University of Zurich[UZH]

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

Federated Governance in a Digital Trust Coalition

Andy Aidoo
Zurich, Switzerland
Student ID: 17-700-451

Supervisor: Dr. Bruno Rodrigues, Katharine O. E. Mueller, Prof. Dr. Burkhard Stiller
Date of Submission: February 28, 2024

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

ifi

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

Abstract

The importance of information and secure communication thereof is well known. A recent paradigm shift in data engineering towards the so called *data mesh* takes a domain-driven approach similar to the well established microservice architecture in software engineering. Rather than separating operational from analytical data and consequently relying on a plethora of data pipelines to aggregate analytical from operational data, business domains who are closest to the data are responsible for developing and maintaining the data products their domain offers. To manage the life cycle of such a data product, federation of identity management alongside its governance become crucial aspects, highlighting areas for further research; establishing trust between domains, especially remains challenging to this day. This Master's Thesis presents a proof of concept for federated governance and identity management in a data mesh architecture aiming to provide means for secure information exchange between domains of various companies. A standardized onboarding process allows new domains to join through one of the two joiner-processes; they can start an application to attain membership-status of the Trust Coalition by filling out the sign up form. Provided that a two-thirds majority of the pre-established coalition participants vote in favor of the admission, a domain's application is accepted and authorization servers of the Trust Coalition members can now authenticate users of the newly added domain. To verify the identity of users, we employ a two-factor authentication mechanism using possession-based factors in form of AnonCreds verifiable credentials; access to the credentials is secured through a mobile client that has to be unlocked utilizing either a knowledge-based or biometric authentication factor. An accelerated onboarding process is offered for reputable domains; within the framework of this Master's Thesis, a domain is deemed reputable if it possess an extended validation or organisation validated TLS certificate. To verify the owner ship of the domain, the crypotgraphic key material is extracted from its certificate and used to encrypt a freshly generated mnemonic phrase; given the cipher text can be submitted as plain text, the application is accepted.

Zusammenfassung

Die Bedeutung von Information und Notwendigkeit ihrer sicheren Übermittlung ist wohlbekannt. Der jüngste Paradigmenwechsel im Bereich Data-Engineering hin zum sogenannten Data Mesh verfolgt einen domänenorientierten Ansatz, ähnlich der bereits etablierten Microservice-Architektur im Software-Engineering. Anstatt operative von analytischen Daten zu trennen, sind Geschäftsbereiche, die den Daten am nächsten sind, für die Entwicklung und Pflege, der von ihrem Bereich angebotenen Data Products, verantwortlich. Dies bringt den Vorteil mit sich, dass Unternehmen nicht mehr von einer Vielzahl an Datenpipelines abhängig sind, um analytische aus operativen Daten zu aggregieren. Um den Lebenszyklus eines solchen Data Products zu verwalten, ist eine Föderation des Identitätsmanagements zusammen mit einer bereichsübergreifenden Governance von wesentlicher Relevanz; beide Aspekte zeigen Bereiche für weitere Forschung auf. Der Aufbau einer Vertrauensbeziehung zwischen verschiedenen Domänen bleibt bis heute eine Herausforderung. In dieser Masterarbeit wird ein Proof-of-Concept für föderierte Governance in einer Data-Mesh Architektur vorgestellt, der den sicheren Informationsaustausch zwischen Domänen verschiedener Organisationen ermöglichen soll. Ein standardisierter Onboarding-Prozess ermöglicht es neuen Domänen, durch einen der beiden Joiner-Prozesse beizutreten; Organisationen können durch Ausfüllen des Anmeldeformulars einen Mitgliedschafts-Antrag an die Trust Coaltion stellen. Vorausgesetzt, dass eine Zwei-Drittel-Mehrheit der bestehenden Mitglieder für eine Aufnahme stimmt, wird der Antrag einer Domäne angenommen und die Autorisierungsserver der Mitgliedr der Trust Coalition können nun Verifiable Credentials authentisieren und den Angestellten der neu aufgenommenen Organisation sicheren Zugriff gewähren. Um die Identitäten der Nutzer zu verifizieren, verwenden wir einen Zwei-Faktor-Authentifizierungsmechanismus mit besitzbasierten Faktoren in Form von AnonCreds Verifiable Credentials; der Zugriff auf die Anmeldedaten wird durch einen mobilen Client gesichert, der entweder mit einem wissensbasierten oder biometrischen Authentifizierungsfaktor freigeschaltet werden muss. Für vertrauenswürdige Domänen wird ein beschleunigter Onboarding-Prozess angeboten. Im Rahmen dieser Masterarbeit gilt eine Domäne als vertrauenswürdig, wenn sie über ein Extended Validation oder Organisation Validated TLS-Zertifikat verfügt. Um den Besitz der Domäne zu verifizieren, wird das kryptografische Schlüsselmaterial aus dem Zertifikat extrahiert und zur Verschlüsselung einer neu generierten mnemonischen Phrase verwendet; wenn der verschlüsselte Text als Klartext übermittelt werden kann, wird der Antrag akzeptiert.

Acknowledgments

Throughout the formulation of this thesis, the guidance of my supervisor, Dr. Bruno Rodrigues, has been instrumental and I want to express my profound appreciation for it. During weekly meetings he provided insightful feedback that significantly contributed to the development and refinement of my master's thesis.

I want to extend my gratitude to Prof. Dr. Burkhard Stiller for providing a platform that fosters curiosity at the department of informatics within the University of Zurich. This project at the Communication Systems Research Group has been demanding and rewarding.

Contents

Chapter 1

Introduction

There are various undertakings under the umbrella term *data exchange* for public consumption or in contractually bound partnerships, only and even ecosystems thereof. Three major sections in this problem class concern themselves with open government (OG) data, open banking (OB) and open finance (OF). Switzerland's central government has committed towards the *open by default* principle, which obliges them to provide access to their data to *'... reinforce transparency, participation, and innovation in all sectors of society'* [54]. In the financial sector, efforts in OB and OF aim at sharing data about financial products, as well as Personally Identifiable Information (PII). Open finance differs from open banking since the former concept includes access to aforementioned data (with explicit consent from the concerned customer) by third-parties that do not play a direct role in transactions. According to the German Federal Financial Supervisory Authority (BaFin), open finance is a novel topic of discussion, and the development of legal frameworks is still in its children's shoes [22].

The importance of information for businesses has been known for several decades [59]; according to a white paper published by the Word Economic Forum (WEF), manufacturers focus on data-driven business intelligence mostly based on internal data. Their centralized approaches fail to optimize resource allocation; switching to a decentralized point of view, where data is shared across companies has the potential for process optimization valued upwards of $100 billion. As one of the most prominent barriers, they identify building trust between collaborating companies [7].

Internet worms and Distributed-Denial of Service (DDoS) attacks pose serious risks resulting in large financial damages to victims; for defensive countermeasures to be effective, sophisticated and coordinated mechanisms by a multitude of parties are required [72]. Cooperative defense mechanisms are beneficial in critical infrastructures, especially. However, the sensitivity of the shared information poses intricate challenges and postulates some form of trust management [23].

Aforementioned information sharing undertakings all have a common denominator; each data provider requires assurances that only the right people have access to the right data for the right reasons.

This Master's Thesis aims at contributing to state-of-the art research in communication security for highly decentralized systems. We developed a prototype for participating in the governance of a decentralized autonomous organization (DAO) through invocation of Ethereum smart contracts and identity management based on distributed layer technology (DLT). *Hyperledger Indy* [1], the underlying blockchain, and *Hyperledger Aries* [2], a library for decentralized identity solutions and digital trust, enable issuing and revoking AnonCred [3] verifiable credentials (VC). Lastly, the controller *trust coalition agent* (TCA) responsible for exposing functionality to manage the life cycle of VCs uses authentication and authorization based on self-sovereign identity (SSI).

1.1 Motivation

The Internet was created without an identity layer; consequently, applications reachable via the Internet, often require complex mechanisms to authenticate users that request resources [14]. To that extent, few centralized service providers maintain registries with PII for billions of users. Said digital identity providers pose a serious cluster risk; PII of millions of users have been part of numerous data breaches [48]. Rather than relying on federated authentication by a few enormous identity providers, digital identities should be in control of the corresponding user [14]

Motivated by the continuous increase of DDoS attacks - the largest attack seen by Cloudflare in 2023 exceeded a staggering 201 million requests per second [71] - this Thesis devises a framework allowing for information exchanges across company borders. Due to the widely distributed nature of DDoS attacks, an ideal counter strategy also involves a distributed defense to mitigate attacking traffic at different points closer to their origin [61]. Nonetheless, there exist challenges in different dimensions to provide agile and robust communication between cooperating domains. For example, technical, legal, economic, and social dimensions impact how such a data architecture is deployed and operated.

As cybersecurity becomes increasingly data-oriented, a data mesh structure specifically crafted to share DDoS-related information becomes an interesting solution to improve defense systems working in isolation. While a data lake refers to a centralized architecture to store structured or unstructured data, a data mesh is a decentralized and distributed architecture where an organization can independently exchange information about attacks and collaborate [6]. In the case of DDoS attack fingerprints, a data mesh could be used to share information about the attack's origin, type, and volume and the methods used to detect and mitigate attacks.

A DDoS fingerprint [4] data mesh represents a shared repository for exchanging information in near real-time or conducting forensic studies on shared DDoS traces to improve defense systems continuously. Specifically, it enables the use of artificial intelligence/machine learning algorithms to discover new patterns that can be transformed into signatures,

[1] Hyperledger Indy
[2] Hyperledger Aries
[3] AnonCreds Specification
[4] DDoS Clearing House

enables a global view of attacks and the expansion of observation points across multiple organizations that share data in the DDoS fingerprint data mesh, and improves integration and automation by being a data-driven approach at its core.

1.2 Thesis Goals

This thesis tackles the federated governance aspect within a coalition of trusted domains and standardized identity management. Implementing federated governance for such a trust coalition requires a holistic strategy, especially given the unique challenges and opportunities presented by the sensitivity of the exchanged information. In this case, the goal of the thesis is the development of a governance framework and a prototype agent thereof to collaboratively govern access to DDoS fingerprints. Essentially, members of the trust coalition share their individual perspectives or footprints of cyberattacks.

By relying on cutting-edge concepts in data engineering and digital identity, the published data should remain confidential, *i.e.* only accessible by authenticated and authorized users; enable trusted domains to exchange information and collectively extract value from each other's information. Such swarm intelligence allows for more comprehensive and adaptive defense mechanisms by adopting a model in which multiple entities share and collaboratively analyze attack data.

This collective defense is expected to provide a more effective barrier against cyber threats than siloed defenses, thus, promoting a more resilient ecosystem against DDoS attacks. Ultimately, the prototype's identity management and authentication should be designed for interoperability to unlock similar synergies in other use cases where information communication between digital domains is at the core.

1.3 Methodology

This thesis follows a systematic approach that consists of four phases.

1. Literature review
2. Requirements engineering
3. Prototype implementation
4. Prototype evaluation

For the initial period, the tasks involve reviewing existing literature, standards, and best practices. Subsequent tasks concern themselves with structured requirements engineering based on Pohl et al. [58]. The second reference point is marked by the definition of a project description that records the involved stakeholders alongside their general requirements. The subsequent milestone is achieved once an illustration of the system context constraints

the scope of the prototype and the involved components receive their responsibilities. Before entering phase four we will develop use cases for a client application with user-centricity, privacy-preservation, and interoperability in mind. The final practical phase consists of developing and testing the prototype TCA to participate in the federated governance of a trusted coalition and authentication based on SSI. Before concluding, we evaluate the TCA against but not limited to scalability, assurance of a credential's authenticity, and holder's identity. The focus on the aforementioned metrics aims to document the effectiveness of the prototype and reveal potential areas for improvement.

1.4 Thesis Outline

This thesis consists of six chapters; after the introduction 1 follows Chapter 2 background. Its first Section 2.1 IT Security covers concepts to provide sufficient context for the remainder of the thesis and refers to further resources where apt. The concepts included are the three original communication security goals: confidentiality, integrity and availability. We further cover the fundamentals of digital identities in 2.2, two data engineering paradigms data lake 2.3 and data mesh 2.4. A section about the Internet and DDoS attacks 2.5 conveys the context that inspired the governance framework. Lastly, Related Work 2.6 summarizes previous research conducted in this field of study.

In chapter 3, we establish the design principles that govern the development of the prototype with section 3.1. In Section 3.2 we introduce the stakeholders of the Trust Coalition. We further describe the different aspects of the solution; the sections summarize the three categories of use cases: membership life cycle in Section 3.3, verifiable credentials life cycle in Section 3.4 and client access control in Section 3.5. Lastly, Section 3.6 describes the rationale behind the selection of the identity solution used to build the prototype.

The developed prototype builds the foundation for the documentation of the implementation in Chapter 4; it describes the various components of the implemented software system. It includes a service for managing DIDComm connections explained in Section 4.1, a Section 4.2 about the service that concerns itself with verifiable credentials and the verification of them with Section 4.3. Further, Section 4.4 describes how the governance model works. Section 4.5 includes information about the developed API that connects aforementioned services. The library that is pivotal for the services and its configuration is included in Section 4.6. Subsequently, Section 4.7 explains how revocation is enabled with an additional component. Finally, Section 4.8 depicts the developed shell script to *i.a.* start and stop the prototype.

With chapter 5 follows a chapter about devising a measurement of the prototype and a discussion about the results. In section 5.1 metrics, we evaluate TCA based on scalability and identity/authentication/federation assurance levels, and a requirements-based evaluation with Section 5.2 before presenting the system in action within Section 5.3. Subsequently Section 5.4 states remarks about implications and limitations of the prototype.

To conclude, chapter 6 delivers an executive summary in Section 6.1, reflection about the project in Section 6.2 and suggests directions for future work ins Section 6.3.

Chapter 2

Background

This Chapter presents a foundational base of concepts used in this thesis. It provides an overview of fundamental IT security concepts - the original communication security goals, an introduction to digital identities, data lakes and mesh architecture, and how distributed denial-of-service attacks play a role on the Internet. Finally, related research is reviewed, and a border to this thesis is established.

2.1 IT Security

Information technology (IT) is ubiquitous; virtually everyone relies on even the most critical aspects of one's life. Even before today's dependence on IT, Voydock and Kent [66] formulated three categories of communication security goals that will be elaborated on in the subsequent passages, namely *Confidentiality*, *Integrity* and *Availability* (CIA). These security attributes serve as the foundation for exploring the subsequent sections.

2.1.1 Confidentiality

To satisfy the confidentiality of data, unauthorized access to it must be prevented; this applies to data at rest and in-transit [39]. Thus, before accessing data, requesters of resources must be authenticated. Authentication means providing a piece of information that distinctly defines the identity of a system user. Typically, a user's knowledge, ownership, or biometric factors are authentication factors. Password-based authentication is an example of a knowledge-based factor, while smart cards or fingerprints are instances of the ownership- or biometric factor, respectively [55, 65]. Subsequently, it must be checked whether the requester is authorized to perform reading, editing or deleting operations, depending on the request type. Only then may a request be granted or rejected [39].

In the event of a data breach or when data is sent over a network, confidentiality of the data should still be guaranteed, *i.e.* access to the data itself must not reveal any useful information. This can be achieved by applying a cipher to the data. When data is

encrypted, an algorithm transforms its values using a so-called key, rendering it unusable. The decryption of the cipher text requires the same key in secret-key encryption, while public-key cryptography utilizes a key pair; the public key can be used to encrypt data, and the secret key provides the means for deciphering it [66]. Figure 2.1 shows the flow that allows Alice and Bob to exchange encrypted messages using a secret key in the upper half, while the bottom half shows the functional equivalent through public key cryptography.

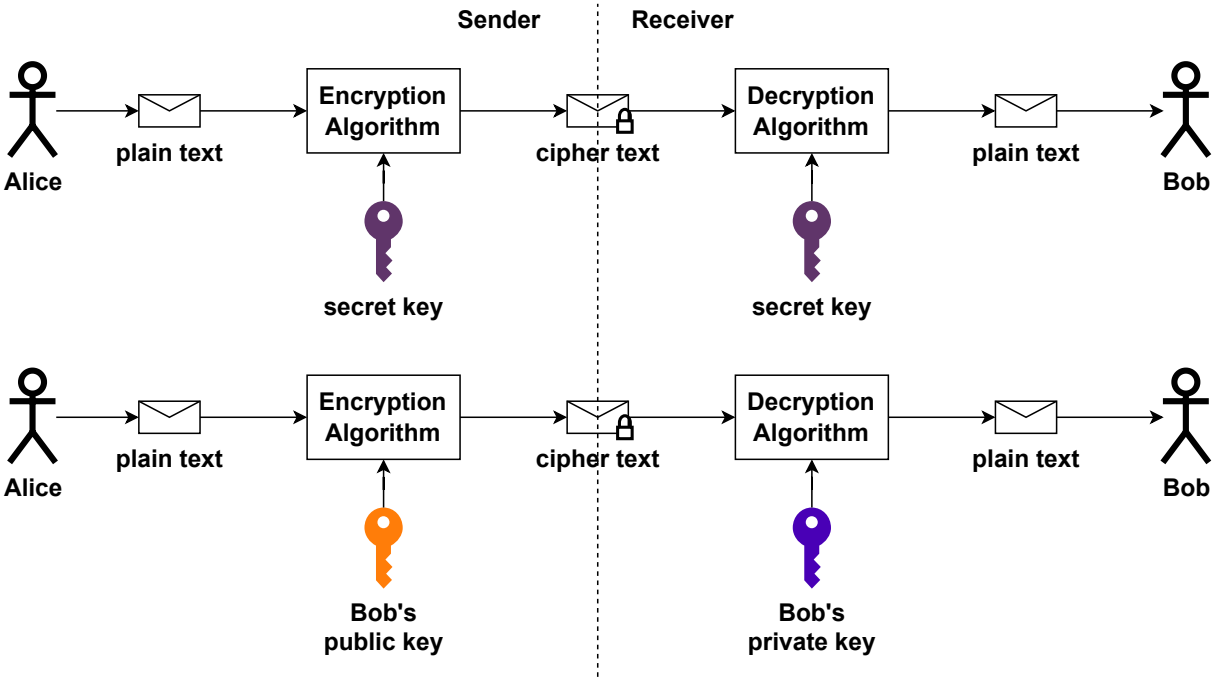

Figure 2.1: Message encryption with secret and public key cryptography

2.1.2 Integrity

Data integrity refers to ensuring that modifications to data are only possible in a previously defined manner, typically involving cryptography. Generally speaking, data is augmented with a validation code that the data can be checked against to validate if the data at hand is the same that was used to generate the validation code. Additionally, Voydock and Kent [66] include authenticity, *i.e.* monitoring the author of a modification within this communication security goal. Similarly to the previous Section, Confidentiality 2.1.1, two paradigms can ensure data integrity: secret and public-key cryptography.

In secret-key (also called symmetric key) cryptography, two parties possess the same encryption key alongside a shared hash function. An author of a data item passes it to the hash function, which produces, in our example, a string with 32 characters, encrypts the output with the secret key, and adds the resulting cipher text to the data's meta-information. The cipher text is the Message Authentication Code (MAC). To validate whether the data contents have been tampered with, the second party can generate a MAC with the same hash function and key, analogous to the data item author, as illustrated in 2.2.

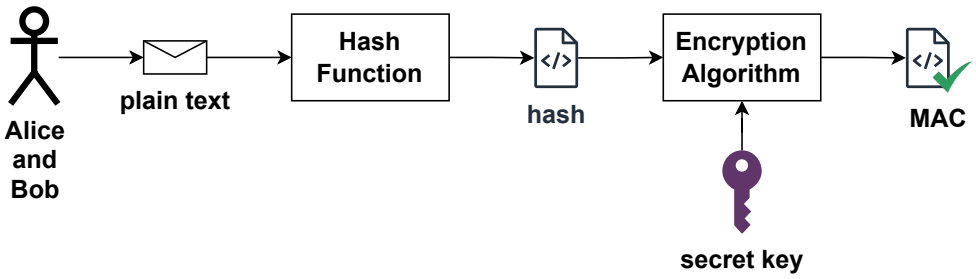

Figure 2.2: MAC generation to authenticate the integrity of a message

When the resulting MAC matches with the MAC in the metadata, the second party can infer that the data item has not been modified since the author created the MAC [64]. Since several people, at least two, retain a copy of the key used in creating the MAC, the author of a change cannot be uniquely distinguished.

When using a public key scheme to ensure data integrity, the author of a change is uniquely identified by the signature of a data item. Analogously to the generation of the MAC, an author uses a hash function to create a hash that is signed (in this case) using the sender's private key - this produces a signature that only the author can generate with its private key. To verify the integrity of a data item the receiving party uses the same hash function as the sender. Subsequently, the message signature can be verified by anyone with access to the author's public key; it can be used to generate a second hash from the digital signature. If the hashes are equivalent, the message is authentic. This approach, however, requires a mutually trusted third party that records the ownership of a public key's counterpart[52]. For an illustration of the process, *cf.* Figure 2.3.

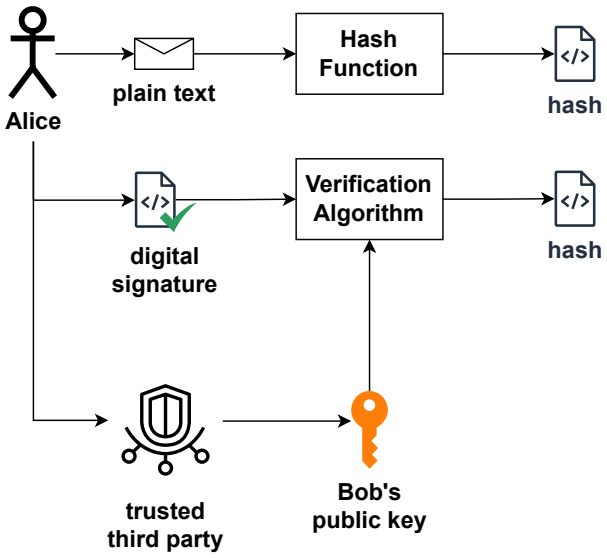

Figure 2.3: Verification of message authenticity through validating the digital signature

2.1.3 Availability

According to the American National Research Council [16], availability is the most important pillar of the communication security goals. This is mainly a consequence of the dependence on information and digital services that the users expect. Further, availability builds the foundation for the other two goals, as without it, confidentiality and integrity are non-relevant [60]. An information system ought to provide its services to authorized users promptly. Availability can be adopted from an operational and security-focused point of view. From the operational perspective, availability refers to a system's response time and performance. At the same time, the security aspect relates to contingency plans in the event of *i.a.* natural disasters and (unintentionally) malicious acts [16].

2.2 Digital Identities

To conform with the communication security goals confidentiality, integrity and availability, each application or information management system must maintain a registry of accounts linked to an identity and manage the authorizations of these accounts concerning the application. The NIST SP 800-63 standard defines an identity as *'An attribute or set of attributes that uniquely describe a subject within a given context'* [26]. The past two decades have seen an explosion in the number of applications and a significant evolution in identity management systems. Initially, users managed their credentials like usernames and passwords for every application.

With a growing number of applications, users either find themselves juggling many credentials or reusing them, which poses a serious security risk - according to Laurent and Bouzefrane [40] on average, each year, 400,000 persons are victims of online identity theft in France alone. To ease the burden on the end-user and increase online security, the concept of identity providers (IdP) was introduced; rather than using (new) credentials for each application, applications rely on IdPs to authenticate their users. Nowadays, many websites and applications offer social logins where large corporations like Apple, Facebook, and Google function as IdPs. This convenience, however, compromises user privacy. The large IdPs are incentivized to track each authentication request, build an accurate profile about the users' interests and habits, and sell such data to the highest bidder [48].

The newest paradigm shift in digital identities is generally referred to as decentralized identifiers (DID) and verifiable credentials (VC) or self-sovereign identity (SSI). Rather than relying on large IdPs to reveal a user's identity to an application via OpenID Connect (OIDC) or a SAML-based authentication method, applications can directly ask the users for their identities, which are comprised of one or more VCs and cryptographically signed by a trusted party. Distributed Ledger Technology (DLT) can build the foundation for validating such VCs by functioning as a tamper-proof ledger of public keys [5]; however, DIDs need not necessarily be hosted on a blockchain. The public keys can alternatively be obtained from a public-key infrastructure (PKI) or a web page that is under the control of the issuer [70].

The aforementioned NIST standard defines three assurance types that are associated with a digital identity to balance security with user experience primarily:

1. Identity Assurance Level (IAL)
2. Authentication Assurance Level (AAL)
3. Federation Assurance Level (FAL)

Recapitulatory, IAL is concerned with how strong a correlation between an offline and digital identity is *i.e.*, an account registration process that results in a high IAL must take measures to prevent impersonation and may be time-consuming. AAL, on the other hand, addresses the certainty concerning the identity of a user is verified by using one or several factors (described in 2.1.1) in the authentication process, *i.e.* strong authentication prevents logging into the account of someone else. Finally, FAL depicts the integrity and confidentiality with which an IdP transmits identity information.

2.3 Data Lake

The term data lake was coined by James Dixon in 2010; as per his definition, data lake is a design pattern intended to house a humongous amount of heterogeneous data in its raw format from a single application or system that provides its data to several groups of users [20]. Nowadays, data lakes function as central data repositories housing data from various sources and typically feature large volumes of diverse data sets, frequently exceeding the petabyte barrier. Their functionalities can be grouped into three overarching themes - improved business intelligence, compliance with legal stipulations and promoted innovation. They complement decision support systems by acting as a single source of truth that provides the necessary data for the analysis of products and processes, on-demand and from various points of view [62]. During the ingestion of the data, the structure generally remains untouched, and upon successful load, the data is available for analytical applications; this proves valuable for deriving the most recent insights, as well as the discovery of new and potentially invaluable information [50]. Moreover, since they allow for cost-efficient storage of unprocessed data at scale, their usage as a data archive is suitable [21] - this is especially meaningful for industries with stringent regulatory requirements.

Data lakes store data elements in a flat architecture; at the minimum, each data item possesses a system-wide unique identifier along with its data in its raw format. With this approach, every item can be stored, regardless of its underlying schema [50]. However, this flexibility and lack of schema come at a cost; without any data governance and a large enough data repository, the aforementioned value added diminishes as the schemaless data becomes incomprehensible and the data lake turns into a so-called data swamp [67], *i.e.* without proper context, a 13-digit number could be anything from an OASI- to employee- or order number and renders it useless. The heterogeneous nature of the data comprising a data lake requires implementing a governance framework lest the data cannot be found,

trusted or compliance with legal requirements can hardly be achieved [3]. Meta data that capture high-level information may be leveraged to combat the disintegration of a data lake. Essentially, they allow data consumers to understand the existing data and any implication that might come with the usage thereof [67]. When exploring data, its sensitivity should be visible to the consumer with appropriate measures to prevent privacy breaches, *e.g.* a data lake that contains PII or financial data must not make them available to anyone without the proper justification for its usage [69].

2.4 Data Mesh

The pioneer behind the most recent paradigm shift in data engineering, Zhamak Dehghani, calls her innovation data mesh. She identifies four core principles with implications to not only the data architecture but also to the organizational structure: *Domain-Driven Data Ownership Architecture*, *Data as a Product*, *Self-Serve Infrastructure as a Platform* and *Federated Computational Governance* [34].

According to Dehghani's first principle of domain-drive data ownership architecture, rather than having a centralized data team responsible for providing a single source of truth for all data of an entire enterprise, a data mesh consists of several autonomous participants. It ought to be organized by domains and the responsibilities distributed accordingly, *i.e.* data ownership and the responsibility for its whole life cycle lies with the team with the best knowledge about the data that also usually created it. This approach is comparable to the microservices architecture already well established in software development [49]. This implies a shift from pushing data into a central repository to each business unit serving its data via well-defined interfaces [19]. The aforementioned data architecture principle does have its economic, technical, and regulatory limitations. However, the subsequent principles aim to address them [34].

The second principle, data as a product, emphasizes data quality; according to Norman, a good product offers *i. a.* discoverability, which increases the understanding of a product [53]. Often, the discoverability of data is an issue in a data lake, and the distributed nature of a data mesh only aggravates it even further [18]. Cagan identifies four necessary but not sufficient requirements for a successful product: *value*, *usability*, *feasibility*, and *business viability*. The first requirement refers to whether a product adds value to a customer, while the second refers to how intuitive its usage is. Further, the economic expenditure needs to be lower than the value it adds, and last but not least, regulatory or business strategic policies may also act as burdens [13]. In the context of a data mesh, a product comprises of three components. The data and its associated meta-information build the centerpiece. Secondly, code related to preparing the data before serving it, accessing it via API, and access management. Finally, the infrastructure for building, deploying, and running the aforementioned code also lies inside the scope of the data product [18]. These additional responsibilities spawn overhead; historically, providing only the data and metadata to a data platform team was often met with resistance that can be partly justified by the lack of incentives for the business unit [19]. Nonetheless, in a data mesh, the impact of a data product can be measured more accurately by us-

ing the number of requests for a data product and developing an appropriate incentive structure [34].

Rather than replicating the elaborate infrastructure required for managing the whole life cycle of a data product, a declarative approach that abstracts the complexity in the form of a self-serve platform is necessitated. A complete list of requirements is outside the scope of this Master's Thesis; however, some of the key specifications include:

- Federation of Identity management
- Scalable storage to virtually any size
- Data encryption (stored and in transit)
- Locality of data storage and computation [18, 19, 34]

Finally, federated computational governance is tasked with balancing centralization - enforcing global policies - and decentralization - providing the autonomy needed by domain teams to innovate[69]. Generally speaking, data governance *i.a.* entails specifying a framework for data quality standards, policies of how to use the data, and encouragement for desirable behavior in the usage thereof that are in line with the enterprise's mission statement, strategy, culture, values and norms [68]. As addressed in the previous section 2.3, different sensitivity levels of data products require policies on who can access and share them. Wider and Akhatar [69] propose that domain teams should handle access and classification while sharing and reclassification should be monitored by a central instance. Overriding a data product that lowers the sensitivity level should be reviewed manually, while a stricter classification ought to be approved automatically, lest the central governance becomes a bottleneck. When different domains make further use of the data products, the resulting sensitivity levels can drop or increase *e.g.* in the case of data product with aggregated customer information that answers the question *How many of the customers live in Europe?* all PII is removed (provided that only a number is returned) from the response and the sensitivity level drops. On the other hand, the combination of various anonymized data sources may result in PII, as demonstrated by Narayanan and Shmatikov [51].

2.5 The Internet and DDoS Attacks

The Internet's original goal was to develop an open and scalable architecture that allows for interconnecting numerous heterogeneous networks with the help of the Internet Protocol (IP). Members of one network should be able to communicate with all others, regardless of the underlying hardware. This goal was achieved using a best-effort approach where a network's basic function delivers a semantically correct data packet from source A to destination B [15]. An IP packet consists of a header and a data payload, where the former includes *i.a.* a source and destination IP address and the latter entails data that solely the target evaluates [33]. Hence, a transmitter cannot decide whether the destination server accepts the request or it has simply wasted its resources. Furthermore,

the legitimacy of the source IP address cannot be authenticated, either. The openness of IP leads to the possibility of attacks that aim to overwhelm a server with bot-generated traffic to prevent it from serving legitimate requests - a so-called *Denial of Service* (DoS) attack and directly pose a threat to the availability, as defined in Section 2.1.3, of a system. DoS attacks where the requests originate from various sources are called *Distributed Denial of Service* (DDoS) attacks and are generally more detrimental and harder to protect against [57].

2.6 Related Work

This Section provides an overview of related research conducted into governance models for a data mesh, and DIDs and VCs.

2.6.1 Governance Models for a Data Mesh

Data mesh has picked up in popularity and consequently sparked an interest in the academic community. Wider et al. [69] curated and consolidated the various aspects of a data mesh into a novel conceptual model. Further, they discuss a decentralized approach toward general data governance to impose company-wide data quality controls without excessive strain on central authorities. Such a governance approach minimizes risks for potentially sensitive data while ensuring the veracity of their data. They highlight the importance of balancing local and global policies to grant domains the autonomy to work in their area of expertise. Finally, they hint at the opportunity of cross-organizational data product exchange as devised by this Thesis.

Goedegebuure et al. [25] conducted a comprehensive gray literature review to provide insight into practitioners' point of views concerning the four core principles outlined in 2.4. They, too, emphasize the need for striking a balance between centralization and decentralization in a federated governance model. They propose a federated governance team composed of *i.a.* company-domain representatives, subject matter experts and employees from the legal department that perform numerous activities to ensure interoperability of data products and compliance to company-wide quality standards. One such responsibility includes central identity and access management to maintain data security. This approach, however, constraints information sharing to company internal stakeholders only, or would create administrative overhead in the management of digital identities for external parties with access to the data mesh. A further limitation of their grey literature review concerns itself with the lack of existing resources with respect to automation in administering policies developed by the federated governance team.

Damjanovic-Behrendt and Behrendt [17] define governance mechanisms and factors for digital platforms that form an information sharing ecosystem aiming to create an environment for sustainable collaboration. They classify onboarding of new participants as one of the key challenges and describe three governance mechanisms that include a *laissez-faire*, legislatory and community driven approach. The first governance paradigm relies

on each member governing itself, while the others build on either legislation or common values and norms. Their goal is to derive a governance model that allows for the interoperability of digital platforms while overcoming the complexity of a multi-sided digital platform. As further work, they mention autonomous governance paradigms that *i.a.* build on blockchain technology and artificial intelligence.

2.6.2 Decentralized Identitifiers and Verifiable Credentials

As identified in 2.4, identity federation is a key requirement of a decentralized data exchange network that needs a solution that is enforced globally. DIDs and VCs have been proposed for use cases in managing and scaling digital identity solutions; resolving DIDs to a DID document containing pseudonymous information about a digital identity can build the foundation of user authentication and authorization. Brunner et al. [12] broadly depicted the features of DIDs, VCs, and SSI and outlined an exemplified process for issuing, sharing, and verifying a VC.

Further work developed guidelines for using DIDs and VCs [5]. As key challenges, Avellaneda et al. identify the development of best practices and recommendations to address the full life cycle of a digital identity; *e.g.* a user might lose control over an identifier, digital key, or cell phone as a whole.

Verifiable credentials have been leveraged in enterprise use cases; the OpenID foundation whose mission is to create identity standards, has published a white paper in June 2022. Yasuda et al. report about an employee onboarding process for the government of Queensland, Australia, whose throughput time could be reduced from five working days to an hour [70].

Rather than focusing on foundational principles and theoretical frameworks, Bolte [11] conducted a comprehensive, developer-focused evaluation of verifiable credential Software Development Kits (SDK).

Various digital identity solutions based on DIDs and VCs are available. To become a viable solution, it must support issuing, storing, presenting, verifying, and deleting verifiable credentials. The shortlist relevant for this Thesis includes solutions from Microsoft, Mattr Limited, and Hyperleger Foundation. Microsoft EntraID provides a VC solution that supports the aforementioned five fundamentally required functionalities. Cloud administrators of the solution can configure the *Verified ID* service to issue VCs to its mobile client *Microsoft Authenticator*; credential revocation is supported through the implementation of W3C *bitstring status lists* [42] and the mobile client is capable of storing and presenting VCs. Furthermore, it offers features to verify VCs originating from any issuer using the verified ID service [29–31]. Mattr Limited offers its services through its digital identity platform; users of their service can issue VCs in the JSON-LD format. They also offer features to store, present and validate VCs. Similarly to Microsoft's solution, revocation of credentials is achieved through revocation lists. Issuers record lists of active credentials and to prevent them from tracking where their credentials are used, verifiers authenticate the validity of a VC by requesting a list of credential status [43, 46, 47]. Hyperledger, on the other side, approaches revocation through zero-knowledge proofs

(ZKP); ZKP is a cryptographic method where a prover can convince a verifier of the truth of a statement without revealing any information beyond the validity of the statement itself [1]. In essence, credential holders can construct ZKPs proving the validity of their credential without revealing a correlative identifier [10, 27, 37].

Building on the aforementioned concepts and ideas, we contribute towards a data and self-serve platform by presenting a proof of concept for federated identity management based on DIDs and VCs. Furthermore, we encode and enforce global joiner policies through Ethereum smart contracts to demonstrate a model for interoperability between various domains and organizations.

Chapter 3

Design

Beginning with Section 3.1, we describe the foundational design principles that guide the development of the envisioned Trust Coalition. Section 3.2 introduces various user types that potentially benefit from the developed solution. The subsequent Section 3.3 details requirements for credential issuers and verifiers. Thirdly, Section 3.4 addresses the life cycle of VCs. The fourth Section 3.5 describes how VCs can be leveraged to authenticate and authorize users. Lastly, Section 3.6 documents the reason that lead to the decision of the selected Tools Hyperledger Aries and Indy. Figure 3.1 presents all involved parties of the envisioned network alongside a summary of their excepted capabilities.

3.1 Design Principles

The governance of a decentralized data architecture with numerous independent nodes requires a balance between local and global policies that tackle the enforcement of user authentication and authorization. To govern a coalition of companies that share data, there must be agreed-upon standardization regarding digital identities. More so, there needs to be a process tasked to establish trust between digital domains; companies need to be able to join and leave the network alongside their employees.

The subsequent design principles will drive the implementation of the prototype. The barriers to entry should be low to drive the adoption of the proposed solution. Operators and users require an intuitive onboarding, lest they require extensive familiarity with the decentralized identity paradigm. Prospects are expected to operate their webservers and have basic knowledge of software-containerization and the command line interface. On the other hand, users are assumed to be familiar with a modern smartphone, willing to install a mobile wallet and secure their wallet, recording their digital identities with a strong PIN or leveraging their devices' biometric authentication mechanisms.

The aforementioned identities shall prioritize user-centricity; credential holders shall be the sole owners of their data, while issuers retain the capability to revoke and invalidate digital credentials. Secondly, the solution must support the selective disclosure of one of more credential attributes.

Finally, given two disjunct subsets of credential attributes, there must not be an identifier such that a correlation between the two subsets can be drawn. This characteristic of privacy preservation enhances end-user privacy; *e.g.* when credential holders reveal their first and last name in two separate credential presentations, a verifier must not be able to construct a holder's full name.

A uniform life cycle for the management of credentials also falls within the responsibility of global governance. Essentially, we plan to exchange digital identity information by enforcing robust security controls concerning the confidentiality and integrity of shared information. Issuers are required to register their DID alongside a credential definition on the DLT, as depicted in the first step of Figure 3.1. Subsequently, a domain can request membership through the coalition DAO, they start an application process to record the domain's issuer credential definition on a decentralized whitelist.

Provided that the onboarding process passes through, existing members now trust the domain's issued VCs, and it becomes part of the trust ecosystem. All TC members can look up other members and validate their issued VCs.

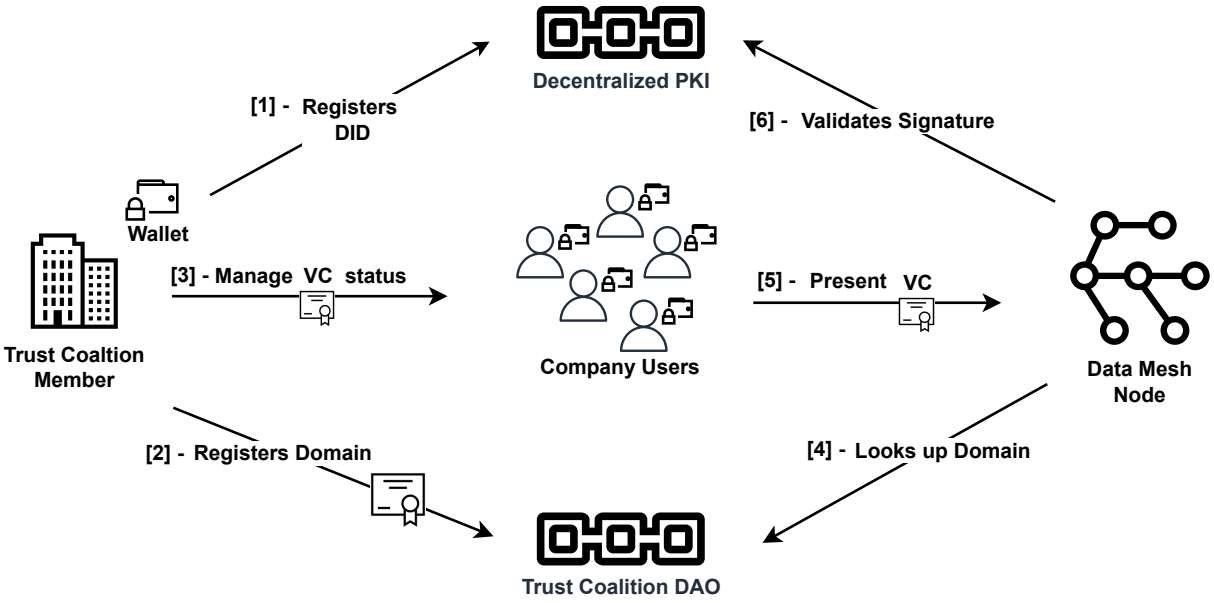

Figure 3.1: High-level architecture of the Trust Coalition network

3.2 Stakeholders

Pohl and Rupp [58] define stakeholders as groups of people or organizations that directly influence system requirements. A Project that neglects them often incurs difficulties throughout the project in the form of requirement changes during the later stages that tag along with additional and high integration costs. As primary stakeholders, we identified the roles of chief information security officers (CISO), computer science professors alongside their research group members, and computer security incident response teams (CSIRT).

According to UZH's role description of a CISO [73] their responsibilities include *i.a.* the development of measures aimed at protecting the confidentiality, integrity, and availability of information and data - in other words, they are tasked with ensuring the communication security goals as outlined in Section 2.1. The CISO's subordinates consist of various knowledge workers who perform actions that often require elevated privileges, like overhauling a domain's network firewall configurations, monitoring real-time network traffic, and analyzing it for intrusion detection. For a concrete incident, actions performed with sufficient privileges may require continuous monitoring to comply with auditing standards.

Professors with their working groups,, on the other hand, may require information exchange with various parties of the same and external domains to conduct cutting-edge research into communication mechanisms for *i.a.* the security and economic viability of highly decentralized systems. One such branch of research may be the development of innovative technologies for DDoS attack mitigation; researchers require read access and possibly write entitlements for their daily activities from source systems outside their university's domain.

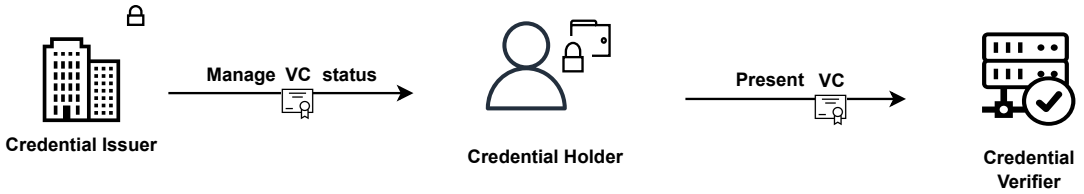

Figure 3.2: Stakeholder roles in the Trust Coalition

Given numerous stakeholders from aforementioned groups, the CIA communication security goals need to be upheld, and each stakeholder assumes either the role of a credential issuer, holder, or verifier; Figure 3.2 consolidates the common requirements by the representatives of the stakeholder roles. A trusted issuer must be capable of issuing a VC to a holder. Subsequently, issuers need the capability of maintaining VCs for lifecycle events such as address changes or termination of employment. In other words, each issuer must retain the right to credential revocation and invalidation for the credentials issued by them. Verifiers require functionality that allows them to verify the authenticity of a VC. Credential holders, however, require secure storage for their VCs to prove the ownership and validity of a VC. Based on the requirements of the three roles, trusted issuer, holder, and verifier, we develop use cases addressing their needs through TC membership and VC life cycle processes in sections 3.3 and 3.4, respectively.

3.3 Membership Life Cycle

Running an instance of the prototype is a prerequisite for becoming a coalition member. To protect the integrity of the Trust Coalition, processes to prevent impersonation need to be established, as each member can issue and validate a VC, and thus, domain applications should be subject to high-security assurance levels, *i. e.* we want to ensure that only validated and reputable domains may join the network.

There are two verification processes; based on the domain's TLS certificate policy, a voting round for existing members is started. Provided the domain possesses an Extended Validation (EV) or Organization Validated (OV) TLS certificate, its application is directly approved once a validation challenge has passed. To enforce this requirement, we will leverage the existing PKI standard *X.509*. In Figure 3.3, the onboarding process is included in the federated governance of the TC memberships. Further, each TC member manages their DIDs in a trusted and decentralized PKI.

Since the proposed solution should serve as a starting point for various data products, we expect and, therefore, need to accommodate a heterogeneous environment concerning identity management process maturity levels. Some companies may possess an IdP, while others maintain an Excel sheet with employee information. Thus, we provide means for the interoperability of digital identities by relying on DIDs and VCs. A PKI based on DLT is a trusted registry recording the public key material and further DID information. The credential ownership, integrity, and validity are cryptographically verifiable through usage of the public key material of the published DIDs.

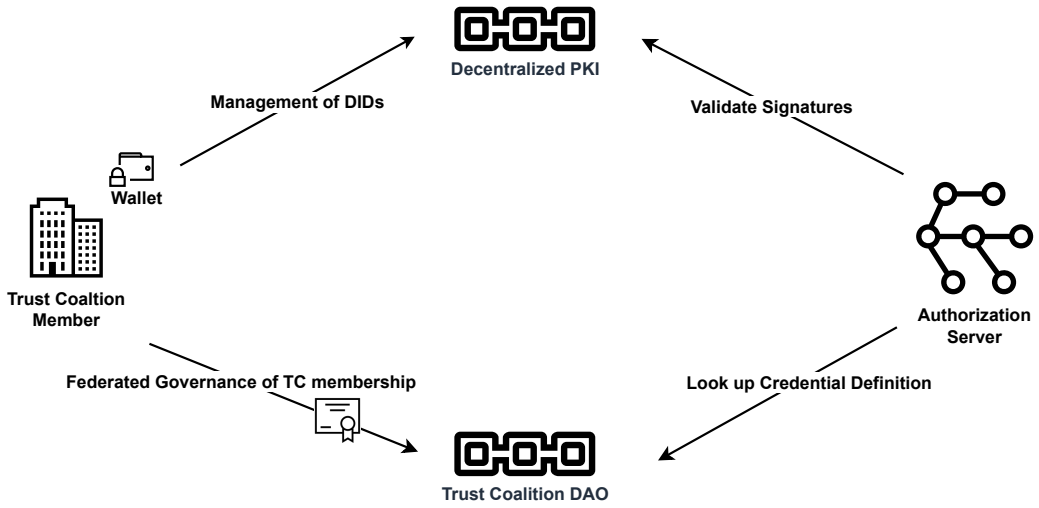

Figure 3.3: Digital Trust Framework for the TC

When a user of a different domain wants to access data, they must provide information about the domain that has issued their VCs. Verifiers can then validate whether the provided domain is part of the TC by requesting up-to-date membership information; concretely, they verify whether a domain has published a credential definition to the DAO, as shown in the step *lookup credential definition* in illustration 3.3.

After receiving a proof presentation, the verifier can authenticate the data by validating the signature of the VC with the public key material published by the issuer, as depicted in the step *validate the signature*. Ultimately, there must be a mechanism for removing companies from the Trust Coalition - granting access to resources is still at the discretion of the data owners and is thus dependent on local controls, *i.e.* each node can apply its own access policies. Check Appendix A for a complete list of the use cases.

3.4 Verifiable Credentials Life Cycle

One core principle shaping the treatment of identity information is user-centricity. To adhere to this guideline, users must be the data owners of their digital credentials. Further, presenting subsets of the owned credential sets must be supported; such presentations shall be done in a privacy-preserving manner. Verifiers, on the other hand, require the public cryptographic key material associated with the credential issuer.

A valid credential is based upon a credential definition; essentially, the onboarding processes for DID management and TC membership have been completed, as shown in Figure 3.3. This credential definition *i.a.* records its attributes and publishes the public key required to determine an issued attribute's authenticity. Once a trust relationship towards the coalition has been established, members manage their employee's VCs through issuance and revocation events that change the status of a VC through a GUI.

In the event of a new hire, a node operator needs to establish a connection to the employee's mobile wallet and issue the VC based on a predefined schema. Provided a reputable TC member, *i.e.* they exercise security controls aimed at stopping identity theft, their credentials are classified as valid. Users can reveal their identity and permissions by answering proof presentation requests whenever an action requires authenticated and authorized requests.

Proof verifiers validate whether the provided information is of integrity. They compare the signature of the proof and check if the VC is still valid. If employees resign, their VCs cannot be accepted after their last day of work, lest their VCs become a security vulnerability. In such cases, issuers must be able to revoke an issued VC within the GUI. Revocation of a VC will result in an invalid signature in step six of the graphic 3.1. After this last verification step, each application can authorize requests based on context-specific access controls. Further, the VCs must be editable to accommodate changes in an address or last name; this can be accomplished by revoking the now incorrect VC and issuing a new one to the employee. *Cf.* A of the appendix for the derived use cases concerning the life cycle of VCs.

3.5 Access Control

Applications serving data products to members of the Trust Coalition require the ability to authenticate and authorize employees from other coalition members. Concrete policies thereof depend on the domains' specific IT strategies. Such access control policies may be administered externally or inside the application; regardless of the policy administration point, before authorizing a request, the requester must be authenticated based on one to three of the authentication factors as defined in 2.1.1.

By implementing the authentication and authorization flow depicted in figure 3.4, data mesh nodes can provide access to information to any employee from the other members

while ensuring confidentiality in the wider sense. Through the prototype, user authentication shall require two authentication factors. VCs are possession-based factors, while access to them requires either biometric or knowledge-based authentication.

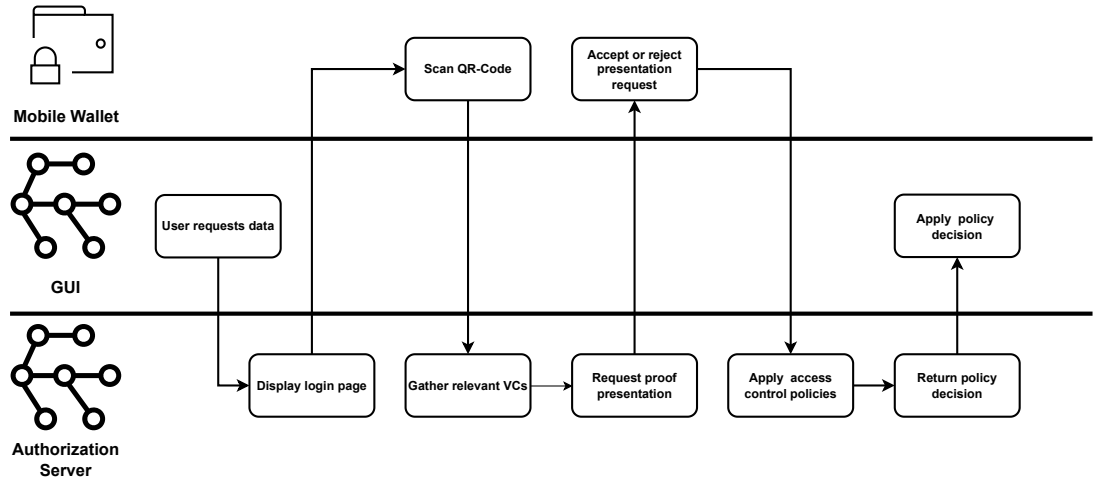

Figure 3.4: Flow to authenticate and authorize a requester

An application providing data only intended to authorized users can either delegate authentication and authorization or implement their custom access control policies and replace the authorization server in Figure 3.4 with a custom module. Before access to data can be granted, the user must pass through the access control flow.

As a starting point, unauthenticated users who request protected resources are met with a redirect to an authorization server. As depicted in figure 3.2, this node acts as a credential verifier. On the login page, a QR code intended to be scanned with the user's mobile wallet establishes a secure connection to the authorization server. The server gathers the set of VCs required to retrieve the protected resources. It requests a proof presentation through the tunnel to the user's mobile wallet, which prompts the user to consent or dissent to sharing the requested credentials. Federated governance of TC memberships enables these servers to distinguish between trusted and illegitimate issuers of VCs. If the user agrees to share the VCs entailing the relevant identity information and potential user roles, authorization can be based on any access control model. As a final step, the end-user application enforces the decision reached by the authorization server. Such a standardized approach to user authentication based on verifiable credentials can be integrated into existing infrastructure, resulting in minimal impact. Applications maintaining their list of users can securely authenticate users through portable digital identities. In essence, through federated governance of the Trust Coalition memberships, individual members need not establish identity federation relationships with each other; rather, they collectively regulate the openness of their data ecosystem.

3.6 Tools Decision

This sections serves as rationale behind selecting *Hyperledger Aries* [1] as the toolkit for decentralized identity that interacts with an instance of the *Hyperledger Indy* [2] blockchain; table 3.1 provides an overview of the criteria used for the tool selection. In section 2.6, we already established that all three solutions offer functionalities to issue, store, present, verify, and delete verifiable credentials.

In previous sections, we established the ground principles to which the final prototype must adhere: some of which are user-centricity, in other words, users are the data owners, privacy preservation, and selective disclosure to share only the necessary credentials.

Criteria	**Aries**	**Mattr**	**Verified ID**
User is data owner	●	●	●
Selective Disclosure	●	●	●
Privacy Preservation	●	◐	◐

● = functionality provided, ◐ = functionality partially provided

Table 3.1: Decentralized Identity Solutions evaluated based on our design principles

Mattr and Hyperledger support various standards. Amongst them is the Internet Engineering Task Force (IETF) proposed standard BBS Signatures Suite that supports selective disclosure of credential attributes [24, 44]. Through adherence to the Verifiable Credentials Data Model [63] from the standard body World Wide Web Consortium (W3C), Microsoft's solution, too, supports selective disclosure of VCs [28]. Also, all three solutions act according to the principle of user-centricity; for each provider, mobile applications are available in Apple's Appstore and Google's Play Store [9, 32, 45].

Both solutions of the software vendors Microsoft Inc. and Mattr Ltd. handle revocation through comparable privacy mechanisms. Verified ID supports revocation through the implementation of the W3C standard Bitstring Status List [28], while Mattr supports the specification Revocation List 2020 [44]; their approach defines privacy preservation as preventing the issuer from knowing when their issued credentials have been verified. To achieve this privacy preservation in the looser sense, issuers store a list of boolean values where the index of the boolean corresponds to the status of a credential. Rather than retrieving a single value from the issuer, verifiers request the whole or part of the revocation registry and look up the status of a presented VC themselves [41, 42].

Hyperledger, on the other hand, argues that the aforementioned approach does not truly preserve the privacy of a credential holder, as revocation lists allow credentials to be correlated to their presenter. Rather than revealing information that can be correlated to the presenter, they opt to implement a ZKP-based revocation mechanism. A cryptographic accumulator records a large number that is composed of the identifiers of active VCs; if a holder's VCs are active and thus part of the accumulator value, they can construct a ZKP

[1] Hyperledger Aries
[2] Hyperledger Indy

for a verifier without revealing correlatable information while proofing that their credential has not been revoked [27]. This approach fits the definition of privacy preservation adopted by this Master Thesis more precisely.

Further, only the solutions provided by Hyperledger are available under an open-source license; direct access to the tools and algorithms used in this Thesis aims to ensure replication of the work. Recapulatory, we evaluated three decentralized identity solutions, two of which are closed-source solutions and have a laxer notion of privacy preservation compared to the open-source alternatives offered by Hyperledger. These two facts have primarily, have led to the selection of the Hyperledger decentralized identity suite as the fundamental tool for the prototype.

Chapter 4

Implementation

In this chapter, we explore the core concepts of implementing the Trust Coalition Agent (TCA) to evaluate the proposed system later. The prototype comprises four core services: connection, verifiable credentials, verification, and governance services, as illustrated in figure 4.1.

The backend infrastructure receives API calls from all services; an NGINX reverse proxy reroutes requests to Hyperledger's Aries framework Aries Cloud Agent Python (ACApy) to *i.a.* establish DIDComm [1] connections and verify proof presentations.

Since the web client acts as a VC issuer, it requires backend functionality to manage the VCs, and thus, it sends requests to ACApy and the VC lifecycle-API. Finally, to become a verified issuer and participate in the coalition's governance, the correct smart contracts must be invoked, and the client covers such requirements with the governance service.

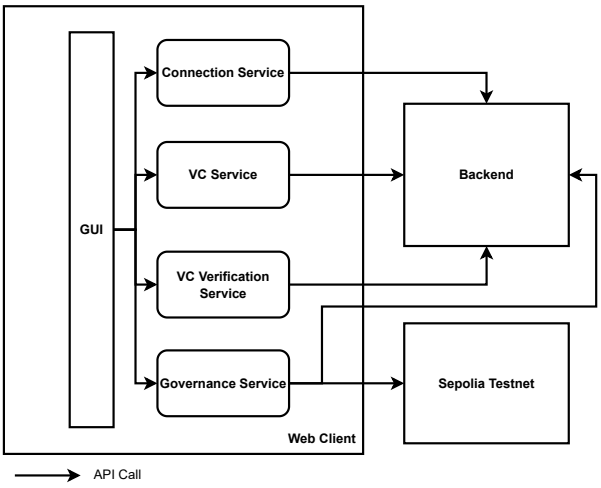

Figure 4.1: Architecture of the Angular client application

[1] Aries RFC 0005: DID Communication

4.1 Connection Service

Users can connect to other agents and agent-like things based on DIDComm through a connection service. The DIDComm protocol attempts to establish secure peer-to-peer connections for encrypted and verifiable information exchange. The prototype offers two ways of establishing a connection in step 1 that results either in the branches *a* and *b* to establish a DIDComm connection in the concluding step 3 of Figure 4.2.

To establish a connection, there are always two roles involved: *inviter* and *invitee*; in the version *a* of the protocol to establish a connection, the mobile wallet assumes the role of invitee. The inviter initiates a connection by creating a connection invitation. At this point, the inviter must have already created and published its DID alongside public key material and information about reachable endpoints to interact with the agent. Irrespective of the means by which an invitation has been transferred, an invitee must initiate a connection request; in a scenario where the invitee possesses a mobile wallet, scanning a QR-Code is a viable option, as denoted in step 2a in the edge layer.

The invitee sends relevant information to the inviter in a connection request message; provided no errors occur, the inviter technically completes the flow with a connection response message. Supplementary, invitees should send a acknowledgment message to the inviter [8]. Issuing and verifying VCs can happen through a secure and private peer-to-peer channel.

The webclient of the prototype creates connection request and response messages through the REST-interface of ACApy, which in turn delegates functionality requiring the DID layer to a Hyperledger Indy instance, more precisely a ledger hosted by the British Columbia Government BCovrin [2].

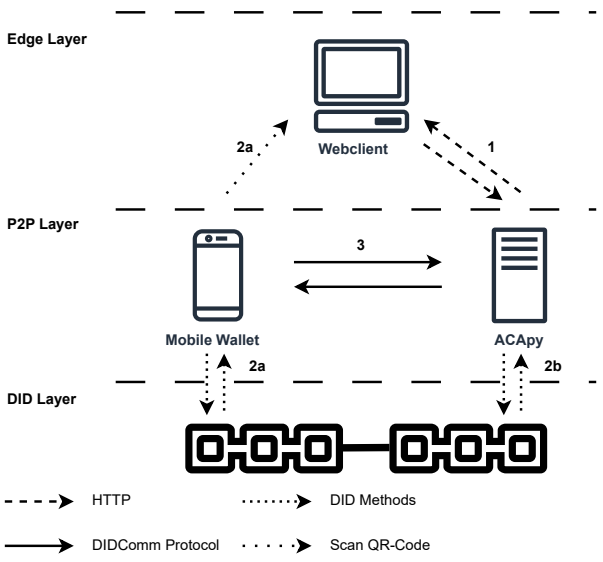

Figure 4.2: Flow to establish a DIDComm connection

[2]BCovrin Test Indy Network

4.2 Verifiable Credentials Service

Hyperledger defines a standard protocol for issuing verifiable credentials as a basis for interoperability. By default, two roles are involved; the credential issuer and holder communicate through a previously established DIDComm connection (as depicted in Figure 4.2). The features of the verifiable credentials service encompass issuing a VC, revoking and verifying it. Within the standard protocol, a total of five messages are exchanged. However, for this thesis, only three are relevant: the offer-credential, request-credential, and issue-credential messages [37]. On behalf of ACApy, credential definitions need to be registered on Hyperledger Indy so that users can subsequently issue VCs based on these schemas. As a starting point of the credential issuance process, issuers send a credential offer to potential VC holders; provided they want to accept the credentials, they send a credential request message to the issuer. The VCs are issued to and stored in the holders' wallets in the final mandatory message. To revoke verifiable credentials, issuers must specify a web resource, a tails server hosting a tails file, that can be reached from the Internet in the credential definition. A credential definition essentially includes all necessary information to assess the validity and integrity of an issued VC; *i.e.* they list the public key that can be utilized to verify the signature of a proof presentation and provide information about how revocation is handled. When users want to issue VCs to another agent, they select one of the predefined schemas and complete the generated form with accurate information. They can then offer the credentials by submitting the form; Table 4.1 lists all schema attributes used to authenticate web client users. Once the offer has been accepted in the mobile wallet, the credentials are exchanged without any further user interaction.

Field Name	**Purpose**	**Example**
First Name	Application user's first name	Alice
Last Name	Application user's last name	Doe
EmployeeId	Application user's identification number that is used as foreign key	a1b2-c3d4e5-f6
Role	Application user's role is either *admin* or *operator*	admin

Table 4.1: VC Schema of Web Client User

Records of the issued credentials must be kept to support the requirements concerning the life cycle of VCs. Through API requests to the backend infrastructure, concretely the lifecycle-api, users can view issued credentials and revoke them. The lifecycle-api offers users access to a PostgreSQL database for that purpose. The endpoint responsible for serving the issued VCs expects a valid authorization token within the request header; provided that an authorized user has logged in within the last hour of the request, a list of tiles with the issued credentials alongside their validity is rendered. Holders of VCs that are revokable must create a proof of non-revocation that leverages aforementioned tails file; when their credentials have been revoked, they are unable to construct a valid proof. The revocation of a VC results in an update in the tails file, which can be done through the GUI in the VC service section. If an erroneous credential has been issued, the mishap can be corrected by revoking the faulty VC and issuing a new credential. Finally,

each issuer possesses the necessary features to be a credential holder, and other issuers can create VCs and send them through a secure connection to our agent.

4.3 Verification Service

In a proof presentation flow to *e.g.* authenticate an application user, two parties are involved: verifier and proofer. A verification service has been developed; it illustrates the proof presentation flow by enabling users to walk through the protocol manually. The issuer begins the flow by sending a request-presentation message to a prover and either receives a presentation invoking the presentation verification process or termination the flow by abandoning it [36].

Before accessing the verification service, the client application requests user identification by initiating the authentication flow. Figure 3.4 described this flow, starting by creating a new connection invitation. Subsequently, the GUI displays it in a QR code and observes changes in the state of the invitation. On the GUI, the user is asked to scan the invitation with a mobile wallet that stores the corresponding VCs. Once done, a new connection between the client application and the user's mobile wallet has been established, analogous to the flow described in section 4.1. This state change invokes an augmented version of aforementioned proof presentation flow. The GUI provides feedback on the current step of the authentication process; the first update informs about the successful establishment of the DIDComm connection, and the next required step involves answering a proof presentation request. In the background, the verifier sends a DIDComm message to the user about the goal of the request-presentation message that will subsequently follow - in this case, a proof request to login.

The web client authenticates users based on credentials issued by the ACApy instance of the backend infrastructure (*cf.* Table 4.1 for the complete schema of the VC). Therefore, it constructs a request-presentation message that includes the credential definition a VC must be based upon. Given the holder possesses a VC from the correct issuer, the mobile wallet can construct a valid proof.

To respond with a presentation message, the prover constructs two proofs: one whose data integrity can be validated and a proof of non-revocation confirming that the credential has not been revoked yet. The second proof requires the tails file, a giant random number at its core. Conceptually, each credential is assigned a unique identifier that is linked to the tails file, and to verify that a credential has not been revoked, the proofer can demonstrate that a VC is still linked to the tails file without revealing the identifier in a ZKP. Utilizing ZKP proof enables the presentation of VCs in a privacy-preserving manner.

As soon as a presentation message has been received, both aforementioned proofs are verified, and the processing is indicated on the GUI. The verifier confirms the authenticity of a VC by validating the credential signature notably, without having to contact the issuer; given a VC of integrity, the credential information, which in this case includes a user id and application role, is checked against the issuer database in the backend infrastructure.

Field Name	Purpose	Example
Given Name	Person's first name	Bob
Family Name	Person's last name	Doe
Employee Id	Unique identifier of the identity	y32s-7ksd6-ms
Department	Name of the department that employs person	IT Security

Table 4.2: VC Schema of identity standard claims

In the event of an authorized user, the lifecycle-API issues an authorization token valid for the then-coming hour. The second collection of attributes, called standard claims schema, consists of four attributes listed in Table 4.2. Based on these attributes, access controls can be developed; *e.g.* rule-based access control models can leverage the values present in the VCs. As previously mentioned, each issuer registers its credential definition on a trusted ledger and provides information about its credential definitions to verifiers through the DAO. Thus, each organization has a unique identifier within the scope of the trust coalition network.

Based on these identifiers, each domain can apply their access control policies; such an approach allows for adjustable and extensible access control models. A cross-domain role model becomes imaginable where selective disclosure becomes crucial; in a scenario where maintenance of specialized machinery is mandated to a service provider, access to the physical location of the machinery can be granted based on a collection of VCs. Rather than sending all attributes of a VC when at least one attribute is required, selective disclosure allows the creation of proof presentations based on the individual attributes of all VCs a holder possesses. One VC would attest the employment and possibly employee certificates; the service provider issues this VC. The service recipients can then issue and manage the VCs granting access to their infrastructure.

Within the section of the verification service, the authenticity of previous proofs can be viewed. Each historic proof records *i.a.* a timestamp at which the presentation request was received, the web client's role in the proof presentation flow, and what connection has been involved. Each proof lists the schema, name, value, and status of the revealed attributes in the proof presentation message. Attributes tampered with or revoked receive a status of *inactive*. To summarize, through sending a request-presentation from one agent to another, a verification flow can be invoked that allows *i.a.* IoT-devices and webservers to authenticate holders of VCs with their explicit consent in a privacy-preserving manner without contacting the issuer of the VC.

4.4 Coalition Governance

Participation in the coalition's governance occurs by casting votes to either support or oppose the admission of a prospect. As established in the previous Section 4.2, issuers publish credential definitions that include publicly reachable endpoints, the location of their tails file, and public key material. Traversing the onboarding process of the TC membership life cycle means a domain registers its VC issuer so other members accept

Field Name	**Purpose**	**Example**
Company Name	Informs about the name of the domain	ACME INC.
Company Domain	Publicly reachable website of the domain	itsec.acme.com
Contact Person Lastname	Informs about the last name of the domain contact person.	Faber
Contact Person Email	Informs about the email of the domain contact person.	af@itsec.acme.com
Credential Definition	Identifier of credential definition used by the issuer	-
Wallet Address	Records a 42-character hexadecimal address of an Ethereum wallet	0x5f2...ea9

Table 4.3: Domain Joiner Form

their issued VCs. Table 4.3 contains an extract of the online form a prospecting domain must complete.

The values of the fields *Company Domain*, *Credential Definition*, and *Wallet address* serve as pivotal identifiers that pair the opportunities offered by the decentralized identity paradigm with time-tested protocols utilizing the X.509 PKI standard and incorporate a state-of-the-art governance model - a decentralized autonomous organization.

A Solidity smart contract is deployed on the Sepolia Testnet to standardize the onboarding process; every user can fill out aforementioned form to then invoke a smart contract that adds a prospect to the joiner list. For every domain on the joiner list, an existing member receives a voting right to reject or endorse an active application. An application becomes inactive if it has not reached a predefined threshold of favorable votes within one week of the application time. When most existing members support an application, the application's domain information is added to the coalition's membership list. All previously mentioned services communicated with the backend infrastructure. So does the governance services; additionally, it enables invoking Solidity smart contracts on the Sepolia Testnet. Each application starts the same by completing aforementioned joiner form. When the field *Company Domain* has been filled out, the webclient relays this information to the lifecycle-api and expects a response to whether or not the domain is reputable. All domains possessing an X.509 certificate with the certificate policy set to EV or OV are classified as reputable. For reputable domains, a second and faster onboarding process exists. Figure 4.3 illustrates the accelerated onboarding process that includes a domain ownership challenge response flow. Only the domain owner of a reputable TLS certificate can complete said flow.

After clicking a button to start the accelerated onboarding process, the user must invoke a smart contract through the GUI. Before starting this process, this presumes the user has set up an Ethereum wallet. A successful transaction results in the domain being added to the active accelerated joiner requests. After that, the webclient sends a request to the lifecycle-api to generate a cipher text whose plain text can be used to complete the accelerated onboarding process. After checking that the domain has been applied to become a member, the lifecycle-api requests a TLS certificate from the provided domain and extracts the public key material. As a next step, the lifecycle-api generates a random

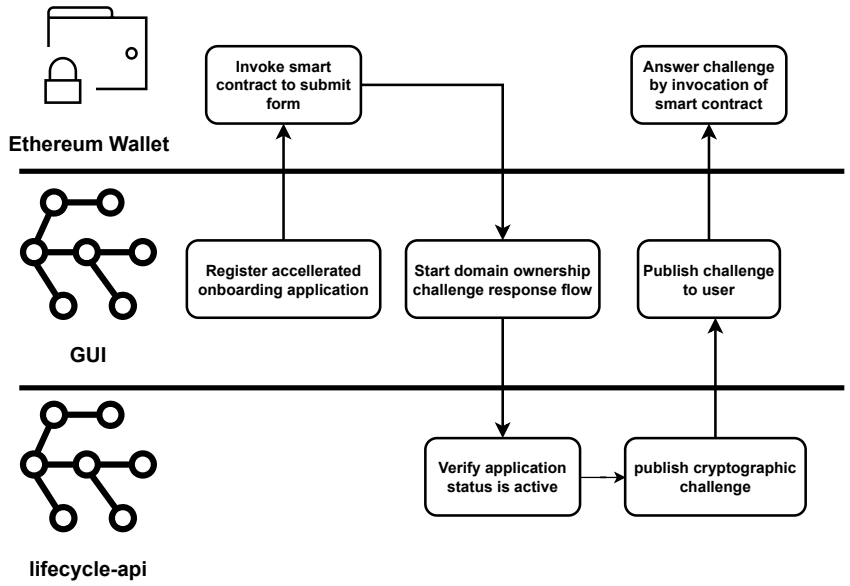

Figure 4.3: Domain ownership challenge response flow

128-bit mnemonic phrase and encrypts it; depending on the key type, a symmetric or public key encryption scheme is utilized. The lifecycle-api then invokes a smart contract through which it publishes the challenge. The smart contract excerpt of listing 4.1 expects as inputs a domain name and secret passphrase from the lifecycle-api, which needs to originate from the wallet with which the smart contract has been deployed. For attacks aimed at emptying the API's wallet, an adversary would require more funds than the defender since publishing a challenge requires substantially fewer computational steps than an attacker's applications would, resulting in higher costs.

```
contract CredentialDAO {
    ...
    mapping(string => string) private challenges;
    ...
    function publishChallenge(
        string memory domain,
        string memory passphrase
    ) public {
        require(
            msg.sender == chairperson,
            "not allowed"
        );
        require(
            bytes(
                acceleratedJoinerRequests[domain]
                    .applier.companyName
                ).length != 0,
            "not registered"
        );
        challenges[domain] = passphrase;
    }
    ...
}
```

Listing 4.1: Method to publish a challenge for the accelerated onboarding process

The domain names of the applicants serve as keys for a Solidity mapping pointing towards the mnemonic phrases generated by the lifecycle-api. Provided the applicant is the owner of the TLS certificate of the domain, they can decrypt the cipher text and submit it within the intended form field. The challenge is served in a form that includes the cipher text alongside the public key material involved in generating the cipher text.

As described in the first Subsection Confidentiality 2.1.1 of IT Security 2.1, the goal behind authenticating a user of a system, in this case, the web client, is retrieving a piece of information that uniquely identifies said user in the system and possession based factors can serve such a purpose. Concretely, to mitigate domain impersonation, a prospect must prove possession of the private key associated with the domain that has passed a vetting process to attain an EV or OV TLS certificate. Listing 4.2 reveals a snippet of a Solidity smart contract method responsible for onboarding a new user within the accelerated process.

The function expects two strings as input: the domain for which the answer is submitted and the answer itself. Given that the mapping of domain names to passphrases includes the provided domain, it compares the passphrase generated by the lifecycle-api. If this condition is also met, the smart contract retrieves the application from the corresponding mapping and marks it as accepted and inactive. The application is subsequently added to two members' mappings; the first allows the query of a domain by the wallet address. This mapping is essential in validating that a member of the TC has invoked a smart contract method.

The second mapping organizes the identical information by domain name. Essentially, this trust network is built on three pillars that leverage asymmetric, also known as public, key cryptography. Verifiable credentials heavily rely on numerous private and public key pairs; part of the DID Document contains the public key material. As the full name of PKI suggests, it is a collection of public keys and establishes trust on the Internet using an X.509 certificate. Finally, Ethereum wallet addresses are directly derived from a public key whose private counterpart is used in invoking smart contracts and participating in the governance of the trust coalition

```
1  contract CredentialDAO {
2      mapping(address => Company) private members;
3      mapping(string => Company) private membersByDomain;
4      mapping(string => string) private challenges;
5      mapping(string => Application) private acceleratedJoinerApplication;
6      ...
7      function submitChallengeAnswer(
8                  string memory domain,
9                  string memory answer
10             ) public {
11         require(
12             bytes(challenges[domain]).length != 0,
13             "not allowed"
14         );
15         string memory passphrase = challenges[domain];
16         require(equalStrings(passphrase, answer), "incorrect");
17         Application memory acceleratedJoinerApplication =
18                 acceleratedJoinerRequests[domain];
19
```

```
20          acceleratedJoinerApplication.accepted = true;
21          acceleratedJoinerApplication.active = false;
22          Company memory applicant =
23              acceleratedJoinerApplication.applier;
24          members[applicant.walletAddress] = applicant;
25          membersByDomain[applicant.companyDomain] = applicant;
26      }
27      ...
28  }
```

Listing 4.2: Solidity smart contract method to submit answer to challenge for accelerated onbaording process

Whenever an authorization server is tasked with authenticating a user, it assumes the role of a verifier. It must collect information from a trusted source to construct a presentation-request message. Retrieving the right information can be achieved through home-realm discovery, where users provide the domain name for which their VC issuer owns the credential definition, provided that the domain is part of the trust coalition. The verifier can consequently initiate the flow to authenticate and authorize a user (*cf.* Figure 3.4). In essence, to know what VC to accept, verifiers of VCs can resort to a trusted data source whose contents are maintained through federated governance.

4.5 TCA Life Cycle API

The lifecycle-api is built using Python; more precisely, the FastAPI library exposes 13 endpoints from four categories and is part of the backend infrastructure as noted in Figure 4.1. This category covers use cases to authenticate and authorize the web client user.

Two factors are required in the authentication and authorization flow as denoted in figure 3.4; a verifiable credential is a possession-based authentication factor. Access to said knowledge-based or biometric authentication factors guards VC. Once the VC has been authenticated, the API looks up a credential identifier and role revealed in the proof presentation. The lifecycle-api then additionally verifies whether a user with presented credential identifier exists in its database; given there is a match, it reviews if it has been issued with the correct role required for the intended action.

Each request to a protected resource expects a valid authorization token in the request's header. The authorization token comes as a JSON web token (JWT) that entails the employee ID and the role associated with the account. JWTs are user claims encoded in a simple security token format; in its payload, JWTs store a set of claims that typically includes information that allows verifiers to authenticate their users. These JWTs can be digitally signed and encrypted to uphold the integrity and confidentiality of the token credentials. In the authentication implementation, a JWT is valid for 60 minutes before users need to retrieve a new authorization token by traversing the authentication flow.

Further, the lifecycle-api is key in maintaining TC memberships; a module is responsible for the accelerated onboarding process and checking a domain reputation. One route takes as a request body one attribute with a domain name to request the domain's TLS

certificate in a later step and verify the certificate policy's value. Provided an EV or OV TLS certificate policy is encountered, the endpoint returns *true* in all other cases *false* is returned. The second endpoint of this category generates and registers the challenges that need to be overcome in the accelerated onboarding process as described in figure 4.3. Domain owners can prove the ownership of a domain by decrypting cipher text that has been encrypted with the help of the public key material found in the TLS certificate.

Whenever a reputable domain possesses a TLS certificate whose public key utilizes the Rivest-Shamir-Adlemann (RSA) algorithm, the plain text is encrypted with the encountered public key. This assures that only the domain owner can submit the plain text as a response to the challenge. Alternatively, TLS certificates can possess a public key that uses an elliptic curve (EC) algorithm. One fundamental difference between the two algorithms is how they handle keys; RSA utilizes asymmetric keys where only one part must be kept secret, while EC requires a secret key shared between the encryptor and decryptor. In a setting where secret key cryptography is used, the counterparties need a reliable way of exchanging such a secret key; Figure 4.4 visualizes the collaborative derivation of a shared key.

In the exchange, there are two parties herein termed Alice and Bob for this illustration. Alice's keys are present on the left side of the illustration; Bob's counterparts are on the right. To initiate the key exchange, Alice and Bob agree upon a start value represented by the green key with the hexadecimal color code *#4D8033*. The only purpose of that number is to provide a common starting point when individually building the shared key - this value can be publicly known without weakening the strength of encryption.

Alice applies her private key to the common starting value; here, her private key is represented by the purple color with code *#92006D*. The resulting key is then composed of a publicly known value, the generator value, and Alice's private key. Bob follows the same steps but replaces Alice's private key with his own. Alice and Bob then exchange the key that they have just composed - a shared key. In the final step, Alice applies her private key once more to the shared key created by Bob and ends up with the value *#60356A* representing the shared secret key in this model. Bob can, analogously, create the shared secret key by applying his private key to the shared key sent by Alice.

Through this interaction, Alice and Bob can establish a communication channel to ensure that only their counterpart can de-/encrypt the messages. The proof of domain ownership is achieved through the decryption of a cipher text; provided with a certificate using an EC algorithm, we employ an elliptic curve integrated encryption scheme (ECIES) that follows the conceptual approach illustrated in figure 4.4.

The third category of endpoints revolves around the life cycle of the issued VCs, *i.e.* it provides functionality to issue, view, and revoke VCs. A PostgreSQL database records a list of all issued VCs by the ACApy instance; a credential exchange and thread ID are required to revoke VCs. Neither identifier is known at the time of issue. When a VC is issued, the connection identifier is stored alongside the proposed attributes and their values. Once the potential holder has accepted the credential offer, the state change is registered within ACApy and relayed to the lifecycle-api. It subsequently extracts the identifiers for the credential exchange and message flow. An overview of all issued credentials alongside their validity status can be retrieved through the API. Before invoking

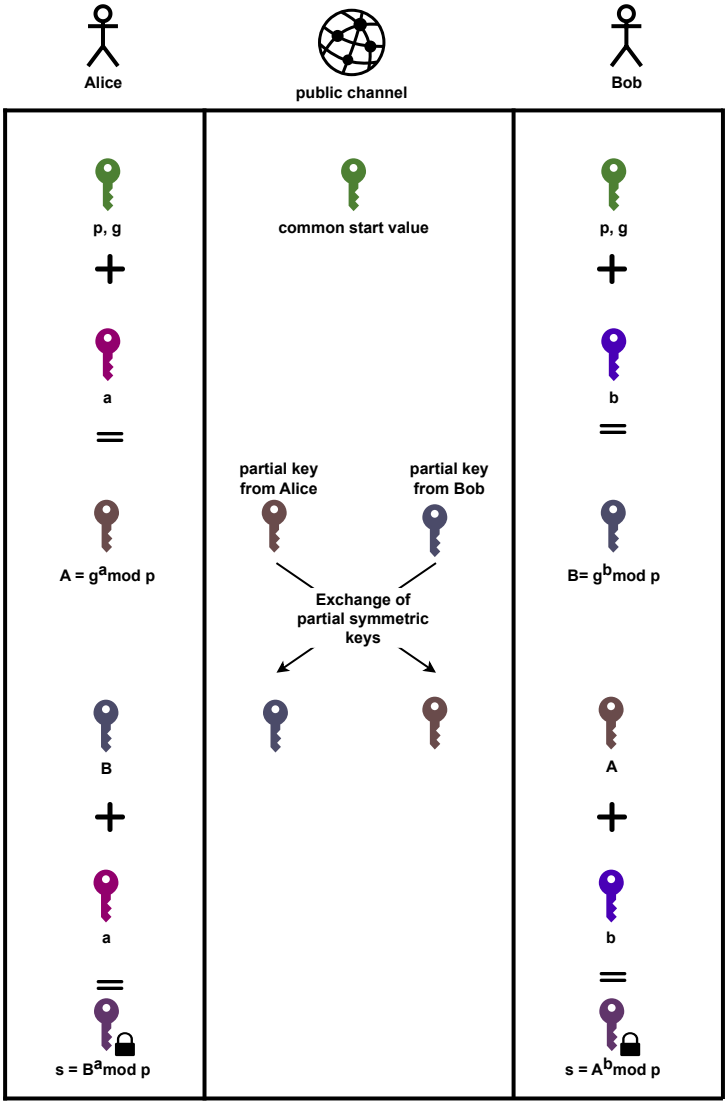

Figure 4.4: Visualization of a Diffie-Hellman key exchange to derive a shared secret

the function to query the database, the lifecycle-api verifies that the user is privileged to read all issued credentials and thus expects an authorization header in the request. This endpoint depends on a function that validates if the authorization token present in the request is authentic and still valid. Given an authorized user, the endpoint retrieves all issued credentials from the database and returns them to the user. The user obtains a reference to the credential exchange by retrieving issued VCs. In a scenario where issued VCs need to be revoked, reference to said credential exchange is a required piece of information. Whenever the lifecycle-api receives a request with a valid authorization token, the credential exchange ID updates the validity status of the associated credential attributes in the database.

Finally, whenever the API needs to react to a state change of the ACApy instance, it offers an endpoint for machine-to-machine(M2M) interaction. ACApy is configured to notify the lifecycle-api on all internal state changes (*i.e.* when a connection invitation has been accepted or a proof presentation is done). The remaining endpoints listen for these

webhook-requests.

Only one of the six M2M endpoints processes data when being called; ACApy retries webhook-requests up to five times if it cannot reach an endpoint. Thus, these five endpoints serve to minimize unnecessary errors. From a credential exchange event with the status done, *i.e.* the VC has been issued and accepted by the holder, the id of the credential exchange and message flow are extracted. Finally, the demo setup includes an interactive documentation of the lifecycle-api through Swagger UI [3].

4.6 Aries Cloud Agent Python

ACApy is a Python implementation of the Hyperledger Aries library; it is responsible for interacting with the DLT, communicating with other agents through the DIDComm protocol, and managing VCs. It possesses a command line interface (CLI) with many arguments; some of the arguments are automatically set through a managed script.

ACApy is available as a Docker image and configured as a docker-compose service as illustrated in listing 4.3. The provided bash script *./manage* builds and instantiates TCA with the commands *./manage build* and *./manage start*. Some notable parameters are *-e*, which denotes the endpoints disclosed through DID Documents; this endpoint receives all inbound messages from other agents to this ACApy instance. The argument *–tails-server-base-url* contains the URL at which the tails server is reachable and hosts the tails file to allow for the construction and validation of proofs demonstrating that a credential has not been revoked in a privacy-preserving manner.

The subsequent section will occupy itself with describing the tails server. Listed as the third argument is *–genesis-url* and determines what blockchain DIDs are written and retrieved from. A genesis file contains information about the network configuration. In the case of Hyperledger Indy, it *i.a.* includes the IP addresses and ports at which the nodes responsible for maintaining the integrity of the blockchain are reachable. One of these validator nodes needs to be contacted to publish a schema or credential definition.

Finally, the last argument we pay special attention to is *–seed*. This value generates the agent's DID; so that an agent can be started, its seed value must be registered on the ledger beforehand. Should this step be omitted, ACApy will not start and shutdown after displaying an error. The manage script consists of four features. One command builds all the necessary services, while another stops the whole software system. An agent can be started using two separate commands. The first *provisions* the agent; *i.e.* a new agent seed is generated and registered on the ledger. Further, the two standard schemas and credential definitions are also published. Recapitulatory, Hyperledger's DID framework ACApy offers extensive and flexible configuration, many of which are set by the manage script and integral to the proposed trust coalition.

```
1 services:
2   ...
3   dao-aries-agent:
```

[3] Swagger UI

```
4       image: bcgovimages/aries-cloudagent:py3.9-indy-1.16.0\_0.11.0rc2
5       command: >
6         start
7         -e ${DAO_AGENT_TUNNEL_URL}
8         --tails-server-base-url ${TAILS_SERVER_URL}
9         --genesis-url ${INDY_URL}/genesis
10        --seed ${DAO_SEED}
11        ...
12      ...
13  ...
```

Listing 4.3: Docker compose snipped of ACApy's configuration

4.7 Indy Tails Server

The final component of the backend infrastructure is the tails server; as previously mentioned, its responsibility is to maintain a web resource such that the VC holder can construct a proof of non-revocation, while verifiers require it to validate the authenticity of a VC. Until now, the trust ecosystem consists of agents and agent-like things that can be categorized as issuers, verifiers, or credential holders. Further, each issuer needs to publish a credential definition that informs about the underlying schema, the public key material that can be used to verify the integrity of credentials, and finally, the location of a web resource with the specific functionality to allow the construction and verification of ZKPs [56].

The contents of a tails file are known to issuers, verifiers, and VC holders; conceptually, it is a gigantic number that consists of identifiers of active certifiable credentials. When VCs are issued, the holders receive each credential, plus a private factor and a *witness* number that multiplied result in the accumulator value of the tails file.

Oversimplified, the accumulator value is divisible by each non-revoked VC; since one of the design principles is privacy preservation, neither the verifier nor the issuers should be able to correlate the VC holder, and such a unique identifier cannot be used. Luckily, a verifier need not be concerned with the exact value of the private factor such that ZKP can be utilized [4].

Rather than revealing the value of the private factor, it is sufficient to prove that it is part of the accumulator value [27]. Alternatively, revocation could have been implemented using a revocation list where the issuer provides an API endpoint that will return *true* if the credential is active and *false* for revoked VCs. Such an approach would inform the issuer whenever holders show their credentials; it is comparable to how the relying party needs to contact the IdP for each login.

Rather than retrieving the status of a single VC, verifiers could request a (sub-) set of all the active credentials of an issuer. The second approach is already more privacy-preserving than the first; however, the identifier of the credential does not change, and

[4] *cf.* Credential Revocation for further explanation

verifiers can correlate proof presentations to historical proofs. Finally, a ZKP can confirm that a holder's VCs are part of a data structure encompassing all active VCs while fully preserving the holder's privacy.

4.8 Manage Shell Script

To deploy an instance of the TCA, the shell script named *manage* has been developed. The help command provides the usage instructions displayed in Listing 4.4. The available commands include the commands *build*, *provision*, *start*, and *stop*.

The first command initiates the construction of the whole infrastructure; it builds the Angular web client using the Angular CLI and tails server with help of Docker. The provision command is responsible for fulfilling the prerequisites on the first startup; *i.e.* a seed value for the agent must be registered on Hyperledger Indy before the agent can boot up. Further, when deploying the TCA to a local network, it needs to be reachable from the Internet; to that extent, an NGROK tunnel is started and the tunnel URL extracted and set as the argument *–tails-server-base-url* as shown in Listing 4.3. Analogously to the tails-server, the agent also needs to be reachable via the Internet. The respective tunnel URL is extracted and set as the *-e* variable in aforementioned listing. Finally, the two schemas *standard credentials* and *controller user* are registered on Hyperledger Indy through the shell script.

Once provisioned, the agent can subsequently be started with the *start* command. Contrary to the provisioning command, starting the agent does not register the seed on Hyperledger Indy nor are the schemas and credential definitions registered. It starts the NGROK tunnels, extracts the URLs and sets the respective environment variables.

The last command *stop* shuts down all active Docker containers with the command *docker-compose*. Since the NGROK tunnel is not managed by Docker, the script also extracts the process ID of NGROK and subsequently terminates the process.

```
$: ./manage --help
    Description:
    This script provides commands to manage and interact with the trust
       coaltion agent.

    Usage:
    ./manage build
        - builds the infrastructure after the initial download.
    ./manage provision
    - Register seed on Hyperledger Indy.
    - Start all Docker containers.
    - Register schemas and credential definitions on Hyperledger Indy.
        1. standard credentials attributes:
               + sub
               + family_name
               + given_name
               + email
               + valid_from
               + employee_id
               + department
               + role

        2. controller user attributes:
               + first_name
               + last_name
               + employee_id
               + role

    - Ngrok tunnels to agent.

    ./manage start
    - Start all Docker containers (requires provision to have been run
       before).
    - Ngrok tunnels to agent.

    ./manage stop
    - Kill ngrok tunnel and stop all active Docker containers
```

Listing 4.4: Usage of the manage bash script

Chapter 5

Evaluation

This Chapter contains three main parts; Section 5.1 introduces several metrics that capture the scientific contribution of this Master Thesis. Subsequently, Section 5.2 evaluates the implemented prototype based on the previously gathered use cases (*cf.* Appendix A). In Section 5.3, we describe the concrete evaluation scenario, *i.e.* the hard- and software setup of each involved actor is highlighted. Finally, we discuss the findings in Section 5.4; finally,

5.1 Metrics

Within this section, two types of metrics can be found: the first Subsection 5.1.1 concerns itself with metrics that are coupled to the performance of the prototype, whereas the remaining metrics are explained in Subsection 5.1.2 and tied toward security assurances that are essential for the adoption of such a Trust Coalition.

5.1.1 Performance Metrics

Scalability, availability, and interoperability constitute central non-functional requirements that were not the main focus of this prototype. Nonetheless, evaluating the implementation of the Trust Coalition against aforementioned performance metrics may provide valuable insight.

In the context of this evaluation, the metric scalability shall measure how many (concurrent) participants the network can potentially scale to. The mapping type is used as underlying data type to store Trust Coalition membership. It utilizes the keccak256 hash function that maps any input to a 256-bit value [2], *i.e.* 2^{256} domains can technically join the network. Further, we need to identify potential bottlenecks of each Trust Coalition Agent. The TCA can assume one of two roles: issuer or verifier. As discovered in 4.1, a prerequisite for issuing VCs is an active connection to the potential VC holder. Consequently, all communication occurs in the P2P layer through DIDComm (*cf.* third step

in Figure 4.2). Assuming the issuers only distribute VCs to their employees, a scenario where a web server cannot handle the load caused by manually issuing VC becomes hard to imagine. However, when revokable credentials are issued, the issuer must maintain a tails file with a technical limitation of 32'768, set within the Hyperledger Aries framework. This limits the number of VCs based on a credential definition to 32'768; regardless, issuers can publish multiple credential definitions. Thus, issuers do not realistically restrict the amount of VCs a domain can issue. VCs are issued to modern smartphones with many gigabytes of available storage, while VCs require only megabytes of data.

Since virtually any number of agents can hold VCs, verifiers may experience many concurrent proof presentation responses. However, the verification process places most of the computational load on the VC holders. According to [56], a tails file with 32'768 revokable VCs consumes 8.4MB, and an iPhone 12Pro requires approximately seven seconds to generate a proof. For reference, if a maximum of 3000 revokable VCs is issued, the proof generation time is reduced to roughly four seconds. ACApy, on the other hand, is served through the *aiohttp* framework and can handle concurrent requests. The command *docker container stats* delivers a live data stream with statistics concerning the containers' resource consumption. Passing through the authentication and authorization process depicted in Figure 3.4 results in approximately 0.2 MiB of memory usage; the same is true for issuing a VC to a mobile agent. We use 0.3 MiB of memory usage per verification process to estimate the number of concurrent verification flows. Extrapolated, this results in more than 3'000 concurrent VC verification processes per GiB of memory.

For the Trust Coalition to function, various components must be operational simultaneously. *I.e.* verifiers and issuers require an instance of Hyperledger Indy to be operational. Additionally, to establish a DIDComm connection, the ledger hosting the DIDs must be reachable.

As a consequence, there are many points of failure. However, since the trust network is built on DLT, it is fault tolerant through redundancy by design; should one node hosting the ledger experience downtime, other nodes hosting the blockchain will continue to operate and sustain the trust network's functionality. The same applies to the Ethereum blockchain responsible for maintaining the members list.

As many domains are intended to be part of the Trust Coalition, the system must function on a wide range of hardware and software; therefore, interoperability is required and primarily achieved through containerization with Docker. This ensures that the TCA can easily be instantiated on various platforms and environments. Further, through mobile clients downloadable for Android and iOS, a wide range of credential holders can use their preferred mobile platform. Figure 5.1 shows an exemplary VC that is used to authenticate a user of the web client.

5.1.2 Security Assurances

NIST SP 800-63 [26] examines extensively how to ensure a real-life identity exists and is the correct person applying for a digital identity. The IAL of a digital identity belongs to one of three categories (IAL 1-3). The lowest assurance level, IAL1, does not require

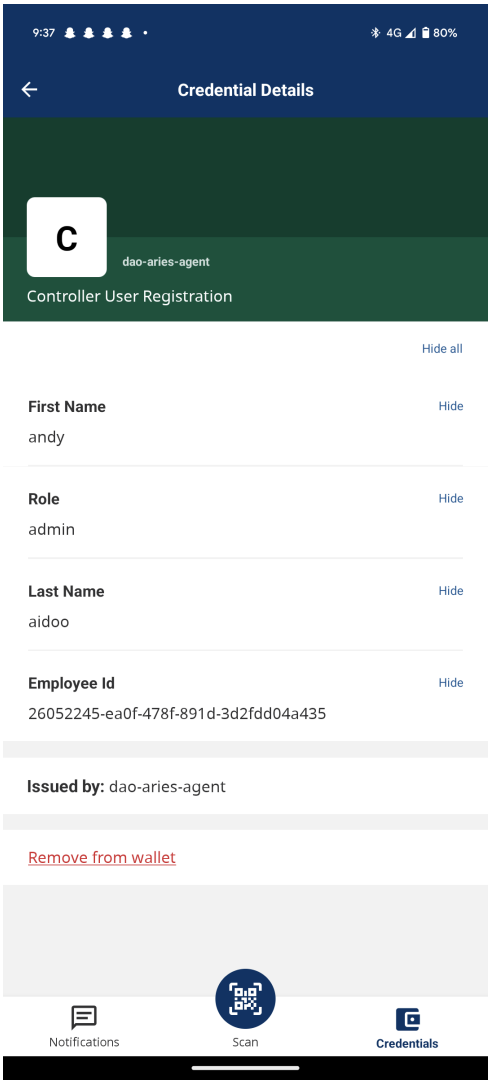

Figure 5.1: Verifiable Credentials issued to BC Wallet

measures guaranteeing the veracity of the attributes; attributes may be self-asserted and need not be validated or verified.

IAL2 requires the physical or digital verification of *e.g.* a government ID; IAL3 requires the validation to be physical only. The IAL of the VCs primarily depends on factors of the individual domains such as: *How sure can I be that the applicant does not impersonate another domain?* and *How well do companies part of the Trust Coalition know about the identity of their employees?*. As a countermeasure for impersonation, the Trust Coalition votes about accepting or rejecting applications using Ethereum smart contracts. Hypothetically, each coalition member can physically verify whether an applicant is who they say they are. Thus, the IAL depends on how extensively the coalition members want to shape the vetting process.

The same standard also guides assigning one of three assurance levels for authentication. The laxest AAL can be achieved by traditional single-factor authentication, whereas AAL2 requires the usage of at least two authentication factors utilizing standard cryptographic

technologies. AAL3 may be achieved with at least two-factor authentication and requires proof of possessing an impersonation-resistant cryptographic key. Authentication based on VC, as proposed by this thesis, with a wallet like *BC Wallet* can always be classified as two-factor authentication. Accessing the VCs inside the wallet requires biometric or PIN-based authentication. Further, the VCs themselves are possession-based authentication factors that are impersonation-resistant. Thus, the requirements for even the most stringent AAL can be satisfied.

The Swiss non-profit association eCH [1] develops digital standards in conjunction with federal and municipal authorities, various universities, and private organizations. They aim to create a secure, interoperable, and seamless digital ecosystem to foster electronic collaboration between (inter-) national agencies. Similarly to NIST, they publish a standard for authentication of digital identities eCH-0170 [2].

They define four assurance levels (1-4) for four categories - namely: authentication (VSA), registration (VSR), federation (VSF), and controls assurance level (VSS). The first three assurance levels have a corresponding counterpart in NIST SP-800-63, albeit with differing criteria for the individual assurance levels. VSS can only be found in eCH-0170, though [38]. Its category depends on how the identity and access management process is audited, its process maturity levels, and liability commitments. Again, these criteria mainly depend on local factors. Table 5.1 summarizes the assurance levels reached by implementing an instance of our prototype.

Category	**NIST SP 800-63**	**eCH-0170**
AAL/VSA	3	4
FAL/VSF	3	4
IAL/VSR	1-3	1-4
VSS	N/A	1-4

Table 5.1: Assurance level according to NIST and eCH

5.2 Prototype Evaluation

To determine whether the prototype meets the derived requirements, we employ a requirements based evaluation approach. From the ten use-cases, we implemented seven through the global governance of the Trust Coalition. This evaluation aims to identify any gaps and builds the foundation to provide recommendations for further improvement of the prototype. Table 5.2 summarizes all initial use cases alongside the information on whether they have been implemented, or not.

Use cases A.1 and A.2 concern themselves with the efficient and secure onboarding of reputable domains; this transaction only requires one actor, the applicant. Domain owners are required to complete an onboarding form and provide proof of domain ownership by submitting the plain text of a mnemonic phrase that has been encrypted using the

[1] E-Government Standards

[2] eCH-0170 Qualitätsmodell zur Authentifizierung von Subjekten

public key material of the domain's TLS certificate. Provided that the plain text can be submitted in the onboarding form (*cf.* Figures 5.2 and 5.3), the onboarding application is accepted without manual intervention from existing Trust Coalition members. The prototype supports the flow depicted in Figure 4.3 for TLS certificates with public keys based on the RSA or elliptic curve algorithm. The field *Decryptor Public Key* denotes the public key used in the encryption. When using the ECIES, additionally, the field *Encryptor Public Key* is returned such that the applicant can derive the shared secret key with their private key material as illustrated in Figure 4.4. Both forms, for the application and the proof of domain ownership, invoke an Ethereum smart contract (*cf.* Listings 4.1 and 4.2). Thus, the two provided scenarios are supported.

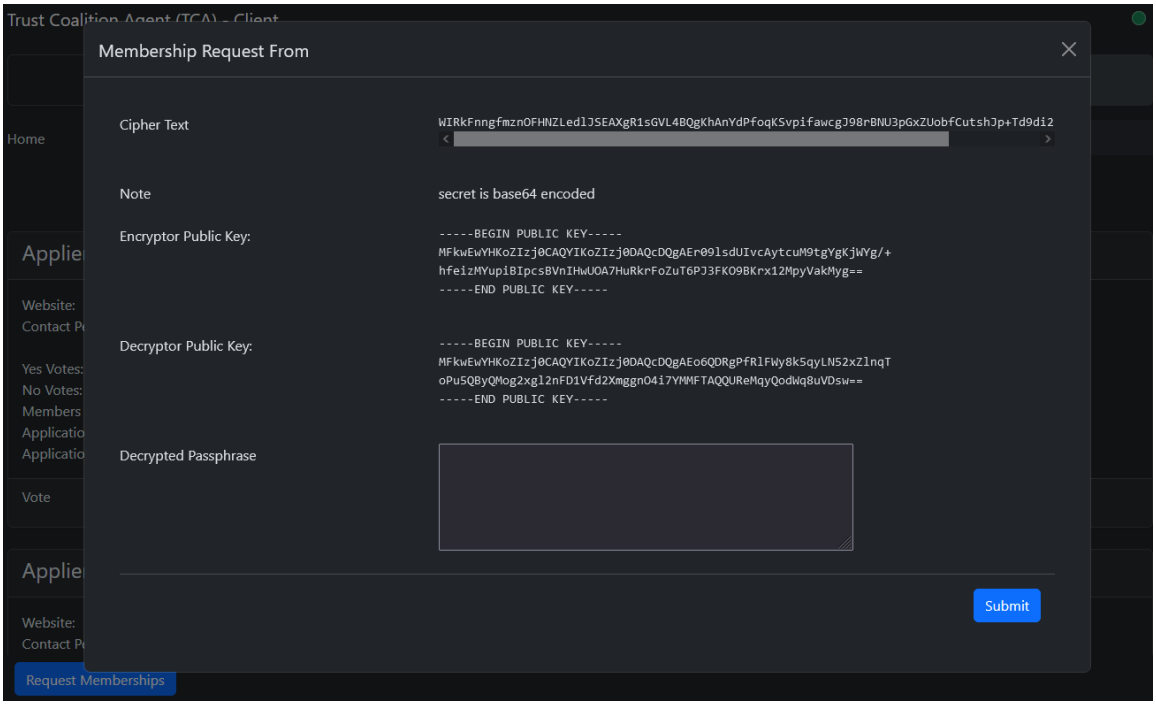

Figure 5.2: Onboarding Challenge for a reputable domain using the ECIES

The third use case A.3 concerns itself with the onboarding of a domain with a DV TLS certificate. For this transaction various stakeholders are involved: the applicant and the existing Trust Coalition members. Since anybody can register a domain with a self-signed or DV TLS certificate, application from such domains are subject to a manual vetting process. An applicant must complete the onboarding form and submit it to invoke an Ethereum smart contract to request TC membership. The applicant provides a contact person that is responsible for answering any questions that existing members might have.

Use case A.4 describes the requirement to vote on active applications from domains with a DV TLS certificate; an active application for the exemplary domain *nzz.ch* is depicted in Figure 5.4. Existing members can vote *Yes* or *No* to support or reject an application; pressing one of the buttons results in the invocation of an Ethereum smart contract that verifies that the vote comes from an existing member for an active application. In the event of a supportive vote, the count of yes votes is increased by one and it is checked whether the required number of votes has been reached. Analogously, when a member

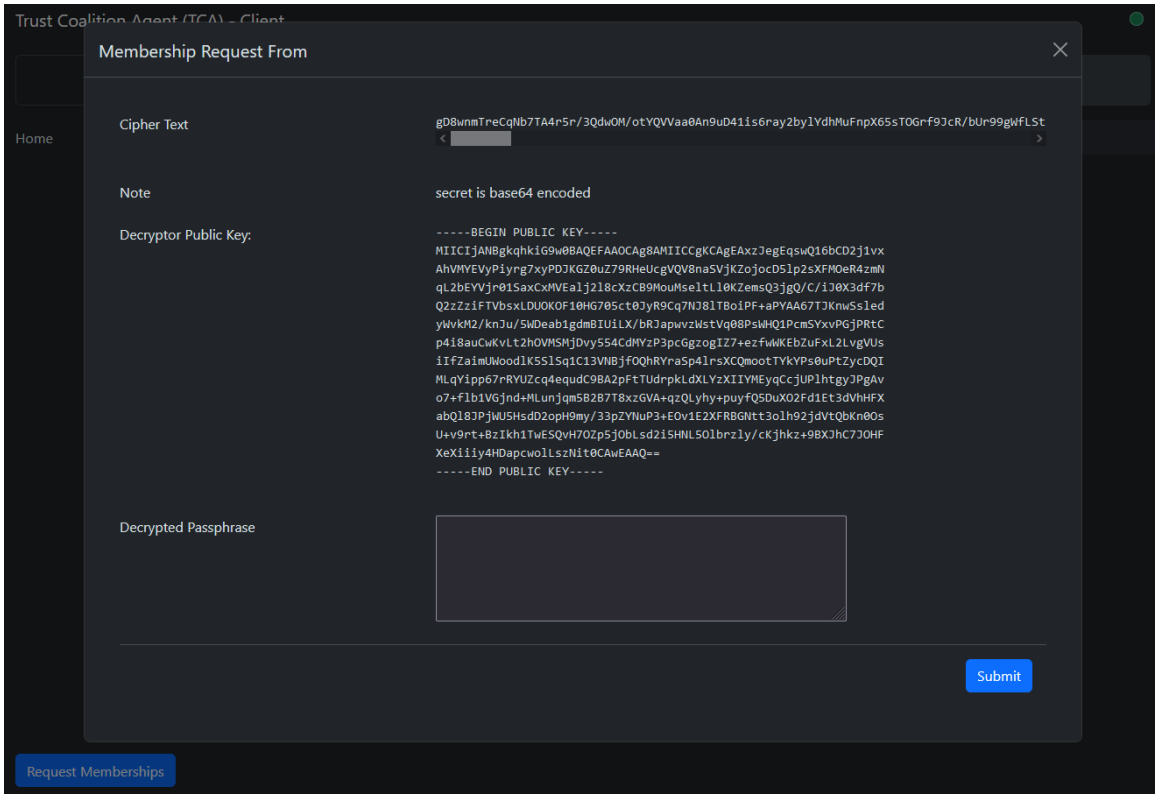

Figure 5.3: Onboarding Challenge for a reputable domain using RSA

rejects the application, the count of no votes is increased by one. Should one-third of the members reject an application, the application *active* and *accepted* attributes are set to false. Whenever a vote results in a two-thirds approval, the application *active* attribute is set to false, while the *accepted* attribute is set to true. Subsequently, the domain is added to the mappings *members* and *membersByDomain*. As a result, existing members can retrieve the credential definition which the VCs of the new domain are based upon. With that functionality, we cover use case A.5; Figure 5.5 lists all available information about the domain. Notably, the identifier of the domain's credential definition is listed such that existing members can confidently accept and verify VC issued by the domain.

```
contract CredentialDAO {
    mapping(address => Company) members;
    mapping(string => Company) membersByDomain;
    ...
    function setCredentialsDefinition(string memory credDef) public {
        require(
            members[msg.sender].walletAddress != address(0),
            "not allowed"
        );
        Company memory member = members[msg.sender];
        string memory domain = member.companyDomain;
        membersByDomain[domain].credentialDefinition = credDef;
    }
}
```

Listing 5.1: Solidity smart contract method to update a domain's credential definition

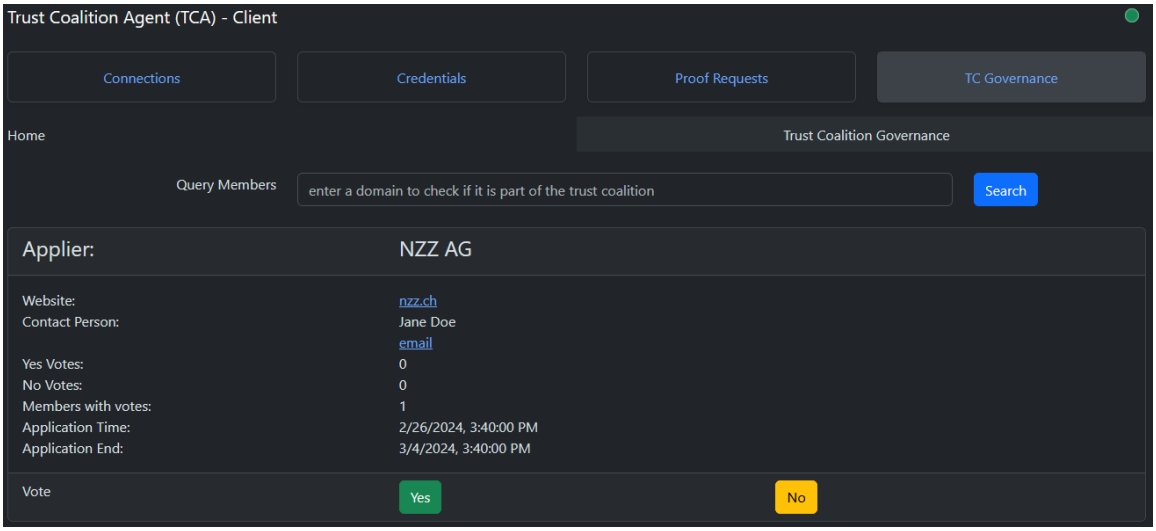

Figure 5.4: Onboarding Members Vote

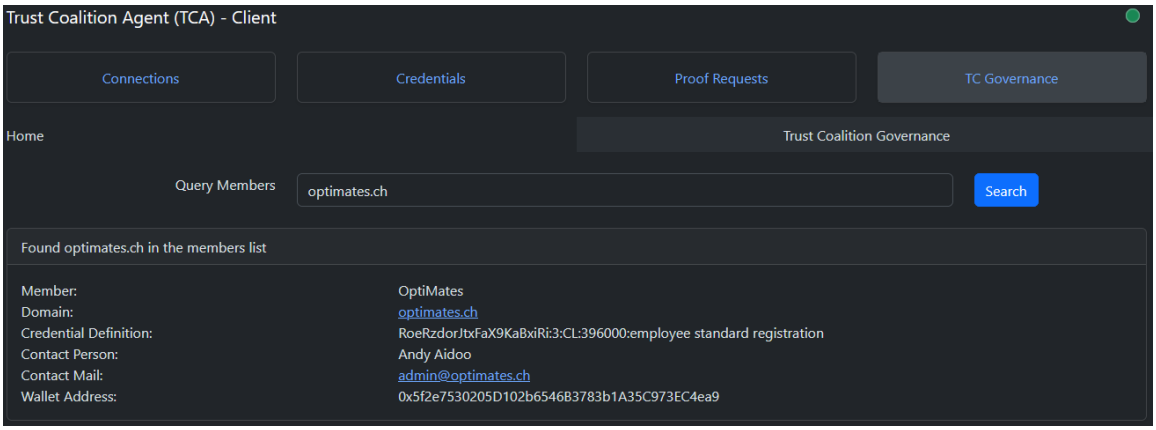

Figure 5.5: Domain Information of an existing Member

By invoking the method in Listing 5.1, the sixth use case A.6 is partially implemented. Domains who want to leave the Trust Coalition on their own accord can achieve their goal by updating/invalidating the credential definition that they used to issue VC to their employees. The functionality within the GUI is missing, however, members can still invoke this method, manually. When the attribute *Credential Definition* is set to *removed*, employees of the domain can still discover their home-realm, however, they will not be able to provide a proof presentation with the credential definition restricted to removed. This invalidates all VCs issued by a domain.

Use case A.9 is initiated by a webclient user of a domain and includes two roles: issuer and credential holder. It captures the requirement to issue certificates of employment that allow VC holders to authenticate their identity to other members' authorization servers. To cover this use case, the prototype provides functionality to issue VC to an individual's mobile wallet and the issuer retains a reference to the VC. The tamper-proof credentials are stored securely (*cf.* Figure 5.1); access to it requires a secret PIN or the mobile device's biometric authentication mechanism. This impersonation-resistant and possession based factor is digitally signed by the issuer, extending trust to the credential holder.

Use case A.10, too, is initiated by a webclient user of a domain and is triggered by the issuer. The second involved role is the credential holder. It requires the functionality to revoke an issued credential such that the invalidity of the credential is noted. The user who revokes the VC must be authentication to ensure authorization. Whenever a credential is revoked through the prototype, the new status is reflected immediately. Figure 5.6 illustrates the presentation of a revoked VC. Rather than deleting the credential from the issuer reference list, the revoked status is noted in the validity of the credentials as shown in 5.7.

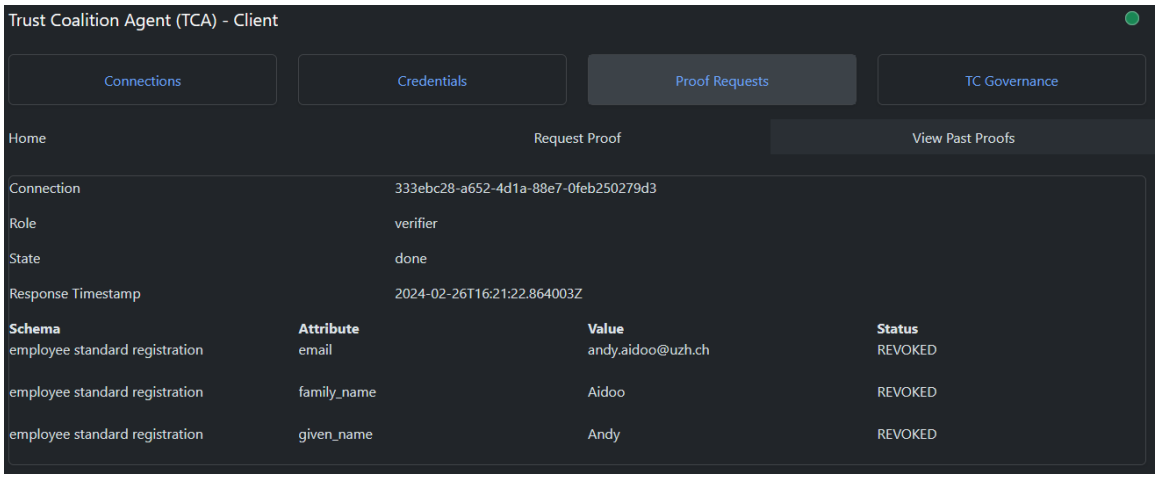

Figure 5.6: Proof Presentation of revoked VC

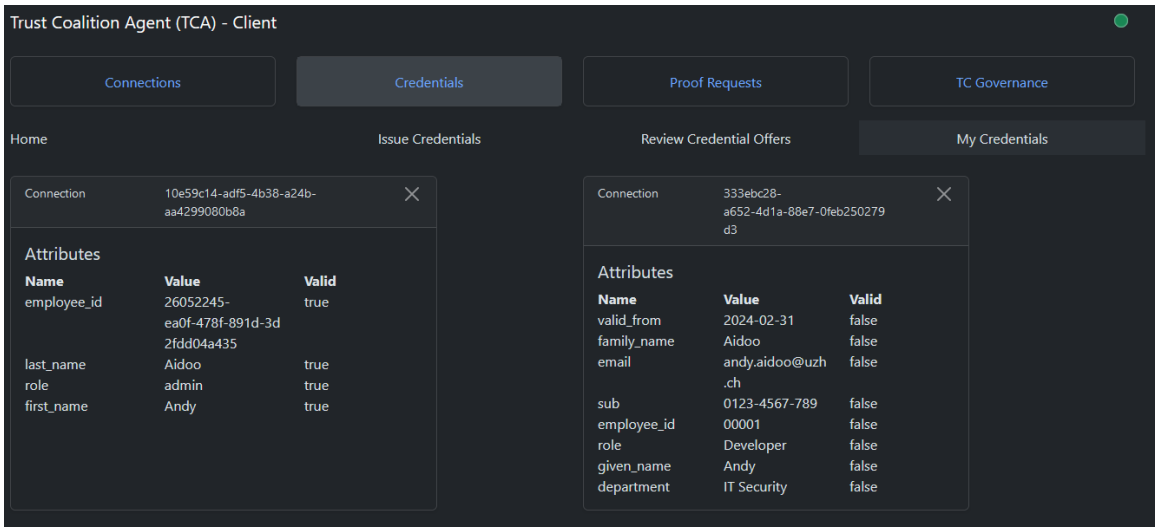

Figure 5.7: Issued VCs by the Webclient alongside their Validity

Use cases A.7 and A.8 have not been implemented within the global Trust Coalition framework; this is mainly due to the maximum code size limitation of 24kb imposed by Ethereum [35] and the priority of these use cases being *low*. Nonetheless, a similar effect can be achieved without relying on the DAO; each domain retains the possibility to manually blacklist certain domains and not accept their issued VCs.

As indicated by Table 5.2, all but three use cases have been completely implemented. Notably, Section 4.1 describes the flow to establish a DIDComm connection with Figure 4.2.

This and the features described in Section 4.3 include functionality that has not been captured within the use cases. During the requirements gathering phase of this thesis, aforementioned features were overlooked. Since they have proven to be pivotal features of the prototype, they were added without the development of the respective use cases.

ID	Title	Evaluation
1.	Onboarding of a domain with Extended Validation (EV SSL) Certificates	●
2.	Onboarding of a domain with Organisation Validated (OV SSL) Certificates	●
3.	Onboarding of a domain with Domain Validated (DV SSL) Certificates	●
4.	Member vote for domain JOINER event	●
5.	Issue membership certificates	●
6.	Voluntary leaver	◐
7.	Forced leaver	○
8.	Member vote for FORCED_LEAVER event	○
9.	Issue employment certificates	●
10.	Revocation of employment certificates	●

● = functionality provided, ◐ = functionality partially provided, ○ = not implemented

Table 5.2: Initial Use Cases and their Implementation Status

5.3 Issuer Architecture

A fully running the Trust Coaltion Agent software system has been deployed and made available through the public Internet. The backend infrastructure is hosted on a Raspberry Pi and a Virtual Private Server (VPS) with approximately 1.9 GiB of RAM running the Linux distribution Debian 12. Figure 5.8 illustrates the architecture of the deployed implementation. A domain points to the static IP address of the VPS; requests to it are processed by an NGINX reverse proxy server. Through the root route, the web client is served that will subsequently perform requests to the backend infrastructure on behalf of the user, namely to ACApy's admin API and the lifecycle-api.

In addition, the webclient interacts with the the Sepolia test net such that users can participate in the governance of the Trust Coalition. ACApy, as any other agent in the ecosystem, is required to store private key material; the ACApy instance is backed by a PostgreSQL database to this end. Listing 4.3 includes the two environment variables *DAO_AGENT_TUNNEL_URL* and *TAILS_SERVER_URL*. The former variable records the endpoint that other agents use in the DIDComm protocol, *i.e.* they can establish connections, request proof presentations, or issue VCs to it. The latter specifies the URL of the web resource hosting the tails file. To satisfy the requirements of this thesis, these two URLs need to be reachable by VC holders and verifiers.

The lifecycle-api, however, only needs to be reachable by issuers to persist the issued verifiable credentials and since it is hosted in a network without a static IP address, a

secure, publicly accessible endpoint to the lifecycle-api is necessitated; an NGROK tunnel serves this purpose. Requests to the lifecycle-api are forwarded to the Raspberry Pi via the aforementioned tunnel. As a consequence, PII that might be entailed in VCs, are stored locally and at a known place providing an extra layer of security, as only very limited functionality is exposed through the tunnel. Similarly to the VPS environment, the lifecycle-API is served behind an NGINX reverse proxy. In this environment, the lifecycle-api persists the validity of the credentials in the PostgreSQL database *lifecycle-db*.

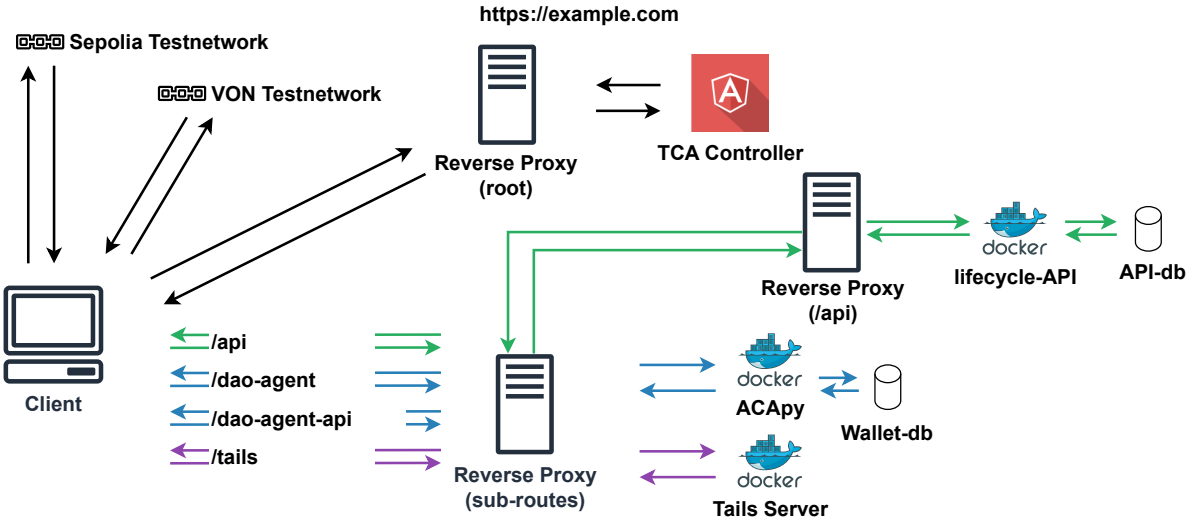

Figure 5.8: Architecture of the deployed TCA software system

5.4 Discussion

Federated governance in a digital Trust Coalition requires a balance between global policies that each coalition member adheres to, while allowing a certain degree of autonomy to *i.a.* promote the adoption of such a governance model. The global policies define a standardized approach for the onboarding of new members. Around this requirement, we devised and implemented a software system capable of integrating any domains democratically; reputable domains are offered an accelerated onboarding process that includes a challenge-response mechanism where a proof of domain ownership is required. Further, the system is capable of managing VCs and performing authentication based on them. The remainder of this section offers an interpretation of the metric results previously derived; furthermore, we comment on the fulfillment of the developed use cases and remark how this thesis successfully meets the general objectives set out in section 1.2.

As explored in Subsection 5.1.1 the proposed network is scalable to virtually any number of participants. For context, depending on the source, the number of atoms in the observable universe is estimated to be between 10^{78} and 10^{82}; 2^{256} (corresponds to 10^{77}) domains can technically join the network through adding their domain name to the members list. The other potential bottlenecks are the webservers that issuers and verifiers are deployed to

and the VC holder's mobile wallet. Since VCs are intended to be issued manually, for the use cases cover by this thesis, the webservers only have to keep up with the speed humans can fill out a credential form. The modest setup described in 5.3 can accommodate for several thousand credentials issued, concurrently. The most realistic bottleneck for issuers may occur after an issuer has been active for an extended period of time; companies with tens of thousands of employees may hit the limitation of 2^{15} or 32'768 credentials that is due to the maximum size of the tails-file per credential definition. Verifiers, on the other hand, can validate around 3'000 proof presentations concurrently per GiB of available RAM. Should the resources of a verifier (almost) be exhausted, time-tested horizontal and vertical scaling techniques can be employed. Consequently, verifiers do not limit the scalability of the proposed framework. The final stakeholder group, VC holders, neither constrain the scalability of the Trust Coalition. One of the design principles, *user-centricity*, stipulates that users must explicitly consent to the sharing of their VCs; manual verification of the presentation request restricts the quantity of credentials shared, rather than the computational power of the holders' mobile phones.

A functioning ecosystem relies on the availability of DLT, issuers, verifiers and holders. By using DLT as trust anchors, the availability of the system is mainly promoted through redundancy. This availability is crucial, as both ledgers used by this prototype provide critical services and may serve numerous issuers, verifiers and VC holders.

Verifiable credentials can be leveraged as possession-based authentication factors. Coupled with a mobile client they can offer the highest AAL/VSA based on the digital identities guides as defined by NIST [26] and eCH [38]. This fact is due to the excellent tools the Hyperledger Foundation provides. In IT security, especially, the security of a software system is only as strong as its weakest link; thus, an inconclusive rating for the IAL justifies further elaboration. Access to the Trust Coalition in the narrow sense is based on a registry that assigns each domain a credential definition, *i. e.* to create a proof presentation request, the validator needs to know which connection it wants a proof from and what credential definition it accepts. Being part of the Trust Coalition means having registered their credential definition in the domain whitelist. Should two-thirds of the Trust Coalition members mindlessly approve every request, anyone can join the coalition without enforcing IAL2 or higher.

To minimize the administrative load encountered in the participation of membership votes, we utilize existing trust in PKI; when a domain applies to become a Trust Coalition member, the TLS certificate of a domain may be leveraged already establish trust between a domain and their certificate authority (CA). Domains with an OV or EV certificate policy have passed vetting processes by a CA and can be directly added to the coalition without a vote from existing members. Proving domain ownership can then be done by solving a challenge like the decryption of a message. Figure 5.2 shows the onboarding challenge for a reputable domain with a TLS certificate based on the elliptic curve algorithm. Similarly, Figure 5.3 is the onboarding challenge when TLS certificate with an RSA public key is encountered.

In an alternative setting where the involved parties share contractual obligations, rather than intrinsic motivation to cooperatively maintain the integrity of the Trust Coalition,

processes that result in higher IAL can reliably be developed. A standardized vetting process for the employees may be part of the stipulations.

Two of the initially planned use cases have not been implemented in Ethereum smart contracts. Namely, there is currently no method available for initiating a voting round to decide on the revocation of a domains affiliation to the Trust Coalition. Consequently, the use case which covers the requirement to cast votes for such a ballot is non-existent, as well. Consequently forced removal of a domain is possible with a global policy. Should domains want to exclude single domains, they are still able to achieve the desired access controls through the maintenance of a local domain-blacklist. Further, removing one's entry from the members list has not been implemented, either; nonetheless, domains may change the identifier for their credential definition and set it to a value such that their employees cannot produce valid proof presentations. The missing uses cases are mainly attributable to the Ethereum smart contract code size restriction and underscores the importance of careful planning and optimization when enforcing global governance policies through Web3.

Chapter 6

Final Considerations

This Chapter includes a summary of the whole thesis, reflection concerning the project, and suggestions for further work in this academic category.

6.1 Summary

This Master Thesis presented a proposition and a prototype to achieve federated governance in a digital trust coalition that allows domains to join the coalition; further, we developed an exemplary web client application to address various potential use cases. Said webclient is designed to manage the governance of affiliation within the coalition and the life cycle of verifiable credentials, providing a user-centric and secure solution. With the help of Ethereum smart contracts, a process to establish digital trust between various domains that make up the members of the trust coalition has been presented.

Through a democratic voting system, each member receives a vote to either endorse or oppose a membership application; two-thirds of the members must vote in favor of the affiliation for it to be accepted. A two-thirds majority was chosen mainly since domains voluntarily contribute to the data mesh architecture; such a high required approval of the members gives individual domains more agency to exclude other domains and keep their data secure. Finally, a software system that may be leveraged to manage VCs and authentication with VCs can be deployed by any domain to ensure the interoperability of the digital identities for trust coalition members.

6.2 Conclusions

In conclusion, a modular software system called trust coalition agent has successfully been developed and deployed; an NGINX reverse proxy serves the controller agent and handles requests to the services tails-server, dao-agent, and lifecycle-api. Additionally, two PostgreSQL databases persist data from ACApy and the lifecycle-api.

Conducting research in cyber security and digital trust has produced numerous instructive takeaways that were instrumental in meeting the general objectives set out for this thesis. For starters, authenticating a person's or domain's identity through digital channels only remains challenging. The Internet's original goal, as mentioned in Section 2.5, is an open and scalable architecture that allows for interconnecting numerous heterogeneous networks and is not designed with security in mind.

The Trust over IP [1] foundation envisions a four-layered trust model for the Internet by following its architecture to standardize the exchange of encrypted and cryptographically signed digital credentials. The lower two levels focus on the technical requirements, while the upper levels concern themselves with satisfying human expectations that remain subject to a governance framework and agreed policies. When numerous domains access and provide information for each other, deciding whose signed credentials to accept requires federated governance and was, in the case of this thesis, responsible for the majority of the use case (see appendix for a complete list A).

Concerning the prototype implementation, we overcame a multitude of obstacles. Despite its matured ecosystem, configuring a customized instance of ACApy still requires familiarization with concepts filled with tribal knowledge; a novel mental model for the decentralized identity paradigm needs to be developed and adopted. Nonetheless, after working through a course on the online learning platform edx [2] and the practical exercise to gain familiarity with the self-documenting Aries OpenAPI [3] milestones like issuing VCs to a mobile agent, requesting proof presentation from it and revocation of VCs regularly delivered learning successes.

In the methodology section of the introduction 1.3, we divided the available time into five phases; the work schedule mostly adhered to the sequential order. Initially, the necessary theoretical knowledge was acquired, we defined general principles before we subsequently defined the prototype's scope. After that, we gathered requirements and developed use cases before their implementation. During the development and testing of the prototype, we encountered individual gaps in how we envisioned the underlying concepts in the decentral identity paradigm would work. We overcame these obstacles by improving the theoretical basis and adjusting the design and implementation accordingly.

6.3 Future Work

A trust ecosystem suitable for exchanging anonymized DDoS attack data can be established through the developed prototype. This research revealed several interesting research directions to build trust ecosystems for various domains. For instance, to accommodate the data sensitivity of government-issued verifiable credentials like a digital ID or driver's license, research into requirements of issuers and wallets to hold credentials is warranted. Future work could explore options for a trust framework for digital driver's licenses or electronic health data.

[1] TOIP Model
[2] Becoming a Hyperledger Aries Developer
[3] GitHub ACApy Demo

Concerning the trust coalition agent prototype, several aspects justify further improvements. Many enterprise applications rely on an authorization server or IdP to authenticate end-users. Consequently, the prototype could be extended with the functionality of serving relying parties the end-user's identity. This would allow seamless integration into federated identity management systems already deployed in large-scale organizations. Finally, the current implementation requires maintaining a list of blacklisted domains; future work could also focus on the removal of trust coalition members directly in the Ethereum smart contract.

Bibliography

[1] Imad Aad. "Zero-Knowledge Proof". In: *Trends in Data Protection and Encryption Technologies*. Ed. by Valentin Mulder et al. 2023, pp. 25–30.

[2] Nico Acosta, Nikola Matić, and Andrew Athan. *Mapping Types*. Feb. 2023. URL: https://github.com/ethereum/solidity/blob/develop/docs/types/mapping-types.rst (visited on Feb. 27, 2024).

[3] IBM Analytics. *The governed data lake approach*. 2016.

[4] Aidoo Andy. *Federated Governance in a Digital Trust Coalition*. Jan. 2024. URL: https://github.com/uzhAndy/federated-governance-in-a-digital-trust-coalition (visited on Feb. 28, 2024).

[5] Oscar Avellaneda et al. "Decentralized Identity: Where Did It Come From and Where Is It Going?" In: *IEEE Communications Standards Magazine* 3.4 (2019), pp. 10–13.

[6] Mark W Beall and Mark S Shephard. "A general topology-based mesh data structure". In: *International Journal for Numerical Methods in Engineering* 40.9 (1997), pp. 1573–1596.

[7] Francisco Betti et al. *Share to Gain: Unlocking Data Value in Manufacturing*. Tech. rep. World Economic Forum, Jan. 2020.

[8] Daniel Bluhm, Timo Glastra, and Stephen Curran. *0160: Connection Protocol*. 2024. URL: https://github.com/hyperledger/aries-rfcs/blob/main/features/0160-connection-protocol/README.md (visited on Feb. 20, 2024).

[9] Daniel Bluhm, Timo Glastra, and Stephen Curran. *0160: Connection Protocol011: Credential revocation*. Sept. 2023. URL: https://github.com/hyperledger/aries-rfcsindy-hipe/blob/main/features/0160-connection-protocoltext/0011-cred-revocation/README.md (visited on Feb. 18, 2024).

[10] Daniel Bluhm et al. *Aries RFC 0454: Present Proof Protocol 2.0*. Jan. 2024. URL: https://github.com/hyperledger/aries-rfcs/blob/main/features/0454-present-proof-v2/README.md (visited on Feb. 18, 2024).

[11] Philipp Bolte. *Self-sovereign Identity: Development of an Implementation-based Evaluation Framework for Verifiable Credential SDKs*. Master's thesis. Oct. 2021.

[12] Clemens Brunner et al. "DID and VC:Untangling Decentralized Identifiers and Verifiable Credentials for the Web of Trust". In: *2020 the 3rd International Conference on Blockchain Technology and Applications*. ICBTA 2020. Dec. 2020.

[13] Marty Cagan. *Inspired: How to create tech products customers love*. 2017.

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

[15] D. Clark. "The Design Philosophy of the DARPA Internet Protocols". In: *SIGCOMM Comput. Commun. Rev.* 18.4 (Aug. 1988), 106–114.

[16] National Research Council, System Security Study Committee, et al. *Computers at risk: safe computing in the information age*. 1990.

[17] Violeta Damjanovic-Behrendt and Wernher Behrendt. "Governance Mechanisms for Federated Digital Platform Ecosystems". In: Dec. 2020.

[18] Zhamak Dehghani. *Data Mesh Principles and logical architecture*. Dec. 2020. URL: https://martinfowler.com/articles/data-mesh-principles.html (visited on July 9, 2023).

[19] Zhamak Dehghani. *How to move beyond a Monolithic Data Lake to a distributed data mesh*. May 2019.

[20] James Dixon. *Pentaho, Hadoop, and Data Lakes*. 2010.

[21] Huang Fang. "Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem". In: *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE. 2015, pp. 820–824.

[22] Bundesanstalt für Finanzdienstleistungsaufsicht. URL: https://www.bafin.de/EN/Aufsicht/FinTech/Geschaeftsmodelle/OpenBanking_OpenFinance/OpenBanking_OpenFinance_node_en.html.

[23] Roberto Garrido-Pelaz, Lorena González-Manzano, and Sergio Pastrana. "Shall we collaborate? A model to analyse the benefits of information sharing". In: *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. 2016, pp. 15–24.

[24] Timo Glastra. *0646: W3C Credential Exchange using BBS+ Signatures*. Oct. 2024. URL: https://github.com/hyperledger/aries-rfcs/blob/main/features/0646-bbs-credentials/README.md (visited on Feb. 19, 2024).

[25] Abel Goedegebuure et al. "Data Mesh: a Systematic Gray Literature Review". In: *ArXiv* abs/2304.01062 (2023).

[26] Paul A Grassi, Michael E Garcia, and James L Fenton. *Digital identity guidelines: revision 3*. Tech. rep. National Institute of Standards and Technology, June 2017.

[27] Daniel Hardman. *0011: Credential revocation*. Sept. 2019. URL: https://github.com/hyperledger/indy-hipe/blob/main/text/0011-cred-revocation/README.md (visited on Feb. 18, 2024).

[28] Microsoft Inc. Dec. 2023. URL: https://learn.microsoft.com/en-us/entra/verified-id/verifiable-credentials-standards (visited on Feb. 19, 2024).

[29] Microsoft Inc. *Revoke a verifiable credential as an issuer - microsoft entra verified ID*. Jan. 2024. URL: https://learn.microsoft.com/en-us/entra/verified-id/how-to-issuer-revoke (visited on Feb. 18, 2024).

[30] Microsoft Inc. *Tutorial - configure Microsoft entra verified ID verifier - microsoft entra verified ID*. Dec. 2023. URL: https://learn.microsoft.com/en-us/entra/verified-id/using-authenticator (visited on Feb. 19, 2024).

[31] Microsoft Inc. *Tutorial - issue Microsoft entra verified ID credentials from an application - microsoft entra verified ID*. Oct. 2023. URL: https://learn.microsoft.com/en-us/entra/verified-id/verifiable-credentials-configure-issuer (visited on Feb. 18, 2024).

[32] Microsoft Inc. *Using the Microsoft Authenticator with Verified ID*. 2023. URL: https://learn.microsoft.com/en-us/entra/verified-id/verifiable-credentials-configure-verifier (visited on Feb. 18, 2024).

[33] *Internet Protocol*. RFC 791. Sept. 1981. URL: https://www.rfc-editor.org/info/rfc791 (visited on Sept. 29, 2023).
[34] *Introduction to Data Mesh - Zhamak Dehghani*. May 2021.
[35] *Introduction to smart contracts*. Aug. 2023. URL: https://ethereum.org/developers/docs/smart-contracts (visited on Feb. 26, 2024).
[36] Nikita Khateev and Stephen Curran. *Aries RFC 0454: Present Proof Protocol 2.0*. Jan. 2024. URL: https://github.com/hyperledger/aries-rfcs/blob/main/features/0454-present-proof-v2/README.md (visited on Feb. 20, 2024).
[37] Nikita Khateev, Stephen Klump, and Stephen Curran. *Aries RFC 0453: Issue Credential Protocol 2.0*. Jan. 2024. URL: https://github.com/hyperledger/aries-rfcs/blob/main/features/0453-issue-credential-v2/README.md (visited on Feb. 20, 2024).
[38] Annett Laube-Rosenpflanzer et al. *Ech E-Government standards*. Oct. 2017. URL: https://www.ech.ch/de/ech/ech-0170/2.0.
[39] Kenneth C Laudon, Jane Price Laudon, and Detlef Schoder. *Wirtschaftsinformatik: Eine Einführung*. 2010.
[40] Maryline Laurent et al. "1 - Digital Identity". In: *Digital Identity Management*. Ed. by Maryline Laurent and Samia Bouzefrane. 2015, pp. 1–45.
[41] Dave Longley and Manu Sporny. *Revocation List 2020*. Apr. 2021. URL: https://w3c-ccg.github.io/vc-status-rl-2020/ (visited on Feb. 19, 2024).
[42] Dave Longley, Manu Sporny, and Eric Steele. *Bitstring Status List v1.0*. Feb. 2024. URL: https://w3c.github.io/vc-bitstring-status-list/ (visited on Feb. 18, 2024).
[43] Mattr Ltd. URL: https://learn.mattr.global/tutorials/revocation/web-credentials/overview (visited on Feb. 18, 2024).
[44] Mattr Ltd. URL: https://learn.mattr.global/docs/products/supported-standards (visited on Feb. 19, 2024).
[45] Mattr Ltd. URL: https://learn.mattr.global/docs/pi-platform/wallet (visited on Feb. 19, 2024).
[46] Mattr. Ltd. *Create a web credential*. URL: https://learn.mattr.global/tutorials/create/web-credentials/basic (visited on Feb. 18, 2024).
[47] Mattr Ltd. *Direct verification of web credentials*. URL: https://learn.mattr.global/tutorials/web-credentials/verify/verify-credential (visited on Feb. 18, 2024).
[48] Zoltán András Lux et al. "Distributed-ledger-based authentication with decentralized identifiers and verifiable credentials". In: *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE. 2020, pp. 71–78.
[49] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. "Data mesh: concepts and principles of a paradigm shift in data architectures". In: *Procedia Computer Science* 196 (2022), pp. 263–271.
[50] Natalia Miloslavskaya and Alexander Tolstoy. "Big data, fast data and data lake concepts". In: *Procedia Computer Science* 88 (2016), pp. 300–305.
[51] Arvind Narayanan and Vitaly Shmatikov. "De-anonymizing social networks". In: *2009 30th IEEE symposium on security and privacy*. IEEE. 2009, pp. 173–187.
[52] Corporate Nist. "The digital signature standard". In: *Communications of the ACM* 35.7 (1992), pp. 36–40.

[53] Don Norman. *The design of everyday things: Revised and expanded edition*. 2013.
[54] Federal Statistical Office. *Masterplan Open Government Data 2024-2027*. URL: https://www.bfs.admin.ch/bfs/en/home/services/ogd/masterplan.html.
[55] Aleksandr Ometov et al. "Multi-factor authentication: A survey". In: *Cryptography* 2.1 (2018), p. 1.
[56] Lucas O'Neil, Pradeep Kumar Prakasam, and Stephen Curran. *Indy Tails Server*. Jan. 2024. URL: https://github.com/bcgov/indy-tails-server/blob/main/README.md (visited on Feb. 22, 2024).
[57] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. "Survey of network-based defense mechanisms countering the DoS and DDoS problems". In: *ACM Computing Surveys (CSUR)* 39.1 (2007), 3–es.
[58] Klaus Pohl and Chris Rupp. *Basiswissen requirements engineering: Aus-und Weiterbildung nach IREB-Standard zum certified professional for requirements engineering foundation level*. 2021.
[59] Michael E Porter, Victor E Millar, et al. *How information gives you competitive advantage*. 1985.
[60] Suhail Qadir and SMK Quadri. "Information availability: An insight into the most important attribute of information security". In: *Journal of Information Security* 7.3 (2016), pp. 185–194.
[61] Bruno Rodrigues et al. "Blockchain signaling system (BloSS): cooperative signaling of distributed denial-of-service attacks". In: *Journal of Network and Systems Management* 28 (2020), pp. 953–989.
[62] Philip Russom. *Data Lakes - Purposes, Practices, Patterns, and Platforms*. Tech. rep. Jan. 2017, p. 40.
[63] Manu Sporny, Dave Longley, and David Chadwick. *Verifiable Credentials Data Model v1.1*. Mar. 2022. (Visited on Feb. 19, 2024).
[64] James M Turner. "The keyed-hash message authentication code (hmac)". In: *Federal Information Processing Standards Publication* 198.1 (2008), pp. 1–13.
[65] Ignacio Velásquez, Angélica Caro, and Alfonso Rodríguez. "Authentication schemes and methods: A systematic literature review". In: *Information and Software Technology* 94 (2018), pp. 30–37.
[66] Victor L Voydock and Stephen T Kent. "Security mechanisms in high-level network protocols". In: *ACM Computing Surveys (CSUR)* 15.2 (1983), pp. 135–171.
[67] Coral Walker and Hassan Alrehamy. "Personal data lake with data gravity pull". In: *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*. IEEE. 2015, pp. 160–167.
[68] Kristin Weber, Boris Otto, and Hubert Österle. "One size does not fit all—a contingency approach to data governance". In: *Journal of Data and Information Quality (JDIQ)* 1.1 (2009), pp. 1–27.
[69] Arif Wider, Sumedha Verma, and Atif Akhtar. "Decentralized Data Governance as Part of a Data Mesh Platform: Concepts and Approaches". In: *arXiv preprint arXiv:2307.02357* (2023).
[70] Kristina Yasuda et al. *OpenID for Verifiable Credentials: A Shift in the Trust Model Brought by Verifiable Credentials*. June 2022. (Visited on Jan. 24, 2024).
[71] Omer Yoachimik and Jorge Pacheco. *DDoS threat report for 2023 Q4*. Jan. 2024. URL: https://blog.cloudflare.com/ddos-threat-report-2023-q4 (visited on Feb. 28, 2024).

[72] Guangsen Zhang and Manish Parashar. "Cooperative detection and protection against network attacks using decentralized information sharing". In: *Cluster Computing* 13.1 (2010), pp. 67–86.

[73] University of Zurich. URL: `https://www.informationsecurity.uzh.ch/de/Organisation/Rolle-des-CISO.html`.

Abbreviations

AAL	Authentication Assurance Level
ACApy	Aries Cloud Agent Python
BaFin	Bundesanstalt für Finanzdienstleistungsaufsicht
CA	Certificate Authority
CIA	Confidentiality, Integrity and Availability
CISO	Chief Information Security
CLI	Command Line Interface
CSG	Communication Systems Group
DAO	Decentralized Autonomous Organization
DDoS	Distributed Denial of Service
DLT	Distributed Ledger Technology
DoS	Denial of Service
FAL	Federation Assurance Level
ECIES	Ecliptic Curve Integrated Encryption Scheme
EV	Extended Validation
IETF	Internet Engineering Task Force
IAL	Identity Assurance Level
IdP	Identity Provider
IP	Internet Protocol
IT	Information Technology
IoT	Internet of Things
JWT	JSON Web token
JSON	JavaScript Object Notation
MAC	Message Authentication Code
M2M	Machine-to-Machine
OASI	Old-age and survivors's insurance
OIDC	OpenID Connect
OV	Organization Validated
PKI	Public-key Infrastructure
RSA	Rivest-Shamir-Adleman
SDK	Software Development Kit
SSI	Self-Sovereign Identity
TCA	Trust Coalition Agent
UZH	University of Zurich
VC	Verifiable Credential
VSA	Vertrauensstufe der Authentisierung

VSR	Vertrauensstufe der Registrierung
VSF	Vertrauensstufe der Föderierung
VSS	Vertrauensstufe der Steuerung
VPS	Virtual Private Server
PII	Personally Identifiable Information
WEF	World Economic Forum
W3C	World Wide Web Consortium
ZKP	Zero Knowledge Proof

List of Figures

List of Tables

Listings

Abbreviations

AAL	Authentication Assurance Level
ACApy	Aries Cloud Agent Python
BaFin	Bundesanstalt für Finanzdienstleistungsaufsicht
CA	Certificate Authority
CIA	Confidentiality, Integrity and Availability
CISO	Chief Information Security
CLI	Command Line Interface
CSG	Communication Systems Group
DAO	Decentralized Autonomous Organization
DDoS	Distributed Denial of Service
DLT	Distributed Ledger Technology
DoS	Denial of Service
FAL	Federation Assurance Level
ECIES	Ecliptic Curve Integrated Encryption Scheme
EV	Extended Validation
IETF	Internet Engineering Task Force
IAL	Identity Assurance Level
IdP	Identity Provider
IP	Internet Protocol
IT	Information Technology
IoT	Internet of Things
JWT	JSON Web token
JSON	JavaScript Object Notation
MAC	Message Authentication Code
M2M	Machine-to-Machine
OASI	Old-age and survivors's insurance
OIDC	OpenID Connect
OV	Organization Validated
PKI	Public-key Infrastructure
RSA	Rivest-Shamir-Adleman
SDK	Software Development Kit
SSI	Self-Sovereign Identity
TCA	Trust Coalition Agent
UZH	University of Zurich
VC	Verifiable Credential
VSA	Vertrauensstufe der Authentisierung

VSR	Vertrauensstufe der Registrierung
VSF	Vertrauensstufe der Föderierung
VSS	Vertrauensstufe der Steuerung
VPS	Virtual Private Server
PII	Personally Identifiable Information
WEF	World Economic Forum
W3C	World Wide Web Consortium
ZKP	Zero Knowledge Proof

Appendix A

List of use cases

	Content
Name	Onboarding of a domain with Extended Validation (EV SSL) Certificates
Priority	High
Short Description	As a domain owner with an EV SSL certificate, I want to be able to sign an application with the domains valid EV SSL Certificate to join the data mesh without the voting process.
Trigger	User Interaction
Agent	CISO/CSIRT lead
Precondition	Domain owns a valid EV SSL certificate
Postcondition	Process start to issue membership certificate
Result	Entry on ledger to request membership certificate
Scenario	A reputable company that follows company industry standards and possesses an EV SSL certification is unlikely to conduct fraudulent activities. These certificates provide the highest assurance that a legitimate company is behind an application to the Data Mesh. Such request should follow a frictionless onboarding process.

Table A.1: Membership Life Cycle - Onboarding of a domain with Extended Validation (EV SSL) Certificates

	Content
Name	Onboarding of a domain with Organisation Validated (OV SSL) Certificates
Priority	High
Short Description	As a domain owner with an OV SSL certificate, I want to be able to sign an application with the domains valid OV SSL Certificate to join the data mesh without the voting process.
Trigger	User Interaction
Agent	CS professor
Precondition	Domain owns a valid OV SSL Certificate
Postcondition	Process start to issue membership certificate
Result	Entry on ledger to request membership certificate
Scenario	Domain owner with OV SSL certificates similarly to EV SSL levels provide reasonable assurance that they aren't malicious actors either. An application that is signed with an OV SSL should analogously follow a frictionless onboarding process.

Table A.2: Membership Life Cycle - Onboarding of a domain with Organisation Validated (OV SSL) Certificates

	Content
Name	Onboarding of a domain with Domain Validated (DV SSL) Certificates
Priority	High
Short Description	As a domain owner with an DV SSL certificate, I want to be able to sign an application with the domains valid DV SSL Certificate to join the data mesh without the voting process.
Trigger	User Interaction
Agent	Network/Server administrator
Precondition	Domain owns a valid DV SSL certificate
Postcondition	Launch of a membership certificate vote for JOINER event.
Result	Entry on ledger to start vote for approval of JOINER event.
Scenario	Applications signed with an DV SSL Certificate have not been checked for legitimacy; they may be created by anybody with the technical knowledge. The request may be generated by a malicious actor and should thus pass through a second verification process - member can reject or collectively issue a member certificate, when the community deems the requestor fitting to the Data Mesh.

Table A.3: Membership Life Cycle - Onboarding of a domain with Domain Validated (DV SSL) Certificates

	Content
Name	Member vote for domain JOINER event
Priority	High
Short Description	As a DAO member, I want to vote about the approval of a new member
Trigger	Application with low assurance of requestor identity
Agent	DAO
Precondition	Application to start onboarding process
Postcondition	Voted to approve or deny the application
Result	Entry of vote on ledger
Scenario	A domain that owns a DV SSL certificate wants to join the network. Since its authenticity has not been established with a high enough assurance, it is subject to a vetting process. Member are informed about a candidate and they are advised to form a consensus whether the application should be approved or rejected by casting a vote.

Table A.4: Membership Life Cycle - Member vote for domain JOINER event

	Content
Name	Issue membership certificates
Priority	High
Short Description	As a Decentralized Autonomous Organisation (DAO) I want to issue a member ship certificate that can be used to create digital identities for access to the Data Mesh.
Trigger	Application with high assurance of requestor identity
Agent	DAO
Precondition	Approved JOINER event by members of DAO
Postcondition	Creation and issuance of membership certificate
Result	Entry on ledger to issue membership certificate
Scenario	A new member has been confirmed, either from passing a vote or having provided a valid OV/EV SSL certificate. The DAO then signs the public key of the requestor and publishes it on the ledger.

Table A.5: Membership Life Cycle - Issue membership certificates

	Content
Name	Voluntary leaver
Priority	Low
Short Description	As a member, I want to be able to leave the Data Mesh for any reason on my own accord.
Trigger	User Interaction
Agent	Membership responsible
Precondition	Agent is a valid member of the mesh network
Postcondition	Agent is no longer a member of the Data Mesh
Result	Entry on ledger that records the revocation of the member's public key.
Scenario	A member of the Data Mesh no longer wants to grant access to their data to other members of the mesh network. As a consequence, the member is not allowed to access the data of other members, anymore. Their public key must be invalidated and they can no longer issue self-signed credentials for their employees. Further, their node ip address will no longer be queried during data retrieval.

Table A.6: Membership Life Cycle - Voluntary leaver

	Content
Name	Forced leaver
Priority	Low
Short Description	As DAO, I want to be able to revoke the membership status after a vote to do so has been approved.
Trigger	FORCED_LEAVER event
Agent	DAO
Precondition	Approved vote by the majority of the members
Postcondition	Party is no longer a member of the Data Mesh
Result	Entry on ledger that records the revocation of the member's public key.
Scenario	A member of the Data Mesh has been voted to be expulsed from the mesh network for violating the DAOs terms or any other reason. To prevent their self-signed credentials from being accepted, their membership certificate is revoked, they no long have access to other members' data and their node will no longer be queried during data retrieval.

Table A.7: Membership Life Cycle - Forced leaver

	Content
Name	Member vote for FORCED_LEAVER event
Priority	Low
Short Description	As a member of the Data Mesh, I want to be able to vote about the revocation of another member's affiliation.
Trigger	FORCED_LEAVER event
Agent	DAO
Precondition	A new vote has been created to decide about the revocation of a member status
Postcondition	The revocation vote has been accepted or rejected
Result	Entry of vote on ledger
Scenario	An existing member is accused of fraud or has breached the membership agreement. A member notices it and wants to vote with the other members of the DAO about the expulsion of the suspected member.

Table A.8: Membership Life Cycle - Member vote for FORCED_LEAVER event

	Content
Name	Issue employment certificates
Priority	High
Short Description	As a member of the Data Mesh, I want to be able to grant access to the Data Mesh to my operators
Trigger	User Interaction
Agent	Membership responsible
Precondition	Member possess valid Membership Certificate
Postcondition	Creation and issuance of team member certificate
Result	Entry on ledger to issue team member certificate
Scenario	A new member wants to create a user with which they can access data in the mesh network. The owner of the private key from the application issues credentials for the employee that wants to access the Data Mesh.

Table A.9: Verifiable Credentials Life Cycle - Issue employment certificates

	Content
Name	Revocation of employment certificates
Priority	High
Short Description	As a member of the Data Mesh, I want to be able to revoke access to the Data Mesh to my operators
Trigger	User Interaction
Agent	Membership responsible
Precondition	Member possess valid Membership Certificate
Postcondition	Revocation of team member certificate
Result	Entry on ledger to revoke team member certificate
Scenario	An existing member wants to revoke access to the Data Mesh. The owner of the private key from the application revokes credentials for the employee that no longer wants to access the Data Mesh.

Table A.10: Verifiable Credentials Life Cycle - Revocation of employment certificates

Appendix B

Installation Guide

Before running an instance of the TCA, make sure that the following prerequisits are installed:

- Angular CLI
- git
- ngrok
- npm
- Pipenv

For detailed guidelines for the installation refer to the README.md of the published repository [4]:

1. Clone this repository:
   ```
   git clone https://github.com/uzhAndy/federated-governance-in-a-digital-trust-coalition
   cd federated-governance-in-a-digital-trust-coalition
   ```
2. Build the infrastructure by executing:
   ```
   ./manage build
   ```
3. For the initial start run:
   ```
   ./manage provision
   ```
4. For subsequent starts run:
   ```
   ./manage start
   ```
5. To shut down all components run:
   ```
   ./manage stop
   ```