DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Establishment of a Minimum Viable Self-Sovereign Identity Network

Kilian Käslin

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Establishment of a Minimum Viable Self-Sovereign Identity Network

Etablierung eines minimal tragfähigen Self-Sovereign Identitätsnetzwerks

Author:	Kilian Käslin
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ulrich Gallersdörfer, M.Sc.
Submission Date:	15.06.2020

I confirm that this master’s thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.06.2020 Kilian Käslin

Acknowledgments

Foremost I want to thank my advisor Ulrich Gallersdörfer. In our weekly meetings, he provided crucial feedback on the current state of the thesis and valuable new ideas. Thus helping me navigate the research process and shaping the final thesis. He successfully upheld an environment in which I could develop my own ideas and perspective on the subject.

Furthermore, I want to thank Professor Dr. Florian Matthes for his help in developing the thesis title, the thesis goals and the opportunity to write my thesis at his chair for Software Engineering for Business Information Systems (SEBIS).

I want to thank all participants of the user survey. They allowed me to evaluate the central artifacts of my research. I especially want to thank Dr. Carsten Stoecker, Daniël Du Seuil, Flavia Soare, Heather Vescent, Kaliya Young, Kim Hamilton Duffy, Oliver Terbu, Shivam Sethi, field experts, who provided valuable insights. Finally, I want to thank my family for making my studies possible, and thus this thesis realizable. Furthermore, I want to thank my friends for the support throughout the process of conducting my research. Kaja Ninnis proved to be a great help proofreading the thoughts I put on paper.

Thank you.

Abstract

In this masters thesis, we investigate the Self-Sovereign Identity (SSI) concept, a new identity management system. The proponents of SSI argue that it improves upon classical identity management solutions by giving the user more control over his identity. Entities map their digital identity to Decentralized Identifiers (DIDs) registered on verifiable data registries, for example, blockchains. Using Verifiable Credentials (VCs) entity's identity characteristics can be attested and proven to third parties.

To investigate the SSI concept, this master's thesis implements a prototypical user-centric minimum viable SSI network with a focus on the user journey. We develop a network consisting of two institution web services providing interfaces for user interaction. User's can login to the institution web service using their DIDs and request and present VCs.

Furthermore, we compile a list of SSI use-cases that include enterprise and user-centric use-cases field experts regard as candidates for first SSI applications. Additionally, we assemble a list of challenges that must be solved to mature the SSI concept and its application from the perspective of field experts. We identify four groups of challenges, security and trust, regulatory, user experience, and market acceptance challenges.

Moreover, we investigate the potential centralization of the SSI trust network, by developing an issuer desirability function. We attest a centralization in regards to VC-issuing entities.

Keywords: Self-Sovereign Identity, Distributed Identifiers, Verifiable Credentials, Verifiable Presentations, Trust Network, Use-Cases, Challenges

Contents

Acronyms

AHP	Analytic Hierarchy Process
API	Application Programming Interface
DID	Decentralized Identifier
eIDAS	electronic Identification, Authentication and trust Services
eSSIF	European SSI Framework
FR	Functional Requirements
GDPR	European General Data Protection Regulation
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
JWT	JavaScript Object Notation Web Tokens
KYC	Know Your Customer
MADM	Multi Attribute Decision Making
mvSSIn	minimum viable SSI network
NFR	Non-Functional Requirements
OIDC	OpenID Connect
PII	Personally Identifiable Information

RPC	Remote Procedure Call
RQ	Research Question
SIOP	Self-Issued OpenID Connect
SSI	Self-Sovereign Identity
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
VC	Verifiable Credential
VP	Verifiable Presentation
ZKP	Zero Knowledge Proof

1 Introduction

A multitude of identity management solutions exist, facilitating interactions on the internet. However, the proponents of the emerging Self-Sovereign Identity (SSI) concept criticize that classical identity management solutions suffer from a multitude of issues. Companies and governments can gather, evaluate, and utilize user information to their advantage. Furthermore, in classical solutions, identifiers are mostly not controlled by the user but by the identity providers. Therefore, these identifiers can be retracted or altered without the user's consent.

The SSI concept proposes a privacy-preserving, internet-wide identity layer. Using blockchain technology, Decentralized Identifiers (DIDs) can be created, serving as unique identifiers. DIDs are controlled solely by the user without dependency from centralized identity providers. The SSI concept is extended by a credential layer. The credential layer allows issuers to provide Verifiable Credentials (VCs). VCs store signed identity characteristics in a tamper-proof manner. Users can store VCs and share the contained private information selectively in the form of Verifiable Presentations (VPs), thereby proving characteristics of their identity to third parties. Although applied implementations exist, the SSI concept is still under development.

1.1 Problem Statement

Although implementations of the SSI concept already exist, it is unclear how a mvSSIn can be established and to which extend the promised qualities of the SSI concept apply. To analyze how a mvSSIn can be established, we develop a prototypical implementation.

A specific quality promised by the creators of the SSI concept is the decoupling from centralized identity providers. Centralized identity providers can gather, evaluate, and utilize users' information to their advantage [32, 8]. Furthermore, the identity providers have control over the users' accounts. This undermines the users' privacy and the users' autonomy. We analyze if the SSI concept requires new centralized entities. Therefore, we discuss this question in a user survey with potential users and field experts. As a result, we develop a function assigning the desirability to request a VC from an issuing entity.

Furthermore, since the SSI concept is still under development and only a few production-level applications exist, it is unclear which challenges and use-cases the SSI community is currently working on. To gain insight, we question field experts.

1.2 Research Questions

In this master's thesis, we answer the following research questions (RQ):

RQ1 What are the essential requirements for the implementation of a minimum viable SSI network?

To investigate what a mvSSIn is comprised of, we develop a set of essential requirements. We examine which requirements are needed to fulfill a login with DID use-case and an access management use-case [59]. Furthermore, we investigate if further requirements are needed to cover the central features of SSI described in the core specifications of DIDs and VCs, presented in 3.

RQ2 How can a prototypical minimum viable SSI network be implemented?

Based on the requirements we develop in RQ1, we implement a mvSSIn in a prototypical form to investigate if exiting SSI frameworks and specifications allow for the development of a prototype covering the requirements of RQ1. This research question is closely related to RQ3 since the implementation sets the constraints for answering RQ3.

RQ3 What does a user journey taking place in the prototypical implementation of a minimum viable SSI network look like?

In conjunction with the implementation of a prototypical mvSSIn, we examine what a mvSSIn user journey looks like. The user journey covers two perspectives, the user-perspective, and the institution-perspective.

RQ4 How can the issuer desirability be modeled?

A central role in the SSI concept can be attributed to issuers. They provide users with credentials to prove identity attributes. We develop a function that assigns the desirability to request a credential from an issuer, based on issuer characteristics, to investigate the issuers' potential centralization. Furthermore, we intend this function to be used in further modeling of the SSI ecosystem.

RQ5 Which SSI use-cases do field experts regard as candidates for first real-world SSI applications?

To answer this research question, we question field experts which use-cases they regard as candidates for first real-world SSI applications. This covers use-cases they are currently working on, use-cases for which they see interest from the market, and use-cases they perceive as promising. Thereby, we compile a list of use-cases to which the SSI concept and, to some extent, our prototype can be applied. Furthermore, the use-cases can serve as a starting point for future work.

RQ6 Which challenges must be solved by the SSI community from the perspective of field experts to mature the SSI concept and its applications?

To further our understanding of the SSI concept, its fallacies, and which fallacies might apply to our prototype, we question field experts to answer this research question. Since each interview candidate has a different perspective, we compile a list of challenges as perceived from different angles.

1.3 Research Approach

To develop the prototypical implementation of a mvSSIn, the corresponding user journey, and an issuer desirability function we manly follow the design science methodology as defined by Hevner in *A Three Cycle View of Design Science Research* [26]. Figure 1.1 reveals a conceptual overview of the process prescribed by the design science methodology. The methodology is subdivided into three cycles. The Relevance Cycle creates a set of requirements derived from the environment and field tests the artifacts produced in the Design Cycle. The Design Cycle implements the artifacts based on the requirements and iteratively evaluates and improves those artifacts. The Relevance Cycle provides grounding for the design cycle, in terms of knowledge applicable to the development process, and returns the results obtained in the Design Cycle to the knowledge base.

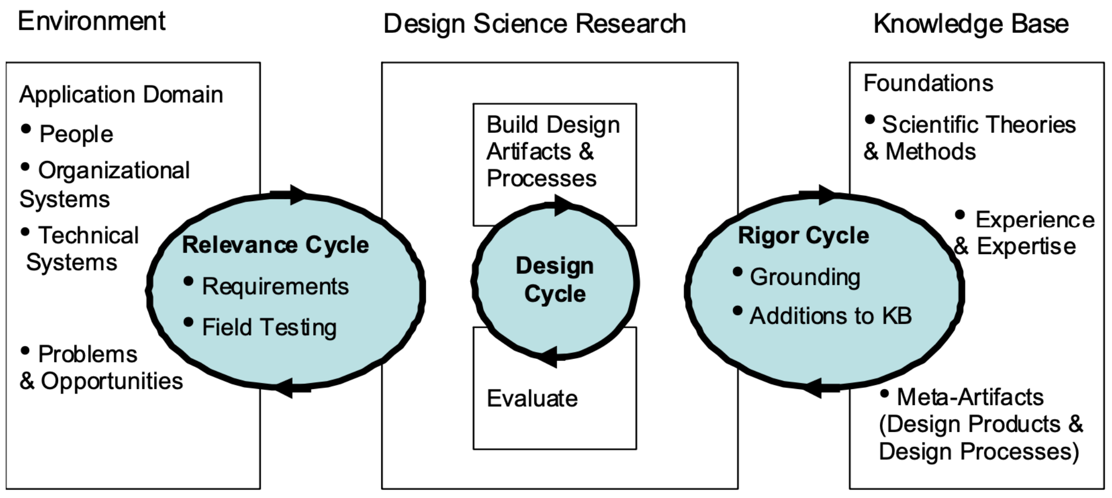


Figure 1.1: Design Science Research Cycles [26]

In the first iteration of the Relevance Cycle, we develop a set of initial requirements derived from an analysis of the application domain. Based on the requirements, documentation, and specification gathered during the literature review, we develop a mvSSIn and a mvSSIn user journey. To evaluate these two artifacts, we conduct a user study consisting of a set of interviews, described in the following paragraph. After each interview, if needed, we update the requirements and improve the mvSSIn implementation as well as the mvSSIn user journey.

We conduct the field testing of the two artifacts described in the previous paragraph, by first demonstrating the implementation and the initial user journey to an audience of potential users and field experts. We gather their feedback using semi-structured interviews [21]. Additionally, we use the semi-structured interviews to answer RQ5 and RQ6. We ask the interview partners which use-cases they regard as candidates for first real-world SSI applications and which challenges must be solved by the SSI community to mature the SSI concept and its applications. In a final set of questions, we gather information for the issuer desirability function described in the following paragraph.

As a last artifact we develop an issuer desirability function. Based on the assumption that the issuer desirability can be classified as a Multi Attribute Decision Making (MADM) problem, as described by Tzeng and Huang in *Multiple attribute decision making : methods and applications* [48], we solve the MADM problem following the Analytic Hierarchy Process (AHP) as defined by Saaty in *Decision making with the analytic hierarchy process* [40].

To attain the knowledge required for the initial requirements and the development of the three Design Cycle artifacts, we conduct a literature review [67]. In the literature review, various sources were consulted, such as conference papers, journals, books, blogs, and reputable online pages.

The results obtained in the development of the three artifacts and the user survey provide additions to the knowledge base.

1.4 Contribution

This thesis contributes to the knowledge base:

- A prototypical implementation of a mvSSIn.
- A mvSSIn user journey.
- An issuer desirability function, mapping the desirability to request a VC from a specific issuer based on the issuer's characteristics.
- A set of SSI use-cases field expert regard as candidates for first real-world applications.
- A set of challenges that must be solved to mature SSI concept and its applications from the perspective of field experts.

1.5 Outline

In this section, we present the outline of the thesis. Following this introductory chapter, we list the related work of this thesis in chapter 2. We split it into two parts, work done in the context of SSI and academic literature describing SSI. In chapter 3, we explain the fundamentals of the SSI concept relevant to our implementation and user journey. The fundamentals explained, consist of the components, the roles, and the trust architecture of the SSI concept. In chapter 4, we list and explain the requirements for the mvSSIn implementation we derive from the SSI fundamentals and the user survey. We present the mvSSIn implementation we develop against the requirements in chapter 5. This consists of the underlying SSI implementation we relied on, the key components of our implementation, and the three services we implement and their interaction. In chapter 6, we present the mvSSIn user journey we develop in conjunction with the implementation. We segment the user journey into two parts, the user perspective and the institution perspective. Following, we present the results of our user survey, in chapter 7. The results consist of feedback we receive for the mvSSIn implementation and user journey and the expert input regarding the challenges and promising use-cases of the SSI concept. In chapter 8, we explain the issuer desirability function we develop. We describe the process of how we develop the function and its limitations. Finally, this master's thesis ends in chapter 9, with a reflection on the RQs, a conclusion, and an outlook on future work.

Disclaimer: The thesis' text makes use of terms in the generic masculine to ensure easier readability. However, we mean both female and male readers equally.

2 Related Work

In this chapter, we present related work to this thesis. We structured it into two categories, work done related to SSI and academic literature describing SSI.

2.1 Work on Self-Sovereign Identity

First, Allen's essay *The Path to Self-Sovereign Identity* [3] has to be introduced. In his essay Allen coins the term Self-Sovereign Identity and describes the path to SSI. At the beginning of the internet, central certification bodies issued identities on the internet, centralized identity. Then, big corporations began to provide identities that allowed users to login to multiple sites with the same identity, federated identity. The currently prevalent user-centric identity aimed to give the user full control over their identity. However, according to Allen, powerful institutions copied the concept and hindered the full achievement of the original goals. He then introduces Self-Sovereign Identity and provides a starting point for further discussion and development. Therefore, he introduces ten principles that aim to ensure the user's control over his identity.

Use Cases for Decentralized Identifiers [59] presents the current consensus of the motivations behind DIDs from the perspective of use-cases. It is based on input and collaboration with implementers currently building SSI products. In this thesis we implement the single sign on use-case and to some extent the life-long, recipient-managed credentials use-case as part of the mvSSIn in chapter 5 and 6. Furthermore, the use-cases presented in *Use Cases for Decentralized Identifiers* mostly match the potential use-cases the field experts elaborate on in the user survey described in chapter 7.

Verifiable Credentials Use Cases [61] presents a set of credential use-cases focused on concrete scenarios. The use-cases are intended to serve as a basis for further standardization, especially in regards to the interoperability of low- and high-stakes claims. The document first approaches the presentation of simple use-cases from the perspective of user-needs. It splits the user needs into seven key domains and addresses each user-need with a credential use-case. Following, the authors present complex use-cases, including preliminary threat models. We identify the investigation of threat models and development of mitigation approaches as one of the central challenges to the SSI concept in the user survey described in chapter 7.

2.2 Academic Literature

In his master's thesis "Analysis and Evaluation of Blockchain-based Self-Sovereign Identity Systems" [41], Schäfer first investigates the evolution of online identities, similarly to Allen. He then analyses the SSI concept and identifies four layers, a Distribute Public Key Infrastructure, a Distributed Key Management System, VCs, and VPs. Further analysis results in the conclusion that wallets play a central role in SSI implementations. Schäfer then continues to investigate different DID Methods. He identifies Sovrin's DID Method to have the broadest spectrum in features. Furthermore, he concludes that governance models are needed to create trust in SSI implementations and to further their development. We follow up on Schäfer's suggestion for future work to implement an SSI prototype.

Mühle, Grüner, Gayvoronskaya, and Meinel provides an overview of the SSI concept, with a focus on the components that they identify as essential to the SSI architecture in their paper "A survey on essential components of a self-sovereign identity" [31]. The central components from their perspective consist of the identification, authentication, VCs, and storage. They conclude that the SSI concept promises its users more control and a user-centric experience, leaving only the VC-issuers as centralized entities in the system. This is in accordance with our findings in chapter 8, where we investigate the potential centralization of VC-issuers.

In "A first look at identity management schemes on the blockchain" [10] Dunphy and Petitcolas, evaluate representative approaches to the SSI system using a seminal framework that characterizes the nature of successful identity management schemes. They identify two particular hurdles. Firstly, the current approaches lack a concept for user experience. This is especially so regarding the usability of the intended implementation since users are not equipped to conduct effective cryptographic key management. In this thesis, we develop a user journey and UI with usability as a main goal. Secondly, Dunphy and Petitcolas identify the tightening regulatory landscapes as a further hurdle. We discuss this issue with field experts in the user survey we present in chapter 7.

Gruner, Muhle, Gayvoronskaya, and Meinel develop a general quantifiable trust model for blockchain-based identity management in "A quantifiable trust model for blockchain-based identity management" [24]. They model digital identities, claims, attestations, as well as their relations as directed graphs and trust functions that define the flow of trust from one element to another. The trust score of an attestation is linked to the trust score of the issuing entity. In turn, the trust score of the attestation determines the trust score of a claim. Successively, the trust score of an identity is derived from the claims that are issued to it. This trust architecture results in a closed-loop since the issuing entities trust is dependent on the trust score of its identity. Therefore, the authors propose an initial set of pre-trusted entities. Their trust model provides an approach to automated reasoning on the trustworthiness of claims and digital identities. This allows entities to set specific scores of trustworthiness they want to adhere to, or they expect from parties interacting with them. Contrary to our issuer desirability function, their approach generates a measure of trust in the individual elements of a system. In contrast, our approach tries to identify if VCs will be requested from specific issuers in an over proportional manner due to an issuer's higher desirability.

3 Fundamentals

In this chapter, we present the fundamentals of the SSI concept. First, we introduce the main components. Second, we describe the roles, and third, we describe the trust architecture. The fundamentals serve as the basis for the requirements for the mvSSIn implementation, described in chapter 4.

3.1 Components of Self-Sovereign Identity

In this section, we present the main components of the SSI concept, as defined by the W3C Community Group [60]. The SSI concept consists of two layers, the Identity Layer and the Credentials Layer. We present a distilled version of each layer's main components: first, the Identity Layer, and second the Credentials Layer components.

3.1.1 Identity Layer

The Identity Layer is the base layer of the SSI ecosystem. On this layer, entities can create and proof control over globally unique DIDs. Having a unique DID allows entities to interact with other entities of the SSI ecosystem.

DIDs follow a specific format and identify an entity, the DID Subject. DIDs point to a DID Document on a DID compatible verifiable data registry, for example, a compatible blockchain, a distributed ledger, or a decentralized network. In other words, the Identity Layer is a global key-value database. Different DID Methods exist; each defines how users can create and manage respective DIDs and which data registry stores DID Document data.

Two main specifications describe DIDs and serve a basis for this section:

- *A Primer for Decentralized Identifiers* [56] is an introduction to DIDs and their basic concepts. It gives an introduction to the DID Format, DID Documents and DID Methods. Furthermore, it introduces some of the privacy relevant concepts of the SSI concept, pairwise DIDs, off-chain private data in form of VCs and selective disclosure in the form of VPs. We explain VCs and VPs in 3.1.2.
- In *Decentralized Identifiers (DIDs) v1.0* [62] the authors present a technical view. They specify a common data model, a URL format for DID Methods, and a set of operations for DIDs, DID Documents, and DID Methods. The specification

is derived from use-cases in active development [59] and is but is still under development. However, it is intended to become a W3C Recommendation.

In the following subsections the DID Format, the DID Method, and the DID Document are explained.

DID Format

As seen in figure 3.1, the DID Format follows a similar pattern as URNs [39]. The DID Format consists of a DID specific scheme, a DID Method and a DID Method-specific unique string. The DID specific scheme identifies the string as a DID. The DID Method defines how the DID interacts with a specific decentralized storage option, explained in detail in the following *DID Methods* subsection. The DID Method-specific unique string is the unique identifier of the DID in the DID Method's namespace.

$$\overbrace{\text{did}}^{\text{Scheme}}:\underbrace{\text{example}}_{\text{Method}}:\overbrace{\text{123456789abcdefghijk}}^{\text{DID Method Specific String}}$$

Figure 3.1: The DID syntax. Adopted from [62]

In section 5.1.1 we present an example DID Format implementation.

DID URL

The DID URL is a DID extension to identify a resource. Therefore, a path, query, fragment, or parameters are appended to a DID. A DID URL allows, for example, to point to a specific part of a DID Document. In figure 3.2, a DID extended by a fragment pointing to a specific public key in its DID Document can be seen.

$$\text{did:example:123456789abcdefghijk}\overbrace{\text{\#public-key-1}}^{\text{Fragment}}$$

Figure 3.2: A DID extended by a fragment. Adopted from [62]

DID Methods

Multiple DID Methods exist; each defines how to interact with their DIDs and DID Documents stored on a DID-compatible verifiable data registry. The DID Method specification defines how a DID and its respective DID Document are created, resolved, and managed. Each DID Method specification must at least specify its method name and a method-specific syntax for the DIDs unique identifier. Furthermore, it must define how a DID Document can be interacted with using the four CRUD operations:

1. **C**reate: How does the client create a DID and its associated DID Document, including cryptographic operations necessary to establish proof of control.
2. **R**ead: How does the client resolve a DID to a DID Document and verify the documents validity.
3. **U**pdate: How does the user update the DID Document and what constitutes an update, including all cryptographic operations necessary to establish proof of control.
4. **D**eactivate: How does the user deactivate a DID and its associated DID Document, including all cryptographic operations necessary to establish proof of control.

The prove of control usually consists of a dynamic prove over control over the private key associated with the DID or public keys registered in the created DID Document, explained in *DID Binding*.

All DID Methods are listed with their associates specifications in the DID Method Registry [58], a GitHub repository. To register a new DID Method, the developer has to provide at least an experimental version of the DID Method, a specification and has to create a GitHub pull request containing a link to the specification.

In section 5.1 we present an example DID Method implementation.

DID Documents

DIDs point either directly to a DID Document or a link stored on a DID compatible verifiable data registry. The link, in turn, points to a DID Document stored on a publicly available storage. The DID Document's central purposes are to allow for the proof of ownership over a DID, to allow others to act on behalf of the DID Subject and to express communication canals to the DID Subject or associated web services.

DID Documents follow the JSON-LD format specification [46]. The JSON-LD format allows storing Linked Data in JSON objects. Linked Data is a data format in which data is interlinked with other data in a computer- and human-readable form. "It allows an application to start at one piece of Linked Data, and follow embedded links to other pieces of Linked Data that are hosted on different sites across the Web" [55]. Furthermore, it provides a mechanism for typed data. The schemas the properties in a JSON-LD object must adhere to are usually set in the object's *context* property.

In conclusion the JSON-LD format allows to set a standardised schema for all DID Documents and to link to network resources in a standardized manner. Therefore, all DIDs Documents contain a link to the same context document [57] in their *context* property.

DID Documents typically include the following components:

1. A required DID Subject.
2. An optional set of public keys.
3. An optional set of authentication methods.
4. An optional set of delegates.
5. An optional set of service endpoints.
6. Two optional timestamps indicating the creation and last update date.
7. An optional JSON-LD signature.

In the following we describe DID Document's components relevant to the mvSSIn implementation.

The DID Subject is the entity described by the DID Document and identified by the DID. The DID is stored in the *id* property, as seen in figure 3.3

```
{
    "@context": "https://www.w3.org/ns/did/v1",
    "id": "did:example:21tDAKCERh95uGgKbJNHYp"
}
```

Figure 3.3: Example JSON-LD document, containing the *id* property. Adapted from [62]

The optional set of public keys contains one or more public cryptographic keys that third parties can use to authorize and authenticate interactions with the DID Subject or associated entities. DID Documents store each public key as an object in an array in the *publicKey* property, as seen in 3.4. Each public key object must contain *id*, *controller*, *type*, and specific public key properties. The *id* property is the unique identifier of the public key. The *controller* property must be a DID. It identifies the DID controlling the corresponding private key. The *type* property must be a public key type register in the *Linked Data Cryptographic Suite Registry* [65]. The specific public key properties are specific to each public key type, for example in case of a *RsaVerificationKey2018* key type a *publicKeyPem* property is part of the public key object. Each DID Method must specify how keys can be revoked. The specification proposes two approaches, removing the key from or storing a revocation list in the DID Document.

The optional set of service endpoints stores one or more network addresses, on which a web service operates on behalf of the DID Subject, stored in the *service* property, as seen in figure 3.5. The web services allow to communicate with the DID Subject or associated entities. Each service endpoint must contain *id*, *type*, and *serviceEndpoint* properties. The *id* property must be an URI uniquely identifying the service endpoint. The *type* property defines the type of the service endpoint, a broad array of possibilities exists. The *serviceEndpoint* property must be the URL of the web service. A service

```json
{
  "@context": ["https://www.w3.org/ns/did/v1", "https://w3id.org/security/v1"],
  "id": "did:example:123456789abcdefghi",

  "publicKey": [{
    "id": "did:example:123#_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A",
    "type": "JwsVerificationKey2020",
    "controller": "did:example:123",
    "publicKeyJwk": {
      "crv": "Ed25519",
      "x": "VCpo2LMLhn6iWku8MKvSLg2ZAoC-nlOyPVQaO3FxVeQ",
      "kty": "OKP",
      "kid": "_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A"
    }
  }, {
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }, {
    "id": "did:example:123456789abcdefghi#keys-2",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:pqrstuvwxyz0987654321",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }, {
    "id": "did:example:123456789abcdefghi#keys-3",
    "type": "Secp256k1VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyHex": "02b97c30de767f084ce3080168ee293053ba33b235d7116a3263d29f1450936b71"
  }],
}
```

Figure 3.4: Example JSON-LD document, containing a set of public keys. Adapted from [62]

endpoint can, for example, enable the dynamic proof of control over a DID's private key, as described in *DID Binding*.

```
{
   "@context": "https://www.w3.org/ns/did/v1",
   "id": "did:example:123456789abcdefghi",
   "service": [
      {
      "type": "VerifiableCredentialService",
      "serviceEndpoint": "https://example.com/vc/"
    }, {
      "id": "did:example:123456789abcdefghi#openid",
      "type": "OpenIdConnectVersion1.0Service",
      "serviceEndpoint": "https://openid.example.com/"
    }, {
      "id": "did:example:123456789abcdefghi#vcr",
      "type": "CredentialRepositoryService",
      "serviceEndpoint": "https://repository.example.com/service/8377464"
    }]
}
```

Figure 3.5: Example DID Document, containing a set of service endpoints. Adapted from [62]

Key Rotation

Key rotation is the process of replacing an old key pair with a new key pair. It is regarded as best practice in cryptography, to avoid compromised keys [2, 37, 5]. The SSI concept allows rotating keys by updating the DID Document. Therefore, the DID Subject generates a new key pair, replaces the corresponding old public key in the DID Document's *publicKey* property. The DID Subject can then sign interactions with the new private key.

VCs signed with a replaced private key can still be verified since the issuer's old DID Document dating back to the creation date of the VC can be retrieved containing the old public key.

DID Binding

We identify three types of DID bindings. The binding of a DID to a DID Document, a DID to a private key and a DID to a real-world identity, explained in section 3.1.2.

To verify the binding between a DID and a DID Document, an entity resolves the DID to its DID Document, and checks that the *id* property value is equal to the DID. The process to resolve a DID is described in the associated DID Method's specification.

Proving control over a DID's private key is possible statically and dynamically. For both approaches, either the DID Document must store a public key, or the DID itself must be a public key representation. In the static way, the DID Subject signs the DID Document with its private key. The control over the private key can thereby be verified for the time no later than the registration or update of the DID Document. To prove control in a dynamic way, two steps have to be undertaken:

1. Send challenge in the form of a nonce to the DID Subject.

2. Verify the returned signed challenge against the DID's public key.

A more sophisticated way of proving control is defined in *Self-Issued OpenID Connect Provider DID Profile* [66]. SIOP is an adaptation of an existing authentication protocol. We implement an SIOP process and describe it in more detail in section 5.3.3.

The DID Subject can realize a binding between a real-world identity and its DID to a limited extent by adding a service endpoint to the DID Document that point to a real-world identity. Since a trusted third party does not confirm this, the reliability of such an association is limited. On the Credentials Layer, such an association can be attested by a trusted third party to create a reliable link between a DID and a real-world identity.

Pairwise DIDs

The pairwise DID paradigm prescribes an approach in which for each interaction or service an entity interacts with a new DID is created. This approach aims to reduce the correlatability of an entity's actions. We discuss this approach with field experts in 7.4.3.

3.1.2 Credentials Layer

On top of the Identity Layer, the Credentials Layer allows entities of the SSI ecosystem, to make attestations about each other. The concept is independent of DIDs but was developed in conjunction with DIDs. It defines a method to express simple statements or complex documents in digital and machine-verifiable form. These digital documents are called Verifiable Credentials (VCs).

A VC stores one or more claims about a DID. Usually, an issuer creates and shares a VC with the entity described in the VC. The entity, called holder, stores the VC and can present it in a privacy-preserving manner to third parties, so-called verifiers. When sharing a VC it is usually referred to as a VP.

Two main specifications describe VCs and serve a basis for this section:

- *Identity Credentials 1.0* [63] expounds how a entity's digital identity can be expressed as a set of credentials containing claims. Furthermore, it describes a set of mechanisms for issuing and requesting credentials.
- *Verifiable Credentials Data Model 1.0* [64] builds on the previously introduced specification by defining a credential data model and a mechanism to express these credentials in a cryptographically secure, privacy-preserving, and machine-verifiable way on the web.

In the following subsections, we explain the Credentials Layer's components, claims, VCs, and VPs.

Claims

Claims are the centerpiece of VCs. A claim is a statement about a subject. A subject is a thing about which claims can be made. In this thesis, we also referred to the subject as an entity. A statement consists of a property and a value. In essence, a claim is a subject-property-value relationship, as seen in figure 3.6.

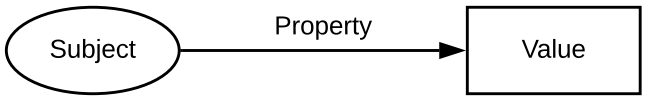


Figure 3.6: The basic structure of a claim. Adapted from [64]

In figure 3.7 an example claim can be seen. The claim expresses that Bob is an alumnus of the Technical University of Munich. The subject *Bob* has the property *alumniOf* and the property the value *Technical University of Munich*.

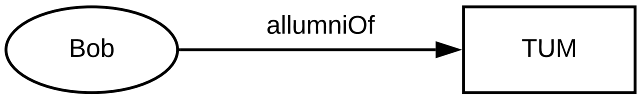


Figure 3.7: Example claim. Adapted from [64]

A graph of claims combines multiple claims. For example, in the SSI context, the claim's subject would usually be a DID. To create a valid claim assigning a property and value to a real-life entity, first, a claim binding the DID to the real-life entity is needed. A single claim can encapsulate both claims, as seen figure 3.8. This example also shows that one claims value can be the subject of another claim.

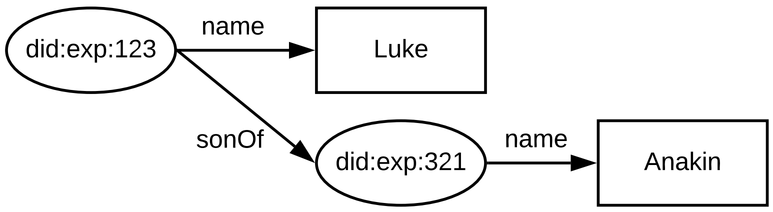


Figure 3.8: Example claim graph

Verifiable Credentials

A VC is a set of one or more claims bundled with their respective proofs and associated metadata, as seen in figure 3.9. The issuer makes all claims contained in a VC. The issuer's DID and his public key, as well as further information, are included in the VC's metadata. The third component of a VC are the proofs associated with each claim. Each proof is the issuer's cryptographically verifiable signature of a respective claim. Thus, "a verifiable credential is a [shareable] set of tamper-evident claims and metadata that cryptographically prove who issued it" [64].

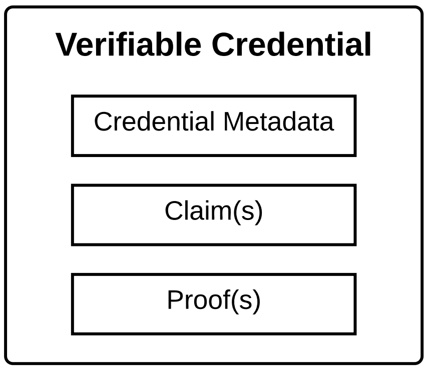


Figure 3.9: The basic components of a VC. Adapted from [64]

A VC can also be displayed as a graph, as seen in the credential graph and credential proof sections of figure 3.11. A corresponding JSON representation can be seen 3.10. The example VC contains an *alumniOf* claim, as previously presented in figure 3.7. To create an association between the claim and a real-life entity, an association between the claim's subject DID, and the real-world entity is necessary. Furthermore, to trust the credential, the issuer DID needs to be a well know DID associated with the Technical University of Munich. We explain how such an association is created in section 3.3. To verify the integrity and authenticity of the VC, a verifier first resolves the issuer's DID. Secondly, he verifies the claim's proof against the public key linked to in the *creator* property of the VC and stored in the resolved DID Document.

```json
{
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuer": "did:exp:333",
  "credentialSubject": {
    "id": "did:exp:123",
    "alumniOf": {
      "id": "Technical University of Munich"
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2020-01-01T12:21:22Z",
    "creator": "did:exp:333#privateKey1",
    "nonce": "abc123abc123abc",
    "signatureValue": "AbC123=[...]321aBc"
  }
}
```

Figure 3.10: Example VC JSON

Verifiable Presentations

A VP is a packaged subset of data of one or more VCs. VPs package data in such a way that the original authorship and the integrity of the data are verifiable. Directly forwarded VCs are also VPs. However, a presenter of a VP usually signs it with his private key so that a third party can verify who presented the VP.

As a privacy feature, VPs allow an entity to share only selected portions of VCs in its possession. As an example, someone in possession of a VC encapsulating a passport can share only his age with a third party and not all the information stored in the document. Additionally, some SSI implementations and SDKs further extend the selective disclosure with Zero Knowledge Proofs (ZKPs). ZKPs allow proving a statement without sharing anything else [44]. This permits, for example, to prove that the age stored in a passport VC is over a certain threshold. The exact workings of ZKP are out of scope for this master's thesis.

A VP graph can be seen in figure 3.11. To verify the integrity and the authorship of the represented VP, first, a verification of the contained credential is necessary, explained in *Verifiable Credentials*. Additionally, the VP proof is verified. Therefore the verifier resolves the presenter's DID. Following, he verifies the VP's proof against the public key linked to in the *creator* property of the VC and stored in the resolved DID Document.

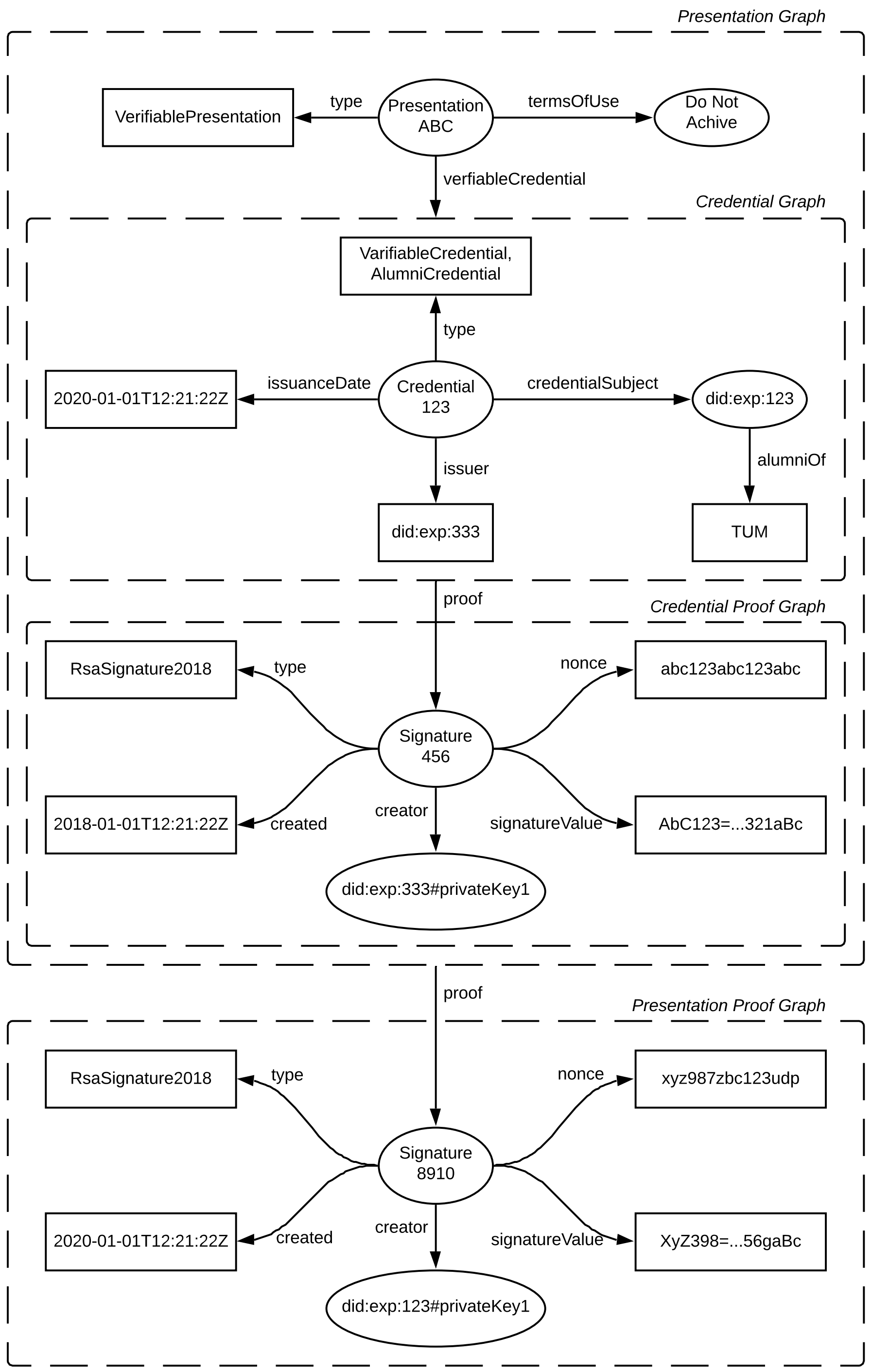


Figure 3.11: Example of a Verifiable Presentation graph. Adapted from [64]

3.2 Roles in Self-Sovereign Identity

In this section, we present the fundamental roles of the SSI ecosystem. A role is a set of capabilities an entity can have. The roles are not mutually exclusive; an entity can have one or any number of roles. We present the roles in regards to the two layers of SSI. First, we present the roles of the Identity Layer and second, the roles of the Credentials Layer.

Two Identity Layer roles are defined, the DID Subject and the DID Verifier [63]. Both roles rely on a shared verifiable data registry.

1. The **DID Subject** creates a DID, usually by creating a DID Document on the verifiable data registry. The DID Document has to follow the schema prescribed by the verifiable data registry. Furthermore the DID Subject responds to prove of control challenges by the DID Verifier.
2. The **DID Verifier** verifies the DID Subject's control over a DID. The verification either happens statically, by relying on the DID Document data or dynamically, by sending a challenge to the DID Subject, as described in *DID Binding*.

The Credentials Layer extends the set of roles defined for the Identity Layer [64]. Three roles are defined for the Credentials Layer, the issuer, the holder, and the verifier. All three rely on the shared verifiable data registry, as seen in figure 3.12. Each Credentials Layer role extends on the roles defined for the Identity Layer. Each role except the verifier is a DID Subject, and each role except the holder is a DID Verifier.

1. The **issuer** verifies the holder's control over a DID and issues VCs for the holder. The VC has to follow the schemas prescribed by the verifiable data registry.
2. The **holder** acquires, stores, and presents VPs to a verifier.
3. The **verifier** requests and verifies VPs from the holder. To verify a VP, the **verifier** checks the VP conformity to the schemas prescribed by the verifiable data registry, the holder's, and issuer's control over the DIDs and private keys associated with the VP as well as the correctness of the signatures as described in *Verifiable Presentations*.

3.3 Trust Architecture in Self-Sovereign Identity

In this section, we describe the trust architecture of the SSI concept in regards to the two layers of SSI. The trust architecture interlinks with the roles described in section 3.2. First, we present the trust architecture of the Identity Layer and second, the trust architecture of the Credentials Layer.

The Identity Layer creates trust over the control over DIDs. Including the trust over the binding of a DID and a DID Document. An entity can proof control over a DID either

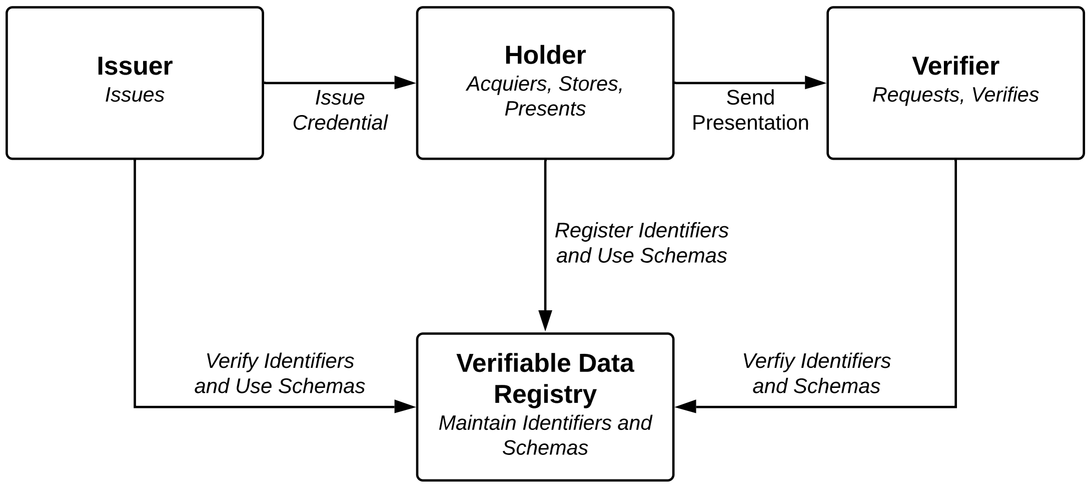

Issuer
Issues
Issue Credential
Holder
Acquiers, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas
Verfiy Identifiers and Schemas

Figure 3.12: The roles and information flow of SSI. Adapted from [64]

passively or dynamically, as described in *DID Binding*. Having a unique identifier allows entities to partake in the SSI ecosystem. Furthermore, an entity can establish a binding between a DID and its real-world persona on the Identity Layer in two ways:

1. An entity can advertise its DID on a web service linked to its real-world persona, for example, on a web site or social media page. The trust in the binding is dependent on the trust towards the validity of the information advertised on the web service. A malicious entity can compromise this trust. For example, by setting up a web service in another entity's name or by manipulating a web service to advertise an incorrect DID.

2. An entity can set its web service's URL as a service endpoint in the DID's associated DID Document. This approach requires that the linked web service provides a possibility to request proof of control over the DID. This approach suffers, like the previous approach, from malicious web services spoofing their identity.

A combination of both approaches yields the highest trust in the validity of the binding. In this case, both the web service and the DID Document would have to be compromised. Both approaches are only applicable for entities with well-known associations to a web service. Entities with no well-known association with a web service can create no reliable connection between real-world identity and a DID on the Identity Layer. This leads to the reliance on well-know entities to issue VCs containing claims about the DID holder to prove a binding between a DID and the holder's real-world identity.

On the Credentials Layer, trust over the validity of claims, attested in VCs, is created. Using claims, holders can trustfully prove the properties of their identity to other entities. The trust in a claim is dependent on multiple factors:

1. The trust in the binding of the issuer with a real-world identity described previously.
2. The trust in the correctness of the claims.

The trust in the correctness of the claims is dependent on the trust towards the issuer's capabilities. The issuer has to verify the correctness of the claims to be attested; in other words, the binding between an entity's DID and identity attributes. We think this can be decomposed in the ability to verify the binding between a DID and a real-world entity. As well as the ability to verify if a real-world entity has the property attested in the claim.

The described Credentials Layer trust architecture can, for example, be undermined by an entity colluding with an issuer. If the issuer attests incorrect claims on behalf of the credential holder, no trace of wrongful association can be detected in the SSI system.

As stated in "A survey on essential components of a self-sovereign identity," the trust architecture leads to issuers having a centralized position of trust in the SSI system.

4 Requirements

After familiarizing with the fundamentals of the SSI concept, the next step of developing a prototypical mvSSIn is to conduct a requirements analysis. The initial requirements are derived from the fundamentals and two basic use-cases, capturing the basic features of the Identity and the Credentials Layer. The two use-cases we cover are a single sign-on use-case and a credentialised access management use-case based on the life-long, recipient-managed credentials use-case [59]. We further improve and extend the requirements by field testing the implementation in a user survey, as described in chapter 7.

This section presents the requirements of our prototypical mvSSIn implementation. The requirements are divided into two sections, Functional Requirements (FR) and Non-Functional Requirements (NFR). We follow the keywords defined by the ISO to indicate the relevance of each requirement: *shall* indicates a requirement, *should* indicates a recommendation and may indicates that something is permitted [30].

4.1 Functional Requirements

In the following, we list the FRs describing the behavior of the mvSSIn.

FR01: Create DIDs

The mvSSIn shall allow users and institutions to create DIDs.

FR02: Delete DIDs

The mvSSIn shall allow users to delete DIDs.

FR03: DID Login

The mvSSIn shall allow users to log in to web services using a DID.

FR04: Request VCs

The mvSSIn shall allow users to request VCs.

FR05: Issue VCs

The mvSSIn shall allow institutions to issue VCs.

FR06: Present VPs

The mvSSIn shall allow users to present VCs.

FR07: Verify VPs

The mvSSIn shall allow institutions to verify VCs.

FR08: Out of wallet storage of VC

The mvSSIn may allow users to store their VCs outside of the DID Wallet. Users may either set a cloud storage controlled by the user to store their VCs or may store VC on an Identity Hub [27].

FR09: Key Recovery

The mvSSIn may allow users and institutions to backup and recover their keys.

FR10: Key Rotation

The mvSSIn may allow institutions to rotate their keys. Institutions may rotate their keys manually or may set an automation.

4.2 Non-Functional Requirements

In the following, we list the NFR specifying the overall characteristics and driving the technical architecture of the mvSSIn.

NFR01: Simple UI

The user shall be able to easily set up the DID Wallet, create a DID, log in with a DID, request a VC, and present a VP. All actions shall provide logical feedback to the user. DIDs shall be easily distinguishable and shall display a reference to the web services they were used with.

NFR02: Explanatory UI

The DID Wallet may provide explanations to the user on how to use the application.

NFR03: Use the ETHR DID Method

The mvSSIn shall support the ETHR DID Method [51], explained in 5.1. It should reuse the software components provided by uPort and the SSI community.

We added this requirement based on an analysis of the software landscape of the SSI ecosystem. Due to time constraints, we needed to rely on the reuse of free-to-use open-source software components. We found that the ETHR DID Method has an active community, extensive specification, documentation, and free-to-use open-source software components.

NFR04: Extend web services

The mvSSIn should be implemented in such a way that existing web services can reuse the approach to integrate the login with DID, the issuance of VCs, and the verification of VPs in the same or a similar manner.

NFR05: Paired DIDs

The mvSSIn may implement paired DIDs. Paired DIDs are DIDs that are created for each service or interaction.

NFR06: DID communication

The mvSSIn may implement functionality to allow communication between DID Subjects. This type of communication allows the establishment of a communication channel based solely on the knowledge of a DID.

5 Implementation

In this chapter, we present the prototypical mvSSIn we develop. The implementation is one of the artifacts produced in the Design Cycle. We develop it based on the requirements defined in chapter 4 and in conjunction with the user journey described in chapter 6. We evaluat the implementation in a user survey described in chapter 7. We use the feedback from the user survey to improve the requirements and, in turn, improve the implementation. The implementation's technical success, indicates the technical feasibility of a mvSSIn, especially in regards to the maturity of reusable free-to-use open-source software components.

The prototypical mvSSIn consists of a smartphone app, the DID Wallet, and two web services. We realize the smartphone app using React Native [19], a JavaScript library for iOS and Android app development. We deploy the web services on a server running Node.js [35]; a JavaScript runtime, running express.js [33]; a web application framework. All three software artifacts rely on the ETHR DID Method.

The DID Wallet allows users to create, manage, and use ETHR DIDs, as well as to store, manage, and present VCs. The two web services, each serve a web site consisting of several web pages, each mimicking an institution's online presence. The web services are the counterpart to the DID Wallet; they allow the user to login with a DID, and to request and present VCs. The implementation models a 1-n relationship between the institutions and DID Wallet users communicating with the institutions via the web service. In figure 5.1, we depict a high-level view of the mvSSIn.

In the following, we first present the DID Method used in the implementation, ensued by the primary software components. Furthermore, we present the three services we develop and how they integrate the software components. And finally, we explain the interactions between the three software services.

5.1 ETHR DID Method

In this section, we present the DID Method used in the implementation, the Ethereum based ETHR DID. It was developed by the uPort Team and is defined in *ETHR DID Method Specification* [51]. It is compliant with the *Decentralized Identifiers (DIDs) v1.0* [62] specification. The on-chain implementation of the ETHR DID Method; the ETHR DID registry [50], allows for the retrieval of data from and storage of data on the Ethereum

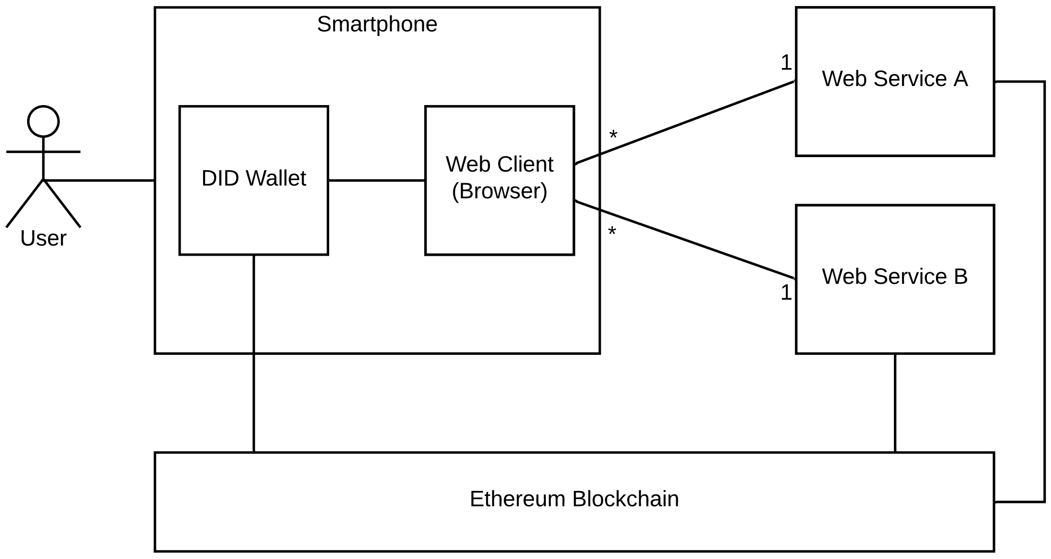


Figure 5.1: A high-level view of the mvSSIn

blockchain. The ETHR DID registry follows the ERC1056 standard [6]. The standard describes creating and updating Ethereum based identities. It especially improves on the previous ERC-725 standard [54], by removing the fee for identity creation.

In the following, we present the ETHR DID Format and the ETHR DID's implementation of the four minimal DID Method operations.

5.1.1 DID Format

A ETHR DID consists of three parts:

1. The ETHR specific prefix: "did:ethr".
2. An Ethereum network identifier: "mainnet" | "ropsten" | "rinkeby" | "kovan". It determines on which Ethereum network the DID is anchored. If none is used, the DID is anchored on the Ethereum mainnet.
3. A method specific identifier, a hex encoded Ethereum address, for example: "0xb9c5714089478a327f09197987f16f9e5d936e8a".

A full ETHR DID can be seen in the *id* property in figure 5.2.

5.1.2 Create

To create a new ETHR DID, any Ethereum smart contract or key pair account can be used as an identity. In other words, any valid hex-encoded Ethereum address can be used as the method-specific identifier, as described in section 5.1.1. Alternatively, a new key pair is generated. At this stage, no interaction with the Ethereum network is required. This means that no fee is due for the creation of an ETHR DID. No on-chain registration is needed, based on the assumption that it is impossible to brute force an Ethereum private key. But the ETHR DID registry, an ERC1056 compliant smart contract, allows for additional attributes to be stored on-chain, see section 5.1.3.

5.1.3 Update

The update operation allows to add or change attributes of a DID Document. These attributes consist, for example, of the identity owner, delegates, or service endpoints, as described in 3.1.1. On each update, the ETHR DID registry stores a change event in the current Ethereum block. The event contains the values added to or changed in the DID Document. If a previous change exists, it additionally contains the number of the Ethereum block of the last change event. Finally, the ETHR DID registry stores the Ethereum block number of the most recent change. The ETHR DID registry stores it in its change mapping, where it is mapped to by it's associated DID. This allows retrieving the block number, storing the last change to the DID Document, on the invocation of the next update or read operation.

5.1.4 Read

The ETHR DID Method's read operation resolves valid ETHR DIDs to a DID Document. The operation uses read-only functions of the ETHR DID registry. In the ETHR DID registry, the block number of the Ethereum block containing the last change event associated with an ETHR DID is retrieved from the change mapping. If no mapping can be found, no supplementary on-chain data exists. In this case, the DID Document can be created solely by inserting the ETHR DID and the underlying Ethereum address into a predefined pattern. Such a document can be as seen in figure 5.2.

But if a mapping is found, the block indexed in the mapping is searched for the most recent change event associated with the DID. The attribute contained in the change event is appended to or updated in the DID Document generated from the predefined pattern. If the change event contains a previous change event block number, the process is repeated until the event containing no previous change event block number is reached.

The ETHR DID Document constructed in this process follows the DID JSON-LD context, but uses an *ethereumAddress* property instead of a *publicKeyHex* property, as seen in 5.2.

5.1.5 Delete

The delete operation allows for the DID to be deleted. Two cases have to be distinguished:

1. If no additional attributes were written to the Ethereum blockchain, it is sufficient to delete the associated private key.
2. If attributes were written to the Ethereum blockchain, the registry's update function is called, and the DID Document's *id* property is set to "0x0". This indicates that the DID Document is no longer valid.

```json
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:ethr:0xb9c5714089478a327f09197987f16f9e5d936e8a",
  "publicKey": [
    {
      "id": "did:ethr:0xb9c5714089478a327f09197987f16f9e5d936e8a#owner",
      "type": "Secp256k1VerificationKey2018",
      "owner": "did:ethr:0xb9c5714089478a327f09197987f16f9e5d936e8a",
      "ethereumAddress": "0xb9c5714089478a327f09197987f16f9e5d936e8a"
    }
  ],
  "authentication": [
    {
      "type": "Secp256k1SignatureAuthentication2018",
      "publicKey": "did:ethr:0xb9c5714089478a327f09197987f16f9e5d936e8a#owner"
    }
  ]
}
```

Figure 5.2: Example ETHR DID Document. Adapted from [51]

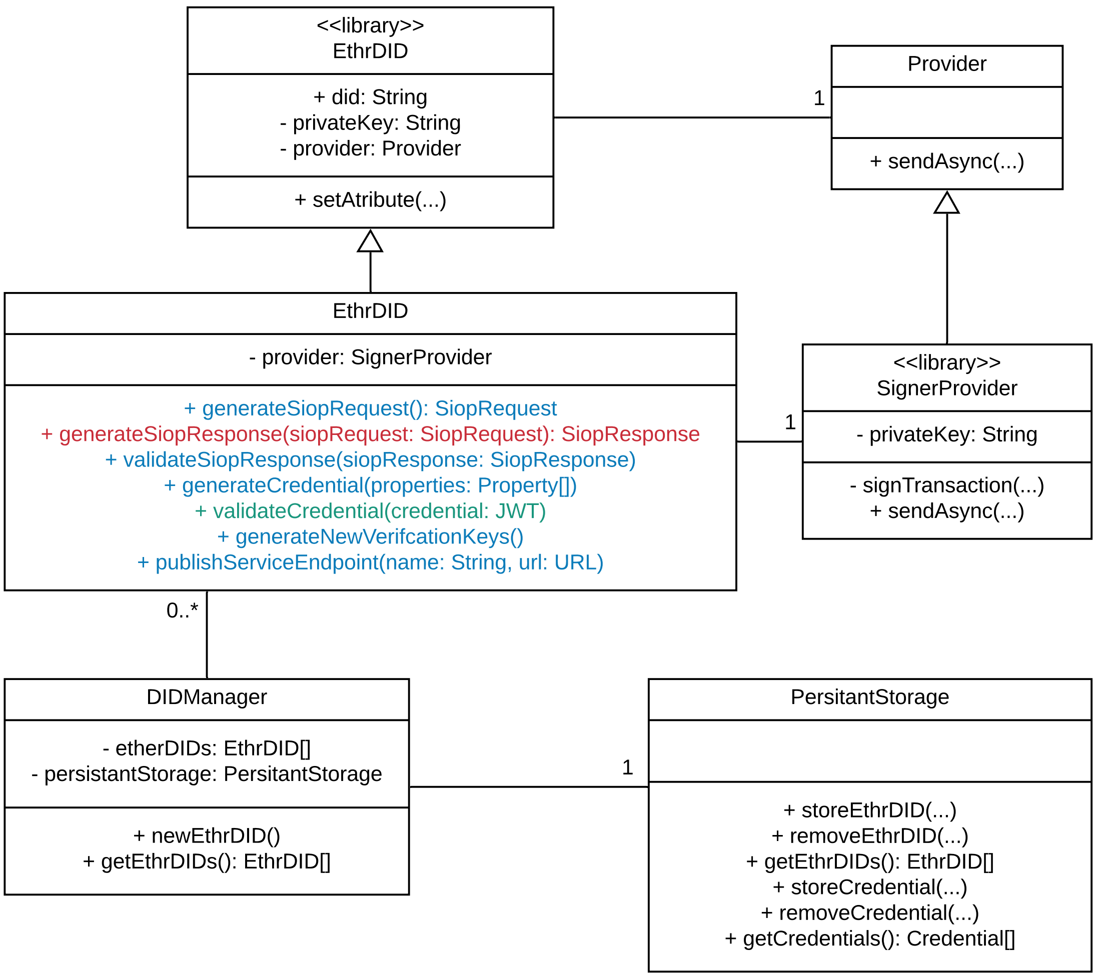
5.2 Components

In this section, we present the main software components we reused throughout the mvSSIn implementation. These components consist of JavaScript modules and classes. Corresponding structural overviews can be seen in figure 5.3 and 5.4. In figure 5.3, the red methods are associated with the DID Wallet, blue methods are associated with the web service A, and green methods are associated with the web service B.

5.2.1 SignerProvider

Multiple ways exist to interact with the Ethereum blockchain. To avoid running a local Ethereum node requiring computing power and storage space, we decided to use a third-party provider, Infura [28]. Infura provides an API that allows interaction with an Ethereum node using JSON-RPC over HTTP.

To forward our blockchain transactions to Infura's API we rely on the third-party Signer-Provider [14] module. The module is compliant with the Web3 provider specification [1].



<<library>>
EthrDID
+ did: String
- privateKey: String
- provider: Provider
+ setAtribute(...)
1
Provider
+ sendAsync(...)
EthrDID
- provider: SignerProvider
+ generateSiopRequest(): SiopRequest
+ generateSiopResponse(siopRequest: SiopRequest): SiopResponse
+ validateSiopResponse(siopResponse: SiopResponse)
+ generateCredential(properties: Property[])
+ validateCredential(credential: JWT)
+ generateNewVerifcationKeys()
+ publishServiceEndpoint(name: String, url: URL)
1
<<library>>
SignerProvider
- privateKey: String
- signTransaction(...)
+ sendAsync(...)
0..*
DIDManager
- etherDIDs: EthrDID[]
- persistantStorage: PersitantStorage
+ newEthrDID()
+ getEthrDIDs(): EthrDID[]
1
PersitantStorage
+ storeEthrDID(...)
+ removeEthrDID(...)
+ getEthrDIDs(): EthrDID[]
+ storeCredential(...)
+ removeCredential(...)
+ getCredentials(): Credential[]

Figure 5.3: Structure of the DID components

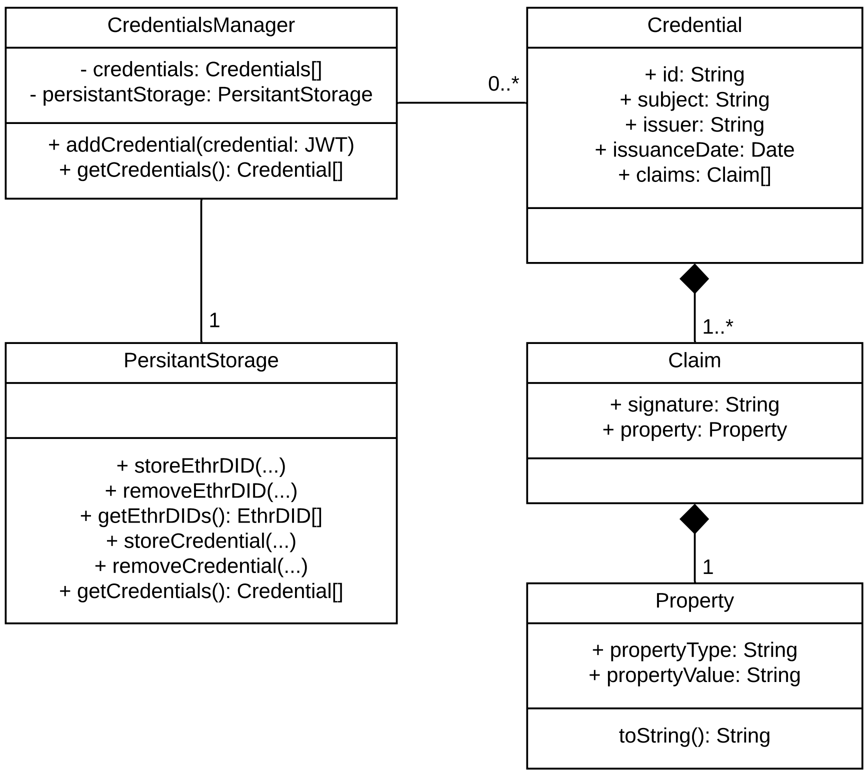


Figure 5.4: Structure of the credential components

A Web3 provider must implement the *sendAsync* method, that handles the JSON-RPC. Furthermore, we provide the SignerProvider with the DID's private key to sign all transactions. In other words: "Using a configured Web3 Provider, the application can request signatures, estimate gas, and submit transactions to an Ethereum node."[1]

A prerequisite for transactions to Ethereum blockchain is that the Ethereum address, associated with the DID and the private key used to sign the transactions, has a sufficient balance to cover the transaction costs. However, in our mvSSIn, only the institutions write to the blockchain; therefore, users do not need any balance.

5.2.2 PersistantStorage

The PersistantStorage module allows to store, delete, and retrieve DIDs and VC persistently. The smartphone implementation is an abstraction of the AsyncStorage [18] module provided by React Native. The web service implementation stores the information in a JSON document.

To store DIDs and VCs, they are passed to the PersistantStorage module as their string representations. On retrieval, the PersistantStorage module returns their associated instantiated objects.

5.2.3 EthrDID

The EthrDID class is an extension of the EthrDID class provided in uPort's Ethr-DID library [49]. uPort's library allows to create and manage ETHR DIDs, as described in section 5.1, represented as JavaScript objects. Their implementation of the EthrDID class allows to create new DIDs, where the DID string and private key are generated on instantiation or a previously generated private key is passed. Furthermore, an instantiated EthrDID object allows to create an associated on-chain DID Document and to update or to add new attributes to the DID Document. Additionally, the EthrDID object also allows to sign and to verify messages from other DIDs. It stores the DID string, the private key, and a reference to the SignerProvider module to communicate with an Ethereum node.

In our implementation, we extended the base EthrDID class, to allow for the SIOP DID Profile we explain the in 5.3.3. Therefore, it internally stores a representation of the public key compliant to the SIOP DID Profile. Furthermore, the EthrDID class exposes methods for the generation of SIOP requests and responses, as well as methods for the validation of SIOP responses.

Additionally, we expose methods for the generation and validation of VCs.

Furthermore, we added two convenience functions based on the base EthrDID class's ability to add new attributes to its associated DID Document. The first one allows publishing service endpoints to the DID Document. The second one generates new key pairs to sign VCs and publishes the public key to the DID Document. Verifiers can then retrieve the published public key from the DID Document to check VP signatures.

5.2.4 DIDManager

The DIDManager class instantiates to a singleton object. It performs the following three actions:

1. On instantiation, it loads all previously-stored EthrDIDs from the PersistantStorage module. They are held in an internal array for fast access.
2. It instantiates and stores new EthrDID objects. On instantiation, the EthrDID object is stored in the internal array, and its string representation is persistently stored using the PersistantStorage module.
3. It provides access to all previously generated and stored EthrDIDs.

5.2.5 Credential

An instantiated Credential object encapsulates a VC, as described in section 3.1.2. A Credential object has four properties:

1. *subject*: The subjects DID string.
2. *issuer*: The issuers DID string.
3. *issuanceDate*: The issuance date of the credential.
4. *claims*: Array of Claim objects.

5.2.6 Claim

An instantiated Claim object encapsulates a claim, as described in section 3.1.2. A Claim object has two properties, the claim's property, and a signature.

The claim's property is represented as a Property object. The Property object's properties consist of a *propertyType* and a *propertyValue*. The *propertyType* is the string representation of the claims property type, for example, "isAllumniOf". The *propertyValue* is the string representation claims property value, for example, "Technische Universität München".

The Claim's *signature* property stores the issuer's signature.

5.2.7 CredentialsManager

The CredentialsManager class instantiates to a singleton object. It performs the following three actions:

1. On instantiation, it loads all previously-stored Credentials from the PersistantStorage module. The CredentialsManager holds them in an internal array for fast access.
2. It allows to instantiate and store Credentials using their JWT representation. The CredentialsManager stores the new Credential object in an internal array and its string representation is persistently stored using the PersistantStorage module.
3. It provides access to all previously received and stored Credentials.

5.3 Services

In this section, we present the three services we implement. Each service relies on a subset of the components explained previously. In the following, we present each service, which component it implements, and the service it thereby provides. Furthermore, we introduce each service's UI. Finally, we explain how the three services communicate with each other.

5.3.1 DID Wallet

The DID Wallet implements the following components:

1. The DIDManager and the EthrDID, allow the user to create ETHR DIDs. When logging in to a new web service, they allow for the automatic generation of a paired DID. The generated EthrDID objects allow for logging in with DID the DID SIOP Profile.
2. The CredentialsManager and it's associated classes, allow the user to receive, store, and share VCs.
3. Deep Links [4, 25], allow for data to be passed between the DID Wallet and a browser, and vice versa. This facilitates communication between the two web services and the DID Wallet.

The DID Wallet's UI consists of four screens:

1. The **Password Screen**, allows the user to set a password on the initial startup and requires the user to type in their password to unlock the app on all the following startups.
2. The **Home Screen** allows the user to navigate to the identifier and the credentials screen.
3. The **Identifier Screen**, allows the user to create new ETHR DIDs, delete ETHR DIDs, and to navigate to the **Credential Screen**. Furthermore, it allows users to select an ETHR DID in the login with DID process.

4. The **Credential Screen**, allows the user to view his stored VCs and delete stored VCs. When accessed via the **Identifier Screen**, it displays only the VCs associated with the ETHR DID selected in the navigation path. Furthermore, it allows the user to select a VC in the present VC process.

5.3.2 Web Services

Both web services have in common that they implement the DIDManager and the EthrDID. They both create one EthrDID object and, accordingly, one ETHR DID.

Web Service A

Web Service A relies on the DIDManager to create a single EthrDID object to facilitate the following functionalities;

1. the login with DID following the DID SIOP Profile,
2. the generation of VCs,
3. the addition of service endpoints to the DID Document,
4. the generation and publishing of new key pairs to sign VCs.

Web Service A's UI consists of three web pages:

1. The **Login Page** allows the user to log in with a DID.
2. The **Request VC Page**, allows the user to request a VC containing a claim that grants access to Web Service B's protected resource.
3. The **Admin Page** allows the institution to view its current DID, associated Ethereum address and balance, as well as published service endpoints and its DID Document. Furthermore, it allows publishing new service endpoints to its DID Document and the generation and publishing of new key pairs.

Web Service B

Web Service B relies on the DIDManager to create a single EthrDID object to facilitate the following functionalities;

1. the login with DID following the DID SIOP Profile,
2. the verification of VCs.

Web Service B's UI consists of two web pages:

1. the **Login Page** allows the user to log in with a DID.

2. the **Present VC Page**, allows the user to present a VC to access the protected resource.
3. the **Protected Resource Page** displays the protected resource.

5.3.3 Interactions

In this subsection, we present how the DID Wallet and the two web services communicate from a technical perspective.

Credentials Interaction

In this section, we describe how a VC is requested, stored, and presented in the prototypical mvSSIn implementation. In the mvSSIn prototype, users can request a single type of VCs from institution A's web service, attendance certifications. We implement three different VCs adhering to the attendance certification type. Each VC attests that the user attends a specific university course and allows for the access of a protected resource on institution B's web service.

To request the VC from web service A, the user first must log in with a DID, as described in section 5.3.3. After logging in, initially, a connection between the used DID and a real-life identity must be created. To create the association, we assume that the institution shared a code with authorized users. A logged-in user enters the code to web service A and request a VC. Relying on the entered code, a connection between the user's identity, authorization, and DID is created.

Based on the connection between the user and his DID, the web service creates a VC, containing the user's DID and proof over the user's attendance of a particular university course. Next, responding to the user's VC request, the web service redirects the browser to a Deep Link. The Deep Link contains the VC's JWT representation and points to the DID Wallet. Being redirected to the Deep Link triggers the browser to open the DID Wallet and forwarding the VC's JWT representation. The now opened DID Wallet persistently stores the VC and presents the user with an updated VC list.

As previously mentioned, web service B stores a protected resource. To access the resource, the user must first log in with a DID, as described in subsection Login with a DID. After logging in, a user can decide to prove his authorization to access the protected resource. To check the user's authorization, the web service challenges the user to provide a VP containing his access rights. Therefore, the web service redirects the browser to a Deep Link. The Deep Link contains a nonce, a return link, and points to the DID Wallet. Being redirected to the Deep Link triggers the browser to open the DID Wallet, and to forward the nonce and the return link.

After receiving the nonce and the return link, the DID Wallet presents the user with his list of VCs. The user selects the VC containing the proof over his access rights. The DID

Wallet then appends the nonce to the selected VC and signs it with the private key associated with the DID contained in the VC. By signing the nonce, the DID Wallet proves that the user controls the DID associated with the VC. The signature is needed since the DID used to request the VC might differ from the DID used to login to web service B. The DID Wallet then opens the browser and forwards the return link appended with the signed VC and nonce in a JWT representation. The browser opens the link and thereby transfers the selected VP to web service B. Finally, web service B verifies the validity of the transmitted VP. Upon success, the web service forwards the browser to the protected resource.

Login with a DID

The DID based login we implement follows the SIOP DID Profile and the pairwise DID paradigm.

The SIOP DID Profile as described in *Self-Issued OpenID Connect Provider DID Profile* [66] is a DID authentication flavor to use OpenID Connect (OIDC) [34] to integrate Identity Wallets in a generic way into web applications.

Our implementation of the SIOP flow is depicted in 5.5. The flow's interactions take place between four entities, the user, a web client being the user's smartphone browser of choice, the DID Wallet, and the institution's server.

In the first four steps, the user requests and is served the institution's web site via the browser. On clicking the "Sign in with SSI" button on the institution's web site, the browser forwards a sign-in request to the institution's server via HTTPS. The server generates and responds with a SIOP request. The SIOP request consists of metadata and a request object. The request object contains data that allows for the verification of the institution's DID binding. Furthermore, it contains a nonce to be signed by the user agent and a callback URL.

In response to the SIOP request, the browser forwards it to the DID Wallet. The DID Wallet opens and validates the request in regards to a correct DID authentication. Upon success, the user is prompted by the DID Wallet to enter the password protecting his private keys. The DID Wallet then either generates a new DID if the web service is unknown or lets the user select a preferred DID to authenticate with the institution.

Following, the DID Wallet generates a SIOP response consisting of metadata and at least of the user's DID and a signature proving the user's control over his DID. The DID Wallet transmits the response by opening the callback URL appended with the response object in the browser. The SIOP response transmitted via HTTPS to the institution's server is then validated in regards to a correct DID authentication. Upon success, the institution's server redirects the browser to the success web page.

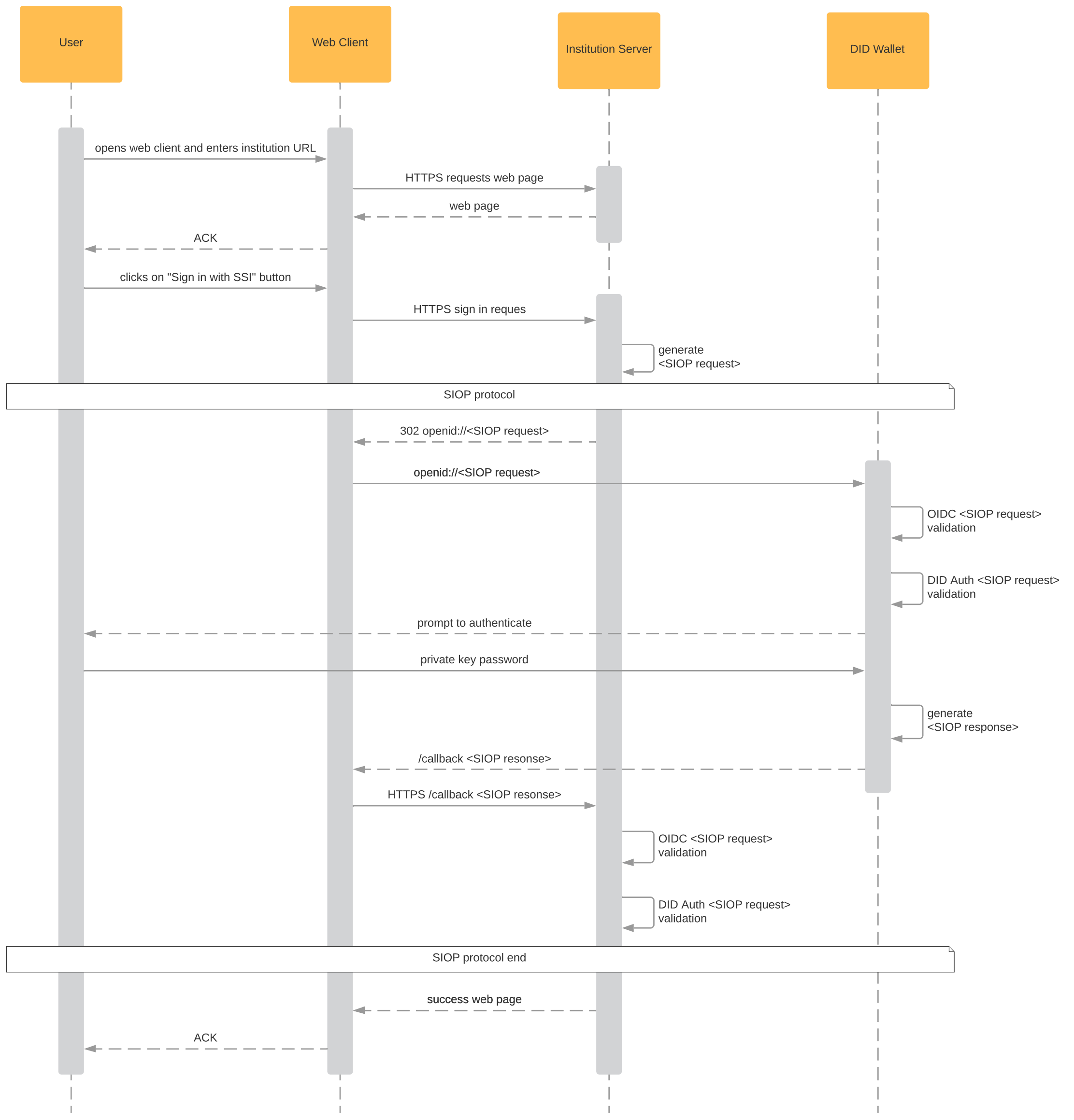


Figure 5.5: SIOP flow with a mobile browser as the user-agent and a DID Wallet app as the SIOP. Adapted from [66]

5.4 Limitations

In this subsection, we present a list of limitations we perceive are inherent to our implementation. These consist of feedback we received in the user survey and features the SSI concept allows for, but we did not implement.

The prototype does not include:

1. An identity hubs or similar out of wallet storage of VCs.
2. A key recovery mechanisms, like seed phrase or social key recovery.
3. Complex VCs attesting more than one claim.
4. A fully implemented SIOP flow. We do not fully implement all requirements of OIDC.
5. Full key rotation. We do not implement key rotation because uPort's ETHR DID library lacks a delete operation for DID Document properties. However, we provide a limited form of key rotation for institution A. Institution A can generate and publish new keys for the signing and verification of VCs.

6 User Journey

In this chapter, we present the user journey of the mvSSIn we implement as part of this thesis. We develop the user journey in conjunction with the mvSSIn implementation. We view the user journey from two different perspectives, the user and the institution perspective.

In the following, we first present the preliminary decisions we took. Secondly, we present user's and the intentional user journey coupled with screenshots of our implementation.

6.1 Environment

In this section, we present our assumptions regarding the environment in which the user journey takes place. We characterize the environment by the users' device choices and the interaction method between users and institutions.

We assume that the user participates in the mvSSIn using his smartphone. We base this assumption on the increasing percentage of smartphone internet market share, as shown in figure 6.1. This is especially relevant in regards to emerging markets, seeing a substantial increase in smartphone ownership [38] and a high share of mobile internet traffic. As an example, India has a mobile market share of over 70% since mid 2016 [9].

We further assume that the user interacts with the institutions by accessing their respective web site using a browser.

6.2 User Perspective

6.2.1 Initialization

In this section, we describe the user's initial steps to partake in our mvSSIn. The user first downloads the DID Wallet from an app store.

As seen in figure 6.2, when starting the DID Wallet for the first time, the user sets a password to protect his data. After setting the password, the DID Wallet shows the Home Screen. The Home Screen consists of two buttons, leading to the Identifier Screen and the Credentials Screen.

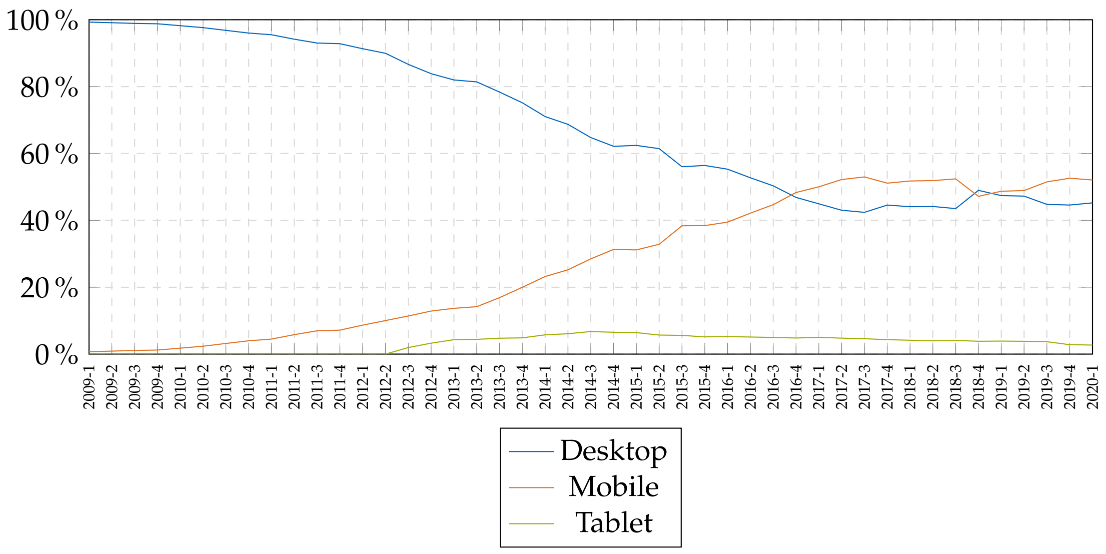


Figure 6.1: Global Desktop / Mobile / Tablet Internet Market Share [47]

6.2.2 Creation of a Decentralized Identifier

In this section, we describe how a user can manually create a DID, as seen in figure 6.2. The user navigates to the Identifier Screen, where with a click of the plus button, he can create a new DID. The Identifier Screen displays all paired DIDs and manually created DIDs, and allows to delete DIDs.

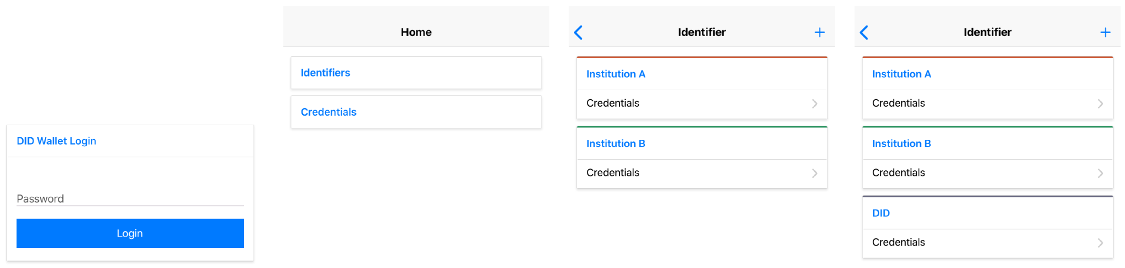


Figure 6.2: Screenshots of DID creation

6.2.3 Deletion of a Decentralized Identifier

In this section, we describe how a user can delete a DID, as seen in figure 6.3. The user navigates to the Identifier Screen, where by swiping a DID to the right, a delete button appears. He clicks the delete button and thereby removes the private key and all associated credentials from the app.

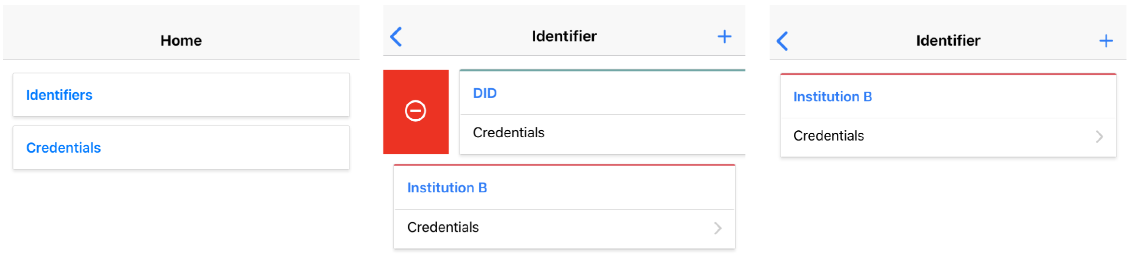
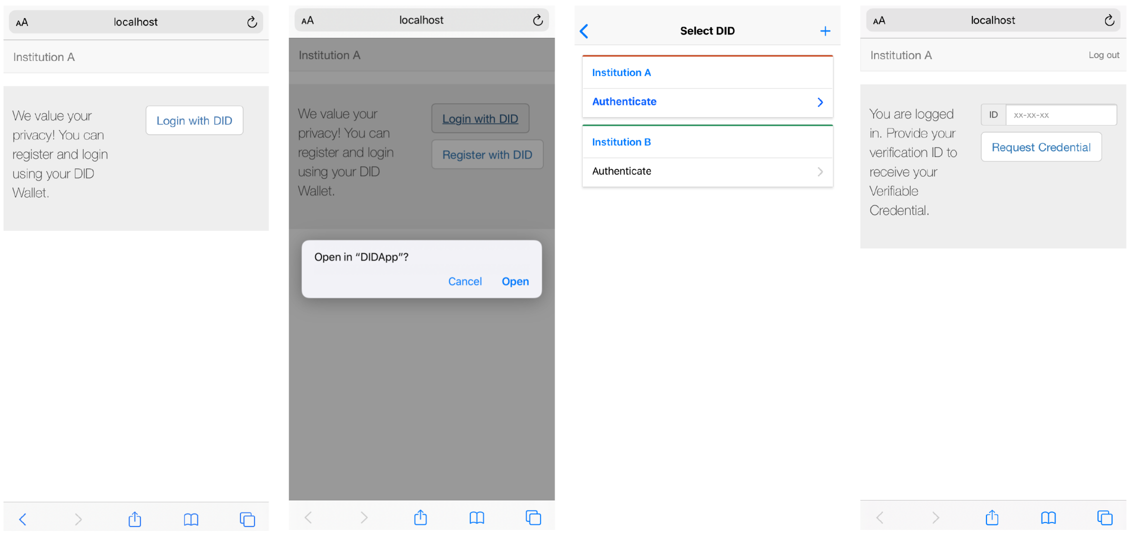


Figure 6.3: Screenshots of DID deletion

6.2.4 Login with a Decentralized Identifier

In this section, we describe how the user logs in to web services in our mvSSIn, as seen in figure 6.4. We intentionally develop a similar approach to a password manager to create a familiar experience for the user.

The user navigates to the login page of the web service using his browser. The login page provides a *Login with DID* button. The user selects the *Login with DID* button, following the user is prompted to open the DID Wallet. On the first time logging in, after selecting the prompt's open button, the DID Wallet automatically generates a paired DID. On all subsequent visits, the DID Wallet opens and shows the Identifier Screen. The Identifier Screen displays all stored DIDs, but highlights the paired DID. This allows the user to either select the previously generated paired DID, or to create a new DID if he wants to create a new account. After the DID Wallet generated the paired DID or the user selects a DID, the DID Wallet forwards the user to the browser where the web service's login area is displayed.



We value your privacy! You can register and login using your DID Wallet.

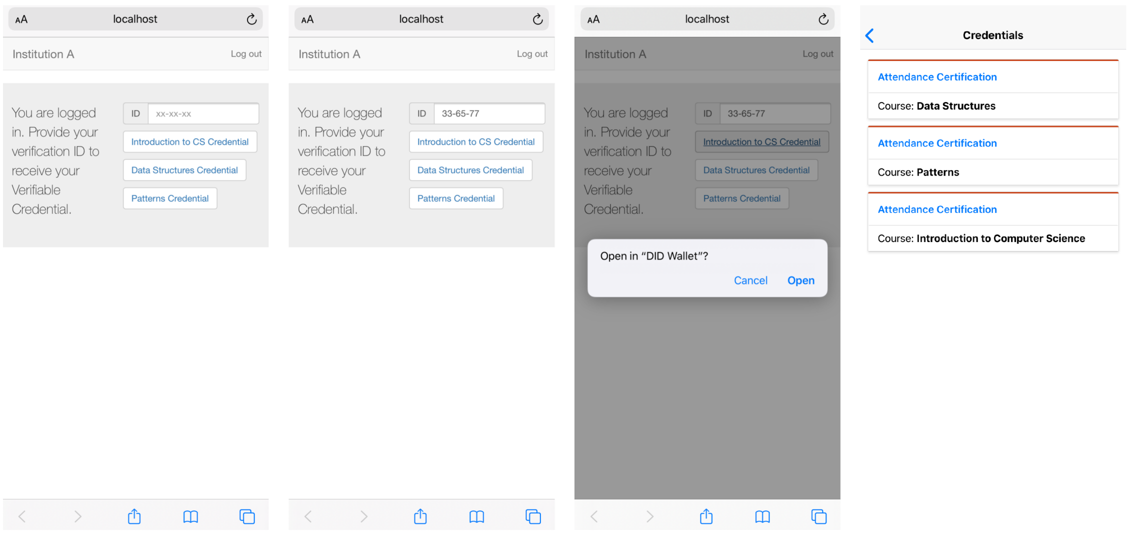
You are logged in. Provide your verification ID to receive your Verifiable Credential.

Figure 6.4: Screenshots of SSI login process

6.2.5 Request of a Verifiable Credential

In this section, we describe the process of requesting a VC, as seen in figure 6.5. In our mvSSIn institution, A's web service provides a page in its login area on which the user requests VCs. After navigating to the request page, the user fills out a field with an authentication code. We assume that the user received this code, either through email, the postal service, or in person. The code allows the institution to create a connection between an email or a real-life identity with the DID used to log in. After typing the code into the field, the user then selects one of the three request VC buttons. Following the user's selection, the browser prompts the user to open the DID Wallet. After the selection of the prompt's open button, the DID Wallet shows the Credential Screen, containing all previously received VCs as well as the new VC.

Multiple alternatives to the described authentication process are possible. In a low stake use-case, the institution can provide VCs without a code. Depending on the need for the proof of digital or real-life identity attributes, institutions can use alternative identification in high stake use-cases. Possible alternatives are, for example, video identification, identification using a third-party VC, or the transmission of digital or scanned documents. By using alternative authentication processes, a multitude of identity characteristics, including personal information like name, age, and address or membership of a group, can be verified.



You are logged in. Provide your verification ID to receive your Verifiable Credential.

ID xx-xx-xx

ID 33-65-77

Open in "DID Wallet"?

Attendance Certification

Course: **Data Structures**

Attendance Certification

Course: **Patterns**

Attendance Certification

Course: **Introduction to Computer Science**

Figure 6.5: Screenshots of VC request process

6.2.6 Sharing of a Verifiable Presentation

In this section, we present the process of sharing a VP, as seen in figure 6.6. In our mvSSIn, we model a simple use-case, the access of a protected resource. To access the protected resource, the user logs in to institution B's web service with his DID, as described in section 6.2.4. In the login area, the user navigates to the protected resource web page, where he requests access to the protected resource by clicking on the *Select Credential* button. Following the user's selection, the browser prompts the user to open the DID Wallet. After the selection of the prompt's open button, the DID Wallet shows the Credential Screen, containing all previously received VCs. The user selects one of the VCs. After the selection, the DID Wallet forwards the user to the browser where either a descriptive error message or the protected resource is displayed.

6.3 Institution Perspective

6.3.1 Enablement of a Web Service

In this section, we describe how an institution creates or extends a web service to provide SSI capabilities. Using our prototype, the integration of SSI capabilities requires technical knowledge of the underlying technologies since we do not provide a GUI for this step. With our prototype, institutions can integrate three types of SSI capabilities into their web service; login with SSI, issuing of VCs, and the verification of VPs.

First, the institution must extend their backend with software components that provide the functionally necessary for the SSI capability they want to add to their web service.

Furthermore, the institution must extend its fronted and decide how they want to

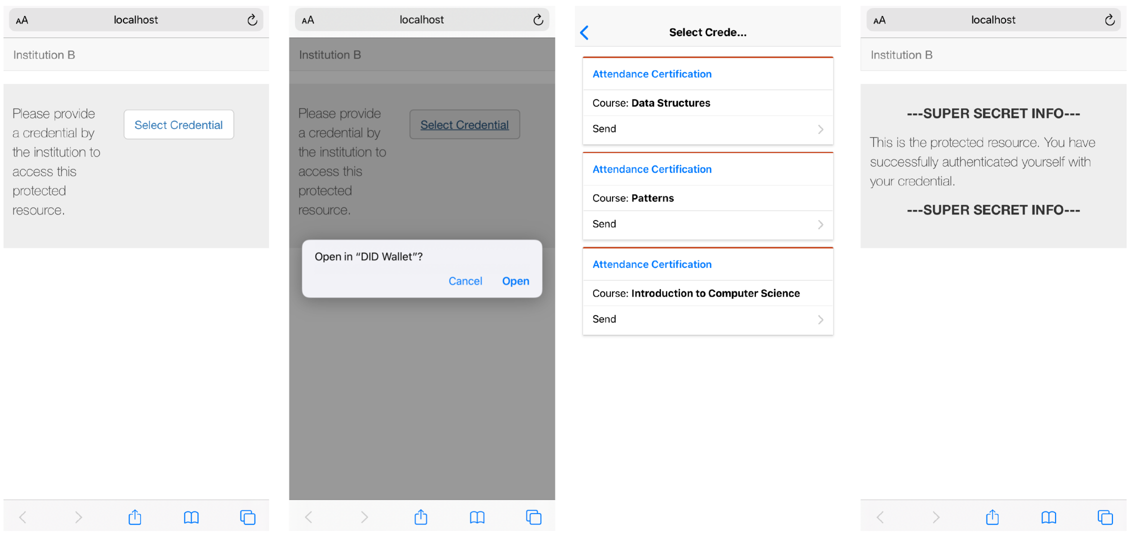


Figure 6.6: Screenshot of the VC presentation process

integrate the SSI capabilities.

The login with SSI capability requires the extension of a login page with a *Login with SSI* button.

The issuance of VCs requires the institution to add an according user interface to its web service. Furthermore, the institution must decide which VCs it wants to issue and which process it wants to follow to assure the correct issuance. In our implementation, the institution shares a one-time code with the user to connect the user's real-world identity with a DID.

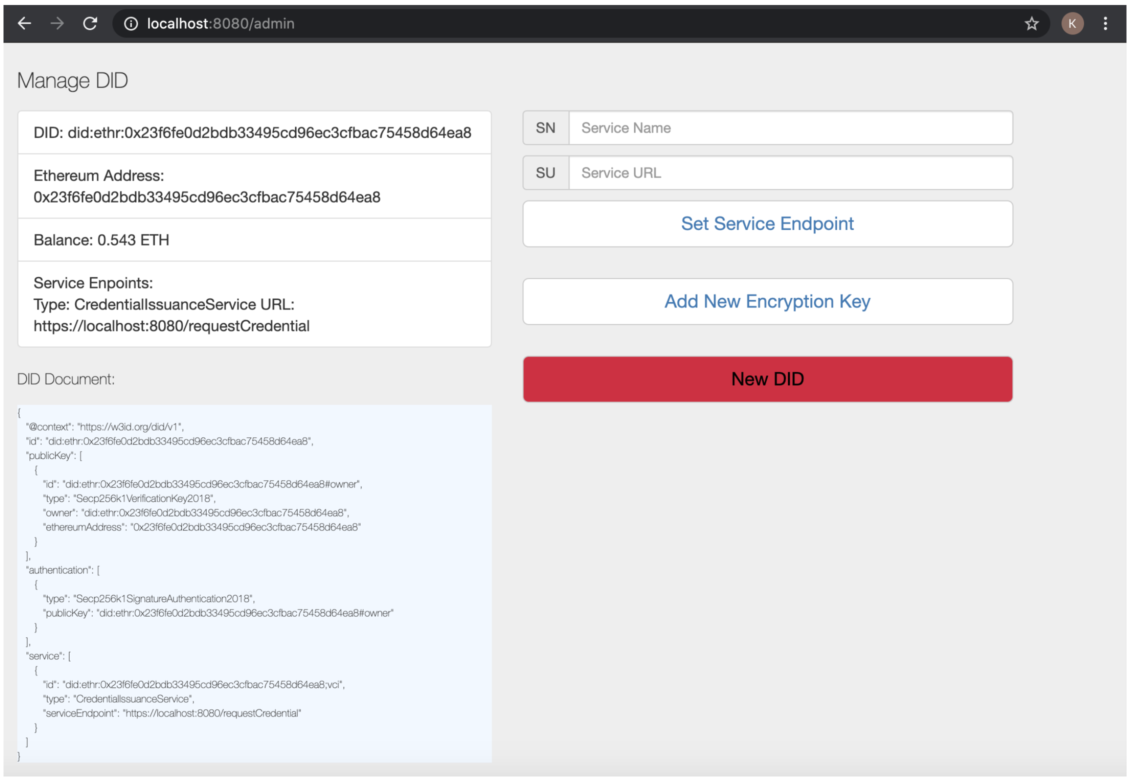
The verification of VPs requires an according user interface. Furthermore, the institution must decide on which VPs to accept. The acceptance is usually dependent on, which issuers are trusted, and which data a VP must attest.

6.3.2 Creation of a Decentralized Identifier

In this section, we describe how an institution can create a DID. As seen in figure 6.7, our prototype provides an admin page as part of the institution's web service. After implementing its SSI capabilities, an institution creates an initial DID, by selecting the *New DID* button in the admin page. Institutions usually create a single DID, since they want to be known by a single DID in the SSI ecosystem.

6.3.3 Publication of a Service Endpoint

In this section, we describe how an institution can add service endpoints to its DID Document. As seen in figure 6.7, our admin page provides an interface to set a service



Manage DID

DID: did:ethr:0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8

Ethereum Address:
0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8

Balance: 0.543 ETH

Service Enpoints:
Type: CredentialIssuanceService URL:
https://localhost:8080/requestCredential

DID Document:

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:ethr:0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8",
  "publicKey": [
    {
      "id": "did:ethr:0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8#owner",
      "type": "Secp256k1VerificationKey2018",
      "owner": "did:ethr:0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8",
      "ethereumAddress": "0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8"
    }
  ],
  "authentication": [
    {
      "type": "Secp256k1SignatureAuthentication2018",
      "publicKey": "did:ethr:0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8#owner"
    }
  ],
  "service": [
    {
      "id": "did:ethr:0x23f6fe0d2bdb33495cd96ec3cfbac75458d64ea8;vci",
      "type": "CredentialIssuanceService",
      "serviceEndpoint": "https://localhost:8080/requestCredential"
    }
  ]
}
```

Figure 6.7: Screenshots of the institution's admin interface

endpoint. Under the condition that the balance of the Ethereum address associated with the institution's DID covers the transaction cost, the institution can use the interface to publish its service endpoints to its DID Document. In figure 6.8 an estimate transaction cost can be seen. Therefore, the institution adds the service name to the *Service Name* field, the service URL to the *Service URL* field, and clicks the *Set Service Endpoint* button. Upon success, the admin page lists the service endpoint in under *Service Endpoints* on the left-hand side.

As an example, after setting up a credentials service, the institution adds it to its DID Document by setting the *Service Name* to *CredentialsIssuanceService* and the *Service URL* to the URL on witch credentials can be requested.

$$transactionCost = (gas = 34236) * (gasCost = 34GWEI) * (EUR/ETH = 185EUR) = 0.24EUR$$

Figure 6.8: Estimated transaction cost for publication of a service endpoint [13, 12]

6.3.4 Addition of a New Encryption Key

In this section, we describe how an institution can add new encryption keys. As seen in figure 6.7, our admin page provides a *Add New Encryption Key* button. Under the condition that the balance of the Ethereum address associated with the institution's DID covers the transaction cost, the institution adds new keys by selecting the *Rotate Keys* button. In figure 6.9 an estimate transaction cost can be seen. The web page's backend adds the public key to the DID Document and sets the VC signing key to the private key. The web services' backend then signs future VCs with the new private key.

As described in section 3.1.1 cryptographical best practice mandates, that the institution updates its keys in a regular interval.

$$transactionCost = (gas = 51805) * (gasCost = 34GWEI) * (EUR/ETH = 185EUR) = 0.36EUR$$

Figure 6.9: Estimated transaction cost for publication of a new encryption key [13, 12]

6.4 Limitations

Due to time constraints, we could not cover all scenarios relevant to a real-world application in our user journey.

Following scenarios are not part of our user journey:

1. A key recovery scenario. We do not provide for a key recovery mechanism that allows the DID Subject to restore private keys in case of loss.
2. An identity export and import scenario. We do not provide for a mechanism that allows users to export and import their identity, to avoid vendor lock-in. An

identity consists of all DIDs and associated VCs. No standardized mechanism has yet been developed to allow for cross-app import and export.

3. A password lost scenario. We do not provide for a mechanism that allows users to regain access to the DID Wallet if their password is lost.
4. An institution DID deletion scenario. Since our prototype allows institutions to store service endpoints and additional private keys on the blockchain, the *id* property of the DID Document stored on the blockchain has to be set to *0x0* to indicate the deletion of the DID, as described in 5.1.5. We do not provide a scenario that allows for this.
5. In-app explanations. We do not provide explanations to educate users and institutions about the SSI concept and how to use the prototype.

7 User Survey

In this chapter, we describe the user survey we conduct. First, we present the idea behind the survey. Secondly, we explain our selection of interview partners. Finally, we present the outcomes of the user survey and how we improve the prototypical mvSSIn implementation based on the interviewee's feedback.

7.1 Idea

For the qualitative assessment of the mvSSIn implementation and user journey, we conduct semi-structured interviews. Based on the feedback, we improve the mvSSIn implementation and the user journey in three iterations.

Moreover, the interviews serve to deepen our understanding of the SSI concept from a technical, user experience, and a business perspective. We investigate which challenges the SSI community is currently working on and which use-cases SSI field experts regard as promising.

Furthermore, we investigated how issuing entities can be characterized. Based on user feedback, we develop the issuer desirability function explained in chapter 8.

7.2 Interview Guideline

We split the interviews into four parts. We use the questions described in the following as conversation starters and delve deeper into the subject when interesting discussion threads arise. We ask not all interview partners the full spectrum of questions, due to time constraints and comprehension problems by non-expert interview partners in regards to complicated technical questions.

We first ask questions regarding the background of the interview partner, which profession does the partner pursue, which login services does the partner use, and which background knowledge does the partner have in regards to blockchains and SSI.

In the second part, we present the mvSSIn implementation and user journey to the interview partner. The presentation consisted of a presentation of the DID Wallet itself, the institution web pages, and the interactions between both. First, we show how a user logs in using the DID Wallet. Secondly, we demonstrate how to request and present a credential using the DID Wallet. After the presentation, we ask if the partner understood the approach and if he has any change requests. We also inquire about the partner's preference regarding alternative implementation approaches. We ask if the partner would prefer to use one or multiple DIDs and if the partner would rather log in to a web site with a DID or a VP.

In the third part of the interview, we ask questions regarding the SSI concept in general. What the partner thinks about the SSI concept in general, which use-cases the partner thinks are promising and which challenges must be solved from the partner's perspective to mature the SSI concept and its applications.

In the final part of the interview, we gather the information for the development of the issuer desirability function. For this purpose, we ask two questions: Which attributes characterize an issuing entity? How can the importance of those attributes be rated in comparison to each other? We describe the results in chapter 8

7.3 Selection of Interview Partners

We interview a diverse group of partners since we want insights from partners with a non-technical background, with blockchain knowledge and with SSI knowledge. We divide the partners into three groups, as seen in figure 7.1.

	SSI Knowledge	**No SSI Knowledge**
Blockchain Knowledge	7	1
No Blockchain Knowledge	0	6

Table 7.1: Selection of interview partners

7.4 Outcome

7.4.1 Login Services

The interview partners all currently use password, username, or email combinations to log in to web services. Most use password managers, either as dedicated apps or integrated in their browser of choice to store login combinations.

Additionally, most partners use third party login mechanisms like Facebook Login [17], Google Sign-In [23] and Sign-In with Apple [42]. However, all partners dislike third-party login mechanisms for their negative privacy implications. Some partners try to circumvent these issues by creating dedicated Facebook or Google accounts. The partners report that they use the third-party login mechanisms, despite their privacy implications, because of their ease of use.

Furthermore, we receive feedback that partners wish to use more secure types of login system, for example, FIDO2 [20] based systems.

In conclusion, we can deduce that our interview partners mostly rely on classical login services, but that interest exists for an easy-to-use and more secure approach.

7.4.2 Implementation Feedback

We improve the implementation of the mvSSIn and the user journey in three iterations. In the following, we present the feedback we receive in each iteration and how we improve upon it.

Iteration 1

After the presentation of the initial implementation of mvSSIn, some of the interview partners report that they are confused by the large number of DIDs they could choose from. We improve on this issue by color-coding the DIDs.

Furthermore, the partners wish to have information about which DID was used for which web service. We add this in a prototypical form; we extend the displayed DIDs by a used with field displaying the institution's name if the DID user logged in to the institution's web service with the DID.

Iteration 2

Kaliya Young criticizes that from a user-centric perspective, the second iteration of our prototype is to technically focused. We display DIDs and the VC claims in clear-text. To improve this, she suggests that we should try to hide the DIDs in sub-interfaces. Therefore, she introduces us to the concept of card-like icons to which the user can map to his mental identity model, hiding the underlying DID or VC. For example, a bank credential displayed as a bank card.

Kim Hamilton Duffy adds that a security issue in our system is that we do not bind the one time code to a specific DID. When requesting the one-time code from institution A, the requesting entity should have to share its DID. Sharing the DID allows the institution to link the code to the specific DID. The preemptive coupling avoids that the code is transmitted to and used by a third party.

Daniël Du Seuil points out that if we further develop our prototype, we should introduce Identity Hubs and key-recovery. Identity Hubs allow storing credentials outside of the DID Wallet to mitigate the risk of losing credentials. He mentions two approaches, using the users' existing cloud storage or relying on IPFS [29]. Key-recovery introduces functionality to create a backup of the users' private keys. Multiple approaches exist; he mentions Shared Key-Recovery, also known as Social Key-Recovery, where multiple confidants receive so-called shards, which can, when combined, recover the user's private key. Furthermore, he suggests we should improve how we present the information stored in VCs as VPs to the verifiers, for example, by allowing for selective disclosure of claims.

All three field experts point us to pairwise DIDs. This concept is further discussed in section 7.4.3 and explained in section 3.1.1.

We add the card like icons and the pairwise DID approach to our prototype. We do not add Identity Hubs, key-recovery, and improved VP presentation to our prototype due to time constraints.

Iteration 3

For the final state of our implementation and user journey, we receive positive feedback. However, we have to explain the user journey to field experts and common users alike. This leads us to the conclusion that we should extend the implementation by an in-app explainer. Furthermore, web-services offering SSI-login should explain the concept or refer to an explanatory web-page.

Furthermore, Heather Vescent suggests to investigate alternative naming. Does *DID Wallet* match what we are doing? Since we are using DIDs as logins, would *Identity Manager* maybe be more suitable?

Due to time constraints, we did not research explaining the user journey in the app, nor did we research what name transports our apps functionality best.

7.4.3 A/B-Testing

In the first A/B testing question, we ask if users prefer one or multiple DIDs. All field-experts prefer multiple DIDs, for the presented user-centric use-case, especially since Personally Identifiable Information (PII) is involved. As stated by the SSI experts, multiple DIDs reduce the ability of service providers to create user profiles by correlating user actions based on the DID used.

In iteration 1 and 2 of the prototype, the user has to manually create his DIDs and chose which DID to use for a service. In iteration 3, we add pairwise DIDs. To fully implement this concept, a new DID has to be created and used for each service the user interacts with. However, according to interviewed SSI experts, this approach has the following issues. If a user has two VCs requested with different DIDs and presents the VCs to a verifying party, two problems arise:

1. The verifying party would now know at least two DIDs the user owns.
2. It would be unclear to the verifying party if the two VCs are associated with the same real-life person.

We explain our prototypical solutions to the second issue in 5.3.3. However, according to Kim Hamilton Duffy, the protocols for solving the issues of multiple DIDs are not jet standardized. Therefore from her perspective, for first real-world applications, a single DID approach would be advisable to test the SSI concept.

According to Oliver Terbu and Kaliya Young, a solution to avoid the disclosure of the user's DID is the use of ZKPs (ZKPs) [22, 45]. However, according to Oliver Terbu, the current state of research has not yet arrived at a state where the use of ZKPs makes sense from a user experience perspective since the generation of ZKPs still takes several seconds. Nevertheless, he sees potential for this type of VC presentation in the future. This approach could also remedy the need for multiple DIDs since ZKP allow proving a statement without sharing anything else, including DIDs. However, Kaliya Young adds that it is unclear if verifying parties will accept and trust ZKPs.

Dr. Carsten Stoecker provides insight into use-cases where single DID approaches are applicable. He describes two enterprise use-cases where relying on a single DID is not an issue. Firstly, if an enterprise communicates with third parties, it wants to be known by a single DID. This approach allows entities to discover it with its single DID and to correlate that they are communicating with the same entity. Furthermore, he presents a second enterprise use-case, an enterprise SSI solution for employee monitoring. Since the inherent feature of this use-case is the monitoring of employees, the single DID approach is a valid choice.

Concerning the question, if the DID Wallet should use a DID or a credential to log in, all participants did not have a preference. Kaliya Young explains that from a user perspective, the critical aspect is not whether the DID Wallet uses a DID or a VC for logging in, but that the user interface makes it clear to the user that the DID Wallet stores the login to a web site.

7.4.4 Challenges

In this subsection, we present the challenges the SSI community has to address to further the SSI concept and its possible applications, according to our interview partners. First, we shortly introduce the challenges non-expert interviewees thought the SSI community has to address after the prototype presentation. Secondly, we present the challenges field experts report. We display an overview of all challenges in figure 7.2.

We order the challenges the field experts report into three categories. Firstly, SSI applications have to follow legal regulations, foremost the European General Data Protection Regulation (GDPR) [16]. Secondly, security and trust issues remain, threat models, have not yet been fully worked out. These missing threat models undermine the trust in the SSI concept. Thirdly, users have to be attracted to SSI services. Therefore, SSI implementations must provide an easily understandable user experience and provide users with useful credentials. Therefore, issuing parties have to be incentivized to partake in an SSI system.

We identify two groups in the non-expert interviewees. One group saw high potential if the concept is clearly explained to users, since the concept, as presented in our prototype, is easy to use. The other group doubts the potential for the SSI concept since users are reluctant to learn and integrate new tools into their behavior.

Regulations

From Oliver Terbu's perspective, a challenge are current privacy regulations, especially the GDPR. He states that together with regulators, the responsibilities and the legal framework for storing and managing personal data on the blockchain has to be clarified.

Flavia Maria Soare deepens our understanding of implications of the GDPR in regards to the SSI concept, from a law perspective. Depending on the amount of information stored on the blockchain, the issue of not being able to delete the data stored could be conflicting with the requirements of the GDPR. Furthermore, SSI implementations have to handle so-called availability breaches. An availability breach happens when the user loses access to his private keys and subsequently loses control of the DIDs and the linked VCs. Therefore, the user interface of SSI applications has to inform the user that these issues exist, as mandated by the GDPR.

Security and Trust

From Kim Hamilton Duffy's view, the threat models for SSI systems are not fully worked out. These threat models are especially complex for intricate implementations of the SSI concept since each additional software component adds a new threat model that has to be taken into account. Furthermore, she explains that from her perspective, though several approaches exist, key-recovery is not jet fully solved. Key-recovery approaches have not yet been extensively tested, especially in a use-case with non-tech-savvy users. Therefore, she recommended that the SSI community should study the fitness of SSI systems from the perspective of security and disaster recovery first in low stakes use-cases.

Furthermore, Kim Hamilton Duffy gave us insights into the development of the SSI community's approach towards VCs. She told us that the SSI community moved away from storing VC on the chain, since storing even synonymous data on-chain can be tracked. The newer approach is to create off-chain VCs accompanied by a blockchain registry storing a cryptographic accumulator. The cryptographic accumulator would allow for inclusion checks if the issuer revoked the VC.

Additionally, she mentions an ongoing discussion in the field, if DID Revolvers should only return the current DID Document. She strongly disagrees since this would render VCs created by an institution before rotation of its keys useless. The older VCs would be useless since their signatures would be incorrect if checked against the new public key. To correctly verify pre-rotation VCs the DID Document dating back to the creation of the VC is required. To avoid this kind of issue, she hopes that the community will agree upon some core use-cases like lifelong credentials so that such fundamental changes will not be discussion points in the future.

Daniël Du Seuil elaborates that from a security perspective, the biggest issue is to assure trust in information. To create trust, an application must first make sure that an interaction party is the person or company the user expects and that the party controls all interactions. According to him, the SSI community must do more research to identify and solve all security issues involved. Therefore, as Kim Hamilton Duffy, he advises, that first SSI applications should provide a low level of assurance over information. The applications should only be used for non-critical information, for example, student cards. As the complexity of assurance grows, he predicts that the market will adapt, grow, and provide more sophisticated services based on SSI systems. According to him, the regulator should support and accompany this process to ensure the system's security.

Dr. Carsten Stoecker adds that from his perspective, a standard application is needed so that communicating parties can trust that the other party stored their private keys correctly and thus trust that the communication is not compromised.

User Experience

From Kaliya Young's user-centric perspective, the current challenge is how to move from the earl early adopter state of the SSI system to an early majority. Therefore, she and Dr. Carsten Stoecker state, usability issues, and user experience issues have to be addressed. Firstly, SSI applications should provide a clear and easy to use UI that maps DIDs and VCs to the mental identity model of users. Furthermore, the concept has to be explained so that users will trust the system. Additionally, as previously mentioned, Kaliya Young and Kim Hamilton Duffy explain that a reliable key-recovery mechanism for the common user has to be developed.

Daniël Du Seuil elaborates on user acceptance from the perspective of the European SSI Framework (eSSIF) [11], the SSI initiative of the European Union. According to him, based on the trust created by the involvement of the regulator, namely the European Union, user acceptance of the eSSIF depends on the added value it can provide. From his perspective, only a small group of users see this added value in the security features. Hence, to reach the early majority, a balance between the usability and the security of the system has to be created. Additionally, appropriate use-cases have to be picked where the eSSIF can improve efficiency.

Market Acceptance

Kaliya Young and Kim Hamilton Duffy state that there are still many unknowns regarding market adoption. For example, it is not yet clear to which extent market participants will trust and start to accept VPs and cryptographical proofs like ZKPs since standards are very new or still under development.

A further challenge, according to Oliver Terbu, is to motivate institutions to act as issuers in the SSI system. The Sovrin Foundation proposed an approach in which monetary transaction incentives the use of VCs [43]. However, according to Oliver Terbu, digital identity should not be a commodity. Therefore, he advocates to follow alternative approaches. One of the approaches aims to indirectly produce value by allowing for cost reduction through increased efficiency and by increasing customer numbers through harmonization and merging of systems. It remains to be seen if the approaches will yield the intended results.

According to Daniël Du Seuil, the added value of eSSIF is dependent on the number of parties that support the system. From Daniël Du Seuil's eSSIF perspective, this will start with state and federal governments and extend to institutions and companies, like banks and retailers.

Type	Short	Explanation
Regulations	Storing on Blockchain	What data can be legally stored on the blockchain?
	Availability Breaches	How can users be educated about availability breaches?
Security and Trust	Threat Models	Which threat models apply to SSI and its applications?
	Communication Safeguarding	How can trust in communication partners be created?
	Key Recovery	How can users recover their keys in case of loss?
User Experience	UI	What does a UI look like that maps to the users identity model?
	Usability	What does a SSI system look like that is easy-to-use?
Market Acceptance	Trust in VPs and ZKPs	Will entities trust VPs and ZKPs?
	Motivate Issuers	How can institutions be motivated to act as issuers?

Table 7.2: Overview of SSI challenges

7.4.5 Use-Cases

Regarding the possible use-cases, we receive a broad spectrum of answers. We subdivide the use-cases can into two groups: enterprise and user-centric use-case. Furthermore, some interview partners elaborate on which types of use-cases should be approached first.

User-Centric Use-Cases

The non-technical interview partners come up with ideas for user-centric use-cases in their fields of work. A student and art historian develop the idea of a digital student ID allowing for easy access to scientific documents. A sound technician working in the movie industry states that an SSI enabled service could be beneficial for the secure transmission and retrieval of unreleased media artifacts like television programs or movies. Flavia Maria Soare, a lawyer, develop the idea of credential-based access management to case files. Based on a credential stating that a person is a lawyer and a credential by the lawyer's client, state authorities grant access to case files.

Oliver Terbu reports that he sees many inquiries from business partner in the area of Know Your Customer (KYC) processes [53]. He and Kaliya Young elaborates on KYC mostly in regards to banking. Most governments have strict regulations for banking providers regarding the authentication of customer data. The authentication is currently often done using video identification services. By providing a possibility to store and reuse a successful video identification in a VC, banks hope to improve the user experience when registering for a banking service. Furthermore, using login services and authenticating user's online banking interaction based on a user's DID and a VC could improve the security of online banking services.

From a personal perspective, Oliver Terbu saw the potential for integration of the SSI concept into a smart city system, where citizens can interact with service providers in a secure and privacy-preserving manner.

Kaliya Young, Oliver Terbu, and Daniël Du Seuil state that governments are interested in the technology to attest their citizens digital credentials. The aim is to give the citizens more control over their digital identity and to improve the government's bureaucracy's efficiency. However, according to Oliver Terbu, all government-level projects are still in the early stages.

With Daniël Du Seuil, we discuss the SSI initiative of the European Union, the eSSIF. He first explains that the eSSIF envisions verifiable identities issued by governments. Furthermore, the eSSIF follows the eIDAS [7, 15] regulation, to inject the level of trust eIDAS provides into the decentralized approach. The main aim of the eSSIF is to put the citizen in control of their information and allow the citizens to share their information cross-sector and cross-border easily. Thereby increasing efficiency and reducing paperwork. We investigated which nations are currently interested in this approach. He named foremost Spain and the Netherlands, but also Germany, Italy, and smaller nations like Luxembourg and Malta. Hindering factors for nations are the lack of a sufficient digital infrastructure to integrate this approach, but also the existence of state-level solutions already providing similar services. In the current state of the project, the eSSIF team created a pilot and is investigating in stakeholder sessions, how governments and corporations could integrate and interact with the system. Furthermore, the meeting serves to align the system to governments' and companies' needs and to define how companies and governments can align their software solutions to the new system.

Kim Hamilton Duffy and Kaliya Young introduce us to educational credentials. Educational institutions attest to their students their graduation in the form of a VC. This would increase efficiency for the institution by avoiding paperwork, as well as for the students who could request and share the proof of his graduation in digital form. Furthermore, relying on the SSI concept would permit the lifelong storage of digital education documents.

Enterprise Use-Cases

Dr. Carsten Stoecker mentions an extensive list of use-cases in the enterprise sector, especially in regards to enterprise identity and product identity. According to him, the SSI system allows for fraud prevention, improving efficiency in regards to public data like addresses and payment processing. The credentialing component of SSI allows for environment, safety, health, anti-bribery, child labor, and labor rights certificates. Furthermore, attribute-based access management could be implemented based on a SSI system.

Additionally, Dr. Carsten Stoecker explains that pharmaceutical companies are interested in improving third party risk management using the SSI concept. The envisioned use-case solves supplier on-boarding is similarly to KYC in banking. The suppliers get credentials for their product and production quality, by certification bodies. Based on the trust in the certification bodies and the credentials it issued to the suppliers, pharmaceutical companies can dynamically rearrange their supply chains. Oliver Terbu adds that relying on product and production quality credentials can be useful in real-time enterprise verification in supply-chains, not only in the pharmaceutical industry. Other sectors are interested in SSI supply chain use-cases. The US Department of Homeland Security, for example, is currently researching SSI supply-chain approaches using credentials [52]. Supply chain credentials could be of value not only for companies but also for the end consumer. As Heather Vescent explains, credentials along the supply-chain could be used to track the production conditions. The end consumer could then have a more educated decision if a product adheres to the standards he expects, like fair trade and sustainable production.

Kim Hamilton Duffy introduces us to two further corporate use-case. In the first use-case, an organization uses a credential anchored its legal entity to create credentials identifying its officers and allowing them to perform specific actions, for example, filing certain paperwork. The officer uses his credential to sign his actions, creating a chain of credentials allowing the backtracking of actions. The second use-case was implemented as part of a pilot for the government of British Colombia [36]. Using this pilot, the government can attest digital safety credentials to enterprises. The credentials can be further used, for example, for the negotiation of insurance contracts.

Approach to Use-Cases

Kim Hamilton Duffy explains that from her security perspective, it is advisable to start with simple use-cases with low PII included, to test the SSI concept in first real-world applications. As an example, she mentions the enterprise use-case in which an organization creates credentials for its officers explained in Enterprise Use-Cases.

Furthermore, Kim Hamilton Duffy explains that use-cases anchoring into existing trust networks, to make existing processes more efficient, are a good starting point. As examples, she mentions the educational use-case explained in User-Centric Use-Cases and the British Columbia pilot explained in Enterprise Use-Cases.

8 Issuer Desirability Function

In this section, we develop an issuer desirability function. The function we develop assigns issuer the desirability to request a VC from them. We intend to model the decision making of future SSI users, deciding from which issuer to request a VC to investigate the potential for centralization in regards to issuers.

The issuer desirability is dependent on multiple attributes. Thus it can be classified as a Multi Attribute Decision Making (MADM) problem [48]. Multiple approaches exist to solve MADM problems [48]. The function we develop in this master's thesis follows the Analytic Hierarchy Process (AHP), as defined in *Decision making with the analytic hierarchy process* [40].

The AHP relies on expert knowledge. We gather the expert knowledge in the research conducted in this thesis of the SSI concept and ecosystem, the implementation, and especially in the user study. In the user study, we ask our interview partners to name the attributes they deemed important to the issuer desirability and weight them pairwise, as described in steps 1 and 2. In the following, we describe the development of the issuer desirability function in four steps.

Step 1: In AHP, problems are first decomposed into a decision hierarchy of interrelated elements. The top element is the goal of the decision. Child elements with further sub-elements are the decisions goals from a broader perspective. Leaf elements are the attributes of the decision alternatives to chose from. In other words, each element is a criterion of evaluation of each alternative. In this consideration, the goal is to select an issuer. As seen in figure 8.1, the **Issuer Desirability** is split into two leaf elements and one intermediary element:

1. The **Monetary Cost** captures the fees an issuer demands for the issuing of a VC.
2. The **Time Cost** captures the amount of time it takes to request and receive a VC from an issuer.
3. The intermediary element, **VC Usefulness**, captures the VC's usefulness to the user for further use.

The intermediary element, **VC Usefulness**, is split into three leaf elements:

1. The **Trust in Issuer** captures how much the participants of the SSI ecosystem trust the issuer to make valid claims, and thus the acceptance rate of VCs issued by former.

2. The **VC Quality** captures how far the issuer correctly follows all agreed-upon technical standards.

3. The **VC Validly** captures how long the VC is valid.

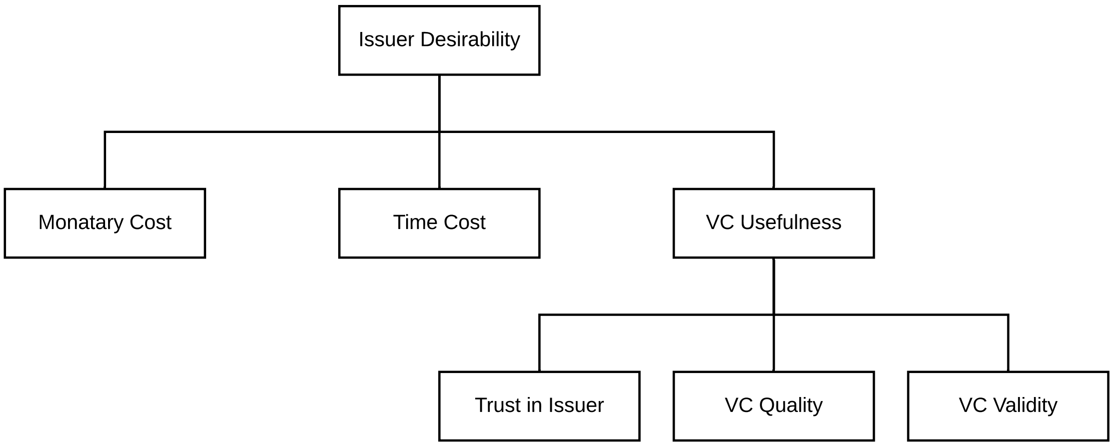


Figure 8.1: The hierarchy structure of the issuer desirability

Step 2: In the second step of AHP, a set of matrices of pairwise comparative weights is constructed. Each matrix captures the importance of each criterion pairwise compared to the other criteria on the same level of the decision hierarchy. We set the compared importance by selecting an intensity of importance of one criterion compared to another. The importance intensity ranges from 1 to 9. 1 indicating equal importance and 9 extreme importance of one criterion compared to another. To indicate a criterion being less important than another, one is divided by the value of the compared importance in the opposite direction, so 1 divided by a number between 1 and 9. In table 8.1 and 8.2 the sets construed for each level of the decision hierarchy can be seen. In the tables, we compare the criteria from row to column. As an example, table 8.1 shows that the importance of the VC usefulness compared to the monetary cost has a value of six.

Step 3: In the third step of AHP, the pairwise comparative weights are translated into criterion weights. Multiple approaches exist [48]. This thesis follows the method described by Saaty in *Decision making with the analytic hierarchy process* [40]. Each row is summed and divided by the sum of all pairwise comparative weights of the matrix, as seen in table 8.1 and 8.2. The weights of subcriteria must be multiplied by their parent's weight to arrive at global weights, as seen in table 8.2.

Step 4: In the last step, we combine the previously obtained weights vector to a function, displayed in figure 8.2. To arrive at an evaluation of different issuers, the alternative's criteria expressions must be multiplied with the corresponding weight and summed. A high sum indicates a high issuer desirability. The expression must be translated into comparable numeric values to be able to multiply the weights with the expressions.

Criteria	**Monetary Cost**	**Time Cost**	**VC Usefulness**	**Weights (W)**
Monetary Cost	1	1	1/6	0.125
Time Cost	1	1	1/6	0.125
VC Usefulness	6	6	1	0.75

Table 8.1: Pairwise comparison matrix of the main criteria in respect to issuer desirability

Criteria	**Trust in Issuer**	**VC Quality**	**VC Validity**	**Ws**	**Global Ws**
Trust in Issuer	1	5	8	0.650	0.487
VC Quality	1/5	1	5	0.288	0.216
VC Validity	1/8	1/5	1	0.062	0.046

Table 8.2: Pairwise comparison matrix of the main subcriteria in respect to VC usefulness

8.1 Limitations

In this section, we present the limitations of the function we develop. Firstly, the function assumes a decision-making situation where the user can choose from a multitude of issuers. In the use-cases we gathered in the user survey, in 7.4.5, none of the use-cases had this characteristic. All use-cases assumed centralized trust anchors that provide the credentials. A use-cases with centralized trust anchors results in a situation in which the user has no choice from whom he requests a credential.

Secondly, we do not account for changing user goals. The function assigns issuer desirability to issuers based on a static assumption of user preferences. However, user preferences shift depending on the use-case and the user's situation. In some use-cases or situations, for example, users might require fast certification and might be willing to pay a fee to receive a credential. In other use-cases or situations, the contrary might apply.

Additionally, to assign an issuer a desirability, his characteristics must be translated into numeric values. We do not provide a framework to do so.

$$IssuerDesirability = MonetaryCost * 0.125 + TimeCost * 0.125 + TrustInIssuer * 0.487 + \\ VCQuality * 0.216 + \ VCValidity * 0.0616$$

Figure 8.2: The issuer desirability function

9 Discussion and Future Work

This chapter summarizes the results of this master's thesis and gives an outlook on possible future work.

9.1 Reflection on the Research Questions

In this section, we present a summary of the results of our work. We structured the results by the research questions defined in section 1.2.

RQ1 What are the essential requirements for the implementation of a minimum viable SSI network?

The functional requirements we developed in chapter 4 are based on the fundamentals of the SSI concept. They aim to cover all key features associated with user-centric use-cases. We derived the non-functional requirements from considering technical aspects and the user-experience. We improved and extended both types of requirements based on the feedback we received in the user survey in chapter 7.

The requirements served as a foundation for the development of the mvSSIn implementation and user journey. Table 9.1 provides an overview of the defined requirements and their degree of fulfillment. The requirements can be completely fulfilled (✔), partially fulfilled (~), or unfulfilled(✘).

As indicated in the table, we were able to fulfill most requirements. We did not fulfill FR08, NFR02, and NFR06 due to time constraints. However, we developed our implementation and user journey in such a way that they can be extended by the missing features.

We fulfilled FR09 only partially. We implemented no key recovery for the user due to time constraints. The institution's web service stores the institution's private key in a JSON file in its root directory. Although this is not advisable from a security perspective, it allows the institution to export the key and store it for safekeeping. The private key can be re-imported by pasting it back in the root directory.

Additionally, we fulfilled FR10 only partially. We implemented a simple form of key rotation for the institution. The implemented mechanism allows the institution to update the DID Document with a new public key for VP validation and to set a corresponding private key for VC signing internally. However, uPort's Ethr-DID library used in our implementation does not provide the functionality to rotate the keys used to prove control over the DID. Subsequently, we did not implement a key rotation for the key directly associated with the DID.

ID	**Requirement**	**Fulfillment**
FR01	Create DIDs	✔
FR02	Delete DIDs	✔
FR03	DID Login	✔
FR04	Request VCs	✔
FR05	Issue VCs	✔
FR06	Present VPs	✔
FR08	Out of wallet storage of VC	✘
FR09	Key Recovery	~
FR10	Key Rotation	~
NFR01	Simple UI	✔
NFR02	Explanatory UI	✘
NFR03	Use the ETHR DID Method	✔
NFR04	Extend web services	✔
NFR05	Paired DIDs	✔
NFR06	DID communication	✘

Table 9.1: Requirement Reflection

RQ2 How can a prototypical minimum viable SSI network be implemented?

Based on the requirements developed in RQ1, we implemented a mvSSIn in a prototypical form described in chapter 5. For the implementation, we relied on the ETHR-DID Method and uPort's Ethr-DID library. We developed a DID Wallet and two institutions' web services. The DID Wallet allows users to create, store, and use DIDs. The two web services allow logging in with a DID. The DID Wallet automatically generates the DIDs used for the login following the pairwise DID paradigm.

Furthermore, the DID Wallet allows to store VCs and present VPs. VCs can be requested from the first institution's web service and presented as VPs to the second institution's web service to access a protected resource.

Based on the feedback we received in the user survey, the DID Wallet's UI hides the underlying technical concepts from the user. The DID Wallet displays DIDs as login items and the VCs as certifications. We aimed to thereby allow the user to map his mental identity model to his digital identity.

RQ3 What does a user journey taking place in the prototypical implementation of a minimum viable SSI network look like?

In conjunction with the implementation of a prototypical mvSSIn we developed a mvSSIn user journey, as described in chapter 6. The user journey covers all features we implemented. We developed the user journey from two perspectives, the user perspective, and the institution perspective. The user perspective encapsulates how a user can create and use DIDs and how a user can request, receive, and present credentials.

The institution perspective encapsulates how an institution can extend or create a web service with SSI features. The features include logging in with SSI, issuance, and verification of VCs. Furthermore, we present supporting activities. These activities consist of how an institution can publish its service endpoints in its DID Document and how to rotate VC verification keys.

RQ4 How can the issuer desirability be modeled?

A central role in the SSI concept can be attributed to issuers. To analyze the potential centralization of issuers, we developed a function that assigns issuers a desirability. The function assigns the desirability based on the expression of a set of characteristics. The issuer's characteristics relevant to the desirability of requesting a VC are: the monetary cost, the time cost, the trust in the issuer, and the expected quality and validity of the issued VC.

However, the value of the function is limited. We do not take into account that requesting entities have different requirements depending on the use-case. Furthermore, in currently envisioned use-cases, only one issuer attests specific VC's, like a government institution or university. Therefore, we can attest centralization in regards to issuers to the SSI trust network without having to rely on our function.

RQ5 Which SSI use-cases do field experts regard as candidates for first real-world SSI applications?

In table 9.2 we present the use-cases field experts regard as candidates for first real-world SSI applications. We differentiate between user-centric and enterprise use-cases.

RQ6 Which challenges must be solved by the SSI community from the perspective of field experts to mature the SSI concept and its applications?

The field experts we interviewed stated that the following challenges must be solved by the SSI community to mature the SSI concept and its applications:

Type	Short	Explanation
User-Centric	Government Credentials	Governments issue VCs that allow their citizens to request, store and present official government documents in digital form.
	Education Credentials	Educational Institutions issue VCs that allow their graduates to store and prove their graduation.
	KYC in Banking	To reduce the need for multiple video identifications, identifiers issue VCs to customers. The VCs can be reused for subsequent creation of bank accounts and to increase security during online banking interactions.
Enterprise	Government Credentials	Governments issue VCs attesting companies fulfillment of standards. Using the VCs companies can prove their compliance with standards to customers and potential contract partners.
	Supply Chain Credentials	Based on the government VCs and VCs from certification bodies, companies can dynamically manage their supply chains. Furthermore, products are certified along the supply chain allowing customers to better evaluate consumption decisions.

Table 9.2: Promising use-cases as stated by field experts

From a law perspective, SSI systems must comply with existing regulations, especially with the rules mandated by the GDPR. Therefore, together with regulators, the responsibilities and the legal framework for SSI systems must be clarified.

Furthermore, the SSI system and its applications must create trust in the information provided in VCs. Therefore the threat models of the SSI system and its applications must be elaborated. Consequently, the SSI community has to fix potential security issues. Additionally, applications must be designed so that the common user can understand and use the system securely. Additionally, a secure key recovery system for the common user must be developed and tested.

To achieve market acceptance, not only the applications must be understood by the users so that the user can map his identity model to the application, but the applications must provide added value. The added value is mostly dependent on institutions' motivation to issue VCs. Therefore, strategies to involve institutions into the development of SSI solutions must be created and applied. Furthermore, it is currently unclear if institutions and market participants will accept VPs and ZKPs. To mitigate this issue, SSI systems should be developed in close collaboration with possible stakeholders.

9.2 Conclusion

In this master's thesis, we developed three artifacts, the prototypical mvSSIn implementation, a corresponding user journey, and an issuer desirability function. Additionally, we discussed which SSI use-cases and challenges the SSI community is currently working on.

The successful prototypical mvSSIn implementation shows that SSI applications can be build based on the available documentation, specification, and free-to-use open-source software.

The developed user journey depicts how users and institutions can interact with our mvSSIn implementation. We aimed to develop a user journey that is easy to understand based on the feedback we received from users and field experts. Therefore, it resembles software the user is already familiar with and follows a paradigm in which the user can map his mental identity model to his digital identity.

The developed issuer desirability function assigns a desirability to an issuer based on the expression of certain characteristics. However, the informative value of the function is limited to specific scenarios that are improbable in the real world. Nonetheless, by investigating the limitations of the function, we can conclude that centralization in issuers is inherent to most currently envisioned SSI use-cases. The reason for this is that most currently envisioned SSI use-cases mimic real-world trust networks that rely on centralized institutions to attest identity characteristics to individuals.

By questioning field experts on which SSI use-cases they regard as candidates for first real-world SSI applications, we were able to compile a list of use-cases. We subdivided the use-cases into two groups: enterprise and user-centric use-cases. However, a use-case can overlap into both groups.

The enterprise use-case the field experts mentioned most often is credentialised supply change management. Based on VCs attested to suppliers, product and production quality can be verified. Furthermore, suppliers can then attest VCs to the delivered products. The VCs allow backtracking of the product along the supply chain.

The most notable user-centric use-case group are use-cases utilizing government credentials. The concept most prominently advanced by the eSSIF project proposes that governments can attest their constituents digital documents based on an SSI system. The idea is that these digital documents will replace or offer an alternative to their paper equivalents. The proponents of government-issued VCs argue that such a system will give the citizen more control over their identity and increase efficiency. These results show the broad range of applications envisioned for the SSI concept.

By questioning field experts on which SSI challenges must be solved by the SSI community to mature the SSI concept and its applications, we could show that many unknown factors and issues remain. The threat models of SSI applications are not yet fully worked out. Additionally, it remains to be seen how SSI applications can comply with legal regulations like the GDPR. Furthermore, it is unclear if the market will accept SSI solutions.

In conclusion, we can say that it is possible to create an SSI solution that the common user can understand. Furthermore, we can attest that the active SSI community envisions a broad array of use-cases leveraging the SSI concept, attracting attention from governments, institutions, and companies. However, many unknowns and issues remain that require attention to forward the SSI concept.

9.3 Future Work

In this master's thesis, we presented a detailed look into a mvSSIn and shortly investigated the SSI trust network as well as use-cases and challenges. During our research, a broad set of further research questions arose, which we could not answer in the scope of this master's thesis. This section presents starting points for further research.

Our mvSSIn implementation has some limitations. Future work could investigate and extend our implementation to integrate identity hubs, a key recovery mechanism, and complex credentials. Furthermore, we implemented a simple use-case. Future work could implement and investigate one of the use-cases from RQ5. For example, credentialised supply change management.

We relied on uPort's Ethr-DID library to implement our mvSSIn. Future work could investigate alternative libraries, including their advantages and limitations.

In RQ5, we introduced the European SSI Framework project. The eSSIF project proposes a framework in which governments can attest their constituents digital documents. Future work could investigate the eSSIF and similar projects to create insights into government credentials and user-centric SSI applications.

In RQ6, we investigated the challenges that remain in the SSI concept. Future work could develop a threat model for the SSI concept or one of its applications. Furthermore, future work could address how SSI applications, especially user-centric approaches, can comply with the GDPR and how the market can be educated about the SSI concept. Furthermore, a current discussion topic in the SSI community is how to incentivize entities to issue VCs. Future work could investigate different approaches, including their advantages and limitations.

Additionally, as mentioned in this master's thesis, field experts assume that future SSI application might use ZKPs to reduce the amount of data shared when presenting information to a verifier. Future work could investigate how ZKPs protect the user's privacy and if potential verifiers will accept attestations based on ZKPs.

List of Figures

List of Tables

Bibliography

[1] *0x Documentation: Web3 Providers Explained*. URL: https://d2uvd02r4antif.cloudfront.net/docs/guides/web3-provider-explained.

[2] C. Allen. *Exploring the Lifecycle of a Cryptographic Key*. URL: https://www.cryptomathic.com/news-events/blog/exploring-the-lifecycle-of-a-cryptographic-key-.

[3] C. Allen. *The Path to Self-Sovereign Identity*. URL: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[4] *Allowing Apps and Websites to Link to Your Content | Apple Developer Documentation*. URL: https://developer.apple.com/documentation/uikit/inter-process_communication/allowing_apps_and_websites_to_link_to_your_content.

[5] Apograf Team. *Symmetric Cryptography and Key Management: Considerations on Key Exhaustion, Rotation and Security Models*. URL: https://www.cryptomathic.com/news-events/blog/symmetric-cryptography-and-key-management-considerations-on-key-exhaustion-rotation-and-security-models.

[6] P. Brændgaard and J. Torstensson. *ERC1056: Lightweight Identity*. URL: https://github.com/ethereum/EIPs/issues/1056.

[7] *BSI - eIDAS-Verordnung*. URL: https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/eIDAS/eIDAS_node.html.

[8] C. Cadwalladr and E. Graham-Harrison. *Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach*. Tech. rep. URL: https://www.theguardian.com/news/2018/mar/17/....

[9] *Desktop vs Mobile vs Tablet Market Share India | StatCounter Global Stats*. URL: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/india#quarterly-200901-202001.

[10] P. Dunphy and F. A. P. Petitcolas. "A first look at identity management schemes on the blockchain." In: *IEEE Security & Privacy* 16.4 (2018), pp. 20–29.

[11] *eSSIF: The European self-sovereign identity framework*. URL: https://medium.com/@SSI_Ambassador/essif-the-european-self-sovereign-identity-framework-4572f6875e12.

[12] *ETH Gas Station | Consumer oriented metrics for the Ethereum gas market*. URL: https://ethgasstation.info/.

[13] *Ethereum (ETH) Preis, Charts, Marktkapitalisierung und andere Messgrößen | CoinMarketCap*. URL: https://coinmarketcap.com/de/currencies/ethereum/.

[14] *ethjs/ethjs-provider-signer: A simple web3 standard provider that signs eth_sendTransaction payloads.* URL: `https://github.com/ethjs/ethjs-provider-signer`.

[15] *EUR-Lex - 32014R0910 - EN - EUR-Lex.* URL: `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014R0910`.

[16] European Parliament and Council of the European Union. *EUR-Lex - 32016L0680 - EN - EUR-Lex.* URL: `https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:32016L0680`.

[17] Facebook Inc. *Facebook Login.* URL: `https://developers.facebook.com/docs/facebook-login/`.

[18] Facebook Inc. *AsyncStorage · React Native.* URL: `https://github.com/react-native-community/async-storage`.

[19] Facebook Inc. *React Native · A framework for building native apps using React.* URL: `https://reactnative.dev/`.

[20] *FIDO2: Moving the World Beyond Passwords using WebAuthn & CTAP.* URL: `https://fidoalliance.org/fido2/`.

[21] A. Galletta and W. E. CROSS. "The Semi-Structured Interview as a Repertoire of Possibilities." In: *Mastering the Semi-Structured Interview and Beyond.* From Research Design to Analysis and Publication. NYU Press, 2013, pp. 45–72. ISBN: 9780814732939. URL: `www.jstor.org/stable/j.ctt9qgh5x.7`.

[22] C. Garman, M. Green, and I. Miers. *Decentralized Anonymous Credentials.* Tech. rep. 2013.

[23] Google LLC. *Google Sign-In for Websites | Google Developers.* URL: `https://developers.google.com/identity/sign-in/web`.

[24] A. Gruner, A. Muhle, T. Gayvoronskaya, and C. Meinel. "A quantifiable trust model for blockchain-based identity management." In: Institute of Electrical and Electronics Engineers Inc., July 2018, pp. 1475–1482. ISBN: 9781538679753. DOI: `10.1109/Cybermatics{\_}2018.2018.00250`.

[25] *Handling Android App Links | Android Developers.* URL: `https://developer.android.com/training/app-links`.

[26] A. R. Hevner. *A Three Cycle View of Design Science Research.* Tech. rep. 2. 2007.

[27] *identity-hub/explainer.md at master · decentralized-identity/identity-hub.* URL: `https://github.com/decentralized-identity/identity-hub/blob/master/explainer.md`.

[28] Infura. *Infura Documentation.* URL: `https://infura.io/docs`.

[29] *IPFS Powers the Distributed Web.* URL: `https://ipfs.io/`.

[30] ISO. *Foreword - Supplementary information.* URL: `https://www.iso.org/foreword-supplementary-information.html`.

[31] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel. “A survey on essential components of a self-sovereign identity.” In: *Computer Science Review* 30 (2018), pp. 80–86.

[32] H. Ng. *The Issue with Current Centralized Identity Management System.* URL: `https://medium.com/usechain/the-issue-with-current-centralized-identity-management-system-c05aeafa8538`.

[33] Node.js Foundation. *Express - Node.js web application framework.* URL: `https://expressjs.com/`.

[34] *OpenID Connect Core 1.0.* URL: `https://openid.net/specs/openid-connect-core-1_0.html`.

[35] OpenJS Foundation. *Node.js.* URL: `https://nodejs.org/en/`.

[36] *OrgBook BC Public Beta.* URL: `https://orgbook.gov.bc.ca/en/home`.

[37] M. Paolini-Subramanya. *Why Key Rotation - Mahesh Paolini-Subramanya - Medium.* URL: `https://medium.com/@dieswaytoofast/why-key-rotation-f374c71b9c6f`.

[38] J. Poushter and R. Stewart. *Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies But advanced economies still have higher rates of technology use FOR MEDIA OR OTHER INQUIRIES.* Tech. rep. 2016. URL: `www.pewresearch.org.`.

[39] *RFC 8141 - Uniform Resource Names (URNs).* URL: `https://tools.ietf.org/html/rfc8141`.

[40] T. L. Saaty. *Decision making with the analytic hierarchy process.* Tech. rep. 1. 2008, pp. 83–98.

[41] M. Schäfer. “Analysis and Evaluation of Blockchain-based Self-Sovereign Identity Systems.” PhD thesis. Boltzmannstraße 3, 85748 Garching bei München: Technische Universität München, Nov. 2019.

[42] *Sign in with Apple - Apple Developer.* URL: `https://developer.apple.com/sign-in-with-apple/`.

[43] Sovrin Foundation. *Sovrin Foundation Launches Test Token for Decentralized Identity Network - Sovrin.* URL: `https://sovrin.org/sovrin-foundation-launches-test-token-for-decentralized-identity-network/`.

[44] Sovrin Foundation. *The Sovrin Network and Zero Knowledge Proofs.* Oct. 2018. URL: `https://sovrin.org/the-sovrin-network-and-zero-knowledge-proofs/`.

[45] Sovrin Foundation. *The Sovrin Network and Zero Knowledge Proofs - Sovrin.* URL: `https://sovrin.org/the-sovrin-network-and-zero-knowledge-proofs/`.

[46] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström. *JSON-LD 1.1.* URL: `https://json-ld.org/spec/latest/json-ld/`.

[47] StatCounter. *Desktop vs Mobile vs Tablet Market Share Worldwide*. URL: `https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#quarterly-200901-202001`.

[48] G.-H. Tzeng and J.-J. Huang. *Multiple attribute decision making : methods and applications*. CRC Press, 2011. ISBN: 1439861579.

[49] uPort. *uport-project/ethr-did: Create ethr DIDs*. URL: `https://github.com/uport-project/ethr-did`.

[50] uPort Team. *EthereumDIDRegistry*. URL: `https://github.com/uport-project/ethr-did-registry/blob/develop/contracts/EthereumDIDRegistry.sol`.

[51] uPort Team. *ETHR DID Method Specification*. URL: `https://github.com/decentralized-identity/ethr-did-resolver/blob/develop/doc/did-method-spec.md`.

[52] US Department of Homeland Security. *News Release: DHS Awards 159K to Prevent Credential Fraud | Homeland Security*. URL: `https://www.dhs.gov/science-and-technology/news/2019/11/12/news-release-dhs-awards-159k-prevent-credential-fraud`.

[53] US Directorate of Enforcement and US Ministry of Finance. *:: 8.12 : 'Know Your Customer' (KYC) Guidelines - Anti-Money Laundering Standards ::* URL: `https://archive.is/20120801052416/http://www.directorateofenforcement.gov.in/KYC11.html`.

[54] F. Vogelsteller and T. Yasaka. *ERC-725 Proxy Account*. URL: `https://github.com/ethereum/EIPs/issues/725`.

[55] W3C. *Linked Data*. URL: `https://www.w3.org/standards/semanticweb/data`.

[56] W3C Credentials Community Group. *A Primer for Decentralized Identifiers*. URL: `https://w3c-ccg.github.io/did-primer/`.

[57] W3C Credentials Community Group. *DID Context*. URL: `https://www.w3.org/ns/did/v1`.

[58] W3C Credentials Community Group. *DID Method Registry*. URL: `https://w3c-ccg.github.io/did-method-registry/#the-registry`.

[59] W3C Credentials Community Group. *Use Cases for Decentralized Identifiers*. URL: `https://w3c-ccg.github.io/did-use-cases/`.

[60] W3C Credentials Community Group. *W3C Credentials Community Group Work Items | community*. URL: `https://w3c-ccg.github.io/community/work_items.html`.

[61] W3C Credentials Community Group, N. Otto, S. Lee, B. Sletten, D. Burnett, M. Sporny, and K. Ebert. *Verifiable Credentials Use Cases*. URL: `https://www.w3.org/TR/vc-use-cases/`.

[62] W3C Credentials Community Group, D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, and M. Sabadello. *Decentralized Identifiers (DIDs) v1.0*. Nov. 2019. URL: `https://www.w3.org/TR/did-core/`.

[63] W3C Credentials Community Group, M. Sporny, and D. Longley. *Identity Credentials 1.0*. URL: https://opencreds.org/specs/source/identity-credentials/.

[64] W3C Credentials Community Group, M. Sporny, D. Longley, and D. Chadwick. *Verifiable Credentials Data Model 1.0*. URL: https://www.w3.org/TR/vc-data-model/.

[65] W3C Credentials Community Group, M. Sporny, D. Reed, and O. Steele. *Linked Data Cryptographic Suite Registry*. URL: https://w3c-ccg.github.io/ld-cryptosuite-registry/.

[66] W3C Credentials Community Group, O. Terbu, I. Basart, K. Den Hartog, C. Lundkvist, D. Stark, D. Zagidulin, D. Strockis, and O. Steele. *Self-Issued OpenID Connect Provider DID Profile*. URL: https://identity.foundation/did-siop/.

[67] J. Webster and R. T. Watson. *Analyzing the past to prepare for the future: Writing a literature review*. Tech. rep. 2. 2002. URL: http://www.misq.org/misreview/announce.html.