SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Information Systems

Design and Implementation of a Decentralized Trusted Issuer Registry for Self-Sovereign Identity

Michael Schmidmaier

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Information Systems

Design and Implementation of a Decentralized Trusted Issuer Registry for Self-Sovereign Identity

Entwurf und Implementierung eines dezentralen Registers vertrauenswürdiger Aussteller für Self-Sovereign Identity

Author:	Michael Schmidmaier
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Felix Hoops
Submission Date:	December 15, 2023

I confirm that this bachelor’s thesis is my own work and I have documented all sources and material used.

Munich, December 15, 2023 Michael Schmidmaier

Acknowledgments

First, I want to thank my advisor, Felix Hoops, for his invaluable support and continuous guidance throughout the last months. His feedback and ideas kept me on track and allowed me to work on a relevant topic that really interests me.

My sincere thanks also go to Prof. Dr. Florian Matthes, who gave me the opportunity to write my thesis at his chair and provided helpful and constructive feedback along the way.

Finally, I would like to express my warmest thanks to my family, friends, and girlfriend. Their support and encouragement helped me a lot, especially during the most challenging times.

Abstract

Verifiable Credentials enable subjects to make claims about one another whose integrity and authenticity can be securely verified. Nevertheless, verifying an issuer's trustworthiness is not possible from the credential alone. Trusted Issuer Registries (TIRs) were created to address this issue. However, the existing designs do not fully meet the needs of Self-Sovereign Identity. This thesis aims to design a novel decentralized, general-purpose Trusted Issuer Registry architecture. To that end, we analyzed existing TIR designs and gathered user requirements from an actual use case using interviews with five experts from the Gaia-X project. The findings indicated, among others, the necessity for issuer identification, authorization, and hierarchical sub-registries, alongside a focus on availability, scalability, security, and flexibility. From that, we inferred an improved TIR design, which was implemented as a prototype. The design leverages Decentralized Identifiers to reference TIRs and leaves implementation-specific details unspecified, specifying only a data model to which every TIR must resolve. The TIR's implementation proves the concept and demonstrates its capabilities. Ultimately, this thesis proposes an improved Trusted Issuer Registry design, contributing to the enhancement of issuer verification in Self-Sovereign Identity.

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**
1.1 Problem Statement 1
1.2 Research Questions 2
1.3 Structure 3

2 Background **4**
2.1 Self-Sovereign Identity 4
2.2 Decentralized Identifiers 4
2.3 Verifiable Credentials 5
2.4 Trusted Issuer Registries 5
2.5 Gaia-X 6
2.6 Tezos 6

3 Related Work **7**

4 Analysis of Existing Trusted Issuer Registry Designs **8**
4.1 Internet X.509 Public Key Infrastructure 9
4.2 European Blockchain Services Infrastructure 11
4.3 TRAIN 13
4.4 Digital Credentials Consortium 15
4.5 Open Credentialing Initiative 16
4.6 Trust Over IP Foundation 18
4.7 Discussion 20
4.7.1 Limitations 23

5 Requirements Analysis with Expert Interviews **25**
5.1 Methodology 25
5.2 Interview Results 26
5.2.1 Functional Requirements 26
5.2.2 Non-functional Requirements 27

5.2.3 Non-Requirements . . . 29
5.2.4 Other Findings . . . 29
5.3 Discussion . . . 30

6 Improved Trusted Issuer Registry Design 31
6.1 Design Overview . . . 31
6.2 TIR Referencation & Discovery . . . 31
6.3 TIR Methods . . . 33
6.4 Data Model . . . 33
6.5 Example TIR Methods . . . 39
6.6 Discussion . . . 40
6.6.1 Requirements . . . 40
6.6.2 Design Decisions . . . 43
6.6.3 Limitations . . . 44

7 Prototype Implementation 46
7.1 General Considerations . . . 46
7.2 Tezos Smart Contract . . . 47
7.2.1 Endpoints . . . 47
7.2.2 Data Model . . . 48
7.3 Core Package . . . 50
7.3.1 TIR Resolution . . . 50
7.3.2 Issuer Verification . . . 51
7.4 Backend . . . 52
7.5 Frontend . . . 52
7.5.1 TIR Management . . . 53
7.5.2 Issuer Verification . . . 53

8 Conclusion 58

Abbreviations 60

List of Figures 61

List of Tables 62

Bibliography 63

1 Introduction

In today's digital world, users usually rely on centralized authorities to manage their digital identity. However, this approach leads to potential risks of data breaches, unauthorized access, and lack of user autonomy [1]. As a solution, Self-Sovereign Identity (SSI) is often being proposed: An emerging approach to digital identity that aims at giving individuals complete control over their own identity [2]. Regarding SSI, the World Wide Web Consortium (W3C) has standardized the concepts of Decentralized Identifiers (DIDs) [3] and Verifiable Credentials (VCs) [4], which allow entities to create identities themselves and claims to be made about them without the need for a central registry. Since VCs are signed with public-key cryptography, their cryptographic integrity and authenticity can be easily verified. Nevertheless, verifying the issuer's trustworthiness remains a significant problem: without further information, no statement can be made as to whether the issuer can be trusted [5]. To enable automatic verification of issuers while conforming to the SSI principles, trust anchor mechanisms enabling verifiers to delegate trust can be used. One trust anchor method that has already been applied to Verifiable Credentials is the concept of Trusted Issuer Registries (TIRs): registries that, in their simplest form, contain a list of trustworthy issuers. Although architectures for TIRs have already been proposed and used in practice [6], further research is needed to overcome their shortcomings. This thesis aims to propose a new Trusted Issuer Registry design that addresses the drawbacks of existing solutions. To that end, we will first analyze existing solutions to this problem and gather requirements from an actual use case of SSI. From these results, we will infer how an improved decentralized Trusted Issuer Registry can be designed and implemented.

1.1 Problem Statement

The core concepts of SSI have already been standardized as W3C recommendations. With DIDs [3] exists a way to uniquely identify subjects in a decentralized manner. With VCs [4] exists a way for subjects to make verifiable claims about one another. For example, a government could assert a DID that it belongs to a certain real person, or a university could issue digital bachelor certificates. While the two standards may not be flawless in every aspect, they provide good core functionality to build

upon. The current problems of SSI rather remain in niche areas that have not been specified in detail. An example of this is the verification of a subject's credibility [5].

When using verified credentials, the verifier has to verify not only the cryptographic signature but also the issuer's trustworthiness: Is the issuer qualified and allowed to issue such a credential? Is the issuer who he claims to be? In other words, can the issuer be trusted? This verification can be performed in two ways: manually and automatically.

Manual verification can be performed in multiple ways, e.g., with extended online research. Unfortunately, this option is not reasonable for a large-scale appliance of Verifiable Credentials. Additionally, this requires knowing with certainty the issuer's legal identity. This is a problem of its own since, in general, a DID is not associated with a legal identity. For VCs to become widely accepted and to show their advantages over conventional physical certificates, an automated method for verifying the issuer's credibility is needed.

An automated verification has to build upon existing trust as a so-called root of trust. One solution is to rely on Trusted Issuer Registries: registries that verifiers can search to check whether a certain issuer is credible. Although architectures for TIRs have already been proposed, further research is needed to address their existing drawbacks. For example, most approaches show shortcomings in one or more of the following characteristics: true decentralization, scalability, security, and universal applicability. Regarding security, many existing approaches lack sufficient integrity protection and permissions to limit delegated trust. For example, educational institutions might be fully trusted to issue educational credentials but not for every other type of credential, like driving licenses. Therefore, there is a need for further research into Trusted Issuer Registries and how they can be designed to meet the needs of Self-Sovereign Identity.

1.2 Research Questions

This work is guided by three main research questions and their related sub-questions. To build upon requirements from an actual use case, we decided on the European digital infrastructure project Gaia-X, which will be introduced in the next chapter.

RQ1: What are the advantages and disadvantages of existing centralized and decentralized Trusted Issuer Registry designs?

RQ2: How can a general-purpose Trusted Issuer Registry be designed to meet the needs of Self-Sovereign Identity in Gaia-X ecosystems and address the drawbacks of existing solutions?

RQ2.1: What specific functionalities should a Trusted Issuer Registry provide in Gaia-X ecosystems?

RQ2.2: What are the requirements for a Trusted Issuer Registry in Gaia-X ecosystems?

RQ2.3: What is a suitable technical infrastructure for a Trusted Issuer Registry?

RQ2.4: How can scalable governance be achieved?

RQ3: How can the design be implemented using a concrete technology?

1.3 Structure

This thesis's structure closely follows the research questions. Chapter 2 provides the required basic background information. In Chapter 3, we will shortly discuss related work to this thesis. However, most related work is discussed in Chapter 4 in which we analyze existing Trusted Isser Registries and discuss their advantages and disadvantages in multiple dimensions to answer **RQ1**. In Chapter 5, we conduct a requirements analysis from the user's perspective using expert interviews, answering **RQ2.2**. Based on the previous results, we will then derive an improved Trusted Issuer Registry design in Chapter 6 to answer **RQ2**. We will then address **RQ3** by implementing our design as a prototype in Chapter 7. Finally, we will conclude the thesis and summarize our work in Chapter 8. Because of this thesis's size, the four main chapters, 4, 5, 6, and 7 encompass their related methodology, results, and discussion together in one chapter.

2 Background

2.1 Self-Sovereign Identity

Self-Sovereign Identity (SSI) is an emerging approach to digital identity that places individuals at the center of controlling their personal identity data [2]. In this model, users have the autonomy to manage their own identity, control the sharing of their personal data, and provide consent on how and when their information is used. SSI shifts the control of identity from centralized authorities and corporations to the individual, empowering them with privacy, security, and full ownership of their identity. This approach is facilitated through decentralized ledger technologies, which provide secure and transparent governance without relying on any central authority [7]. SSI enables the creation of digital identities that are portable, persistent, private, and entirely controlled by the individual [2]. This vision is implemented in the Decentralized Identifiers and Verifiable Credentials specifications, which will be discussed in the following.

2.2 Decentralized Identifiers

Decentralized Identifiers (DIDs) [3], standardized by the W3C, are a key component in the architecture of Self-Sovereign Identity. A DID is a unique identifier that is fully controlled by the individual, organization, or entity it represents, without the need for any centralized authority or intermediary. This autonomy is facilitated by the use of various DID methods, which are specific mechanisms or protocols defined within the DID ecosystem. These methods dictate how DIDs are generated, resolved, updated, and deactivated, adhering to a set of standardized rules. DID methods enable a variety of underlying infrastructures, for example, simple web servers and blockchains, but DIDs can also be created merely from public key material. Currently, the DID Specification Registry lists over 180 unique DID methods [8].

DIDs resolve, depending on the DID method, to a so-called DID Document. This document is a specific JSON object that associates the DID with data like public keys or service endpoints. The associated data itself can be directly referenced with DID URLs.

In general, DIDs are URIs [9] consisting of the DID URI scheme, a DID method identifier, and a DID method-specific identifier: **did:<method>:<identifier>**. To give an example, this would be a DID using the web method, which creates an identity from a domain name or a web URL: **did:web:example.com**. In this case, a DID resolver would retrieve the DID Document from *https://example.com/.well-known/did.json*. To, for example, directly reference a verification method, a DID URL like this can be used: **did:web:example.com#key1**.

2.3 Verifiable Credentials

Verifiable Credentials (VCs) are digital credentials conforming to the 2022 "Verifiable Credentials Data Model v1.1" W3C standard [4] that aim to be machine-readable and securely verifiable. They are regarded as an essential building block of SSI, allowing users to prove their identity or qualifications while preserving privacy [7]. Work is currently underway on version 2 of the standard, which is intended to be significantly more detailed and mature [10].

VCs consist of three high-level parts: metadata, claims, and proofs. The metadata comprises properties like a credential type, issuer, and issuance date. The issuer can be a DID but could also be any other URI. The claims can be any sort of data, for example, a digital diploma. The proofs are one or more signatures that allow verifying the credential's contents and the issuer's authorship.

Verifiable Credentials can be represented in either JSON [11] or JSON-LD [12] syntax. As a special form of proof, a VC can also be represented as a JSON Web Token (JWT) [13], enabling for example selective disclosure of credential claims [14]. The European Union adopted the JWT representation in its specification for a European digital identity wallet [15].

With Verifiable Credentials, there are three roles that entities can take. The first one is the **issuer** who creates and issues credentials. Then there is the **holder**, who is the subject of a credential and owns it. If the holder decides to present his credential to someone, this entity is a **verifier**. Verifiers verify the signature of a credential and decide whether they trust the issuer.

2.4 Trusted Issuer Registries

As described before, the problem of verifying an issuer's trustworthiness can be solved by introducing a Trusted Issuer Registry (TIR), a form of trust anchor. TIRs are registries storing, in their simplest form, a list of trusted issuers and can then be searched by verifiers to check the status of a specific issuer. Existing TIRs are often

more sophisticated and offer different functionalities. Those will be discussed in detail in Chapter 4. In this thesis, we will use the following specific terms regarding TIRs.

- **TIR maintainer:** A subject governing a TIR. A TIR can have multiple maintainers.
- **Sub-registry:** Some TIRs allow delegating trust to other TIRs, creating a hierarchy. From the perspective of the delegating TIR, the trust-receiving TIRs are referred to as sub-registries.
- **Credential schema:** TIRs may use schemas like JSON Schema [16] to specify which types of credentials an issuer may issue. Those are referred to as credential schemas.

2.5 Gaia-X

Gaia-X is a large European state-subsidized project aiming at building a secure, interoperable data infrastructure that allows sovereign sharing of data and services [17]. The project was started in 2019 by the German and French Ministries of Economic Affairs and currently consists of more than 350 members from industry and science. Today, the project is still under active development. Gaia-X has committed to the values of Self-sovereign Identity and extensively utilizes Verifiable Credentials [18]. For this reason, we chose the project as the source of requirements for our own Trusted Issuer Registry design.

2.6 Tezos

Tezos [19] is a public and open-source blockchain that distinguishes itself through its self-amendment functionality. Launched in 2018 [20], it utilizes a Proof-of-Stake (PoS) consensus mechanism, which is more energy-efficient than the Proof-of-Work (PoW) system used by networks like Bitcoin. A special feature of Tezos is its on-chain governance model, allowing stakeholders to vote on protocol upgrades and amendments, enabling the blockchain to evolve over time without the need for hard forks. Furthermore, Tezos supports smart contracts and thus offers a platform for developing decentralized applications.

3 Related Work

Existing Trusted Issuer Registry Designs

Several Trusted Issuer Registry designs already exist. We will analyze them in detail in Chapter 4 and will, therefore, not go into them further here.

Survey Papers

In late 2022, Sporny et al. [6] published a white paper as part of the Rebooting the Web of Trust XI design workshop, which also conducted an analysis of existing Trusted Issuer Registry designs. Their analysis covers several approaches we have also examined, but it is rather unstructured, undetailed, and contains few sources. They also derived a design from the analysis, but as it did not introduce new concepts and was not specified in detail, we did not include it in our analysis.

To the best of our knowledge, no other survey paper for existing Trusted Issuer Registry designs exists.

eIDAS

The EU Regulation 910/2014 [21], also known as eIDAS ("electronic IDentification, Authentication and trust Services"), establishes a comprehensive framework for electronic identification and trust services for electronic transactions within the European Union. The regulation requires member states to publish trust lists, which are official records of trusted entities authorized to provide electronic identification and trust services. These trust lists aim to solve the same problem as Trusted Issuer Registries. Therefore, their format, specified as the European Telecommunications Standards Institute (ETSI) TS 119 612 standard [22], is relevant for the design of interoperable TIRs and has already been leveraged in TRAIN [23], a TIR we analyze in Chapter 4.

4 Analysis of Existing Trusted Issuer Registry Designs

For our first research question, we researched online to identify existing Trusted Issuer Registry designs and similar technologies. We then selected six solutions that were representative of their concept for further evaluation. These existing solutions were then compared and examined for their advantages, disadvantages, and noteworthy characteristics. Our analysis is limited to the information described publicly in specifications or documentation. We have not tested the solutions in practice.

Analysis Structure

From comparing the different approaches, we extracted common characteristics for easier classification and structuring of our analysis results. Our classification, which will be explained in the following, does not claim to be a complete taxonomy but is merely intended to allow an organized comparison of the various designs.

Trust Concept The analyzed solutions apply different concepts. Potential concepts are Trusted Issuer Registry, certificate chains, or a combination of both.

Use Case Naturally, TIRs are designed for a specific application. This can be a specific use case or a more general-purpose use.

Storage TIRs are also characterized by their type of storage. A TIR's contents could be, e.g., stored on a centralized web server, decentralized on a blockchain, or a combination of both.

Functionality TIRs can offer different functionalities, ranging from basic ones to a wide range of features like issuer identification or authorization.

Scalability TIRs can have specifically documented design decisions for better scalability. This can be a hierarchical structure to improve governance delegation but also measures like redundant endpoints or other scaling data structures.

Performance TIRs can, for example, have specific features to facilitate caching or utilize efficient data structures that ultimately improve performance.

Security A TIR's security depends on many factors. Those depending on the architecture are, for example, the TIR's abilities to limit delegated trust or its integrity protection.

Complexity A TIR is also characterized by its complexity, which is defined by factors like the simplicity of the TIR's setup, its number of features, and the usability of managing its contents.

4.1 Internet X.509 Public Key Infrastructure

Although it is not a Trusted Issuer Registry, a comparison with the Internet X.509 Public Key Infrastructure (PKI) [24] is useful when talking about trust anchors. Just as with Verifiable Credentials that can be securely verified using cryptography, communication partners on the internet (usually client & server) can establish cryptographically secure communication using TLS. However, at this point, the server's identity has not yet been confirmed. This problem is solved by the server presenting an X.509 certificate confirming its identity (a Domain Name System (DNS) domain name). A hierarchical public key infrastructure is required to verify this certificate. At the highest level are the Certificate Authorities (CAs), which, after an identity check, issue certificates to servers across potentially further intermediate Certificate Authorities. This creates a chain of trust, an unbroken chain of certificates from the CA to the server. The client usually has a preconfigured list of root certificates from the certificate authorities that are regarded as trustworthy. If the certificate chain presented by the server begins with a trusted root certificate and the chain is unbroken, the server's identity is verified. This method allows the client to be sure that their communication partner is actually who they claim to be.

Trust Concept In contrast to the other existing solutions, the TLS certificate system uses a different concept to that of the Trusted Issuer Registry: certificate chains, also known as chains of trust [24, Ch. 4.4.2.]. In this approach, the trust anchors issue a trust certificate to each entity they classify as trustworthy. If these, in turn,

certify their trust to another entity, a chain of certificates is created from the trust anchor to the last link. If an entity now wants to prove its trustworthiness, it can simply present the certificate chain. This approach has some advantages but also disadvantages compared to TIRs, which we will discuss below.

Use Case The primary use case of the Internet PKI is to enable secure communication with the objectives of confidentiality, integrity, and authenticity. Using the terms of TIRs, it therefore enables issuer identification but not authorization. Since the Internet PKI makes no assumption on its use cases, we classify it as general-purpose.

Storage As explained above, the Internet PKI uses the concept of certificate chains. Therefore, in its basic form, no data about trusted entities is stored publicly. Certificate chaining can be regarded as a very decentralized form of storage.

Functionality As previously noted, the Internet PKI offers issuer identification but no authorization.

Scalability, Performance Due to the decentralized nature of certificate chains, the system is not dependent on the reliable availability of servers and is therefore not limited in its functionality at large scales. The strict hierarchy also allows for good scalability in terms of governance. The widespread use of HTTPS on the Internet underlines its scalability in practice.

Security The concept of certificate chains has some security implications. For example, it is more privacy-preserving than a registry, which verifiers must query and thus may collect private information on verifications taking place. The absence of a registry also enables a theoretical availability of 100%. On the other hand, this places a greater burden on holders, as they have to present the complete and up-to-date certificate chain for each verification process.

However, certificate chaining in its basic form also enables certificates to be issued without the knowledge of those concerned [25]. This enables attackers to issue false certificates unnoticed after compromising a certificate authority and use them for man-in-the-middle (MITM) attacks. This already happened in 2011, for example, in the attack on DigiNotar, in which rogue Google certificates were used for MITM attacks [26].

Another characteristic feature is that the list of trusted root certificates in browsers is already preconfigured by the manufacturer and is not intended to be

configured by the user. This pre-configuration gives browser manufacturers a significant influence over the entire certificate system. Google, for example, has made it a requirement after the DigiNotar incident that CAs additionally comply with the Certificate Transparency Standard [27]. The Google-developed standard requires that CAs enter all certificates issued by them in a public log, enabling the public to detect the issuance of rogue certificates [28]. Google was able to enforce this requirement thanks to its considerable influence as the manufacturer of the Chrome browser [29].

Finally, the Internet PKI also provides the equivalent to sub-registry authorization: name constraints for certificate authorities [30]. With those name constraints, certificate authorities can be limited to only issuing certificates for certain subdomains.

Complexity The system's base concept of chaining certificates is relatively simple. Nevertheless, its extensions like the previously mentioned Certificate Transparency or Certificate Authority Authorization [31] complicate the system as a whole.

4.2 European Blockchain Services Infrastructure

In 2018, the European Union's member states, Norway, Liechtenstein, and the European Commission founded the European Blockchain Partnership (EBP). One of the EBP's goals is the creation of the European Blockchain Services Infrastructure (EBSI), whose key component is a public permissioned blockchain [32]. The whole project is still under active development and not yet used in production [33].

One of EBSI's use cases is providing an infrastructure for the secure issuance and verification of digital credentials such as diplomas [34] or digital student IDs [35]. For verifying the issuers' trustworthiness, the Trusted Issuers Registry service [36], [37] is provided. It consists of an Ethereum smart contract deployed on the EBSI blockchain, managing the stored issuers and an API for interacting with the registry. Unfortunately, the Trusted Issuers Registry's public documentation was not very extensive until the end of November 2023, and all in all, the public information available was limited. The general documentation [37] and the API documentation [36] also appear to be inconsistent. For example, the general documentation refers to credential types, which in turn cannot be found in the data returned by the test API. For this reason, we mainly analyzed the architecture described in the general documentation.

Trust Concept EBSI utilizes a combination of the Trusted Issuer Registry concept and certificate chaining. All entities in the registry, e.g., trusted issuers, are both

stored in the registry and receive a verifiable credential, which could theoretically be used to form a certificate chain. However, this is not described in the documentation [37].

Use Case The high-level goal is to provide a trust anchor for cross-border services based on Verifiable Credentials that are not yet specified in detail. At the time of our analysis, three credential use cases were mentioned: identity credentials, educational credentials, and social security passes [32]. Nevertheless, EBSI mentions more future use cases like "Asylum Process Management" [32]. All in all, the EBSI use cases are centered around legal processes building upon the public permissioned blockchain. Therefore, we classify their use case as specific.

Storage The TIR is stored in an Ethereum smart contract deployed on EBSI's public, permissioned blockchain. The fact that the blockchain is permissioned fits the government use case but also has disadvantages. For example, the strong security guarantees of permissionless blockchains are weakened, as the node operators could theoretically make changes to the supposedly unchangeable data together. [38]. Although EBSI states that theoretically, all organizations from EBP member states can become node operators after applying, there will always be a residual risk of abuse of power due to centralization. Although it could be argued that this makes the storage only distributed, not decentralized, we classify it as decentralized.

Functionality The EBSI TIR provides issuer authorization with credential types and probably also with credential schemas. Credential Schemas are, for example, JSON Schemas [16] that specify which credentials an issuer may issue. Credential types refer to the type property of Verifiable Credentials and define which types an issuer is allowed to issue. To the best of our knowledge, there currently is no documentation about credential schemas, but their existence is proven by the VerifiableAccreditationToAttest data model [39] and verification code we found in the @cef-ebsi/verifiable-credential npm package[1]. In contrast, only issuer authorization using verifiable credential types is documented. Finally, the TIR does not provide issuer identification.

Scalability, Hierarchy, Governance To enable scalable governance, the EBSI TIR uses a strictly hierarchical concept of Trusted Accreditation Organisations (TAO), which is similar to the Certificate Authorities of the Internet PKI. In contrast to the

[1] @cef-ebsi/verifiable-credential, Version 5.0.0-alpha.5, file: /src/validators.ts

Internet PKI, there is only one single root: the TIR maintainer. Because there is only one EBSI TIR, this centralizes all power around EBSI.

Performance It is an advantage that the TIR's complete data is stored in a single smart contract because it avoids resolution steps between every level of hierarchy. Additionally, the documentation suggests caching of issuer verification results, which is facilitated by listening to certain TIR events that are emitted on changes that are unfortunately not further documented yet [37].

Security The EBSI TIR provides extensive delegation of trust with issuer identification and issuer authorization with credential types and schemas. The credential types and schemas provide a useful and flexible form of limiting the delegated trust. Additionally, the schemas are stored in a so-called Trusted Schema Registry on the EBSI ledger, can be referenced with a specific revision, and thus are integrity protected [40]. It is unclear from the documentation where the credential types are stored. We therefore assume that they are also stored securely in the Trusted Schema Registry. Nevertheless, there is a high centralization around EBSI despite their efforts for technical decentralization. As discussed in the storage section, this centralization undermines the security advantages of decentralization. For the relatively specific use case of EBSI, however, it does not appear to be a problem.

Complexity The system is rather complex and tries to achieve many things at once. Still, the REST API provides a relatively simple interface to interact with the smart contract [36].

4.3 TRAIN

The so-called Trust mAnagement INfrastructure (TRAIN) approach of Johnson Jeyakumar et al. [23], proposed in 2022, is another technical concept aimed explicitly at solving SSI's problem of decentralized trust. The approach builds on existing standards and established infrastructure. In general, it works by distributing trust lists via web servers. These trust lists contain information about the trusted issuers based on the existing ETSI standard TS 119 612 [22]. The TIR's structure is then defined with specific DNS entries forming a trust scheme. Trust schemes can point with URI resource records to locations of ETSI trust lists that should be included in the TIR. PTR resource records can be used to include the contents of other trust schemes. The *ServiceTypeIdentifier* of ETSI trust lists is used in TRAIN to link to JSON Schemas [16] stored on web servers that define the type of VCs the issuer may

issue. In September 2023, the Gaia-X Federation Services project invited tenders to implement a TRAIN component [41]. The tendered project was awarded to T-Systems International GmbH and is planned to be completed by February 2024 at the latest [42].

Trust Concept The TRAIN approach solely follows the Trusted Issuer Registry concept.

Use Case TRAIN is a general-purpose approach that makes few assumptions on its use case.

Storage The TIR's structure is stored on the DNS. The TIR's content is stored on web servers referenced by URLs. Although Jeyakumar et al. originally only intended HTTPS URLs, extending their design to a wide range of URI methods would be possible. Due to its strict hierarchy, it is debatable whether the DNS is truly decentralized. Instead, it is often only referred to as distributed. With the TIR's contents stored on web servers, the TIR's whole storage combines distributed and centralized storage.

Functionality TRAIN provides issuer identification and issuer authorization with external JSON credential schemas. There is no way to limit the trust delegated to sub-registries, who are always trusted completely. Compared to the other solutions, a rather unique feature of TRAIN is the support for non-DID issuer VCs, secured by X.509 certificates stored in the TIR.

Scalability To facilitate scalability, TRAIN allows hierarchical trust delegation by pointing to other Trust Schemes that are fully trusted.

Performance TRAIN has no specifications regarding performance, like caching policies. The DNS, however, has its own caching mechanisms in place [43], which can be leveraged to improve TIR resolution speed.

Security To counteract possible attacks on the DNS, like cache poisoning, TRAIN explicitly recommends the usage of DNS Security Extensions (DNSSEC), which enhance DNS security by providing identification and integrity to DNS data through digital signatures. Nevertheless, DNSSEC is not enforced by TRAIN, thus creating a potential attack vector. Additionally, as explained above, TRAIN includes issuer authorization with credential schemas. These schemas, however, are only referenced

by a URL. There is no mechanism in place that ensures the schemas' integrity, which creates an attack vector [4, Ch. 8.2]. Furthermore, when linking to other trust schemes, this creates an unlimited trust delegation. There is no way to restrict the delegated trust. This problem will be analyzed in more detail in the summary at the end of this chapter. Finally, the centralized storage of schemas and trust lists is a disadvantage of TRAIN because it creates a single point of failure for availability.

Complexity In general, the TRAIN approach is relatively simple and has no significant barriers to set up. One of the reasons for this is that TRAIN is based on established technologies such as web servers, the DNS, and ETSI Trust Lists. The use of ETSI Trust Lists, however, also introduces some complexity. The schema utilizes XML and was originally not designed for the SSI use case, leading to high verbosity.

Additionally, regarding usability, the design choice to use the DNS for storing the TIR's structure is problematic. Some domain providers like Cloudflare[2] offer APIs for managing DNS settings, but many do not. This complicates managing TIRs and potentially limits the choice of domain providers.

4.4 Digital Credentials Consortium

The Digital Credentials Consortium (DCC) is an initiative of universities and academic institutions from Europe and North America that explores how academic credentials can be digitalized securely [44]. Their approach utilizes Verifiable Credentials and Decentralized Identifiers and thus also faces the problem of decentralized trust [45]. As a solution, in 2020, they created the "Issuer Registry MVP" [46], a simple and minimalistic TIR, which has apparently only been used for testing purposes so far. The TIR is essentially a simple JSON object containing metadata like a timestamp of the last update and an issuer's map containing the issuer's DIDs and some additional identity information.

Trust Concept The Issuer Registry MVP is clearly a Trusted Issuer Registry.

Use Case The TIR's use case is to provide lightweight trust to verifiers that verify VCs issued by DCC members. As its title says, the TIR is an early prototype and aims to provide a first point of reference, yet to be a complete trust infrastructure. In general, the use case is specific and only comprises the DCC's academic applications.

[2] https://developers.cloudflare.com/api/operations/dns-records-for-a-zone-update-dns-record

Storage The TIR is meant to be stored centrally on web servers as plain JSON and existing test instances are currently hosted with GitHub pages[3].

Functionality The TIR provides issuer identification but no issuer authorization. DIDs of issuers are connected with the issuer's legal name, location, and website.

Scalability There are no scalability measures like hierarchy specified. With rising issuers, the TIR's JSON object would simply grow in size.

Performance Since the TIR is stored in one place, only one resolution step is required. In addition, the issuers, indexed by their DID, are stored in a JSON object. In contrast to an array, the objects are therefore automatically stored in a quickly retrievable data structure after parsing.

Security In its simplicity, the TIR does not employ special security measures. For example, there is no issuer authorization.

Complexity The DCC Issuer Registry MVP is very simple. This has the advantage that it is easy to set up for TIR maintainers and easy for verifiers to verify issuers. Nevertheless, due to its simplicity, there are many features missing that might be beneficial for certain use cases. Regarding usability, the TIR's first implementation stored on GitHub utilizes basic pull-request functionality for making and verifying changes. This manual process, however, has the disadvantage that mistakes can happen. For example, we found a change that forgot to update the TIR's "updated" property[4]. Nevertheless, this could easily be mitigated using a custom frontend.

4.5 Open Credentialing Initiative

The Open Credentialing Initiative (OCI) is a project of multiple companies from the pharmaceutical industry that creates technologies supporting their supply chain based on Verifiable Credentials and Decentralized Identifiers [47]. To support their use case, they also created a Trusted Issuer Registry [48]. It is based on an Ethereum smart contract that manages so-called namespaces essentially containing maps[5] of hashed issuer DIDs mapped to their boolean trust status. Each namespace,

[3] TIR instance linked to in the TIR's specification, containing real DIDs and test DIDs: https://digitalcredentials.github.io/issuer-registry/registry.json
[4] https://github.com/digitalcredentials/issuer-registry/commit/57e325b
[5] To be precise, namespaces contain lists of key-value pairs.

corresponding to one TIR instance, is solely managed by one Ethereum account. Every issuer map in a namespace corresponds to a credential type's hashed JSON-LD context URL. For this credential type, the map now defines for hashed issuer DIDs whether they are trusted or not. The TIR is governed decentrally by multiple so-called statekeepers who together use the multi-signature wallet Safe-Global[6]. They thus decide on TIR changes by voting [49, Ch. 7].

Trust Concept As its name already states, the OCI's created a Trusted Issuer Registry.

Use Case Although OCI's registry is built for their specific pharmaceutical use case, the TIR makes no assumptions on its use case and thus is applicable for general purposes.

Storage The TIR is stored as a whole decentrally on the Ethereum blockchain.

Functionality The TIR provides no issuer identification but issuer authorization with credential types. The lack of issuer identification is unique compared to the other designs and highlights that this functionality is not needed for a minimalistic TIR. It only allows checking whether an issuer DID is trusted for a certain credential type: trusted, not trusted, or not included in the TIR.

Scalability OCI's TIR does not support hierarchy. The documentation does not provide information on the scalability.

Performance The TIR's architecture does not explicitly facilitate caching. The fact that the entire TIR is stored in one place is beneficial for performance.

Security The TIR design has two noteworthy design decisions that also regard security: decentralized governance and identifier hashing. The TIR's statekeepers use a multi-signature wallet to decide on changes by voting. This improves security as no statekeeper himself can make malicious changes to the TIR and there is no single point of failure in general. For example, it is no problem when one statekeeper loses access to his wallet, gets compromised, or is ill.

Another remarkable design decision is the use of hashed identifiers: the credential type URLs and the issuer DIDs are only stored hashed in the smart contract. By that,

[6] https://safe.global/

uninvolved parties observing the smart contract cannot know which credential types and DIDs are used. Depending on the use case, this might be a reasonable privacy measure. Nevertheless, it might not be needed in many other cases, as we have seen with the previous solutions. Those build trust upon transparency and explicit issuer identification, which would render identifier hashing unnecessary.

Another relevant security aspect is that the linked credential type JSON-LD contexts are not integrity-protected [4, Ch. 8.2]. This would, for example, require a content hash of the credential type context. The hash of the identifier does not protect the integrity. Additionally, JSON-LD contexts are not designed for validating data and thus are very limited in defining allowed forms of credentials [12].

Complexity As the TIR design only provides relatively simple functionality and can be set up with a single smart contract, it can be classified as relatively simple. The multi-signature wallet adds a level of complexity; however, this is not necessarily required for the design but rather a policy of OCI. The used multi-signature wallet provides a graphical user interface to interact with the TIR and thus seems to provide a good user experience for the statekeepers.

4.6 Trust Over IP Foundation

The Trust Over IP (ToIP) Foundation is a consortium of over 300 companies collaborating on a new architecture of digital trust that is interoperable and decentralized [50]. Since they also leverage Verifiable Credentials for their work, they identified the problem of trust and created a version 1 working draft of their ToIP Trust Registry Protocol in September 2021 [51]. In contrast to the previous solutions, they do not specify precise technologies but only a simple HTTP API defined with OpenAPI. A noteworthy feature is that the registry not only stored trusted issuers but also trusted verifiers. The Trust Registry specification intentionally leaves some questions open for future consideration. Version 2 of the protocol is currently in development [52].

Trust Concept As its name implies, the design follows the Trusted Issuer Registry concept. However, it must be added that the registry contains both trusted issuers and verifiers.

Use Case Trust Over IP is working on a comprehensive digital trust architecture. For this reason, the design is general-purpose.

Functionality The ToIP Trust Registry only provides issuer and verifier authorization with credential types but no identification.

Storage, Scalability, Performance The specification only defines the user interface and no implementation details about the backend, for example, regarding storage, hierarchy, or governance.

Security The registry only defines credential/presentation type URIs and thus has the same issue as OCI's TIR: the credential type is not integrity-protected. This allows unnoticed changes to the credential type, which is a security issue [4, Ch. 8.2]. Furthermore, as with OCI's TIR, JSON-LD contexts are not designed for data validation [12] and thus are not a good choice for restricting credential usage.

Complexity With only an API specified, the TIR is easy to set up, and thus we classify it as simple.

4.7 Discussion

	X.509 PKI	EBSI	TRAIN	DCC	OCI	ToIP
TIR concept		✓	✓	✓	✓	✓
Chaining concept	✓	✓				
General-purpose	✓		✓		✓	✓
Storage type	**decentral.**	**decentral.**	**combinat.**	**central.**	**decentral.**	n/a
Stored in one place	n/a	✓		✓	✓	n/a
Issuer identification	✓		✓	✓		n/a
Issuer authorization		✓	✓		✓	✓
Hierarchy (sub-registries)	✓	✓	✓			n/a
Subregistry authorization	✓	✓		n/a	n/a	n/a
Caching intended		✓				
Fully integrity-protected	✓	**x**[7]		✓		

Table 4.1: Partial overview of the analyzed TIRs' distinctive characteristics

The solutions examined are very diverse and differ in various characteristics, as Table 4.1 shows with a partial overview of the most distinct characteristics. In the following, we will summarize and discuss the results and limitations of the analysis.

Trust Concept As shown, both the certificate chain concept and the registry concept have their advantages and disadvantages. Because this thesis is about Trusted Issuer Registries, certificate chains are out of scope, and we will not discuss it further.

Use Case The evaluated approaches have different use cases. While some, e.g., the TIRs of EBSI and the DCC, are specifically built for a limited use case, others, such as TRAIN, are designed for more general-purpose applications in their domain and are not bound to highly specific technologies or data models. There is no right or wrong regarding the specificity of the use case. For our work, however, the TIRs with a general purpose are the most relevant.

Storage The TIR approaches we analyzed use different storage concepts. A distinction can be made between centralized and decentralized storage systems. Regarding TIRs, we found two facets of centralization and decentralization in relation to storage: storage location and control over the stored data.

[7] As explained above, it is not fully clear but likely whether the EBSI TIR's credential schemas and types are integrity-protected because of inconsistent and vague documentation.

Centralized storage is, for example, data on ordinary web servers. The data is stored on a single server controlled by a single party. This solution is simple, fast, low-cost, and thus widely used for storing digital data. However, this results in a single point of failure and only limited possibilities for scaling [53], [54].

Decentralized storage, on the other hand, could be a smart contract on a blockchain or a file in the InterPlanetary File System (IPFS). There, data is stored on multiple nodes, and no single party has full control over the storage. This can provide resilience, almost full availability, data privacy, and integrity. Nevertheless, the advantages also come at a cost. Decentralized systems are significantly more complex, more expensive, often relatively slow, and rely on trust in potentially unknown third parties. Some features may also be undesirable. For example, the immutability of blockchains conflicts with data protection and therefore cannot be used in every way for legal reasons [53], [54].

In some cases, it is not clear whether a type of storage is decentralized or centralized. For example, the Domain Name System, leveraged in the TRAIN approach, is distributed across many servers, but there is a power imbalance between the individual servers due to the strict hierarchy of the DNS. Therefore, the DNS is often only referred to as distributed and not as decentralized [55, Ch. 2].

It is not easy to decide which type of storage is more suitable for a TIR, as both types have advantages and disadvantages. For TIRs, high availability seems to be an essential factor that can be easily achieved at lower costs with decentralized systems. Still, a final decision between decentralized and centralized storage can only be made in the context of an actual use case.

Functionality The analyzed TIRs had different functionalities, ranging from few to many. One basic feature provided by half of the TIRs is issuer identification. Although this is not needed for basic trust anchor functionality, connecting an issuer's DID with a legal identity can help strengthen the trust in issuers.

Another functionality is issuer authorization. Although all but the Internet PKI's[8] and the DCC's solutions support some form of issuer authorization, but it is not always well implemented. For example, OCI and ToIP use credential types, i.e., JSON-LD contexts, to restrict the credentials an issuer may issue. However, JSON-LD contexts are not intended for validating data and thus lack basic validation functionality that might be wanted [12, Ch. 1]. For example, it is not possible to enforce a property to have a specific value. Furthermore, some TIRs have no integrity protection for

[8] The Internet X.509 PKI does, of course, not have issuers like VC issuers. However, their equivalent, end-entity certificates, have no special authorization as they are not meant to issue further certificates.

the referenced credential schemas or contexts. This issue will be discussed further below in the security part.

A less common functionality is sub-registry authorization, which defines permissions for the hierarchy level below. The importance of this functionality when using hierarchy will be outlined below.

One final point, although out of scope but still interesting, is the verification of verifiers besides that of issuers, as provided by the ToIP TIR. Like verifiers do not know which issuer to trust, holders often might not know whether a verifier is trustworthy. In principle, this problem could be solved by a technology similar to a Trusted Issuer Registry, with little or no adaptation, e.g., regarding the authorization. For this thesis, however, this is out of scope.

Scalability For better scalability, three of the analyzed solutions introduce a hierarchical structure that allows trust to be delegated to sub-registries. However, one of them, TRAIN, does not have sub-registry authorization functionality and can thus only delegate full trust. The other analyzed designs do not have documentation regarding scalability.

Performance To improve performance, caching can be an option. However, only EBSI facilitates caching by emitting special change events. Another performance aspect is the number of resolution steps required to resolve the entire TIR. While most TIRs are stored collectively in one place and only require one resolution step, TRAIN requires one step for each path to a sub-registry.

Security We have identified multiple, partly recurring characteristics of security flaws. For example, TRAIN, OCI, and ToIP do not ensure the data integrity of external resources such as credential schemas. Simply referring to external resources can be dangerous. For example, malicious issuers could first verify themselves with a permitted schema and get themselves included in the TIR before then changing the schema. Of course, this would require the attacker to have access to the schema. However, external resources could also be changed unintentionally due to carelessness by their legitimate owner. In any case, it is undesirable for the contents of a TIR to be changed without action by the maintainer. A TIR is a single source of truth and must therefore only ever have a valid, intentional state.

In addition, there is the general problem of limiting delegated trust. Although the variety of solutions demonstrates the demand for trust delegation, transitive trust has known disadvantages that need to be addressed. The Verifiable Credentials Specification even explicitly refers to the disadvantages and advises users to be

aware of them [4, Ch. 5.2]. It is clear that there is rarely complete trust between two entities. Instead, we argue that two parties usually only trust each other in certain aspects. To give a real-world example, a university may be thoroughly trusted to issue diplomas but not to issue digital passports. However, since full trust is necessary for automated verification, a tradeoff must be made between simple, broad delegation and precise specification. One solution can be the use of credential schemas, as done by TRAIN and EBSI. However, TRAIN does not provide authorization for their trust registries. For the reasons just explained, it is advisable and important that, in addition to issuers, delegated trust to sub-registries can also be restricted.

Furthermore, multiple solutions have a high centralization around certain entities. Depending on the use case, this can be an issue. When pursuing the goals and values of Self-Sovereign Identity, it is important to try to avoid centralization. For example, in this case, the completely decentralized OCI TIR is preferable to the EBSI TIR.

Complexity

We have seen a variety of complexity, ranging from very simple approaches like the DCC Issuer Registry to relatively complex approaches like the ones from EBSI and TRAIN. There is no simple answer to an optimal solution. There is always a tradeoff between complexity and functionality. Ultimately, sufficient functionality must be provided with the lowest possible complexity for the corresponding use case.

Finally, usability is another important factor for a good TIR. While all solutions seem to provide adequate usability, we noticed that TRAIN is building upon the DNS to structure its TIR. As explained above, we regard this as problematic since not all domain providers have APIs that enable building an easy-to-use frontend for maintainers. In addition, frontends would need to integrate each individual domain provider separately, as there is no standardized interface. Such aspects must be taken into account when designing a TIR.

4.7.1 Limitations

Our findings provide a valuable overview of common characteristics of existing solutions. They can serve as a basis for more detailed evaluations and the development of improved Trusted Issuer Registry designs. However, our analysis is limited to a subset of all solutions that we consider unique and representative of certain concepts. Even if there are currently few trusted issuer registries in principle, as the associated technologies are still emerging, there may be TIRs unknown to us whose analysis could be relevant for an overview. Furthermore, our review is limited to solutions with public documentation or specifications. We also did not test the analyzed solu-

tions in practice. The detailed analysis of all existing designs with potential practical tests regarding, e.g., scalability and performance and classification in a sophisticated taxonomy remains a topic for future research.

5 Requirements Analysis with Expert Interviews

5.1 Methodology

As the concept of Trusted Issuer Registries and SSI in general is still very young, there are currently only a few clearly defined use cases and no generally established requirements. To address this, we conducted semi-structured expert interviews from an actual use case to explore the requirements for Trusted Issuer Registries. For the actual use case, we decided on the Gaia-X project which has committed to SSI and utilizes Verifiable Credentials extensively [18]. The interviews, lasting 45 to 60 minutes, were conducted digitally over Zoom in either English or German, depending on the interviewee. The sessions were recorded and transcribed for detailed analysis. After the interviews, we extracted common themes and requirements from the participants' responses to derive the final results.

Interview Partners

The Gaia-X experts were selected from personal contacts with one degree of separation, based on their experience in specific subprojects of the GAIA-X 4 Future Mobility project family. This makes them potential users and beneficiaries of Trusted Issuer Registries with specific requirements. Thus, their expertise provided valuable insights into the practical requirements of such systems. Before the interviews, each participant received a brief introduction (5-10 minutes) to Self-Sovereign Identity and the concept of Trusted Issuer Registries. This step ensured that all interviewees had a basic understanding of SSI despite it not being their specific field of expertise.

Interview partner A is a Senior Researcher and has been in this position for about one year. In GAIA-X, A is working on a service that will use SSI for user authentication, authorization, and user information retrieval.

Interview partner B is a PhD Candidate and has been in this position for about two years. In GAIA-X, B is working on a different service than A that will use SSI for user authentication and authorization.

Interview partner C is a Systems Architect and has been in this position for about

seven years. In GAIA-X, C is working on multiple subprojects. One of them will use SSI for authentication and authorization.

Interview partner D is a Researcher and Software Developer and has been in this position for about one year. In GAIA-X, D is working on a trust anchor service that will attest membership in an industrial association, among other things.

Interview partner E is a Scientific Project Manager and has been in this position for about six years. In Gaia-X, E has the role of a Project Owner and is working on multiple subprojects using SSI for authentication and authorization.

5.2 Interview Results

From the interview results, we derived the following requirements for an improved TIR design. Due to the lack of established existing TIR designs, some requirements are less precise than desirable and rather formulated as goals.

5.2.1 Functional Requirements

For simplicity, we only define core functional requirements. The first two are basic write functionality for managing the registry.

- **FR1: Register Trusted Issuer:** The TIR must enable registering new trusted issuers.
- **FR2: Update Trusted Issuer:** The TIR must enable updating existing trusted issuers. This includes updating certain attributes and deleting issuers.

All respondents indicated a preference for larger hierarchical TIRs over smaller, use case-specific TIRs for convenience. For this reason and because it flexibly improves scalability, the TIR must provide functionality to endorse sub-registries.

- **FR3: Endorse Sub-Registries:** The TIR must allow delegating trust to sub-registries and thus create an additional hierarchy level.

Finally, the TIR must provide functionality for verifying an issuer's trustworthiness as it is its whole purpose. For the use case of Gaia-X, this includes verifying an issuer's identity and often also verifying an issuer's qualification to issue a particular credential. These aspects are also related to the security requirements and therefore elicited further below.

- **FR4: Verify Issuer's Trustworthiness:** The TIR must allow verification of an issuer's trustworthiness.

5.2.2 Non-functional Requirements

Security

Gaia-X pursues the goal of exchanging valuable commercial data, which therefore also requires special protection. For this reason, security measures must be taken to prevent major damage. All respondents clarified that their use case requires the secure and reliable verification of VCs. Additionally, as B's use case involves a large number of international issuers that might not all be thoroughly trusted, the TIR design must minimize the possibilities of a malicious issuer to cause harm. The 'principle of least privilege' shall be applied. From the findings, we infer the following requirements.

- **NFR1: Issuer Authorization**: The TIR must allow specifying specific issuer qualifications.
- **NFR2: Limit Sub-Registry Trust Delegation:** The TIR should allow limiting the delegated trust for endorsed TIRs.

Although it should not happen, it is foreseeable that, in practice, issuers will sometimes be wrongly classified as trustworthy. Therefore, it is essential to offer functionality to revoke trust in an issuer and to publish this decision transparently in the registry.

- **NFR3: Issuer Revocation:** Trust in issuers must be revocable. Since the whole concept of Trusted Issuer Registries relies on the delegation of trust, Verifiers must be completely sure that the TIR's data is not corrupted. Thus, data integrity should be ensured by design.
- **NFR4: Data Integrity:** The integrity of the TIR's contents must be ensured using state-of-the-art methods.

Privacy and data protection is a major consideration for all respondents. They were of the same opinion that only already published information may be public in the TIR. Personal or sensitive data must not be published unencrypted. This requirement also aligns with the EU's GDPR, whose Article 17, "Right to erasure," indirectly prohibits storing personal data on immutable storage such as blockchains. In addition, B and E pointed out that the existence of an issuer's verification must remain confidential.

- **NFR5: Sensitive Data:** Sensitive issuer data must not be public.
- **NFR6: Verification Leak:** Issuer verifications must not leak the issuer's identity.

Performance

The respondents had different views on the expected number of their VC issuers, mainly because the growth of Gaia-X and its subprojects is not clearly foreseeable at this point. To have something to work with, we use as a point of reference the number of 1000 issuers, which is higher than most explicitly stated numbers. Trust anchors with a higher number of issuers can be expected to introduce additional levels of hierarchy using sub-registries.

- **NFR7: Scalability:** The architecture must scale well to at least 1000 issuers inside a single TIR.

All interviewees stated that Verifiable Credentials will be, among others, used for authentication at online services in their respective use cases. Therefore, for usability reasons, an issuer verification must take place almost without delay.

- **NFR8: Verification Latency:** Verifiers must be able to verify an issuer within 5 seconds.

Respondents B, D, and E indicated they expect low dynamics in the group of trusted issuers. After an issuer is verified as trusted and included in the registry, few updates are expected for the following months. Furthermore, there is no strict requirement for an immediate insertion of new issuers into the registry, as the inclusion is part of a more extensive verification process, which itself may take several days. Therefore, we impose a soft requirement that changes must be published within one day, which allows caching for higher availability.

- **NFR9: Write Latency:** Changes made to the TIR must be effective for all verifiers within 24 hours.

Verifiable Credentials are essential for all respondents' use cases and are planned to be involved in many parts of Gaia-X. Since a TIR itself is essential for securely and automatically verifying VCs, high availability is crucial. Ultimately, verification must be possible at any time. Fortunately, the low dynamics of the issuers allow for meaningful caching.

- **NFR10: Availability:** Issuers must be verifiable at all times. This may be achieved through caching (see NFR9).

Integration

Portability and interoperability are fundamental principles of SSI and Gaia-X and therefore also important for subcomponents such as TIRs. Some participants explic-

itly highlighted the importance of interoperable trust anchors (B, C, E) and portable architectures (B, E).

- **NFR11: Portability:** The TIR must be portable to different base technologies.
- **NFR12: Interoperability:** The TIR should utilize common technologies, standards, and interfaces to facilitate interoperability.
- **NFR13: Adaptability:** The TIR architecture must be flexible enough for use in different use cases.

5.2.3 Non-Requirements

From the interviews, we can also infer additional non-requirements, potential requirements that are not needed for the interviewees' use cases.

Decentralized Governance

Due to the decentralized nature of SSI, the idea of completely decentralized trust anchors arises. For example, a so-called DAO (Decentralized Autonomous Organization), often implemented as a smart contract, could make important decisions, such as adding a new issuer, with voting among a group of maintainers. Although some interviewees acknowledged the potential benefit of voting mechanisms for building trust, all respondents concurred that such a functionality is not needed for their respective use cases.

Transparency

Respondents B, D, and E emphasized the importance of transparency in establishing trust in a trust anchor. A TIR whose changes are publicly logged by design, such as by using a public blockchain, could facilitate transparency. However, since the respondents did not deem technological transparency at the TIR level essential, we did not establish it as a requirement.

5.2.4 Other Findings

All respondents either had no strong preference between a decentralized and a centralized storage solution or slightly preferred a decentralized solution. Instead, some referred to related requirements like availability, scalability, and security. Two persons mentioned the tradeoff of blockchain costs (B, D). D highlighted the issue of blockchain storage being public and suggested using a mixture of decentralized and

centralized storage. Additionally, A and B stated that they expected trust anchors to be trusted more if they seemed neutral, as they took on the role of a trustee. Finally, all respondents considered small transaction costs in cryptocurrencies, as incurred when using a low-cost blockchain like Tezos, acceptable.

5.3 Discussion

The expert interviews conducted for the Gaia-X project have provided valuable insights into the requirements for Trusted Issuer Registries within a larger use case of Self-Sovereign Identity. Our results can serve as a basis for developing new Trusted Issuer Registry designs, including for use cases outside Gaia-X.

Nevertheless, our study is limited in several aspects. First, the number of interview participants was limited to only five individuals. While these participants provided in-depth and knowledgeable perspectives, a larger and more diverse group of interviewees might have offered a broader range of insights and perspectives.

Furthermore, since all interviewees were from the GAIA-X 4 Future Mobility project family, our findings might not reflect the broader needs of the entire Gaia-X ecosystem. More inclusive research is needed for a comprehensive understanding of requirements for the whole Gaia-X.

Another limitation of our study is that the projects our interview participants are working on are still under development, and their requirements are not yet fully established. This means the insights and requirements for Trusted Issuer Registries we gathered are based on their current understanding and needs, which might change as their projects evolve. As a result, the conclusions we have drawn about what is needed for Trusted Issuer Registries might have to be adjusted in the future.

These limitations highlight the need for further research involving a broader and more diverse range of participants. Given that Gaia-X is still in development, our findings on Trusted Issuer Registries may need adjustments to reflect evolving requirements and a wider scope within the Gaia-X ecosystem.

6 Improved Trusted Issuer Registry Design

We used the findings of our previously discussed requirements analysis to design an improved decentralized Trusted Issuer Registry. The overall goal is to satisfy the requirements obtained from the interviews and avoid the disadvantages of existing solutions. To achieve portability and interoperability, we aim to design a protocol rather than a complete system. In Chapter 7, our prototype implementation will demonstrate a possible application of this protocol.

6.1 Design Overview

Our design leverages DIDs to identify Trusted Issuer Registries. The DID's service property allows linking to the actual service endpoint of the registry. We do not specify how and where the actual registry is stored and implemented. Instead, we leave the implementation-specific details to so-called TIR methods and only specify the data schema to which every TIR must be resolvable. TIRs can delegate trust not only to issuers but also to other TIRs, so-called sub-registries. Delegated trust can be limited in two ways: firstly, it is possible to define for each issuer, with varying degrees of specificity, which credential schemas they may use. Secondly, context information can also be stored in natural language, which supports the manual verification of VCs needed in case the automated credential schema validation is insufficient. In the following, we will explain the design in more detail.

6.2 TIR Referencation & Discovery

In our design, TIRs can be referenced with the DID service property. For this, a DID controller can add a service to his DID Document as shown in Figure 6.1.

As defined by the DID specification, a service has three required properties. In our case, they must be filled as follows:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    ...
  ]
  "id": "did:example:123456789abcdefghi",
  "verificationMethod": [...],
  "authentication": [...],
  "service": [{
      "id": "did:example:123456789abcdefghi#tir",
      "type": "TrustedIssuerRegistry2023Web",
      "serviceEndpoint": "https://example.com/tir.json"
  }]
}
```

Figure 6.1: Example of a DID Document referencing a TIR

id A DID fragment URL string referencing this exact service property in its DID Document. This is useful for directly referencing a TIR in case the DID Document contains multiple TIR services.

type A TIR method identifier string that always starts with "TrustedIssuerRegistry2023". TIR methods are explained further below.

serviceEndpoint A URI string or a set of URI strings linking to the endpoint at which the TIR can be found and handled method-specifically. In the case of a set of URIs referencing multiple endpoints, all endpoints are to be treated equally, and resolvers are urged to choose a random endpoint. This feature makes it possible to achieve higher scalability and accessibility through redundancy and client-side load-balancing (NFR7, NFR10).

The URI scheme is restrictive and therefore not an optimal choice, but unfortunately, it is imposed by the DID specification. In cases where the TIR method-specific endpoint cannot be accessed via established URI schemes, we expect fictitious unregistered URI schemes to be used. This should not lead to misinterpretations since the service type defines a clear context for the endpoint. Nevertheless, we highly recommend the use of established URI schemes whenever possible.

We decided on this sort of TIR referencing because it decouples the referencing

and implementation by using the established DID standard. This design decision thus creates portability (NFR11) and interoperability (NFR12) between the different implementations and overall makes the TIR decentralized. Additionally, the specification allows the discovery of Trusted Issuer Registries as subjects can publish their own TIRs in their public DID Document. Furthermore, the decisions for portability and optional redundancy give TIR maintainers the possibility to achieve near-full availability (NFR10). Finally, portability also facilitates scalability (NFR7) as maintainers can purposely select base technologies that scale well to their use case.

6.3 TIR Methods

TIR methods are specifications defining how certain types of TIRs can be resolved from a service endpoint URI to the TIR data model we specified. They are needed since our design does not specify how exactly TIRs are implemented. The term ’TIR method’ is derived from the terminology of the DID specification, as both TIR and DID methods aim to create portability while maintaining interoperability.

Every TIR method has a Pascal case identifier starting with "TrustedIssuerRegistry2023" followed by a method-specific identifier like "Web." In this example, the full identifier would be "TrustedIssuerRegistry2023Web". We will define several example methods further below.

6.4 Data Model

We defined a general data model for our TIR design. An overview of the data model in Javascript syntax is shown in Figure 6.2.

TIR Model

A TIR consists of several metadata properties and one map of issuers. All properties except "extraMetadata" are required.

- **method:** A string denoting the TIR method identifier, e.g., "TrustedIssuerRegistry2023Web".
- **methodProtocol:** A number denoting the protocol version of the TIR method specification. This field allows TIR methods to evolve naturally without creating new TIR method identifiers after a specification change.

```
interface TIR {
  method: string                // TIR method identifier
  methodProtocol: number        // TIR method protocol version
  protocol: number              // TIR protocol version (currently: 1)
  issuer: string                // DID of the TIR's maintainer
  lastUpdated: string           // ISO 8601 timestamp
  ttl: number                   // Cache time to live (seconds)
  extraMetadata: null | {
    [key: string]: string
  }
  issuers: {                    // Issuers map: id -> issuer
    [key: string]: Issuer
  }
}

interface Issuer {
  id: string                    // URI
  trustedSince: string | null   // ISO 8601 timestamp
  trustedUntil: string | null   // ISO 8601 timestamp
  revoked: boolean | null

  // Trust Context: supports manual verification by humans
  tcDescription: string | null
  tcIdentity: string | null

  // Credential Schemas: enables automatic verification
  credentialSchemas: CredentialSchema[] | null
}

interface CredentialSchema {
  id: string                    // URI linking to the schema
  type: string
  hash: string                  // SHA-256 hash of the schema
  inheritable: boolean
}
```

Figure 6.2: Our Trusted Issuer Registry's Data Model

- **protocol:** A number defining the protocol version of the hereby specified Trusted Issuer Registry design. For the version described here, we specify the value 1.
- **issuer:** The DID of the TIR's maintainer. This backlink enables serializing the TIR into a Verifiable Credential and checking whether a TIR reference in a DID Document was made by its maintainers or someone else.
- **lastUpdated:** An ISO 8601 timestamp denoting the last time the TIR was updated. This property allows for smarter caching and, thus, faster resolution of the TIR (NFR10). In addition, this property becomes the issuanceDate property of the TIR credential.
- **ttl (time to live):** A number of seconds denoting the time how long the TIR may be cached. This allows TIR maintainers to define a simple caching policy, like a cache expiration of 24 hours as required by NFR9.
- **extraMetadata:** An optional object containing extra metadata string-string key-value pairs. This makes our design more flexible and allows it to be used for purposes not foreseen by us (NFR13).
- **issuers:** An object mapping issuer IDs to Issuer objects. The property contains all issuer information of the TIR.

Issuer Model

Issuers only have one required field:

- **id:** A string ID in URI format. This is the minimal information needed to check whether an issuer is trustworthy. In accordance with NFR5, TIR maintainers can decide how much extra information they want to store about issuers. In fact, the id property is redundant, since the id can be inferred from the "issuers" map key. Nevertheless, we have defined it to store complete issuer objects. Of course, TIR method implementers can decide whether they actually want to store the ID redundantly or not, and can thus save storage space.

In addition, we define several optional fields that provide a broader context of how the issuer is trusted:

- **trustedSince:** An ISO 8601 timestamp string defining from what on the issuer was classified as trusted.

- **trustedUntil:** An ISO 8601 timestamp string that defines until when the issuer is classified as trustworthy. We expect this to be used as an expiration date that requires issuers to be verified again by the registry maintainers.
- **revoked:** A boolean value indicating whether the trust in the issuer has been revoked. This field enables revoking trust in an issuer while keeping the registry entry for transparency reasons.
- **tcDescription (trustContextDescription):** A string field that maintainers can use to define the trust in the issuer in natural language. The value enables maintainers to decide for themselves how to balance the tradeoff between detailed trust specification and broad delegation of trust.
- **tcIdentity (trustContextIdentity):** A string field that maintainers can use to connect an issuer with his legal identity. We have refrained from specifying the field in detail, as the requirements vary depending on the application. Maintainers can fill the field with natural language or structured data like a serialized JSON map.
- **credentialSchemas:** A list of credential schema objects. This field enables maintainers to define in detail the Verifiable Credentials the issuer may issue. If the property is undefined, the feature is not used, and complete trust is delegated for all possible Verifiable Credentials and potential sub-registries.

CredentialSchema Model

A CredentialSchema consists of four required properties:

- **id:** A URI string referring to an external resource containing the actual schema.
- **type:** A string defining the schema's type. As Verifiable Credentials are often presented in a JSON format, we expect "JsonSchema" to be a common value here, referring to the JSON Schema specification [16]. Other schema types could be, for example, "SHACL" [56]. The purpose of the loose specification is to enable a wide variety of schema forms that are not yet foreseeable. For instance, the increasingly frequently used JWT VCs cannot be validated directly with JSON schema. The field allows maintainers to decide which schema type is most appropriate for their use case.
- **hash:** A SHA-256 hash of the schema, ensuring the external resource's integrity in accordance with NFR4. We opted for the popular SHA-256 hash algorithm

because it is widely used and regarded as secure, thus making it a reasonable choice for securing the schema's data integrity [57].

- **inheritable:** A boolean value that defines whether the trust defined in the schema can be delegated. This value is important in the event that the issuer operates its own TIR, a sub-registry, under its DID. If the schema is inheritable, the TIR also trusts the issuers in the issuer's sub-registry within the set schema restriction.

Verifiable Credential Serialization

As described above, our high-level design does not contain any specifications for concrete implementations of individual TIR methods. The only requirement is that the TIR must be resolvable into our previously defined data model. This data model can be serialized into a simple verifiable credential like the following example. A credential proof may also be included, which makes especially sense when using intransparent storage solutions like web servers. We have not specified this as a requirement, as the functions of some storage solutions, such as blockchains or IPFS, already fulfill the desired security levels depending on the use case. The decision to insert a proof can therefore be made by the implementers of individual TIR methods themselves. However, it is important to note that a verifiable credential without proof loses its most important property, verifiability. For the reasons just mentioned and to enable interoperability, we specify the VC Serialization for all TIR methods, regardless of the support for proofs.

When serializing a TIR to a VC, some credential properties get filled with certain TIR values.

- **issuer:** the TIR's issuer property
- **issuanceDate:** the TIR's lastUpdated property
- **expirationDate:** the TIR's lastUpdated timestamp extended by the number of time-to-live (ttl) seconds.
- **credentialSubject:** the complete TIR as a JSON object
- **proof:** might be filled if the TIR method supports VC proofs

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "id": "http://example.edu/credentials/1872",
  "type": ["VerifiableCredential"],
  "issuer": "did:example:1234567890",
  "issuanceDate": "2023-11-23T11:23:24Z",
  "expirationDate": "2023-11-24T11:23:24Z",
  "credentialSubject": {
    "method": "TrustedIssuerRegistry2023Web",
    "methodProtocol": 1,
    "protocol": 1,
    "issuer": "did:example:1234567890",
    "lastUpdated": "2023-11-23T11:23:24Z",
    "ttl": 86400,
    "extraMetadata": {"key": "value"}
    "issuers": {
      "did:web:example.com": {
        "trustedSince": "2023-11-23T11:23:24Z",
        "trustedUntil": "2024-11-23T11:23:24Z",
        "revoked": false,
        "tcDescription": "An example issuer only used for example
           purposes.",
        "tcIdentity": "John Doe Company, Example Street, 12345,
           Example-City",
        "credentialSchemas": [{
          "id": "https://example.com/testschema.json",
          "type": "JsonSchema2020",
          "hash": "d14a028c2a3a2bc...8ea62ac5b3e42f", // SHA-256 hash
          "inheritable": true
        }]
      }
    }
  },
  "proof": {...}
}
```

Figure 6.3: Example VC representation of our TIR design

6.5 Example TIR Methods

To further demonstrate our TIR design, we specified basic TIR methods that use different technologies and functionalities.

Web

The first TIR method we specify is a simple implementation based on HTTP with the identifier "TrustedIssuerRegistry2023Web". This method simply hosts the TIR's JSON representation on a web server. Changes to the TIR are simply done by updating the hosted JSON file. It is important to configure the server's Cross-Origin Resource Sharing (CORS) settings accordingly so that every host may read the TIR.

The example DID Document in Figure 6.1 shows how the Web method is referenced. The credentialSubject's value of the VC in Figure 6.3 shows the exact data being stored with the Web method.

WebVC

The WebVC method is similar to the Web method and directly stores a verifiable credential on a web server. Its identifier is "TrustedIssuerRegistry2023WebVC". The data hosted would exactly be the contents Figure 6.3 with a modified method identifier. As with the Web method, it is important to configure the server's CORS settings correctly. The main advantage of the WebVC method is the use of a credential proof, enabling the secure sharing of the TIR.

IPFS & IPFSVC

The concepts of the Web and WebVC methods could be easily transferred to the decentralized IPFS. The data being stored would be the same, with the only difference being the new method identifier and the DID service's service endpoint pointing to a URL with the "ipfs" scheme.

Tezos

We also specified a TIR method with decentralized storage using smart contracts of the Tezos blockchain. Unfortunately, there does not exist a URI scheme for Tezos. Therefore, we decided to leverage the "tz" DID method's scheme [58] in order to reference the Tezos network and the smart contract address. The smart contract's storage represents the TIR's data model, thus resolving a Tezos TIR is simply fetching

and parsing the contract's storage. The exact implementation of the smart contract including its governance features will be described in Chapter7.

6.6 Discussion

In the following, we will discuss our design and compare it to the requirements defined in Chapter 5 and the design decisions of existing solutions analyzed in Chapter 4.

6.6.1 Requirements

To discuss compliance with the requirements, we will go through them step by step and explain how our design meets them.

- **FR1: Register Trusted Issuer:** The TIR must enable registering new trusted issuers.
- **FR2: Update Trusted Issuer:** The TIR must enable updating existing trusted issuers. This includes updating certain attributes and deleting issuers.

Since our design is very high-level, the functional requirements FR1 and FR2 fall into the scope of specific TIR method implementations. Our example TIR methods, Web and Tezos, both support registering and updating the TIR.

- **FR3: Endorse Sub-Registries:** The TIR must allow delegating trust to sub-registries and thus create an additional hierarchy level.

Our design supports sub-registries using the inheritable property of CredentialSchemas. If an issuer has an inheritable schema and a DID that references a TIR, this TIR is considered a sub-registry authorized for the corresponding schema and its subsets.

- **FR4: Verify Issuer's Trustworthiness:** The TIR must allow verification of an issuer's trustworthiness.

This functional requirement depends solely on the implementation of a TIR verification service. By our design, every TIR can be resolved and then checked for issuer inclusion.

- **NFR1: Issuer Authorization**: The TIR must allow specifying specific issuer qualifications.

The issuer's credentialSchemas property allows fine-grained issuer authorization using credential schemas. The type property enables the use of all sorts of schemas possible.

- **NFR2: Limit Sub-Registry Trust Delegation:** The TIR should allow limiting the delegated trust for endorsed TIRs.

The issuer authorization with credential schemas is also transferable to sub-registries by not differentiating issuers and sub-registries. The inheritable property of CredentialSchemas defines whether a schema is also usable for sub-registries.

- **NFR3: Issuer Revocation:** Trust in issuers must be revocable.

Trust in issuers can be revoked using the issuer's revoked property.

- **NFR4: Data Integrity:** The integrity of the TIR's contents must be ensured using state-of-the-art methods.

In general, integrity protection falls into the scope of TIR methods. Depending on the TIR method, an already resolved TIR can be integrity protected using the Verifiable Credential serialization with a proof. External resources, more precisely credential schemes, are protected by a SHA-2 hash.

- **NFR5: Sensitive Data:** Sensitive issuer data must not be public.

Our design does not enforce the publication of potentially sensitive data, like an issuer's legal identity. The only required issuer property is its ID, which is not sensitive in the Gaia-X use case.

- **NFR6: Verification Leak:** Issuer verifications must not leak the issuer's identity.

In our design, verifiers resolve a complete TIR and locally search it for issuer inclusion. Therefore, the entities serving the TIR cannot know which issuer is about to be verified. Using a proxy that automatically resolves a TIR at specific times, the entities serving the TIR cannot even know in general that some issuer is being verified. Nevertheless, this design does not make issuer verifications fully anonymous, but only in a similar way as k-anonymity [59] with k being the TIR's number of issuers: The entities serving the TIR know from detecting the resolution that a verifier is generally interested in the TIR, or in case no proxy is used, that a verifier is verifying some issuer in the TIR. Therefore, they cannot decide which of its k issuers is being verified. However, the smaller k is, the more informative a detected verification is. Malicious TIR maintainers could, for example, create TIRs

with only one issuer and thus know when a verification is happening. However, such an attack is unrealistic because there is no reason a verifier would delegate trust to a TIR with only one issuer, having no added value compared to simpler verification methods. In principle, verifiers who value the anonymity of verifications should only use TIRs with a minimum number of issuers. However, even in this case, malicious maintainers could intentionally add fake issuers that they know will never issue VCs. This example highlights that verifiers must only use TIRs that they fully trust not to be malicious.

- **NFR7: Scalability:** The architecture must scale well to at least 1000 issuers inside a single TIR.

Scalability is hard to measure and depends on many factors. A detailed analysis would be a paper of its own and is, therefore, out of scope. To get a rough indication, we can conservatively extrapolate the storage space of a TIR. An average issuer with all properties and three credential schemas in JSON representation has about 1 kilobyte of data. A TIR with 1000 issuers would, therefore, be around 1 Megabyte. This is a significant amount of data but does not reach the technological limits of either the Web or the Tezos method.

- **NFR8: Verification Latency:** Verifiers must be able to verify an issuer within 5 seconds.

In general, this requirement depends on the implementation of verification software. Caching and using, e.g., hash maps, allow the verification of an issuer in far less than five seconds.

- **NFR9: Write Latency:** Changes made to the TIR must be effective for all verifiers within 24 hours.

With the "ttl" property of the TIR, maintainers can freely define a caching policy, like a cache expiration of 24 hours.

- **NFR10: Availability:** Issuers must be verifiable at all times. This may be achieved through caching (see NFR9).

Availability always depends on the base technology and, thus, on the TIR methods used. In general, the caching with the ttl property and the potential redundancy of TIR service endpoints in the DID service property allow TIR maintainers to achieve near-full availability.

- **NFR11: Portability:** The TIR must be portable to different base technologies.

- **NFR12: Interoperability:** The TIR should utilize common technologies, standards, and interfaces to facilitate interoperability.

Portability and interoperability are achieved through the concept of TIR methods, allowing all conceivable base technologies while preserving interoperability between them.

- **NFR13: Adaptability:** The TIR architecture must be flexible enough for use in different use cases.

Our design does not make specific assumptions on its use case and can be used general-purposely in different applications. The TIR's extraMetata property allows for adding additional information, most of the extensive issuer data model's properties are optional, and features like caching, issuer identification, or authorization may or may not be used. The freely configurable TIR methods and CredentialSchema types allow maintainers to choose for themselves which technologies suit their use case best.

6.6.2 Design Decisions

In Chapter 4, we already discussed the design decisions of existing solutions. In the following, we will discuss those individually for our design. Table 6.1 shows how our design compares to the existing TIRs we analyzed in Chapter 4.

	X.509 PKI	EBSI	TRAIN	DCC	OCI	ToIP	Our Design
TIR concept		✓	✓	✓	✓	✓	✓
Chaining concept	✓	✓					
General-purpose	✓		✓		✓	✓	✓
Storage type	**decentral.**	**decentral.**	**combinat.**	**central.**	**decentral.**	n/a	**flexible**
Stored in one place	n/a	✓		✓	✓	n/a	
Issuer identification	✓		✓	✓		n/a	✓
Issuer authorization		✓	✓		✓	✓	✓
Hierarchy (sub-registries)	✓	✓	✓			n/a	✓
Subregistry authorization	✓	✓		n/a	n/a	n/a	✓
Caching intended		✓					✓
Fully integrity-protected	✓	✓		✓			✓

Table 6.1: Comparison of our design and the analyzed TIRs, based on Table 4.1

Storage Our analysis showed that centralized storage, like web server hosting, and decentralized storage solutions, like IPFS and blockchains, both have advantages and disadvantages. The interviewees also had no clear preference regarding centralized or decentralized forms of storage. For this reason, we have decided not to specify the

type of storage in our architecture. Instead, we rely on the previously introduced TIR methods backed by DID services, which enables us to achieve extensive portability, interoperability, and adaptability in accordance with NFR11, NFR12, and NFR13.

Functionality Following NFR13, we aimed to develop a general-purpose design. Because of that, our design enables issuer identification, issuer authorization, and sub-registry authorization. We only did not include the verification of verifiers, as done by the ToIP TIR, since it is out of scope of a thesis about Trusted Issuer Registries. Nevertheless, at first glance, our design should be applicable to verifiers as well, with a few adjustments.

Scalability For better scalability, our design enables hierarchical trust delegation with sub-registry authorization. Further aspects concerning scalability depend on the utilized TIR methods.

Performance Our design enables caching to improve performance. Like TRAIN, our design requires multiple resolution steps when using hierarchy. This is a disadvantage compared to TIRs that store all data in one place. However, caching may suffice to counter this issue. In cases where caching is not an option and hierarchy is needed, our architecture is inferior to those that store everything in one place, like EBSI's TIR.

Security In our analysis of existing TIRs, we identified the recurring security issues of data integrity and limiting delegated trust. We avoided these issues with the hashing of external resources and fine-grained authorization with credential schemas for both issuers and sub-registries. Another issue we identified was high centralization. Our concept of TIR methods enables the use of completely decentralized TIR implementations. Additionally, the interoperability created by referencing TIRs with DIDs facilitates decentralization. Therefore, TIR maintainers can decide for themselves how the value the tradeoff between centralization and decentralization.

6.6.3 Limitations

Overall, our TIR addresses several disadvantages of existing TIRs and combines multiple of their advantages and functionalities in one design. Nevertheless, our design also has some limitations, which we mostly discussed previously. Our requirements are mostly met or depend on TIR methods. One issue is that the verifications are only partially anonymous to the TIR service operators. We discussed how this issue can be

partly mitigated using certain verifier policies. In general, it is a fundamental issue of Trusted Issuer Registries and might be solved best by solutions like certificate chains that do not require third parties for issuer verification. Another shortcoming of our design is that resolving a complete TIR with hierarchy requires multiple resolution steps, as the complete TIR is not stored in one place by design. The problem stems from our decision to prioritize interoperability over performance. Both issues, anonymous verifications in TIRs and the performance-interoperability tradeoff are a potential topic for future research.

Another limitation is the use of unregistered names for credential schema types and TIR method identifiers, e.g., "JsonSchema" or "TrustedIssuerRegistry2023Web", thus creating uncertainty for implementers. This is an issue that also DID methods suffer from. The DID Specification Registries [8], specified in a W3C Group Note document, were created as a solution. Its concept of a single source of truth could also be applied to our Trusted Issuer Registry. However, this would undermine the efforts for decentralization. For now, we urge implementers to choose names that are expressive and clearly unique. Nevertheless, the issue remains and poses a topic for further improvement.

7 Prototype Implementation

We implemented an extensive prototype consisting of multiple parts to demonstrate our design. The first component is a Tezos smart contract that allows operating a Trusted Issuer Registry on the Tezos blockchain. Secondly, we built a backend application that can automatically resolve TIRs by their DID and cache the result to efficiently serve issuer verification API requests. Finally, we implemented a web frontend that allows managing TIRs and verifying issuers by utilizing the backend API or performing the resolution and verification standalone. Furthermore, we extracted the code shared by backend and frontend into a separate core package and specified the API in an OpenAPI schema [60]. The complete code is available open source and structured as a monorepository [61].

7.1 General Considerations

Programming Language

Building software consisting of multiple interrelated components requires a wise choice of technologies, starting with the programming language. We decided on Typescript because of several reasons. Javascript is widely used as a programming language on the Internet and therefore offers many packages such as JSON Schema validators. In addition, JsLigo is an easy-to-learn programming language for Tezos smart contracts in the syntax style of Javascript. Typescript extends Javascript with strict typing and thus improves code quality and developer experience. For this reason, we have chosen Typescript for the backend and frontend. Another advantage of this is that the shared programming language facilitates functionality reuse. For the smart contract, we opted for JsLIGO for the above reasons.

Deployment and Reusability

The prototype was implemented with the intention of making the work easily reusable and deployable. To this end, we utilized Docker Compose for containerizing both the frontend and backend components, simplifying the process of setting up the entire application. This approach ensures a seamless deployment experience, as Docker

Compose orchestrates the configuration and interconnection of multiple containers. Alongside this, we have also specified the backend API and shared data models in an OpenAPI 3.1 [60] document. The schema enables automatic code generation, further facilitating the development process.

7.2 Tezos Smart Contract

The Tezos smart contract can be regarded as the backend of a Trusted Issuer Registry operating on the Tezos blockchain. It is implemented using LIGO[1], a programming language that compiles to Michelson, Tezos's native smart contract language. More specifically, we used JsLIGO, one of LIGO's two syntaxes that is inspired by the Javascript syntax.

7.2.1 Endpoints

Our contract provides five endpoints:

- **setMetadata(ttl: nat, extraMetadata: string):** sets the contract's manually editable metadata properties
- **addIssuer(issuer: Issuer):** inserts a new issuer into the registry
- **updateIssuer(issuer: Issuer):** updates an existing issuer in the registry
- **deleteIssuer(id: string):** removes an issuer from the registry
- **transferOwnership(newOwner: address):** transfers the contracts ownership to a new Tezos account

The "transferOwnership" endpoint is important because it enables transferring ownership to a new secure account in case, e.g., the owner's account is compromised.

Contract Events

The Tezos blockchain provides a way for emitting event-like information that can be detected by off-chain applications using Michelson's EMIT instruction[2]. Although the feature could be useful in further developments of a TIR built upon Tezos, it is currently not needed. In our current implementation, every change to the contract is important and should thus emit an event. In addition, we only have the loose

[1] https://ligolang.org/
[2] Tezos Technical Documentation - Contract Events, `https://tezos.gitlab.io/nairobi/event.html`

requirement that changes to the TIR must be received by the verifiers within 24 hours and not immediately (NFR9). It is therefore currently sufficient to simply listen for changes in the contract and check the lastUpdated field.

7.2.2 Data Model

Our contract's data model is shown in Figure 7.1. It introduces three additional properties to the TIR type that are Tezos-specific:

- **protocolTezos:** A constant natural number defining the protocol version of the Tezos TIR method. The property enables protocol participants to detect future protocol changes. We specify the number 1 for the protocol version described here.
- **owner:** A Tezos address identifying the account that is allowed to make changes to the contract.
- **issuerIds:** A set of strings holding the IDs of the stored trusted issuers. This property is required to keep track of the stored issuers, as the Michelson Big-map type used for the "issuers" property does not store the keys.

The other properties of the TIR type are directly transferred into Michelson types, as seen in Figure 7.1. As the data type for the "issuers" property, we chose Big-map. The reason for this is that Big-maps are not completely serialized on contract interactions, in contrast to normal maps. If a map were used, each individual interaction with the smart contract would cost more and more gas as the number of issuers increased. This is avoided by using a Big-map[3] [20]. For simplicity, the "extraMetadata" field that can take any JSON object is stored as a serialized string.

[3] https://docs.tezos.com/smart-contracts/data-types/complex-data-types#big-maps

```
export type credentialSchema = {
    id: string,
    @type: string,
    hash: string,
    inheritable: bool,
}
export type issuer = {
    id: string,
    trustedSince: option<timestamp>,
    trustedUntil: option<timestamp>,
    revoked: option<bool>,
    tcDescription: option<string>,
    tcIdentity: option<string>,
    credentialSchemas: option<list<credentialSchema>>
};
export type storage = {
    protocolTezos: nat,
    owner: address,
    issuerIds: set<string>,

    protocol: nat,
    issuer: string,
    lastUpdated: timestamp,
    ttl: nat,
    extraMetadata: string,
    issuers: big_map<string, issuer>,
}
```

Figure 7.1: The Tezos TIR data model in JsLigo code

7.3 Core Package

The core package contains the frontend's and backend's shared logic for resolving TIRs and verifying issuers given a resolved TIR. The code is bundled into a separate Typescript package for easier reuse, allowing for simpler containerization with Docker.

7.3.1 TIR Resolution

The first basic functionality is resolving single TIRs. As explained in the design chapter, this task consists of two steps: resolving the TIR service in the TIR's DID document and then retrieving the registry method-specific from the service endpoint. For the DID resolution, we utilize the Typescript package did-resolver[4] which can be extended to support all kinds of DID methods. In our current implementation, only the "web" method is supported, but more methods can be added easily. After obtaining the DID Document, its services are searched for known TIR methods. Our prototype currently supports the "Web" and "Tezos" TIR methods, and it is simple to add new ones. From there on, TIR method-specific code retrieves the TIR from the service endpoint and maps it into the common data model. As explained in the design chapter, the Web method just needs to fetch the TIR'S JSON file and can directly return the value.

The Tezos method is a bit more complicated. First, the smart contracts storage is fetched using the Taquito library[5]. Then, the storage is parsed into our data model. Since the TIR's issuers are stored in a Big-map that cannot be fully fetched at once, all issuers must be fetched individually. As this can take a relatively long time, the issuers are only loaded all at once if requested. It is also possible to lazy-load issuers later on demand. This functionality is used in our web frontend.

To resolve a complete TIR including potential layers of hierarchy, we implemented a Breadth-first search algorithm that explores the TIR tree layer by layer starting, from its root. First, the root TIR is resolved as described before. Then, for every issuer that has a DID ID and either no credential schema or at least one inheritable credential schema, its DID will be resolved and checked for a TIR. In this way, layer by layer, the complete TIR will be resolved. The resolution's output is stored via an abstract map interface. The map maps issuer IDs to their corresponding issuer object, their trust path of parent DIDs, and if existent, their TIR. The abstract interface allows the use of external databases like Redis which may be superior to simple in-memory objects when using large TIRs, depending on the context.

[4] did-resolver, version 4.1.0, https://www.npmjs.com/package/did-resolver/v/4.1.0
[5] https://tezostaquito.io/

Although our TIR resolution algorithms are sophisticated and work well, they have their limitations and optimization potential. For example, the "lastUpdate" attribute of the TIRs is currently ignored, which could be used for smart caching and thus faster resolution. In addition, there is currently no proper treatment of the special case where an issuer is included in the TIR more than once. Currently, only the first appearance of an issuer is stored, and duplicates are ignored. However, this is problematic as the duplicates could have different metadata, authorizations, and sub-registries. This problem should be addressed before use in production, yet the current handling is fine for our prototype.

7.3.2 Issuer Verification

The other important functionality of the Core package is verifying an issuer's trustworthiness given an already resolved TIR (FR4). The resolved TIR is passed using the same abstract map interface used for resolution. Besides the map interface, the other verification inputs are the issuer's ID, a verification date, and a verifiable credential JSON object. All these parameters are optional, but combined, an ID or a verifiable credential must be present.

The verification algorithm works as follows. First, the issuer to be verified is fetched from the map. If not found, the issuer is not trusted. Then, if found, all issuers on the trust path are verified: their revocation status is checked, their "trustedSince" and "trustedUntil" timestamps are compared with the issuance date, and their schemas are searched for a match with the passed VC. Of course, dates and schemas are only checked if the needed information is specified for the respective issuer. The partial results of the trust path issuers and the overall result are then returned, including information on the conducted verifications and their explanations. An example of the detailed verification result can be seen later on in the Frontend section.

Even though our algorithm performs a detailed verification of the issuer, it has some limitations. First of all, our algorithm does not verify the credential's proof and JSON-LD contexts as this is out of scope. In principle, however, such functionality could still be easily integrated. Additionally, only Draft 07 JSON Schemas are currently supported because of the jsonschema[6] package used. Nonetheless, new schema types can be added easily.

[6] https://www.npmjs.com/package/jsonschema

7.4 Backend

Our backend is built upon the common NodeJs web framework Express[7] and a Redis[8] cache. Its primary purpose is to decouple the time-consuming TIR resolution from the time-critical issuer verification. The backend has a CRON job that resolves a predefined TIR at freely adjustable times, for example, midnight. Thus, the costly resolution can be performed automatically during low-load periods.

The backend provides four endpoints as specified in the OpenAPI schema. They are essentially a facade for the core package and have no additional implementations besides caching the resolved TIRs using Redis, thus enabling fast verifications.

- **POST /setAutoResolveDID:** sets the DID that is resolved by the backend when its CRON job is triggered
- **GET /resolveSingleTir:** resolves a single TIR by its id without potential sub-registries and returns the result
- **POST /resolveTIR:** resolves a complete TIR by its DID including its sub-registries and caches the result in the Redis database
- **POST /verify:** verifies an issuer given its id, an issuance date, and a VC

Although the backend implementation successfully demonstrates the resolution-verification decoupling and works hand in hand with the frontend, it needs some improvements before being used in production. First, no request authorization is in place, and every request can, e.g., set the auto-resolve DID. Additionally, the backend's cache currently ignores the TIR's time-to-live property and could thus cache a TIR for too long. Finally, since the backend is based on the core package, it also has its limitations, such as ignoring the "lastUpdated" field when resolving the TIR.

7.5 Frontend

Our frontend is a web app implemented using the Vue Javascript framework[9]. Its primary purpose is to be a user interface for TIR maintainers and verifiers. Therefore, the app is split up given its two main functionalities: managing TIRs and verifying issuers given a trusted TIR. Before the application can be used, the trust anchor

[7] https://expressjs.com/
[8] https://redis.io/
[9] https://vuejs.org/

TIR's DID has to be entered into the navigation bar's DID input. The DID will be stored in the browser's Local Storage and is restored every time the user returns to the page.

7.5.1 TIR Management

The frontend's TIR management page uses the core package to resolve the specified TIR and display its contents in a human-readable way. Figure 7.2 shows a screenshot of the user interface after resolving an example TIR from Tezos. The screenshot also shows the expand functionality explained previously in the core package section. Since the Tezos Big-map storing the issuers cannot be fully fetched at once, the user can click the expand button to lazy-load an issuer's data.

For Tezos TIRs, the maintainers can connect their Tezos wallet with the easy-to-use Beacon SDK[10] and make changes to the TIR using a handy user interface. New issuers can be registered, existing issuers can be updated or deleted, and the TIR's metadata can be changed. The user interface for registering new issuers is shown in Figure 7.3.

Our frontend provides several usability features, like the simple lazy-loading of issuers or an automatic hash generation for credential schemas. Nevertheless, further features like date pickers or form validation could be added as future improvements.

7.5.2 Issuer Verification

For demonstration reasons, the issuer verification can be done in two ways: completely standalone by the frontend using the core package directly or with an API request to the backend. The user interface is the same. Verifiers can input an issuer's ID and optionally a date and a JSON Verifiable Credential. The system will then verify the issuer and visualize the trust path, including the subordinate verifications. The more information provided about the issuer or verifiable credential, the more detailed the verification result will be. It is also possible to only enter one verifiable credential. In this case, the issuer ID and the issue date are automatically taken from the credential. Once the verification has been completed, the result is visualized as a trust path. The necessary information is displayed for each link in the chain of trust in order to check the individual partial verifications carried out. If partial verifications have failed, the corresponding trust context is also displayed, which supports the verifier in making a manual decision. Figure 7.4 shows an issuer's verification using the backend that failed due to an invalid issuance date. Nevertheless, other verifications regarding, for example, revocation or credential schemas were

[10] https://www.walletbeacon.io/

successful. In cases where a grey question mark is shown, the issuer data did not include enough information for the specific verification.

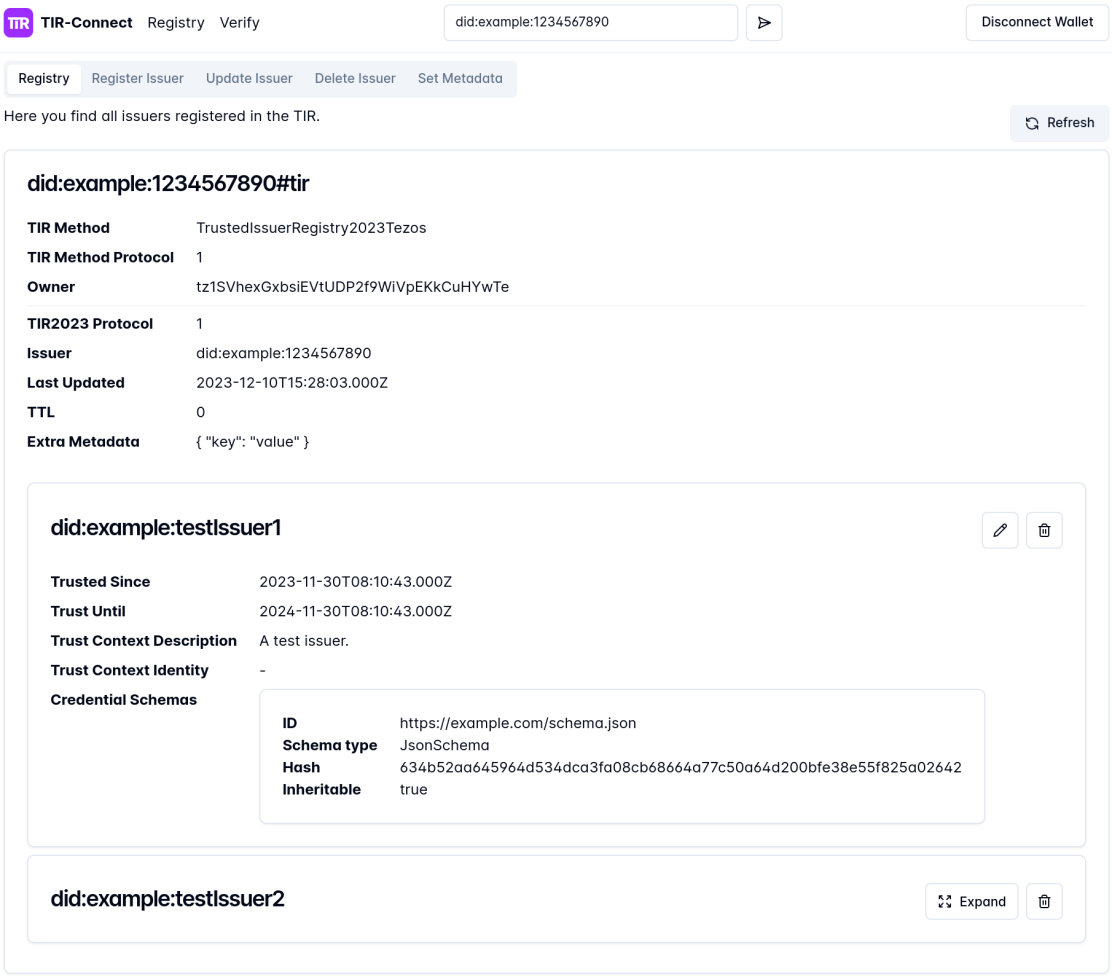

Figure 7.2: Screenshot of the web app displaying the contents of an example TIR

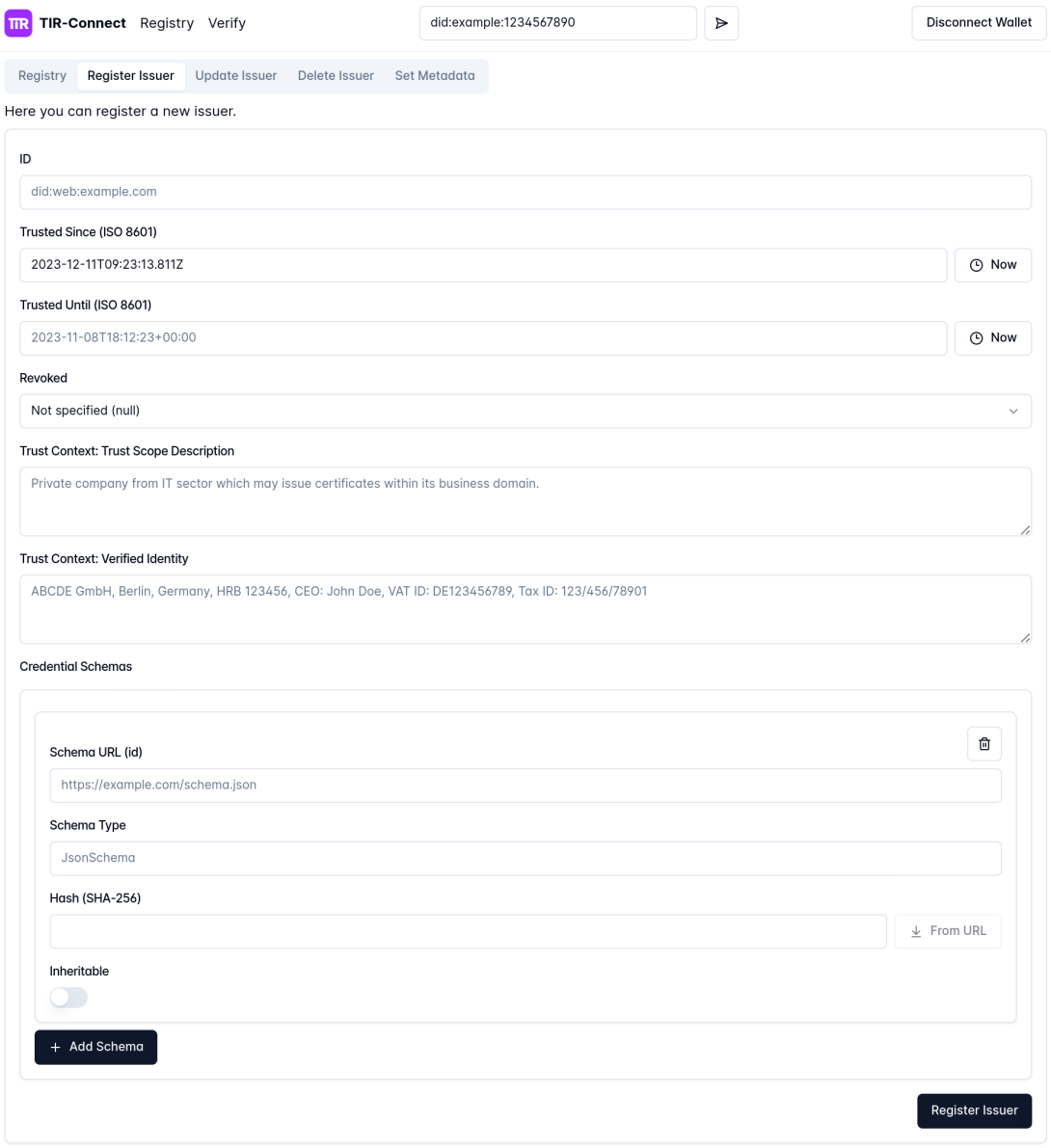

Figure 7.3: Screenshot of the web app's user interface for registering a new issuer to a Tezos TIR

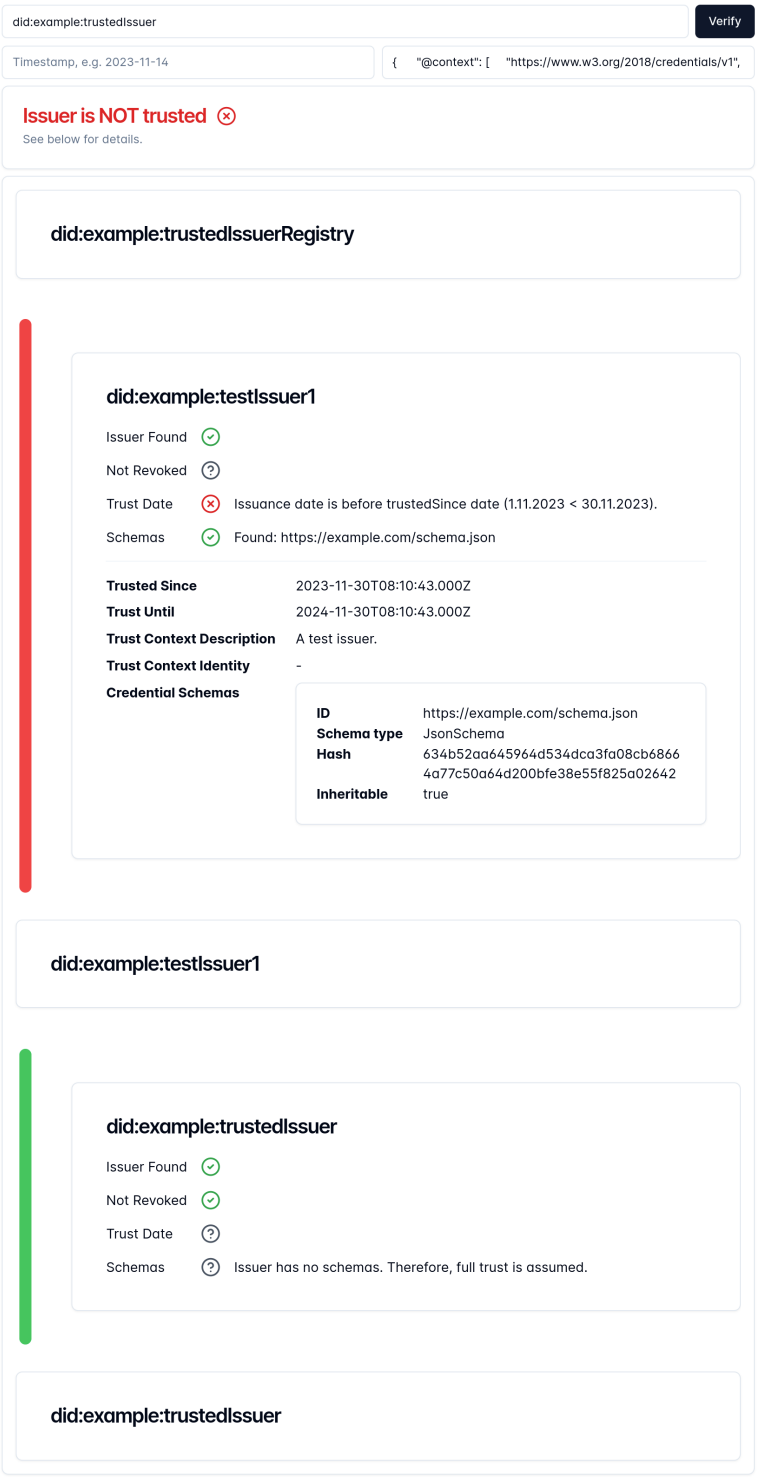

Figure 7.4: Screenshot of the web app's user interface for issuer verification

8 Conclusion

The primary goal of this thesis is to propose a new decentralized Trusted Issuer Registry design. To that end, we analyzed existing TIR designs, gathered requirements from an actual use case, inferred an improved TIR design, and demonstrated it using a comprehensive implementation. As explained in the structure section of Chapter 1, our methodology closely follows the research questions. The following paragraphs summarize our findings related to each methodology step and research question.

The analysis of existing TIR designs related to our first research question identified multiple architectural advantages and disadvantages regarding their storage type, functionality, scalability, security, and complexity. The evaluated solutions employed both centralized and decentralized storage methods, significantly influencing key aspects such as failure risk, scalability, cost efficiency, and control over data. Functionality varies across TIRs, with issuer identification and authorization being key but inconsistently implemented features. Scalability is addressed in some TIRs through hierarchical structures, yet often without sub-registry authorization. Security concerns arise from issues like data integrity of external resources and the challenges of delegated trust, with some TIRs exhibiting high centralization that conflicts with Self-Sovereign Identity principles. Complexity ranges from simple to intricate systems, underscoring a tradeoff between functionality and simplicity. Overall, the analyzed solutions pursue interesting approaches but also have potential for improvement, highlighting the gap an improved design could fill.

The requirements analysis using expert interviews with future users of TIRs provided valuable insights into the specific needs of TIRs within the actual use case of Gaia-X. We established four basic functional requirements and 13 non-functional requirements, the latter being the most important for our TIR design. Regarding functionality, we found that issuer identification, issuer authorization, and hierarchy using sub-registries with authorization are required for the SSI use case of Gaia-X. We have also established that decentralized management of the TIR through voting is not required and that the participants had no clear preference between centralized and decentralized storage types. Instead, availability and scalability were more important. The TIR design should also be portable and interoperable.

From the evaluation of existing TIRs and the requirements analysis, we inferred an improved decentralized TIR architecture that meets the established requirements

and learned from the existing TIRs' advantages and disadvantages. The design is highly portable and interoperable by separating architecture and technology using the concept of TIR methods. In general, TIRs are referenced using DID services linking to TIR method-specific endpoints. Our hierarchical design provides issuer identification, authorization, and sub-registry authorization to limit all delegated trust.

Finally, we implemented a prototype application for our design using several components. A Tezos smart contract allows the operation of a TIR on the Tezos blockchain. A Typescript core package combines TIR resolution and issuer verification logic. This logic is then used in a backend that can cache resolved TIRs and serve verification requests. A web frontend allows the manual management of a Tezos TIR and verifying issuers, including a detailed visualization of the trust path. The whole application proves the concept of our TIR design, demonstrates its functionality and can serve as an already advanced starting point for use in production.

It's important to acknowledge the limitations of our research. Our analysis of Trusted Issuer Registries was based on a selected subset of existing solutions, potentially overlooking emerging TIRs not covered in public documentation. Further research could analyze existing TIRs not only in theory but also in practice. The expert interviews, while insightful, were limited in number and confined to a specific project within Gaia-X, which may not fully represent the broader requirements of the ecosystem. Our TIR design, while innovative, faces challenges with data privacy of issuer verifications and the need for multiple resolution steps due to its decentralized and interoperable nature. Moreover, some of our implementation requires further refinement. For example, the resolution algorithm for TIRs does not yet exploit the design's full potential in terms of caching. Furthermore, due to the scope of this thesis, there has not been an in-depth evaluation of our design and prototype. Further research is needed to, e.g., evaluate the design's scalability in practice. These limitations highlight the need for ongoing research and development to enhance the comprehensiveness and applicability of TIRs in the evolving landscape of Self-Sovereign Identity.

In conclusion, this thesis presents a novel decentralized Trusted Issuer Registry design based on an extensive analysis of previous solutions and requirements, whose implementation is demonstrated comprehensively in a prototype application. We are optimistic that Trusted Issuer Registries can solve the problem of issuer trust with Verifiable Credentials and thus level the ground for a more widespread use of Self-Sovereign Identity.

Abbreviations

CA Certificate Authority

DCC Digital Credentials Consortium

DID Decentralized Identifier

DNS Domain Name System

EBSI European Blockchain Services Infrastructure

ETSI European Telecommunications Standards Institute

OCI Open Credentialing Initiative

PKI Public Key Infrastructure

SSI Self-Sovereign Identity

ToIP Trust Over IP

TIR Trusted Issuer Registry

TRAIN Trust mAnagement INfrastructure

IPFS InterPlanetary File System

VC Verifiable Credential

W3C World Wide Web Consortium

List of Figures

6.1 Example of a DID Document referencing a TIR 32
6.2 Our Trusted Issuer Registry's Data Model 34
6.3 Example VC representation of our TIR design 38

7.1 The Tezos TIR data model in JsLigo code 49
7.2 Screenshot of the web app displaying the contents of an example TIR . 55
7.3 Screenshot of the web app's user interface for registering a new issuer to a Tezos TIR 56
7.4 Screenshot of the web app's user interface for issuer verification . . . 57

List of Tables

4.1 Partial overview of the analyzed TIRs' distinctive characteristics 20

6.1 Comparison of our design and the analyzed TIRs, based on Table 4.1 . 43

Bibliography

[1] R. Soltani, U. T. Nguyen, and A. An, “A Survey of Self-Sovereign Identity Ecosystem,” *Security and Communication Networks*, vol. 2021, e8873429, Jul. 19, 2021, ISSN: 1939-0114. DOI: 10.1155/2021/8873429.

[2] C. Allen. “The Path to Self-Sovereign Identity,” Life With Alacrity. (Apr. 25, 2016), [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[3] M. Sabadello, M. Sporny, A. Guy, and D. Reed, “Decentralized Identifiers (DIDs) v1.0,” W3C, W3C Recommendation, Jul. 2022. [Online]. Available: https://www.w3.org/TR/2022/REC-did-core-20220719/.

[4] D. Longley, B. Zundel, K. D. Hartog, D. Burnett, G. Noble, and M. Sporny, “Verifiable Credentials Data Model v1.1,” W3C, W3C Recommendation, Mar. 2022. [Online]. Available: https://www.w3.org/TR/2022/REC-vc-data-model-20220303/.

[5] C. Brunner, U. Gallersdörfer, F. Knirsch, D. Engel, and F. Matthes, “DID and VC: Untangling Decentralized Identifiers and Verifiable Credentials for the Web of Trust,” in *Proceedings of the 2020 3rd International Conference on Blockchain Technology and Applications*, ser. ICBTA ’20, New York, NY, USA: Association for Computing Machinery, Mar. 21, 2021, pp. 61–66, ISBN: 978-1-4503-8896-2. DOI: 10.1145/3446983.3446992.

[6] Manu Sporny, Oskar van Deventer, Isaac Henderson Johnson Jeyakumar, Shigeya Suzuki, Konstantn Tsabolov, Line Kofoed, and Rieks Joosten, “Verifiable Issuers & Verifiers,” in *Rebooting the Web of Trust XI.* Dec. 19, 2022. [Online]. Available: https://github.com/WebOfTrustInfo/rwot11-the-hague/blob/master/final-documents/verifiable-issuers-and-verifiers.pdf (visited on 10/11/2023).

[7] R. Soltani, U. T. Nguyen, and A. An, “A Survey of Self-Sovereign Identity Ecosystem,” *Security and Communication Networks*, vol. 2021, C. Galdi, Ed., pp. 1–26, Jul. 17, 2021, ISSN: 1939-0122, 1939-0114. DOI: 10.1155/2021/8873429.

[8] O. Steele and M. Sporny, "DID Specification Registries," W3C Group Note, Oct. 27, 2023. [Online]. Available: https://w3c.github.io/did-spec-registries/ (visited on 12/06/2023).

[9] T. Berners-Lee, R. T. Fielding, and L. M. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," Internet Engineering Task Force, Request for Comments RFC 3986, Jan. 2005, 61 pp. DOI: 10.17487/RFC3986.

[10] I. Herman, G. Cohen, O. Steele, M. Jones, M. Sporny, O. Terbu, and T. T. Jr, "Verifiable credentials data model v2.0," W3C, W3C working draft, Dec. 2023. [Online]. Available: https://www.w3.org/TR/2023/WD-vc-data-model-2.0-20231202/.

[11] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," Internet Engineering Task Force, Request for Comments RFC 8259, Dec. 2017, 16 pp. DOI: 10.17487/RFC8259.

[12] G. Kellogg, D. Longley, and P.-A. Champin, "JSON-LD 1.1," W3C, W3C recommendation, Jul. 2020. [Online]. Available: https://www.w3.org/TR/2020/REC-json-ld11-20200716/.

[13] M. B. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," Internet Engineering Task Force, Request for Comments RFC 7519, May 2015, 30 pp. DOI: 10.17487/RFC7519.

[14] O. Terbu and D. Fett, "SD-JWT-based Verifiable Credentials (SD-JWT VC)," Internet Engineering Task Force, Internet-draft, Oct. 23, 2023, 24 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/01/.

[15] European Commission, *The European Digital Identity Wallet Architecture and Reference Framework - Version 1.0.0*, Jan. 2023. [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework (visited on 12/06/2023).

[16] A. Wright, H. Andrews, B. Hutton, and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents," Internet Engineering Task Force, Internet Draft draft-bhutton-json-schema-01, Jun. 10, 2022, 78 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-bhutton-json-schema-01 (visited on 11/22/2023).

[17] "Vision & Mission," Gaia-X: A Federated Secure Data Infrastructure, [Online]. Available: https://gaia-x.eu/what-is-gaia-x/vision-and-mission/ (visited on 12/06/2023).

[18] *Gaia-X Architecture Document - 23.1O Release*. [Online]. Available: https://docs.gaia-x.eu/technical-committee/architecture-document/latest/ (visited on 11/22/2023).

[19] L. M. Goodman, *Tezos — a self-amending crypto-ledger*, Sep. 2, 2014. [Online]. Available: https://tezos.com/whitepaper.pdf.

[20] "Tezos Developer Documentation." (Sep. 11, 2023), [Online]. Available: https://docs.tezos.com/overview/ (visited on 12/01/2023).

[21] *Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC*, Jul. 23, 2014. [Online]. Available: http://data.europa.eu/eli/reg/2014/910/oj/eng (visited on 12/11/2023).

[22] "ETSI TS 119 612 V2.2.1 (2016-04)," [Online]. Available: https://www.etsi.org/deliver/etsi_ts/119600_119699/119612/02.02.01_60/ts_119612v020201p.pdf (visited on 08/12/2023).

[23] I. H. Johnson Jeyakumar, D. W. Chadwick, and M. Kubach, "A novel approach to establish trust in verifiable credential issuers in Self-sovereign identity ecosystems using TRAIN," in *Open Identity Summit 2022*, Bonn: Gesellschaft für Informatik e.V., 2022, pp. 27–38, ISBN: 978-3-88579-719-7. DOI: 10.18420/OID2022_02.

[24] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Internet Engineering Task Force, Request for Comments RFC 8446, Aug. 2018, 160 pp. DOI: 10.17487/RFC8446.

[25] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements," in *2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 511–525. DOI: 10.1109/SP.2013.41.

[26] H. Hoogstraaten, *Black Tulip Report of the Investigation into the DigiNotar Certificate Authority Breach*. Fox-IT BV, Aug. 13, 2012. DOI: 10.13140/2.1.2456.7364.

[27] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," Internet Engineering Task Force, Request for Comments RFC 6962, Jun. 2013, 27 pp. DOI: 10.17487/RFC6962.

[28] B. Laurie, "Certificate transparency," *Communications of the ACM*, vol. 57, no. 10, pp. 40–46, Sep. 23, 2014, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/2659897.

[29] Jürgen Schmidt. “TLS-Zertifikate: Google zieht Daumenschrauben der CAs weiter an,” Security. (May 27, 2016), [Online]. Available: https://heise.de/-3215053 (visited on 11/29/2023).

[30] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” Internet Engineering Task Force, Request for Comments RFC 5280, May 2008, 151 pp. DOI: 10.17487/RFC5280.

[31] P. Hallam-Baker, R. Stradling, and J. Hoffman-Andrews, “DNS Certification Authority Authorization (CAA) Resource Record,” Internet Engineering Task Force, Request for Comments RFC 8659, Nov. 2019, 17 pp. DOI: 10.17487/RFC8659.

[32] “Home - EBSI,” European Blockchain Services Infrastructure (EBSI), [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/ebsi/ (visited on 11/22/2023).

[33] “Early Adopters Programme,” European Blockchain Services Infrastructure (EBSI), [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Early+Adopters (visited on 11/22/2023).

[34] “EBSI’s success stories - Bachelor/Master Degree,” European Blockchain Services Infrastructure (EBSI), [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Bachelor+-+Master+Degree (visited on 11/22/2023).

[35] “EBSI’s success stories - Municipality Credentials,” European Blockchain Services Infrastructure (EBSI), [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Municipality+Credentials (visited on 11/22/2023).

[36] *Trusted Issuers Registry API*, European Blockchain Services Infrastructure (EBSI). [Online]. Available: https://hub.ebsi.eu/apis/pilot/trusted-issuers-registry/ (visited on 10/11/2023).

[37] “Issuer Trust Model,” EBSI Hub. (Nov. 21, 2023), [Online]. Available: https://hub.ebsi.eu/vc-framework/trust-model/issuer-trust-model (visited on 12/04/2023).

[38] S. Solat, P. Calvez, and F. Naït-Abdesselam, “Permissioned vs. Permissionless Blockchain: How and Why There Is Only One Right Choice,” *Journal of Software*, vol. 16, pp. 95–106, Dec. 25, 2020. DOI: 10.17706/jsw.16.3.95-106.

[39] “Issue VC for Trusted Issuer,” EBSI Hub, [Online]. Available: https://hub.ebsi.eu/tools/cli/issue-vc-trusted-issuer (visited on 12/04/2023).

[40] *Trusted Schemas Registry API*, European Blockchain Services Infrastructure (EBSI). [Online]. Available: https://hub.ebsi.eu/apis/pilot/trusted-schemas-registry (visited on 12/07/2023).

[41] GXFS. "Second Phase of the XFSC Specification," GXFS.eu. (Sep. 21, 2023), [Online]. Available: https://www.gxfs.eu/second-phase-of-the-xfsc-specification/ (visited on 10/18/2023).

[42] GXFS. "Trades awarded in second GXFS Specification Phase," GXFS.eu. (Nov. 2, 2023), [Online]. Available: https://www.gxfs.eu/trades-awarded-in-second-gxfs-specification-phase/ (visited on 12/06/2023).

[43] "Domain names - concepts and facilities," Internet Engineering Task Force, Request for Comments RFC 1034, Nov. 1987, 55 pp. DOI: 10.17487/RFC1034.

[44] "Digital Credentials Consortium," [Online]. Available: https://digitalcredentials.mit.edu/ (visited on 11/25/2023).

[45] J. Chartrand, S. Freeman, U. Gallersdörfer, M. Lisle, A. Mühle, and Sélinde Van Engelenburg, *Building the Digital Credential Infrastructure for the Future: A White Paper by the Digital Credentials Consortium*. Digital Credentials Consortium, 2018. [Online]. Available: https://digitalcredentials.mit.edu/docs/white-paper-building-digital-credential-infrastructure-future.pdf (visited on 11/25/2023).

[46] Kim Hamilton Duffy, *Issuer Registry MVP*. Digital Credentials Consortium, Nov. 20, 2020. [Online]. Available: https://github.com/digitalcredentials/docs/blob/main/identity/issuer_registry.md (visited on 07/24/2023).

[47] "About OCI," Open Credentialing Initiative, [Online]. Available: https://www.oc-i.org/about-the-open-credentialing-initiative (visited on 12/06/2023).

[48] *Trusted-issuer-registry*, GitHub: Open Credentialing Initiative, 2023. [Online]. Available: https://github.com/Open-Credentialing-Initiative/trusted-issuer-registry.

[49] R. Celeste, C. Wirrig, E. Waldorf, G. Jürgens, and L. Leifermann, *OCI Governance*, May 15, 2022. [Online]. Available: https://open-credentialing-initiative.github.io/OCI-Governance/ (visited on 12/06/2023).

[50] "About Us," Trust Over IP, [Online]. Available: https://trustoverip.org/about/about/ (visited on 12/07/2023).

[51] D. O'Donnell and D. Reed, *ToIP Trust Registry Protocol V1 Specification*, GitHub, Sep. 16, 2021. [Online]. Available: https://github.com/trustoverip/tswg-trust-registry-tf/blob/main/v1/docs/ToIP%20Trust%20Registry%20V1%20Specification.md (visited on 12/07/2023).

[52] D. O'Donnell, *ToIP Trust Registry Protocol V2 Specification*, GitHub, Nov. 16, 2023. [Online]. Available: https://github.com/trustoverip/tswg-trust-registry-tf/blob/main/v2/requirements.md (visited on 12/07/2023).

[53] J. Golosova and A. Romanovs, "The Advantages and Disadvantages of the Blockchain Technology," in *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, Nov. 2018, pp. 1–6. DOI: 10.1109/AIEEE.2018.8592253.

[54] P. Poonpakdee, J. Koiwanit, and C. Yuangyai, "Decentralized Network Building Change in Large Manufacturing Companies towards Industry 4.0," *Procedia Computer Science*, 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops, vol. 110, pp. 46–53, Jan. 1, 2017, ISSN: 1877-0509. DOI: 10.1016/j.procs.2017.06.113.

[55] S. Yin, Y. Teng, N. Hu, and X. D. Jia, "Decentralization of DNS: Old Problems and New Challenges," in *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, ser. CIAT 2020, New York, NY, USA: Association for Computing Machinery, Jan. 4, 2021, pp. 335–341, ISBN: 978-1-4503-8782-8. DOI: 10.1145/3444370.3444594.

[56] D. Kontokostas and H. Knublauch, "Shapes constraint language (SHACL)," W3C, W3C recommendation, Jul. 2017. [Online]. Available: https://www.w3.org/TR/2017/REC-shacl-20170720/.

[57] S. Debnath, A. Chattopadhyay, and S. Dutta, "Brief review on journey of secured hash algorithms," in *2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix)*, Nov. 2017, pp. 1–5. DOI: 10.1109/OPTRONIX.2017.8349971.

[58] Wayne Chang, Michael Klein, and Simon Bihel, *The Tezos DID Method*, Spruce Systems Inc., Mar. 3, 2021. [Online]. Available: https://github.com/spruceid/did-tezos/blob/main/index.html (visited on 12/10/2023).

[59] L. Sweeney, "K-ANONYMITY: A MODEL FOR PROTECTING PRIVACY," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, Oct. 2002, ISSN: 0218-4885. DOI: 10.1142/S0218488502001648.

[60] Darrel Miller, Jeremy Whitlock, Marsh Gardiner, Mike Ralphson, and Ron Ratovsky, *OpenAPI Specification v3.1.0*, Feb. 15, 2021. [Online]. Available: `https://spec.openapis.org/oas/v3.1.0`.

[61] Michael Schmidmaier, *Trusted Issuer Registry 2023 Prototype Implementation*, GitHub, Dec. 14, 2023. [Online]. Available: `https://github.com/Trusted-Issuer-Registry-2023/tir2023-prototype`.