framework.

(iii) Phase 3: Validation and Evaluation: In the final phase, the designed evaluation framework is validated for its practical applicability. This will be achieved by conducting case studies on two prominent SSI systems: Sovrin and uPort. Evidence will be gathered from technical documentation, white papers, and other available resources to assess the extent to which these systems satisfy the derived FRs. The results will be analyzed to answer the research questions, evaluate the effectiveness of the framework itself, and demonstrate its ability to provide a structured comparison of different SSI implementations.

1.4 Thesis Outline

The structure of the thesis is organized as follows:

- Chapter 2 (Fundamentals) 2: This chapter establishes the theoretical foundation for the thesis. It begins with a background on Self-Sovereign Identity (SSI), Functional Requirements (FRs), and Non-Functional Requirements (NFRs). It then provides a review of related work, analyzing existing approaches to SSI evaluation and requirements engineering. Finally, it consolidates this analysis into a precise problem statement to address.

- Chapter 3 (Design) 3: This chapter details the core contribution of the thesis. It presents the design of the systematic framework for deriving Functional Requirements from Non-Functional Requirements in the context of SSI. It describes the NFR-to-FR mapping process, the formulation of the requirements catalog, and the design of the final evaluation process.

- Chapter 4 (Results) 4: This chapter demonstrates the practical application of the framework designed in Chapter 3. It details the selection of case studies (Sovrin and uPort) and applies the evaluation framework to assess their compliance with a selection of critical derived Functional Requirements.

- Chapter 5 (Evaluation) 3: This chapter analyzes and discusses the results obtained in the previous chapter. It provides a comparative analysis of the case studies, answers the research questions proposed at the beginning of the thesis, and critically evaluates the strengths and limitations of the developed framework.

- Chapter 6 (Final Considerations) 6: This chapter summarizes the entire work, reiterates the key findings and contributions, and discusses potential avenues for future research that can extend upon the foundation laid by this thesis.

Chapter 2

Fundamentals

This chapter explores the theoretical and contextual groundwork for the thesis. It begins by introducing the fundamental concepts of requirements engineering and Self-Sovereign Identity (SSI). It then reviews the existing work related to SSI requirements and evaluation frameworks, which leads to a clear and focused problem statement that this thesis aims to solve.

2.1 Background

2.1.1 Self-sovereign identity

Self-sovereign identity (SSI) represents a paradigm shift from centralized and federated identity models toward user-controlled digital identity management. Unlike centralized systems where a single authority controls user identities, or federated systems where identity providers act as intermediaries, SSI empowers individuals with direct control over their identity data and its disclosure [4, 8, 10, 40, 54].

SSI is founded on foundational principles developed by Allen [1] and Cameron [3], while Allen defines the key properties of the SSI system are Existence, Control, Access, Transparency, Persistence, Portability, Interoperability, Consent, Minimalization, and Protection [1].

SSI systems implement a three-party trust model consisting of issuers (entities that create and sign credentials), holders (individuals who store credentials in digital wallets), and verifiers (entities that validate presented credentials) [11, 22, 28], this three components' structure of SSI are called "Trust of Triangle" [33, 54]. Figure 2.1 illustrates this three-way' interaction.

This interaction is enabled by a set of key techniques, largely standardized by the World Wide Web Consortium (W3C) [59, 60]:

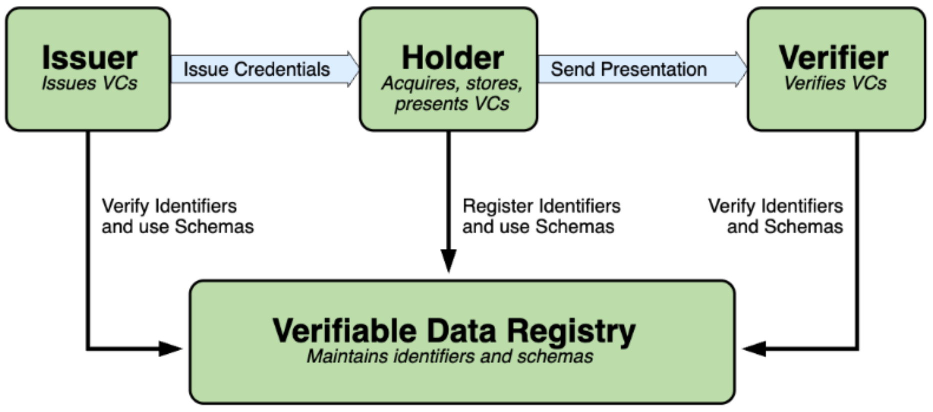
Issuer
Issues VCs
Issue Credentials
Holder
Acquires, stores, presents VCs
Send Presentation
Verifier
Verifies VCs
Verify Identifiers and use Schemas
Register Identifiers and use Schemas
Verify Identifiers and Schemas
Verifiable Data Registry
Maintains identifiers and schemas

Figure 2.1: Trust Triangle (Source: [60]

1. Decentralized Identifiers (DIDs): Globally unique, user-controlled identifiers that serve as the anchor for an identity, without reliance on a central registry [11, 27, 36, 59].
2. Verifiable Credentials (VCs): Tamper-evident and cryptographically verifiable digital versions of physical credentials (e.g., a driver's license, a diploma) [42, 55], it's a set of claims, which is a statement about a subject, from the same entity [60].

Self-sovereign identity architecture is illustrated in figure 2.2

2.1.2 Functional and Non-Functional Requirements

2.1.2.1 Functional Requirements (FR)

Functional requirements are used to describe what the system really should do, which might be expressed as functions, tasks, or services the system should perform (i.e, Users should be able to log in with their username and password) [14, 26, 61]. In other words, the system's basic behavior can be considered as the functional requirements [37].

2.1.2.2 Non-Functional Requirements (NFR)

In contrast with functional requirements, non-functional requirements describes how the system shall do something rather than what the system shall do [12]. In other words, it's a systems' property or quality, e.g., speed or accuracy property [38].

2.1.3 Softgoal Inderdependency Graph (SIG)

The Softgoal Interdependency Graph (SIG) is a visual, goal-oriented modeling technique from the Non-Functional Requirements (NFR) Framework used to analyze and reason

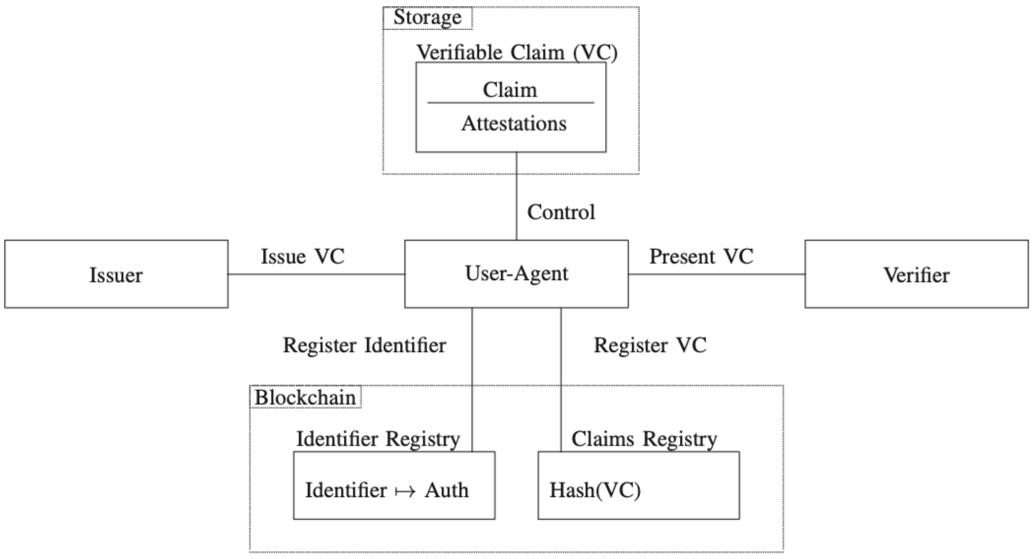

Figure 2.2: SSI Architecture (Source: [28]

about a system's quality attributes [6]. Its primary purpose is to systematically refine high-level, ambiguous NFRs (e.g., Privacy, Security) into concrete design decisions and to make the trade-offs between competing goals explicit.

A SIG is composed of two main elements:

- **Nodes (Goals):** These represent requirements at different levels of abstraction. The most important are Softgoals, which are the NFRs themselves (e.g., Privacy), and Operationalizations, which are the specific technical mechanisms or design choices implemented to achieve them.
- **Edges (Contributions):** These are labeled links that show how different elements influence each other. Contribution links model the impact one goal has on another, using labels such as:
  - MAKE (++) and HELP (+) for positive contributions.
  - BREAK (–) and HURT (-) for negative contributions (trade-offs).

2.2 Related work

2.2.1 SSI Evaluation and Assessment Frameworks

Several researchers have attempted to evaluate and compare SSI systems, though most of them lack systematic approaches for translating abstract principles into concrete requirements. Satybaldy et al. [39] proposed an evaluation framework comparing Sovrin, uPort,

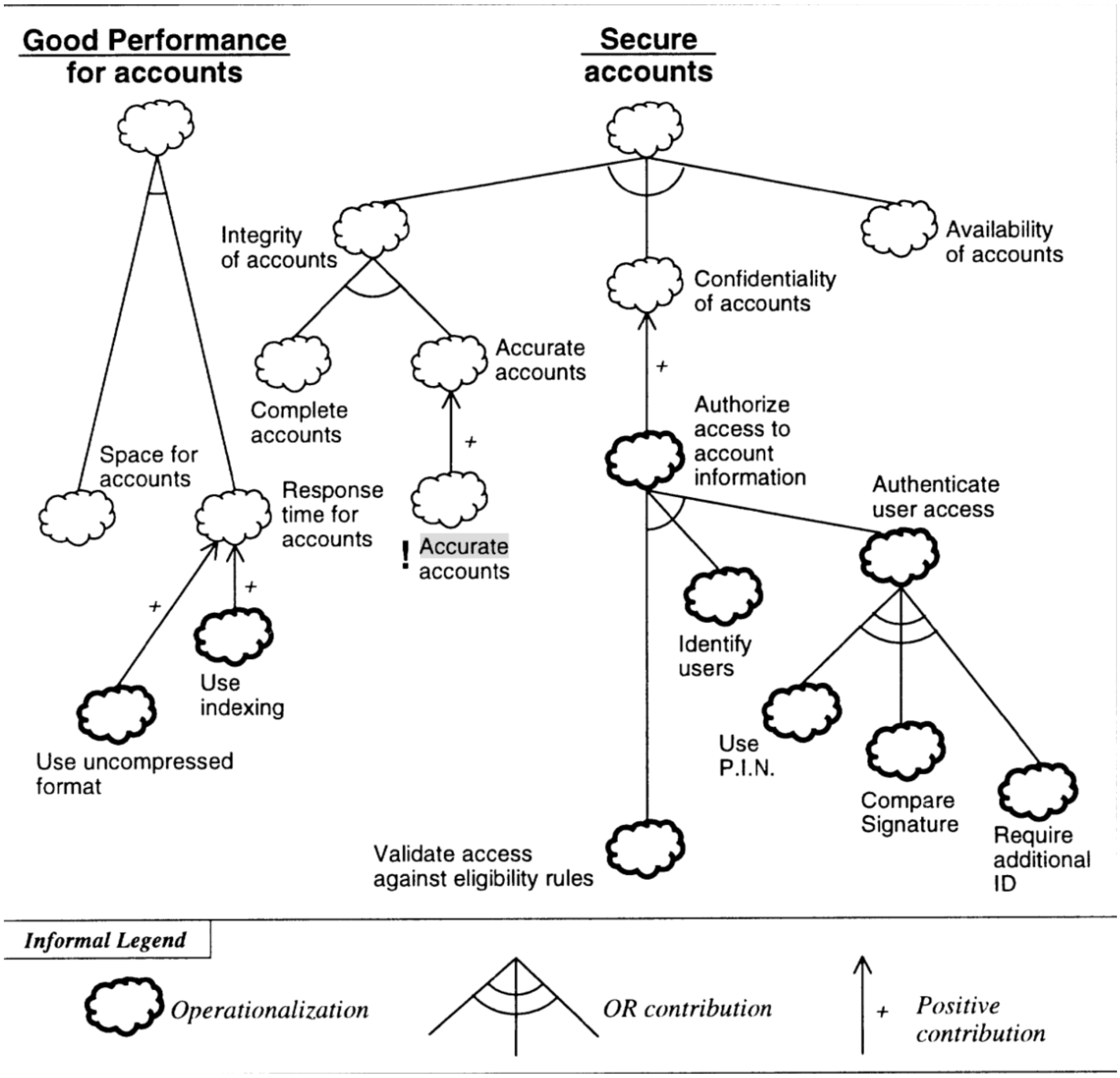

Figure 2.3: SIG example (Source: [6])

ShoCard, Civic, and Blockstack by utilizing criteria derived from established models, including Cameron's laws of identity. The framework emphasizes key aspects such as security, data integrity, privacy, and usability, which are essential for assessing the effectiveness of SSI systems. However, their approach does not provide a systematic methodology for deriving functional requirements from the abstract principles they evaluate.

Pava-Diaz et al. [32] conducted a contextual analysis and quantification of SSI principles implementation, exploring nine prominent frameworks including Sovrin, uPort, Jolocom, ShoCard, Litentry, Civic, KILT, Idena, and ION, highlighting their features, functionalities, and compliance with digital identity principles. However, their approach primarily focused on high-level principle compliance rather than systematic functional requirement derivation.

Schardong et al. [40] conducted a comprehensive systematic review of SSI literature, providing mapping and taxonomy of theoretical and practical advances in SSI. While comprehensive, their work did not address the specific challenge of bridging abstract principles with concrete implementation requirements.

Taken together, these studies highlight a shared trend in the literature. While Satybaldy et al. and Pava-Diaz et al. provide valuable frameworks for high-level comparison and principle quantification respectively, and Schardong et al. offer a broad taxonomy of the field, they all share a common limitation: none of them present a formal, repeatable methodology for systematically deriving concrete and verifiable functional requirements from the abstract principles they discuss.

2.2.2 SSI Requirements and System Analysis

Several comparative analyses of SSI systems have been performed, with detailed analysis of ShoCard, Sovrin, Civic and uPort. These systems use certain decentralization techniques based on the authors' criteria and principles, though none fully complied with the complete SSI requirements. This finding highlights a critical gap: existing SSI systems struggle to fully satisfy the foundational principles, potentially due to the lack of systematic translation from principles to implementation requirements.

Nokhbeh Zaeem et al. [31] provided a survey of blockchain-based self-sovereign identity requirements, use-cases, and comparative study, organizing SSI use cases into seven functional groups based on analysis of existing solutions and standards. Their operational reference model (ORM) framework serves as one of the few structured approaches to organizing SSI functionality, though it does not establish clear traceability from NFRs to functional requirements, which are still high-level.

2.2.3 NFR-to-FR Mapping Methodologies

The broader requirements engineering field has developed several approaches for handling non-functional requirements, though none specifically address the SSI domain. Chung et al. [6] presented the NFR Framework in "Non-Functional Requirements in Software

Engineering," offering structured graphical facilities for stating NFRs and managing them by refining and inter-relating NFRs, justifying decisions, and determining their impact. Their Softgoal Interdependency Graphs (SIGs) methodology provides a foundation for systematic NFR decomposition that has influenced this thesis's approach.

Recent research has explored characteristics of NFR methods, comparing approaches like MOQARE and the IESE-NFR-method that aim at deriving detailed requirements from quality attributes, but use different concepts and processes [16]. However, these approaches have not been applied specifically to decentralized identity systems or SSI contexts.

Chung et al. [5] proposed a framework for representing and integrating NFRs with FRs in use case models, presenting NFR propagation rules to eliminate redundant specifications. While providing valuable integration concepts, their approach lacks the domain-specific considerations necessary for SSI systems.

2.2.4 Evaluation Frameworks for Decentralized Systems

Soltani et al. [43] conducted a comprehensive survey of the self-sovereign identity ecosystem, providing an overview of the key challenges, research opportunities around self-sovereign identity, and evolution of identity management systems, noting that existing identity management models face a set of challenges.

2.3 Problem statement

As pointed out previously, Self-Sovereign Identity (SSI) represents a transformative approach to digital identity management that empowers individuals with direct control over their identity data and its disclosure, moving away from centralized and federated identity models [10, 24, 40, 46]. However, while functional requirements define what a system should do and are essential for system structure and functions analysis [15, 38], existing SSI research predominantly concentrates on the analysis and categorization of Non-Functional Requirements, along with evaluating how well SSI solutions comply with abstract principles, without establishing systematic approaches to derive concrete functional specifications from high-level SSI properties.

This gap between principle-centered research and the concrete functional specifications required for systematic SSI system development and assessment motivated this research. The goal of this thesis is to answer the question: **How can we establish a systematic methodology to derive concrete functional requirements from abstract SSI principles and enable system evaluation?** In response, a structured NFR-to-FR mapping methodology that systematically transforms SSI principles into verifiable functional specifications was developed, accompanied by an evaluation framework that assesses SSI systems based on derived functional criteria, thereby establishing explicit connections between theoretical NFRs and practical system implementation requirements.

Chapter 3

Design

This chapter details the core contribution of this thesis: the design of a systematic framework for deriving and specifying Functional Requirements (FRs) from the high-level principles of Self-Sovereign Identity (SSI). Building upon the comprehensive classification of SSI properties by Cucko et al. [7], this chapter's objective is to create a concrete, testable, and traceable bridge between those abstract Non-Functional Requirements (NFRs) and the specific functionalities(FRs) a system must implement.

3.1 Design of the NFR-to-FR Mapping Framework

The proposed methodology is characterized by an iterative, component-centric(Identity Holder, Issuer, Verifier), process-based, and traceable approach. It is structured into three distinct phases, encompassing four detailed steps, as shown in Figure 3.1

1. Phase 1: Foundation and Context Setting – Understanding the 'Why' and 'What': understanding NFRs, system components, and general process flow of SSI systems.

2. Phase 2: NFR Decomposition and FR Derivation – Breaking down the 'How': categorizing NFRs, deciding relevant SSI process/scenario, mapping them to components, defining concrete requirements: identifying concrete FRs.

3. Phase 3: Validation, Refinement, and Documentation – Ensuring quality and usability: validating FRs, prioritizing them, and formulating them systematically. Any Functional Requirement (FR) that fails validation against key criteria—such as clarity, testability, or correct component assignment-is returned to Phase 2 for refinement. This iterative feedback loop continues until all requirements successfully pass validation, guaranteeing that the final FR catalog is clarified, testable, assigned to correct components.

This structured approach aims to bridge the gap between abstract NFRs and concrete FRs, providing a clear path for system design and evaluation in DI ecosystems. For the

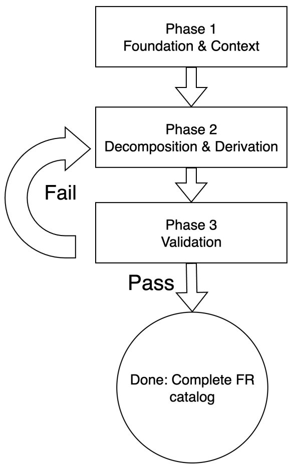

Figure 3.1: Derivation process

whole derivation process, a simplified and adapted softgoal Interdependency Graph (SIG) methodology, as proposed by Chung et al. [6], is used. As illustrated in Figure 3.8, the new graph applies the 3-phase methodology proposed to help in systematically deriving FRs from NFR, but reduces complexity by removing irrelevant components (e.g., label links) and keeping components focusing on the goal: FR derivation.

The ”trust-triangle” model (Issuer, Holder, Verifier), defined in the W3C Verifiable Credentials Data Model [60], provides the component-centric basis for analysis. This model is selected because it’s widely accepted in the SSI community as it delineates the core roles and interactions within a decentralized identity ecosystem, making it a natural fit for component-based requirements analysis and responsibility assignment.

3.1.1 Phase 1: Foundation and Context Setting

This initial phase focuses on establishing a solid understanding of the NFRs, system components, and general process flow in the SSI system.

3.1.1.1 Step 1: SSI NFR, system components, and general process understanding

This consolidated step analyzes the SSI principles by literature review and decides a final list, with an understanding of necessary system components and general process in the SSI system, providing a comprehensive foundation for subsequent analysis.

Literature Review and Principles Analysis

The step commences with a comprehensive literature review examining foundational SSI literature [1, 3], DI standards (W3C [59], DIF [9]), and academic research [7, 13, 41] to identify key SSI principles and their interpretations in the DI/SSI domain. Among these works, Cucko et al. [7] provide a comprehensive and systematic analysis of SSI properties through an iterative methodology that analyzed and grouped properties from an extensive literature review.

This research adopt Cucko et al.'s comprehensive classification and list (Table 3.1) as the definitive set of SSI Non-Functional Requirements (NFRs) because: (1) it represents a systematic and thorough analysis of SSI properties available in current literature, (2) it provides empirically validated definitions that eliminate ambiguity, (3) it includes practitioner-validated importance rankings that enable prioritization, and (4) it covers the complete spectrum of SSI concerns from technical security to user experience and ecosystem sustainability. The properties and their definitions are used as the foundational NFRs for our derivation methodology, ensuring comprehensive coverage of SSI system requirements.

Table 3.1: SSI Properties and Definitions(Source: [7]

No.	**Property**	**Definition**
1	Existence and Representation	Entities must have an independent existence. They should be able to create as many identities as required without the intervention of a third party.
2	Decentralization and Autonomy	Entities must have autonomy over their identity data without relying on any third party (centralized system). They should be responsible for managing all operations related to their identity and data (creating, storing, updating, sharing, revoking).
3	Ownership and Control	Entities must own and control their digital identities and the involved data (e.g., self-asserted claims or claims provided by third parties, identifiers, encryption keys). They should be able to control the usage/sharing of their identity data and delegate control to autonomous agents and/or guardians of their choice.
4	Privacy and Minimal Disclosure	Entities should be able to protect their privacy by utilizing selective disclosure and data minimization. They should be able to disclose the minimum amount of identity data required for any particular interaction.
5	Single source	Entities should be the single source of truth regarding their identities. They should be able to create self-asserted claims, accumulate claims from third parties, and distribute them when required. Third parties should not be able to exchange entities' data without their knowledge and consent.
6	Consent	Entities should be able to give deliberate and well-understood consent for the usage/sharing of their identity data (e.g., consenting to accept data related to their identity).
7	Security and Protection	Digital identity should be secure and well protected with reliable cryptographic mechanisms. Entities must be authenticated and authorized properly prior, to be able to use their digital identity. Any identity information must be transmitted/transferred via a secure channel to prevent cyber attacks.

Table 3.1: SSI Properties and Definitions(Source: [7]

No.	Property	Definition
8	Verifiability and Authenticity	Entities must be able to prove their identity reliably. They must provide verifiable proof of authenticity of digital identity data. Relying parties should be able to verify that digital identities are controlled by their owners and have not been tampered with.
9	Accessibility and Availability	Entities must have unrestricted access to their identity information. They must be able to retrieve claims and assertions (self-asserted or provided by a third party) that constitute their identity, and must be accessible and available from different platforms when required.
10	Recoverability	The identity must be robust enough to be recoverable.
11	Usability and User Experience	The usability of agents and other identity system components should be maximized. User interfaces should allow entities to control, manage, and use their identities intuitively, reliably, and effectively. It should offer a consistent user experience, hide underlying complexity, and should be easy to use.
12	Transparency	The identity system and algorithms must be transparent enough for every involved entity. They should be free, open-source, well-known, and independent of any particular architecture. Entities should be well aware of all their partial identities and their corresponding interactions.
13	Standard	Identities must be based on open Standards to ensure maximal portability, interoperability, and persistence. Entities should be represented, exchanged, secured, protected, and verified using open, public, and royalty-free Standards.
14	Persistence	Identities must be persistent, and should exist for at least as long as it is required by their owner. Longevity and the dynamic nature require firm separation between identity and its claims that can be modified or removed as appropriate.
15	Portability	Identities must be portable. Entities should be able to move or transfer their identity data to agents or systems of their choice securely. Portability ensures entities' control over their data and improves persistence over time.
16	Interoperability	Identities must be as widely usable/available as possible, and not limited to a specific domain. Global identities might increase persistence and identity autonomy.
17	Compatibility with legacy systems	Identity should be backward compatible with legacy identity systems to ensure quicker acceptance.
18	Cost	The cost of identity creation, management, and adoption should be minimized.

System Component Definition: Based on the W3C specification [60] and the survey executed by Alexander et al.[28] and Cucko et al.[7], the core architectural components are established as the foundation for responsibility assignment:

Issuer: An entity that attests to certain attributes of the identity holder by issuing and digitally signing, i.e., a Verifiable Credential (VC) that contains one or more attributes or claims describing the identity holder[7].

Holder: An entity that is responsible for obtaining, storing, managing, controlling, and sharing its identity data/attributes with other entities[7].

Verifier: An entity that requests proof of identity from the identity holder to identify and verify the holder's identity and provide him/her with a (digital) service or product[7].

Use case of SSI

To enable systematic functional requirements derivation, this research adopts the Operational Reference Model (ORM) framework used by Nokhbeh et al. that organizes SSI use case into seven functional groups based on analysis of existing solutions and standards[31]:

1. Issuer Discovery
2. Connection Creation
3. Credential Creation
   (a) Credentials proposed by verifier
   (b) Hierarchical identity (e.g., nationality, state, etc.)
4. Verification with Credentials
   (a) Verification of business good
   (b) Online identity verification initiated by the verifier
   (c) Optimized selection of identities
   (d) "Identity payments" via contact-less technologies
   (e) IoT devices verify on behalf of users
   (f) Issuing tickets to each person only once
5. Backup/Recovery
   (a) Recover from permanently lost wallet, also if the user wants to change devices
   (b) Recover from temporarily lost wallet
6. Derive/Share Credentials
   (a) Cross platform user driven sharing of personal data
   (b) Derive new credentials from existing credentials
   (c) Third party app authenticates through wallet
7. Sunset/Delete/Revoke Credentials
   (a) Setting expiration dates for identity credentials

(b) Revoke credentials from third party app
(c) Temporarily deactivate an identity
(d) Delete credentials by identity owner
(e) Revoke identity proof by issuer

General Process flow

Based on the analysis presented by Cucko et al. [7], SSI interactions follow a generalizable process flow derived from observing SSI use cases across various domains. Their analysis reveals that the process can be generalized, and properties can be connected to specific steps in the process flow. The full details are explained in section 3.2.

Following Cucko et al.'s framework [7], the SSI process can be decomposed into four core stages: (i) Identifier (DID) generation [59], (ii) Acquisition of verifiable credentials (VCs) from identity issuers [60], (iii) Storage of VCs in digital wallets, and (iv) Interaction with verifiers through verifiable presentations (VPs). Figure 3.2 illustrates this general SSI process flow and the properties connected to specific process steps as established by Cucko et al.

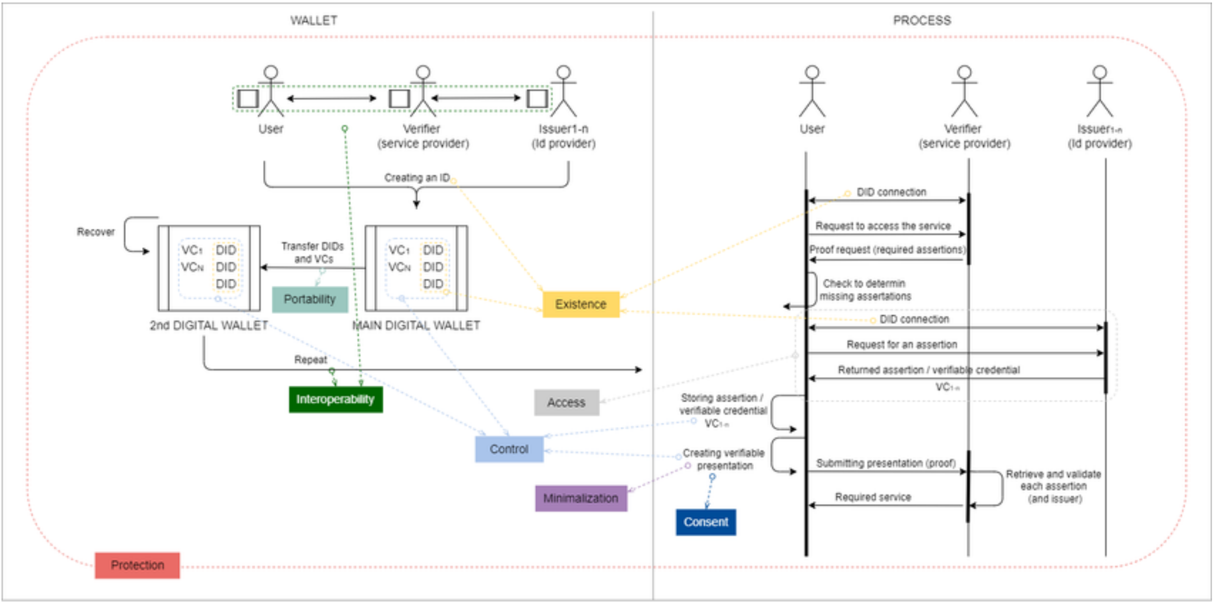
WALLET
PROCESS
User
Verifier (service provider)
Issuer1-n (Id provider)
Creating an ID
Recover
VC1 DID
VCN DID
DID
Transfer DIDs and VCs
Portability
2nd DIGITAL WALLET
MAIN DIGITAL WALLET
Existence
Repeat
Interoperability
Access
Control
Minimalization
Protection
User
Verifier (service provider)
Issuer1-n (Id provider)
DID connection
Request to access the service
Proof request (required assertions)
Check to determin missing assertations
DID connection
Request for an assertion
Returned assertion / verifiable credential
VC1-n
Storing assertion / verifiable credential VC1-n
Creating verifiable presentation
Consent
Submitting presentation (proof)
Retrieve and validate each assertion (and issuer)
Required service

Figure 3.2: The general SSI process flow and connected properties (Source: Cucko et al. [7])

Each interaction requires establishing a pairwise, secure DID connection between interacting parties. The process involves three primary actors within the trust triangle: the identity holder (user), the issuer (credential provider), and the verifier (service provider). Understanding how these actors interact across different process stages is crucial for systematic component responsibility assignment.

While Cucko et al. provide a comprehensive overview of this generalized process flow, Preukschat et al introduced more detailed information about the architecture of SSI and technical, legal, and governance concepts behind SSI [33].

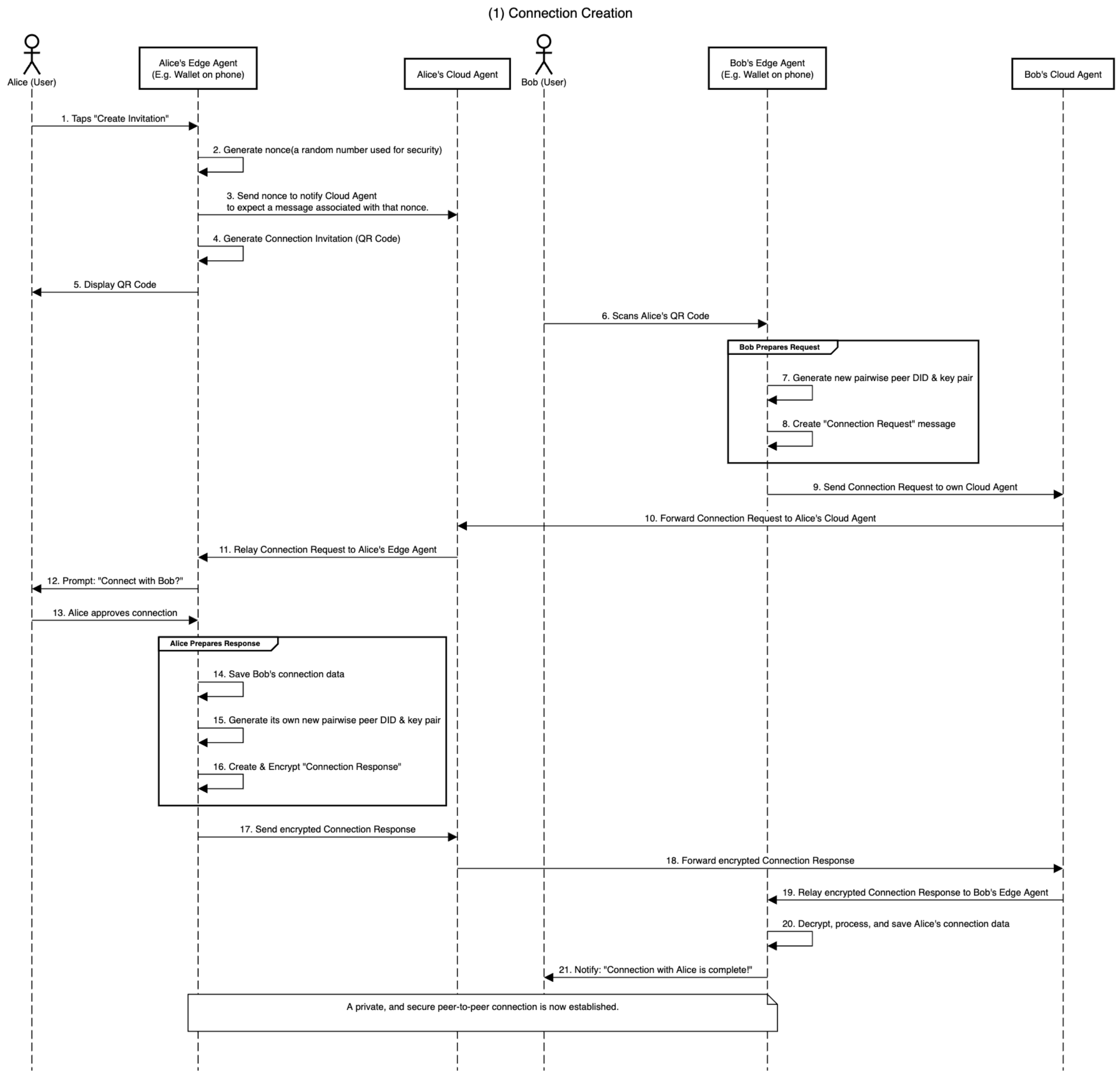
(1) Connection Creation
Alice (User)
Alice's Edge Agent (E.g. Wallet on phone)
Alice's Cloud Agent
Bob (User)
Bob's Edge Agent (E.g. Wallet on phone)
Bob's Cloud Agent
1. Taps "Create Invitation"
2. Generate nonce(a random number used for security)
3. Send nonce to notify Cloud Agent to expect a message associated with that nonce.
4. Generate Connection Invitation (QR Code)
5. Display QR Code
6. Scans Alice's QR Code
Bob Prepares Request
7. Generate new pairwise peer DID & key pair
8. Create "Connection Request" message
9. Send Connection Request to own Cloud Agent
10. Forward Connection Request to Alice's Cloud Agent
11. Relay Connection Request to Alice's Edge Agent
12. Prompt: "Connect with Bob?"
13. Alice approves connection
Alice Prepares Response
14. Save Bob's connection data
15. Generate its own new pairwise peer DID & key pair
16. Create & Encrypt "Connection Response"
17. Send encrypted Connection Response
18. Forward encrypted Connection Response
19. Relay encrypted Connection Response to Bob's Edge Agent
20. Decrypt, process, and save Alice's connection data
21. Notify: "Connection with Alice is complete!"
A private, and secure peer-to-peer connection is now established.

Figure 3.3: Use case #1: Connection creation

After the literature review, and for better analysis of the process flow, the final use cases list is decided: Connection Creation, Credential Creation, Credentials Verification, Backup/Recovery, and Delete/Revoke Credentials. The new list eliminates Redundancy and clarifies the scope and follows a more logical and natural lifecycle.

The foundation for any interaction is the creation of a secure connection, as shown in Figure 3.3. The connection creation involves one party creating an invitation, and then another party responding to it. To ensure the security of the data transferred, a secure communication channel is built.

Once the secure connection is established, the Holder can start the application for credentials, in which the Holder sends a request to the Verifier, and receives the signed credentials once approved by the Issuer. The Holder then stores the credentials in the wallet. This flow is shown in Figure 3.4.

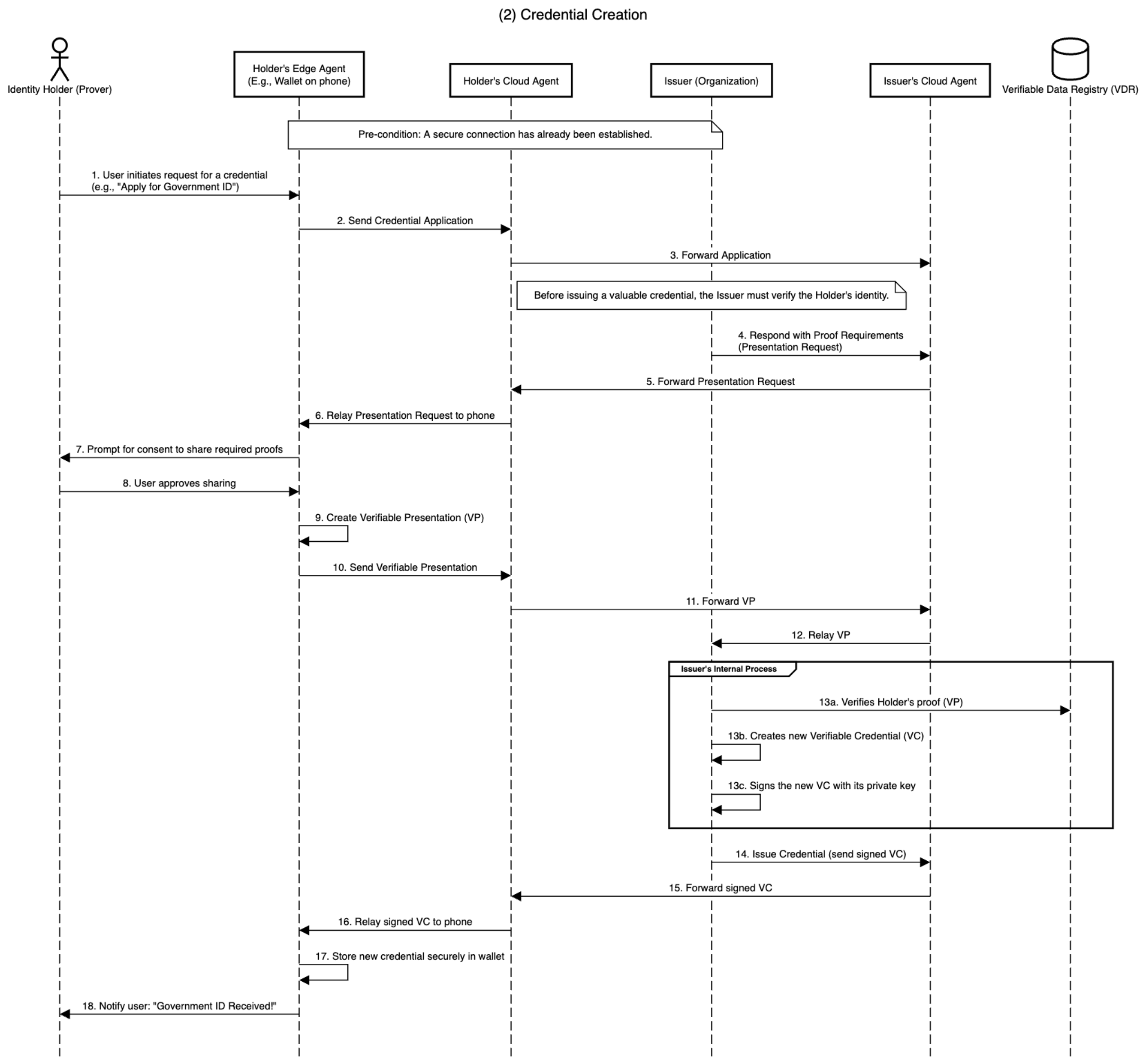
(2) Credential Creation
Identity Holder (Prover)
Holder's Edge Agent (E.g., Wallet on phone)
Holder's Cloud Agent
Issuer (Organization)
Issuer's Cloud Agent
Verifiable Data Registry (VDR)
Pre-condition: A secure connection has already been established.
1. User initiates request for a credential (e.g., "Apply for Government ID")
2. Send Credential Application
3. Forward Application
Before issuing a valuable credential, the Issuer must verify the Holder's identity.
4. Respond with Proof Requirements (Presentation Request)
5. Forward Presentation Request
6. Relay Presentation Request to phone
7. Prompt for consent to share required proofs
8. User approves sharing
9. Create Verifiable Presentation (VP)
10. Send Verifiable Presentation
11. Forward VP
12. Relay VP
Issuer's Internal Process
13a. Verifies Holder's proof (VP)
13b. Creates new Verifiable Credential (VC)
13c. Signs the new VC with its private key
14. Issue Credential (send signed VC)
15. Forward signed VC
16. Relay signed VC to phone
17. Store new credential securely in wallet
18. Notify user: "Government ID Received!"

Figure 3.4: Use case #2: Credential creation

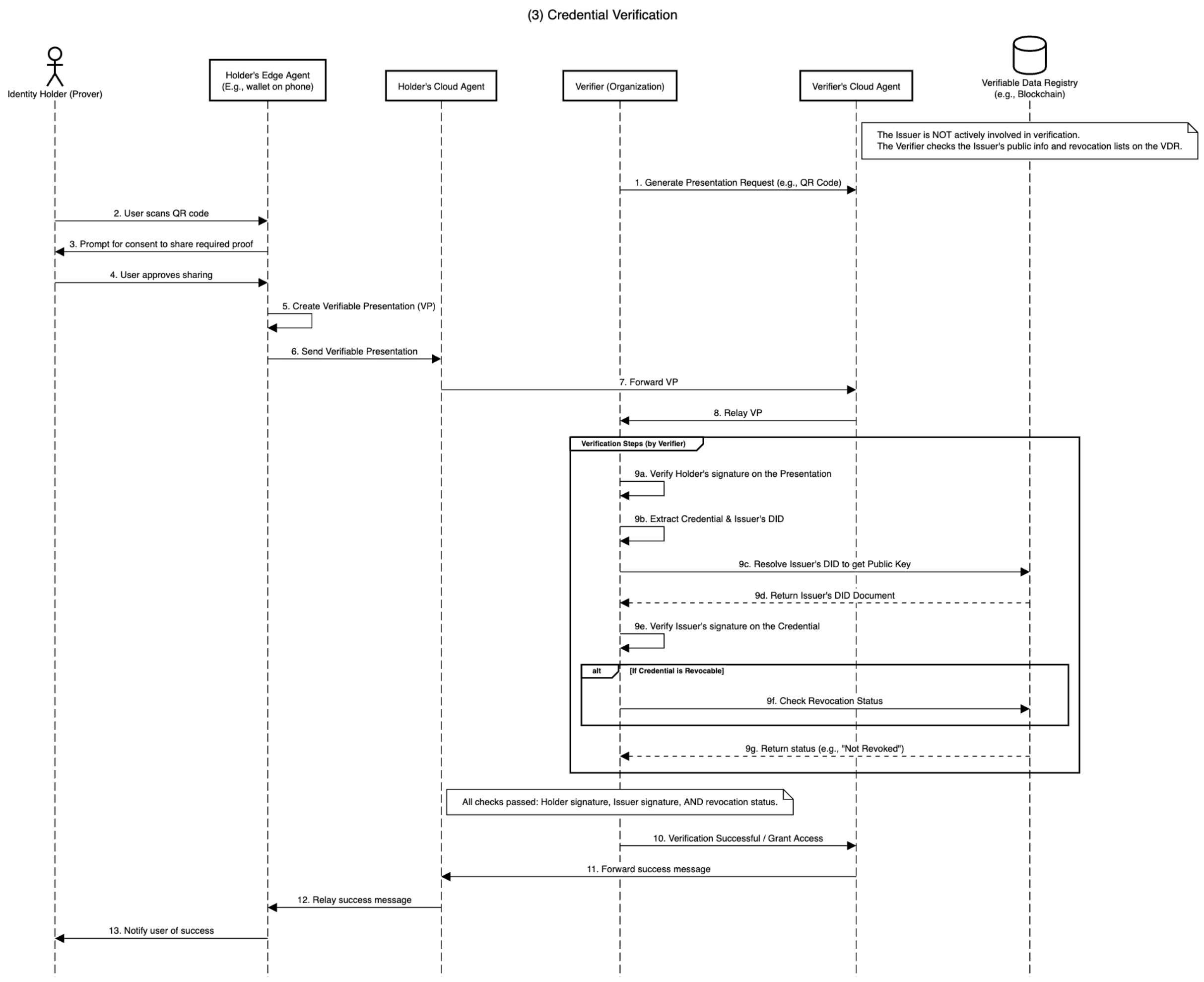
(3) Credential Verification
Identity Holder (Prover)
Holder's Edge Agent (E.g., wallet on phone)
Holder's Cloud Agent
Verifier (Organization)
Verifier's Cloud Agent
Verifiable Data Registry (e.g., Blockchain)
The Issuer is NOT actively involved in verification. The Verifier checks the Issuer's public info and revocation lists on the VDR.
1. Generate Presentation Request (e.g., QR Code)
2. User scans QR code
3. Prompt for consent to share required proof
4. User approves sharing
5. Create Verifiable Presentation (VP)
6. Send Verifiable Presentation
7. Forward VP
8. Relay VP
Verification Steps (by Verifier)
9a. Verify Holder's signature on the Presentation
9b. Extract Credential & Issuer's DID
9c. Resolve Issuer's DID to get Public Key
9d. Return Issuer's DID Document
9e. Verify Issuer's signature on the Credential
alt
[If Credential is Revocable]
9f. Check Revocation Status
9g. Return status (e.g., "Not Revoked")
All checks passed: Holder signature, Issuer signature, AND revocation status.
10. Verification Successful / Grant Access
11. Forward success message
12. Relay success message
13. Notify user of success

Figure 3.5: Use case #3: Credential verification

To gain access to some resources and services, the Holder needs to prove identity to the Verifier, in which the Holder creates a Verifiable Presentation with the credentials in the wallet. To ensure privacy, the Holder can select which attributes to disclose, and the Verifier then checks the signature to confirm its authenticity. As explained in Figure 3.5, the Verifier performs authentication without directly contacting the Issuer of the identity.

SSI also supports backup and recovery to ensure the security of the Holder by providing three recovery methods - offline recovery, social recovery, and multi-device recovery, as shown in Figure 3.6.

Finally, the identity life cycle should include invalidation as well, for which SSI provides two options. As shown in Figure 3.7, the Holder can just delete the credentials in the wallet, or the Issuer can revoke the credential's validity to prevent misuse of the outdated or compromised credentials.

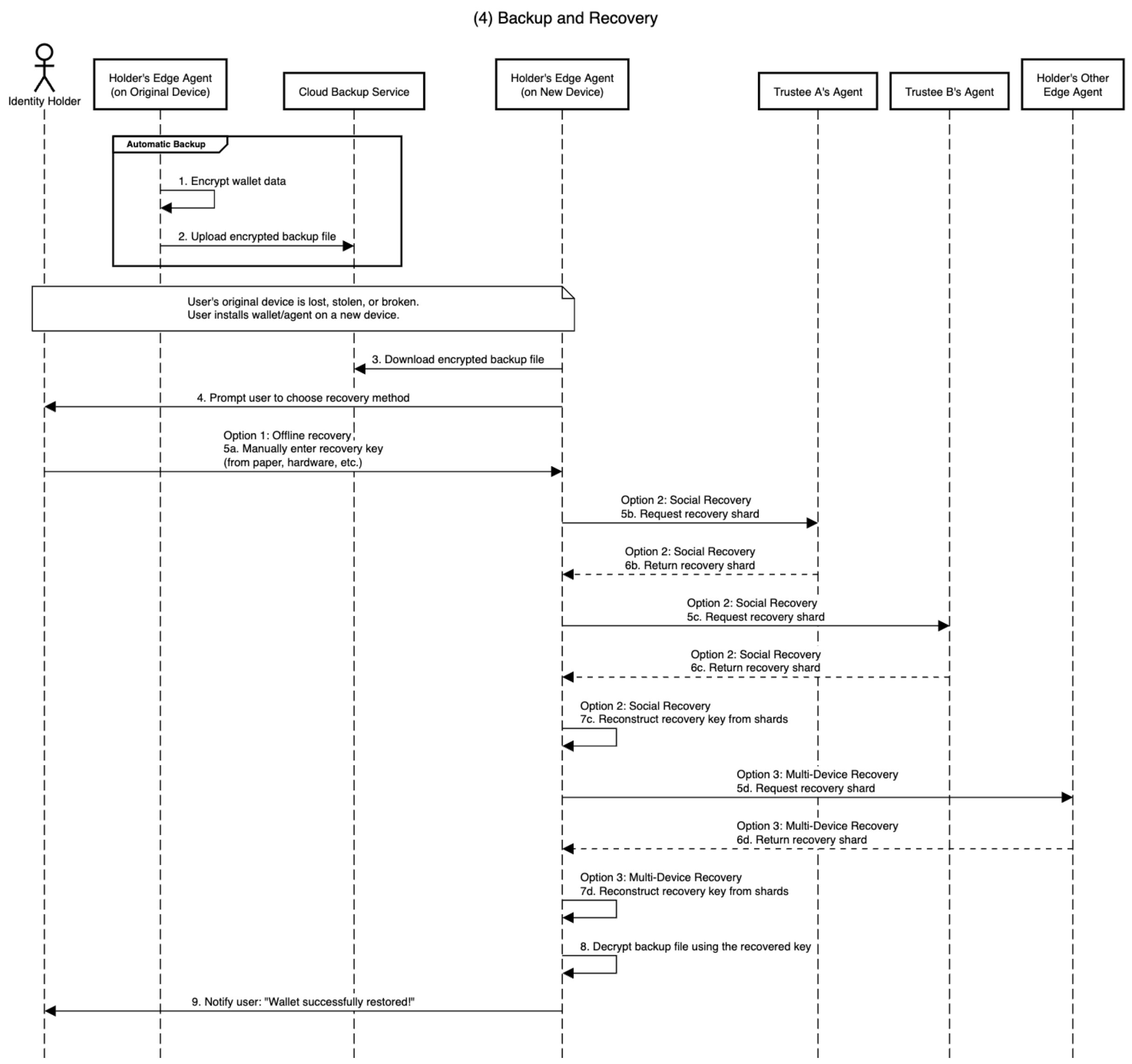
(4) Backup and Recovery
Identity Holder
Holder's Edge Agent (on Original Device)
Cloud Backup Service
Holder's Edge Agent (on New Device)
Trustee A's Agent
Trustee B's Agent
Holder's Other Edge Agent
Automatic Backup
1. Encrypt wallet data
2. Upload encrypted backup file
User's original device is lost, stolen, or broken.
User installs wallet/agent on a new device.
3. Download encrypted backup file
4. Prompt user to choose recovery method
Option 1: Offline recovery
5a. Manually enter recovery key
(from paper, hardware, etc.)
Option 2: Social Recovery
5b. Request recovery shard
Option 2: Social Recovery
6b. Return recovery shard
Option 2: Social Recovery
5c. Request recovery shard
Option 2: Social Recovery
6c. Return recovery shard
Option 2: Social Recovery
7c. Reconstruct recovery key from shards
Option 3: Multi-Device Recovery
5d. Request recovery shard
Option 3: Multi-Device Recovery
6d. Return recovery shard
Option 3: Multi-Device Recovery
7d. Reconstruct recovery key from shards
8. Decrypt backup file using the recovered key
9. Notify user: "Wallet successfully restored!"

Figure 3.6: Use case #4: Backup and recovery

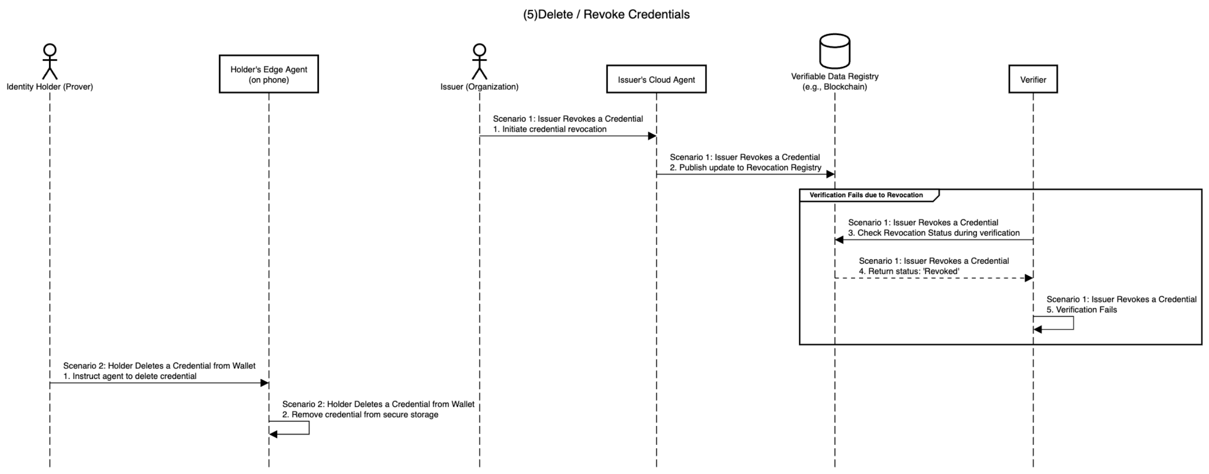
(5)Delete / Revoke Credentials
Identity Holder (Prover)
Holder's Edge Agent (on phone)
Issuer (Organization)
Issuer's Cloud Agent
Verifiable Data Registry (e.g., Blockchain)
Verifier
Scenario 1: Issuer Revokes a Credential
1. Initiate credential revocation
Scenario 1: Issuer Revokes a Credential
2. Publish update to Revocation Registry
Verification Fails due to Revocation
Scenario 1: Issuer Revokes a Credential
3. Check Revocation Status during verification
Scenario 1: Issuer Revokes a Credential
4. Return status: 'Revoked'
Scenario 1: Issuer Revokes a Credential
5. Verification Fails
Scenario 2: Holder Deletes a Credential from Wallet
1. Instruct agent to delete credential
Scenario 2: Holder Deletes a Credential from Wallet
2. Remove credential from secure storage

Figure 3.7: Use case # 5: Delete/Revoke credentials

3.1.2 Phase 2: NFR Decomposition and FR Derivation

This phase involves NFR categorization, NFR-component mapping based on process/scenario and NFR definition, and identifying concrete FRs based on the NFR-component mapping.

3.1.2.1 Step 2: NFR Categorization

NFRs are categorized to guide subsequent analysis and design efforts. NFRs are categorized using the comprehensive SSI property classification developed by Cucko et al. [7], which systematically organizes self-sovereign identity requirements based on extensive literature analysis. The framework identifies five primary categories:

Controllability: Existence and representation, decentralization and autonomy, ownership and control.

Privacy: Privacy and minimal disclosure, single source, consent.

Security: Security and protection, verifiability, and authenticity.

Usability and UX: Accessibility and availability, recoverability, usability, and user experience.

Adoption and Sustainability: Transparency, standards, persistence, portability, interoperability, compatibility with legacy systems, cost.

3.1.2.2 Step 3: Process/Scenarios-component mapping and FR derivation using adapted and simplified Softgoal Interdependency Graphs(SIGs)

Inspired by the methodology established by Chung et al. [6], high-level NFRs are systematically connected with related processes/scenarios. For each related process/scenario, relevant components(Holder, Issuer, Verifier) are identified, finally, FRs are derived based on the scenario and scenario-component mapping, the whole process follows an adapted and simplified Softgoal Interdependency Graph (SIG), as figure 3.8.

The whole derivation process starts from the NFR selected from the NFRs identified in step 1 3.1.1.1, based on the process flow analysis[7, 31, 33], relevant processes/scenarios are identified, based on the definition of the selected NFR and the processes/scenarios identified and the analysis of process flow, relevant NFRs are specified for its parent process/scenario, then the FRs are derived from the process analysis, NFR definition, and relevant component mapped.

For better clarity and testability, the derived functional requirements use the format: the format: [Component] MUST(REQUIRED/SHALL)/MUST NOT(SHALL NOT)/SHOULD (RECOMMEND)/SHOULD NOT(NOT RECOMMEND)/MAY(OPTIONAL) [action/capability]. With the verb from RFC[2], this format ensures clear requirements assignment in the distributed SSI architecture.

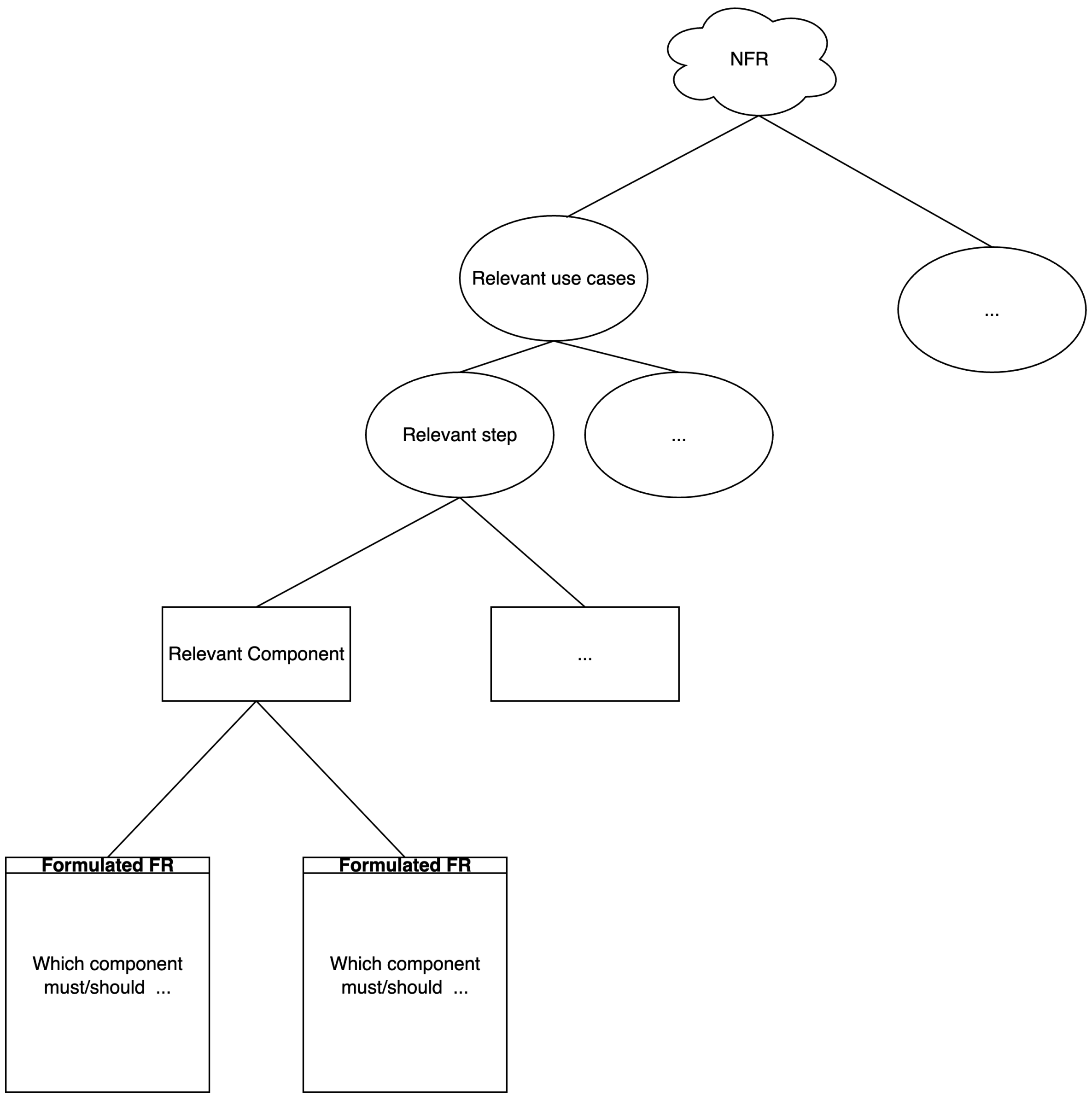

Figure 3.8: Derivation flow

3.1.3 Phase 3: Validation, Refinement, and Documentation

This phase is to ensure the quality and usability of derived FRs by validating FRs and formulating them systematically.

3.1.3.1 Step 4: Validation and Formulation

Validation Framework:

Each derived FR is subjected to systematic assessment against three established criteria:

(i) Clarity assessment examines the degree to which requirements are expressed in unambiguous terms that preclude misinterpretation.

(ii) Testability analysis confirms that functional requirements can be objectively verified through measurable criteria and evaluation methods.

(iii) Component assignment verification ensures that implementation responsibility is clearly designated to appropriate system components.

Completeness evaluation will be executed at the end to determine whether derived functional requirements adequately address their source NFR and associated process/scenario.

FR Formulation:

To ensure clarity, consistency, and verifiability, each FR is systematically documented according to a standardized template. The template comprises the following fields:

- **Unique Identifier:** A structured identifier (formatted as FR-[Component]-[NFR]-[Number]) is assigned for systematic organization and unambiguous referencing.
- **Description:** A precise statement articulates the required function and explicitly delineates the responsible system component.
- **Source Traceability:** A clear statement to the requirement's origin, including the source NFR and scenario/process.
- **Validation Status:** An indicator of the requirement's quality assurance status, confirming the completion of its formal review process.

3.2 Mapping between use case/process, component, and NFR

This section is to map NFR to its relevant process/scenario and then relevant components based on the analysis of the SSI process described in [7, 31, 33]. This is crucial for the derivation, as it indicates the most important question: who should do what?

While a comprehensive list of 18 NFRs has been identified, this thesis provides a detailed, step-by-step analysis for a representative subset of the most foundational principles: “Privacy and Minimal Disclosure” and “Security and Protection” to save space and for brevity. The remaining NFRs, while important, are summarized in a table at the end of this section to provide a complete overview of their mapping to the system’s functions.

Privacy and Minimal Disclosure

Definition: Entities should be able to protect their privacy by utilizing selective disclosure and data minimization. They should be able to disclose the minimum amount of identity data required for any particular interaction [7].

Relevant use cases, steps, and corresponding components:

(i) **Credential Verification (Presentation):** This is the most direct implementation of the NFR, as it governs the actual disclosure of data.

- Step: Create Verifiable Presentation
- Component: Holder
- Description: The Holder’s agent constructs a proof for the Verifier. This process directly supports minimal disclosure by allowing the Holder to share only a specific subset of claims from a credential (selective disclosure) or even just a cryptographic proof about an attribute (e.g., ”is over 18”) without revealing the attribute’s actual value (data minimization).

(ii) **Connection Creation:** This use case establishes the foundation for privacy by preventing correlation of a Holder’s identity across different interactions.

- Step: Generate Pairwise Peer DID
- Component: Holder
- Description: By generating a new, unique Decentralized Identifier (DID) for each relationship, the Holder’s agent ensures that an identifier shared with one party cannot be used to track the Holder’s activities with another party. This is a fundamental mechanism for protecting privacy.

(iii) **Credential Creation (Issuance):** This process is a prerequisite for enabling privacy, as the structure of the credential dictates how it can be shared later.

- Step: Create Verifiable Credential
- Component: Issuer
- Description: To enable the Holder to practice minimal disclosure, the Issuer must structure credentials in a compatible format (e.g., as ”atomic VCs” with single claims, or formats supporting Zero-Knowledge Proofs).

(iv) **Credential Revocation:** The process of checking a credential’s validity must itself be privacy-preserving to avoid leaking data.

- Step: Check Revocation Status

- Components: Issuer, Verifier
- Description: The Issuer must maintain the revocation registry in a way that doesn't publicly leak information about revoked credentials, and the Verifier must be able to check the status without revealing the Holder's identity to the registry.

All relevant components: Holder, Issuer, Verifier

Security and Protection

Definition: Digital identity should be secure and well protected with reliable cryptographic mechanisms. Entities must be authenticated and authorized properly prior, to be able to use their digital identity. Any identity information must be transmitted/transferred via a secure channel to prevent cyber attacks[7].

According to its definition, it can be split as follows:

1. Information must be transferred securely.
2. Entities must be cryptographically verified.
3. Digital assets must be protected by reliable cryptographic mechanisms.

Relevant use cases and components:

(i) **Connection Creation:** This use case is foundational as it establishes the secure channel required for all subsequent interactions.
   - Step: Perform Cryptographic Handshake
   - Components: Holder, Verifier (or any two connecting parties)
   - Description: The parties exchange public keys and perform a cryptographic handshake to establish a mutually authenticated, end-to-end encrypted communication channel. This fulfills the NFR's ”secure channel” requirement, protecting all future data transfers from eavesdropping or man-in-the-middle attacks.

(ii) **Credential Creation (Issuance):** This process ensures that the credential itself is a secure, tamper-proof cryptographic object.
   - Step: Sign Verifiable Credential
   - Component: Issuer
   - Description: The Issuer applies a digital signature to every credential. This provides a “reliable cryptographic mechanism” that guarantees the credential's authenticity (it came from the claimed Issuer) and integrity (it has not been altered).

(iii) **Credential Verification (Presentation):** This is the primary use case where the NFR's "authenticated and authorized properly" clause is enforced.

- Steps: Create Verifiable Presentation (VP), Verify Holder's signature, Verify Issuer's signature, Check Revocation Status.
- Components: Holder, Verifier
- Description: A comprehensive security check is performed. The Verifier cryptographically verifies: 1) the Holder's signature on the presentation to authenticate them, 2) the Issuer's signature on the credential to ensure its integrity, and 3) the credential's status against a revocation registry to ensure it is still valid.

(iv) **Backup and Recovery:** This use case ensures the Holder's core security assets (private keys) are protected at rest.

- Step: Encrypt Wallet Data
- Component: Holder
- Description: The Holder's agent encrypts the wallet backup using user-controlled keys. This protects the Holder's private keys and credentials from being compromised, ensuring the digital identity remains "secure and well protected" even if the storage device or cloud service is breached.

(v) **Credential Revocation:** This provides a critical security function for managing the lifecycle of a credential.

- Step: Publish to Revocation Registry
- Component: Issuer
- Description: The Issuer can officially and publicly invalidate a credential that is no longer valid (e.g., if it was compromised or the conditions have changed). This prevents the fraudulent use of outdated or stolen credentials, securing the ecosystem.

All relevant components: Holder, Issuer, Verifier.

3.2.1 Summary Mapping for Remaining NFRs

The remaining NFRS and corresponding components are specified in table 3.2

3.3 Functional Requirements Prioritization

Rather than establishing arbitrary priority rankings, this thesis builds upon the empirical research conducted by Cucko et al. [7] and the NFR-Component mapping, which systematically surveyed SSI experts to determine the relative importance of different SSI properties. Their comprehensive study involved 32 participants working in fields of Science,

Table 3.2: Mapping of remaining NFRs to Use Cases and Components

NFR	**Relevant Use Case(s)**	**Component(s)**
Existence and Representation	Connection Creation	Holder
Decentralization and Autonomy	Credential Verification	Holder
Ownership and Control	Credential Verification, Backup & Recovery	Holder
Single Source	Credential Verification	Holder
Consent	Credential Verification	Holder
Verifiability and Authenticity	Credential Verification	Verifier, Holder
Accessibility and Availability	Backup and Recovery	Holder
Recoverability	Backup and Recovery	Holder
Usability and User Experience	All Use Cases	Holder
Transparency	Credential Verification	Verifier, Issuer
Standard	All Use Cases	All
Persistence	Connection Creation	Holder, Verifier
Portability	Backup and Recovery	Holder
Interoperability	All Use Cases	All
Compatibility with legacy systems	All Use Cases	All
Cost	All Use Cases	all

Healthcare, Education, Government and Public Service, Business, Sales, Management, Agriculture, and Retail. who evaluated the SSI properties using a 5-level Likert scale ranging from "Not important"(Irrelevant) to "Very important (Mandatory)." After analyzing the survey results, the average importance level score of each NFR is calculated, as Table 3.3 shows. The average importance score is quite reasonable because (i) when executing the questionnaire, the definition of each SSI property is clearly explained, without ambiguity (ii) the participants are carefully selected through various channels(Project, groups, organizations, etc.), and they are all experts in the topic of SSI, rather than just researchers.

As the derivation methodology keeps clear traceability for the derived functional requirements, and all the derived functional requirements contribute together to the corresponding original SSI properties, the root average NFR importance score is used as all the derived functional requirements priority level.

Table 3.3: SSI Principles Importance Scores(Source: Cucko et al. [7])

Property	**AVG**
Security and Protection	4.86
Verifiability and Authenticity	4.79
Privacy and Minimal Disclosure	4.76
Standard	4.59
Consent	4.55
Recoverability	4.52
Ownership and Control	4.48
Portability	4.33
Accessibility and Availability	4.33
Persistence	4.33
Interoperability	4.30
Usability and User Experience	4.30
Transparency	4.26
Existence and Representation	4.25
Decentralization and Autonomy	4.07
Single source	4.00
Cost	3.96
Compatibility with legacy systems	3.67

3.4 The Derived Functional Requirements Catalog

While the proposed NFR-to-FR mapping framework was systematically applied to all 18 SSI properties identified by Cucko et al. [7], this section presents comprehensive derivation examples for two strategically selected NFRs to demonstrate and validate the effectiveness of the proposed seven-step framework.

3.4.1 Complete Derivation Examples

3.4.1.1 Selection of Example NFRs

The selection of NFRs for implementation follows a systematic, empirically-grounded approach based on the NFR-Component mapping in section **??** and the comprehensive classification and importance analysis conducted by Cucko et al. [7]. Their work provides the an categorization of SSI principles and their relative importance according to industry practitioners and researchers.

Category Importance Analysis:

Cucko et al.[7] identified five categories of SSI properties and conducted a survey to determine the importance ratings of 18 individual SSI principles. To establish which categories merit detailed analysis, the average importance score for each category are calculate based on their survey results:

Table 3.4: SSI Category Importance Analysis

Category	Principles Count	Total Score	Average Score
Controllability	3	12.8	4.27
Privacy	3	13.31	4.44
Security	2	9.65	4.83
Usability and UX	3	13.15	4.38
Adoption and Sustainability	7	29.44	4.21

Based on this quantitative analysis, the two highest-priority categories are Security with an average score of 4.83, Privacy with 4.44. These categories represent the most critical aspects of SSI systems according to domain experts and practitioners, encompassing the fundamental requirements for trustworthy, privacy-preserving decentralized identity solutions.

The Security category achieves the highest average importance score, reflecting the critical nature of cryptographic integrity and protection mechanisms in establishing trust within decentralized systems. Privacy follows as the second most important category, emphasizing the core SSI principle of user control over personal information disclosure.

Roles and Importance of Properties for Different Roles: An entity might play different roles (Holder, Verifier, Issuer) in different situations, and for each different role, the priority of SSI properties changes. Cucko et al. [7] analyzed the importance of properties for the roles, as presented in Table 3.5.

Table 3.5: The Importance of Properties for Different Roles (Source: Cucko et al. [7])

Roles	Properties / NFRs
Holder	Existence and Representation, Decentralization and Autonomy, Ownership and Control, Single Source, Accessibility and Availability, Recoverability, Persistence, Portability, Interoperability
Holder, Issuer, Verifier	Privacy and Minimal Disclosure, Consent, Security and Protection, Usability and User Experience, Transparency, Cost
Issuer, Verifier	Verifiability and Authenticity, Compatibility with legacy systems

Selection and Rationale: Within each of the top two categories, the highest-scoring individual properties are selected according to Cucko et al.'s survey results [7] to ensure that our implementation addresses the most critical aspects of each domain. The complete scoring data for each category is presented in Tables 3.6, 3.7.

Furthermore, the NFR-Component mapping in section ref subsec: mapping$_nfr_comprevealsbothsingle$
$componentandcross-componentcharacteristicsthatnecessitatedifferentanalyticalapproaches.The$
$componentrequirementsnecessitatingcoordinatedimplementationacrossallthreetrusttrianglecomp$

Based on these considerations and the empirical importance rankings, two NFRs were selected for comprehensive derivation examples to provide optimal coverage of SSI property

Table 3.6: Security Category SSI Principles and Importance Scores

SSI Principle	Importance Score	Ranking
Security and Protection	4.86	1st
Verifiability and Authenticity	4.79	2nd

Table 3.7: Privacy Category SSI Principles and Importance Scores

SSI Principle	Importance Score	Ranking
Privacy and Minimal Disclosure	4.76	1st
Consent	4.55	2nd
Single Source	4.00	3rd

characteristics. ”Security and Protection” was chosen as the primary cross-component exemplar, with the highest overall importance score (4.86), ensuring demonstration of cryptographic security and protection mechanisms against various attack vectors. ”Privacy and Minimal Disclosure” was selected as the secondary cross-component example (4.76), illustrating how users can share only necessary information for specific purposes through coordinated system-wide privacy-preserving protocols.

This selection strategy ensures comprehensive methodology validation across different NFR characteristics, including system-wide security coordination and privacy-preserving interactions, thereby demonstrating the framework’s applicability to the spectrum of SSI requirements.

3.4.1.2 Derivation for NFR: Privacy and Minimal Disclosure

Based on the analysis of use cases and component mapping, derived FRs for “Privacy and Minimal Disclosure” are as follows:

(i) Credential Verification (Presentation)

- Create Verifiable Presentation : [Holder] should be able to create a Verifiable Presentation that discloses only the specific claims requested by the Verifier, without revealing other claims contained within the same credential (Selective Disclosure).
- Create Verifiable Presentation (Advanced): [Holder] should be able to generate a proof that an attribute satisfies a specific condition (e.g., ’age is over 18’) without disclosing the actual value of the attribute (Data Minimization via Zero-Knowledge Proofs).

(ii) Connection Creation

- Generate new pairwise peer DID and key pair: [Holder] should generate a new, unique pairwise Decentralized Identifier (DID) for each new connection, ensuring that the identifier used with one Verifier cannot be used to track or correlate the Holder’s activity with another Verifier.

(iii) Credential Creation (Issuance)

- Creates new Verifiable Credential: [Issuer] should structure and issue credentials in a format that technically enables future selective disclosure and data minimization by the Holder.

(iv) Delete / Revoke Credentials

- Publish update to Revocation Registry: [Issuer] should publish revocation information using a privacy-preserving mechanism (e.g., cryptographic accumulators) that does not publicly expose lists of revoked credential identifiers.
- Check Revocation Status during verification: [Verifier] should be able to check the revocation status of a credential without revealing the full credential or the Holder's identity to the public revocation registry.

3.4.1.3 Derivation for NFR: Security and Protection

Based on the analysis of use cases and component mapping, derived FRs for "Security and Protection" are as follows:

(i) Credential Verification

- Create Verifiable Presentation (VP): [Holder] should apply a digital signature to the Verifiable Presentation to authenticate themselves as the presenter and authorize the specific disclosure of information.
- Verify Holder's signature: [Verifier] should verify the Holder's signature on a Verifiable Presentation to authenticate that the Holder is in control of the identity and has authorized this specific interaction.
- Verify Issuer's signature: [Verifier] should verify the Issuer's signature on a presented credential to confirm its authenticity and ensure it has not been tampered with.
- Check Revocation Status: [Verifier] should check a trusted revocation registry to ensure the presented credential is still valid and has not been revoked by the Issuer.

(ii) Connection Creation

- Create and Encrypt "Connection Response": [Holder] should perform a cryptographic handshake to establish a mutually authenticated and end-to-end encrypted communication channel for all subsequent interactions.

(iii) Credential Creation

- Signs the new VC with its private key: [Issuer] should apply a digital signature to every verifiable credential it creates, ensuring its authenticity and integrity can be cryptographically verified throughout its lifecycle.

(iv) Backup and Recovery

- Encrypt wallet data: [Holder] should encrypt all wallet data at rest, including private keys, using strong, user-controlled keys to protect it from unauthorized access.

(v) Credential Revocation

- Publish update to Revocation Registry: [Issuer] should be able to update a trusted and secure registry to officially revoke a credential, preventing its fraudulent use after it is no longer valid.

3.5 Validation and Formulation

This section provides a systematic validation and formal documentation of the Functional Requirements (FRs) derived from the Non-Functional Requirements (NFRs) analyzed in the previous section. Each derived FR is subjected to a quality assurance process assessing its clarity, testability, and component assignment, and is then documented using a standardized template for traceability and reference.

3.5.1 NFR: Privacy and Minimal Disclosure

The following FRs are derived to satisfy the "Privacy and Minimal Disclosure" NFR. Each has been subjected to the validation framework.

FR-HOLDER-PRIVACY-01

- **Description:** The Holder component must be able to create a Verifiable Presentation that discloses only the specific claims requested by a Verifier, without revealing other claims contained within the same credential (Selective Disclosure).
- **Source Traceability:** NFR: Privacy and Minimal Disclosure. Use Case: Credential Verification (Presentation).
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement clearly distinguishes between requested and unrequested claims from a single source credential.
  - Testability: Verifiable. A test can create a credential with multiple claims (e.g., name, age, nationality), request proof for a subset (e.g., name and nationality), and assert that the resulting presentation does not contain the omitted claim (age).
  - Component Assignment: Correct. The Holder's agent is responsible for constructing the presentation.

FR-HOLDER-PRIVACY-02

- **Description:** The Holder component must be able to generate a cryptographic proof that an attribute satisfies a specific condition (e.g., ‘age $>$ 18‘) without disclosing the actual value of the attribute (Data Minimization via Zero-Knowledge Proofs).
- **Source Traceability:** NFR: Privacy and Minimal Disclosure. Use Case: Credential Verification (Presentation).
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement clearly states that a condition must be proven without revealing the underlying data value.
  - Testability: Verifiable. A test can use a credential containing a specific date of birth, request proof for the predicate ‘age $>$ 18‘, and assert that the resulting proof is cryptographically valid while also asserting that the date of birth value is not present in the proof payload.
  - Component Assignment: Correct. The Holder’s agent is responsible for generating proofs.

FR-HOLDER-PRIVACY-03

- **Description:** The Holder component must generate a new, unique pairwise Decentralized Identifier (DID) for each new connection, ensuring that the identifier used with one Verifier cannot be used to track or correlate the Holder’s activity with another Verifier.
- **Source Traceability:** NFR: Privacy and Minimal Disclosure. Use Case: Connection Creation.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The terms ”new,” ”unique,” and ”for each connection” are precise and well-defined.
  - Testability: Verifiable. A test can establish a connection with Party A, record the Holder’s DID, establish a second connection with Party B, record the new DID, and assert that the two DIDs are not identical.
  - Component Assignment: Correct. The Holder’s agent is responsible for managing DIDs for its connections.

FR-ISSUER-PRIVACY-01

- **Description:** The Issuer component must structure and issue credentials in a format that technically enables future selective disclosure and data minimization by the Holder.

- **Source Traceability:** NFR: Privacy and Minimal Disclosure. Use Case: Credential Creation (Issuance).
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies that the format must *technically enable* these privacy features, which is a clear directive for the Issuer's implementation.
  - Testability: Verifiable. A test can define a privacy-preserving schema (e.g., an atomic credential schema) and require the Issuer to successfully create and issue a credential based on it, which can then be used to satisfy the test cases for FR-HOLDER-PRIVACY-01 and -02.
  - Component Assignment: Correct. The Issuer is solely responsible for the structure of the credentials it creates.

FR-ISSUER-PRIVACY-02

- **Description:** The Issuer component must publish revocation information using a privacy-preserving mechanism (e.g., cryptographic accumulators) that does not publicly expose lists of revoked credential identifiers.
- **Source Traceability:** NFR: Privacy and Minimal Disclosure. Use Case: Delete / Revoke Credentials.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement explicitly forbids the public exposure of a list of revoked identifiers.
  - Testability: Verifiable. A test can revoke a credential, inspect the data written to the public registry, and assert that the credential's unique identifier is not present in plaintext or any other correlatable form.
  - Component Assignment: Correct. The Issuer is responsible for managing its revocation lists.

FR-VERIFIER-PRIVACY-01

- Description: The Verifier component must be able to check the revocation status of a credential without revealing the full credential or the Holder's identity to the public revocation registry.
- **Source Traceability:** NFR: Privacy and Minimal Disclosure. Use Case: Delete / Revoke Credentials.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement clearly states what information must *not* be revealed during the check.

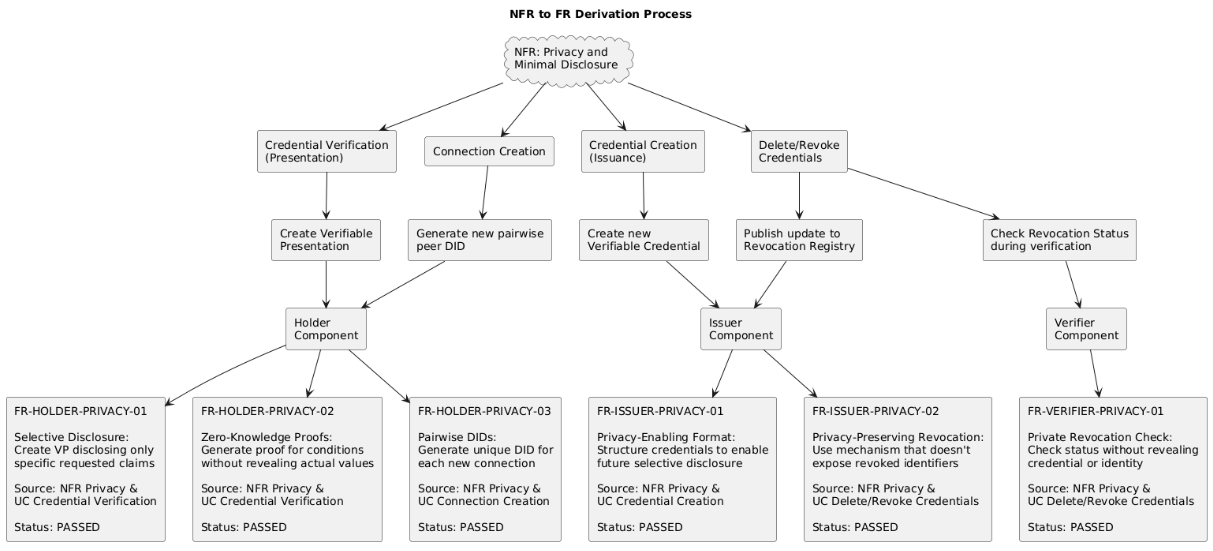
NFR to FR Derivation Process
NFR: Privacy and Minimal Disclosure
Credential Verification (Presentation)
Connection Creation
Credential Creation (Issuance)
Delete/Revoke Credentials
Create Verifiable Presentation
Generate new pairwise peer DID
Create new Verifiable Credential
Publish update to Revocation Registry
Check Revocation Status during verification
Holder Component
Issuer Component
Verifier Component
FR-HOLDER-PRIVACY-01 Selective Disclosure: Create VP disclosing only specific requested claims Source: NFR Privacy & UC Credential Verification Status: PASSED
FR-HOLDER-PRIVACY-02 Zero-Knowledge Proofs: Generate proof for conditions without revealing actual values Source: NFR Privacy & UC Credential Verification Status: PASSED
FR-HOLDER-PRIVACY-03 Pairwise DIDs: Generate unique DID for each new connection Source: NFR Privacy & UC Connection Creation Status: PASSED
FR-ISSUER-PRIVACY-01 Privacy-Enabling Format: Structure credentials to enable future selective disclosure Source: NFR Privacy & UC Credential Creation Status: PASSED
FR-ISSUER-PRIVACY-02 Privacy-Preserving Revocation: Use mechanism that doesn't expose revoked identifiers Source: NFR Privacy & UC Delete/Revoke Credentials Status: PASSED
FR-VERIFIER-PRIVACY-01 Private Revocation Check: Check status without revealing credential or identity Source: NFR Privacy & UC Delete/Revoke Credentials Status: PASSED

Figure 3.9: Derivation for "Privacy and Minimal Disclosure"

- Testability: Verifiable. A test can monitor the network traffic between the Verifier and the revocation registry during a verification check and assert that the Holder's DID or other personally identifiable information is not part of the query.
- Component Assignment: Correct. The Verifier is responsible for performing the check.

Completeness Evaluation: The set of derived functional requirements provides comprehensive coverage for the "Privacy and Minimal Disclosure" NFR. The requirements address the full lifecycle of privacy concerns: establishing a non-correlatable context for interaction (FR-HOLDER-PRIVACY-03); ensuring credentials are created in a privacy-enabling format (FR-ISSUER-PRIVACY-01); giving the Holder fine-grained control over disclosure during presentation (FR-HOLDER-PRIVACY-01, FR-HOLDER-PRIVACY-02); and ensuring that background system processes like revocation checks do not compromise privacy (FR-ISSUER-PRIVACY-02, FR-VERIFIER-PRIVACY-01). Together, these requirements adequately address the source NFR and its associated processes.

3.5.2 NFR: Security and Protection

The following FRs are derived to satisfy the "Security and Protection" NFR. Each has been subjected to the validation framework.

FR-HOLDER-SECURITY-01

- **Description:** The Holder component must apply a digital signature to every Verifiable Presentation it creates to authenticate itself as the presenter and authorize the specific disclosure of information.

- **Source Traceability:** NFR: Security and Protection. Use Case: Credential Verification.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies the action (apply signature) and the purpose (authenticate and authorize).
  - Testability: Verifiable. A test can generate a presentation and assert that a valid, verifiable digital signature is present in the proof section.
  - Component Assignment: Correct. The Holder is the only entity that can authorize the presentation of their credentials.

FR-VERIFIER-SECURITY-01

- **Description:** The Verifier component must cryptographically verify the Holder's signature on a Verifiable Presentation to authenticate that the Holder controls the identity and has authorized the specific interaction.
- **Source Traceability:** NFR: Security and Protection. Use Case: Credential Verification.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies the cryptographic action (verify signature) and the target (the Holder).
  - Testability: Verifiable. A test can present a validly signed VP and assert that the verification check passes. It can also present an invalidly signed VP (e.g., wrong key, tampered payload) and assert that the check fails.
  - Component Assignment: Correct. The Verifier is responsible for authenticating the party it is interacting with.

FR-VERIFIER-SECURITY-02

- **Description:** The Verifier component must cryptographically verify the Issuer's signature on a presented credential to confirm its authenticity and ensure it has not been tampered with.
- **Source Traceability:** NFR: Security and Protection. Use Case: Credential Verification.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies the target of the verification (Issuer's signature) and the purpose (authenticity and integrity).
  - Testability: Verifiable. A test can present a valid credential and assert the signature check passes. It can also present a credential with a tampered claim or an invalid signature and assert the check fails.

  - Component Assignment: Correct. The Verifier must validate the credentials it receives.

FR-VERIFIER-SECURITY-03

- **Description:** The Verifier component must check a trusted revocation registry to ensure a presented credential is still valid and has not been revoked by the Issuer.
- **Source Traceability:** NFR: Security and Protection. Use Case: Credential Verification.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies a clear action (check registry) and condition (ensure not revoked).
  - Testability: Verifiable. A test can present a revoked credential and assert that the verification process fails. It can also present a non-revoked credential and assert that it passes this specific check.
  - Component Assignment: Correct. The Verifier is responsible for ensuring the revocation status of the credential.

FR-HOLDER-SECURITY-02

- **Description:** The Holder component must perform a cryptographic handshake to establish a mutually authenticated and end-to-end encrypted communication channel for all subsequent interactions.
- **Source Traceability:** NFR: Security and Protection. Use Case: Connection Creation.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies both mutual authentication and end-to-end encryption.
  - Testability: Verifiable. A test can establish a connection and monitor the network traffic, asserting that the payload of subsequent messages is encrypted and unreadable.
  - Component Assignment: Correct. Both connecting parties (of which the Holder is one) are responsible for establishing the secure channel.

FR-ISSUER-SECURITY-01

- **Description:** The Issuer component must apply a digital signature to every verifiable credential it creates, ensuring its authenticity and integrity can be cryptographically verified throughout its lifecycle.

- **Source Traceability:** NFR: Security and Protection. Use Case: Credential Creation.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies the action (apply signature) and the scope (every credential).
  - Testability: Verifiable. A test can request a new credential from an Issuer and assert that the returned object contains a cryptographically valid digital signature from the Issuer.
  - Component Assignment: Correct. The Issuer is the sole entity responsible for signing the credentials it creates.

FR-HOLDER-SECURITY-03

- **Description:** The Holder component must encrypt all wallet data at rest, including private keys, using strong, user-controlled keys to protect it from unauthorized access.
- **Source Traceability:** NFR: Security and Protection. Use Case: Backup and Recovery.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement specifies the data to be protected (all wallet data) and the method (encryption at rest).
  - Testability: Verifiable. A test can inspect the wallet's storage file on the device's file system and assert that its contents are not in plaintext.
  - Component Assignment: Correct. The Holder's agent is responsible for the security of its local wallet.

FR-ISSUER-SECURITY-02

- **Description:** The Issuer component must be able to update a trusted and secure registry to officially revoke a credential, preventing its fraudulent use after it is no longer valid.
- **Source Traceability:** NFR: Security and Protection. Use Case: Credential Revocation.
- **Validation Status:** Passed.
  - Clarity: Unambiguous. The requirement clearly states the action (update registry) and the purpose (prevent fraudulent use).
  - Testability: Verifiable. A test can have the Issuer revoke a credential and then query the public registry to assert that the credential's status is now correctly listed as ”revoked.”

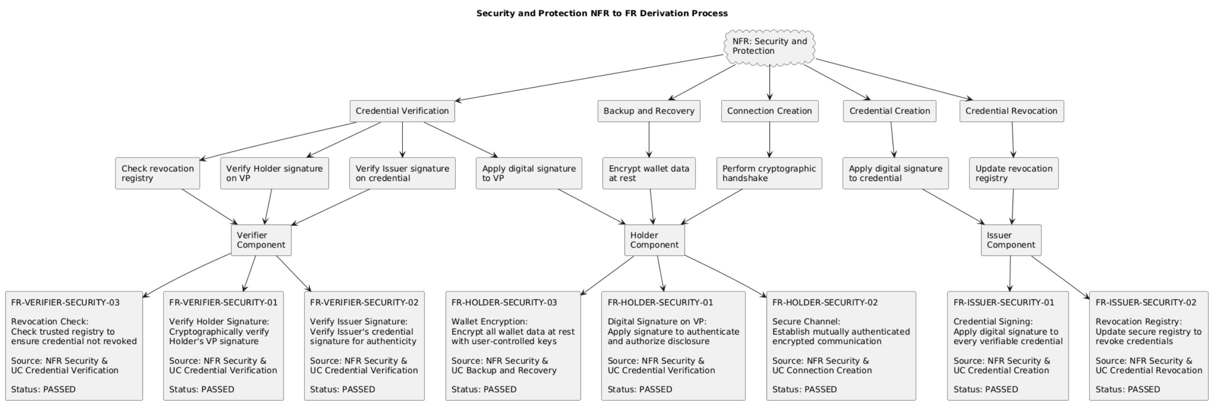
Security and Protection NFR to FR Derivation Process
NFR: Security and Protection
Credential Verification
Backup and Recovery
Connection Creation
Credential Creation
Credential Revocation
Check revocation registry
Verify Holder signature on VP
Verify Issuer signature on credential
Apply digital signature to VP
Encrypt wallet data at rest
Perform cryptographic handshake
Apply digital signature to credential
Update revocation registry
Verifier Component
Holder Component
Issuer Component
FR-VERIFIER-SECURITY-03 Revocation Check: Check trusted registry to ensure credential not revoked Source: NFR Security & UC Credential Verification Status: PASSED
FR-VERIFIER-SECURITY-01 Verify Holder Signature: Cryptographically verify Holder's VP signature Source: NFR Security & UC Credential Verification Status: PASSED
FR-VERIFIER-SECURITY-02 Verify Issuer Signature: Verify Issuer's credential signature for authenticity Source: NFR Security & UC Credential Verification Status: PASSED
FR-HOLDER-SECURITY-03 Wallet Encryption: Encrypt all wallet data at rest with user-controlled keys Source: NFR Security & UC Backup and Recovery Status: PASSED
FR-HOLDER-SECURITY-01 Digital Signature on VP: Apply signature to authenticate and authorize disclosure Source: NFR Security & UC Credential Verification Status: PASSED
FR-HOLDER-SECURITY-02 Secure Channel: Establish mutually authenticated encrypted communication Source: NFR Security & UC Connection Creation Status: PASSED
FR-ISSUER-SECURITY-01 Credential Signing: Apply digital signature to every verifiable credential Source: NFR Security & UC Credential Creation Status: PASSED
FR-ISSUER-SECURITY-02 Revocation Registry: Update secure registry to revoke credentials Source: NFR Security & UC Credential Revocation Status: PASSED

Figure 3.10: Derivation for "Security and Protection"

- Component Assignment: Correct. The Issuer is the only entity with the authority to revoke its own credentials.

Completeness Evaluation: The set of derived functional requirements provides comprehensive coverage for the "Security and Protection" NFR. The requirements address the establishment of secure channels (FR-HOLDER-SECURITY-02), the cryptographic integrity of credentials at creation (FR-ISSUER-SECURITY-01) and during their lifecycle (FR-ISSUER-SECURITY-02), robust multi-factor authentication and authorization during verification (FR-HOLDER-SECURITY-01, FR-VERIFIER-SECURITY-01, -02, -03), and the protection of sensitive cryptographic material at rest (FR-HOLDER-SECURITY-03). Together, these requirements adequately address the source NFR and its associated processes.

3.6 Design of the Evaluation Process

The evaluation process is designed to systematically assess decentralized identity and self-sovereign identity systems through the functional requirements derived using the NFR-to-FR mapping framework presented in the previous sections. This evaluation methodology provides a structured approach to determine how comprehensively SSI systems implement the functional capabilities necessary to satisfy established SSI principles.

3.6.1 Evaluation Framework Overview

The evaluation framework operates on the foundation of the derived functional requirements catalog, applying a counting-based methodology to provide objective and reproducible assessments. The framework transforms the theoretical NFR-to-FR mappings into a practical evaluation tool that can systematically compare different SSI implementations.

The evaluation process follows a systematic approach based on the derived FRs:

1. Application of the derived FR catalog to target SSI systems
2. Multi-source evidence collection for each functional requirement
3. Objective assessment of functional requirement satisfaction
4. Aggregation of results using counting methodology

3.6.2 Functional Requirements Assessment Methodology

For each NFR examined in the derivation process, the framework evaluates the target system's implementation of the corresponding derived functional requirements. The assessment methodology employs a binary evaluation approach where each functional requirement is assessed as either satisfied or not satisfied based on concrete evidence from the target system.

3.6.2.1 Evidence Collection Framework

The evaluation relies on multiple evidence sources to ensure comprehensive assessment of each derived functional requirement: **Evidence Sources:**

- Official technical documentation and system specifications
- Architectural documentation and design principles
- Academic literature and peer-reviewed analysis of the systems
- White papers and technical reports published
- Community documentation and developer resources

3.6.2.2 FR Satisfaction Criteria

A functional requirement is considered satisfied if there is demonstrable evidence of its implementation within the target system. The satisfaction criteria require:

1. **Implementation Evidence**: Concrete proof that the functionality described in the FR exists in the system
2. **Component Assignment Verification**: Confirmation that the functionality is implemented by the appropriate system component as specified in the FR formulation
3. **Functional Completeness**: Evidence that the implementation addresses the full scope of the functional requirement as derived from the source NFR

3.6.2.3 NFR-Level Assessment

For each SSI principle (NFR) that underwent the derivation process, the evaluation:

1. Counts the total number of functional requirements derived for that NFR
2. Assesses each derived functional requirement against the satisfaction criteria
3. Counts the number of satisfied functional requirements
4. Records the results as satisfied/total functional requirements for that NFR

This produces a clear numerical representation of how comprehensively each system implements the functional requirements necessary to satisfy each examined SSI principle.

3.6.2.4 System-Level Aggregation

The overall system evaluation aggregates results across all assessed NFRs:

1. Sum all satisfied functional requirements across all examined NFRs
2. Sum all total derived functional requirements across all examined NFRs
3. Present the overall ratio of satisfied to total functional requirements

This aggregation provides a comprehensive view of the system's functional completeness based on the derived requirements while maintaining transparency about individual NFR implementation levels.

3.7 Case Study Selection

The selection of appropriate case studies is critical for validating the proposed evaluation framework and demonstrating its practical applicability across different SSI system architectures. This section presents the rationale for selecting Sovrin and uPort as the primary case studies for framework validation, based on their representative characteristics, technical accessibility, and established recognition within the SSI domain.

3.7.1 Selection Criteria

The case study selection process employed multiple criteria to ensure that the chosen systems would provide comprehensive validation of the evaluation framework while representing the diversity of approaches within the SSI ecosystem:

Technical Documentation Availability: Systems must provide sufficient technical documentation, specifications, and implementation details to enable thorough evaluation against the derived functional requirements.

Open-source: Open-source implementations are preferred to allow direct verification of functional requirement satisfaction.

Diversity: Systems of different blockchain types are preferred, uPort is built on public permissionless Ethereum, while Sovrin applies public permissioned Hyperledger[25, 30, 41, 44].

This case study evaluation is deliberately focused on the single, strategically selected NFR of "Privacy and Minimal Disclosure" rather than all 18 SSI principles identified by Cucko et al. [7] for the following methodological reasons:

Demonstrative Sufficiency: The primary objective is to validate the evaluation framework's effectiveness and demonstrate its practical application. A detailed analysis of this foundational NFR provides sufficient evidence of the framework's capability to systematically assess SSI systems without requiring exhaustive analysis of all principles.

Research Scope Management: A comprehensive evaluation of all 18 NFRs would require extensive analysis beyond the scope of this thesis. This focused approach provides meaningful validation while maintaining a manageable scope and concentrating on the core contribution: the systematic NFR-to-FR mapping methodology and evaluation framework design.

Scalability Demonstration: By showing a detailed evaluation for the selected NFR, the framework establishes a clear pattern that can be systematically extended to the remaining 17 principles, proving the methodology's scalability and completeness for future research.

Chapter 4

Results

4.1 Results: Case Study 1 - Sovrin

4.1.1 Overview of Sovrin

Sovrin is a public service utility that enables self-sovereign identity on the Internet through a decentralized network architecture [44, 46, 57]. The platform allows individuals to collect, hold, and selectively share identity credentials—such as driver's licenses or employment certificates—without depending on centralized databases that traditionally control access to such credentials [18]. As an open-source project, Sovrin provides the tools and libraries necessary to create private and secure data management solutions that operate on its dedicated identity network [45, 58].

By analyzing existing white papers and academic researches [23, 29, 35, 57, 58], the Sovrin architecture is derived as Figure 4.1

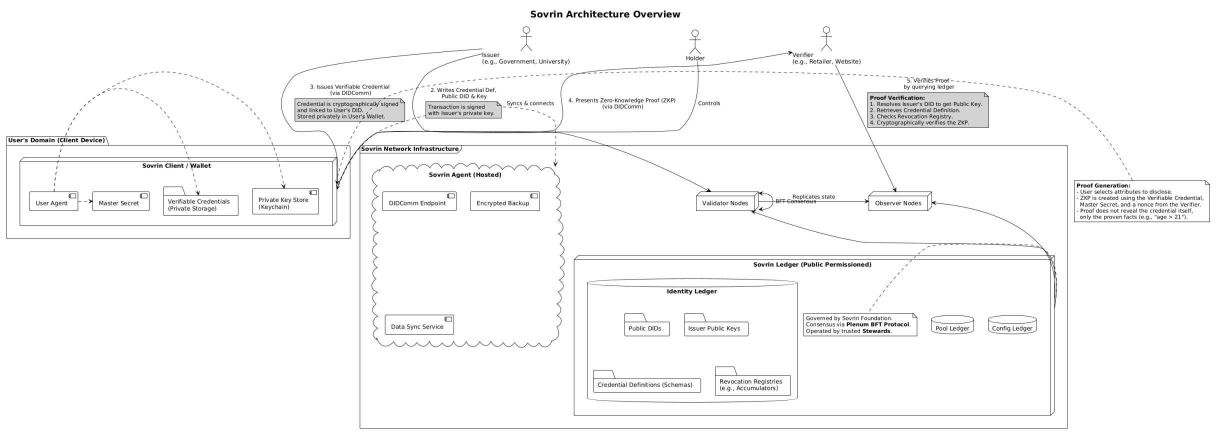
Sovrin Architecture Overview
Issuer (e.g., Government, University)
Holder
Verifier (e.g., Retailer, Website)
3. Issues Verifiable Credential (via DIDComm)
Credential is cryptographically signed and linked to User's DID. Stored privately in User's Wallet.
2. Writes Credential Def, Public DID & Key
Transaction is signed with Issuer's private key.
Syncs & connects
4. Presents Zero-Knowledge Proof (ZKP) (via DIDComm)
Controls
5. Verifies Proof by querying ledger
Proof Verification:
1. Resolves Issuer's DID to get Public Key.
2. Retrieves Credential Definition.
3. Checks Revocation Registry.
4. Cryptographically verifies the ZKP.
User's Domain (Client Device)
Sovrin Client / Wallet
User Agent
Master Secret
Verifiable Credentials (Private Storage)
Private Key Store (Keychain)
Sovrin Network Infrastructure
Sovrin Agent (Hosted)
DIDComm Endpoint
Encrypted Backup
Data Sync Service
Validator Nodes
BFT Consensus
Replicates state
Observer Nodes
Proof Generation:
- User selects attributes to disclose.
- ZKP is created using the Verifiable Credential, Master Secret, and a nonce from the Verifier.
- Proof does not reveal the credential itself, only the proven facts (e.g., "age > 21").
Sovrin Ledger (Public Permissioned)
Identity Ledger
Public DIDs
Issuer Public Keys
Credential Definitions (Schemas)
Revocation Registries (e.g., Accumulators)
Governed by Sovrin Foundation. Consensus via Plenum BFT Protocol. Operated by trusted Stewards.
Pool Ledger
Config Ledger

Figure 4.1: Sovrin Architecture Overview

4.1.2 NFR4 - Privacy and Minimal Disclosure

This subsection evaluates Sovrin's self-sovereign identity system against six functional requirements derived from the "Privacy and Minimal Disclosure" non-functional requirement. The assessment employs an evaluation methodology based on concrete evidence from official documentation, technical specifications, and academic literature.

4.1.2.1 FR-HOLDER-PRIVACY-01: Create Verifiable Presentation with Selective Disclosure

Requirement: [Holder] should be able to create a Verifiable Presentation that discloses only the specific claims requested by the Verifier, without revealing other claims contained within the same credential (Selective Disclosure).

Assessment Result: SATISFIED

Implementation Evidence: Sovrin implements selective disclosure through AnonCreds (Anonymous Credentials) [18, 35, 53]. The system enables holders to create verifiable presentations revealing only specific attributes from credentials.

Technical implementation includes [18]:

- Cryptographic commitments where holders blind secret values before sharing
- Attribute-based selective disclosure where only necessary claims are revealed
- Zero-knowledge proofs to verify possession of unrevealed attributes
- Link secret binding ensuring multiple credentials belong to the same holder without revealing correlated identifiers

Component Assignment Verification: Correctly assigned to the Holder component. Holders control presentation creation through wallet-based proof generation [21].

Functional Completeness: The implementation supports revealing arbitrary subsets of credential attributes while maintaining cryptographic integrity without revealing non-requested attributes, preventing correlation through unlinkable presentations.

4.1.2.2 FR-HOLDER-PRIVACY-02: Create Verifiable Presentation with Zero-Knowledge Predicates

Requirement: [Holder] should be able to generate a proof that an attribute satisfies a specific condition (e.g., 'age is over 18') without disclosing the actual value of the attribute (Data Minimization via Zero-Knowledge Proofs).

Assessment Result: SATISFIED

Implementation Evidence: Sovrin provides predicate proofs enabling holders to prove attribute conditions without revealing actual values [18].

Technical implementation includes [18]:

- CL-signatures foundation with 2048-bit RSA groups using safe primes
- Non-interactive zero-knowledge proofs using Fiat-Shamir transformation

Component Assignment Verification: Correctly assigned to the Holder component. Holders generate zero-knowledge predicate proofs [21].

Functional Completeness: Supports complex mathematical predicates on credential attributes while maintaining unlinkability across multiple presentations through the generation of cryptographically sound proofs without disclosing values.

4.1.2.3 FR-HOLDER-PRIVACY-03: Generate New Pairwise Peer DID and Key Pair

Requirement: [Holder] should generate a new, unique pairwise Decentralized Identifier (DID) for each new connection, ensuring that the identifier used with one Verifier cannot be used to track or correlate the Holder's activity with another Verifier.

Assessment Result: SATISFIED

Implementation Evidence: Sovrin implements pairwise pseudonymous identifiers where unique DIDs are generated for each relationship, preventing cross-context correlation [56].

Technical implementation includes [45]:

- Base58 encoded 16-byte UUID identifiers following did:sov: method specification
- Ed25519 verification keys for authentication and digital signatures
- X25519 key agreement keys for secure communication establishment
- Cryptographically unique identifier generation, preventing accidental reuse across contexts

Component Assignment Verification: Correctly assigned to the Holder component. Holders control DID generation through local wallet operations, with private keys remaining under the Holder's exclusive control.

Functional Completeness: Generates cryptographically unique DIDs for each verifier relationship, prevents identifier reuse across different contexts, and supports unlimited pairwise relationships without correlation risk.

4.1.2.4 FR-ISSUER-PRIVACY-01: Create Verifiable Credentials Supporting Future Selective Disclosure

Requirement: [Issuer] should structure and issue credentials in a format that technically enables future selective disclosure and data minimization by the Holder.

Assessment Result: SATISFIED

Implementation Evidence: Sovrin structures credentials using AnonCreds format that inherently enables selective disclosure and data minimization through cryptographic design [18].

Technical implementation includes [18]:

- Credential Definition structure includes issuer's public keys for CL-signatures
- Blinded link secret scheme during credential issuance preserves holder privacy and prevents correlation
- JSON-based credential format with both raw and encoded attribute values supporting cryptographic operations
- Schema-based credential structure enabling selective attribute disclosure

Component Assignment Verification: Correctly assigned to Issuer component. Issuers create credentials using documented APIs, such as "indy_issuer_create_and_store_credential_def" and "indy_issuer_create_schema", with credential definition publication enabling future verification [17].

Functional Completeness: All issued credentials inherently support selective disclosure, preserve the ability for zero-knowledge proof generation, and maintain compatibility with privacy-preserving presentation protocols.

4.1.2.5 FR-ISSUER-PRIVACY-02: Publish Update to Revocation Registry with Privacy-Preserving Mechanisms

Requirement: [Issuer] should publish revocation information using a privacy-preserving mechanism (e.g., cryptographic accumulators) that does not publicly expose lists of revoked credential identifiers.

Assessment Result: SATISFIED

Implementation Evidence: Sovrin implements cryptographic accumulators for privacy-preserving revocation that don't expose revoked credential identifiers publicly [19].

Technical implementation includes [19]:

- Cryptographic accumulators based on modular arithmetic

- Dynamic accumulator updates without revealing specific revoked credentials
- Tails file mechanism containing massive prime factors for witness generation
- Non-reversible accumulator design using massive prime numbers

Component Assignment Verification: Correctly assigned to Issuer component. Issuers update revocation registries through ledger transactions with accumulator value updates reflecting revocation state without revealing IDs [20].

Functional Completeness: The implementation ensures revocation updates do not expose credential identifiers or enable correlation through credential IDs or tails indices.

4.1.2.6 FR-VERIFIER-PRIVACY-01: Check Revocation Status During Verification

Requirement: [Verifier] should be able to check the revocation status of a credential without revealing the full credential or the Holder's identity to the public revocation registry.

Assessment Result: SATISFIED

Implementation Evidence: Sovrin enables verifiers to check credential revocation status without revealing credential identity or holder information to the revocation registry [19].

Technical implementation includes:

- Non-revocation proofs generated using cryptographic accumulators
- Accumulator membership proofs demonstrating credential validity without revealing identifiers
- Verifier validation against public accumulator values on the ledger
- Privacy-preserving verification requiring no tails file downloads by verifiers

Component Assignment Verification: Correctly assigned to the Verifier component. Verifiers process non-revocation proofs during presentation verification for local verification without exposing holder identity to the registry [20].

Functional Completeness: Revocation status verification preserves holder privacy, transmits no correlatable identifiers to revocation registry, and provides cryptographically sound verification of non-revocation status.

Functional Requirement Category	**Satisfied**	**Total**
Credential Verification (Presentation)	2	2
Connection Creation	1	1
Credential Creation (Issuance)	1	1
Delete/Revoke Credentials	2	2
Total NFR Assessment	**6**	**6**

Table 4.1: Privacy and Minimal Disclosure FR Satisfaction

4.1.2.7 Assessment Summary

Overall Functional Requirements Satisfaction for NFR 4: 6/6 SATISFIED (100%)

Sovrin demonstrates comprehensive implementation of all privacy-focused functional requirements through sophisticated cryptographic mechanisms including complete selective disclosure through CL-signatures and AnonCreds, advanced zero-knowledge capabilities including predicate proofs, privacy-preserving revocation using cryptographic accumulators, and unlinkable pairwise relationships preventing cross-context correlation. All assessments are supported by concrete implementation evidence from official Hyperledger documentation and technical specifications.

4.2 Results: Case Study 2 - uPort

4.2.1 Overview of uPort

UPort is an open-source identity management system built on the Ethereum blockchain. Its purpose is to provide users with Self-Sovereign Identity (SSI), giving them full control over their digital identity and personal data, independent of any central authority [25, 30]. Its technical overview is illustrated in Figure 4.2

A user's identity is represented by a smart contract, which allows for features like passwordless authentication and social key recovery. While the identity exists on the blockchain, sensitive personal data is kept off-chain in a secure "digital wallet" on the user's mobile device. This allows users to selectively share their information with services by simply scanning a QR code and approving the request through the uPort app [25, 30].

4.2.2 NFR4 - Privacy and Minimal Disclosure

This subsection evaluates the uPort self-sovereign identity system and its successor, Veramo, against six functional requirements derived from the "Privacy and Minimal Disclosure" non-functional requirement. The assessment employs an evaluation methodology based on concrete evidence, such as the uPort whitepaper and academic papers.

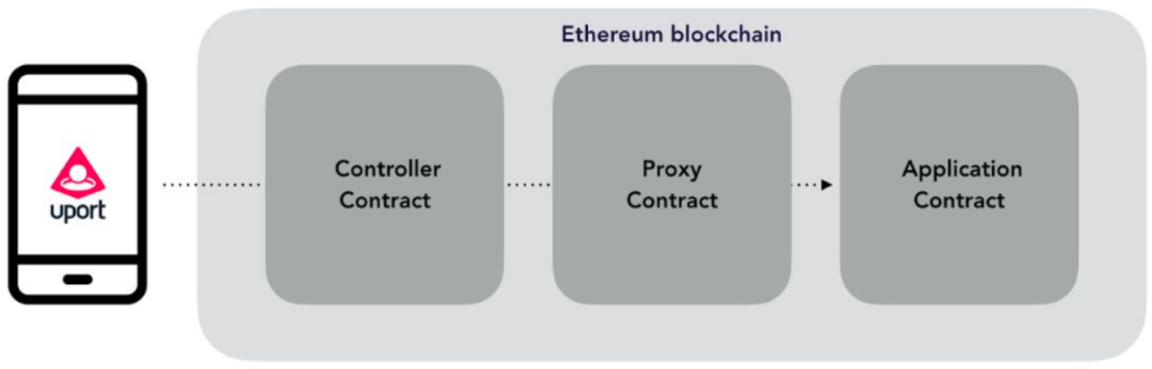

Figure 4.2: uPort Technical Overview(Source: [25]

4.2.2.1 FR-HOLDER-PRIVACY-01: Create Verifiable Presentation with Selective Disclosure

Requirement: [Holder] should be able to create a Verifiable Presentation that discloses only the specific claims requested by the Verifier, without revealing other claims contained within the same credential (Selective Disclosure).

Assessment Result: SATISFIED

Implementation Evidence: The uPort architecture was fundamentally designed to support selective disclosure by structuring identity data as individual, signed attestations that can be selectively shared.

Technical Implementation:

- **Requesting Specific Claims:** The selective disclosure flow explicitly allows a relying party (a "dapp") to request only the claims it needs [50].
- **Granular Attestations as JWTs:** The core technology for credentials is the JSON Web Token (JWT), which encapsulates individual claims [25].
- **Presentation Generation Logic:** The holder's software uses libraries specifically designed to construct presentations containing only the approved claims [52].

Component Assignment Verification: Correctly assigned to the Holder component. The holder's mobile application is the agent responsible for receiving the disclosure request and constructing the presentation.

Functional Completeness: The implementation is functionally complete. The documented flow and underlying library support allow a holder to cryptographically prove a subset of claims from their credentials without revealing any other data from those credentials.

4.2.2.2 FR-HOLDER-PRIVACY-02: Create Verifiable Presentation with Zero-Knowledge Predicates

Requirement: [Holder] should be able to generate a proof that an attribute satisfies a specific condition (e.g., “age is over 18”) without disclosing the actual value of the attribute (Data Minimization via Zero-Knowledge Proofs).

Assessment Result: NOT SATISFIED

Implementation Evidence: Analysis of the provided sources—including the uPort whitepaper, specifications, and library documentation—reveals no mechanism for generating or verifying zero-knowledge predicate proofs. The system’s approach to data minimization is exclusively focused on the selective disclosure of entire attributes.

Technical Implementation:

- **No Mention of Predicate Proofs:** The uPort whitepaper discusses public/private key cryptography and encryption for data sharing, but makes no mention of ZKPs for predicate proofs [25].
- **Focus on Full Attribute Disclosure:** The selective disclosure flow is explicitly about sharing the values of the requested attributes, not proving claims about them[50].

Component Assignment Verification: This requirement is unfulfilled by any component in the documented uPort system.

Functional Completeness: The system is not functionally complete in this regard. It lacks the cryptographic capability to prove a condition about an attribute without disclosing the attribute’s actual value.

4.2.2.3 FR-HOLDER-PRIVACY-03: Generate New Pairwise Peer DID and Key Pair

Requirement: [Holder] should generate a new, unique pairwise Decentralized Identifier (DID) for each new connection, ensuring that the identifier used with one Verifier cannot be used to track or correlate the Holder’s activity with another Verifier.

Assessment Result: PARTIALLY SATISFIED

Implementation Evidence: uPort’s primary architecture uses a single, persistent public identifier for recovery, which conflicts with the goal of pairwise unlinkability. However, the underlying libraries provide the technical capability to create multiple identifiers.

Technical Implementation:

- **Persistent ‘uPortID‘:** The whitepaper defines a single, stable public identifier for each user, which enables correlation if used for all interactions [25].

- **Underlying Library Support for Multiple DIDs:** The ‘ethr-did‘ library, which provides the DID method for uPort, is capable of managing multiple identities, but it doesn’t mention the unique and new DID generation for each new connection. [47].

Component Assignment Verification: Correctly assigned to the **Holder** component. The holder’s mobile app manages the primary ‘uPortID‘ and has access to the underlying libraries to create new identifiers if a developer chooses to implement such a pattern.

Functional Completeness: The system is only partially complete because its default and encouraged architecture does not enforce pairwise-unique DIDs.

4.2.2.4 FR-ISSUER-PRIVACY-01: Create Verifiable Credentials Supporting Future Selective Disclosure

Requirement: [Issuer] should structure and issue credentials in a format that technically enables future selective disclosure and data minimization by the Holder.

Assessment Result: SATISFIED

Implementation Evidence: The uPort credentialing system was built to ensure that issued credentials are created with a granular structure, making them inherently suitable for future selective disclosure operations by the holder.

Technical Implementation:

- **Granular Claim Structure:** Credentials are created with a “claim” or “credentialSubject” object that contains multiple distinct key-value pairs[51].
- **Issuance of JWTs:** This granular structure is then packaged into a signed JWT, which the holder stores [25].

Component Assignment Verification: Correctly assigned to the Issuer component. Issuers use the “uport-credentials” library to construct, sign, and issue VCs in this granular, disclosure-ready format.

Functional Completeness: The credential issuance process is functionally complete. Any credential created using the standard uPort methodology is inherently structured to support selective disclosure by the holder at a later time.

4.2.2.5 FR-ISSUER-PRIVACY-02: Publish Update to Revocation Registry with Privacy-Preserving Mechanisms

Requirement: [Issuer] should publish revocation information using a privacy-preserving mechanism that does not publicly expose lists of revoked credential identifiers.

Assessment Result: SATISFIED

Implementation Evidence: uPort designed a novel on-chain revocation scheme, "EthrStatusRegistry2019", which uses a smart contract to track revocation status without revealing any personally identifiable information.

Technical Implementation:

- **Smart Contract-based Registry:** An Ethereum smart contract is used as the registry [49].
- **Revocation by Credential Hash:** computing a cryptographic hash of the entire credential JWT. A verifier then calls a revoked() method on the registry smart contract, using this hash to check if the credential has been marked as revoked by a valid issuer address, not by publishing the credential's ID [48].

Component Assignment Verification: Correctly assigned to the **Issuer** component. The issuer is the entity authorized to call the 'updateStatus' function on their own registry smart contract.

Functional Completeness: The mechanism fully achieves privacy-preserving revocation. It prevents the "scarlet letter" effect by not publishing any correlatable identifiers on-chain, thus protecting the privacy of holders whose credentials have been revoked.

4.2.2.6 FR-VERIFIER-PRIVACY-01: Check Revocation Status During Verification

Requirement: [Verifier] should be able to check the revocation status of a credential without revealing the full credential or the Holder's identity to the public revocation registry.

Assessment Result: SATISFIED

Implementation Evidence: The "EthrStatusRegistry2019" mechanism allows a verifier to check a credential's revocation status privately and efficiently.

Technical Implementation:

- **Off-Chain Information Transfer:** The verifier receives all necessary information to perform the check privately from the holder [48].
- **Private Registry Lookup:** The verifier performs a lookup to retrieve the relevant information. This lookup does not require broadcasting a public transaction that modifies the blockchain's state [25].

Component Assignment Verification: Correctly assigned to the **Verifier** component. The verifier's software performs this check as part of its validation logic without needing to broadcast any private information.

Functional Completeness: The revocation check is functionally complete. It allows a verifier to obtain cryptographically certain information about a credential's status while preserving the privacy of the entire interaction.

4.2.2.7 Assessment Summary

Functional Requirement Category	**Satisfied**	**Total**
Credential Verification (Presentation)	1	2
Connection Creation	0	1
Credential Creation (Issuance)	1	1
Delete/Revoke Credentials	2	2
Total NFR 1 Assessment	**4**	**6**

Table 4.2: uPort Privacy and Minimal Disclosure FR Satisfaction. Note: Based strictly on provided sources.

Overall Functional Requirements Satisfaction for NFR 1: 4/6 SATISFIED (67%)

Based strictly on the provided sources, the uPort system demonstrates a strong foundation for privacy in specific areas. It has a complete and well-documented implementation of selective disclosure for full attributes and a novel, privacy-preserving on-chain revocation registry. However, it is not functionally complete regarding advanced privacy techniques like zero-knowledge predicate proofs. Furthermore, its primary architectural model, which prioritizes a persistent public identifier for reputation and recovery, is only partially aligned with the goal of perfect, unlinkable pairwise privacy.

Chapter 5

Evaluation

This chapter provides a comprehensive evaluation of the derivation methodology developed. The evaluation initially assesses the strengths and limitations by performing a largely qualitative evaluation to examine the approach's properties, such as traceability and flexibility. Subsequently, it discusses the results after applying case studies to Sovrin and uPort, providing a comparative analysis and validating the methodology's practicality.

5.1 Evaluation of the Framework

This section evaluates the derivation framework as a research methodology. The evaluation focuses on the framework's advantages, inherent limitations, and analysis of its flexibility, feasibility, and ability to produce deterministic outcomes. This qualitative assessment helps to contextualize the framework's contribution in SSI research.

5.1.1 Advantages

5.1.1.1 Systematic Traceability

The framework's primary strength lies in establishing clear, traceable connections between abstract SSI principles and concrete implementation requirements. The three phases, four-step methodology ensures that each derived functional requirement can be traced back to its source NFR, use case, and responsible component. This addresses a critical gap in existing SSI evaluation approaches that often lack systematic derivation methodologies.

5.1.1.2 Objective Assessment Capability

By translating subjective principles into measurable functional requirements, the framework enables objective, reproducible evaluations. The satisfaction criteria (satisfied/partially satisfied/not satisfied) with concrete evidence requirements eliminates ambiguity in assessment outcomes. The case studies demonstrate this objectivity—Sovrin's 100% versus uPort's 67% satisfaction rates are based on verifiable implementation evidence rather than subjective interpretations.

5.1.1.3 Component-Centric Responsibility Assignment

The framework's mapping to the trust triangle (Issuer, Holder, Verifier) provides clear responsibility assignment for each functional requirement. This architectural grounding ensures that derived requirements are implementable and that evaluation results can guide specific development efforts.

5.1.1.4 Empirically-Grounded Prioritization

By leveraging Cucko et al.'s expert survey data for NFR prioritization, which incorporates domain expertise. This ensures that the most critical SSI principles receive appropriate emphasis in the evaluation process.

5.1.2 Disadvantages and Limitations

5.1.2.1 Limited Examples Illustrated

The current case studies cover only one identified SSI principle. While this demonstrates the methodology's effectiveness, more illustrative examples with remaining NFRs might inspire potential improvements.

5.1.2.2 Static Evaluation Approach

The framework provides snapshot assessments based on available documentation and specifications, but lacks mechanisms for evaluating dynamic system behavior, performance characteristics, or real-world deployment challenges. This limitation is particularly relevant for NFRs like "Usability and User Experience" where user testing would provide more meaningful insights than documentation analysis.

5.1.2.3 Documentation Dependency

The framework's effectiveness depends heavily on the availability and quality of technical documentation and white papers. Systems with poor documentation may appear less compliant despite having robust implementations, while systems with comprehensive documentation may appear more compliant than their actual capabilities warrant.

5.1.3 Flexibility Assessment

5.1.3.1 Adaptable Methodology

The framework demonstrates strong flexibility in its core methodology. The four-step process can accommodate different NFR sets, alternative component architectures beyond the trust triangle, and various evidence sources. The case studies show successful application to architecturally distinct systems (Sovrin's advanced cryptographic approach versus uPort's blockchain-centric design).

5.1.3.2 Extensible NFR Coverage

The framework's systematic approach enables straightforward extension to additional SSI principles. The established pattern of NFR decomposition, process mapping, and component assignment can be replicated for any of the remaining 16 SSI principles identified by Cucko et al.

5.1.3.3 Configurable Evidence Requirements

The evidence collection framework accommodates multiple sources (technical documentation, academic literature, white papers) and can be adapted to different levels of detail based on evaluation objectives and resource constraints.

5.1.3.4 Limited Architectural Flexibility

However, the framework's grounding in the trust triangle model may limit its applicability to SSI systems with fundamentally different architectures. Emerging approaches like multi-party computation or alternative consensus mechanisms might require architectural extensions to the current component model.

5.1.4 Feasibility Analysis

5.1.4.1 Resource Requirements

The framework demonstrates practical feasibility with moderate resource requirements. Each case study required analysis of available documentation, technical specifications, and academic sources—resources typically available for established SSI systems. The systematic approach enables efficient evaluation once the initial NFR-to-FR derivation is complete.

5.1.4.2 Scalability Considerations

Scaling to comprehensive evaluation (all 18 SSI principles) would require significant but manageable effort. Based on the current two-NFR implementation producing 10+ functional requirements, full coverage might generate 80 to 100 functional requirements. While substantial, this represents a finite, achievable scope for thorough SSI system evaluation.

5.1.4.3 Skill Requirements

Effective application requires domain expertise in SSI technologies, cryptographic concepts, and systems architecture. The framework does not eliminate the need for expert knowledge but rather provides a structured approach for applying such expertise systematically.

5.1.4.4 Tool Integration Potential

The framework's structured approach and clear criteria suggest potential for tool-supported automation, particularly for evidence collection and requirement satisfaction tracking. This could significantly improve the feasibility of large-scale or repeated evaluations.

5.1.5 Deterministic Outcomes

5.1.5.1 Reproducible Results

The framework's structured methodology and explicit evidence requirements support reproducible evaluations. Different evaluators applying the framework to the same system with the same evidence sources should reach consistent conclusions about functional requirement satisfaction.

5.1.5.2 Clear Decision Criteria

The satisfaction model with explicit criteria (implementation evidence, component assignment verification, functional completeness) provides deterministic decision-making. The case studies demonstrate this clarity, in which each functional requirement assessment includes specific evidence justifying the satisfaction determination.

5.1.5.3 Transparent Traceability

The framework's documentation requirements ensure that evaluation outcomes can be traced back through the derivation process to source NFRs, enabling verification and review of assessment logic.

5.1.5.4 Limitations of Determinism

However, deterministic outcomes depend on the availability and interpretation of evidence. Where documentation is ambiguous or incomplete, evaluator judgment remains necessary. The framework provides structure for such judgments but cannot eliminate all subjectivity, particularly in borderline cases or partial implementations.

5.1.6 Framework Validation

The case studies provide initial validation of the framework's effectiveness. The ability to differentiate between Sovrin's comprehensive privacy implementation and uPort's selective approach demonstrates the framework's sensitivity to meaningful implementation differences. The alignment between assessment outcomes and systems' documented design priorities suggests the framework captures relevant aspects of SSI system capabilities.

However, broader validation would benefit from:

- Application to additional SSI systems with different architectural approaches
- Comparison with expert assessments of the same systems
- Extension to additional NFR categories beyond privacy and minimal disclosure
- Evaluation of framework outcomes against real-world deployment experiences

5.1.7 Implications for SSI Development

The framework provides practical value for multiple stakeholders in the SSI ecosystem:

For Developers: Clear functional requirements provide implementable specifications that ensure compliance with SSI principles.

For Evaluators: Systematic methodology enables consistent, objective assessment of SSI systems.

For Researchers: Structured approach supports comparative analysis and identification of gaps in current SSI implementations.

For Standards Bodies: Evidence-based evaluation framework can inform standards development and compliance verification processes.

The framework represents a significant step toward bridging the gap between abstract SSI principles and concrete system functional requirements, providing a foundation for more rigorous SSI system development and evaluation.

Chapter 6

Final Considerations

6.1 Summary

This thesis addressed a critical gap in Self-Sovereign Identity (SSI) research by developing a systematic framework to bridge the gap between abstract SSI principles and concrete, implementable functional requirements. The research was motivated by the challenge that existing SSI evaluation approaches predominantly focus on high-level principle compliance without providing systematic methodologies for deriving verifiable functional specifications.

The primary contribution of this thesis is a structured three-phase, four-step NFR-to-FR mapping methodology that transforms abstract Non-Functional Requirements into concrete Functional Requirements within the SSI domain, and the validation framework for the methodology.

Building upon the comprehensive SSI property classification by Cucko et al. [7], the derivation framework employs a component-centric approach grounded in the W3C trust triangle model (Issuer, Holder, Verifier) and incorporates simplified Softgoal Interdependency Graphs for systematic requirement derivation.

The methodology was validated through detailed case studies of two prominent SSI systems: Sovrin and uPort. These implementations demonstrated the framework's practical applicability and its ability to provide objective, reproducible assessments. Sovrin achieved 100% satisfaction for Privacy and Minimal Disclosure requirements through advanced cryptographic mechanisms, including zero-knowledge proofs and privacy-preserving revocation, while uPort achieved 67% satisfaction with limitations in minimizing privacy.

The evaluation framework successfully differentiated between these systems' implementation approaches, providing empirically-grounded assessments that align with their documented design. This demonstrates the framework's effectiveness in capturing meaningful differences in SSI implementations while maintaining objectivity and reproducibility.

6.2 Conclusions

This research establishes that systematic evaluation of SSI systems is not only possible but essential for the field's continued maturation. The successful transformation of abstract principles into measurable criteria represents a methodological advancement.

The shift toward concrete, testable requirements supports more rigorous engineering practices and facilitates standardization efforts across the decentralized identity landscape.

Most significantly, this work validates that component-centric requirement derivation can bridge the persistent gap between SSI abstract properties and concrete functional requirements. By providing a derivation framework to systematically translate principles into actionable specifications and a validation framework, the research empowers stakeholders to build more trustworthy and compliant SSI applications.

6.3 Future Work

This work also indicates several directions for future research, ranging from immediate extensions to long-term research opportunities:

6.3.1 Framework Extension and Validation

Complete NFR Coverage: The most immediate extension involves applying the systematic methodology to the remaining 16 SSI principles identified by Cucko et al. This would provide comprehensive coverage of SSI requirements and enable broad system evaluation.

Broader System Validation: Expanding the case study scope to include additional SSI systems with diverse architectural approaches would strengthen validation of the framework's generalizability. Candidates include ShoCard, Civic, and other emerging systems based on different technological foundations.

Expert Validation Studies: Conducting comparative studies where domain experts independently assess the same systems using both the systematic framework and their professional judgment would provide important validation of framework outcomes and identify areas for refinement.

6.3.2 Methodological Enhancements

Graduated Assessment Scale: Replacing the current satisfaction model with a more nuanced scoring system (e.g., 1-5 scale: Not Implemented, Partially Implemented, Implemented, Well Implemented, Exemplary Implementation) could capture degrees of compliance and provide more detailed system comparisons.

Dynamic Evaluation Capabilities: Developing methodologies for assessing runtime behavior, performance characteristics, and user experience aspects would address current limitations in evaluating dynamic system properties. This might involve integration with user testing protocols, performance benchmarking, and operational monitoring.

Architectural Flexibility: Extending the framework to accommodate emerging SSI architectures beyond the traditional trust triangle, such as multi-issuer credentials, delegation mechanisms, or alternative consensus approaches, would ensure continued relevance as the field evolves.

6.3.3 Tool Development and Automation

Evaluation Support Tools: Developing software tools to support evidence collection, requirement satisfaction tracking, and automated report generation could significantly improve the framework's practical feasibility for large-scale evaluations.

Integration with Development Workflows: Creating tools that integrate the functional requirements into software development life cycles—such as test case generation, compliance checking, or continuous integration validation—could enhance the framework's utility for SSI system developers.

Standards Integration: Collaborating with standards bodies to integrate the framework's functional requirements into formal SSI standards and certification processes could increase adoption and establish industry-wide evaluation practices.

6.3.4 Empirical Research Opportunities

Real-World Deployment Studies: Investigating the relationship between framework assessment outcomes and real-world deployment success, user adoption, or security incidents could validate the framework's predictive value and identify refinement opportunities.

Cross-Domain Application: Exploring the application of similar systematic NFR-to-FR mapping methodologies in related domains such as blockchain systems, distributed computing, or privacy-preserving technologies could demonstrate the broader applicability of the approach.

6.3.5 Theoretical Extensions

Requirements Interdependency Analysis: Investigating relationships and potential conflicts between functional requirements derived from different NFRs could provide insights into system design trade-offs and optimization strategies.

Context-Sensitive Requirements: Exploring how functional requirements might vary based on deployment context (e.g., healthcare vs. financial services), regulatory environment, or threat model could enhance the framework's practical applicability.

Evolutionary Requirements Framework: Developing mechanisms for systematically updating functional requirements as SSI technologies, standards, and threat landscapes evolve could ensure the framework's long-term relevance.

6.3.6 Broader Impact Research

Policy and Regulation Interface: Investigating how systematic SSI evaluation frameworks could inform regulatory compliance, privacy impact assessments, or policy development in the digital identity space represents an important application domain.

Economic Impact Analysis: Studying the economic implications of systematic SSI evaluation—including development costs, risk mitigation, and market adoption effects—could provide valuable insights for industry stakeholders.

Social Impact Assessment: Explorisng how systematic evaluation of SSI systems relates to broader social outcomes such as digital inclusion, privacy protection, or democratic participation could connect technical evaluation to societal benefits.

The systematic approach developed in this thesis provides a foundation for advancing SSI research and practice. By pursuing these future work directions, the research community can build upon this foundation to develop more sophisticated evaluation methodologies, support better SSI system development, and ultimately contribute to a better realization of user-controlled, privacy-preserving digital identity ecosystems.

Bibliography

[1] Christopher Allen. *The Path to Self-Sovereign Identity.* 2016.

[2] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels.* RFC 2119. Internet Engineering Task Force, 1997.

[3] Kim Cameron. “The laws of identity”. In: *Microsoft Corp* 12 (2005), pp. 8–11.

[4] Tomer Jordi Chaffer and Justin Goldston. “On the existential basis of self-sovereign identity and soulbound tokens: An examination of the “self” in the age of Web3”. In: (2022).

[5] Lawrence Chung and Sam Supakkul. “Representing nfrs and frs: A goal-oriented and use case driven approach”. In: *International Conference on Software Engineering Research and Applications.* Springer. 2004, pp. 29–41.

[6] Lawrence Chung et al. *Non-functional requirements in software engineering.* 2000.

[7] Sandi Cucko et al. “Towards the classification of self-sovereign identity properties”. In: *IEEE Access* 10 (2022), pp. 88306–88329.

[8] Špela Čučko and Muhamed Turkanović. “Decentralized and self-sovereign identity: Systematic mapping study”. In: *IEEE access* 9 (2021), pp. 139009–139027.

[9] Decentralized Identity Foundation. *DIF Presentation Exchange.* Tech. rep. Decentralized Identity Foundation, 2021.

[10] Uwe Der, Stefan Jähnichen, and Jan Sürmeli. “Self-sovereign identity − opportunities and challenges for the digital revolution”. In: *arXiv preprint arXiv:1712.01767* (2017).

[11] Dock Labs. *Self-Sovereign Identity: The Ultimate Guide 2025.* 2025. URL: `https://www.dock.io/post/self-sovereign-identity`.

[12] Jonas Eckhardt, Andreas Vogelsang, and Daniel Méndez Fernández. “Are” non-functional” requirements really non-functional? an investigation of non-functional requirements in practice”. In: *Proceedings of the 38th international conference on software engineering.* 2016, pp. 832–842.

[13] Md Sadek Ferdous, Farida Chowdhury, and Madini O Alassafi. “In search of self-sovereign identity leveraging blockchain technology”. In: *IEEE access* 7 (2019), pp. 103059–103079.

[14] GeeksforGeeks. *Functional vs. Non Functional Requirements.* 2020. URL: `https://www.geeksforgeeks.org/software-engineering/functional-vs-non-functional-requirements/`.

[15] Daniel Gross and Eric Yu. “From non-functional requirements to design through patterns”. In: *Requirements Engineering* 6.1 (2001), pp. 18–36.

[16] Andrea Herrmann, Daniel Kerkow, and Joerg Doerr. “Exploring the Characteristics of NFR Methods–a Dialogue about two Approaches”. In: *International Work-*

ing Conference on Requirements Engineering: Foundation for Software Quality. Springer. 2007, pp. 320–334.

[17] Hyperledger Foundation. *Anoncreds Design.* 2025. URL: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/design/002-anoncreds/README.html.

[18] Hyperledger Foundation. *AnonCreds Specification.* 2025. URL: https://hyperledger.github.io/anoncreds-spec/.

[19] Hyperledger Foundation. *How Credential Revocation Works.* 2025. URL: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html.

[20] Hyperledger Foundation. *Hyperledger Indy HIPE 0011: Credential Revocation.* 2025. URL: https://hyperledger-indy.readthedocs.io/projects/hipe/en/latest/text/0011-cred-revocation/README/.

[21] Hyperledger Foundation. *Indy Walkthrough.* 2025. URL: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/getting-started/indy-walkthrough.html.

[22] Identity.com. *Your Guide to Self-Sovereign Identity (SSI).* URL: https://www.identity.com/self-sovereign-identity/.

[23] Dmitry Khovratovich and Jason Law. "Sovrin: digital identities in the blockchain era". In: *Github Commit by jasonalaw October* 17.38-99 (2017), p. 41.

[24] Marcos Allende López. "Self-sovereign identity: The future of identity: Self-sovereignity, digital wallets, and blockchain". In: (2020).

[25] Christian Lundkvist et al. "Uport: A platform for self-sovereign identity". In: *https://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf* 128 (2017), p. 214.

[26] Ruth Malan, Dana Bredemeyer, et al. "Functional requirements and use cases". In: *Bredemeyer Consulting* (2001), pp. 335–1653.

[27] Carlo Mazzocca et al. "A survey on decentralized identifiers and verifiable credentials". In: *IEEE Communications Surveys & Tutorials* (2025).

[28] Alexander Mühle et al. "A survey on essential components of a self-sovereign identity". In: *Computer Science Review* 30 (2018), pp. 80–86.

[29] Nitin Naik and Paul Jenkins. "Does Sovrin network offer sovereign identity?" In: *2021 IEEE International Symposium on Systems Engineering (ISSE).* IEEE. 2021, pp. 1–6.

[30] Nitin Naik and Paul Jenkins. "uPort open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain". In: *2020 IEEE International Symposium on Systems Engineering (ISSE).* IEEE. 2020, pp. 1–7.

[31] Razieh Nokhbeh Zaeem et al. "Blockchain-based self-sovereign identity: Survey, requirements, use-cases, and comparative study". In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.* 2021, pp. 128–135.

[32] Roberto A Pava-Díaz, Jesús Gil-Ruiz, and Danilo A López-Sarmiento. "Self-sovereign identity on the blockchain: contextual analysis and quantification of SSI principles implementation". In: *Frontiers in Blockchain* 7 (2024), p. 1443362.

[33] Alex Preukschat and Drummond Reed. *Self-sovereign identity.* 2021.

[34] QuarkID Team. *QuarkID Protocol Whitepaper.* White Paper. QuarkID, 2024.

[35] Drummond Reed, Jason Law, and Daniel Hardman. "The technical foundations of Sovrin". In: *The Technical Foundations of Sovrin* (2016).

[36] Drummond Reed et al. "Decentralized identifiers (dids) v1. 0". In: *Draft Community Group Report* (2020).

[37] Requiment. *What Are Functional and Non-Functional Requirements?* 2024. URL: `https://www.requiment.com/what-are-functional-and-non-functional-requirements/`.

[38] Suzanne Robertson and James Robertson. *Mastering the requirements process: Getting requirements right.* 2012.

[39] Abylay Satybaldy, Mariusz Nowostawski, and Jørgen Ellingsen. "Self-sovereign identity systems: Evaluation framework". In: *IFIP International Summer School on Privacy and Identity Management.* 2019, pp. 447–461.

[40] Frederico Schardong and Ricardo Custódio. "Self-sovereign identity: a systematic review, mapping and taxonomy". In: *Sensors* 22.15 (2022), p. 5641.

[41] Daria Schumm, Katharina OE Müller, and Burkhard Stiller. "Are We There Yet? A Study of Decentralized Identity Applications". In: *arXiv preprint arXiv:2503.15964* (2025).

[42] Johannes Sedlmeir et al. "Digital identities and verifiable credentials". In: *Business & Information Systems Engineering* 63.5 (2021), pp. 603–613.

[43] Reza Soltani, Uyen Trang Nguyen, and Aijun An. "A Survey of Self-Sovereign Identity Ecosystem". In: *Security and Communication Networks* 2021.1 (2021), p. 8873429.

[44] Sovrin Foundation. *Frequently Asked Questions.* Sovrin Foundation. 2025.

[45] Sovrin Foundation. *Sovrin DID Method Specification.* 2025. URL: `https://sovrin-foundation.github.io/sovrin/spec/did-method-spec-template.html`.

[46] Andrew Tobin and Drummond Reed. "The inevitable rise of self-sovereign identity". In: *The Sovrin Foundation* 29.2016 (2016), p. 18.

[47] uPort Project. *ethr-did: Create ethr DIDs.* Software Library. uPort Project, 2018. URL: `https://github.com/uport-project/ethr-did`.

[48] uPort Project. *ethr-status-registry: Verifiable Credential Status Resolver Using an Ethereum Contract as a Backend.* Software Library. uPort Project, 2019. URL: `https://github.com/uport-project/ethr-status-registry`.

[49] uPort Project. *Revocation Registry: Ethereum Revocation Registry Contract.* Smart Contract. uPort Project, 2018. URL: `https://github.com/uport-project/revocation-registry`.

[50] uPort Project. *Selective Disclosure Flow.* Technical Specification. uPort Project, 2018. URL: `https://github.com/uport-project/specs/blob/develop/flows/selectivedisclosure.md`.

[51] uPort Project. *uPort Credentials Integration Guide.* GitHub Repository Documentation. Developer Documentation. uPort Project, 2018. URL: `https://github.com/uport-project/uport-credentials/blob/master/docs/guides/index.md`.

[52] uPort Project. *uPort Credentials Tutorial.* GitHub Repository Documentation. Technical Documentation. uPort Project, 2018. URL: `https://github.com/uport-project/uport-credentials/blob/master/docs/guides/tutorial.md`.

[53] WebOfTrustInfo. *Anonymous Credentials in Sovrin.* `https://github.com/WebOfTrustInfo/rwot3-sf/blob/master/topics-and-advance-readings/anonymous-credentials-in-sovrin.md`. 2016.

[54] Wikipedia contributors. *Self-sovereign identity*. Wikipedia, The Free Encyclopedia, 2025. URL: https://en.wikipedia.org/wiki/Self-sovereign_identity.

[55] Wikipedia contributors. *Verifiable credentials*. 2025. URL: https://en.wikipedia.org/wiki/Verifiable_credentials.

[56] Phil Windley and Drummond Reed. *Sovrin: A Protocol and Token for Self-Sovereign Identity and Decentralized Trust*. White Paper. Version 1.0. Sovrin Foundation, 2018.

[57] Phillip Windley. "How sovrin works". In: *Sovrin Foundation* (2016), pp. 1–10.

[58] Phillip J Windley. "Sovrin: An identity metasystem for self-sovereign identity". In: *Frontiers in Blockchain* 4 (2021), p. 626726.

[59] World Wide Web Consortium. *Decentralized Identifiers (DIDs) v1.0*. 2022. URL: https://www.w3.org/TR/did-core/.

[60] World Wide Web Consortium. *Verifiable Credentials Data Model v2.0*. W3C Recommendation. 2025. URL: https://www.w3.org/TR/vc-data-model/.

[61] Jun Ying Zhou. "Functional requirements and non-functional requirements: a survey". PhD thesis. Concordia University, 2004.

Abbreviations

SSI	Self-Sovereign Identity
VC	Verifiable Credential
VP	Verifiable Presentation
UC	Use Case
FR	Functional Requirement
NFR	Non-Functional Requirement
DID	Decentralized Identifier
VDR	Verifiable Data Registry
W3C	World Wide Web Consortium
DIF	Decentralized Identity Foundation
SIG	Softgoal Interdependency Graph
AnonCreds	Anonymous Credentials

List of Figures

List of Tables

Listings

Appendix A

Contents of the Repository

The code repository contains the following content:

Installation

Operation