Architecture and Tools for Self-sovereign Identity Management on Distributed Ledgers

Michael Grabatin

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

angenommenen Dissertation.

Gutachter:

1. Prof. Dr. Wolfgang Hommel
2. Prof. Dr. Arno Wacker

Die Dissertation wurde am 28.07.2023 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 29.11.2023 angenommen. Die mündliche Prüfung fand am 01.12.2023 statt.

Kurzfassung

Identitätsmanagement stellt Nutzer, Dienstbetreiber, Organisationen und Regierungen immer wieder vor große Herausforderungen. Um diesen Herausforderungen zu begegnen, sind viele Identitätsmanagementsysteme entwickelt worden. Die Verfahren reichen dabei von zentralisierten über föderierte bis hin zu nutzerzentrierten Varianten. Diese sind großteils in der ein oder anderen Form aktiv im Einsatz. Selbstbestimmte Identitäten (englisch: *self-sovereign identity (SSI)*) sind die nächste Evolution in dieser Reihe von Ansätzen. Die Verwendung von Blockchain- und Distributed-Ledger-Technologien verspricht dabei die Umsetzung tatsächlicher Nutzerzentriertheit, durch eine Dezentralisierung und Entmachtung klassischer Identitätsprovider. Ziel ist es dabei, die persönliche Verwendung von Identitätsdaten im Digitalen den physischen Ausweisen der realen Welt anzugleichen.

In dieser Arbeit wird ein Konzept für selbstbestimmte Identitäten entwickelt, welches nicht nur persönliche Identitäten im Internet und die elektronische Identifizierung (eID) von Bürgern abdeckt, sonder auch die Identitäten von Geräten im Internet-of-Things (IoT) und Systemen im Cloud-Computing-Umfeld einschließt. Durch die Entwicklung eines solchen, möglichst umfassend einsetzbaren Konzepts, soll es ermöglicht werden, dass das Entstehen weiterer Identitätssilos vermieden werden kann. Die hierzu entwickelte Referenzarchitektur beschreibt ein System, welches auf vielfältige Use-Cases angewendet werden kann. Dazu werden unter anderem Prozesse entwickelt, die es ermöglichen verschiedene Identitätsmanagementsysteme miteinander zu verbinden. Entweder durch das Bereitstellen von Übersetzungsdiensten, die Regeln bereit stellen, um Zusicherungen eines Identitätsmanagementsystems in die eines anderes zu übersetzen, oder durch Gateways, die verschiedene Identitätsmanagementsysteme verbinden.

Zur Evaluation wird das Konzept anhand von drei Prototypen aus den Szenarien Internet, IoT, eID angewendet. Zusätzlich wird gezeigt, dass das Konzept prinzipiell geeignet ist, die individuellen Prototypen in einem gemeinsamen großen Szenario zu verbinden.

Abstract

Identity and access management regularly challenges users, service operators, organizations, and governments. Solutions to those challenges exist in identity management systems, which range from centralized to federated and user-centric designs. Most of the systems in use today follow one of those approaches. Self-sovereign identity (SSI) is the next step in evolving identity management systems. It uses the decentralization of blockchain and distributed ledger technologies to replace classic identity providers. Decentralization offers the possibility of realizing actual user-centric use of digital identities with a system similar to how physical ID cards are handled.

This work develops a concept for self-sovereign identity management, which encompasses personal identities on the Internet and for electronic identification (eID), as well as devices from the Internet of Things and the cloud computing world. Including all of those scenarios into one concept aims to prevent the establishment of more identity silos. The reference architecture developed in this work describes a system that can be applied to all those scenarios. An essential role in preventing identity silos is establishing processes that foster interoperability. The concept shows two options: localization services, which help services interpret assertions from other identity management systems, and gateways, which connect different systems as proxies.

To evaluate the concept, the Internet, IoT, and eID scenarios are implemented as prototypes. They show how the concept can be applied to each use case. Additionally, it is shown how a combined scenario with all three prototypes could work.

Acknowledgement

This work was created during my time as a research assistant at the *Research Institute Cyber Defence* (RI CODE) of the *University of the Bundeswehr Munich* (UniBw M). I am extremely grateful for the unwavering support, invaluable feedback, and extraordinary patience of my doctorate supervisor Prof. Dr. Wolfgang Hommel, holder of the professorship *Professur für IT-Sicherheit von Software und Daten* and executive director of RI CODE. I am also thankful for Prof. Dr. Arno Wacker for serving as the second examiner and the complete examination commission for supporting my work.

This work would not have been possible without my collogues at the *Institute for Software Technology*. They contributed massively to this work by providing professional opinions and created an enjoyable working environment. I'd like to also recognize the positive influence of all projects' partners, who I had the pleasure of working with and which helped shaping this work.

I am deeply indebted to my wife Anna, whose eternal patience and appreciation supported me along the way. Special thanks to my parents and brother, who supported me during my studies and backed my decision to pursue this endeavor. Lastly, I'd like to thank all friends and family for their sustained motivation.

Contents

List of Figures

List of Tables

Abbreviations

2FA two-factor authentication.

AA attribute authority.
AAL authentication assurance level.
ACA-PY *Hyperledger Aries Cloud Agent - Python*.
ACIM application-centric identity management.
ACL access control list.
AES *Advanced Encryption Standard*.
API application programming interface.
ASN.1 *Abstract Syntax Notation One*.
authNZ authentication and authorization.
AWS *Amazon Web Services*.

BAN logic Burrows–Abadi–Needham logic.
BSI *Bundesamt für Sicherheit in der Informationssicherheit*.
BSP biometric service provider.

CA certificate authority.
CBOR concise binary object representation.
CID cryptographic identifier.
CIDR classless inter-domain routing.
CIM centralized identity management.
CLS credential localization service.
CMDB configuration management database.
CPU central processing unit.
CRL certificate revocation list.
CRUD create, read, update, and delete.
CT certificate transparency.

DAG directed acyclic graph.
DDo decentralized identifier document.
DDoS distributed denial of service.
DID decentralized identifier.
DID Auth decentralized identifiers authentication.
DKMS distributed key management system.
DLT distributed ledger technology.
DMZ demilitarized zone.
DNS domain name system.
DoS denial of service.
DSA *Digital Signature Algorithm*.

ECC elliptic curve cryptography.
ECDSA *Elliptic Curve Digital Signature Algorithm*.
eIC electronic identification card.
eID electronic identification.
eIDAS *electronic IDentification, Authentication and trust Services*.
eIDS electronic identification scheme.
EU *European Union*.

FAL federation assurance level.
FCAPS fault, configuration, accounting, performance, and security management.
FIM federated identity management.
FIPS *Federal Information Processing Standard*.

GDPR *General Data Protection Regulation*.
GSM global system for mobile communications.

HMAC hashed message authentication code.
HSM hardware security module.
HTML *HyperText* markup language.
HTTP *HyperText* transfer protocol.
HTTPS *HyperText* transfer protocol secure.

IAL identity assurance level.
IAM identity and access management.
ICANN *Internet Corporation for Assigned Names and Numbers*.
ICAO *International Civil Aviation Organization*.
ID identifier.

IDM identity management.
IdP identity provider.
IDS intrusion detection system.
IETF *Internet Engineering Task Force*.
IGF *Internet Governance Forum*.
iOS Apple's mobile operating system.
IoT *Internet of Things*.
IP Internet protocol.
ISO/IEC ISO/IEC.
ISO/OSI ISO/OSI.
IT information technology.
ITSM IT service management.
ITU-T *International Telecommunication Union Telecommunication Standardization Sector*.

JS *Javascript*.
JSON Javascript object notation.
JSON-LD JSON linked data.
JWT JSON web token.

LAN local area network.
LED light emmiting diode.
LoA level of assurance.
LPWAN low-power wide area network.

MAC mobile ad-hoc cloud computing.
MCC mobile cloud computing.
MFA multi-factor authentication.
MIC message integrity code.
MITM human-in-the-middle.
MQTT message queuing telemetry transport.
MTA mail transfer agent.

NETCONF *Network Configuration Protocol*.
NFC near-field communication.
NIST *National Institute of Standards and Technology*.
nPA *neuer Personalausweis*.
NW *North Rhine-Westphalia*.

OASIS *Organization for the Advancement of Structured Information Standards*.
OAuth *OAuth 2.0*.
OCSP online certificate status protocol.
OIDC *OpenID Connect*.
OOP once-only principle.
OSI open systems interconnection.
OTP one-time password.

P2P peer-to-peer.
PAN personal area network.
PC personal computer.
PCD proximity coupling device.
PEPS *Pan European Proxy Service*.
PFS perfect forward secrecy.
PGP pretty good privacy.
PICC proximity integrated circuit card.
PII personally identifiable information.
PIN personal identification number.
PKI public key infrastructure.
PoS proof of stake.
PoW proof of work.
PSK pre-shared key.
PUF physically unclonable function.

QR code quick response code.

RBFT redundant byzantine fault tolerant.
RDBMS relational database management system.
RP relying party.
RSA *Rivest–Shamir–Adleman*.

SAML security assertion markup language.
SCOOP4C *Stakeholder Community Once-Only Principle For Citizens*.
SDN software-defined networking.
SHA *Secure Hashing Algorithm*.
SMS short message service.
SNMP *Simple Network Management Protocol*.
SP service provider.
SQL structured query language.
SSI self-sovereign identity management.
STEM science, technology, engineering, and mathematics.
StMD *Bayerisches Staatsministerium für Digitales*.

TCP/IP transmission control protocol/Internet protocol.
TDD test-driven development.
TGW trust gateway.
TLS transport layer security.
TOOP *The Once-Only Principle Project*.
TPM trusted platform module.
TTN *The Things Network*.
TTS *The Things Stack*.

UCIM user-centric identity management.
UI user interface.
URI uniform resource identifier.
URL uniform resource locator.
URN uniform resource name.
USB universal serial bus.
UUID universally unique identifier.

VC verifiable credential.
VCTF *Verifiable Claims Task Force*.

W2A wallet to agent.
W2W wallet to wallet.
W3C *World Wide Web Consortium*.
WAN wide area network.
WLAN wireless local area network.
WPAN wireless personal area network.

XML extensible markup language.
XSD XML schema definition.

Chapter 1

Introduction

Contents

A general push for comprehensive digitalization paired with resilient, decentralized networks and the exploration of distributed ledger technology (DLT) with a focus on so-called blockchains has captured the imagination of many enthusiasts, spawned much new research, and fueled the development of new business ventures [117]. In this environment with many diverse ideas and new technologies, there are many potential research topics [79]. One persistent problem with regular computing, cloud computing, the *Internet of Things* (IoT), and the blockchain is the management of identities [146]. Identity management (IDM) has been examined and discussed many times, but a generally applicable and interoperable system still needs to be developed. Blockchain technology and DLT may offer new approaches and solutions to those long-existing problems [48].

The IoT adds connectivity to many existing devices and introduces numerous additional devices like sensors, relays, or control appliances. They all need to connect to services hosted on the Internet to be used to their full potential. In order to keep this influx of new devices and services in check and provide proper security throughout the whole process chain, new identity management approaches are needed. This is especially important as the data collected by IoT devices usually contains a high amount of personal information, which could allow the building of profiles of anybody who is recorded by such devices [171].

Identity management, sometimes synonymously called identity and access management (IAM), describes the technical and organizational aspects of ensuring that only those entities (e. g., people) can access resources (e. g., web services) that are allowed to do so. Later on, this work distinguishes between identity management and identity and access management by using the first term to describe the management of identities without any direct connection to access decisions and uses the latter term to refer to the complete process of managing identities and which resources they can access. Part of the IAM process is the identification, authentication, and authorization of entities by other entities or services. Within those three steps, identification uniquely identifies an entity within a given setting or scope. Standard identification methods are usernames, email addresses, or telephone numbers. The next step, authentication, is used to confirm that an entity

is correctly identified. The most common authentication method currently used on the Internet is providing the matching password to a user ID in a site-specific IAM system [78]. After authenticating an entity, authorization determines which resources can be accessed by the entity. There are many ways to implement authorization, starting with access control lists (ACLs), which explicitly list which user is allowed to access which resource, a solution that can get confusing quickly if there are many users and resources. Group-based or attribute-based systems enhance plain ACLs. The first encapsulates users into groups, which are then associated with access rights. The latter requires users to possess or present a specific attribute to access a resource.

As is apparent, IAM is an integral part of modern-day IT security. Without it, it is not possible to provide paid-for or private services over a publicly accessible network like the Internet. This is why there are many ideas, implementations, and standards out there to improve IAM systems' usability, security, and compatibility [190]. Still, the predominant IDM solution used today for identification and authentication with each service is based on combinations of usernames and passwords. This system is also regularly criticized as being responsible for many breaches because it is hard for people to remember good passwords without reusing them across services. This trend will primarily be confirmed with the continued rise of IoT devices and services, which will introduce a massive amount of typically low-powered devices that need to be managed securely.

This is why current research is, among other things, looking at using blockchain or, more generally, DLT to ensure that IDM keeps up with the demands of the Internet, IoT, cloud computing, and government.

The core concept that makes blockchains interesting tools is their ability to run a distributed database. To keep consistency, a network of non-trusting peers can run this database and still achieve consensus about ordering writes to the database. The first widely noted implementation of this concept is called *Bitcoin* [135]. Bitcoin implements a digital cryptocurrency in a way that solves the problem of being able to copy digital currency at will without introducing a trusted third party. With digital representations of money, creating a copy indistinguishable from the original is possible. If a trusted entity does not manage the digital money, it could be copied by users as much as they like. Within blockchains, a similar problem is called the problem of *double-spending*, i. e., spending the same money twice.

Bitcoin solves *double-spending* by having a decentralized database in which every transaction (i. e., transfer of coins) is recorded. Using this database, every peer in the network can verify whether a *coin* has already been spent or whether the sender of that coin is still the righteous owner. Those digital coins can be associated with anything. They might be exchanged for money as they are in Bitcoin or used to show some other form of possession or property.

Expanding the concept of transferring some coins or property between users is the concept of *smart contracts*. Popularized by the blockchain *Ethereum*, smart contracts allow the peers of a network to form a consensus not only on which coin has been transferred between which peers but also allow the execution of arbitrary code within a transaction, which may trigger further transactions [28]. Using this model, the blockchain becomes the foundation of a distributed network computer. This worldwide computer can then be used to implement cryptocurrencies [169], voting platforms, and much more.

The key to the interest and excitement about blockchains is not necessarily its computational strength or storage capacity, which are small and expensive compared to conventional servers or cloud systems, but the transparent validation of transactions by all peers. This system allows parties that do not necessarily trust each other to form contracts they are bound to, and no one can cheat the other.

1.1 Motivation and Objective

Self-sovereign identity management (SSI) is the advancement and combination of federated identity management and user-centric identity management. Its goal is to put the user in control of their identity by using a federated identity management (FIM)-like setup with decentralized identity providers (IdPs). This can give users ultimate control of their identity and any attached attributes.

Federated identity management is – in part – already decentralized. It is usually structured around multiple IdPs, that manage their users' identities and attributes and service providers (SPs) that provide resources to those users. IdPs and SPs are spread across multiple organizations but inter-connected through a so-called *federation*. The federation dictates the legal, organizational, and technical aspects of exchanging identity information through a contract signed by all participants. Exemplary aspects governed by a federation's contract can be who may join the federation, the actuality and quality of data provided by IdPs, methods used for exchanging data between IdPs and SPs, or usage restrictions and privacy guidelines for SPs consuming identity information.

There are currently three main areas where FIM is used. Figure 1.1 shows their characteristics side by side.

- First and foremost, there is **academia**. FIM enables every participant to use the federation's services with the identity run by their home institution. For example, students can enroll in online courses at other universities, researchers can access resources from other institutions, and all members can use services run by the federation for team management, communication, or documentation. Within academia, there are usually nationwide educational federations managing all national IdPs and SPs, international federations that combine many of those national federations to provide federated access across borders, and some project-specific federations.

- Secondly, FIM is used in selected **enterprise** environments. There it is used to access services of cooperating enterprises or commercial service providers that offer FIM to integrate their services with the enterprise's IT infrastructure more closely. Here, federations are usually smaller and consist of only a few participants.

- As a last example, FIM is also used by commercial services offered to the public, where it is usually labeled as a "social login" for **consumers**. In this case, it is used to lower the barrier to entry of an online service by not having to register a new account, but being able to link an already existing account. Those federations tend to consist of only one IdP and many SPs. They also allow anybody to add an SP that utilizes their IdP's specific federation. Services like this are run by many major Internet cooperation, e. g., Google, Facebook, or GitHub.

Neither of those applications of FIM is genuinely decentralized, as each requires participating users to have access to an IdP. In the three uses of FIM outlined above, this IdP access is naturally limited and acts as a method for making authorization decisions, as membership of IdP and SP in the same federation frequently enables most users of the IdPs to access all SPs. The identities provided by those IdPs are also not truly in the user's possession. Usually, after their contract with the IdP is discontinued they lose access to this identity and related services. This is more related to implicit authorization management than actual IDM.

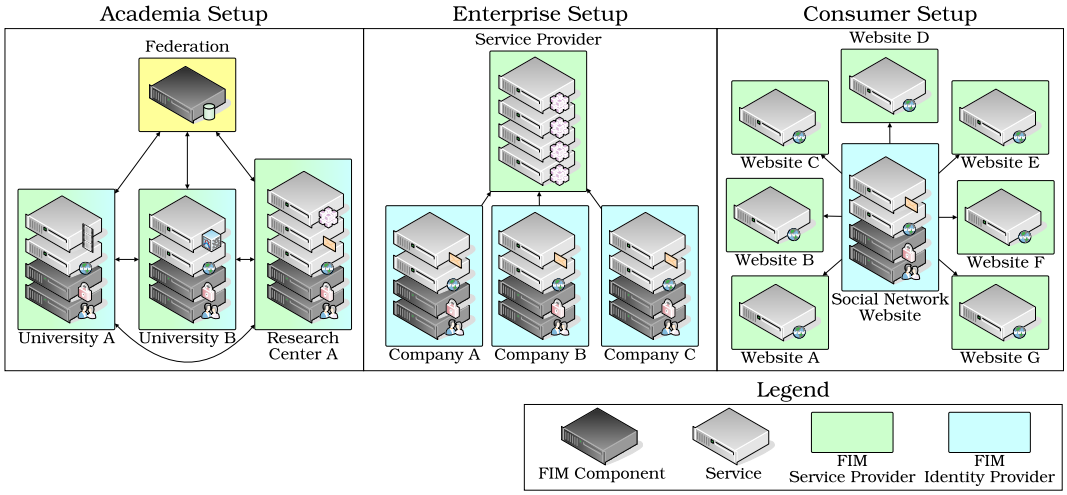

Figure 1.1: Different applications of federated identity management

On top of that, while the system seems decentralized initially, the federated structure imposes an inherent centralization. The party controlling the federation can decide who can join and dictate requirements for doing so. This is acceptable in some enterprise scenarios and probably also in academia but limits the system's usefulness on the Internet, where each large organization runs its own federation.

These shortcomings of FIM, alongside general privacy issues associated with the IdP being always part of every login, are partly responsible for developing user-centric identity management (UCIM). The idea of UCIM is to put the user in complete control of their identity and associated attributes by separating the management of attributes into dedicated attribute authorities (AAs). Attributes are managed by the user and confirmed by third parties (e. g., the university confirms the student's enrollment or completion of a course). This removes the strong connection between the user's home organization and their identity. However, it fails to do so completely, as it requires the user to choose an IdP that manages their identity data. As a result, an IdP may go out of service and remove all the user's identity and connected attributes.

The concept of self-sovereign identity management tries to combat this by mixing FIM and a decentralized database. To build a truly decentralized IDM system, the identity should not be associated with any particular provider but only with the entity it belongs to. A promising solution to host this identity database, which belongs to no one in particular and works by strict predefined rules, is blockchain and, more generally, DLT.

DLTs are very interesting as a basis for creating new IDM solutions. However, as with every new technology, it takes time to determine which of the many cryptocurrencies or smart contract networks will survive in the long run. Because of this, it is unrealistic to have people agree on a specific blockchain for future identity management. It is even possible that trying to find the ideal blockchain environment for identity management is impossible due to contrasting requirements, such as the time it takes for new attributes of an identity to propagate through the network, the cost of creating an identity, or the computational limits of small embedded hardware. Because of this, it is necessary to investigate the possibilities of creating a cross-blockchain framework or protocol that allows users to manage their identities on one specific blockchain and across many blockchains.

Moving identity management to the blockchain provides significant challenges for all participating parties. The following list emphasizes the three main participants in any IDM system.

- First and foremost, the user's view has to be considered. Without an identity provider managing the user's identity, the sole responsibility of doing this management lies with the user. Developing systems that allow users to manage their identities easily is one of the main requirements to get them to accept any new system.
- SPs with blockchain and traditional applications must ensure the identities conform to their requirements. Providing the proper technical tools and organizational frameworks to allow SPs to safely and reliably utilize attributes provided by the users themselves is vital for SPs' adoption of the system.
- IdPs have to adjust to no longer being able to control the user's identity fully. Systems where the user outsources complete control of their identity to an identity provider again should be discouraged. Instead, the IdP should be able to supplement its users' identities with additional information.

The individual DLT solutions usually differ in two major ways: the consensus algorithm and the ledger's design itself [13]. Besides the DLT's properties, the taxonomy created in [13], depicted in Figure 1.2, distinguishes between different components and attributes of crypto-economic design. Those crypto-economic design choices shape the interaction of the participants on the DLT.

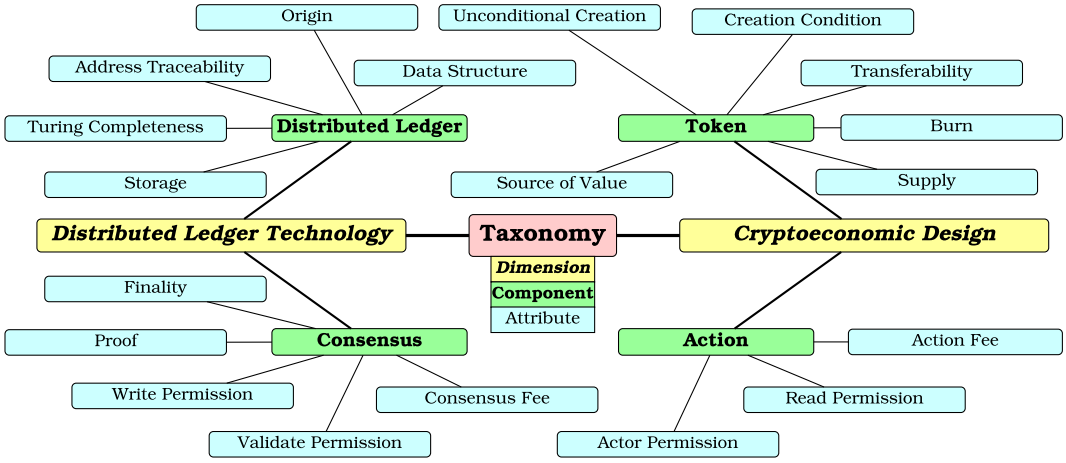

Figure 1.2: Taxonomy of DLT and blockchains [13]

The consensus-forming algorithm of the most popular blockchains follows a proof of work concept [66]. This requires the peers to invest a significant amount of processing power in order to generate a new block. The main criticism of this approach is that the work done and subsequent energy spent is used for nothing else than to form consensus [141]. Alternative systems include proof of stake or simpler systems like round robin. The practical implications of the chosen consensus algorithm are that it determines the resilience of the blockchain to withstand attacks by misbehaving peers.

The agreed-on rule set that is used for transaction validation determines which transactions are valid. While a simpler rule set limits the uses of the blockchain, it is easier to show that transactions are executed correctly and do not have any unintended (side) effects.

Identity management is currently only present in fundamental forms within blockchains. Accounts are usually identified by an address which is a long string of alphanumeric characters. This address is usually generated by calculating the hash of the corresponding public key. During transactions, the users can authenticate themselves by providing the public key and signing the transaction with their private key.

This concept has already been extended to include transactions that require signatures from multiple accounts or n of m accounts to prevent a single person from misusing a company's funds, for example [50]. Nevertheless, basic authentication boils down to controlling one or multiple private key files.

Further investigation reveals the following main difficulties of IDM with blockchains:

- Distributed ledgers are mostly public databases that anybody can access over the Internet. This is great because it allows anybody to participate easily, but a public database is not necessarily the best place for identity management and storing private information. Therefore, a method of storing or communicating private information securely between peers is needed.
- User accounts, particularly their private keys, are usually stored in so-called wallets. Access to these wallets needs to be secured. The most popular option to secure access to wallets are passwords or passphrases, which must be chosen sufficiently strong to prevent an attacker with access to the wallet from stealing the digital currency, property, or identity associated with the blockchain address.
- Inter-blockchain communication is a broad and mostly unexplored field of work, which can yield significant synergies between specialized blockchains.
- While the absolute decentralization DLT offers is excellent for avoiding vendor lock-in and censorship, many end users require help from customer support. As the ledger only works by achieving consensus with all peers, there is no easy way to recover funds, assets, or identities associated with an address to which a customer lost legitimate access (e. g., theft or loss of passphrase).

This work aims to develop a solution for self-sovereign identity management using DLT to improve UCIM and (dynamic) federated identity management. This process involves technical aspects for specific situations and a broader method description applicable to various applications. The following section describes the main focus of this work.

1.2 Research Questions

The research questions posed due to the motivation and objective of improving IAM using DLT are shown in the following list. In Section 7.2, the answers to those questions are provided.

1. **Self-sovereign Identity Management:** Which techniques enable self-sovereign identity management to provide usability, freedom, and privacy guarantees to users of web services and IoT devices, which conventional IAM systems (e. g., centralized identity management or federated identity management) cannot provide?
2. **Security Analysis:** How is the type of security offered by distributed ledgers (i. e., consensus on transaction ordering and prevention of double spending in a network of untrusted peers) useful to IAM systems in general and SSI in particular?

3. **Distributed Ledger Technology:** Which characteristics of a distributed ledger are required to be suitable for a privacy-conscious IAM system, and which of the currently available blockchains and distributed ledgers possess those characteristics?
4. **Entity ↔ Identity Binding:** How can a process be defined to securely and permanently bind a digital identity (digital twin) to a (real world) entity?
5. **Scalability**: Which methodologies enable a distributed ledger to manage the identities of many web services and a growing number of IoT devices?
6. **Inter-organizational Trust:** How can standardization enable organizations to use DLT to trust each other's assertions and data quality guarantees on an international scale?
7. **Interoperability:** Which organizational approaches can ensure that a distributed ledger for SSI can be used by many parties for different use cases and with different implementations?
8. **Reference Architecture:** How can an IAM system for SSI be implemented on a distributed ledger?
9. **Interface Definition:** How does an interface to decentralized identity management need to be designed to be able to use it as a replacement for classic IAM systems?
10. **IoT Device Compatibility:** Which adjustments must be made to use the self-sovereign IAM system on very constrained (i. e., connectivity, power usage, memory, or computational power) IoT devices?
11. **Electronic Identification (eID) Compatibility:** How can SSI be used in an international electronic identification (eID) system, like it is proposed by the *European Union* (EU)'s eIDAS?

1.3 Structure

To answer the research questions described in the previous Section 1.2, this work follows a structured approach described as follows and visualized in Figure 1.3.

As a starting point, Chapter 2 provides the basics of IAM and develops the scenarios used within this work. The basics introduce common IAM operations and concepts like FIM and SSI. The selection and definition of scenarios follow them. Those are analyzed and evaluated to gather requirements for a modern IDM solution. The scenarios are chosen from various currently relevant IAM applications, i. e., the Internet, IoT, and eID. All requirements gathered from the scenarios are grouped and weighted to form a set of criteria that can be referenced throughout this work.

Those requirements are used in Chapter 3 to compare state-of-the-art research, standards, and common solutions. Strengths and deficiencies identified by this process are then used to determine the key aspects that warrant or necessitate developing a new concept.

Chapter 4 describes the development of a new concept. It details the architecture, concepts, and processes necessary to build an SSI solution for various scenarios and applications. The concept aims to be adaptable to as many situations as possible by constantly referencing the scenarios of Chapter 2. As a result of this chapter, an actual implementation of the concept for a specific use case should be possible.

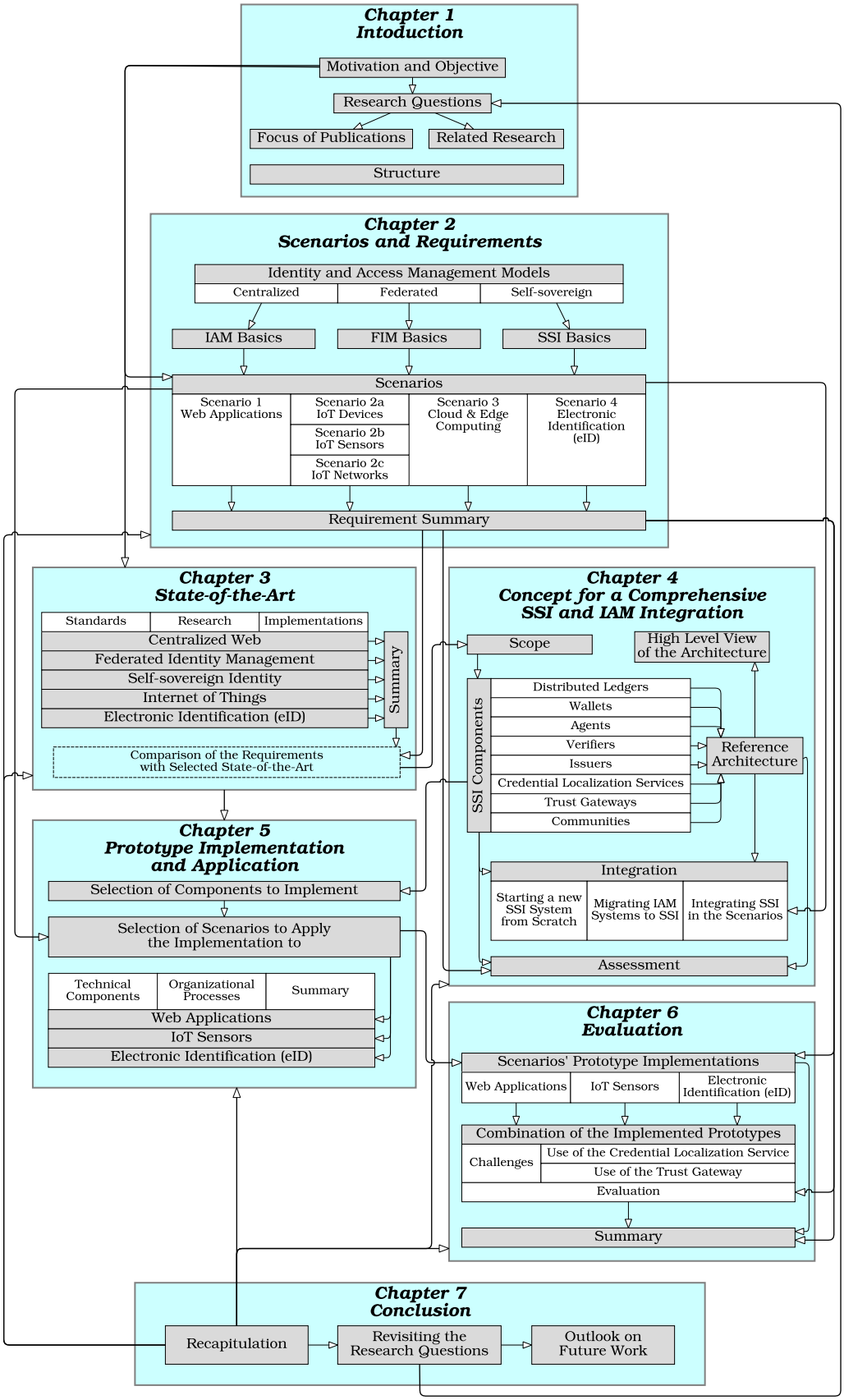

Figure 1.3: Structure of this work's chapters and sections

To test this, Chapter 5 explores multiple applications of the concept by developing prototypes for selected scenarios from Chapter 2. The prototypes are designed to test and validate the realization of SSI applications by following the concept's decisions. They are not intended to provide finished products or components thereof.

Chapter 6 explores the prototype implementations and compares the achieved results against the requirements of Chapter 2. Additionally, this chapter discusses the possibility of combining the individual scenarios and their prototypes into one connected SSI system.

In the end, Chapter 7 summarizes the process and highlights key achievements and potential weak points. This chapter also inspires further research into the topic of SSI.

1.4 Focus of Publications

Parts of this work have been published with the author's participation as conference or journal papers. The following list contains those publications with content related to this work in chronological order.

- Michael Grabatin et al. "Improving the Scalability of Identity Federations through Level of Assurance Management Automation". In: *9. DFN-Forum Kommunikationstechnologien*. Gesellschaft für Informatik eV, 2016
- Michael Grabatin and Wolfgang Hommel. "Blockchain-Basiertes Föderiertes Identity Management Am Beispiel von Ethereum Smart Contracts". In: *Sicherheit in Vernetzten Systemen 24. DFN-Konferenz, 2017, Hamburg, Februar 14-15, 2017*. Hamburg: DFN / Universität der Bundeswehr München, Fakultät für Informatik, INF 2 - Institut für Softwaretechnologie, Professur: Hommel, Wolfgang, 2017, B1–B16
- Wolfgang Hommel et al. "Level of Assurance Management Automation for Dynamic Identity Federations Based on Vectors of Trust". In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 39.3-4 (Jan. 2017). ISSN: 1865-8342, 0930-5157. DOI: 10.1515/pik-2016-0003
- Michael Grabatin and Wolfgang Hommel. "Reliability and Scalability Improvements to Identity Federations by Managing SAML Metadata with Distributed Ledger Technology". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei: IEEE, Apr. 2018, pp. 1–6. ISBN: 978-1-5386-3416-5. DOI: 10.1109/NOMS.2018.8406310
- Michael Grabatin, Wolfgang Hommel, and Michael Steinke. "Policy-Based Network and Security Management in Federated Service Infrastructures with Permissioned Blockchains". In: *Security in Computing and Communications*. Ed. by Sabu M. Thampi et al. Communications in Computer and Information Science. Springer Singapore, 2019, pp. 145–156. ISBN: 9789811358265
- Michael Grabatin et al. "A Matrix for Systematic Selection of Authentication Mechanisms in Challenging Healthcare Related Environments". In: *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*. SAT-CPS '21. New York, NY, USA: Association for Computing Machinery, Apr. 2021, pp. 88–97. ISBN: 978-1-4503-8319-6. DOI: 10.1145/3445969.3450424
- Michael Grabatin and Wolfgang Hommel. "Self-Sovereign Identity Management in Wireless Ad Hoc Mesh Networks". In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. May 2021, pp. 480–486

- Daniela Pöhn, Michael Grabatin, and Wolfgang Hommel. “eID and Self-Sovereign Identity Usage: An Overview”. In: *Electronics* 10.22 (Nov. 2021), p. 2811. ISSN: 2079-9292. DOI: 10.3390/electronics10222811
- Daniela Pöhn, Michael Grabatin, and Wolfgang Hommel. “Modeling the Threats to Self-Sovereign Identities”. In: *Open Identity Summit 2023*. Ed. by Heiko Roßnagel, Christian H. Schunck, and Jochen Günther. Bonn, Germany: Gesellschaft für Informatik e.V., 2023, pp. 85–96. DOI: 10.18420/OID2023_07

Chapter 2

Scenarios and Requirements

Contents

This chapter looks at different aspects of identity and access management and a selection of scenarios, which will form the base for establishing requirements for developing a modern, widely applicable, and secure IAM system. These scenarios are gathered by identifying tangible real-world examples from the following topics of interest and will be examined for how they could be implemented to support SSI.

- **Web Applications (Web Apps):** Web applications are still the primary means of content delivery on the Internet, and most of these web apps require some IAM [78]. For example, to access non-public data, view premium content, conduct purchases, or access a social network. Because there are so many web apps and different providers, the number of identities a regular user has to manage grows constantly. It is a source of common security problems like weak passwords and password re-use. Scenario 1 will examine a scenario for a web app that anybody can use over the Internet to post and receive data. IAM is used in this scenario to support user accounts and to specify who can access which data.
- **The Internet of Things (IoT):** Due to the rising number of "smart" and general IoT devices [184], the management of these has come into focus of many projects and research [30]. Some of this research also focuses on adopting blockchain for IoT [120]. The particular challenges of IAM of IoT devices are

their limitations, i. e., being battery-operated, with unreliable network connection, or having limited or no persistent storage. Those challenges, not found in desktops, notebooks, mobile phones, or servers, create additional aspects for SSI.

IoT-specific requirements are described in Scenarios 2a–c. Scenario 2a discusses the integration of IoT devices into the web app scenario. The IoT device should be able to record, display, and act according to data present on the web app. Scenario 2b adds requirements for more specialized IoT sensors characterized by minimal power requirements, which should be able to run on battery power for multiple months. Scenario 2c describes the infrastructure and IAM requirements for building an IoT network that covers an extended area. In this scenario, the goal is to increase the range of connected IoT sensors using a decentrally shared network infrastructure and discuss IAM-related challenges. This last scenario also contains aspects of the next topic of interest cloud & edge computing.

- **Cloud & Edge Computing:** Synergies of using centralized computation providers to run applications and tasks that require scalability have driven cloud computing [104]. Cloud computing has been so successful that many companies rather use cloud service providers to run their applications than hosting them on their own. This allows them to reduce costs for maintaining their own data center. To keep the benefits of cloud computing but also reduce latency and avert potential bandwidth limitations, an emerging trend is to move services from the cloud physically closer to the customer's location. This – so-called edge computing – migrates the customers' applications within the service provider's network of smaller but geographically highly distributed data centers to suitable locations [164]. To cover requirements from this topic, Scenario 3 extends the example application with cloud & edge computing-specific requirements for IAM. This includes on-demand storage and computation on shared resources near the IoT sensor or the application back-end.

- **Electronic Identification (eID) Systems:** The last and, due to particular national and international regulations and laws, probably the most difficult to evaluate topic of interest covers the inclusion of eID to use government-approved identities in the example. With increasing public pressure, governments and regulators are – and have been for some time – creating initiatives to make government and regulatory processes available to citizens using the Internet [143]. These initiatives range from e-voting to general e-government and digital health care, which all require strong guarantees of the security of the identity of individuals and organizations. These initiatives can provide the basis for all kinds of IAM applications, guaranteeing high data confidence levels. Combining the other scenarios and integrating the eID system based on a German eID project is shown in Scenario 4.

Before describing the scenarios in detail, a baseline for identity and access management systems is developed by analyzing common existing IAM architectures and systems. This analysis shows which options are available for building IAM systems and where individual strengths and weaknesses lie. It also defines important terms that may be used inconsistently across the literature. The regarded architectures and systems are from the following general categories:

- **Identity and access management (IAM)** in Section 2.2
- **Federated identity management (FIM)** in Section 2.3
- **Self-sovereign identity management (SSI)** in Section 2.4

Building on the base IAM architectures and systems, Sections 2.5.1–2.5.6 describe different scenarios from the topics of interest in which IAM plays a critical role. Those scenarios derive requirements for an IAM solution supporting the respective scenario.

In order to structure the determined requirements, they will be assigned to one of the following requirement categories, which are based on the goal of the respective requirement. This approach is based on the KAOS requirements acquisition and categorization method [42].

- **Satisfaction requirements** are required to satisfy entities' requests for key IAM operations (i. e., identification, authentication, or authorization).
- **Information requirements** allow entities to get information about other entities or the system's state. Those requirements are useful to discover entities and services, process attributes correctly, and join the system.
- **Consistency requirements** ensure that digital and physical parts of the system are kept in a consistent state, or this state can be re-established. This is especially important regarding digital twins.
- **Security requirements** aim to maintain the security of entities and the system. As such, they include requirements preserving confidentiality, integrity, and availability.
- **Data protection requirements** keep communication and information of and by entities protected and private. To differentiate from the security requirements category, data protection requirements are focused on data minimization, metadata avoidance, and tracking prevention.
- **Robustness requirements** help to recover from failures and restore the goals of the previous requirement categories.

Requirements from the security and data protection categories may overlap, especially regarding confidentiality and integrity, which in many cases could be assigned to either category. In those instances, requirements that fit the security category are assigned to the security category.

In addition to these categories, each requirement is assigned one of the following priorities that show the relative importance of the requirements:

- **Essential (1)** requirements are the most important requirements. Without those, the system will most likely not be usable at all.
- **Important (2)** requirements enable features or aspects of the system that are most commonly used and expected to be present.
- **Optional (3)** requirements facilitate "nice-to-have" features, only important in some less common situations.

The whole process of determining the baseline and gathering requirements from the topics of interest is depicted in Figure 2.1. The figure also shows where some scenarios are influenced by more than one topic.

After describing the concrete scenarios, aspects of IAM are analyzed and summarized to generalize and group the requirements in Section 2.6.

2.1 Identity and Access Management Models

An overview of established IAM models is provided in this section to set the scene applying the scenarios, which are described further down this chapter. The five IAM models shown have evolved and were adapted to current demands. The first

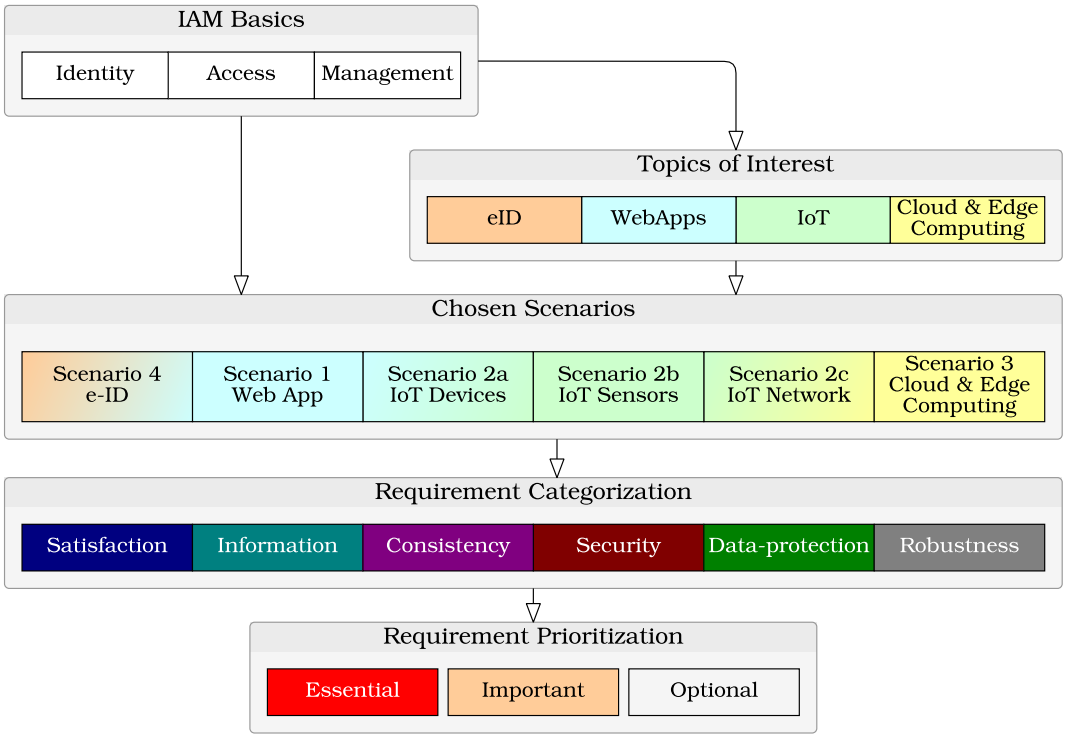

Figure 2.1: The process of deriving requirements from the chosen scenarios

four models are extensively described in [98]. At the same time, the last one has only recently come into focus due to the development of entirely decentralized identity management systems associated with different blockchain projects [134].

- **Application-centric identity management (ACIM)** closely integrates user account management with the application. The identity management system is developed in parallel with the application itself and thus is tailored precisely to the application's requirements and cannot be used outside the application. It is a quick and easy way to implement a basic identity management system. However, it becomes pretty challenging to maintain at scale, e. g., because migrating the user base to a new application is exceptionally cumbersome, and expanding the service portfolio requires users to re-register for each additional service.
- **Centralized identity management (CIM)** separates the identity management system from the application. Because of this, multiple applications can access and use the data from one identity management system simultaneously, making it easier to access multiple applications and allowing developers to use existing frameworks or solutions to integrate IAM into their applications. One drawback of this model is that the available methods and data structure might not suit the application ideally and requires awkward adaptions.
- **Federated identity management (FIM)** enhances CIM by facilitating identity information exchange in one organization's domain and between multiple trusting organizations. The trust between the organizations is usually based on legal agreements. This allows applications to outsource identity management to another entity called the IdP. Organizations and users re-use their existing accounts from IdPs to access SPs from multiple participating organizations. In some contexts, the SP is also called relying party (RP). This flexible

use of one identity also introduces a security issue, as a compromised account can be used to access many systems. Proper safeguards are, therefore, essential to detect and prevent misuse.

- **User-centric identity management (UCIM)** tries to mimic how identities in the real world are used by putting the user in the system's center and allowing them to have and choose between multiple identities based on the application or situation. This grants more flexibility to the user because they can choose the IdP freely and decide which personal information to share with which IdP. This freedom of choice introduces additional complexity because some services might require specific assertions, e. g., the name or address of the user, which cannot reliably be provided by all IdPs, the user could choose. Knowing which IdP to use in which situation is the user's duty, just like knowing when to show a driver's license or gym membership card in the real world.

- **Self-sovereign identity management (SSI)** removes the concept of an organization that acts as IdP for the user. In this model, users can create and act as IdP using a decentralized platform for as many identities as they like. They have complete control over which identities they use to access an application, and the IdP cannot surveil which services are accessed. Because some services might need guarantees about some attributes of a self-sovereign identity, these may be attested by a third party with a trust relationship with the application relying on the attribute's value. These attestations can be linked to one identity by the user to show them to the requesting service.

In order to keep the list of management models short, within this work, the list is reduced to the three most relevant models: centralized identity management, federated identity management, and self-sovereign identity management.

The revised list no longer contains ACIM because, in practice, it has been superseded and substituted by CIM [78]. Implementing IAM only for one application is an antiquated model, and it is nowadays more likely that an application developer will use an existing CIM solution in combination with their application.

The second model omitted from the list, the UCIM, has been removed because while it has been around for some time, it has not caught on in practice. Many of its novel ideas have been inherited by the SSI without the conflict of trying to put the user in total control but having only one (or very limited) choice of IdPs.

Each of the remaining models will be used to gather and compare requirements for those IDM solutions. This collection will serve to collect a list of common identity management system requirements and later be a reference to identify areas of change when transitioning from one of the management solutions to another.

2.1.1 Centralized Identity Management

While most IAM systems in private home environments are, in fact, CIM-based, it is usually the case that each device or service runs its own CIM system. This results in every computer, mobile device, or IoT device having separate accounts and passwords, and updating an account on one system does not affect any other system. On the web, CIM is familiar with many larger web services requiring their username and password combination to access multiple services and applications. For example, a Google account, can access the Google Play store, YouTube, Google Drive, and multiple other Google services. The same is true for many other web-based service providers.

The problems created by storing credentials and attributes centrally are numerous. Across the different systems, such information is stored redundantly without any method for synchronization, verification, or reconciliation. No authoritative source is guaranteed to provide the most up-to-date information. For personal accounts on the Internet, the personal effort required to synchronize personal attributes like name, home address, email address, or banking information between the various online services used today is enormous.

Within organizations where the different services, like email, file sharing, or a wiki, are centralized, access management gets very complicated. A central repository managing all employees' identities reduces management overhead. It prevents situations where an employee joins or leaves the organization, and each service administrator has to create or delete the user's account. Especially with an employee leaving a company, if an administrator forgets to revoke the employee's access, this can result in unauthorized access. This is why most enterprises and organizations have moved from application-specific identity management systems to CIM systems.

2.1.2 Federated Identity Management

FIM removes users' per-service or per-organization identity and facilitates sharing of identities between organizations. Instead, a universal identity and attributes are stored with an IdP that may or may not offer other services directly. This IdP can then be used by other SPs to authenticate and, depending on decisions based on the attributes transmitted by the IdP, authorize the user to use their service. FIM is described in more detail in Section 2.3.

2.1.3 Self-Sovereign Identity Management

SSI combines most aspects of UCIM with an easy to set up IdP. The main criticism of UCIM is that while it is theoretically possible for everyone to create and run their own IdP, it is not realistically feasible as it requires too much technical knowledge. SSI uses the concept of decentralized computing and data storage. Both are provided by the DLT implementations as a way for everybody to act as their own IdP. A more detailed look at SSI is provided in Section 2.4.

2.2 Identity and Access Management Basics

The following section describes IAM basics and defines the corresponding terminology. As a start, Definition 1 specifies the term IDM.

> **Definition 1 (Identity Management)** *"Processes and policies involved in managing the lifecycle and value, type and optional metadata of attributes [...] in identities [...] known in a particular domain [...]" [95]*

The addition of access management to IDM is called IAM. This term encompasses, more broadly, any process or technique related to identification, authentication, and authorization, as described in Definition 2. Frequently, however, IDM and IAM are used almost synonymously.

> **Definition 2 (Identity and Access Management)** *The technical and organizational framework for identifying, authenticating, and authorizing access to services or systems.*

The most commonly thought of entities for IAM systems are natural persons. They must be identified, authenticated, and authorized frequently as part of everyday tasks. A common task for IDM is the reliable identification of natural persons

within different contexts. This task is further complicated because natural persons usually have high mobility (i. e., they travel to different locations and use different devices) and generally do not have any standardized way of identifying for different services. While biometric identification (and authentication) is rising [17], probably due to an increasing number of consumer mobile devices being equipped with fingerprint readers or facial recognition software, the most used identification schema for individuals is still based on usernames and passwords [173]. Within this work, a person is always regarded to be a natural person, i. e., any physical individual human being.

Definition 3 (Person) *A natural person, i. e., any physical individual human being.*

Personal identities are a well-researched topic, and through literature, they are usually based on the seven laws of identity by Kim Cameron [29]. These laws formulate criteria that influence the creation of a universal identity layer for the Internet and are summarized in the following list:

1. **User Control and Consent:** The system must only reveal the user's personal information with their consent.
2. **Minimal Disclosure for a Constrained Use:** When information about a user is revealed, it must be within a defined context of who can read that information and how long it will be retained.
3. **Justifiable Parties:** The IAM system must shield the user from disclosing information to parties that have no justifiable necessity to request it.
4. **Directed Identity:** Identities must be either "omnidirectional" (i. e., public) or "unidirectional" (i. e., private). Public identities are useful for organizations and all identities that want to be discovered. In contrast, private identities are only visible to the entity they are disclosed to and cannot be linked across different entities.
5. **Pluralism of Operators and Technologies:** The identity system must not limit the number of identity providers or the underlying technologies.
6. **Human Integration:** Communication between the human and the computer must be designed to limit or prevent attacks on the user's identity.
7. **Consistent Experience Across Contexts:** An universally used identity system must provide the same or similar user experience across all use cases.

A key concept of society is collaboration between multiple persons. They form organizations that set out to achieve some common goal, and in doing so, the organizations can also represent their members. So to do business with other organizations or show a person's affiliation to an organization, the organization must also be identifiable.

Definition 4 (Organization) *Any legal person, as well as groups of persons that are formed to pursue a common goal, can form an organization.*

The main difference between the identity of a person and that of an organization is that the identified "object" is tangible in the real world for persons. In contrast, an organization may only exist on paper or through shared understanding. An entity can be anything that can be identified individually within a given context.

Definition 5 (Entity) *Any distinctly identifiable item, person, or organization within a domain [95].*

In computer science, such entities may often be websites, IoT devices, or computers. This covers anything that could be inventoried, accessed, or addressed. Being able to identify an entity is the basis for all further IAM, as authentication and authorization would be meaningless if they are not connected to a specific entity.

An identity is defined as a "set of attributes [...] related to an entity" [95]. This standard's definition notes explicitly that an entity may possess multiple identities and that multiple entities may share the same identity. The latter property of that definition is somewhat counter-intuitive in everyday use, which is why, in this work "identity" is used as an "identifier", "unique identity", or "distinguished identity" as defined by [95], which specifies that an identity can be used to distinguish an entity, within a specified domain unambiguously.

Definition 6 (Identity) *An "attribute or set of attributes [...] that uniquely characterizes an [...] [entity] [...] in a domain" [95].*

This attribute set depends on the domain or context in which the entity is identified. For government forms and applications, the attributes that uniquely define a citizen are first name, last name, date of birth, place of birth, tax id, and probably the registered address. Within the domain of personal friends or colleagues, it is usually just the first name or specific nicknames if collisions exist. Instant messaging applications often use the mobile phone number as an attribute to identify and connect the users.

These examples show that the chosen set of attributes necessary for identifying an entity is very domain specific and needs to be chosen to minimize possible collisions, i. e., that two different entities are presumed to have the same identity while still being practical to use.

Identifying an entity is only the first step in securing IAM. Afterward, the entity's identity needs to be verified. The kind of verification necessary, again, depends heavily on the domain, use case, and level of assurance (LoA) required. Authentication may be done by presenting some form of government ID, showing a certificate, or providing the password corresponding to the username.

After identifying and authenticating an entity, the next important step is to define what this entity can do, i. e., what kind of access it has. This is the process of authorization. The primary subjects of access management are usually users, and ISO/IEC 27002:2013 defines the objective of user access management: "To ensure authorized user access and to prevent unauthorized access to information systems." [96].

A general life cycle for IAM was described by [11]. It is not specific to any concrete IAM implementation and can be implemented to some extent by most IAM systems. It showcases the basic steps and procedures necessary to implement an IAM system. Figure 2.2 shows a partitioned and slightly shortened version of this lifecycle.

The deep blue "main path" in Figure 2.2 contains the shortest possible way through the lifecycle and thus represents the minimal feature set of an IAM system. This path contains the steps necessary to register an identity, associate credentials with this identity to perform authentication and define access permissions. To complete the cycle, it also contains removing credentials and the identity itself. The teal colored "administrative functions" represent ubiquitous functions for changing access permissions for an identity. The light blue "convenience functions" are optional functions consisting of mainly pause and undo functions that temporarily disable and later restore access for the identity.

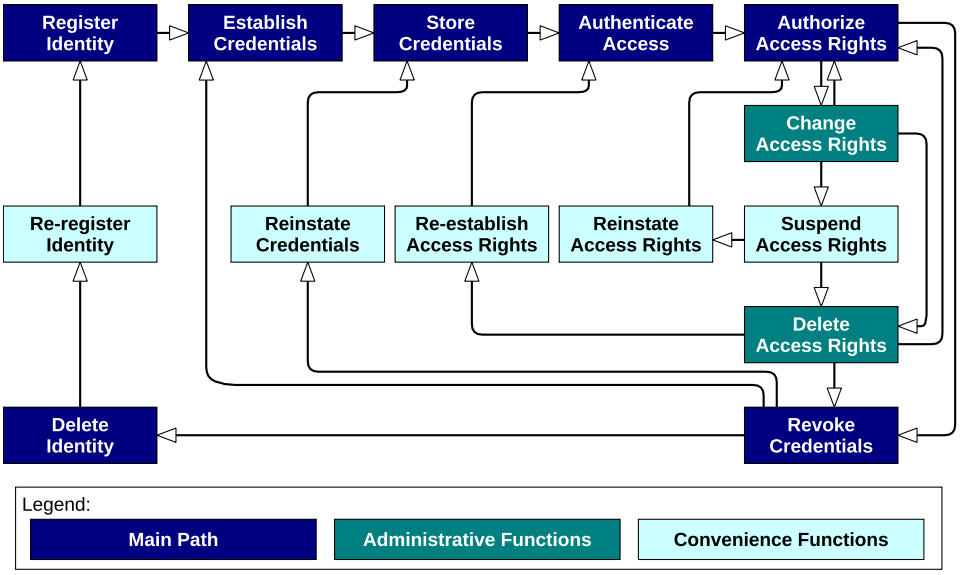

Figure 2.2: IAM life cycle based on [11]

2.3 Federated Identity Management Basics

The following section describes FIM basics and defines the corresponding terminology. It also shows prior research into the management and architecture of FIM.

> **Definition 7 (Federated Identity Management)** *An IAM framework that enables services or systems from multiple organizations to authenticate entities of cooperating organizations, which form an identity federation [32].*

FIM can be used where authentication and authorization infrastructure should be usable between multiple organizational domains. It allows users from one organization to access SPs at another organization without registering a new account with the remote organization. Instead, they can use their home organization's account through the organization's IdP. This results in a general reduction of redundantly storing identity attributes (e. g., name, email, etc.) with multiple services and having only to update changed information at the IdP. Consequently, the SPs benefit from more up-to-date identity information that is usually transmitted from the IdP with every authentication. This system requires the participating organizations to mutually agree on the organizational terms and conditions of the cooperation as well as on technical specifications for exchanging the required data.

> **Definition 8 (Identity Provider)** *A system that provides identifying and authentication services of registered entities to other services within a specified infrastructure or domain [32].*

> **Definition 9 (Service Provider)** *A system that provides services to entities from a defined infrastructure or domain [32].*

One commonly used technical FIM workflow example is the SAML 2.0 Web SSO Schema [38] depicted in Figure 2.3. This example shows the workflow of redirecting the user's request through the web browser from the SP to the IdP, performing the authentication, and returning the result to the SP through another redirect.

This or similar workflows allow the consolidation of identity information with one service. The SPs can focus on service delivery but must trust the identity information provided by the IdPs.

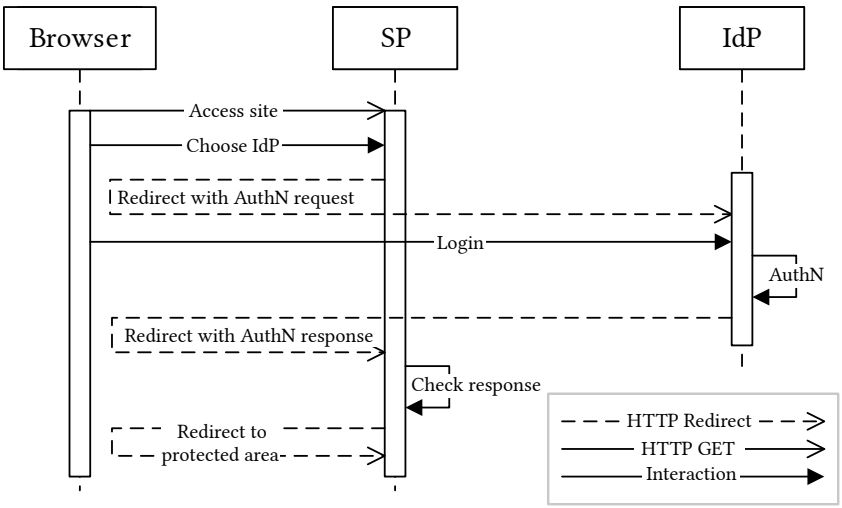

Figure 2.3: SAML-Web-SSO-Schema [38]

The technical protocols that facilitate such sharing of identity information need to be supported by an organizational framework that establishes trust between SPs, IdPs, and users. This organizational trust is much harder to accomplish than purely technical trust because organizational trust usually involves legal contracts, risk assessments, and business decisions that specify the quality and actuality of identity data, determine which IAM parts can be outsourced to an IdP, and whether giving up complete control over the IAM process is feasible as well as what SPs can use the received data for.

The IT service management processes necessary for building, managing, and securing an identity federation have been researched by Wolfgang Hommel [82]. Integrating FIM services with existing IAM infrastructure systematically is a key component of the architecture designed in this work. The results of Hommel's dissertation are used in this work to compare regular FIM architectures to the newly established SSI architectures and to determine possible avenues to move IAM systems to SSI.

Further research into automation, scalability, trust, and interoperability of dynamic identity federations has been done by Daniela Pöhn [142]. Those dynamic identity federations use a trusted third party as a broker to create dynamic and on-demand federations. The benefit of dynamically building smaller federations is the reduced set of organizations that need to agree on common definitions and interfaces. This makes those dynamic federations more capable, as there are fewer constraints. The trusted third party can negotiate many of the (technical) aspects automatically. The results of this work are used to compare the dynamic identity federation architecture that uses a central trusted third party to the SSI architecture without any institutionalized trust.

2.4 Self-sovereign Identity Basics

As a relatively new IAM framework, SSI is currently not used in any wide-scale application. However, there are multiple prototypes and testbeds, for example, from the Sovrin Foundation, the Linux Foundation's Hyperledger Indy, or the formerly uPort now Serto.

Definition 10 (Self-sovereign Identity Management) *Is an IAM framework based on multiple independent attribute sources providing verifiable credentials to entities that collect them in a wallet to augment their individually owned identity. To prove an entity's identity or attributes, the relevant credentials from the personal set are chosen and presented to other entities.*

SSI is closely related to UCIM but executes the idea of putting the user at the center of IAM more thoroughly. With UCIM, an IdP is still responsible for managing the user's attributes, but this IdP is not under the user's control. This constellation somewhat limits how far the user is actually at the center of this IAM framework. With SSI, the users are their own IdPs.

This basic premise is achieved by decentralizing the IdPs and allowing each entity to act as its own IdP in a peer-to-peer network. It does not imply that any entity can credibly assume just about any attributes associated with its identity. Attributes are collected from relevant parties for the use case and can provide the entity with a verifiable credential. Other entities can verify the origin and content of the credential when presented to them by the entity. This allows more natural architectures of IAM to be represented digitally.

This IAM architecture is achieved by removing strict hierarchies that dominated previous IAM architectures and frameworks. In the real world, persons, devices, and organizations are identified differently depending on the transaction's context. An example of this decentralized IAM architecture used in everyday life is shown in Figure 2.4. One person can assume multiple personas and act in different roles. Within those personas, different credentials are used to show or prove identity and personal attributes (e. g., name, address, citizenship, or associations) to other entities. The credentials are issued by an authority but are usually carried by the person they describe. This drastically differs from common IAM approaches, where the issuing entity or some other third party usually must be contacted for each authentication. The credential issuing and validating entity determine the scope where the credential can and should be used. However, both entities must not necessarily agree on the same scope, as shown in the later example of driving licenses.

An employee ID, for example, can be used at the workplace that issued the ID but also with other branches of the company, with contractors and cooperating companies, or with customers. Outside this sphere, the employee ID is more or less useless because nobody can tell a real ID from a fake one. However, within the set of entities that know how to read the employee ID, it is a very straightforward way to assert one's position in the company.

For international travel, government-issued passports are usually used as identification documents. Because of the unique requirements of being readable worldwide, providing robust identification and authentication, being hard to forge, and thus being very valuable, they are usually not used outside of traveling through borders. Given the high reliability and trustworthiness of the document, it could be used almost everywhere to establish one's identity. Nevertheless, it lacks information like company association, which an employee ID can better portray.

An example of a credential usually only recognized by the issuing authority is a sports club or gym membership card. It cannot be used for anything other than showing that a person is a member, but it does so efficiently. If a passport had to be shown to evaluate whether a person is a member of a gym or not, the person's name from the passport would need to be matched against a database of the gyms' members' names. This overhead is usually omitted, acknowledging a significantly reduced forge resistance.

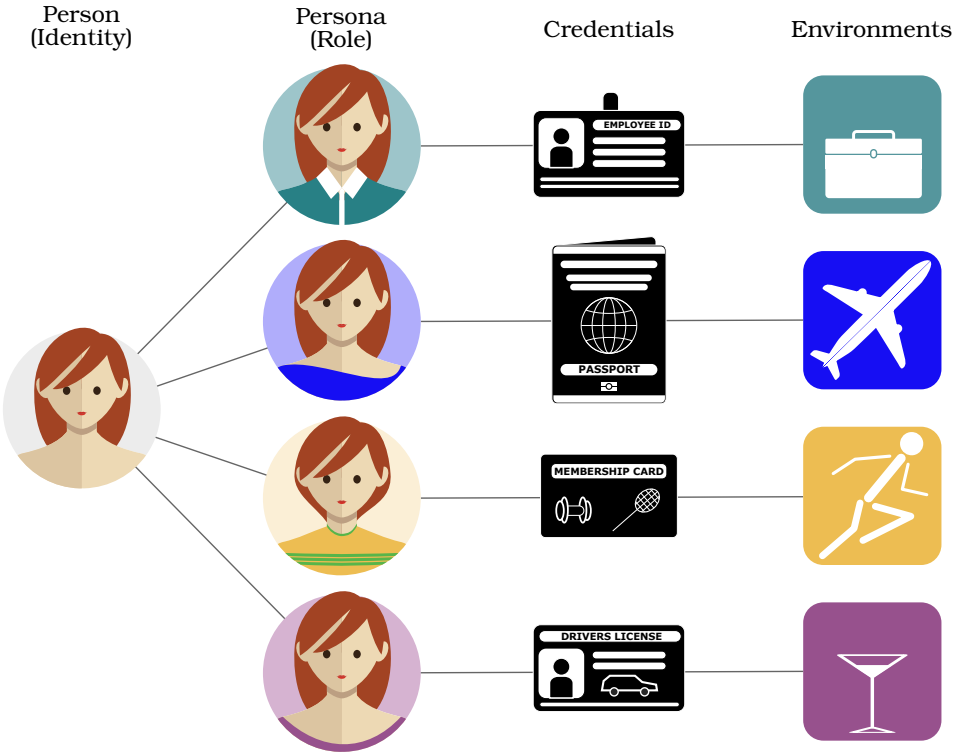

Figure 2.4: Real-world identity with different persona and their respective credentials used in different environments

Until now, the examples have shown scenarios where (usually) the credential issuer and the credential verifier are the same entity or related entities that know about each other's existence. The last example shows using a driver's license to get authorization to purchase alcoholic beverages by showing sufficient age. This example describes a situation where an issuing authority is needed that is universally trusted by bars to establish the person's age. At the same time, using a driver's license for this purpose is not its intended purpose. This additional use case has been established over time.

SSI aims to recreate a similar architecture, but online and with more granular control. Especially the last example of using a driver's license to verify one's age shows a problem where the person is showing too many attributes (e. g., name, address, driving class, date of birth) where the person's age and an identifying photo would be sufficient.

Key challenges of the approach taken by SSI remain similar to those of FIM: The SP has to trust the IdP to have performed the authentication correctly and that each transmitted attribute of the user is correct and up-to-date. While for FIM scenarios, this challenge is met by forming formal federations, a similar but scalable and decentral solution must be found for SSI. The Internet's inherent international infrastructure complicates this further with many varying cultural mindsets, legal frameworks, and digitalization maturity.

Furthermore, putting the user in the position of an IdP imposes many problems that IdP services usually face solely onto the user. This may create an environment or solution that risks that common problems are blamed on users' actions (or lack thereof) – for example, the common problem of forgetting or losing credentials.

Where there used to be a support page or phone number that would help to restore users' access to their account with the IdP, now nobody but the user is responsible for the individual IdP and the individual restore process.

Further challenges will be discussed in the following sections, which describe the scenarios in detail.

2.5 Scenarios

In this section, the scenarios mentioned and motivated by different topics of interest at the beginning of Chapter 2 and Figure 2.1 are described in more detail. For each scenario, a list of requirements is deduced, categorized, and prioritized according to the schema described at the beginning of Chapter 2.

2.5.1 Scenario 1: Web Applications

The first scenario discusses a simple web application implementing a publish subscribe messaging system for data processing and sharing. This or a similar application structure can be found in many real-world applications, like social networks, forums, news sites, video streaming platforms, or – to some extent – even blockchains like Ethereum. This example application will focus on publishing text data, as more complex applications that may require image processing or video streaming do not significantly add to the IAM requirements.

The IAM requirements of this web app are pretty basic at first glance. Figure 2.5 shows the general architecture and the user's interaction possibilities: Users can register an account, create channels, and publish data to them. Other users subscribe to those channels and receive the published data. The user can also configure some predefined or custom processing steps that can be used to validate, transform, or aggregate the data before it is published. The channels where data is published can either be public, meaning that anybody can subscribe to them or private so that only selected other users can receive data through them. It should also be possible to open channels to multiple publishers so that multiple users can post new data to a channel. Those features require some permission and user management.

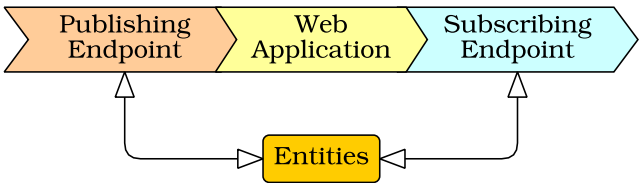

Figure 2.5: General architecture of the publish subscribe web app

Closer inspection of those seemingly trivial operations yields further, less basic questions that need to be answered by the service provider and its users. Starting with registering an account, the service provider needs to determine what information is required from the user. The more information known about a user, the easier it is to connect the users and tailor the service to their needs. However, requesting more information increases the barrier for users to join and thus results in fewer users or more unreliable data. Asking for the "right" information is difficult with a platform as generic as described here: What information about the user is really relevant to provide the service adequately and determine user interests?

This question may not be answerable when the user first connects to the service and also depends on the functions the user would like to use. The kind of data they add and read to and from the platform may also be heavily influenced by the type of data that is already available and will develop over time. To register the user, the service must recognize the user as an individual connecting to the service on repeated visits. If necessary, this registered identity can be augmented with more information (i. e., payment information, topics of interest, or other personal information) later on.

To authenticate a registered user, the SP and the user establish credentials (e. g., a specific password or certificate). Those credentials used by entities to authenticate with the platform need to be tailored to the entities. This is especially apparent because many IoT devices should be able to publish data to and retrieve data from the platform. Additionally, web services and regular people must access the platform through a web browser. Those categories of connecting entities are suited to different kinds of credentials.

While there are disadvantages, regular persons might default to using a password as their credential. Alternatively, they might use systems like *OpenID Connect* (OIDC) or security assertion markup language (SAML) as their authentication credential. IoT systems and web services might use pre-shared keys (PSKs) or certificate-based credentials. Supporting all those and potentially new, developing options poses a significant challenge. This is especially true because not all credential systems offer the same level of security.

In some cases, the need to support multi-factor authentication (MFA) or at least two-factor authentication (2FA) exists to enhance the authentication's strength. MFA can be implemented via various systems, e. g., interactive challenge-response style systems, smart cards, or one-time passwords. The need for MFA can also be determined by the service during the authentication, depending on a risk score based on metadata or actions performed by the authenticating entity.

As authentication credentials may also be lost or even compromised, there needs to be a way to occasionally revoke old and establish new ones. This is usually done by revoking the old credential and establishing a new one, preferably using a third credential or another out-of-band mechanism.

Access and authorization decisions must be handled by the service to ensure each entity can only access the data it is allowed to access. These restrictions must be managed, enforced, and checked to prevent any unintended disclosure of data.

The management of access permissions requires users to be able to display and set those access permissions in a clearly structured way. To keep this permission management assessable and useful for the users, the granularity and method of assigning permissions must also be considered. On the one hand, if there are too few options, the resulting options may be too limited to enact the required restrictions. On the other hand, if the options are very granular, the risk of unintended configurations due to confusion rises.

When an entity's account is deleted, multiple challenges are associated with treating the associated data. The service provider must determine which information needs to be kept for record-keeping processes (e. g., billing, payments, taxation, or audits) and which information, especially personal information, can and must be deleted immediately. Suppose parts of the entity's account data are deleted. In that case, care must be taken to ensure that this will not break references within the data that needs to be kept (e. g., deleting the entities name and user ID from the database but indexing the entity's bills by that ID, making it hard to find bills associated with the entity). Keeping links within the data intact is obviously more challenging if the data is gathered and linked from multiple and external sources,

i. e., IdPs. As those external sources may disappear quickly, without warning, and permanently there need to be contingency plans to prevent or mediate the data loss.

When the service provider stores personal data and operates within the European Union or has customers based there, the service provider must adhere to the *General Data Protection Regulation* (GDPR)[41]. The individual requirements of GDPR and its corresponding local legislation cannot be analyzed from a legal point of view in this work. Any discussion must not be considered legal advice. To ensure compliance in each case, consulting a lawyer is advisable. Nevertheless, the main effects of this regulation are outlined based on the scenario.

The central accomplishment of the GDPR is the specification of data subject rights. Those include that the service provider must explicitly get permission to collect customers' personal information. The service provider must also tell the customers how their personal data is being used and, upon request, must be able to provide them with all data stored about them. Additionally, on request of the user, the service provider must also provide ways to correct wrong personal information, delete personal information, restrict the usage of personal information, object to a previously given consent about using personal information, and provide information for submitting complaints.

A key to satisfying those data subject rights is always knowing what personal data is stored and where it is stored. This is especially important if personal data is stored in attributes that are not expected to contain personal data (e. g., a status message) and includes any copies of the data that may exist without being recognized as personal data storage (e. g., in backups).

Adherence to GDPR or other privacy legislation is integral to any web application that wants to avoid litigation. Both the user and the company providing a service should be able to determine who they are sharing what information with quickly. In the context of privacy protection, privacy by design and privacy by default are principles described by various legislative rule sets [67], including the GDPR. Privacy by design is also part of practical publications, focusing on technical and organizational measures for protecting private information [105]. The core principles of privacy by design, which include privacy by default, as described by [31] are: proactively building an architecture for IT systems that transparently protects and respects the individual users' privacy needs from start to finish without the user having to change or review any settings or having to relinquish system functionality.

To incorporate those principles into the application described in this scenario, the first step is to determine what kind of users' private data is recorded and to specify the purpose of the collection. Like many other platforms on the web, the application requires an email address, username, and authentication credentials (e. g., a password) for registration. All three values are personal data and must be kept private by the application. If the user wants to share their username or email address, they have to manually enable this option (e. g., for others to be able to search for the user and contact them). Implicitly the user also shares their username when creating a public channel. In this case, the user must be made aware that anybody can see a public channel and that it is associated with their username.

If the service can be offered without ever storing any personal information about the entity, many previously described problems can be avoided. In this case, the service provider must decide if offering account registration without personally identifiable information (PII) is a feasible option. In general, this decision may be influenced by certain information that the service provider must gather by law, especially if the service charges the customer. Usually, though, each service re-

quires at least an email address for registration, which is considered personally identifiable information. As a result, providing anonymous access to a user while still being able to contact them contradicts itself.

Some applications need support for multiple entities publishing data to the same channel. In this case, those entities need to be identified and managed by an administrative user of the channel. Additionally, the entities must be authenticated and authorized before they can publish data to a channel.

One (real world) entity also commonly represents multiple personalities (e. g., private and professional life). This entity would usually need multiple accounts to represent the two (or more) personalities. Multiple accounts add to the overall complexity and may lead to increased support efforts because credentials to the account used less often may be lost regularly. The goal of this system is to allow an entity to identify itself and its current personality using the identity and possibly the same credentials.

The following sections contain an abstract collection of the IAM requirements from the described scenario. Each requirement is given a short identifier, a three-letter abbreviation of the requirement's category and a running number. Additionally, each requirement is named with a short descriptive title, followed by a short description and a prioritization (1–3 where lower is more relevant).

2.5.1.1 Satisfaction Requirements

SAT1 **Authentication:** The identification of an entity needs to be secured by authenticating it. This authentication must prevent entities from impersonating another entity without the service provider being able to detect this attempt. This requirement is essential (1) to prevent the impersonation of entities.

SAT2 **Authorization:** Allows the service to specify conditions the identity must meet to use parts of the service. The most basic authorization is being able to authenticate with the service, but more complex models can also be used. This requirement is essential (1) to prevent misuse of the service.

SAT3 **Identification:** The service needs to be able to identify entities to be able to map them to the application's accounts and resources correctly. For example, the user's data must be stored associated with their account and channels in the scenario. This requirement is essential (1), as matching entities to their accounts and resources is necessary for all non-public operations taken by an entity.

SAT4 **Identity provisioning:** There must be a process in place to bind an entity's identity to a service's account. This process must ensure the uniqueness of the identity within the service provider's scope and that all the necessary attributes of the entity are available. This requirement is essential (1) to provide targeted services that rely on identifying entities.

2.5.1.2 Information Requirements

INF1 **Credential establishment:** As part of the registration, a process is required to establish the credentials used by the entity to authenticate with the service. This requirement is essential (1), as without established credentials, authentication is impossible.

INF2 **Documentation:** The protocols and methods to facilitate IAM functions must be documented clearly. This documentation must include the used encryption, signature, and integrity-checking technology. This requirement is important (2) to allow and encourage independent reviews and implementations.

2.5.1.3 Consistency Requirements

Identity de-provisioning: Corresponding to the provisioning process, the entity and its identity must also be de-provisioned. All data no longer necessary must be removed, and subsequent access of the identity must be prevented. This requirement is essential (1) to comply with data protection legislation and remove unnecessary data and access. CON1

Credential recovery: As credentials may be lost or destroyed, there needs to be a way of re-establishing credentials. This requirement is important (2), as there may be cases where recovery is not desired or necessary. Generally, though, most systems will need it. CON2

2.5.1.4 Security Requirements

Access controls: Access to resources must be manageable. This includes access to the application or platform in general and to individual parts and actions. This requirement is essential (1) to ensure privacy expectations and prevent misuse of an application. SEC1

Credential revocation: If a credential is lost or deemed compromised, it needs to be revoked as soon as possible to prevent any unauthorized use. This requirement is essential (1), to ensure lost or stolen credentials cannot be used to gain authorization. SEC2

Mutual authentication: As part of every authentication process, not only should the user authenticate to a server, but the server should authenticate to the user first. This requirement is essential (1) to prevent the user from disclosing identity information to another party (i. e., a human-in-the-middle (MITM) attack or an impersonating service). SEC3

Security by default: Following the security by design principles leads to a product shipped with a secure configuration by default (e. g., does not use unsafe protocols, contain any hard-coded passwords or backdoors). This requirement is essential (1), as any system that is not developed with security by design principles will not be considered by many. SEC4

Security by design: As required by multiple software design and development standards, especially an IAM system needs to take security seriously within the design process. The specific security needs in operation may not always be known during the design, which requires the resulting product to be configurable enough to be as secure as necessary for the actual use case. This requirement is essential (1), as security by design is demanded for any new technology or products. SEC5

Multi-factor authentication: To enhance the security of an authentication process, the system might require a second or more factors to complete authentication. This requirement is important (2), as it is a commonly used feature. However, it is not always necessary. SEC6

2.5.1.5 Data protection Requirements

Privacy by default: The system's default configuration has to be as privacy-preserving as possible. This requirement is essential (1) to ensure the system's setup is easy and has no pitfalls where there is mandatory configuration needed to secure the user's privacy. DAT1

Privacy by design: Separated from the GDPR requirement, any IAM must be designed with the privacy of its users in mind. This requirement is essential (1) to build trust in the system. DAT2

DAT3 **GDPR:** A IAM system has to be compatible with the rules set forth in the European GDPR. This requirement is important (2), as it will be necessary in most but not all cases.

DAT4 **Multiple identities:** In the scenario, there may be the need for a user to send or receive data in different contexts, i. e., the example shows a private and a corporate context. To separate those contexts, the user has to be able to act as two different identities. This requirement is optional (3), as not explicitly supporting different identity contexts can be worked around.

2.5.1.6 Robustness Requirements

ROB1 **Reliability:** The IAM system must be reliable, i. e., resilient to failures. Situations must be prevented where (administrative) users or entities are locked out of a system because of a failure in the IAM system or its infrastructure. This requirement is essential (1), as IAM processes serve as the basis of most other processes.

ROB2 **Accessibility:** Related to the approachability requirement, the service should also be easily accessible. This encompasses (within reason) being able to use the service on any desktop computer or mobile device independent of operating system or hardware. This requirement is important (2), as some combinations of devices may not always be feasible to support.

ROB3 **Approachability:** A general requirement for all services is having a low barrier to entry. This ensures that more customers actually decide to use the service, especially if the service is non-essential or has many competitors. As a result, the IAM necessary for the service should not impede approachability. This requirement is important (2).

ROB4 **Usability:** Any solution should be less or equally complex as existing IAM solutions for end-users and administrators. This requirement is important (2) to prevent using an over-engineered jack-of-all-trades solution as this will interfere with acceptance, reliability, and security.

2.5.2 Scenario 2a: IoT Devices

The introduction of IoT technology to more and more homes and workplaces has led to a rise in "smart" devices and appliances that are installed nearly everywhere. Controlling physical applications (i. e., IoT devices) from the Internet is one of the key components of many modern "smart" applications. There are a wide variety of home automation and connected devices. In a consumer household, these devices are usually mobile phones, personal computers, gaming consoles, TVs, wireless speakers, wearables, thermostats, fridges, medical sensors, alarm systems, automated shutters, or similar.

While devices like thermostats and fridges are commonly in focus, when talking about IoT, they are not the only devices that could be called IoT devices and need to be managed as such. A large portion of the infrastructure used to communicate on the Internet could also be classified as IoT technology. This includes modems, wireless routers, network switches, power supplies, and proprietary connectors connecting IoT devices with proprietary protocols to the Internet.

> **Definition 11 (IoT Device)** *An IoT device, or device for short, is any physical, electronic device with a connection to a local or global network [10, 107, 127]. An IoT device is usually not explicitly turned on or off, can receive and send data, and provides a service to other devices or users.*

In contrast with much simpler IoT sensors, which feature unique challenges on their own and are described in Scenario 2b, the IoT devices covered in this scenario need to be able to receive commands (e. g., via keyboard or touch screen, remote control, or based on events), act on those commands, and be able to send commands to other devices.

Some IoT devices may stay for many years in one place, while others are moved frequently or constantly. All kinds of devices need to be managed, and due to the increasing number and relatively high fluctuation, this kind of management has to be incredibly efficient and flexible.

To manage and monitor IoT devices, they must be represented as digital objects which describe their current status, parameters, and possible actions. This digital representation is often called a digital twin [51].

Definition 12 (Digital Twin) *The digital representation of a physical entity that describes the physical entity and its properties in real time [51].*

The following examples of consumer-focused IoT devices with interesting management requirements or conditions are analyzed for the common IAM aspects. They are chosen because of their different needs concerning key IAM features, like the number of users, frequency of changes, and number of participating parties. Additional requirements for solely industrial IoT devices used in production lines or to monitor shipping processes may require more accurate data and complex access rules, but most IAM requirements should be transferable.

- **Smart Light Switch:** Light switches are quite simple in their operation of switching a light on or off and indicating to the user in which state the light switch currently is. More advanced versions of those switches allow for dimming or choosing the light's color. While relatively simple devices, their integration in a wireless network and managing access permissions is difficult. A classic consumer IoT light switch is only accessed and managed by a few people living in or frequently visiting the household.

- **Smart Bike Lock:** This IoT device is very similar to the smart light switch, as instead of switching on and off a light, it locks and unlocks a bike. However, it has the added challenge of being mobile and constantly in a hostile environment. Those properties require extra care to tamper-proofing and reliable personal area network (PAN), local area network (LAN), or wide area network (WAN) connectivity.

- **Smart Thermostat:** A remote controllable device that allows the users to set the desired room temperature. It is accessible by the same amount of people who can access the light switch from the previous IoT device example. However, the thermostat's settings are also monitored by the utility company for billing purposes, and the users need to be able to check and set the thermostat while they are away via the Internet.

- **Package Delivery Box:** The package delivery box allows couriers to drop off packages at the user's home while they are not at home. The box can be unlocked by the owner, the courier, and others that the owner authorizes doing so. This is the example that has the most frequent changes in identities being able to access the device, the most parties that might be involved, and the most complex authentication workflows. It might be that the access granted to the seller that will send some wares will delegate this access to a postal service or courier, and they might delegate the access further to a specific package deliverer.

- **Smart Fridge:** A smart fridge that can monitor what foods are stored and which will need to be replaced soon. The owner can authorize the device to order items that must be replenished automatically. This example is the most complex one, as it entails delegating the owner's access rights to the whole family and the device making financial decisions that may have significant consequences but might not be noticed immediately.

As can be seen, by a couple of examples above, many applications for IoT devices need to fit different constraints regarding size, power, or connectivity. A crucial requirement for any IAM solution is broad applicability, regardless of the hardware or software platform it is implemented on.

Further, following those examples of consumer IoT devices, the following common IAM features are found:

1. Identifying the device, its digital identity, and its digital twin is important, as there may be many similar devices, and the device's location may change.
2. People (e. g., the owner, family, or friends) and other entities (e. g., service companies) can be allowed to access the device.
3. Administration of the device is done locally by the owner or somebody the administration has been delegated to.
4. The set of entities that can access the device might explicitly be known a priori or be specified by (indirect) association with other entities.
5. The device can trigger actions with other entities (e. g., acting on behalf of one of the users).
6. Constant connectivity to the Internet may not be guaranteed, but functions that do not necessarily need Internet connectivity should still work.

The individual IAM requirements of the device will be determined by following the life cycle of the device. The life cycle is based on the generic IoT device life cycle described by [172], which is adapted for focusing on IAM:

- **Plan and Design:** During the plan and design phase, the beginning of the IoT device's life phase, design decisions must be made. Regarding IAM and security, this process needs to evaluate if the hardware and communication interfaces are compatible with the cryptography required for IAM. This phase is especially important for IoT devices because of the strict constraints on energy consumption and production cost.
- **Provisioning:** During the setup phase, the device is initialized and configured appropriately for its use case. For IAM, this setup includes assigning the device a unique identifier, which can then be used to identify it when it is connected to other devices or services. Additionally, cryptographic material (e. g., private keys) are loaded onto the device.
- **Configuration:** During the middle of the life phase of the IoT device, configuration changes may be necessary to adapt the device to changed requirements. This could be new keys or credentials.
- **Update:** Updates during the use of the IoT device can add new functionality to the device. Focused on IAM, this may be new specifications, cryptographic functions, or efficiency gains.
- **Maintenance:** During maintenance, issues detected with the IoT device can be fixed. If tampering with the device has been detected or is suspected, part of maintenance may require fixing the device's housing and replacing

potentially compromised secrets, like private keys. Due to the low cost and mass production of many IoT devices, this may include replacing the device entirely.

- **Monitoring:** As with any device or software, it is also essential to monitor the operation of an IoT device. This needs to be done to detect problems early and prevent wrong or lost data from the device. Events that might arise during monitoring are tampering with the device, failing components like memory or batteries, or unreliable communication.

- **De-provisioning:** When the IoT device is no longer needed or has to be replaced, it must be ensured that all potentially private data and secret keys are securely erased during de-provisioning. The device's identity has to be revoked, and it should no longer be able to be used with other devices or services.

- **Retire:** If the IoT device has been adequately de-provisioned, no further actions need to be taken to retire the device.

The remainder of this section will explore a scenario where those features are implemented using SSI for a smart lock. That is a lock, which can be tracked, queried for its state, and operated remotely via the Internet and up close via a PAN (e. g., Bluetooth) connection. Unauthorized use or attempts to open or tamper with the device set off an alarm that notifies previously specified contacts. The following sections will use this example, containing all the IAM challenges detected in the IoT device examples and listed above, to detect common operations and requirements for IoT devices.

As a smart lock, this IoT device's primary operations (i. e., opening and closing) rely heavily on only being available to authorized users. Additionally, the smart lock can be accessed digitally in two ways: via the Internet or a PAN (i. e., Bluetooth). While easy identification of a device is useful for managing the devices, it may also create a significant risk to privacy. Especially if the IoT device can be identified without immediate physical access, i. e., via a PAN connection, this can allow an attacker to create device lists and movement profiles. To avoid this, authenticated identification should be commonly used.

The advantages of using a PAN protocol that supports pairing are also used for the smart lock example. This way, the user can initially connect to a device and discover its URL that can be used later to access the device over the Internet. Once the device can be accessed via the PAN connection or the Internet, it can be further configured using its web interface. In order to gain and specify access restrictions, all requirements from Scenario 1 are adopted for this scenario because the basic premise of registering users or entities is very similar to the processes of the example web app.

The first new challenge of IoT devices that may exist in a vast collection of other, eventually very similar devices is finding and identifying a particular device. Finding and identifying an IoT device can be a problem in two ways:

1. Having the physical device in hand and searching for its digital identity and the associated digital twin or

2. searching for the physical device for a given digital identity.

While this is not a classic problem of IAM, it is important to be able to make the connection from the digital to the real world with items like IoT devices, which form a bridge between the digital and the physical world.

In the web app scenario (described in Section 2.5.1), where a web service is usually identified by its URL, which – if it is not known – can usually be sourced from other locations like advertisements, personal messages, or found using a web search engine. The page's authenticity can be checked (at least to some degree) by verifying the web page's X.509 certificate if the web page offers one. At least initially, the URL of a specific IoT device can usually not be found using those processes.

A solution for finding IoT devices as addressable identities on a network can vary from device to device, depending on its capabilities. One of the simplest ways of identifying a physical device is a serial number that can be used to query a database for more information and then connect to the digital identity. Because this method contains an additional step of looking up information in a database, it is limited in scalability and availability. This is also the case if the device's unique identifier is directly printed or marked onto the device (e. g., as a QR-Code). Without a central database, it would be easy to clone or impersonate a device by copying its identifier. Other common approaches to consumer IoT devices are offering a custom application that can scan the local network for relevant devices, connecting to the device via a (third party) web service, or displaying connection information through the device's display.

This scenario makes use of establishing a connection to the device via a wireless PAN protocol. Searching and connecting to devices in a PAN environment is easier, as protocols like Bluetooth are designed with appropriate processes that enable devices to discover another and pair themself with another [33]. The pairing process can also ensure that the identity of the other entity is established securely. During the pairing, the devices can create and exchange identifiers for each other. These identifiers can address the device and the corresponding digital twin. It also allows the creation of unique identifiers for each connection between two entities, which prevents correlation and improves privacy.

The reverse direction is finding the physical device to a given IoT device's identifier. The information provided by a digital twin can be much more descriptive than an identifier for its identity. It can include connections to other objects or persons which would help identify a given device's location. The smart lock's digital twin would indicate which room's door or bike it has been placed on and where they are located. While having this information available would be of great help to find and identify an IoT device, this amount of detail cannot be expected to be available for all devices and, consequently, cannot be necessary for basic IAM functions. A simpler alternative is to include an identification method like activating visual cues (i. e., similar to indicating the location of a remote-locked car with its indicators). This only requires a LED that can be triggered through the digital twin.

The peering process establishes identifiers and trust between two entities. A everyday use case of a smart lock is giving others (temporary) access to the lock. The trust relation must therefore be extendable to more than two entities. If *Alice* (a) trusts *Bob* (b) and he trusts the identity of the lock (c) and gives her access to the lock, *Alice* needs to be able to pair to the lock, a process in which she establishes new pairwise identifiers, and still be able to identify and trust the lock's identity as the one *Bob* has granted her access to. This situation can be formalized for all entities E and the trust relation T as the transitive relation:

$$\forall a, b, c \in E : (aTb \wedge bTc) \implies aTc \tag{2.1}$$

Trust in the device's identity is the basis for identifying and connecting to the device but managing operations on the device requires additional delegation of specific functions. Therefore, a new component of this scenario is the need for quick and easy delegation of access permissions. Someone with permission to open the smart lock, i. e., has a key, can pass this key on to someone they want to

be able to open it. In the physical world, passing the key to another person also implies that the person that had the key is no longer able to access the lock until they get the key back. In a digital setting, it does not have to work that way. The entity that delegates a permission generally retains it and can still use it. In any case of delegation, the entity delegating permissions is usually still accountable for their proper usage. This accountability has to be communicated clearly and must be traceable.

Digital delegation can also be linked to conditions. For example, it can be prevented that the receiver of a permission delegates it further. This is akin to preventing someone that received a key for a lock from passing this key onto someone else. In the smart lock example, additional constraints, which limit the use of delegated permissions, might be location and time. This would allow specifying that the unlock permission can only be used within a specific area and time frame.

Like with a physical key to a lock, (delegated) permissions for the smart lock have to work without relying on a third party or network connection. The lock may be placed in a location (i. e., a basement) that does not have the necessary connectivity for either the lock or the entity operating the lock to access the Internet, but it still needs to function correctly. In particular, locking and unlocking the lock must still work, while the notification features cannot be provided without connectivity. As a result, the IoT device must be able to determine the validity of permissions from entities without having been informed about the particular entity.

As IoT devices exist as real-world physical items, they must be protected from physical attacks. This does not differentiate the smart lock from other locks, where the primary attack vector is physical, but this is an attack vector that may be overlooked in development. Contrary to regular locks, though, the smart lock can detect and react if it detects a tamper attempt (i. e., notify a contact person, or sound an alarm).

Tampering with the smart lock's locking mechanism is not the most relevant tampering attack regarding IAM. Instead, tampering with the IoT device's identity is of more interest in the general goal of this work. An attacker could try to read secret keys from the device, allowing them to impersonate it and, for example, send false alerts or gather information from legitimate users. This kind of tampering needs to be detected by the device, and anyone who interacts with the device needs to be made aware of the possibility of tampering with the device's memory.

In an IoT device that handles more sensitive information (e. g., a log of each user's last interaction), in addition to being tamper-evident, the device should also be tamper-resistant. This requires special hardware designs that prevent or at least inhibit unauthorized access.

The following sections summarize the requirements developed in the scenario description above. The requirements from the previous scenario, described in Section 2.5.1, also apply here, as determined above. These requirements are not explicitly repeated.

2.5.2.1 Satisfaction Requirements

Trust establishment: The system must support methods for establishing trust in other entities' identities. With established trust relationships, entities can securely associate their data. This requirement is essential (1) for more complex IoT scenarios involving multiple parties. SAT5

SAT6 **Access delegation:** Access to certain functions of IoT devices, like the smart lock, must be transferable to other persons or general entities. Delegation allows flexible and decentralized management of permissions. This requirement is important (2) to effectively manage a growing number of devices.

2.5.2.2 Information Requirements

INF3 **Digital identification:** With physical access to the IoT device, its digital identity must be discoverable. This requirement is essential (1) to aid with the management of many devices. Access to the device's identity may be protected, and only authorized entities may be able to discover it.

INF4 **Physical identification:** The physical device must be discoverable with access to the IoT device's digital identity. This requirement is essential (1) to aid with the management of many devices. As with digital identification, access to the physical identification of the device might be protected and only accessible to authorized identities.

2.5.2.3 Consistency Requirements

CON3 **Digital twin:** Many use cases for IoT rely on mapping real-world entities to a digital representation of them. This digital twin is used to control the entity and other entities. This requirement is important (2), as keeping information available to the physical entity synchronized to the digital representation is required to make the right decisions.

2.5.2.4 Security Requirements

SEC7 **Tamper-evident:** Because of their placement in public, the physical security of many IoT devices cannot be guaranteed. As they may be accessible to adversaries, it is important that secret information (e. g., private keys or credentials) cannot be extracted and compromised without it being noticed by the device. If tampering is detected, the device must notify others about the detection. This requirement is essential (1), as it creates the fundament for secure communication with IoT devices.

SEC8 **Delegation parameters:** Delegation of permissions is important for scalable and flexible permission management. However, some permissions should not be able to be delegated or only for a limited number of steps, time, location, or user group. This requirement is important (2) to retain control over permission management.

SEC9 **Secure de-provisioning:** If a device is no longer needed for a specific application, all data must be securely erased from the device before it can be re-used elsewhere. This requirement is important (2) to securely re-use hardware within different contexts.

SEC10 **Secure setup:** The initial setup of devices may offer an attacker a window of opportunity to claim ownership or register the device first. During the setup process, there must not exist a possible race condition between the legitimate service owner and any third party. This requirement is important (2) to ensure a secure setup.

SEC11 **Tamper-resistant:** This requirement extends the requirement for secure private key storage to a more general requirement that demands the protection of all hard and software components. This requirement is important (2), as it drastically increases cost while never being able to mitigate all tampering attempts.

2.5.2.5 Data protection Requirements

Protected application storage: A key security requirement is the protection of encryption keys to prevent them from being read and used to intercept and decrypt or impersonate the device's traffic. Similarly, it is also important to protect the application data on the device, especially if the device is replaced, sold, or sent in for maintenance. This requirement is essential (1) because this application data may also contain sensitive information. DAT5

Correlation resistance: By monitoring the traffic within a network, even without being able to read its contents, conclusions about the communicating parties and their messages might be deducted. To protect the device's identity, it is necessary to prevent information leakage via metadata. This requirement is important (2) to prevent possible large-scale metadata collection by design. DAT6

2.5.2.6 Robustness Requirements

Offline authNZ: Actions that do not inherently need a connection to a network or the Internet should work without. As most actions on an IoT device require authentication, this means that authentication and authorization of other entities must work offline. This requirement is essential (1), as overly relying on connectivity can lead to unusable devices in case of poor connectivity or disturbances. ROB5

Scalability: One of the main appeals of IoT devices is that they are relatively cheap and thus can be used in scenarios where large deployments of thousands of devices are feasible. IAM operations for this many sensors need to consider scalability. This requirement is essential (1), as the amount of IoT devices and services, in general, is rising with digitalization. ROB6

Platform independence: The architectures used to implement IoT devices are very diverse. An IAM solution must therefore be able to run without being dependent on a specific platform or architecture. This requirement is important (2) to cover as many use cases as possible. ROB7

Transitive trust: To scale trust between multiple devices and services, trust relations must be transitive. This requirement is important (2), as it enables better scalability and simplifies more complex business cases. ROB8

2.5.3 Scenario 2b: IoT Sensors

IoT sensors are special IoT devices that can be used for various monitoring applications. Simultaneously, a specific IoT sensor is highly tailored to its intended application. Some examples of IoT sensors are weather stations, fire alarms, and security systems.

> **Definition 13 (IoT Sensor)** *A sensor for IoT applications is any device that sends collected data at regular intervals or at predefined conditions. The sensor is not capable of receiving any messages. It is likely a* Class 0 *device [21] with minimal memory and energy requirements, allowing for a long run time while battery-powered.*

IoT sensors are further characterized by being simple devices with only minimal controls or interfaces deployed for long periods with minimal maintenance and no remote-management application programming interfaces (APIs). Any receiving functionality is omitted or disabled for these devices to save energy. The simple controls may include only an "on-off" switch that connects and disconnects the power source and maybe an additional reset switch. An example of a device used as such a sensor is depicted in Figure 2.6.

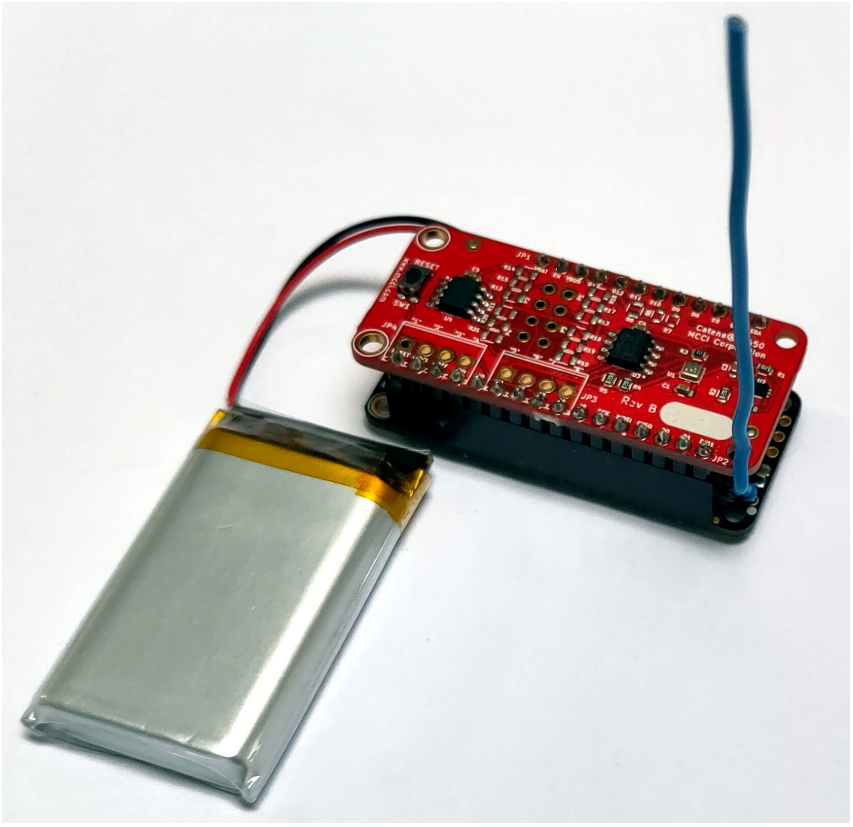

Figure 2.6: Example of a battery-operated IoT sensor with sensors for temperature, pressure, and light

To gather an understanding of the range of applications of IoT sensors, some examples of applications are described in the following section. Each application presents slightly different challenges: for example, with the frequency of communication, the expected run time on battery, or the possible necessary maintenance.

- **Environmental monitoring:** To support high-resolution monitoring of environmental factors, many sensors are deployed over an area of interest. It can monitor agricultural areas like greenhouses, plantations, or irrigation systems. In particular, this example implements a remote weather station that is sampling temperature and humidity information and transmitting averaged data every ten minutes. If some of the sensor's measurements are not received, the system's functionality is not impeded. A system like this is expected to run on battery without charging for multiple months.

- **Security system:** Another application for sensors are security or facility management systems. They are deployed over a smaller geographical region than the environmental sensors. Another example is a door sensor that notifies an application if the door is opened or closed. It only sends changes in the door state and hourly pings to ensure it is still operational. Dropping a message from this sensor can immediately lead to a false representation of the system's state and needs to be avoided. A system like this is likely close to a power source and thus does not rely on battery power and is not limited by battery run time.

- **Fire alarm:** Closely related to the security system sensor, but with ample differences in deployment parameters, the third example application considered is a fire alarm sensor. It notifies a central alarm control application if a fire is suspected or detected. Without any alarm, it reports its operational status to the control application daily. Missing an alarm message from this sensor can have catastrophic effects and must be prevented. For convenience and unrestricted positioning, the system is run with batteries, with an expected run time of approximately 12 years.

This scenario describes IAM-specific requirements for a temperature and humidity monitoring IoT sensor. As the IoT sensor is a special IoT device (described in Section 2.5.2), those requirements apply here without explicitly being listed again. The IoT sensor is the first entity evaluated for requirements of mainly technical interactions, as opposed to interactions with human entities, featured in the web app scenario (Section 2.5.1) and the IoT device scenario (Section 2.5.2).

All the example IoT sensors mentioned above are rarely set up as a single sensor. Instead, many are usually deployed across potentially large areas of land or large building sites. This poses a greater problem for the management of those sensors. Within a company, multiple people need to be able to know about each sensor. For example, the technicians provide maintenance to the sensors, managers need an overview of where sensors are deployed, and security needs to determine if any sensor they find is legitimately placed there. To keep an overview of all sensors and their applications, it is essential to keep a record of each one and to manage access to the sensors' attributes. These kinds of operations would be done with the help of a configuration management database (CMDB) for classical device management.

As the sensor is a simple device, there are no advanced access restrictions to the sensor itself. The sensor's design must protect the firmware and configuration, i. e., disabling debug interfaces, configuring the storage as read-only, or using secure elements. What needs to be configurable is the location the sensor is sending its measurements to and then who can access those measurements from that point onward. The latter part of this can be handled by the application receiving the data. The first part has to be handled prior, and its configuration needs to be restricted to authorized identities.

As the target location of the data can change, it also needs to be able to be changed by the end user. Further sensor customization, like adjusting the sampling interval, does not need to be done often, after the sensor has been configured, tested, and approved for the desired use case. In the examples, the location where the data is sent may change because the sensors are moved to a new backend.

Because sensors are rather small devices, many of them can be deployed. In general, it is a key concern for sensor operators to know their distribution and be able to locate them. In the examples of a fire alarm or security systems sensor, knowing the exact location is especially vital. Meanwhile, depending on the type of environmental monitoring sensor, the accuracy requirement for determining its location is more forgiving.

Locating a sensor can happen in two ways. Either the device is located in the field, and its identity is unknown, or the identity is known but the device must be physically located.

In the first case, the device has to be easily identifiable, and the identification feature has to provide link directly to the device's attributes. This direct link may be restricted by access rights, as not everybody who gets physical access to a sensor should be able to access all of its attributes. Some attributes may need to be publicly accessible, e. g., a contact address to return a lost sensor.

The second case is potentially more challenging to handle than the first one. It is possible that the sensor's location is stored as an attribute of the sensor's identity, but keeping this attribute current is difficult, and errors and inconsistencies while setting the location attribute may further impede locating the sensor.

A common problem with sensors is ensuring the protection of their communication. The sensor is usually communicating with a gateway, that is able to receive messages using special low-power wide area network (LPWAN) protocols. This setup is schematically depicted in Figure 2.7.

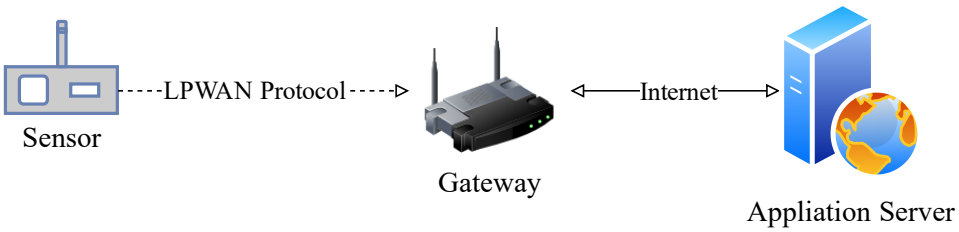

Figure 2.7: Example of IoT sensor communication

It is easy to use common transport security like HTTPS for the second part, from the gateway to the application server. However, the first part of the transmission, from the sensor to the gateway, is harder to protect. This is due to the fact that memory, computation, and energy constraints prevent the sensor from using any asymmetric encryption. Relying on symmetric encryption has drawbacks because of difficult key distribution, which is only possible when the sensor's firmware is flashed, as the device cannot receive messages. This increases the risk of key compromises, exaggerated by the exposed sensor placement without reliable physical security and the key being recoverable from the device's memory. It also misses perfect forward secrecy (PFS), as this usually requires an interactive key exchange, which a transmit-only sensor is incapable of.

Besides the problem of data confidentiality, which might or might not be important to the application of the sensor, another significant problem is ensuring the integrity of the transmitted data. This includes preventing manipulating transmitted data, inserting new data, or replaying transmissions. Each of those is also difficult to ensure because of the stated limitations of being a transmit-only sensor without any persistent writable memory, e. g., to store sequence numbers that last through a reset of the sensor.

Availability is also a problem for sensors which is not solvable through changes on the application level, as is the case for confidentiality and integrity. As the data transmissions are based on a radio link and are extremely low-powered, jamming the device is always possible.

After the transmission of the sensor has reached the gateway, the next difficulty is for the gateway to determine the right application server. Discovery of the right application server is difficult as the sensor has no configuration interface where an address could be stored. Hard-coding an application server is only an acceptable solution in very limited situations and risks jeopardizing long-term use because application service providers may disappear or move. It is also not a viable solution for end-users to re-flash their sensors to adjust the address of the application server. This is especially important if the application server changes regularly or there might be more than one application server, and the set of servers needs to be changed easily.

An example use case for the latter might be an extension of the usage of the door sensor, which is first installed to report to a security application that monitors usage of this door and might trigger a video recording of the door. Later, while setting up a "smart" heating system, this same sensor should be used to determine if the heating should be temporarily disabled because the door is open.

Securing the communication of a sensor also includes reducing the metadata that is generated by each transmission. For example, it is beneficial if a potential attacker cannot deduce the number and type of security sensors deployed on a property by analyzing transmission metadata. While using triangulation to determine the location of a signal source is probably always possible, it is also more difficult than just recording transmissions.

All the sensor variants described above are potentially susceptible to theft or vandalism. For example, the weather station is usually placed in a publicly accessible location. The security system sensor is equally at risk of physical damage, as it is probably part of some kind of security system and thus a target of any potential intruder. The fire alarm system is also only safe in a home environment where access is restricted, and the placement on the ceiling keeps it relatively safe. If deployed in a public or semi-public place like malls or offices, there might be an incentive to manipulate the fire sensor, e. g., to allow smoking in non-smoking zones. This shows the need to detect or prevent physical attacks for each sensor application.

Another factor is the large variety of communication protocols available for sensors. They might use cellular networks with 4G or 5G connectivity, Ethernet (IEEE 802.3 [88]), Wi-Fi (IEEE 802.11 [87]), Bluetooth [89], ZigBee [90], LoRaWAN [122], or some proprietary protocol for layer one communication. To interconnect these devices, bridges must be used. One caveat is that not all the protocols listed here support the definition of an IoT sensor as a send-only device but require two-way communication and pairing.

The following paragraphs summarize the requirements developed in the description above. As IoT sensors are a special kind of IoT device, they also need the same requirements shown in Section 2.5.2. Those requirements are not repeated here explicitly. Instead, only new requirements are discussed.

2.5.3.1 Information Requirements

Product specification: Using an ID assigned to the IoT sensor by the manufacturer, the sensor's messages can be verified to originate from a genuine product by the purported manufacturer. The manufacturer's ID can also link to additional information like measurement accuracy, ranges, or conditions. This requirement is important (2) for automated communication between sensors and devices. INF5

2.5.3.2 Robustness Requirements

Communication protocol independence: As shown in the scenario, a number of competing or complementary communication standards used for communication with IoT sensors. To not limit the applicability, it is important to be independent of a specific standard and support simplex and duplex communication infrastructures. This requirement is important (2), as it allows broad applicability of the solution and also future-proofs it by allowing the communication protocol to be changed. ROB9

Resource efficiency: IoT sensors are frequently optimized for low cost and low power consumption, limiting the amount of available computational resources and memory. The requirements for these sensors must be achieved by being mind- ROB10

ful of those restraints and using as few resources as feasible. This requirement is important (2) to allow the solution to run on as many sensors and devices as possible.

2.5.4 Scenario 2c: IoT Networks

The previous two scenarios described IoT devices that can act as send-only sensors (Section 2.5.3) and Internet-connected devices (Section 2.5.2). In both scenarios, the devices rely on a gateway to connect to the Internet – or any network – to function as intended. This scenario explores the possibility of IoT devices creating and managing their own network to fulfill their roles, especially focusing on how the entities' identities can aid this network management. The design of the distributed network itself is not a key component of this scenario. Instead, the focus is kept on the management of the IoT devices' identities and the identity management-related challenges: To facilitate network management functionality, entities need to connect to a network and actively work on building and maintaining this network. An important challenge for managing IoT devices in a network is the distribution of keys, as discussed by [189]. A DLT-based approach to IAM in IoT networks has been developed as part of this work [70].

This example scenario examines a situation where IoT sensors, as well as IoT devices, are used in a geographic location that does not provide every device with the necessary infrastructure to complete its mission and is thus motivated to participate in the distributed network. The communication technology used in this scenario is based on the LoRa® wireless communication standard and uses a mesh network approach described by [118]. To further differentiate the nodes in the mesh network this scenario will use the following types of devices. Their names are chosen to indicate their primary role within the network.

- **Communicator:** The communicator is a device that can be used by its application or another device to send or relay messages to other devices, especially with the goal of eventually reaching a gateway node. It is similar to an IoT device from Section 2.5.2. Communication is primarily done through text messages and can contain information like status, location, sensor readings, or battery levels. The device itself must not necessarily feature any interface. However, it may be connected to a smartphone or similar device through a personal area network (PAN) to allow the user to input messages.

- **Tracker:** The tracker is a sensor designed to track measurements and make them available to the network. It is similar to an IoT sensor from Section 2.5.3. The information gathered by this device may be the location of equipment, weather information at a point of interest, or capacity information for a parking lot. The tracker itself cannot act as a relay, like the communicator, and thus can have a reduced feature set optimized for power consumption.

- **Gateway:** While the network is self-sustainable, it is connected to the Internet through one or multiple gateways. This allows the network to produce and exchange information with applications outside the network. The gateway must be able to handle messages from multiple communicator nodes and maintain an active Internet connection.

An example of a setup with those types of devices is depicted in Figure 2.8. In a diverse deployment of IoT devices, the proper routing of messages is not apparent, and the reachability of every device cannot be guaranteed. The network of devices has to organize itself while respecting different limitations like connection stability, available bandwidth, power consumption, or latency.

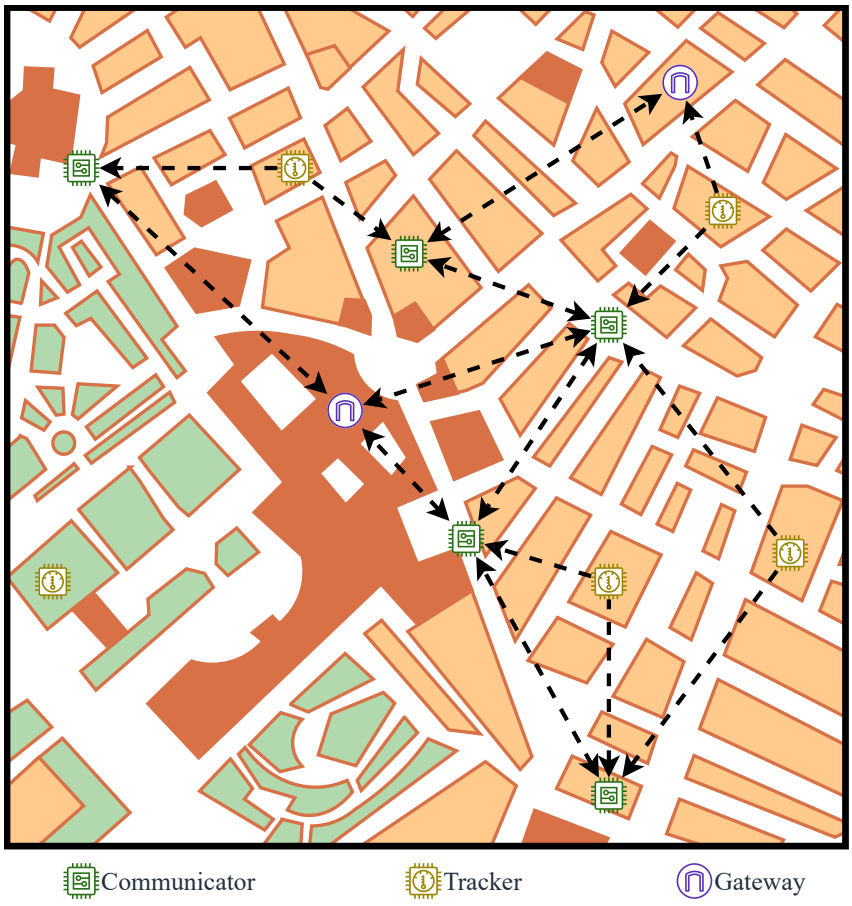

Figure 2.8: Example configuration and overview of an IoT network

The following sections will provide a more in-depth description of the scenario and the tangible use case to be solved. This use case will use the requirements gathered for IoT devices from Section 2.5.2 as the base for the communicator nodes and use the requirements determined for IoT sensors in Section 2.5.3 as the base for tracker nodes. This will tie both of those systems together into a larger coherent environment.

A distributed system's design can be described by three main aspects following the system model by [40]: physical, architectural, and fundamental. The physical model specifies the types of computers that participate in the network, those are constrained to IoT devices in this scenario. This restriction also defines the fundamental model with the key aspects of interaction, failure, and network security.

- **Interaction** is achieved through low bandwidth, long-range radio links, and passing messages between the nodes using LoRa® as the physical layer communication protocol. There is no shared clock, memory, or processing between the nodes provided by the network.
- **Failures** in the network are byzantine – occur arbitrarily – as the simple network does not provide means to detect lost messages or disabled nodes. Detected failures or missing nodes are automatically adjusted by adapting the routing of messages.
- **Security** within the network is provided by using state-of-the-art cryptography for integrity and confidentiality. Availability is achieved through the physical signal modulation that is resistant to deliberate or accidental jamming but cannot be guaranteed.

The main challenge of the network is that it has to work in a decentralized way. It does not need nodes with specific preset roles beyond the classification by device type, which essentially describes the device's capabilities. Key challenges for general distributed networks have been identified by [40], resulting in the following list of facets contributing to the network design. Each of those facets is described with this scenario's specifics and a focus on IAM in mind.

- The network has a high level of **heterogeneity**, as the hard and software components used to build IoT devices are fairly diverse and consist of many proprietary, non-standard, and experimental solutions. As a result, a wide range of devices with varying amounts of hardware performance, using different programming languages and libraries, and competing standards exist. This heterogeneity consequently limits the assumptions that can be made by any IAM layer with regards to hardware, software, or communication method.
- The **openness** of the distributed IoT network is a fairly central point for the network's success. It can only provide large-scale information services if participation is easy. Therefore, the APIs for joining, using, and extending the network must be available publicly. The same is true for the IAM architecture that supports identity operations for entities in this network. Ideally, it is also compatible and usable in a transition phase or alongside other IAM systems.
- Any modern network (distributed or not) should be designed with **security** being one of the main considerations. The key challenge for security in a distributed IoT network arises from the large heterogeneity of devices and participants, resulting in wildly varying capabilities regarding the use of (public-key) encryption for confidentiality, hashing and signatures for integrity protection, tamper resistance, and other environmental factors for maintaining availability. While those restrictions should not hamper the overall security of IAM, the IAM system's design should allow for the usage of different cryptographic methods, depending on the different use cases. This is also relevant to allow the switching of algorithms if they should be found insecure.

- **Scalability** is another aspect that has to be evaluated critically for the long-term success of a distributed IoT network. The network described in this section is built by nodes limited in transmission range, limiting the number of nodes that can be reached directly. Still, the identities of the entities participating may be used globally and should scale to that scope.
- **Failure handling** and failure detection are huge problems for the distributed network. The failure handling component is mostly related to the network layer, though, and not the identity management. Still, precautions must be taken to allow IAM operations to fail and recover orderly.
- As the distributed network is designed for IoT applications, **concurrency** is largely irrelevant. Each node in the network can run independently and concurrently with each other, but usually do not work on a shared task. Many IoT devices, especially those optimized for low energy usage, only run one thread and thus can only process one request at a time. The communication infrastructure (i. e., the wireless frequency) may also not be usable concurrently, and devices may or may not check for collisions. The IAM side is fairly unaffected by this.
- The kind of **transparency** provisions provided by the network envisioned here is somewhat restricted on the network layer but should be reasonably strong on the IAM layer. To differentiate transparency, [40] distinguishes between the following types:
  - **Access transparency**: Access to the IAM operations should be transparent regardless of whether the access is done locally or remotely.
  - **Location transparency**: To establish any IAM exchange between two entities, they must be able to address each other, but this can be done transparent of network location, i. e., the entities do not need to know the hardware or Internet protocol (IP) address (or a similar network layer identifier) of the other entity.
  - **Concurrency transparency**: Concurrent IAM sessions must be possible and should not interfere with each other within the limits set by the available hardware resources. If this were not the case, this might be an avenue to leak private information.
  - **Replication transparency**: Should not be implemented at the IAM layer. Instead, each entity should be identified individually. In case of outages or load distribution between multiple entities, this distribution should not be hidden.
  - **Failure transparency**: In cases of failures on the IAM layer, they should be handled transparently if possible. This may not be possible in many cases, in which the user should be informed with sufficient information about the error to be able to diagnose it.
  - **Mobility transparency**: One of the key elements of a distributed IoT network is that the mobility of entities is an inherently desired property of entities. The physical location of an entity should therefore be transparent when conducting IAM operations.
  - **Performance transparency**: Under normal circumstances, the load of the entities should not affect IAM operations.
  - **Scaling transparency**: The ability to scale transparently, i. e., without impacting the entities, is a critical component of the general system's scalability.

- Like the Internet, a distributed network generally does not offer any **quality of service** guarantees. This limits possible claims about timeliness or adaptability. Fortunately, those are usually not required or generally somewhat forgiving for most IAM operations.

The challenges presented here show the diversity of problems such distributed networks pose in general and for IAM in particular. The network used in this chapter is described as a distributed network.

> **Definition 14 (Distributed Network)** *Within a distributed network, the participating hardware and software components only communicate by message passing via a connected network layer [40].*

Definition 14 is pretty broad and can encompass many instances of distributed networks. In particular, the architecture of the distributed network used in this scenario differs from the more specialized and more narrowly defined peer-to-peer (P2P) network. P2P networks are characterized by all peers having "the same functional capabilities and responsibilities" [40], a feature that is not compatible with the three node types (communicator, tracker, and gateway) distinguished in this scenario. However, the distributed network borrows many of the other properties of a P2P network [168]. In particular, the following traits – usually associated with P2P networks and described by [40] – are used for this distributed network:

- Each device contributes resources to the network. This is true for this distributed network's nodes, with the exception of the tracker. The tracker acts as a device that is generating messages and is usually not listening for or forwarding other messages. The data it provides may be considered contributing resources, but trackers are not obligated to provide their information for all other devices.

- There are no centrally managed systems. Without centrally managed systems, it is more difficult for a party to restrict or limit access to the network, and it eases the ad-hoc setup of the network.

- Providers and users of resources can expect a certain degree of anonymity. This is a common requirement already described in Section 2.5.1 as privacy by design and privacy by default.

- The algorithm for establishing connections between the nodes is especially important. Algorithms for P2P and mesh networks are a highly researched topic, and many proposals and evaluations exist. Because the focus of this scenario is IAM, the used algorithm is less important, except that it should not create any dependency. The network algorithm should be freely interchangeable.

While it is not exactly a P2P network, the distributed network's topology is considered to be a mesh network, as nodes work to build and maintain connections to adjacent nodes. Due to the different device types, this mesh network is unlikely to be a fully connected mesh. Instead, there may be some aspects of star or tree topologies around the more powerful communicator and gateway devices.

Within a distributed network that is run without any central authority, being able to identify individual nodes or devices is essential. Allowing those devices to express aspects of their identity through SSI can improve the flexibility and applicability of the network in various situations.

A key feature – or challenge, depending on individual perception – of distributed networks is that nodes may join and leave the network at any given time, and the network must adjust to those changes. Also, parts of the network may become split from one another at any time and will need to be able to handle the lost connections. As a result, the main operations from the network's perspective are:

- Integrating new nodes
- Removing failed or exiting nodes
- Determining, communicating, updating, and optimizing the routing table
- Merging two networks

All of those operations require the devices to identify each other and address individual devices uniquely in their messages. Each device can build a routing table based on which devices can be received from or sent to directly and which additional devices those neighboring devices can reach. To guarantee the stability of the network, devices must be able to be identified with a certain degree of confidence, and impersonating other devices or flooding the routing table with bogus devices should be prevented; or at least mitigated.

The integration (or alternatively called registration) of a device with the network requires the device to be able to connect to the network in the first place. On the one hand, this requirement has to be solved by supplying the device with the proper hardware (e. g., Ethernet, wireless local area network (WLAN), Bluetooth, LoRa®, etc.) and hardware configuration (e. g., frequency bands, channels, ect.). On the other hand, the device must be able to access the network, especially if the network has some kind of access restriction, e. g., WLAN networks that are protected by a PSK.

Removing a device from the network is difficult, especially if the device does not log off properly. This may be the case if the device suddenly fails, the connection is bad, or the device moves out of range. In some cases, the device may be able to connect unreliably so that it is joining and leaving the network constantly. The connection status of a device can be determined by noting the time of the last activity and presuming the device to have left the network if a threshold value is reached. Alternatively, active querying of the device can result in quicker status updates.

Routing messages within P2P and mesh networks is a highly researched topic [99, 166, 195]. Many algorithms and evaluations of said algorithms, showing individual strengths and weaknesses, exist. Choosing a suitable algorithm should bear the limitations of IoT devices and the general goals of an IoT network in mind.

The same is true for merging previously split network segments back together. During those operations, there may be a lot of temporary traffic through the network as nodes try to catch up one another. Without proper countermeasures, this may overwhelm the devices positioned between the two segments.

In general, managing a P2P or mesh network is a team effort. It requires every node to adhere to the rules of the network and provide the necessary services. As such, in P2P networks, so-called leechers (peers that only use resources instead of providing their share of resources) should be discouraged by the network's design.

The specific use case implemented for this scenario is centered around an off-grid wireless communication system. It does not necessarily have to be used in a remote area without any other networks, but to limit additional challenges with differing communication standards, that would be available to build such a system in an

urban environment, it is assumed that the devices can only communicate through one system and that Internet connectivity is achieved through one or comparatively few gateway systems.

The use case uses LoRa® as the wireless communication system, as it offers long-range communication with low-power consumption at a low bandwidth. It is a communication standard used by many enthusiasts and features relatively cheap hardware, many open-source libraries, code examples, and large-scale networks. This is ideal for testing with small Arduino-based development boards, which are readily available. The low bandwidth, constrained memory, and slow processors also increase the necessity to consider efficiency in the system's design. As a result, it should be easily transferable to more powerful platforms and technologies.

The use case consists of a network of various entities which are placed or move through an area of multiple kilometers:

- **Sensors:** Measure the temperature and humidity of the surrounding environment and provide their measurements to other entities on the network. The measurements should be traceable and the authenticity and reliability of a sensor product should be asserted by the manufacturer and operator.
- **Trackers:** Roaming devices that can transmit their current position. The position can also be augmented by additional data (e. g., temperature of cargo) and protected against tampering or manipulation.
- **Messengers:** Devices that allow users to communicate via text messages through an app on their smartphones.
- **Gateway:** A gateway that can forward messages to a service on the Internet.
- **Web-service:** A portal web service that can display data gathered from sensors or trackers and measure the availability and reliability of the network components.

The following sections summarize the requirements developed in the description above. As this scenario builds on top of the previous two scenarios regarding IoT devices (Section 2.5.2) and IoT sensors (Section 2.5.3), it also implicitly includes the requirements from those scenarios. Requirements that would be repeated here are not explicitly described again.

2.5.4.1 Satisfaction Requirements

SAT7 **Automated integration/registration:** A secure connection between two devices, which previously did not know each other, must be establishable without user interaction. Some general configuration of the network parameters and keys may previously be necessary, however. This requirement is essential (1) for any distributed network to be able to self-organize.

2.5.4.2 Information Requirements

INF6 **Neighbour discovery:** An integral part of the distributed network is the detection of neighboring nodes and the subsequent routing of messages through them. This requirement is essential (1) to dynamically build the distributed system.

INF7 **Service discovery:** As a specialization of the requirement for neighbor discovery, a node might want to discover nodes that can provide specific services. This might be necessary because one service is no longer available because it got disconnected or is no longer in range and should be automatically replaced by a similar service. This requirement is optional (3), as it is not necessary for the network's core functionality but can provide more flexibility.

2.5.4.3 Consistency Requirements

Content verification: The content provided by nodes and consumed by others is hard to verify. Nevertheless, a sensor providing measurements (e. g., the temperature at a specific location) should really provide this measurement to the specified conditions. This requirement is important (2) to build reliable services. CON4

Netsplit/Join: If parts of the network are split off from another (e. g., because a single connecting node fails or moves out of range), upon reconnection of both segments, synchronization of relevant data has to occur. This requirement is important (2) if data about digital twins is stored throughout the network (i. e., on a specialized database node). CON5

2.5.4.4 Data protection Requirements

External tracking resistance: As devices move within the network, their physical location can be associated with the location and movement of people related to the device. This information could be used to build movement profiles. As a result, any pure observer of the network and its messages should not be able to trace selected devices. This requirement is important (2) and should be part of the requirement for privacy by design. DAT7

Internal tracking resistance: Gateway and communicator nodes are at a central position within the network. As a result, they can potentially observe most of the traffic within a specific region of the network. They should not be able to gain sensitive information about the content or nature of the communication. This requirement is important (2) to increase trust in the network. DAT8

2.5.4.5 Robustness Requirements

DDoS protection: As the network is designed to be open and the hardware requirements should be as low as possible, it is evident that a malicious attacker can easily control more resources. The system must therefore tolerate disruptions like flooding the network with new neighbors. This requirement is important (2), as easy attacks on the network limit its usefulness. ROB11

2.5.5 Scenario 3: Cloud & Edge Computing

As part of an IoT ecosystem the data generated by the various sensors and devices is often processed on a cloud infrastructure. If the processing is time critical or the data should not be transported to a cloud service, a modern architecture alternative, called edge computing, utilizes smaller data centers positioned closer to the egress point of the IoT network. The processing systems in cloud and edge computing scenarios are usually controlled by web applications. Those close the loop between the IoT devices (the actors described in Scenario 2a), the IoT sensors (described in Scenario 2b), and the connecting network (shown in Scenario 2c) to the first scenario describing regular web applications (Section 2.5.1) via this scenario. The data and information loop generated by this IoT network is illustrated in Figure 2.9.

Cloud computing, in general, is a highly centralized service offering that uses massive data centers in a few locations worldwide to provide services to customers. Because of the scale of those data centers, the services that customers can order on demand can scale almost endlessly and create an illusion of infinite processing power [57].

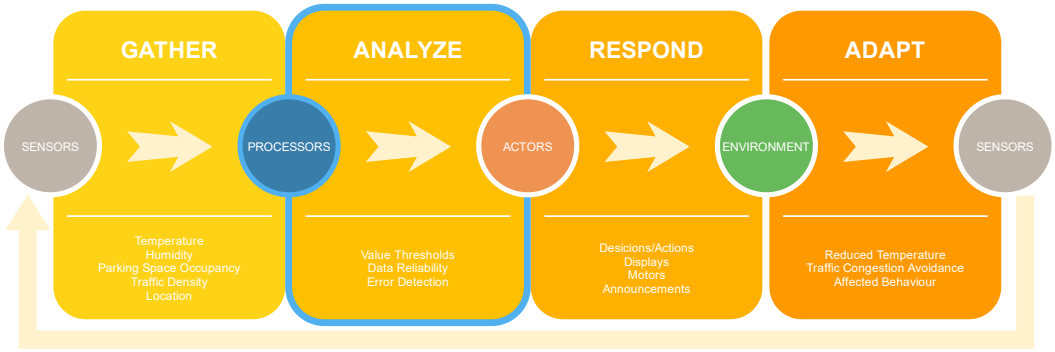

Figure 2.9: IoT data and information loop. The focus of this scenario, the data processing, is highlighted in blue.

Demand for computing resources rises constantly. For example, with the advent of IoT technology, the number of Internet-connected devices has increases drastically. As a result, the applications running classic cloud infrastructure may no longer be sufficient for some use cases. An area where cloud computing can be lacking considerably is latency, as data has to be uploaded to a remote data center, processed there, and the results need to be downloaded afterward. To mitigate the shortcomings of cloud computing, multiple approaches exist to create a distributed cloud computing environment. As those are most important for the data processing described in the previous scenarios, the most prominent approaches are described here.

To decrease the latency of cloud computing, the concept of fog computing adds an intermediary layer between the cloud computing data centers, the IoT network, and its sensors and devices. This is characterized by [19] as a very large number of nodes sharing resources in a predominantly wirelessly accessed network with solid capabilities for streaming and real-time applications and general heterogeneity. This geographically wide-spread distribution layer allows the system to achieve less latency, more location awareness, and better mobility of devices.

Edge computing is very similar to fog computing, as it arguably describes the same principles and goals. The authors of [62] describe edge computing as the move of processing from cloud computing resources to the edge of the network, where the edge devices may be smaller data centers or other mobile devices. In addition to the latency improvements of this distribution of processing, the authors also show privacy and data protection reasons for using edge computing.

For the remainder of this work – and in accordance with the decision of the authors of [57] – the term edge computing is used as equivalent to fog computing. The more important differentiation in terminology is the more mobile device-focused developments of distributed computing: mobile cloud computing (MCC) and mobile ad-hoc cloud computing (MAC)

MCC uses small cloud data centers – so-called *Cloudlets* – in the proximity (logical and physical) of the mobile device to support resource-intensive operations with low (LAN equivalent) latency [165]. Those Cloudlets should be able to be deployed by individual operators similar to Wi-Fi access points, they should be easy to set up and require only minimal maintenance [165]. The main benefit is to conserve energy and general processing capacity of the mobile device [57].

MAC ditches the Cloudlets of MCC in favor of offloading processing to any other mobile devices in the vicinity [191]. This increases the availability of resources, as it no longer requires an operator to have set up a Cloudlet nearby or require

sufficiently fast connectivity to a cloud data center. However, it also increases the risk of data ending up in the wrong hands, so it requires special care for securing the processing. The optimization goals for MAC – reducing energy consumption and increasing the processing capacity of low-powered devices – are still the same as for MCC [57].

Up until now, the scenarios described in this work were a web application for a publish subscribe messaging system (Scenario 1), a smart lock (Scenario 2a), a temperature and humidity sensor (Scenario 2b), and an off-grid communication system (Scenario 2c). This scenario describes a processing system that provides a useful combination of all abovementioned scenarios.

In this example, a number of shipping containers and their cargo is monitored by sensors. They can detect abnormally high temperatures that may indicate a fire within the container or its vicinity. Additionally, they can detect fluids leaking from the cargo or otherwise accumulating inside the container. To secure the cargo, those containers are locked by smart locks that can automatically unlock to support firefighting efforts. Quick response to fire threats is especially important on large container ships, where fire is an extremely dangerous hazard.

To connect the sensors and locks, a local network spanning the whole ship is required, as container ships travel largely through remote ocean areas without a connection to standard network infrastructure. The only constant source of Internet connection may be through very slow satellite Internet. This is an ideal environment to set up a Cloudlet.

On shore, the containers are stored in vast yards or are further transported by train or trucks. In the yards, Internet connectivity may be more reliable, so that cloud computing can be used to process the information. While underway on trucks or trains, connectivity may also be an issue, and operations may have to use a MAC approach for data processing.

The different data processing situations – discussed in this scenario – are shown in Figure 2.10.

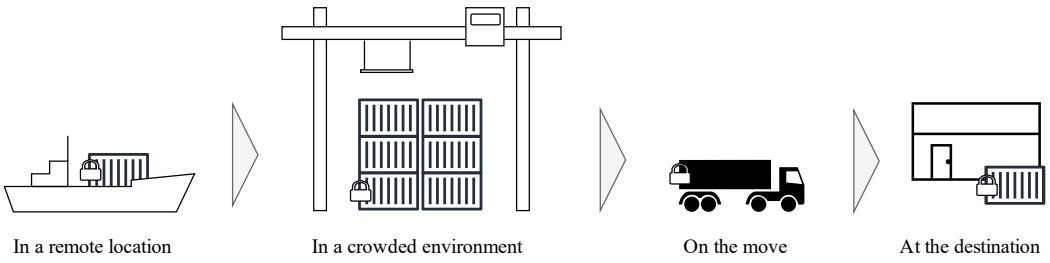

Figure 2.10: Overview of the different data processing situations in a dynamic IoT sensor and IoT device network

Those different situations show the challenge of keeping the general alarm and warning system working regardless of where the container is currently located. To accomplish this, the sensor has to always be able to communicate with the lock of the container while utilizing external information and data processing to determine if any actions are necessary. Additionally, the processing application which decides if the situation warrants unlocking the container has to be available, so it needs to anticipate the location of the container to be reachable via the cloud, a Cloudlet, or MAC. To protect the contents of the container, it is also important

to only automatically unlock the container if it is reasonably certain that there is a problem (e. g., fire or water) inside and prevent any unauthorized party from influencing this decision.

From an IAM point of view, the challenge boils down to supporting the highly dynamic infrastructure. Especially important are portable identities that can be used on-the-fly regardless of being connected to the Internet or not. If the services (i. e., a communication network or data processing) needed by the IoT devices are not free to use – which is highly likely in any business environment – all participating parties need to be able to determine how payment will be processed. This could be in the form of a credential carried by the sensor that specifies which other legal entity will compensate the service provider for any (or a list of) services required by the sensor. This reference credential "I will pay service charges incurred by this entity at the condition described in a credential here" can allow for more complex scenarios.

To prevent unauthorized charges or billing of services that were not actually provided, the sensor and the service provider need to agree on the conditions of the service beforehand. The actual performance of the service should also be monitored and reported to resolve any potential disputes.

The following sections summarize the requirements developed in the description above. As this scenario encompasses the previous scenarios, their requirements are implicitly adopted for this scenario as well. Therefore, only new requirements are described below.

2.5.5.1 Information Requirements

INF8 **Agreement monitoring:** Agreements between entities must be able to be monitored, and violations of the established terms and conditions must be recorded. This requirement is important (2) to support business use cases.

INF9 **Capability exchange:** Exchange information about computing capabilities on-the-fly to determine the best location to do the process offloading. The processing capabilities of an entity are attributes of its identity. This requirement is important (2) to find a suitable location to do cloud computing.

2.5.5.2 Robustness Requirements

ROB12 **Standalone authNZ:** Parts of the network of entities may be separated from most other entities (i. e., the Internet). Even in this situation (and without prior preparation), the connected entities must identify each other correctly. This requirement is important (2), as global connectivity cannot always be guaranteed.

2.5.6 Scenario 4: Electronic Identity (eID)

One ever-increasing topic for online identities is developing identities capable and trustworthy enough to be accepted when dealing with governmental institutions, e. g., for filing taxes, proving one's identity when opening a bank account, or even voting in elections. These government-supported identities are usually called eID. Some kinds of eIDs utilize electronic identification cards (eICs) to store identity information on a physical device. For example, the Austrian *Bürgerkarte*, the Belgian identity card *BELPIC*, the identity card of the United Kingdom, or Germany [7, 149]. Many countries have an eID system in operation and are planning to build and expand the functions and usefulness of the existing system continuously [111, 193]. At the same time, the actual implementation of the guidelines set by the EU usually differs per country [119, 126]. Some of those countries are also looking at SSI-based solutions [143].

The eIDAS [157] legislation was passed within the EU in 2014. It aims to establish eID in all member states and to make those systems work across borders to form a digital single market. According to this legislation, all EU members must accept eIDs from other members for public digital services since the fall of 2018.

As this system encompasses the eID systems of all EU countries, it features a broad spectrum of individual implementations that must work together. Therefore, eIDAS is the main focus of this scenario and is described in more detail. Of the eIDAS-compatible eID systems, the German eID system is described more closely in Section 3.5.

The requirements for the eID scenario are not gathered from individual eID systems or any individual country's implementation but are established by analyzing the scenario for eIDAS. It currently seems that eIDAS is the most complete legal framework for eID by spanning multiple countries.

The goal of eIDAS is to allow natural and legal persons to use their national electronic identification scheme (eIDS) in cross-border applications with other citizens, businesses, and public services in the other EU member states [157, Chapter 1]. It was first established as an EU regulation with the publication of the *Official Journal of the European Union* under *Regulation 910/2014* [157]. This regulation specifies six applications for eID:

1. **Electronic Signatures:** Attaching or linking data to other electronic data the signatory wants to sign [157, Chapter 3, Section 4].
2. **Electronic Seals:** Showing the origin of an electronic document by proving the document's issuer (i. e., a legal person) and integrity [157, Chapter 3, Section 5].
3. **Electronic Time Stamps:** Proving that the time-stamped electronic data existed at the time specified by the time stamp [157, Chapter 3, Section 6].
4. **Electronic Registered Delivery Services:** Allows two parties to exchange electronic data while being able to prove the data has been sent, received, and remained unaltered in transit [157, Chapter 3, Section 7].
5. **Website Authentication:** Proves to the visitor of a website which entity is standing behind the visited website [157, Chapter 3, Section 8].
6. **Electronic Documents:** Any regular – paper-based – document just in electronic form and having the same legally binding effects [157, Chapter 4].

Within those topics, one of the main goals is that citizens should be able to use electronic processes with the same legally binding status as traditional paper-based processes, at least for public services. For example, regarding (digital) signatures, the regulation states: "[...] a qualified electronic signature should have the equivalent legal effect of a handwritten signature." [157]. An electronic signature is *qualified* if it is *advanced* (i. e., adheres to the regulation's requirements), is created by a qualified electronic signature creation device, and is based on a certificate that was issued by a qualified trust service provider.

To accomplish this goal, eIDAS does not introduce a new eIDS but instead calls for interoperability between national eID systems. However, the required interoperability is not easy to guarantee, mainly because of the differing approaches to eID and the overall legal variations for eID in the EU member states. Some of those differences are explored in [111].

The national eID systems in question may already exist or may be built for this expressed purpose. For a national eID system to be included in eIDAS and be recognized as a compatible eID system by the EU, the member states need to have their implementation *notified*. This is a process by which the other member states

validate that the proposed solution meets the EU's requirements. Currently, the eID systems of Austria, Belgium, Croatia, the Czech Republic, Denmark, Estonia, France, Germany, Italy, Latvia, Lichtenstein, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Slovakia, Slovenia, Spain, Sweden, and the United Kingdom, have been notified [138].

With so many countries participating, challenges have come up in different areas. The most prominent ones are analyzed in the remainder of this section to highlight those challenges and use them to deduce requirements for any new eID system.

Interoperability In general, the solutions used as national eID services can differ a lot. For example, the Netherlands and Italy use a system based on SAML, Latvia uses a system compatible with *OAuth 2.0* (OAuth), while most other nations use custom standards based on X.509 certificates for authentication. This fundamental incompatibility in used technologies needs to be addressed by the national eID systems through gateways or proxies.

Identity Data Set The attributes used to legally identify a citizen can vary from country to country. In Germany, a citizen is usually identified by sure and given name, place of birth, date of birth, citizenship, and home address. Banks, for example, must store those attributes to identify customers according to anti-money laundering regulations [65]. As a result, but still differing slightly, the German *Bundesamt für Sicherheit in der Informationssicherheit* (BSI) specifies the given name, birth name, date of birth, and place of birth as necessary attributes for German citizens using the German eID infrastructure [24]. Other countries may use different minimal data sets to identify citizens, including relying on a single national identification number. Still, for any system to be compatible with eIDAS, they must provide at least the family name, first name, date of birth, and a person identifier [49]. The person identifier is a pseudonymous ID, which is unique for each pair of user, IdP, and SP. The challenge for national eID systems is to supply the required but potentially missing attributes and to handle attributes provided by other systems which are not required within their own system.

Attribute Formats If the same attributes for citizens are available in two countries connecting through eIDAS they still might not be in a compatible format, which would allow seamless processing of the attributes. Because there is no simple way to facilitate interoperability between the different eID systems for cross-country usage, the identity information has to be translated in a way that renders it usable in the receiving application. The EU offers documentation for authentication with a focus on SAML [49]. How the individual translation to and from the intermediary SAML should be done is currently not specified. However, the STORK 2.0 project shows two possible options: A national *Pan European Proxy Service* (PEPS), which would route and transform the traffic, or a *MiddleWare* that is implemented as a virtual IdP and is run for each country [16].

Level of Assurances (LoA) Besides differing attribute standards and formats, the guarantees that an attribute has been associated correctly with an identity can vary in confidence. Higher confidence may be acquired by requiring users to appear in person to verify attributes, while lower confidence may be based on other sources. Not all applications and services need the same confidence of identifying an entity correctly. The confidence that an attribute has been correctly established is usually expressed as a LoA by the party issuing an assertion. Claiming an invalid assertion for one's identity with a higher LoA should be more difficult, or better, next to impossible. The eIDAS regulation specifies three assurance levels: low, sub-

stantial, and high. Those levels are based on levels 2, 3, and 4 from [176]. Higher assurance levels also always go hand in hand with more complex and difficult authentication processes but may be necessary for certain critical applications.

Once-only Principle (OOP) Another part of the EU's eID initiative is a concept called once-only principle [52]. This principle aims to prevent redundant presentation and storage of citizens' information at different government services. Instead, the (frequently) required (standard) information should be gathered by the services from other government services, which already possess the required information in a privacy-conscious way. This should save citizens from submitting the same documents repeatedly and increase usability of e-government.

The implementations of the OOP differ from nation to nation. Some use the so-called government-centric model and store all information about citizens and businesses in a central database with a unique identifier and forbid the creation of any other databases that would contain redundant information. An alternative model is the citizen-centric one. Here the citizen can, instead of providing the required data repeatedly, point a government service to another government authority that has the data, which then can provide it to the service. OOP has been tested in two projects the *The Once-Only Principle Project* (TOOP) [109] and the project *Stakeholder Community Once-Only Principle For Citizens* (SCOOP4C) [139].

The following sections summarize the requirements developed in the description above. As eID is closely related to and often used with web applications, this scenario includes all the requirements posed in the first scenario (Section 2.5.1).

2.5.6.1 Satisfaction Requirements

Identity data set matching: In different countries, different identity data sets may be used. As the eID should be usable in different countries, a mapping has to be possible. This requirement is important (2) to build a digital single market, as the EU envisions with eIDAS. SAT8

Message delivery services: In the space of eID, electronic registered delivery services are necessary to ensure messages have been delivered to the recipient and the recipient can be expected to retrieve and act upon them. This allows for sending messages and data electronically equivalent to registered mail. While this requirement describes a service built with IAM, it is still added to the requirements list as messaging entities, and confirming the delivery of messages can be useful in any IAM setting. Delivery services should guarantee non-repudiation of sending, receiving, and integrity of the message. This requirement is optional (3), as it is not a core IAM feature but has many use cases, especially for using eIDs with eIDAS. SAT9

2.5.6.2 Information Requirements

Identity data set: When identifying citizens or organizations, there is usually a legally defined set of required attributes which must be supplied to sufficiently identify the entity. The identity data specifies and contains those attributes. This requirement is essential (1), as the contained attributes are (legally) required to use (government) services. INF10

Level of assurance: Authentication strength and attribute checking can be done at various confidence levels. Some services may require more strict checking than others. LoAs help SPs and IdPs to communicate the required and provided confidence levels. This requirement is important (2), especially for official government services and authentication across borders. INF11

2.5.6.3 Consistency Requirements

CON6 **Once-only:** The users' attributes should only be gathered, checked, and certified once. For subsequent interactions, it should be exchanged appropriately, even with different SPs and IdPs. This requirement is important (2) to increase usability and reduce unnecessary redundancy, which can lead to inconsistent data.

2.5.6.4 Security Requirements

SEC12 **Off-the-record (OTR):** An authentication must only be usable between the two entities that mutually identified each other before starting the authentication. This requirement is essential (1) to prevent misuse and any incentive of recording or passing the authentication data to other services.

SEC13 **Trust service providers:** To facilitate trust between different entities, qualified trust service providers check the requirements and issue trust certificates to participating entities. These trust services act similarly to certificate authorities (CAs) and intermediary CAs. This requirement is important (2) to increase scalability of trust among entities.

2.6 Requirement Summary

This section provides an overview of all requirements gathered in the various scenarios in Table 2.1. The table shows where a specific requirement was defined ($\overset{\text{def}}{=}$), where it was included ($\in$), and its importance, as determined in the scenario where it was defined.

Table 2.1: Summary of all scenarios' requirements

Category	*Requirement*	*Scenario 1*	*Scenario 2a*	*Scenario 2b*	*Scenario 2c*	*Scenario 3*	*Scenario 4*	*Importance*
Satisfaction	SAT1: Authentication	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SAT2: Authorization	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SAT3: Identification	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SAT4: Identity provisioning	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SAT5: Trust establishment		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↑
	SAT6: Access delegation		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	SAT7: Automated integration/registration				$\overset{\text{def}}{=}$	$\in$		↑
	SAT8: Identity data set matching						$\overset{\text{def}}{=}$	↕
	SAT9: Message delivery services						$\overset{\text{def}}{=}$	↓

Continued on next page

Table 2.1: Summary of all scenarios' requirements (Continued)

Information	INF1: Credential establishment	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	INF2: Documentation	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↕
	INF3: Digital identification		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↑
	INF4: Physical identification		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↑
	INF5: Product specification			$\overset{\text{def}}{=}$	$\in$	$\in$		↕
	INF6: Neighbour discovery				$\overset{\text{def}}{=}$	$\in$		↑
	INF7: Service discovery				$\overset{\text{def}}{=}$	$\in$		↓
	INF8: Agreement monitoring					$\overset{\text{def}}{=}$		↕
	INF9: Capability exchange					$\overset{\text{def}}{=}$		↕
	INF10: Identity data set						$\overset{\text{def}}{=}$	↑
	INF11: Level of assurance						$\overset{\text{def}}{=}$	↕
Consistency	CON1: Identity de-provisioning	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	CON2: Credential recovery	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↕
	CON3: Digital twin		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	CON4: Content verification				$\overset{\text{def}}{=}$	$\in$		↕
	CON5: Netsplit/Join				$\overset{\text{def}}{=}$	$\in$		↕
	CON6: Once-only						$\overset{\text{def}}{=}$	↕
Security	SEC1: Access controls	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SEC2: Credential revocation	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SEC3: Mutual authentication	$\overset{\text{def}}{=}$						↑
	SEC4: Security by default	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SEC5: Security by design	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	SEC6: Multi-factor authentication	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↕
	SEC7: Tamper-evident		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↑
	SEC8: Delegation parameters		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	SEC9: Secure de-provisioning		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	SEC10: Secure setup		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	SEC11: Tamper-resistant		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	SEC12: Off-the-record (OTR)						$\overset{\text{def}}{=}$	↑
	SEC13: Trust service providers						$\overset{\text{def}}{=}$	↕

Continued on next page

Table 2.1: Summary of all scenarios' requirements (Continued)

Data Protection	DAT1: Privacy by default	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	DAT2: Privacy by design	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	DAT3: GDPR	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↕
	DAT4: Multiple identities	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↓
	DAT5: Protected application storage		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↑
	DAT6: Correlation resistance		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	DAT7: External tracking resistance				$\overset{\text{def}}{=}$	$\in$		↕
	DAT8: Internal tracking resistance				$\overset{\text{def}}{=}$	$\in$		↕
Robustness	ROB1: Reliability	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↑
	ROB2: Accessibility	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↕
	ROB3: Approachability	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↕
	ROB4: Usability	$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$	$\in$	$\in$	↕
	ROB5: Offline authNZ		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↑
	ROB6: Scalability		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↑
	ROB7: Platform independence		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	ROB8: Transitive trust		$\overset{\text{def}}{=}$	$\in$	$\in$	$\in$		↕
	ROB9: Communication protocol independence			$\overset{\text{def}}{=}$	$\in$	$\in$		↕
	ROB10: Resource efficiency			$\overset{\text{def}}{=}$	$\in$	$\in$		↕
	ROB11: DDoS protection				$\overset{\text{def}}{=}$	$\in$		↕
	ROB12: Standalone authNZ					$\overset{\text{def}}{=}$		↕

Key: $\overset{\text{def}}{=}$ defined in this scenario, $\in$ included from previous scenario, ↑ essential, ↕ important, ↓ optional

Chapter 3

State-of-the-Art

Contents

IAM is a sophisticated topic that features many facets. A huge amount of previous research, standardization, and product development exists. This is emphasized by the research in this area and the multitude of IAM solutions available. An overview of modern IAM systems is described by [146], involving FIM with protocols such as SAML, OAuth, and OIDC, as well as UCIM and SSI

In this work, only the SSI-related portion of the IAM landscape is examined in detail. The other manifestations of IAM (e. g., FIM and UCIM) have already been described in detail in other related works (e. g., [82] or [142]). Its applicability for

one or more scenarios is required to narrow further state-of-the-art considered for selection. As a result, solutions for web-based technologies, IoT applications, mesh networks, and eID are favored. For the selected applications, relevant standardization, research, implementations, and real-world products are evaluated, if available. An additional aid for selecting relevant state-of-the-art is created by classifying selections according to the IAM dimensions consisting of **identity**, **access**, and **management**, each of which can be described by three characteristics:

- Identity is divided into device, personal, and organizational identities.
- Access differentiates between the process of identification, authentication, and authorization.
- Management is split into different approaches centralized, federated, or self-sovereign.

Figure 3.1 shows a 3D representation of those three dimensions. This representation is used throughout this chapter to illustrate the discussed solutions' primary application areas. While the IAM space can be extended with way more dimensions, i. e., MFA, this will show the gaps in the basics of IAM system coverage.

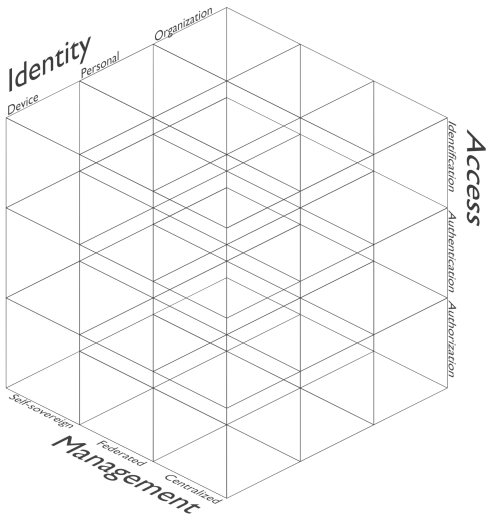

Figure 3.1: IAM dimensions

The remainder of this chapter is structured as follows. First, web-based solutions are described in Section 3.1. This serves as a baseline for modern IAM and is used in the implementation described in Chapter 5 to select state-of-the-art technology. Building on top of mainly centralized web-based IAM, Section 3.2 shows the most important developments in FIM. The main focus of this work, SSI and related work from research, standardization, and products, is described in Section 3.3. Use case-specific state-of-the-art is described for IoT in Section 3.4 and for eID in Section 3.5. A summary highlights the presented technologies and shows the remaining work areas in Section 3.6.

3.1 Centralized Web

One of the most important spaces for digital identities is the Internet. However, most of the IAM systems on the Internet are centralized at each service and many of those build on proprietary and individual solutions. This chapter considers state-of-the-art systems, which are open, accessible, and standardized. Systems that are not centralized but federated are described in Section 3.2.

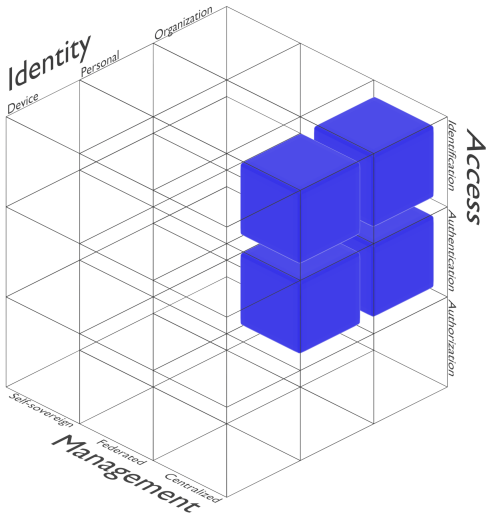

Figure 3.2: IAM dimensions for typical web-based applications

In general, web applications cover the IAM space depicted in Figure 3.2. They focus mainly on centralized identification and authentication of individual persons but also provide identification and authentication for organizations, i. e., websites. The authorization is mainly handled by the web applications internally and thus does not require standardization. There are only a few notable standards in the space of primarily centralized web-based authentication due to considerable fragmentation and individual solutions. One of the most widely used standards for online identity is developed by the FIDO Alliance. Their standards are described in Section 3.1.1. The foundation of security on the Internet, the X.509 certificate, and the corresponding public key infrastructure (PKI) are summarized in Section 3.1.2. Another interesting but ultimately no longer pursued approach that has similarities to SSI by the Mozilla Foundation is presented in Section 3.1.3.

3.1.1 Fast Identity Online

Started in 2013, the FIDO Alliance consists of over 250 members from all over the world and specializes in developing authentication standards for the Internet [59]. Since its foundation, the FIDO Alliance has published multiple standards regarding 2FA, MFA, and password-less authentication, always utilizing special hardware devices that can be connected to computers or smartphones via USB, near-field communication (NFC), or Bluetooth. The three primary standards are:

- the **U2F standard** defining how to integrate second-factor hardware devices into existing password-based authentication at websites [55, 162],
- the **UAF standard** for eliminating passwords at the website alltogether and authenticating purely with a compatible hardware device with local user authentication [161],

- and the **CTAP standard**, where CTAP1 is just another name for the U2F standard, and CTAP2 specifies the communication between hardware devices and browsers for password-less or MFA authentication [102].

The standards developed by the FIDO Alliance are also referenced by other standardization organizations. Together with the W3C's standard for WebAuthn and the FIDO Alliance's CTAP standard, a project for web authentication is developed under the name FIDO2. The WebAuthn part of this project is located with the W3C as it specifies *Javascript* (JS) APIs for browsers, which naturally fit the W3C scope. The FIDO Alliance's CTAP part of FIDO2 specifies the operating system's and security hardware device's properties. An overview of those standards is displayed in Figure 3.3.

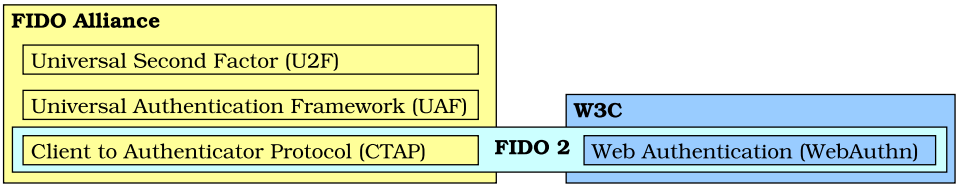

Figure 3.3: Overview of the FIDO Alliance's standards and their connections

As the scenario described in Section 2.5.1 focuses on web authentication and authorization, those standards are highly relevant for consideration in the implementation of this work's concept. The standards described in this section also cover the centralized personal identification and authentication sections of the categorization of IAM aspects shown in Figure 3.2. The implementation will utilize the WebAuthn API for the web scenario described in Section 5.3.

3.1.2 Public Key Infrastructure

Using PKI with X.509 certificates is the primary way of authenticating websites and encrypting traffic on the Internet via the HTTPS protocol. The PKI of the Internet is – in its simplest form – a hierarchical structure with several root CAs vetted by auditing enterprises whose public keys are included in many operating systems and browsers. Individual websites' X.509 certificates are created by those root or intermediary CAs, following further rules on how to validate the identity of websites. A user's browser can then trace the certificate chain of a website back to a root CA that is known to be trustworthy because it is part of the browser's or operating system's certificate store. Using this tree-like structure scales exceptionally well.

The certificates used in the Internet's PKI follow the X.509 certificate standard, which describes how to create those digital certificates. Basically, the X.509 certificates contain the entity's public key and associated metadata, i. e., validity period, issuing time, or cryptographic parameters, issued by CAs and stored encoded in ASN.1 syntax [114]. Due to the availability of free X.509 certificates, especially due to *Let's Encrypt* [1], the number of websites utilizing X.509 certificates is constantly rising, and the use of HTTPS is exceeding 80% of web traffic [54, 84].

This allows users to guarantee that if they enter the correct domain name into their browser's navigation bar, their browser will load and display the website's contents as intended by the website's owner. If the web server and browser are configured correctly, third parties can no longer intercept, read, or modify the page during transit to the user, as would be the case without HTTPS and X.509 certificates.

The Internet's PKI covers the IAM dimensions of organizational identification and authentication, as indicated in Figure 3.2. Certificates for identifying and authenticating users are possible but less commonly used.

3.1.3 Mozilla Persona

In 2011 the Mozilla Foundation tried to pioneer the IAM options for the Internet by developing and publishing *BrowserID* [131]. At the core of BrowserID the user could prove control of their email address in what could be considered a form of web-based SSI. The browser would store a cryptographic proof issued by the user's email provider that they control a specific email. This proof had to be renewed regularly by logging in to the email provider. When authenticating the user to a website, the user could choose which email to use and provide the website with proof of control of the email as authentication. However, the system could not gain any noticeable traction and was abandoned in 2016 [133].

This approach shows how to decentralize personal identification and authentication on the Internet, and that adoption is essential for long-term success.

3.2 Federated Identity Management

The premise of FIM is the partial decentralization of IAM, the basics of which have been described in Section 2.3. Decentralization is achieved by creating identity federations consisting of IdPs and SPs. Users usually are identified by their home IdP and can use this provider to log in to services at the SPs. During the authentication of the user at the SP, additional attributes describing the user can be supplied by the IdP. The SP can use those for authorization decisions. For example, an IdP might assert a user to be enrolled as a student at a university. A user can thus use multiple services with one user account within one federation.

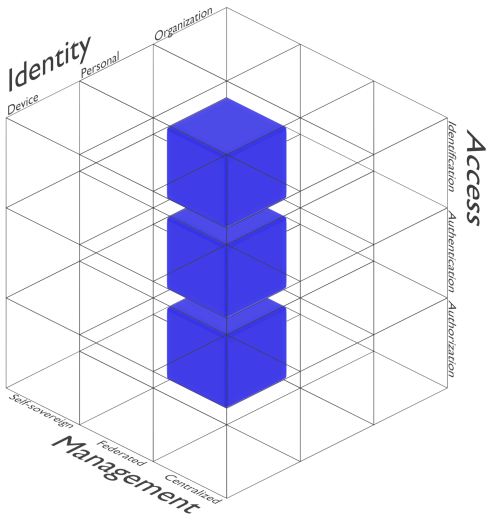

Figure 3.4: IAM dimensions for FIM applications

Within the IAM dimensions described at the beginning of Chapter 3, FIM can express personal identities in a federated environment and cover identification, authentication, and authorization. With SAML and OIDC, there are currently two

prominent systems for implementing FIM. Those are introduced in Section 3.2.1 and Section 3.2.2. Section 3.2.3 presents current research on relevant aspects of FIM.

3.2.1 Security Assertion Markup Language

SAML is a well-established standard for FIM in higher education and enterprise environments. The standard that is usually referenced if SAML is mentioned is SAML 2.0 [38]. For example, it is used for large national and international research and education federations, i. e., eduGAIN, and some commercial products.

Deployments with SAML are usually centered around identity federations [9]. Federations provide a legal framework and necessary centralized infrastructure for processing and exchanging authentication information and users' personal attributes between the different parties.

Due to the need to aggregate and distribute metadata of each participant, the federation's structure is usually pretty static and cannot be modified in "real-time". To alleviate this problem, multiple projects have researched and developed suggestions to build a more dynamic federation system [6, 56, 148].

3.2.2 OpenID Connect and OAuth 2.0

While SAML is predominantly used in enterprise and higher education scenarios, OIDC and OAuth [80] are usually used in the context of social logins, e. g., log in with Google, Facebook, or GitHub [61]. OAuth – as the name implies – only handles authorizations and enables passing authorization tokens to other applications. It does not specify requirements for identification and authentication, however. The predominant use case would allow a third party application to access specific functionalities of an application (i. e., allow an application to read a user's timeline or post to Twitter [186]).

OIDC was developed on top of OAuth to fill the gap and use OAuth as an identification and authentication framework [160]. In its actual use, OIDC is very similar to SAML. Both feature components like the IdP and the SP – or RP as the SP role is called within OIDC – and those components conceptually work almost the same in both standards. On a technical level, OIDC is based on Javascript object notation (JSON), and SAML is based on extensible markup language (XML), but both work on the HTTPS protocol.

In theory, OIDC was designed to allow anybody to run their own IdP, which could be considered UCIM. However, in practice, only established IdPs are used to access RPs.

3.2.3 Research

The current research into FIM is focused on scaling SAML federations and making federations more flexible. As a result, dynamic federations have been discussed in multiple publications. A selection of these is presented here to display the diverse approaches without the intention of providing a comprehensive list. The core idea is to open the relatively static metadata distribution, as is usually required for SAML federations, to an on-demand system. This can produce the benefit of allowing the metadata to contain situation-specific information instead of needing to accommodate every other participant of the federation.

The research utilizes a variety of solutions to achieve on-demand federations. The approach by [6] replaces the static metadata file used in SAML federations with a dynamic trust list. This list is dynamically updated by a trust engine, which is supplied reputation information via a SAML extension.

A different approach is followed by [56], who extend SAML in such a way that it can operate like OIDC. Users can then suggest IdPs to SPs and vice versa. To account for different trust levels, the user-initiated connections between providers can differ in their LoA and the amount of information shared.

In contrast to the preceding approaches, [83, 142, 145] specify and extend the concept involving a trusted third party, which can broker trust relations on demand. Utilizing a trust broker component requires fewer modifications to the SAML protocol and IdP and SP components. Additional functionality can be added to the trust broker to manage LoAs and to help provide the necessary attributes.

As part of this work, [69] proposed using DLT for managing federation metadata. The publication uses the term µ-federation to name the small individual federations that can be formed if metadata can be trusted and accessed by any participant. Metadata management through the DLT is done similarly to how certificate revocation lists (CRLs) work to strengthen the trust in the data within the PKI.

Similarly, the research by [5] also uses blockchain technology to manage SAML federations by using the blockchain as a metadata file and trust anchor list. Entries on the blockchain are managed through a middleware web application. The authors also integrate their concept with a SAML implementation and test the performance in a small test setup.

A method of bridging FIM and SSI is proposed by [192]. It includes an implementation that replaces the IdP with an SSI system while keeping the SP mostly SAML-based. Systems like this could be used to migrate or connect users to SPs that cannot quickly change their IAM infrastructure.

With a focus on FIM, [147] shows a universal framework to foster system interoperability. This framework provides processes for managing federations, including handling security incidents. The interoperability is achieved through a trusted third party, which is a central point of contact for IdPs and SPs.

This short list of research into dynamic federations shows that there has been and still is interest in streamlining metadata management for SAML federations. However, none of the proposed solutions have been adopted in a production environment.

Usually, security incidents can be handled within one company or affect only a selection of business customers or suppliers. In federations, especially in dynamic ones, reacting to security incidents can be increasingly challenging. The federated system makes coordination of the necessary responses from every participant harder. To prepare federations as best as possible, the security management framework *Sirtfi* for FIM environments aims to provide guidance [15]. This framework is actively used by the national and international federations of eduGAIN and can provide insights into how a distributed system can handle security incidents. Its current updated version is [156].

3.3 Self-sovereign Identity

SSI is a relatively new term for a concept that has been in development for a long time already. The move to decentralized identity management can be seen in the evolution of IAM solutions from centralized to federated to user-centric. So far, the user-centric approaches have not gained any large-scale adoption, as outlined with

the example of BrowserID in Section 3.1.3. SSI takes a new try at decentralized identity fueled by the gain in interest of decentralized systems promoted by the development of blockchain and DLT, which is exaggerated by the success of platforms like Bitcoin.

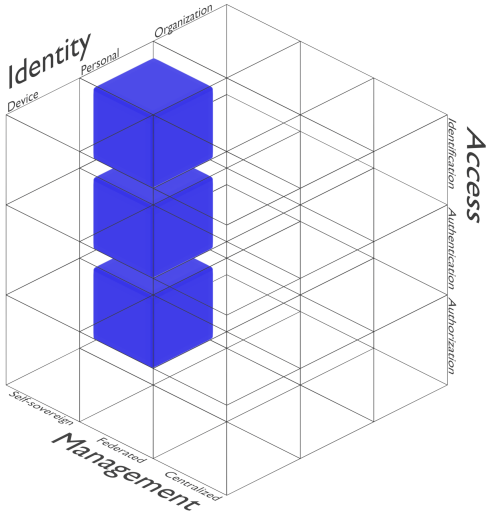

Figure 3.5: IAM dimensions for SSI

While other applications in organizations and IoT are not impossible, the main focus of SSI is personal IDM in a decentralized manner. The IAM envisioned by SSI covers all aspects of identification, authentication, and authorization, as depicted in Figure 3.5.

The current state-of-the-art of SSI is explored in this chapter as follows. Section 3.3.1 describes standards for SSI which are already published or in active development. Solutions using those standards in the form of actual prototypes or production implementations are introduced in Section 3.3.2. Ongoing research into selected relevant topics is showcased in Section 3.3.3.

3.3.1 Standards

Even though SSI is a relatively young concept, there are already strong standardization proceedings and first prototype implementations that are tested and used in real-world scenarios. The four major standardization initiatives for SSI and their relation to one another are shown in Figure 3.6. They specify the process of managing keys, identifying entities, authenticating entities, and presenting verifiable credentials between entities.

None of them are published as a complete standard by a standardization organization and are all still being actively worked on by different working groups. Nevertheless, as they are the next best thing to actual standards in the SSI environment, they will be called standards within the following chapters. The following sections will describe those standards in more detail.

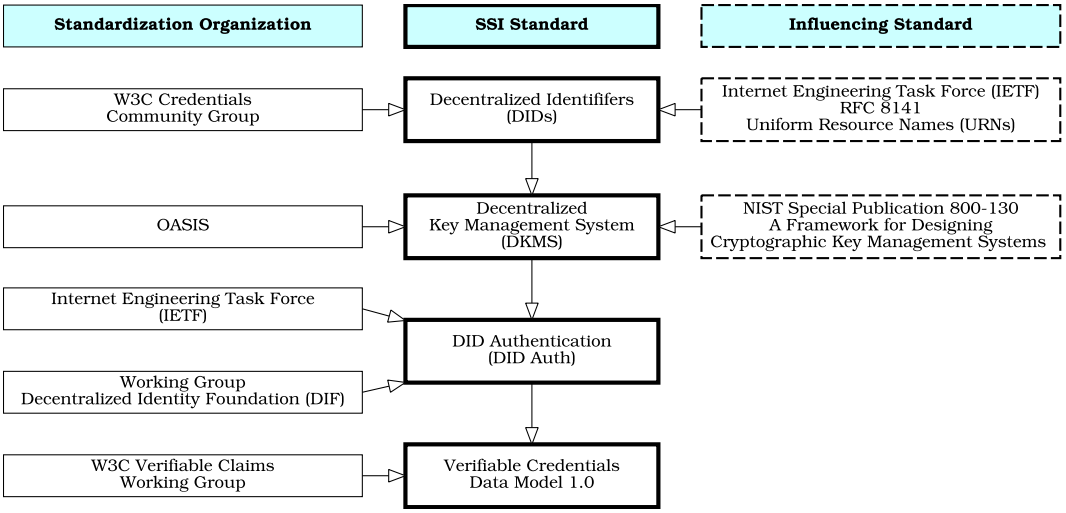

Figure 3.6: Overview of SSI standards

3.3.1.1 Decentralized Identifiers

Decentralized identifiers (DIDs) offer a standardized schema for uniquely identifying entities in a decentralized way. They feature privacy by design, interoperability, and allow the identified entity to control its identifier fully. The standardization of DIDs is currently pursued within the W3C Credentials Community Group. A current version of the proposed standard can be found in [47].

DIDs focus on providing identifiers for any entity. It is noted that while the origin of DIDs lies with SSI, their use is not strictly restricted to SSI, and DIDs may be of use in other FIM environments.

DIDs are used as the standard for entity identifiers in multiple SSI projects and implementations, for example, by the closely related Sovrin [155, 181] and Evernym, as well as in an adapted version for the Ethereum blockchain with uPort [22].

The structure of DIDs is heavily influenced by URIs as defined in [125], URNs that are specified in [106], and UUID URN namespace described in [116]. Following the URI standard, a DID consists of the following three parts. Each part is separated from the next by a colon (`":"`).

- **Schema:** The first part of a DID is always `did`
- **Method name:** The second part specifies a unique method. This method is associated with a schema to resolve a DID and fetch the corresponding decentralized identifier document (DDo) from a specific distributed ledger or any other storage system. It has to specify how to perform create, read, update, and delete (CRUD) operations for both DIDs and DDo. The method name can include colons (`":"`) to form a hierarchy.
- **Method-specific ID:** The third part contains an ID that is specific to the used method. This ID can be extended to contain further parameters and key, value pairs.

A DID can form a DID URL by appending parameters to a DID. The parameters are separated by semicolons (`";"`) from DID and each other. They can contain a set of generic (e. g., `service` for service selection) and method-specific parameters. These method-specific parameters are prefixed with the method name and a colon

(":"). A DID URL may also contain parameters from DID methods other than the one specified as the DID's method. As such, a DID URL can look like the examples in Listing 3.1.

```
Simple DID:
did:example:123456789abcdefghi

DID URLs:
did:example:21tDAKCERh95uGgKbJNHYp;service=agent;foo:bar=high
did:foo:baz:21tDAKCERh95uGgKbJNHYp;foo:baz:hex=b612
```

Listing 3.1: Examples for DIDs and DID URLs [47]

DID URLs can also be further extended after any parameters by (in that order):

- **Paths:** A forward slash ("/") indicates a path and can be used to address specific resources at a DID.
- **Queries:** A query is indicated by a question mark ("?") and can be used to address specific resources at a DID.
- **Fragments:** A fragment is indicated by an octothorp ("#") and can be used to select a specific component from a DDo.

DDos store and publish metadata about the entity identified by the DID. It contains metadata, like authentication methods and the corresponding public key, service endpoints, and other attributes and credentials in JSON linked data (JSON-LD) notation. The location where a DDo can be received is specified by the DID. An example of such a DDo is provided in Listing 3.2.

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // this key can be used to authenticate as did:...fghi
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "service": [{
    "id": "did:example:123456789abcdefghi#service123",
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Listing 3.2: Example DDo [47]

The sections of a DDo are defined in the context definition at the start of the document. There may be more than one context, but the standard specifies that the `https://www.w3.org/2019/did/v1` context always has to be the first context specified. The document referenced by the context key contains definitions of the keys used in the rest of the document.

3.3.1.2 Decentralized Key Management System

In order to build a decentralized identity storage system based on DIDs, there needs to be a way of managing the necessary keys used by the different entities to prove ownership of a DID. In a completely decentralized system, key management poses new challenges (e. g., because there is no authority that can reset a password or help recover a key). To solve this challenge, the *Rebooting the Web of Trust* Workshop wrote a document [46] outlining how a distributed key management system

(DKMS) can be built. The goal is to eventually develop a standard for decentralized key management based on the *National Institute of Standards and Technology* (NIST) SP 800-130 *A Framework for Designing Cryptographic Key Management Systems* [14] standard.

The specification [46] is short and only outlines how a decentralized key management solution should be built. This section describes the general idea of the concept.

The idea uses DIDs as the base layer for managing identities. Those DIDs are used to identify entities and are associated with defined operations that are stored on a distributed ledger layer. On top of this layer, DKMS specifies two new layers: The agent layer and the edge layer. The agent layer contains agents' communication with other agents to authenticate or prove possession of specific attributes. Those agents rely on schema information stored in a distributed ledger, specifying how to interpret values contained in DIDs and DDos.

Between the user on the identity owner layer and the software on the agent layer, there is another layer, the so-called edge layer. This layer provides the user with an easy-to-use interface to manage their DIDs. It must also store and manage the corresponding private keys used with the DIDs. This system of layers is shown in Figure 3.7.

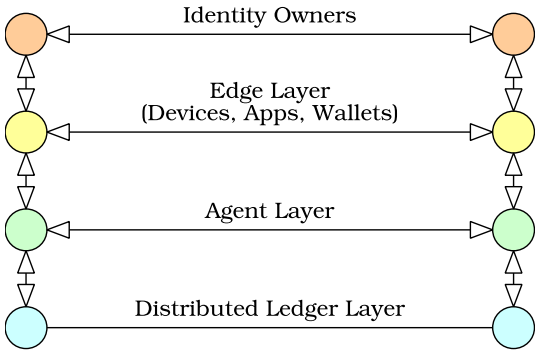

Figure 3.7: DKMS layers adapted from [46]

A design goal for this distributed system is to prevent any single point of failure. For example, there is no single entity (i. e., a popular IdP) that, if compromised (e. g., because of a hack), affects the security of many other entities' accounts. The layered approach provides interoperability, as the software that implements a layer's functions can be replaced by similar software (e. g., there can be more than one app that implements edge layer functionality). The system also specifies multiple options for recovering keys: automated backup processes, escrow services, and social recovery systems. As a result, this system provides increased resilience in the overall identification and authentication process.

The main difference between DKMS to established key management solutions like common provider-specific username/password combinations and PKI is that each edge layer entity is essentially its identity provider from the user's perspective. This leads to a highly decentralized distribution of identity information. On the one hand, a benefit is that the system is fairly resistant, as there is no single point of failure (e. g., a provider or CA that could be hacked). On the other hand, this resilience and decentralization come at the cost of only being useful if the complete DKMS ensures interoperability and provides measures for key recovery.

3.3.1.3 DID Authentication

The paper *Introduction to DID Auth* [124] defines the term decentralized identifiers authentication (DID Auth). It specifies methods that can be used to establish initial authentication between two entities identified by DIDs. There are multiple options for implementing this protocol; no single one of them has been defined as a standard. Instead, the shown options specify flexible ways for the identity owner to prove to a relying party that the identity owner controls their DID. As a result, different players within the developing SSI field use different methods for DID Auth.

Common features, however, are the following:

- Participants are identified by their DID. The DIDs used can be either public or private pairwise DIDs.
- The necessary information for authentication is stored in the participants' DDo and needs to be retrieved by resolving it from the DID.
- Authentication can be done directly between two participants (or their agents).
- Both participants must support the specified authentication method.

The technical specification is not complete yet and offers the users quite a lot of options and flexibility. As a result, a selection of architectures proposed in [124] is described in the following paragraphs. The selection is based on which of the proposed architectures are actively used by SSI projects and which offer unique technical solutions.

The first architecture describes a challenge-response authentication method for authenticating users to web services [124]. The relying party's website displays the user an authentication challenge as a quick response code (QR code) which they then scan with a wallet app on their smartphone. After selecting the user's identity, the app contacts the relying party's server via `HTTP POST` with the challenge's response. If the authentication is successful, the user is redirected within the browser session to the requested page. For users on mobile devices and smartphones, the process can be modified slightly. The resulting architecture uses deep links to redirect the smartphone's browser directly to the wallet app on the user's smartphone [124].

A variation of the workflow introduces an authentication service that receives the authentication challenge on the user's behalf [124]. The service then forwards the challenge to a device specified by the user, e. g., as a push notification on the user's smartphone. The smartphone responds to the RP with the challenge response to the RP's server. As with the first use case, multiple slight variations are proposed for this workflow, as well. For example, the workflow can be changed so the user's smartphone does not directly respond to the challenge but instead routes the response back to the RP's server via the authentication service [124]. Another variation uses the authentication service not only as a relay but with an active authentication web page [124]. The user can then interact with this web page and optionally use a smartphone or another device as an additional factor for authentication. The challenge's response is directed back to the RP's server by the authentication web page.

As many applications require authentication between entities in a non-user interactive way, some architectures describe various methods to achieve this. For those authentications, challenges between the services are passed directly to each other [124]. Those direct connections can be handled over different transport mechanisms (e. g., transport layer security (TLS), HTTP signatures, or authenticated encryption), resulting in variations of the same basic flow [124].

3.3.1.4 Verifiable Credentials

The *Verifiable Credentials Data Model 1.0* [123] is a standardization approach by the W3C to build a protocol for presenting verifiable information. In particular, the standard is used to manage credentials in a secure and privacy-conscious way on the Internet. The core element, a verifiable credential (VC), can be used to generate verifiable presentations that cryptographically prove the enclosed claims to other entities. It does so by utilizing one or more verifiable data registries that store and exchange identifiers and schemas. Furthermore, subjects can gather VCs from multiple issuers and combine them as verifiable presentations containing only the attributes requested by a verifier. An example of a VC represented in JSON is provided in Listing 3.3.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/1872",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuer": "https://example.edu/issuers/565049",
  "issuanceDate": "2010-01-01T19:73:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": "<span lang='en'>Example University</span>"
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "creator": "https://example.edu/issuers/keys/1",
    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0
      Il19..TCYt5XsITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEut
      s01mq-pQy7UJiN5mgRxD-WUcX16dUEMGlv50aqzpqh4Qktb3rk-BuQy72I
      FLOqV0G_zS245-kronKb78cPN25DGlcTwLtjPAYuNzVBAh4vGHSrQyHUdB
      BPM"
  }
}
```

Listing 3.3: Example of a VC in JSON notation [123]

The *Verifiable Credentials Data Model* provides multi-source self-sovereign authentication for all kinds of identities. Device identities are not explicitly mentioned in the verifiable claims use cases [100] or the main standard document [123]. It is, however, possible to use the standard to make claims about non-person entities.

Credentials distinguish between three entity roles: issuer, subject, and holder. The issuer creates the credential and asserts some claim about a subject. Any entity the credential presentation is shown to has to trust the issuer about the claim. The holder has the credential and can use it to present it to other entities. Usually, the subject is the same as the holder, but in some cases – e. g., a credential that confirms services that have been done to a car by a workshop – they might not be the same, and in this example, the holder is the owner of the car but not the subject. Additionally, the verifier is the entity that uses a VC or verifiable presentation to assess the subject's claims.

3.3.2 Hyperledger Indy

Implementations for SSI systems are mainly focused on web applications. Depicted in Figure 3.8 are the most notable projects that have been developed for multiple years now. The Sovrin and Hyperledger Indy systems are described in more detail in this section.

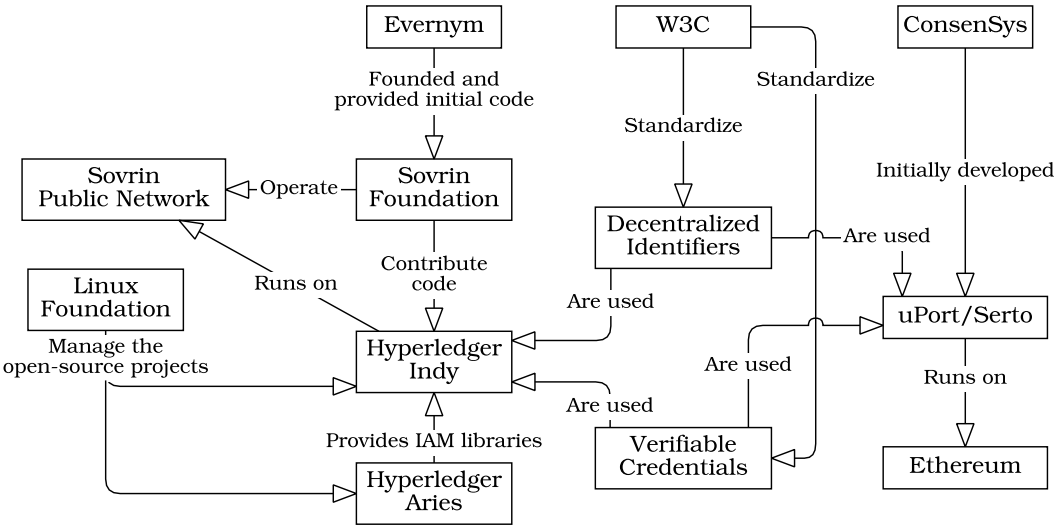

Figure 3.8: Notable SSI implementations ecosystem

Sovrin [182] and Hyperledger Indy [86] provide a public permissioned distributed ledger specialized for identity management. Using a publicly accessible distributed ledger and restricting write access through a permission system differentiates the distributed ledger from most other popular DLTs, as shown in Table 3.1.

Table 3.1: Comparison of validation and access approaches to DLT management

		Validation	
		Permissionless	Permissioned
Access	Public	Bitcoin, Ethereum	Sovrin, Hyperledger Indy
	Private	n/a	R3 Corda

Initially developed by the company Evernym the Open-Source version of their software Sovrin featured the first long-lasting effort to build an identity-focused distributed ledger. Its fundamentals and code were used by the Hyperledger Indy developers to pursue this goal further. Recent development shows a growing ecosystem of Hyperledger projects (e. g., Hyperledger Aries) that enhance the abilities of identity-focused distributed ledgers.

The technical foundations of Sovrin are described in [155]. The complete stack for Sovrin's identity ledger consists of three layers, visualized in Figure 3.9. On the bottom is a shared, globally distributed ledger to exchange information about root identities. This layer is primarily developed by Sovrin and has to be adapted by everybody using Sovrin.

On top of the base layer, agents utilize the base layer to manage the individual clients' identities. Agents can be implemented by different parties to fit different needs and act as easily addressable endpoints for individual identities. End users rely on the clients, which represent the third layer, to readily access their agents. Clients can be developed by multiple parties for different systems with differing requirements.

The distributed ledger is the shared network between all participants within the Sovrin identity network. Purpose-built – and as already described in Table 3.1 – this ledger utilizes a public access and permissioned validation schema. Permissioned validation is preferred over permissionless validation like it is used with

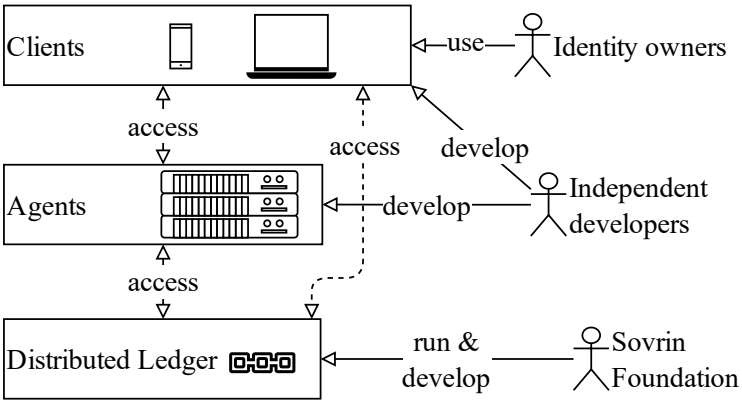

Figure 3.9: Overview of the layered approach for the Sovrin identity network

Bitcoin and Ethereum. The proof of work (PoW) protocols required for permissionless validation are energy intensive and limit scalability. However, they benefit from not requiring one node in the network to trust any other node explicitly. Permissioned validation cannot achieve this and requires a hierarchical approach to manage the validation nodes. There needs to be a set of initial nodes, which are ultimately trusted and can decide which other nodes should also be trusted with validating new transmissions.

On the Sovrin ledger, the initial trusted validator nodes are specified in the distributed ledgers genesis file, which describes the starting state of the ledger. Each modification of the distributed ledger state can be traced back to the genesis file to verify the correctness of all transactions. A consensus protocol is required to select which validator node should create the next transaction. The consensus protocol Sovrin uses is an adaption of the redundant byzantine fault tolerant (RBFT) protocol developed by [115], focusing on speed and scalability.

Besides the validator nodes, Sovrin uses observer nodes that store the complete state of the distributed ledger. Those nodes act as hot-standby to replace a validator node that might have failed and as a read-only copy of the ledger to scale read requests to a more extensive set of nodes.

Distinctively, what is described as the Sovrin distributed ledger is not a single distributed ledger. Instead, the Sovrin network uses four different distributed ledgers for different tasks [155]:

1. The **identity ledger** is primarily used to store identity records. This ledger is actively used to perform identity management operations between users. The other ledgers are used to keep track of internal ledger management.
2. Permission changes between validator and observer nodes are determined by the trustees and stewards casting votes on the **voting ledger**.
3. To manage the actual status of validator and observer nodes, the **pool ledger** is used to track those changes and the results of votes of the voting ledger.
4. General configuration settings and metadata required to run the other ledgers are stored on the **config ledger**.

In order to identify any object on the ledger DIDs and cryptographic identifiers (CIDs) are used. DIDs, as described in Section 3.3.1.1, are unique identifiers following a standardized UUID syntax. CIDs are similarly unique identifiers generated from cryptographic keys, either containing the full public key of an asymmetric key pair or (parts of) the hash of the public key.

A key to self-sovereignty is the identity owner's ability to control their identity completely, but the identity must also be usable effectively. Effective use (e. g., exchanging messages or verifying claims) requires the identity to be permanently addressable, which in turn requires a permanently available hosted service called an agent. The addresses required to exchange information with an identity are the identities' endpoints. The agent is a broker between the identity owners and their endpoints.

Besides connecting the user's client with a permanently available endpoint, the agent can also serve as storage space for data related to the managed identity. Because the data is stored with the agent, it does not have to be stored on-chain and, therefore, can contain personal information. Automatic rules could govern which data is disclosed to whom in real time. For example, the agent's storage can synchronize identity data between multiple clients. Examples of data stored with the agent described by [155] are all in the form of graphs and contain the following:

1. The **identity graph** describes the identity's relation to other identity claims.
2. A general address book-like **relationship graph**.
3. The **reputation graph** stores reputation claims that can show the identity's gathered reputation.
4. The **data graph** contains other identity data (e. g., pictures, videos, or text).

The data stored with an agent should be stored so it can be easily transferred to another agent, as vendor lock-in at a specific agent would contradict SSI goals.

For most users, the easiest way to acquire an agent and an endpoint is to use an agency. Agencies are services that provide agent services to multiple identity owners. However, every identity owner can also set up their agent on their server. Some agents are available as open-source implementations. Users might even skip the agent entirely and only use the wallet directly. The latter option may result in reduced functionality.

Managing one's identity is done through a client application. The primary functions of the clients are the initial provisioning of the user's identity, the management of the user's keychain, and the processing of claims related to the user's identity. Those clients are usually considered smartphone apps but can also be a browser or general operating system extension. In actual use, an identity owner might use multiple different clients to manage their identity. As with the agents, clients can be developed by multiple parties, and there are multiple open-source implementations already.

Provisioning a new identity in the Sovrin identity ledger requires the user to know a trust anchor, an entity with sufficient trust level allowed to add new identities. Using a challenge-response protocol, the new user proves control of the new identity's private key, whose corresponding public key is then added to the ledger as an identity record by the trust anchor.

As with most distributed ledgers and blockchains, the ultimate key to control tokens (i. e., an identity record) on the Sovrin ledger is the user's private key. The client is used to initially provision the user's identity on the ledger and is subsequently responsible for generating the user's set of public and private keys. The client's vital task is to protect this key adequately against compromises, corruption, and accidental loss.

Clients may also contain functionality that allows users to rotate their keys, i. e., exchange an old, eventually compromised key pair for a new one, or revoke their keys entirely, i. e., rotating keys without adding a new key, rendering the identity unusable.

Key recovery is a tough problem for distributed ledgers and SSI systems in general. If you are your own IdP, you can not get help recovering lost keys or credentials from any organization. Besides robust backups, the solution to this problem is peer recovery. A set of close trusted friends can issue a key rollover upon the identity owner's request to add the user's new key. The set of trusted friends is time-locked to prevent an attacker who compromised the user's client from changing the set of recovery contacts and immediately changing the key to lockout the identity owner [155]. Within the time window, the original identity owner could still recover the identity (after fixing the vulnerability that resulted in the compromise) using the previous set of friends to lock out the attacker again. Depending on the length of the time window chosen by the user, it may be important to use active monitoring of one's identity and push notifications for specific events, i. e., changes to keys and recovery contacts.

While key management is an important factor for the client, the users will primarily use the client for managing their identity, claims, and associated data. Sovrin defines claims as any statement made about an identity, even if this statement is self-attested, and uses the term verifiable claim to describe claims by third parties [155]. Verifiable claims are also being standardized by the W3C's *Verifiable Claims Task Force* (VCTF) [123].

Claims can be made either public (on-chain) or private (off-chain). It is strongly advised to keep all claims containing private information private. Private claims can be managed and distributed by the identity owner's agent if necessary.

Beyond the differentiation by storage location of public and private claims [155] introduces the following general types of claims.

1. The most basic claim, the **cleartext claim**, contains all information in plain text. It should only include information that is intended to be released publicly.
2. **Encrypted claims** are similar to cleartext claims but use asymmetric or symmetric encryption to encrypt their information.
3. **Hash signature claims** are a specially crafted tree of claims that can be disclosed individually to different other identities.
4. **Proof of existence (POE) claims** use cryptographic signatures to generate a hash value from any digital object and store this hash within the claim, thus proving the existence of the digital object at a specific point in time. These claims are also called hash claims, but this name can be easily confused with hash signature claims.
5. **Anonymous credentials** are the most complex type of claim, containing neither cleartext, hashed, nor encrypted information, but instead use zero-knowledge proofs to prove statements. They can be used, for example, to show the claim "is over 18" without providing the actual birth date.

The organizational aspects of running the Sovrin identity network are described by [155, 196]. They are divided between the organization of the entity running the Sovrin network, the Sovrin Foundation, and the identity management aspects of the network itself.

Overall, the organizational structure of the corporation "Sovrin Foundation" follows a functional approach, as depicted in Figure 3.10.

The board of trustees governs the operation of the Sovrin identity network by selecting stewards, who run the Sovrin nodes, and deciding over the implementation of recommendations of the Technical Governance Board, who control the open-source side of the software development [155].

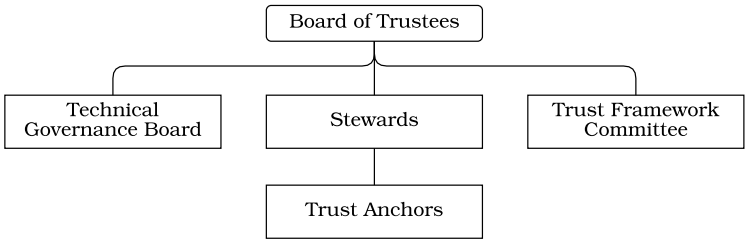

Figure 3.10: Organizational structure of Sovrin

Members of the board of trustees are the managing directors of the Sovrin Foundation. They are elected for terms of one year [18]. The board of trustees uses advisory committees like the Technical Governance Board or the Trust Framework Committee to decide on specific developments.

For example, one of the tasks of the Technical Governance Board is to develop semantics for claims, including specifying a schema, building ontologies, and publishing dictionaries. To improve interoperability, this is important to form a common understanding of the meaning of specific claims.

Management of the identities on the Sovrin ledger is done by combining two well-known approaches: The web-of-trust and a hierarchical trust model. The former has been established as a trust management solution by pretty good privacy (PGP) [63]. The system uses P2P connections between users and derives trust by spanning a graph over the connections formed by the users signing each other's keys. On its own, this system is susceptible to Sybil attacks because creating new (fake) entities is free. To combat this, Sovrin uses trust anchors that are selected based on a reputation system and have to vouch for new identities added by them.

The trust in trust anchors is not derived from their connections. Instead, they are assumed to be trustworthy. The trust of entities is called trust level, and the highest trust level allows an entity to become a trust anchor.

3.3.3 Research

Research around SSI explores a wide variety of topics, from public blockchains [48] and smart contracts [167] to IoT [4, 53]. The following research is the most relevant for the standards and implementation of SSI described in the previous sections. Section 3.3.3.1 shows options for revoking and deleting credentials and associated data. This is important for compliance with GDPR. To branch out from personal identities, the transfer of standards and implementation to IoT and organizational identities is explored in Section 3.3.3.2.

3.3.3.1 Revocation

Revocation of digital certificates like VCs is difficult to achieve and shares similarities with the challenges of revoking X.509 certificates on the Internet's PKI [108]. Taking the issues of X.509 certificate revocation into account, a solution proposed by [3] is similar to how online certificate status protocol (OCSP) works for X.509 certificates. It can be used to issue short-term attestations showing that the VC has not been revoked. This provides a benefit over utilizing short-lived VC as the non-revocation attestation can be performed by a dedicated revocation provider and does not involve the original issuer of the VC to re-issue it.

An overview of revocation methods suggested for use in SSI is provided by [60]. In their research of related work, they find and differentiate between list-based, cryptographic accumulators, and credential update methods for achieving VC revocation. They also propose a new revocation method that issues an additional VC for every VC, which attests to the validity of the linked VC. This procedure is similar to the one proposed by [3]. The paper by [60] performs a survey that finds strengths and weaknesses in the domains of privacy, scalability, and maturity for all mentioned methods. It thus concludes in the title that "the perfect revocation method does not exist yet" [60]. Indeed, revocation requires either a central trusted third party, additional direct communication between the issuer and verifier, or publishing some metadata of VCs to the DLT.

Another challenge is that with SSI, the validity of a VC should also be determinable in an offline scenario. This further increases the difficulty of proving that the VC is currently valid. Another paper that proposes a new revocation protocol and compares it to the related work is published by [35]. The authors' method works based on a gossip protocol and can achieve offline verification of revocation without the need for a centralized authority. All issuers and verifiers need to be part of the gossip P2P network. Occasionally, they can receive up-to-date revocation information, sent as batches of SHA3-256 hashes of the revoked credentials.

3.3.3.2 Identities for general entities

The introduction to SSI and its state-of-the-art in Section 3.3 focused primarily using SSI for personal identities. While that is currently the central application area, some research also explores its use in IoT and corporate identities. The research into corporate identities, however, is lacking relevant publications, leaving all non-personal identity research for SSI to fall into the category of IoT identities.

The research by [53] explores the similarities and differences of identity management approaches by PGP, PKI, and SSI with a specific focus on IoT. They find that SSI can improve IAM for IoT by providing a privacy-oriented decentralized solution that leaves control over identities with the devices' owners.

An architecture for IAM for IoT with SSI characteristics is also presented by [64]. In contrast to the other approaches discussed, the authors utilize IOTA, a special DLT that does not operate on a chain of transactions but a directed acyclic graph (DAG). The paper shows solutions for IoT device enrolment, issuing of VCs, a web-of-trust-style trust scoring system, VC revocation, and key recovery mechanisms.

Special care to include GDPR in the considerations around VC revocation is taken by [8]. The research proposes to use the GDPR-mandated access of users to their data with any service operating in the EU to create VCs. If the user could generate VCs by requesting them from any provider within the scope of GDPR, access to VCs would become much easier.

Another approach to transforming existing information into provable attestations is described by [194]. The approach proposed by the authors allows users to prove any content delivered via TLS to be originating from the website indicated by the X.509 certificate. This can turn almost any website into an issuer without the need for any modifications to the website.

3.4 Internet of Things

Identity management for IoT focuses on devices and their identification, authentication, and, to some extent, authorization. With centralized approaches, authorization is usually not an issue, as authorization decisions can be made entirely

within the central system, and any authenticated system can access at least some parts of it. With federated approaches, the authorization gets more complicated, as the decisions can no longer be performed internally but sometimes rely on input from other federation participants.

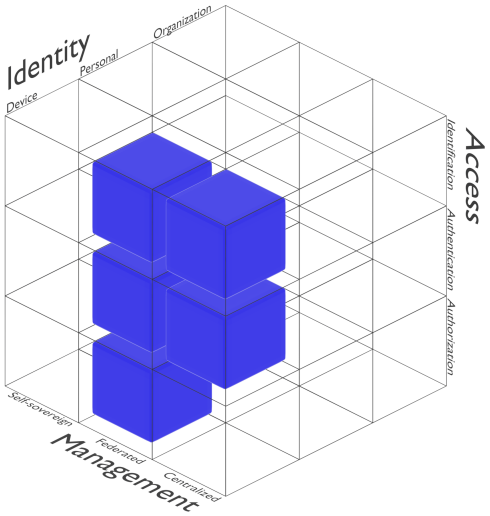

Figure 3.11: IAM dimensions for IoT

The resulting relevant sections of the IAM dimensions are depicted in Figure 3.11. The necessity for IAM in IoT scenarios is especially present in large-scale networks, as depicted by the scenarios in Section 2.5. As a result, Section 3.4.1 explores state-of-the-art IAM in challenging conditions of LPWANs. Further relevant research based on IoT is summarized in Section 3.4.2.

3.4.1 Low Power Wide Area Networks

LPWAN technology offers a method of reducing IoT networks' dependency on cellular, i. e., GSM, 4G, or 5G networks, and outrange the coverage of wireless protocols like Wi-Fi and Bluetooth. An overview of different technologies and standards for LPWANs, i. e., LoRa®/LoRaWAN® or NB-IoT, is provided by different publications like [34, 93].

A closer look at how IAM can be implemented in an LPWAN protocol is shown with the example of LoRaWAN®. The LoRaWAN® protocol stack is used in two different protocol versions. The original series `1.0.X` with `1.0.4` [121] currently being the most recent version and an improved version `1.1` [122]. Version `1.1` fixes some of the issues that resulted in potential security threats in the earlier `1.0.X` versions. Both versions are not directly compatible, especially on the network side, as `1.1` introduces new components to allow for more flexible key distribution and roaming. An overview of the two major versions' architecture is shown in Figure 3.12 and Figure 3.13.

The network server handles join requests, i. e., an end device's first message to the network to establish a connection and generate the required encryption and message authentication code (MAC) keys. In the original LoRaWAN® specification for versions `1.0.X`, the network server stores the *application key*, which is used to derive new *application session keys* and *network session keys*. In version `1.1`, the network server only handles *session keys*, and the join server holds the root *network* and *application key*.

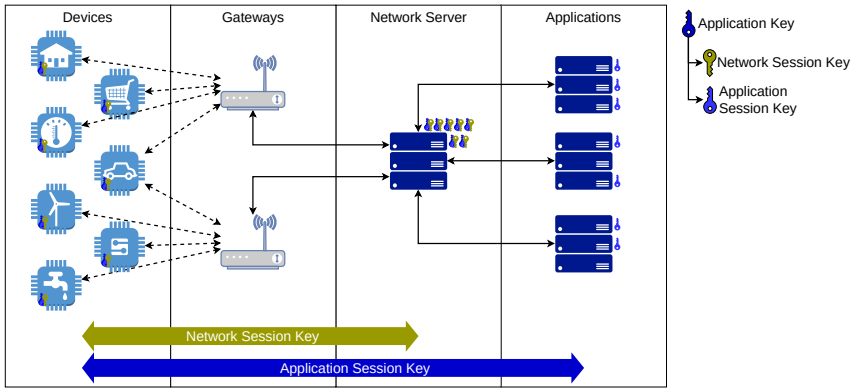

Figure 3.12: LoRaWAN® key management architecture of version `1.0.x` [121]

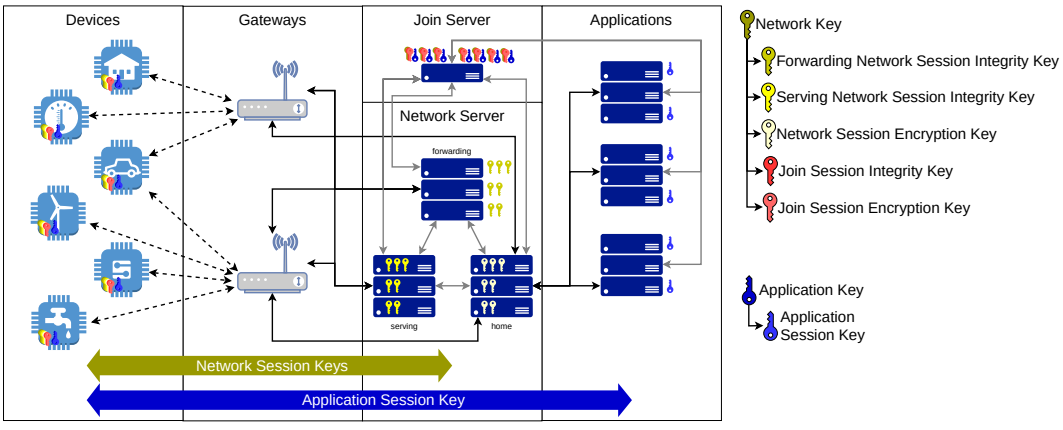

Figure 3.13: LoRaWAN® key management architecture of version `1.1` [122]

The example of LoRaWAN® shows that efficient LPWAN prefer using symmetric keys for authentication. Multiple symmetric keys are required to allow roaming between different network operators and to support an infrastructure where individual entities run gateways and applications. Additional forward secrecy and integrity protections require even more keys, that must be derived from some root keys. Other LPWAN standards also rely on symmetric encryption (e. g., NB-IoT) or leave encryption of messages to the application developer (e. g., Sigfox) [152].

3.4.2 Research

Some of the research into the combination of SSI and IoT was presented in Section 3.3.3.2. This section focuses on research for IoT that is independent of SSI and helps with understanding current IAM solutions for IoT and adapting new ones.

An overview of requirements, challenges, and existing standards for managing IoT devices is provided by [170]. This publication compares many protocols (i. e., SNMP and NETCONF), frameworks (i. e., ITU-T), and platforms (i. e., AWS IoT and the Google Cloud Platform). The paper identifies LwM2M as the most accepted management protocol for IoT [170]. However, any reference to IAM is missing from the paper's descriptions.

Further comparison of management standards for IoT is provided by [2]. The authors compare many of the same approaches for network management and cloud-based management as [170] but also include frameworks for software-defined networking (SDN), semantic-based approaches, and machine learning. The authors can not determine a clear favorite for any application in IoT, as all approaches have specific strengths and weaknesses [2]. IAM is absent in this comparison.

As a basic IDM solution, the publication by [185] presents an architecture for IoT device management with an authentication and authorization framework. The paper uses an identity store to hold all IoT devices' identities and issues tokens that can be used by the devices to perform service requests.

The research in [154] explores the use of blockchain technology to simulate the backend database of a smart building with multiple IoT devices. They compare different DLT varieties from Hyperledger, BigchainDB, and MongoDB as a reference for traditional databases. Their research shows a significant overhead in write and response times by the DLT solutions, which stays about one second above the reference in the best case. This latency can be acceptable in some instances, where it is less relevant, and the advantages of DLT outweigh it.

Another evaluation of applying blockchain technology for IoT and IAM is performed by [158]. They use LoRaWAN® as the IoT protocol and aim to increase the network's reliability by distributing the join server, which can be considered a single point of failure.

On a more fundamental level of research, the security of any IAM protocol is heavily dependent on cryptography. For security in any IoT application, cryptography decides if the CIA goals can be protected. In IoT devices that are optimized for long battery operation, the choice of cryptography is often a trade-off between choosing strong cryptography and keeping code size, traffic, CPU, and memory usage low.

A protocol for distributing keys emphasizing low-power computing environments is described by [188]. The author defines three successive protocols for exchanging keys in a distributed network. The protocols are based on symmetric encryption and offer a configurable amount of node failures or compromises until the network's security is broken.

The research of [43] explores different lightweight cryptographic algorithms for IoT extensively and shows their strengths and weaknesses. The core comparisons shown by the authors are lightweight block ciphers, stream ciphers, hash functions, and elliptic curve cryptography (ECC).

3.5 Electronic Identification (eID)

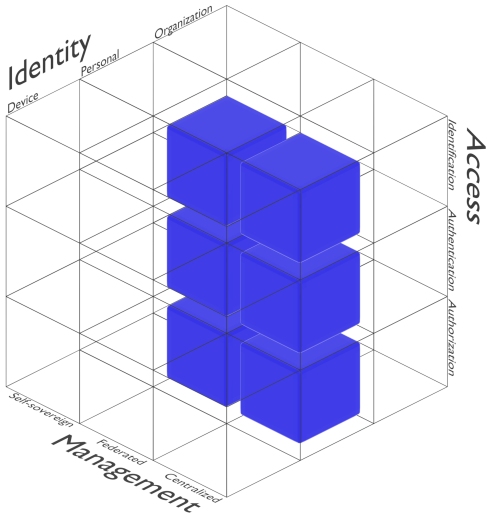

Figure 3.14: IAM dimensions for eID applications

A report from the *World Economic Forum* on digital identities identifies standards as the foundation of all identity systems [130]. The report also notices that the existing standards "lack of coordination and consistency" [130]. A comprehensive specification of the core eID aspects – Identity, Authentication, and Federation – is provided by NIST SP 800-63-3 [74]. NIST SP 800-63 provides information and technical guidelines for digital authentication, focusing on federal systems [74]. The guidelines also include advice for issuing and verifying credentials.

A core mechanic of NIST SP 800-63A is the use of three dimensions for LoAs:

- The **identity assurance level (IAL)** specifies how thorough the mapping between a real-world entity and the digital entity described by the identity has been checked.
- Secondly, the **authentication assurance level (AAL)** shows how strong a current session has been authenticated (e. g., by simple password, MFA, or hardware tokens).
- If federation is used, the **federation assurance level (FAL)** can indicate how secure the communication of an assertion to an RP is and whether the user is aware of the transmission.

In general, though, many different standards affect a small portion of the bigger picture. A selection of those standards by international standardization organizations is categorized by [132] into the following five categories and compared against the standards used for some countries' implementations of eID systems: Biometrics, Cards, Digital signature, Bar code, and Federation.

3.5.1 Standards

A selection of important eID standards is presented in this section. Standards regarding entities' identities can encompass various aspects. For example, they can specify how entities are identified, how they are enrolled and their identity is verified, or how identity data is stored. Standards focusing primarily on the identity of entities are described in Sections 3.5.1.1 to 3.5.1.5.

Standards for access control usually cover topics like suitable authentication, encryption, or signature methods, including, for example, MFA or attributes and their respective LoAs, and utilize, for example, the PKI described in Section 3.1.2. Those standards are described in Sections 3.5.1.6 and 3.5.1.7.

The management side of eID is covered by the NIST SP 800-63C, described in Section 3.5.1.8.

3.5.1.1 NIST SP 800-63A

The NIST SP 800-63A standard consists of ten sections, of which two are normative while the rest are informative [76]. The normative sections describe the requirements for the different IALs and requirements for resolving, validating, and verifying identities.

In general, the identity establishment process consists of three steps:

1. resolution of attributes and corresponding evidence,
2. validation of the gathered evidence (i. e., are the attributes correct), and
3. verification of the connection between the attributes and the entity (i. e., do the attributes belong to the claimed entity).

For each of those steps, the standard specifies requirements that can be fulfilled in five different strengths: unacceptable, weak, fair, strong, and superior.

As additional information, the informative parts of the standard describe different threats, privacy, and usability considerations.

3.5.1.2 Cryptographic Modules FIPS 140-2

As many eID systems use identity cards to store information, the standards regarding their cryptographic modules are important to look at. One of the standards covering cryptographic modules is *Federal Information Processing Standard* (FIPS) 140-3. It covers the design and implementation of cryptographic modules, including interfaces, authentication, key management, roles, and security regarding physical attacks and electromagnetic interference [180].

The security levels described by the standard are levels 1 to 4. They describe increasing protection levels by specifying consecutively higher requirements for the cryptographic module. As such, the levels focus on preventing physical access and specify which limitations must be met by the general computing platform running the software.

3.5.1.3 Machine Readable Format ICAO 9303 (ISO 7501)

The *International Civil Aviation Organization* (ICAO) publishes aviation and travel-related standards. As part of this, they specify many procedures and requirements for international cooperation and general safety. For example, they also define the four-letter airport identification codes.

For travel documents, the ICAO specifies in Document 9303, which is also partially endorsed by ISO/IEC 7501, a machine-readable format for identity data [91, 177]. In this context, "machine-readable" refers to optical character recognition and specifies three types of fields of different character widths and line counts. The information encoded in this field must include the passport holder's name, passport number, nationality, date of birth, sex, and expiration date [91].

3.5.1.4 ISO/IEC 14443 Contactless Chip Cards

Many eID cards use ISO/IEC 14443 for card-based contactless identification, access, and payment systems. This standard's prevalence is also because it is part of the ICAO's regulations for machine-readable identification documents and passports used in international air travel. For example, it is used with the German identity card *neuer Personalausweis* (nPA) [23] and the German passport [112].

The standard calls chip cards proximity integrated circuit cards (PICCs). Compatible reading devices are called proximity coupling devices (PCDs). The standard consists of four parts that specify physical characteristics, radio properties, communication and anti-collision mechanisms, and contactless transmission [175].

3.5.1.5 Biometric Interface ISO/IEC 19784

As more and more identity systems use biometrics as an additional identification feature, a standardized API is needed to capture and verify those across many vendors' devices. The *BioAPI 2.0* standard specified by [178] aims to provide such a specification and – while application independent – is often used for official documents, e. g., passports [25].

The *BioAPI* framework bridges individual applications to specific biometric capture devices via biometric service providers (BSPs). As a result, the applications can interact with various capture devices without having to manage device-specific operations like feature extraction, filtering, and matching.

3.5.1.6 NIST SP 798-63B

Part B of NIST SP 798-63 specifies the types of authenticators that can or cannot be used to achieve a specified AAL [75]. The authentication methods differentiate between memorized secrets (i. e., a password), look-up secrets (i. e., printed recovery codes), out-of-band (i. e., SMS, QR code, or Bluetooth-style code verification), one-time password (OTP) devices, and crypto software or devices. While the lowest AAL 1 does allow all of those authenticators, the highest level AAL 3 requires a (multi-factor) crypto device in combination with a memorized secret, an OTP device combined with a multi-factor crypto device or software, or an OTP device and crypto software and memorized secret [75].

Additional aspects regulated by the standard are FIPS 140 verification, re-authentication, security controls, MITM resistance, verifier-impersonation resistance, verifier-compromise resistance, replay resistance, authentication intent, records retention policies, and privacy controls. Somewhat surprisingly, replay resistance is only required for AAL 3.

The standard also details authenticator lifecycle management, session management, threats and security considerations, and privacy and usability considerations.

3.5.1.7 Digital Signature Standard FIPS 186-4

One key technology to authenticate entities beyond using PSKs is signatures. The [179] standard specifies requirements for signatures using *Digital Signature Algorithm* (DSA), *Elliptic Curve Digital Signature Algorithm* (ECDSA), and RSA.

For each signature method, the standards specify – where applicable – the permitted parameters, hash functions, key pair generation and management, secret and signature generation, and signature verification processes. The definitions in the standard are mostly mathematical and do not feature concrete implementation advice.

3.5.1.8 NIST SP 800-63C

NIST SP 800-63C handles the communication between IdPs and RPs in a federated setting [77]. The central part of this document specifies requirements for the protocol and form of the assertions at the different FALs.

For the assertions, the standard differentiates between two general types of assertions: bearer assertions and holder-of-key assertions. The former can be used by any entity to prove the bearer's identity and attributes to the RP. This kind of assertion is usually used with *OpenID Connect Basic Client* profile or *SAML Web SSO Artifact Binding* profile assertions [77]. The latter assertion type can only be used by the entity that holds the key referenced in the assertion. The entity must directly prove possession of the referenced key to the RP.

Again, the standard follows the normative requirements with informative sections about security, privacy, and usability considerations.

3.5.2 eID Implementations

Actual implementations of eID systems exist in most countries. An overview of worldwide eID systems focusing on SSI was researched and published in [143] in preparation for this work. The German approach to eID, which features centralized, federated, and self-sovereign characteristics, is presented here for reference.

The German official identity document is the German identity card (*Personalausweis*). Since a revision in 2009, identity cards issued after the first of November in 2010 contain a chip that contains all the card's information in digital form and also can be used for online authentication [26]. This new electronic identity card is called the nPA. It contains the usual identifying information like name, date of birth, address, etc., in readable, machine-readable, and digital formats. The readable and machine-readable parts are largely irrelevant for eID. The key digital information is stored on a chip that can be read with an NFC reader. This information also includes a certificate that can be used for online authentication by the identity card holder.

Historically a key weakness of the system was the reliance on special hardware adapters that could connect to personal computers and read the digital information from the identity card. Those devices had to be purchased separately, and adoption is pretty sparse. As an alternative to those readers, many current smartphones are equipped with the necessary NFC readers and can be used with the *AusweisApp2* to read the identity cards [137]. This app can also wirelessly connect to an application on a personal computer within the same LAN and thus be used to log in to governmental websites on a desktop.

3.6 Summary

As part of the SSI-focused dive into standards, technologies, and research around IAM, many sectors of the three-dimensional representation of IAM categories are explored in this chapter. Sectors without relevant state-of-the-art are depicted in Figure 3.15. Most notably, self-sovereign management of devices and organizations is missing, as well as centralized authorization aspects for any kind of identity. The latter is less relevant, as authorization decisions can be taken as required by the use case in a centralized system. The lack of state-of-the-art SSI for devices and organizations can be attributed to SSI being relatively new and focusing on personal identities. Those parts of SSI concerning device and organizational identities will be referenced in further chapters as necessary.

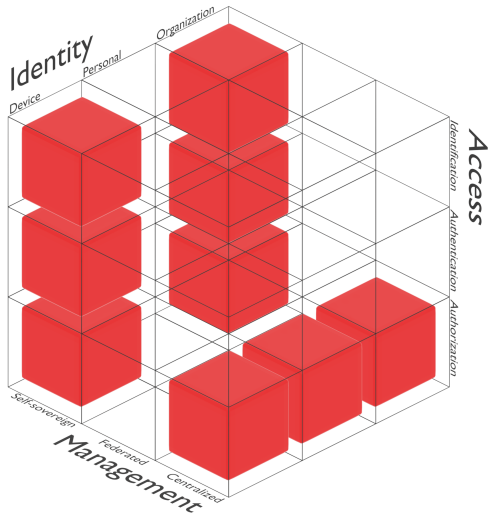

Figure 3.15: IAM dimensions not covered by the state-of-the-art

Not every standard, research paper, or implementation can be mapped against the requirements from Section 2.5, as they often only cover particular aspects of one or two requirements. Instead, the most relevant systems and general topics are selected for comparison:

- **FIDO 2** is chosen as it provides the most complete framework for identification and authentication on the Internet.
- **Mozilla Persona** offers the ability to compare an ultimately failed approach with many similarities to SSI.
- **Dynamic FIM** offers state-of-the-art and time-proven IAM in a partly decentralized environment. Many lessons can be learned here to improve SSI.
- **SSI/Hyperledger Indy** is the go-to implementation for SSI and is under constant development with many supporters. It shows what is currently possible with SSI.
- **German eID** highlights the challenges that arise from building eID systems.
- **IoT/LoRaWAN®** adds the perspective of an IoT-focused environment dealing with IAM for many devices and experiencing decentralization through a roaming protocol.

Section 3.6.1 to Section 3.6.6 discuss the individual requirement categories: satisfaction, information, consistency, security, data protection, and robustness. A combined evaluation is provided in Section 3.6.7.

3.6.1 Satisfaction

The requirements categorized with satisfaction are necessary to satisfy the primary purpose of an IAM system. Not every system requires all of them, but some are common to all. Of the selected systems, all support requirements **Identification** (SAT3), **Authentication** (SAT1), and **Identity provisioning** (SAT4). As indicated by Figure 3.15, authorization is less covered, which reflects in the fulfillment of requirement **Authorization** (SAT2). Authorization is only partly covered by FIM, as it enables both IdP as well as SP to control authorization by allow or deny listing specific combinations of services and users. Hyperledger Indy also covers authorization by utilizing VCs similarly to assertions in FIM. The LoRaWAN® roaming infrastructure uses decentralized authorization to determine which devices can join the network.

Further requirements get more specific and are not covered by all state-of-the-art. Requirement **Access delegation** (SAT6) is only partially covered by ideas within the SSI space and thus also Hyperledger Indy. Delegation of access decisions is part of the roaming protocol developed for LoRaWAN®. All other systems do not consider access delegation.

Without external information, a decentralized system cannot reliably identify which entities are trustworthy and which may try to impersonate others. To establish the necessary trust requirement **Trust establishment** (SAT5) is covered by dynamic FIM and SSI with Hyperledger Indy. Especially the dynamic FIM research discusses establishing trust between participants in detail. However, the participants are usually not wholly unknown beforehand. The SSI approaches also cover first connections but stay within the realm of technical trust establishment and vaguely cover organizational trust by using non-decentralized trust anchors.

To scale any network of entities, there should be a way to connect them without manual configuration. This is covered by requirement **Automated integration/registration** (SAT7). Similar to requirement **Trust establishment** (SAT5), semi-automated connections are supported by dynamic FIM and SSI approaches. Additionally, also LoRaWAN® roaming can automatically configure devices according to foreign networks and join the device.

With IAM systems dynamically incorporating new devices or through a large system expansion adapting different syntaxes, formats, or languages of identity attributes becomes more important. Requirement **Identity data set matching** (SAT8) can be partially fulfilled by dynamic FIM. The other technologies where this might be applicable, SSI with Hyperledger Indy and the German eID system, do not support the automated adaption of identity attributes.

As an extension of plain IAM, a backchannel to message known entities is required by requirement **Message delivery services** (SAT9). In static FIM constructs, this can always be achieved by the application layer. In situations like dynamic FIM, SSI, and eID this is not trivial. Especially with eID, it is legally required to be able to send and prove sending messages to citizens. The three systems can support those messages on an application level. However, SSI could be adapted to integrate message delivery into the digital identity and completely fulfill this requirement.

3.6.2 Information

Requirement **Credential establishment** (INF1) is usually needed to associate credentials with an identity for authentication. It is fulfilled for all systems that were explored with the exception of dynamic FIM and the German eID system, as those two have external processes for establishing credentials.

The usefulness of a system is decided by its applicability in specific situations. Requirement **Documentation** (INF2) aims to ensure that systems provide the proper documentation to adapt and use them. All systems except the orphaned Mozilla Persona project are documented either very well or at least sufficiently.

Connecting the digital and physical world is the job of requirements **Digital identification** (INF3) and **Physical identification** (INF4). Only LoRaWAN® can partially fulfill the digital identification part, as it supports preloading identity information as part of the manufacturing or procurement process and utilizes QR codes to retrieve those IDs from the physical device or packaging. In all other systems, retrieving a digital identity from a physical device must be handled in both procurement and application. The other way around, identifying a physical device from a digital identity is also considered to be part of an application's feature set.

Public information about an entity, especially IoT devices or public organizations, should be available with knowing the entity's identifier, as specified by requirement **Product specification** (INF5). This is realistically only the case in IoT environments, including depending on the software stack that is used LoRaWAN®.

Discovering devices or services within a system is handled by requirements **Neighbour discovery** (INF6) and **Service discovery** (INF7). None of the state-of-the-art covers those aspects. However, the requirement **Capability exchange** (INF9) is closely related, which allows entities to exchange and negotiate capabilities once they know each other. This requirement is partially covered by dynamic FIM approaches and SSI. Continuing in the process from discovery to capability exchange to processing requirement **Agreement monitoring** (INF8) allows entities to monitor agreements or terms of contracts autonomously. Facilities for this are part of SSI approaches, especially if non-personal data is shared through a DLT.

In every application – but especially in eID – it is required that every entity can show certain attributes. This is handled by requirement **Identity data set** (INF10). The presented solutions partially support this for dynamic FIM and SSI and fully support this in the German eID case, as law requires.

Requirement **Level of assurance** (INF11) assumes that in a larger IAM system, not every assertion of attributes can be made with the same level of confidence. The LoAs represent the confidence in the assertion according to predefined criteria. This is already possible with FIM and eID and partially covered by SSI. For SSI, a good way of creating standardized and meaningful LoAs is not yet determined.

3.6.3 Consistency

As a core concept of IAM, credential recovery is important and documented by requirement **Credential recovery** (CON2). All solutions need to consider it. FIDO2, Mozilla Persona, dynamic FIM, German eID, and LoRaWAN®, however, do not specify a process for recovery within the respective system. Instead, the application layer needs to support some alternative authentication method that would allow the re-registering of new credentials. The SSI approach is more integrated and discusses recovery methods. Even though they are more complicated, with SSI, no central service desk can help with recovery.

Related to the recovery is the de-provisioning of an identity described in requirement **Identity de-provisioning** (CON1). While it concerns all systems, most leave the implementation details to the applications. SSI needs to cover de-provisioning and shows processes for both private and public identities. The latter cannot be permanently deleted, depending on the used DLT, but they can still be marked as no longer active.

The requirements **Digital twin** (CON3) and **Content verification** (CON4) aim at IoT applications to keep data between the digital and the physical world consistent. Those requirements are partly the duties of the respective applications but could be integrated or assisted by IAM in certain scenarios, as shown by SSI IoT use cases.

A problem for distributed IAM systems is the potential to diverge due to connectivity issues. As a result, requirement **Netsplit/Join** (CON5) seeks to create such systems in a way that ensures consistency when the systems reconnect. The only fully decentralized system where this might apply is SSI, which uses consent algorithms that can handle such situations depending on the respective implementation.

A requirement from the eID scenario that can also be of value in other areas of application is requirement **Once-only** (CON6). For FIM, SSI, and eID, this can be partially fulfilled. Once the attributes are stored at an IdP, wallet, or eID card, they do not need to be re-entered when visiting different services. The full idea of services sharing information without the user having to re-submit it is not fulfilled.

3.6.4 Security

All modern approaches to IAM should conform to requirements **Security by design** (SEC5) and **Security by default** (SEC4). This is the case for the selected systems.

A part of this, security by design is featured in requirement **Mutual authentication** (SEC3) to require mutual authentication. This prevents impersonation of entities, on either side. There are different approaches to this shown by the state-of-the-art. FIDO2 solves the problem by generating a new key pair for every service the user registers for. This is similar to the pairwise DIDs of SSI. Mozilla Persona and FIM rely on using X.509 certificates to authenticate the websites to the user. Therefore, mutual authentication is out of scope for those. The German eID system also utilizes certificates to authenticate the service to users and their eID cards. LoRaWAN® supports end-to-end encryption and authentication between IoT devices and the application server.

Another part to securing any kind of account is the support for requirement **Multi-factor authentication** (SEC6). The FIDO2 system, with its related standard U2F, fulfills this requirement. German eID also fulfills this by law, at least for any authentication with high LoA. For FIM, there is research and approaches for MFA, but it is not widely supported. SSI has different methods of securing the user's wallet but no explicit support for MFA. The other systems do not consider MFA to be in scope.

For the application's security, it is often necessary to restrict access to certain parts or data, as described by requirement **Access controls** (SEC1). The system for FIDO2, Mozilla Persona, and the German eID do not handle authorization. Dynamic FIM, SSI, and LoRaWAN® contain aspects of managing authorization to resources within the respective standards.

Regardless of the use case for authentication or authorization, all credentials need to support being revoked if misuse is suspected. This is described in requirement **Credential revocation** (SEC2). The only concept handling revocation within its IAM framework is SSI. The other systems require the application to block credentials or accounts to revoke access.

Ideally, an IAM authentication or authorization process is designed according to requirement **Off-the-record (OTR)** (SEC12), and neither party gains information that could be proven or used with a third party. None of the systems provide this feature.

Especially for IoT but also transferable to any other IAM setting requirements **Secure setup** (SEC10) and **Secure de-provisioning** (SEC9) describe a secure setup and de-provisioning procedure. The secure setup requirement is met by all systems. Requirements SEC9 and **Identity de-provisioning** (CON1) share similarities where the latter already found all approaches besides SSI to be offloading de-provisioning to the applications. Requirement SEC9, however, focuses more on the user's side of de-provisioning. FIDO2 devices can be easily erased and re-initialized, fulfilling the requirement. For LoRaWAN®, erasure depends on the hardware but is also possible. The other systems do not handle erasing identity data explicitly, but for the German eID system, re-use with different identities is not supported and thus out of scope.

Again originating from IoT, requirements **Tamper-evident** (SEC7) and **Tamper-resistant** (SEC11) are concerned with detecting manipulation of IAM hardware. This is out of scope for all systems that do not use dedicated hardware. FIDO2 hardware tokens are designed and built to be hard to tamper with, comparable to the German eID card. For LoRaWAN®, no direct provisions for detecting or preventing tampering with end devices are specified. The key storage is dependent on the manufacturer's choice of IoT hardware.

Requirement **Delegation parameters** (SEC8) demands the ability to delegate permissions depending on user-defined parameters. None of the solutions support this, exception for SSI, where delegation is a research topic.

The requirement **Trust service providers** (SEC13) from eID requires some trust anchors that act as ultimately trustworthy authorities. For most IAM systems, this is out of scope and deferred to an X.509 certificate infrastructure, but the German eID, and to some extent, trust anchors in SSI do provide this requirement.

3.6.5 Data Protection

The basis for data protection is defined by requirements **Privacy by default** (DAT1), **Privacy by design** (DAT2), and **GDPR** (DAT3). FIDO2 and the German eID fulfill all of the requirements. So does LoRaWAN®, but its fulfillment of DAT3 cannot be determined. Mozilla Persona does partially fulfill those three requirements, limited by the fact that the identity is always bound to a personalized email address. FIM and SSI are designed to be privacy-friendly, but implementations can weaken the protections, and additional contracts are required to ensure GDPR compliance.

Requirement **Multiple identities** (DAT4) is fulfilled by Mozilla Persona and SSI. FIDO2 can partially fulfill it. On the one hand, every authentication connection utilizes a different key pair, which equates to a new identity. On the other hand, multiple devices can further separate identities. FIM, eID, and LoRaWAN® do not support multiple identities. In the eID case, this is by design. In the case of LoRaWAN® and FIM, it is usually unwanted or unnecessary, as the identities are only usable within a specific environment.

To protect the identity at rest requirement **Protected application storage** (DAT5) concerns how keys and other identity materials are stored. The FIDO2 and German eID approach fulfill this by providing hardware-based security for high LoAs. The other systems can support it depending on the implementation.

Requirement **Correlation resistance** (DAT6) should prevent an external observer from learning details about the entities involved in an exchange by analyzing metadata. Due to exchanges being done over HTTPS for the primary web-based applications, this is not an issue beyond what is observable anyway, i. e., IP addresses. With the involvement of DLT, preventing correlation is harder and dependent on implementation details in SSI because some metadata, i. e., who created or issued VCs, may be recorded in a public database. The situation is even worse for LoRaWAN®, where an observer can identify traffic patterns of individual devices.

A narrower view of requirement DAT6 is described by requirements **External tracking resistance** (DAT7) and **Internal tracking resistance** (DAT8). They specifically demand that an entity should not be able to be tracked geographically either by external or internal observers. Those two requirements are only applicable to SSI and LoRaWAN®, which both fulfill them partially, as it is not possible to track entities generally. However, it is possible to determine their presence locally.

3.6.6 Robustness

A central requirement for robustness is requirement **Reliability** (ROB1). As Mozilla Persona is no longer used or developed, it is not covered by this requirement. FIDO2, German eID, and LoRaWAN® can fulfill the requirement. The remaining dynamic FIM and SSI are not as reliable.

Part of the robustness category are the requirements **Accessibility** (ROB2), **Approachability** (ROB3), and **Usability** (ROB4). Mozilla Persona would fulfill those requirements, but due to it being deprecated is not scored. The FIM system fulfills ROB3, as the users do not need to set up anything. The others all require some explanation to be used. ROB2 is fulfilled by SSI, with the wallet app being a central component that can be utilized on mobile, PC, or in person. The other systems are all limited to certain devices or prerequisite hardware. Requirement ROB4 is fulfilled by FIDO2, as it is easy to set up and hard to use wrong. The other systems are all more difficult to set up and not as straightforward to use.

Tied in with usability is requirement **Platform independence** (ROB7), which is fulfilled by all systems due to the mostly web-based nature. Even LoRaWAN® can be utilized on different microprocessors. However, requirement **Communication protocol independence** (ROB9) is not as universally fulfilled. Usually, the protocols require HTTPS or, in the case of LoRaWAN® LoRa®. Switching transport protocols is possible but requires significant effort.

In some situations, verifying an entity's identity while offline is important, as described by requirement **Offline authNZ** (ROB5). None of the systems can offer this, as they are all primarily focused on communication through the Internet. A relaxed version of ROB5 is requirement **Standalone authNZ** (ROB12), which requires authenticating an entity within a network temporarily disconnected from the Internet. This can be archived with SSI but not with any of the other systems.

The ability to scale the IAM solution is demanded by requirement **Scalability** (ROB6). Most systems scale to some extent but are then limited by hardware costs for FIDO2 hardware, acceptance for Mozilla Persona and SSI, and federation size for FIM. The eID systems do scale worse than others, as high LoA identity cards are expensive and limited to certain services. LoRaWAN® can scale very efficiently to many thousands of devices with preloaded identity data by the manufacturer.

Connected to scalability is requirement **Resource efficiency** (ROB10), wich specifically handles resource efficiency on the entity's side. This requirement is only really applicable to systems that require much computation on the client-side. Because of the low-power IoT scenario, this affects LoRaWAN® and can be fulfilled partially.

Not only scaling the system but also scaling trust relationships without a trust authority is covered by requirement **Transitive trust** (ROB8). This only applies to systems that cover trust relationships, i. e., dynamic FIM and SSI. Both of those partially fulfill this requirement.

Any networked system can be attacked by a distributed denial of service (DDoS) attack. Requirement **DDoS protection** (ROB11) affects IAM-system integrated DDoS protection. This does not apply to systems that require interaction with another entity before authenticating, i. e., FIDO2, Mozilla Persona, FIM, and eID. The remaining SSI and LoRaWAN® do not handle DDoS protection within the protocols but require applications to handle it.

3.6.7 Combined Evaluation

The previous sections evaluated selected state-of-the-art against the requirements found from the scenarios of Section 2.5. Fulfillment of the requirements is heavily dependent on the scope of the considered state-of-the-art.

All systems fulfill the essential requirements **Identification** (SAT3), **Authentication** (SAT1), and **Identity provisioning** (SAT4). Requirements **Security by design** (SEC5), **Security by default** (SEC4), and **Scalability** (ROB6) are also fulfilled by all.

On the other end of the spectrum, requirements **Off-the-record (OTR)** (SEC12), **Offline authNZ** (ROB5), and **DDoS protection** (ROB11) are not fulfilled by any (applicable) state-of-the-art. Any other requirements are at least partly fulfilled by some state-of-the-art.

A table summarizing the individual state-of-the-art applicability of the requirements of Section 2.6 is provided in Table 3.2. The following chapter describes the newly developed concept aiming to fulfill most requirements to provide a comprehensive SSI framework.

Table 3.2: Applicability of the state-of-the-art towards the requirements of Chapter 2

Category	Requirement	FIDO 2	Mozilla Persona	Dynamic FIM	SSI/Hyperledger Indy	German eID	IoT/LoRaWAN®	Importance
Satisfaction	SAT1: Authentication	*	*	*	*	*	*	↑
	SAT2: Authorization	-	-	~	~	-	~	↑
	SAT3: Identification	*	*	*	*	*	*	↑
	SAT4: Identity provisioning	+	+	*	*	*	+	↑
	SAT5: Trust establishment			+	~			↑
	SAT6: Access delegation				~		~	↕
	SAT7: Automated integration/registration			~	~		-	↑
	SAT8: Identity data set matching			~	-	-		↕
	SAT9: Message delivery services			~	~	~		↓
Information	INF1: Credential establishment	*	+		+		~	↑
	INF2: Documentation	+	-	+	+	~	~	↕
	INF3: Digital identification						~	↑
	INF4: Physical identification							↑
	INF5: Product specification						~	↕
	INF6: Neighbour discovery							↑
	INF7: Service discovery							↓
	INF8: Agreement monitoring				~			↕
	INF9: Capability exchange			~	~			↕
	INF10: Identity data set			~	~	+		↑
	INF11: Level of assurance			+	~	+		↕
Consistency	CON1: Identity de-provisioning	-	-	-	+	-	-	↑
	CON2: Credential recovery	-	-	-	~	-	-	↕
	CON3: Digital twin				~			↕
	CON4: Content verification				~			↕
	CON5: Netsplit/Join				~			↕
	CON6: Once-only			~	~	~		↕
Security	SEC1: Access controls			~	~		~	↑
	SEC2: Credential revocation	-	-	-	~	-	-	↑
	SEC3: Mutual authentication	+			+	*	*	↑
	SEC4: Security by default	*	+	+	+	*	+	↑
	SEC5: Security by design	*	+	+	+	*	+	↑
	SEC6: Multi-factor authentication	*		~	-	+		↕

Continued on next page

Table 3.2: Applicability of the state-of-the-art towards the requirements of Chapter 2 (Continued)

	SEC7: Tamper-evident	+				+	-	↑
	SEC8: Delegation parameters				~			↕
	SEC9: Secure de-provisioning	+	-	-	-		~	↕
	SEC10: Secure setup	+	+	+	+	+	+	↕
	SEC11: Tamper-resistant	+				+	-	↕
	SEC12: Off-the-record (OTR)	-	-	-	-	-	-	↑
	SEC13: Trust service providers				~	+		↕
Data Protection	DAT1: Privacy by default	+	~	~	~	+	+	↑
	DAT2: Privacy by design	+	~	+	+	+	+	↑
	DAT3: GDPR	+	~	~	~	+		↕
	DAT4: Multiple identities	~	+	-	+	-	-	↓
	DAT5: Protected application storage	+	~	~	~	+	~	↑
	DAT6: Correlation resistance	*	+	+	~	+	-	↕
	DAT7: External tracking resistance				~		~	↕
	DAT8: Internal tracking resistance				~		~	↕
Robustness	ROB1: Reliability	+		~	~	+	+	↑
	ROB2: Accessibility	~		~	+	~	~	↕
	ROB3: Approachability	~		+	~	~	~	↕
	ROB4: Usability	+		~	~	~	~	↕
	ROB5: Offline authNZ	-	-	-	-	-	-	↑
	ROB6: Scalability	~	~	~	~	-	+	↑
	ROB7: Platform independence	+	+	+	+	+	+	↕
	ROB8: Transitive trust			~	~			↕
	ROB9: Communication protocol independence	~	~	~	~	~	~	↕
	ROB10: Resource efficiency						~	↕
	ROB11: DDoS protection				-		-	↕
	ROB12: Standalone authNZ	-	-	-	~	-	-	↕

Key: ↑ essential, ↕ important, ↓ optional, * completely fulfilled, + fulfilled, ~ partially fulfilled, - not fulfilled, not applicable

Chapter 4

Concept for a Comprehensive SSI and IAM Integration

Contents

This chapter describes the developed concept for a widely usable SSI architecture. It is built on top of existing state-of-the-art IAM and SSI technologies described in Chapter 3. To form this improved system, which fits the requirements from Chapter 2, the concept focuses on the following key areas of development:

- Wide applicability of SSI encompassing web applications, IoT, cloud computing, and eID. The strengths of an SSI system are only fully utilized if the system can be used in many applications.
- Interoperability with other SSI systems, existing IAM solutions, and a clear transition path. It is doubtful that there will be the one SSI system to rule all identities; instead, the different specializations need an interface to be used concurrently.

- Efficient and effective security management for distributed systems. Large, diverse interoperable distributed networks are notoriously hard to secure. For example, with SSI, there is a chance to use the lessons learned from the Internet, by now a massive distributed network.
- Data protection management for end users, relying parties, and attribute authorities. Like the system's security, data protection – both by design and by default – is essential to any IT system's design. Those features are expected by users and required by laws today.

Those developments offer benefits to all participants of an SSI system. Respectively, the most significant improvements to the current state-of-the-art are described for each stakeholders' category below. Further detailing of the contributions of this concept is provided in Section 4.2 and Figure 4.1.

- **Users** benefit from the described SSI concept by using a secure and reliable IAM system that integrates so well that they do not notice any major differences to existing IAM approaches. The **trust gateway (TGW)**, described in Section 4.3.7, makes it easier for users to transfer VCs from one system to another, preventing being locked into a specific IAM ecosystem.
- **Developers** must understand the systems they are developing to build and integrate new technology securely. As a result, the concept provides alternative methods for exchanging and presenting VCs, which do not involve zero-knowledge proofs but instead rely on classic X.509 certificate-like constructs, as described in Section 4.4.
- **Organizations**, including individual **service providers**, **identity providers**, and **attribute providers**, face a hurdle when adopting new technologies, as every technological switch is associated with risks of failure and service interruption, potentially leading to financial and reputation loss. To reduce those challenges, this concept shows a migration path from existing IAM systems. A new component, the **credential localization service (CLS)**, is described in Section 4.3.6. It aims to smoothen a migration and support long-term operations by improving the compatibility of different VC schemes.
- **Administrators** need to understand and effectively run the new components. The complete architecture can be easily adapted to all use cases from the scenarios, as shown in Section 4.5, and provides guidelines for the secure operation of the SSI system.

This chapter's further structure is divided as follows. First, Section 4.1 defines the scope of the concept. Then, Section 4.2 provides a high-level overview of the complete architecture, showing relationships, new and modified parts, and indicating what to expect in the following more detail-orientated sections. Afterward, Section 4.3 lists and characterizes all relevant SSI components affected by this concept, shows connections, and describes inter-component dependencies. The components are then combined to form the reference architecture in Section 4.4. Integration of the reference architecture with selected scenarios from Chapter 2 is shown in Section 4.5. Lastly, Section 4.6 compares the resulting concept to the requirements gathered from the scenarios of Section 2.

4.1 Scope

This chapter specifies a conceptual model and a reference architecture, including the system's components' characteristics. It is designed for a general-purpose SSI system for personal, organizational, and IoT identities. Because of the large

spectrum of possible applications, the core concept will leave some leeway for context-specific adaptations and implementations. Specific use cases are taken from the scenarios described in Chapter 2. For those, integration methods are shown to substantiate the concept of an SSI IAM system from scratch or a previously existing IAM system. Prototype implementations of the conceptual description of this chapter are described in Chapter 5.

4.2 High-Level View of the Architecture

The concept described in this chapter aims to fulfill the requirements gathered in Section 2.6 using existing FIM and SSI concepts where possible and adding new components where necessary. An overview of the architecture, including all components and connections, is depicted in Figure 4.1. The overview highlights new contributions with a blue star (✸) and modified components with a red octagon (⬢).

As shown in Figure 4.1, the concept is divided into four main areas, highlighted by different colors. Each area encompasses further components which fulfill specific tasks. The components are connected through relations that depict protocols necessary to exchange the information required.

- **Users** are the main subject of digital identities. They can be natural persons, organizations, or machines and possess attributes that describe them. Those attributes need to be expressed and exchanged digitally and securely. This is the job of VCs. The protocol for representing attributes as VCs connects the users' digital identities to their digital identity wallets.

- As part of the **SSI core components**, the wallet stores the digital representations of the entity's attributes as VCs. New contributions to the wallets are their cross-device design applicable to smartphones, browsers, servers, and IoT. To form a VC, the correctness of the claims of specific attribute values is asserted by an issuer. These VCs can be presented to an RP to verify the entity's attributes.

- The **supporting infrastructure** is added to augment the core components for better usability and to allow migration from other IAM systems. The central usability component is the agent, which acts on behalf of the user to pass messages from their wallet to the issuers or RPs and vice versa. The concept describes protocols for communication between agents and wallets, as well as agents and RPs or issuers. Trust gateways are a new type of agent with the task of translating between IAM systems. They act as issuers and cloud-based wallets. To do this, they import an entity's attributes from other IAM systems and provide an easy-to-use substitute for a dedicated wallet in another SSI system. If the supplied VCs are not in a format that can be used directly, the RP can use the newly described credential localization service to obtain automated rules for localization.

- As this system aims to be as decentralized as possible, the communication and organization between participants is performed using **distributed ledger technology**. This serves as a metadata store containing descriptions of attribute schemata and public identities. New contributions to the DLT component are the open design, allowing the use of different DLTs and exchanging the DLT by non-decentralized systems like a PKI. It is run by multiple operators and governed by a community, where new adaptions of security management are presented.

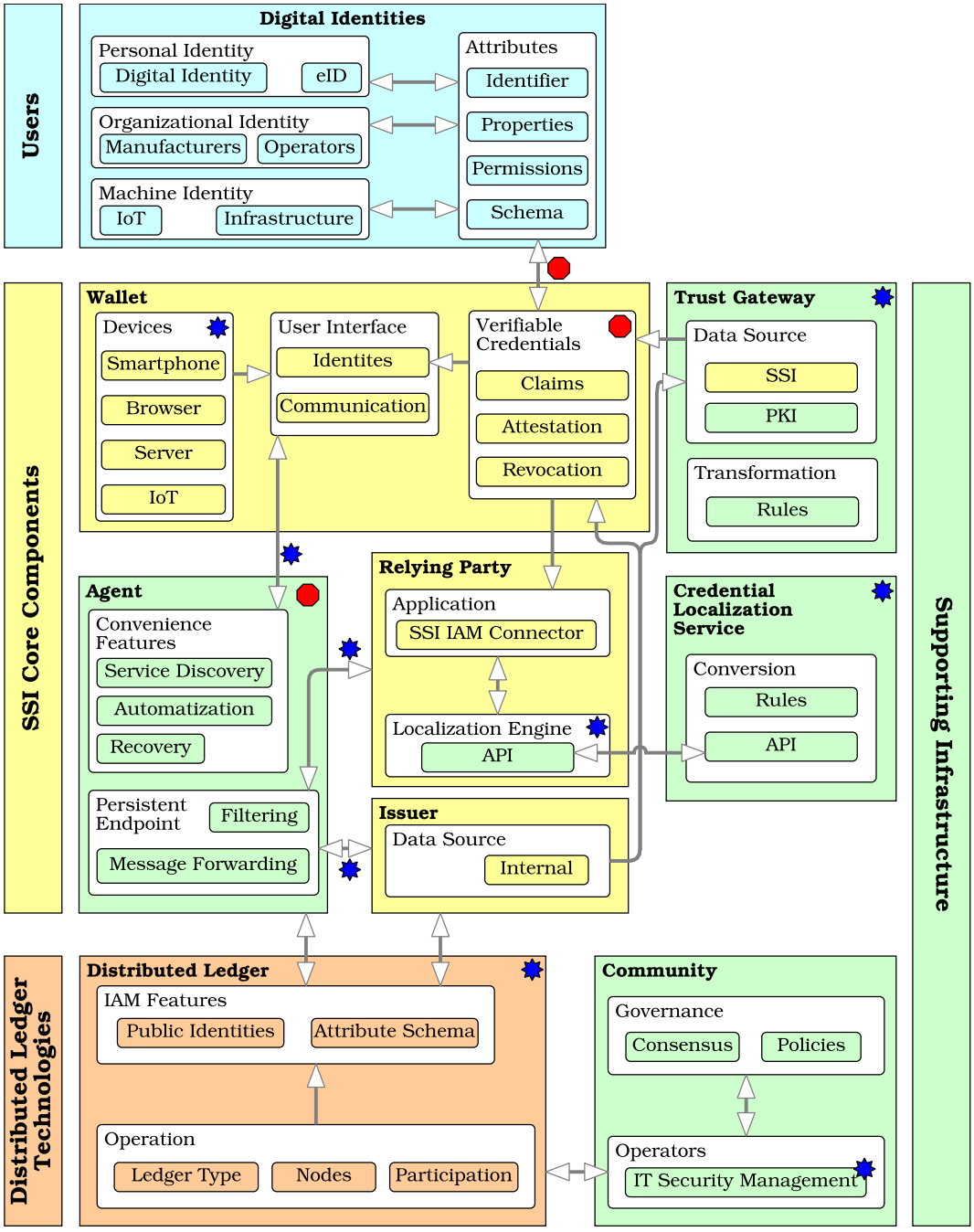

Figure 4.1: Overview of the resulting architecture, components, and key contributions

4.3 SSI Components

As described by [134], the essential components of SSI are identification, authentication, verifiable claims, and storage. The authors describe how each functional components has challenges that must be overcome in a decentralized architecture. In this concept, the components are based on the elements of the resulting system, which the functional components need to be part of. Figure 4.2 shows those components and their primary interactions.

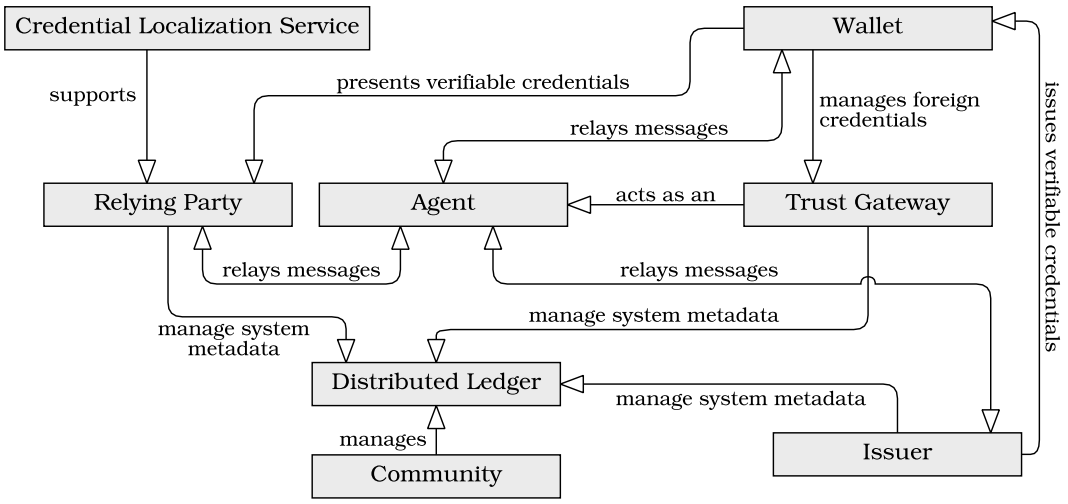

Figure 4.2: Component overview

A systematic approach analyzes each component's properties individually, only referencing other components where interfaces and communication relations require doing so. For each component described in this section, the following four models will be provided, according to the definition by [81] and the OSI management architecture [92, 97]. This established and structured approach helps cover the most relevant aspects of a management system.

- **Information Model:** Describes *what* is managed by characterizing the managed object. This is usually done by creating an object-oriented overview of components and their relations.
- **Organizational Model:** Describes *who* is part of the cooperation scope. This is done by determining the relevant actors, specifying their domains, and determining their principles of cooperation through policies. Usually, systems are assigned the roles of manager or agent for this.
- **Communication Model:** Describes *how* management information and instructions are exchanged. This is usually done by defining a management information protocol.
- **Functional Model:** Describes *how* management takes place. This is usually done by declaring the required functionality for the general management function areas: fault, configuration, accounting, performance, and security management (FCAPS). The security aspect is not described in this model. Instead, it is discussed in its own security management section.

In addition to those four models, each component is viewed from a **security** and **data protection management** point of view. The security management part is based on providing a balanced evaluation of the protections described by the CIA triad. It provides methods based on ISO/IEC 27001 to manage and monitor a

component's security. Data protection management is based on the concepts and requirements stipulated by the GDPR. An overview of the different aspects and components is shown in Figure 4.3.

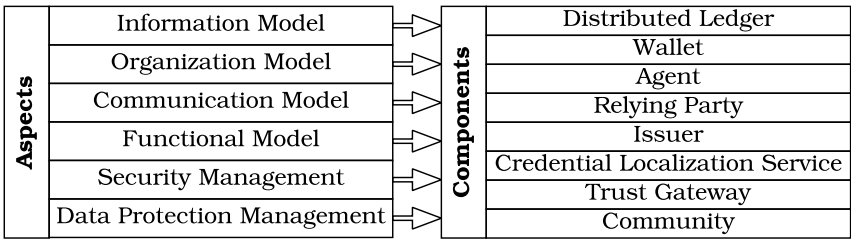

Figure 4.3: Aspects that are considered for each of the components

At the end of this section, in Section 4.3.9, the individual components are put into relation with each other. These relations are then used by the concept described in Section 4.4 to build the complete concept.

4.3.1 Distributed Ledger

At the core of the development of SSI is the use of DLT to share the various amounts of accruing data. The design of the distributed ledger decides many of the resulting system's properties, including security, performance, and scalability. Distributed ledgers are differentiated from federated and distributed databases by having a consistent global state replicated across all participating nodes. Multiple people or organizations run those nodes and keep track of transactions and a consistent state without requiring trusting in the other nodes' correct behavior.

These properties make distributed ledgers ideal for bringing together large groups of like-minded organizations. Compared to regular relational database management systems (RDBMSs), however, the structure of the state-based database is considerably different. The complexity of possible transactions is somewhat limited, and there is no universal query language, like SQL. Optimizing DLT and unifying access is an ongoing effort, which initial adopters will have to take a pass on.

While there is rapid development in the DLT sector, it is not easy or obvious to choose the right distributed ledger for a particular SSI project or correctly identify the need for DLT in the first place. The following sections showcase essential aspects of the distributed ledger when used as an SSI data store. This section also shows how a PKI system can be an alternative to DLTs.

4.3.1.1 Information Model

The information model of the DLT is split into two parts: the distributed ledger itself and the data stored on the ledger. Figure 4.4 shows this model and distinguishes between the two parts by showing distributed ledger managed objects with a yellow background color and the managed stored data with a turquoise background. The following description first focuses on the distributed ledger part and then describes the ledger's contents.

A distributed ledger is formed by the *nodes* operating according to its rules and its initial state defined by the *genesis* file. The genesis file usually contains metadata necessary to operate the distributed ledger, such as the public keys and addresses of the initial participants. The rules are specified by the implementation of a specific ledger and its parameters. The consensus algorithm determines how new

blocks are added to the ledger by describing how consensus about the ledger's state is formed between all nodes. Different consensus mechanisms are possible, and each has advantages and drawbacks.

Any node can enter new transactions into the pool, thus extending the pool of pending transactions. In order to add new information to the ledger those pending transactions must be picked up by a *validator*. Those take part in continuing the ledger and form new blocks by selecting transactions out of the pool of pending transactions. The validators can add the new blocks to the distributed ledger, available to any nodes with read access. The *observer* type of node only observes the process of adding new blocks, noticing any misbehavior or failures. It holds a copy of the distributed ledger and can also serve as a standby node to take over if validators fail or become disconnected.

As transactions can only add new data, the stored data is a sequence of additions. Subsequently, additions may update existing data or mark transactions as deleted without removing previously published data. To optimize the local storage for more space and read efficiency, nodes may create snapshots of the ledger's state at certain points in time and remove any previous transactions. This prevents having to replay all transactions from the start when searching for a value or determining the state at a specific point in time. Each node keeps as much of the ledger stored as necessary for its operation.

Transactions for SSI ledgers following this concept come in three flavors:

- Public identities use the `IdRegistration` to publish a part of their public identity to the ledger. This allows others to identify those entities and access the metadata stored in the metadata document, stored in an off-chain location and only linked to by the transaction.
- The `SchemaDefinition` can be used to define new attributes and their composition. Schemata defined this way can be re-used by different parties.
- The `PrevSigBundle` publication publicizes the no longer valid VCs' public and private keys. This serves the purpose of devaluing old VCs and achieving plausible deniability.

As storage space on the distributed ledger can be quite limited, larger files (e. g., large text files, pictures, or even videos) need to be stored elsewhere. This is called off-chain storage, which is then referenced within the transaction. The contents of the off-chain storage may change or disappear without an additional transaction being issued. Thus the state of the content must be saved on the ledger. This is done by including a hash of the referenced data in the transaction. Keeping the permanently stored transaction's size low also minimizes the risk of accidental or deliberate inclusion of illegal, unwanted, or PII content.

4.3.1.2 Organization Model

Within the distributed ledger, several nodes work together to maintain a consistent state by applying ordered transactions. Those nodes appear in their general roles as observers or verifiers. In the organizational model, those nodes are not shown as individual actors, as they cannot take action other than those specified by the DLT's protocol. The observers only check the verifiers' actions to spot and alert others if the rules of building transactions on the DLT are violated. They are easier to set up than verifiers, which do the actual task of ordering and inserting transactions onto the DLT.

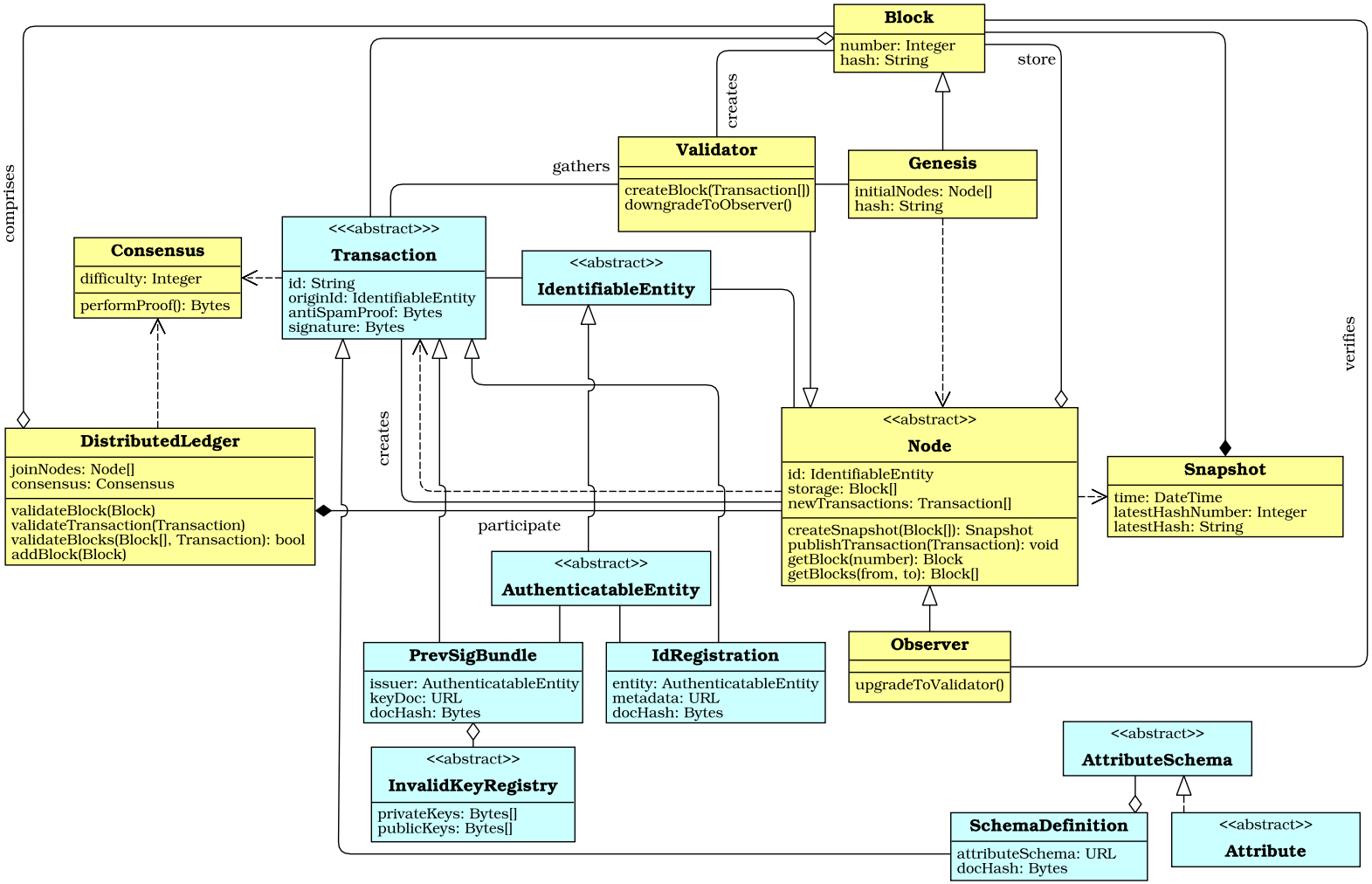

Figure 4.4: Distributed ledger information model

For any meaningful benefit of adding the complexity of a distributed ledger, the nodes must be controlled by multiple individuals or independent organizations. Otherwise, a regular (distributed) database can achieve the same distributed data storage. A regular database also performs better, is easier to maintain, and can recovered more easily from errors.

As the central organizational medium of data exchange in an SSI system, the distributed ledger is connected to most other organizational units, as shown in Figure 4.5. It primarily acts as a trust anchor that the other stakeholders' actors use to retrieve and verify metadata required for trust management.

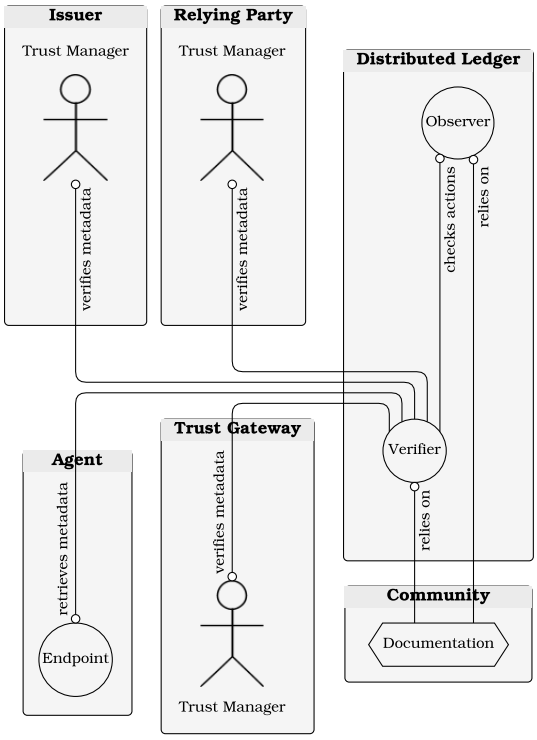

Figure 4.5: Organizational relations of the distributed ledger

For SSI to work, it is essential that relying parties can trust the credentials issued by the issuers. To manage these trust relationships, the issuer, the relying party, and the TGW must have a trust manager. The trust manager utilizes the DLT in the trust establishment process to form connections to the other issuers, relying parties, or TGWs. This trust establishment can be supported with DLT by providing trust anchors, i. e., public key certificates of certain entities indexed by their unique identifiers. These trust anchors can be augmented with further metadata from a metadata document to provide a transparent method of finding the most up-to-date information about a public entity. This method of pinning trust to certain identities does not scale well, as keeping track of the number of identities recognized as trustworthy can get confusing quickly. As a result, this method can be used by governments and large corporations, but smaller businesses or individual entities require a more decentralized approach.

One solution might be the implementation of a trust hierarchy, as it is already used for X.509 certificates on the Internet for HTTPS. The hierarchical approach, however, inherently creates a power discrepancy between entities at the top and those further down in the tree, a solution incompatible with self-sovereignty.

The agent also connects to the distributed ledger to vet any requests and responses from and to the user's wallet. This vetting process can be used to reduce the likelihood of impersonations or other phishing attacks.

4.3.1.3 Communication Model

A distributed ledger provides two distinct interfaces for the communication of management information. The first is for inter-ledger communication and is used to synchronize the ledger state with other nodes of the P2P network. The second interface provides an API to access a node in order to submit new transactions or read the ledger's state.

P2P protocol When adding a new node to a distributed ledger, the process must differentiate between the different ledger types. Joining a ledger with read access only (i. e., for observers) can be done at public ledgers without any prior registration. Private ledgers, however, will require registration and authentication to access. Both paths are shown in Figure 4.6.

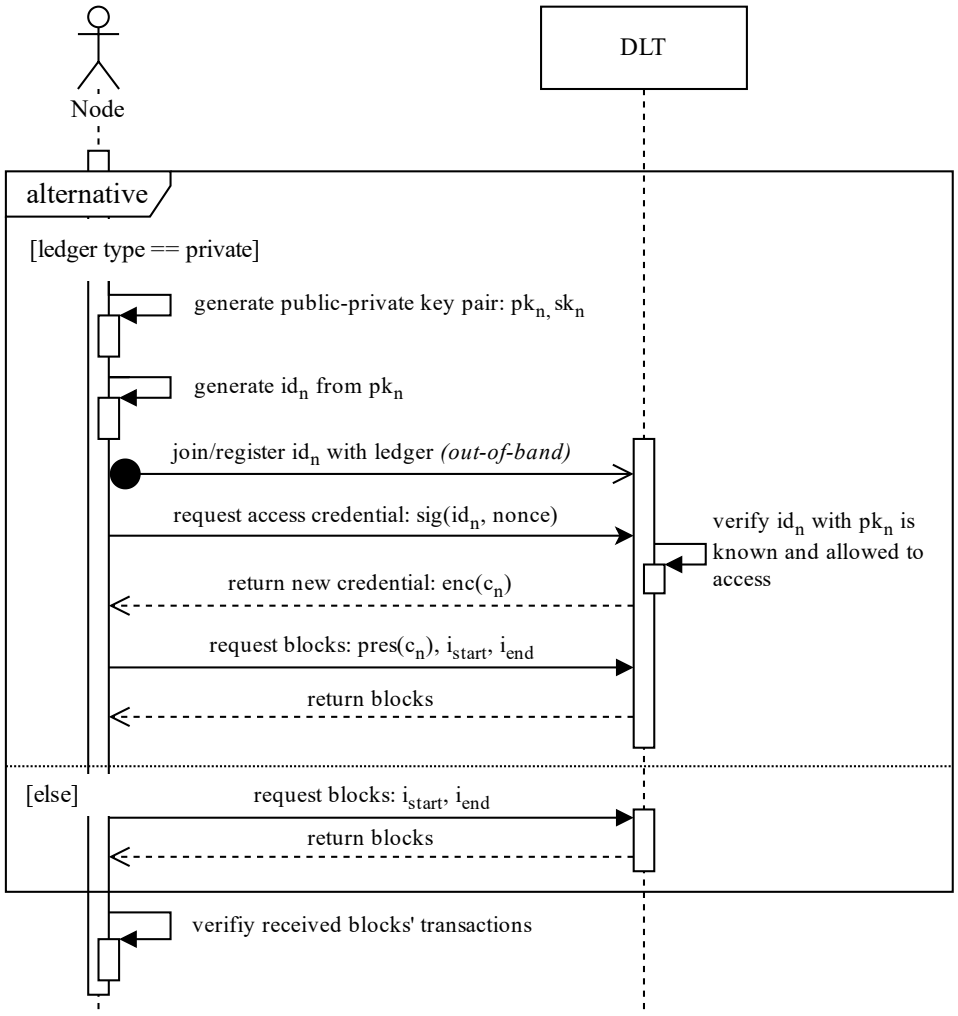

Figure 4.6: Join-sequence of a new node to an existing distributed ledger

The first step to participation in a private distributed ledger is to generate a public-private key pair, which generates a unique ID to identify the new participant. This identifier has to be registered with the operators of the distributed ledger through a management system that is ledger-specific. A distributed ledger focused on IAM can use its own IDM system to control access to the ledger using a VC to represent access permissions. Once this access credential is obtained, a verifiable presentation of this credential can be presented to other ledger participants to request any number of blocks. In a public ledger, this workflow is considerably easier, as every participant can request any number of blocks from any participant already synchronized with the ledger to retrieve the ledger's information. After receiving blocks from the distributed ledger, the receiving nodes must check their correctness. This is done by tracing the transactions from the last verified snapshot or the genesis file.

A new block of the distributed ledger is created according to the distributed ledger's consensus protocol. It is run by a node grouping several transactions, executing the transactions' actions, and storing the process resulting in a new block. The newly formed block is then distributed to all other nodes within the network via a P2P protocol. Similar to the transactions, each node has to check the correctness of the application of the new block's transactions to the ledger to prevent a rogue node from violating the processing rules of the ledger.

A permissioned ledger requires registration for creating new blocks; a permissionless ledger does not. Irrespective of the registration a public-private key pair is always required for submitting new blocks to a ledger, as submissions must be signed by the participant creating the new block. For SSI distributed ledgers, the permission to submit new blocks can be represented by a VC. The adapted flow is depicted in Figure 4.7.

API protocol The ledger's API is, for example, used by the agents to query the DLT's current state. This is described in more detail in Section 4.3.3.3. Additional use cases to access the ledger's API include submitting transactions by entities that are not themselves nodes. Those transactions can register public identities, create VC schemata, or publish discarded private keys.

Submitting a new transaction to the pool of pending transactions can be done by any entity allowed to access the corresponding API on a node participating in the distributed ledger. The node receiving a new transaction must check the transaction's validity against the set of rules set forth by the distributed ledger. If the transaction checks out, it is distributed via the P2P protocol, described in the previous Section 4.3.1.3, to all other nodes of the network. At each step in the distribution, each node has to recheck the validity to prevent a node from submitting and distributing a bogus transaction.

4.3.1.4 Functional Model

As a special type of database, the distributed ledger supports just two operations: adding and retrieving values. Modifications are implicitly done by adding the updated value of an object at a later time. All additions are recorded, and the present state of the ledger can always be reconstructed by replaying all additions from the initial state. Reading from the distributed ledger yields the values stored at the requested time.

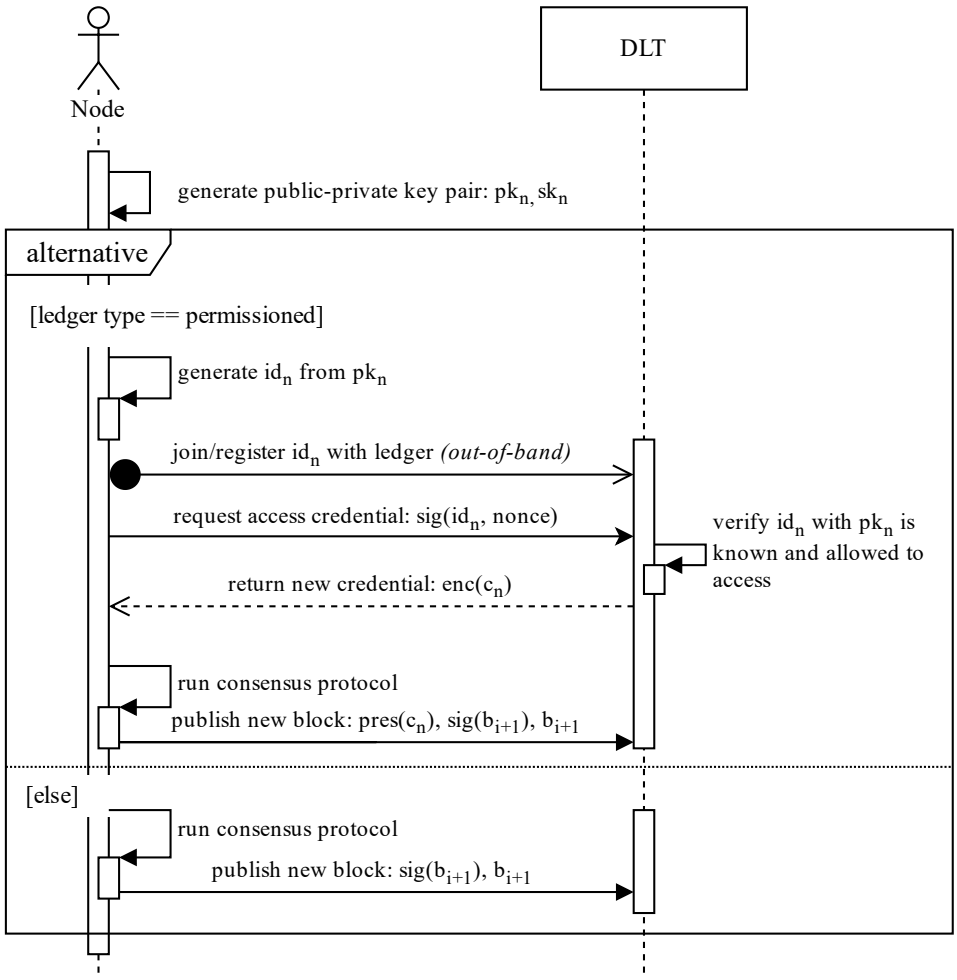

Figure 4.7: Submission sequence of a new node to an existing distributed ledger

More advanced distributed ledgers may also support the execution of code stored on the ledger as a result of a previous addition. These programs are called *Smart Contracts* or *Chaincode*, depending on the ledger used, and their execution can result in further additions to the ledger. This introduces a method to automate operations on the distributed ledger's state.

Faults in a distributed ledger can occur at multiple locations. One of the most significant risks to a distributed ledger is the malicious or accidental submission of a transaction that breaks the – up-until-then – perceived rules of the ledger. Similarly, a transaction might be issued by a compromised entity, resulting in an undesirable ledger state, but without breaking fundamental assertions of the ledger. There are numerous examples of such events happening in all types of – but primarily public/permission-less – DLTs [44, 94]. Reactions to those events have differed strongly: from not recovering at all to reverting transactions and starting over from a previous point in time. A recovery or rollback strategy must be defined for an SSI ledger to react to such events.

The DLT's configuration consists of its genesis file, describing the initial state, and the rules describing valid transactions (i. e., modifications of that initial state). All participants must prepare and agreed upon any changes to those rules. Otherwise, there is the risk of splitting the network into two groups: the one accepting transactions according to the updated rules and those that do not accept the new transactions.

As the backbone of the SSI system, the DLT can also be used to finance the operation. With classic blockchains (i. e., Bitcoin or Ethereum), each transaction requires fees paid to the verifying nodes. For permissioned DLTs, this is not necessarily the case, and membership fees can offset operating costs. The goal of an SSI system is that basic usage (owning, managing, and presenting credentials) should be free. This can be achieved by government subsidies or by charging larger issuers. Those issuers may pass parts of those costs on to the identities receiving a credential.

A DLT's performance is usually measured in transactions per second. A high transactional load is not expected for an SSI-focused DLT following this concept, as transactions are only required to add or update public identities. Most actions, such as issuing and presenting credentials, can – and for privacy reasons should – be done completely off-ledger.

4.3.1.5 Security Management

On a basic level, centralization of power is a key risk for DLT, especially if the consensus protocol allows public participation and the participating parties can collude to gain a majority capable of rewriting parts of the ledger. This is primarily a concern for cryptocurrencies, as such rewrites can allow an attacker to spend currency tokens multiple times. Depending on the value of those tokens, there is a strong financial incentive to try to achieve this. Rewriting history on an SSI-focused distributed ledger is less incentivized, as there is no direct financial gain. However, a successful attack might allow a denial of service (DoS) attack.

Of the CIA triad, integrity is the most critical aspect of running a distributed ledger. The local state of the nodes' databases should always be in sync with the state of the other nodes. To ensure this is the case, proper implementation with rigorous testing is required. Additionally, a process for regular and timely software updates is necessary to keep up-to-date, especially with new and fast-developing DLT.

Connection losses or general availability problems can also compromise the system's effectiveness, as catching up with the most current state of the network can take considerable time. During this time, the node cannot query the current state

or validate new transactions. Catching up to the network is usually a built-in procedure to the distributed ledger's software for nodes, but the sophistication of approaches varies. If there is no built-in quick way to catch up, e. g., using snapshots, additional backup tools to speed up restoring to a relatively recent state are essential. On top of that, in critical systems using multiple nodes on separated hardware within an organization can provide fast fail-over and increase availability.

Confidentiality is mainly important for protecting private keys, which are used to sign messages to other nodes on the network. Accidental loss of the private keys prevents a node from operating. If an attacker steals the key, they can easily impersonate the node. To reduce the attack surface and potential for errors, a node should be run on dedicated hardware with tight access controls, constant system updates, and continuous monitoring.

For permissioned blockchains and especially Hyperledger Fabric, the paper [151] evaluates published attacks and determines threat indicators. Of those, attacks related to smart contracts are ignored, as they are not relevant for an SSI-based DLT, which does not need smart contracts. Additionally, threat indicators specific to the Hyperledger Fabric DLT are omitted, as they do not fit the universal applicability targeted here. After grouping the indicators into categories, the resulting threat indicators are as follows. Any indicator described in [151] is marked with •.

- Transaction-based:
  - **Transaction throughput**• can be used to determine irregularities in the number of processed transactions. Sudden drops might indicate a fault at multiple verifiers.
  - **Transaction latency**• is related to throughput, as the latency is the time between creating a transaction and including it in a block. It will rise if transaction processing slows down.
  - The number of **outstanding transactions**• is another metric that can be used, similar to the transaction throughput and latency. It measures the number of transactions waiting in the pool of to-be-processed transactions.
  - Similarly, the age of **outstanding transactions**• can show whether older transactions are processed.
  - Client applications can track their **outgoing transactions**• to determine if they are processed as expected.
- Network-based:
  - **Incoming network messages**• can be used to dissect the message types and frequency of received messages. A shift in distribution or absolute numbers can indicate network operation changes.
  - The number of **connected peers**• can indicate the healthiness of the network and show connectivity problems.
- Block-based:
  - **Discarded blocks**•, meaning blocks received but which could not be validated, can indicate a misconfiguration or active attack.
  - Observation and checking of the **latest block hashes**• can allow an entity to ensure the correct continuance of the ledger.
- Depending on the used consensus protocol, some of the following consensus-based metrics could be used:

  - Monitoring **consensus failure rates**, i. e., how often blocks with different transactions are issued by separate nodes, can hint at ongoing attacks. In some DLTs' consensus mechanisms, those failures are to be expected, but their occurrence should be randomly distributed. In others, those failures should never occur.
  - If consensus requires the selection of a leading node, the **leader election frequency**• can show potential attacks.
- Others:
  - **Configuration and implementation changes** must be closely monitored to detect any new vulnerabilities they might introduce. The DLT for SSI should be feature complete and not require functional updates.
  - **Transactor identities**• can be monitored to detect the absence of a usually reliable node or the appearance of many new nodes.
  - **Threat intelligence on vulnerabilities**• is essential for all components: the hardware, the operating system, and the software and its dependencies.

Without a central operator responsible for running the distributed ledger, each operator must take responsibility for providing the best security possible.

4.3.1.6 Data Protection Management

The most significant risk concerning data protection with running a distributed ledger arises from the potential of accidental or negligent inclusion of private personal information in transactions to the distributed ledger. As any data stored on the distributed ledger will always stay there and should be treated as publicly accessible, particular care has to be taken to prevent the inclusion of data not meant for publication. Filtering transactions based on potential data protection violations is complicated, as no single node operator can effectively prevent the inclusion of transactions. Instead, a network-wide effort must be made to define clear rules adhered to by those submitting transactions and enforced by those validating the transactions.

If illegal data, which has been found, for example, on the Bitcoin blockchain [128], or if PII, which has to be removable due to regulations such as GDPR, does enter the blockchain, each block containing those is a single point of failure breaking the chaining aspect of blockchains. To limit the extent of data that can be written to the distributed ledger, the types of transactions and their contents are strictly limited in this SSI concept. As [129] suggests, transactions with less than 100 Bytes are mostly safe from containing unwanted content, and transactions with up to 1 KiB should be relatively safe and not contain illegal content.

4.3.1.7 Conclusion

Distributed ledgers are used in this concept for SSI because they ideally fulfill the requirements for decentralized cooperation between many organizations. In particular, they provide a system where investments are protected against the decisions of other participants. For example, if one organization leaves the system, the remaining parties are not adversely affected and do not have to change any systems. Compared to regular databases, the limited set of operations, low throughput of transactions, strong integrity protection, and easy auditability of actions are further reasons why distributed ledgers can be helpful in this scenario. Adding new members to the network is also straightforward.

The choice of a specific DLT is more difficult because of the large number of competing DLTs, fast development, changing focus of projects, and unclear long-term strategy or commitment within projects. However, there are a few more mature exceptions providing a good starting point.

A taxonomy for DLTs has been developed by [13], but it is heavily focused on cryptocurrencies, which are usually unsuitable for SSI applications. Nevertheless, many dimensions can be transferred or adapted.

Tokens, which represent a value and are exchanged in predominantly cryptocurrency-focused distributed ledgers, are less of a concern in SSI distributed ledgers. The value for an SSI-focused distributed ledger arises not through the exchange of tokens. However, a unique feature is the ability to cooperatively write a large metadata file, which is necessary to verify and understand the off-chain identity exchanges. Figure 4.8 shows the distinguishing properties of different DLTs, which can be used to compare those and additional ledgers. This taxonomy is simplified to only include the most relevant aspects. A more detailed taxonomy is shown in Figure 1.2.

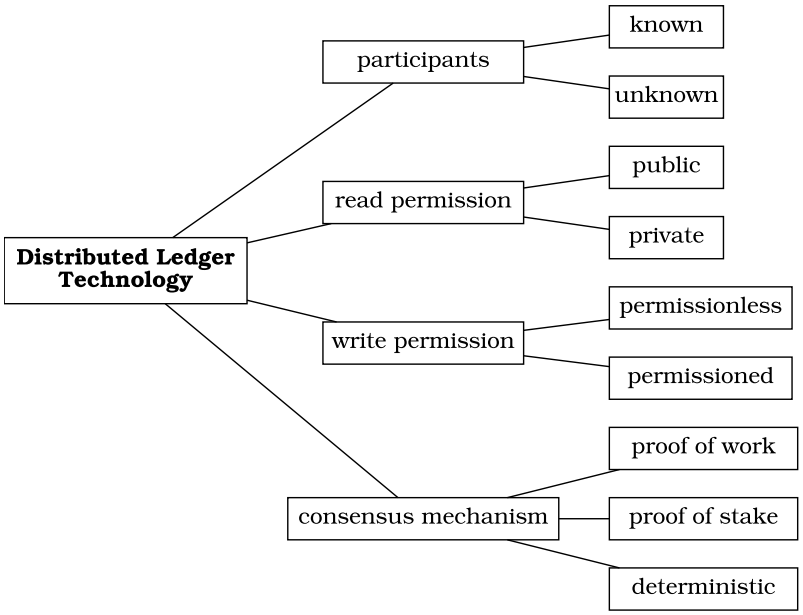

Figure 4.8: Condensed taxonomy of distributed ledgers

4.3.2 Wallet

The wallet is the application used to store and manage private keys and credentials. The name wallet is chosen to reference physical everyday life, where ID cards (the counterparts to digital credentials) are kept in physical wallets. Depending on the use case, the wallet can be implemented in various ways. For example, personal identities can be managed through a wallet within a smartphone app, IoT devices can use a trusted platform module (TPM), or a hosted service can be used to manage the identity of an organization.

4.3.2.1 Information Model

In general, a wallet can be used to manage multiple identities and their associated personas. In most use cases, however, the wallet is almost equivalent to a single identity. To manage the identities, the wallet stores all the necessary data and

keys. As part of an identity, a persona specifies a part of an identity, which is used with specific other entities or in specific settings. This section describes the structure of a wallet and details the credentials' structure.

A wallet application's tasks are primarily:

- To store and manage the credentials associated with the identities and personas managed by the wallet's owner.
- To store and manage private keys used to prove control of an identity through authentication or sign messages.
- To provide a (user-)interface for all management operations of credentials and keys.
- To show the user additional information about identities and personas (e. g., a log of past authentications and credential presentations).

As a result, the components of a wallet are the *credential store*, the *key store*, the *persona store*, and a *user* and *communication interface*. Those components are shown in Figure 4.9. The communication interface is described in Section 4.3.2.3, while the user interface is not specified further in the concept. The necessary features of the user interface depend highly on the intended use and must be determined according to the target audience.

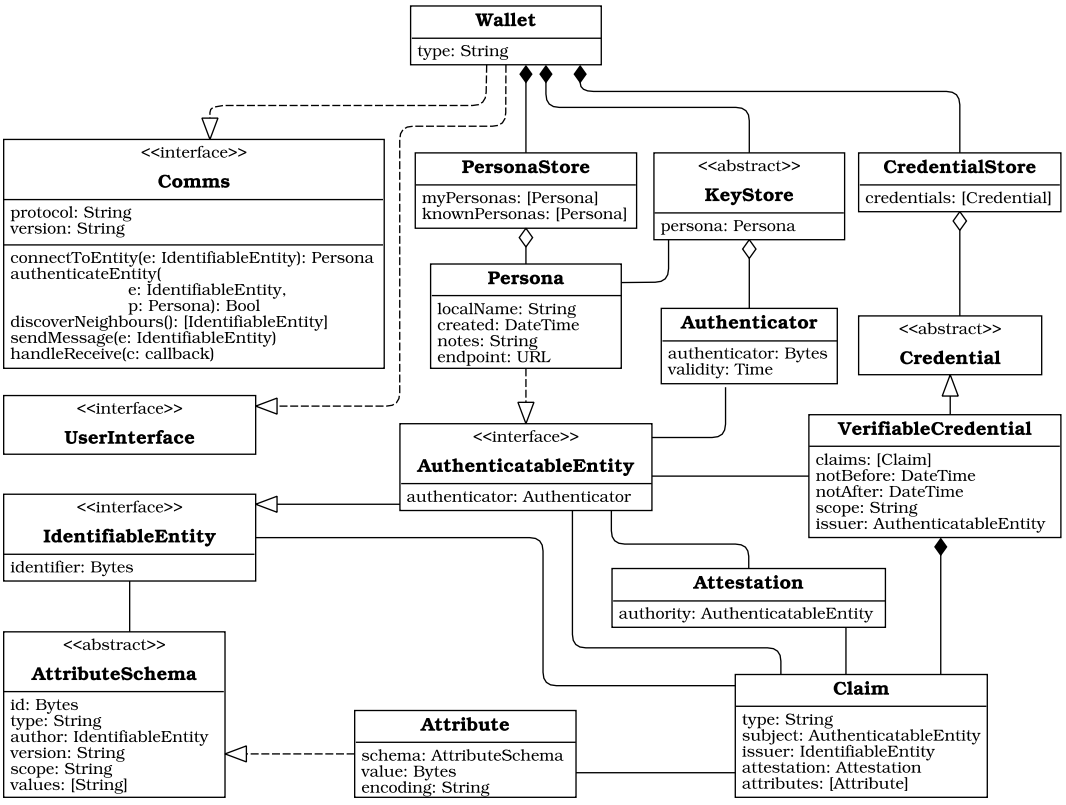

Figure 4.9: Wallet information model

The information model distinguishes between personas, i. e., entities known to the wallet's owner, *authenticatable entities*, and *identifiable entities*. The personas are managed in the wallet's persona store. Authenticatable and identifiable entities describe references to other entities, which can be either identified and authenticated or just identified.

To be able to authenticate to other entities, the wallet needs to store the necessary authentication information. This information is (besides the potential PII in the VCs) extremely sensitive and needs to be protected from unauthorized access and accidental loss. The wallet's key store is responsible for providing this kind of storage.

Based on the design described by [134], the claims about an entity are split into the claim and an attestation. The claim is made about one or more attributes, which are defined by their respective attribute schema. Together they are aggregated in a verifiable credential alongside metadata inspired by X.509 certificate metadata. The resulting abstract credential is stored in the wallet's credential store.

The information model shown in Figure 4.9 and described above only comprises the bare minimum components. Additional functionality like a notification system, a messaging system, or import, export, and backup functions are omitted and must be designed and implemented where required.

4.3.2.2 Organizational Model

For the SSI wallet, only one actor exists, the user. The wallet is a personal storage system for identities and associated data held or managed by the user. Having an identity in one's wallet does not necessitate that the identity belongs to or describes the wallet's owner. Instead, one might have multiple personal identities, entrusted organizational identities, and owned devices' identities in a single wallet. This aligns with SSI's goal of keeping the user in ultimate control of individual identities.

The wallet is provided by its manufacturer and is usually run by the user on their own devices. To build the wallet application, the manufacturer requires access to the community's documentation of the appropriate standards and interfaces. When using the wallet to exchange identity data, the user can choose to use the wallet's endpoint to connect to issuers and relying parties for VC exchanges or use the wallet's ability to connect to an agent's endpoint. The option of using an agent's endpoint enables access to the DLT to verify connections with the data stored there and to use services like the TGW. The agent also provides a service desk as an organizational role. This service desk routes the user's inquiries to the appropriate place, as described in Section 4.3.3.2. Those connections between the wallet and the other components are shown in Figure 4.10.

There is no organizational unit for the identity's subject itself. The subject's relation to the identity holder and their wallet within the SSI system can vary depending on the use case.

4.3.2.3 Communication Model

The wallet component needs to exchange identity management information with other wallets and agents and retrieve information from the distributed ledger. As a result, three communication protocols concern the wallet directly:

- The wallet to agent (W2A) protocol is required to connect wallets through the Internet or any other extensive network where individual participants cannot be addressed consistently. A more detailed description of this protocol is provided in the agent's description in Section 4.3.3.
- Wallet to wallet (W2W) communication is done to synchronize user wallets or to prove identities to other wallets within a LAN. Details of this protocol are described in the following part of this section.

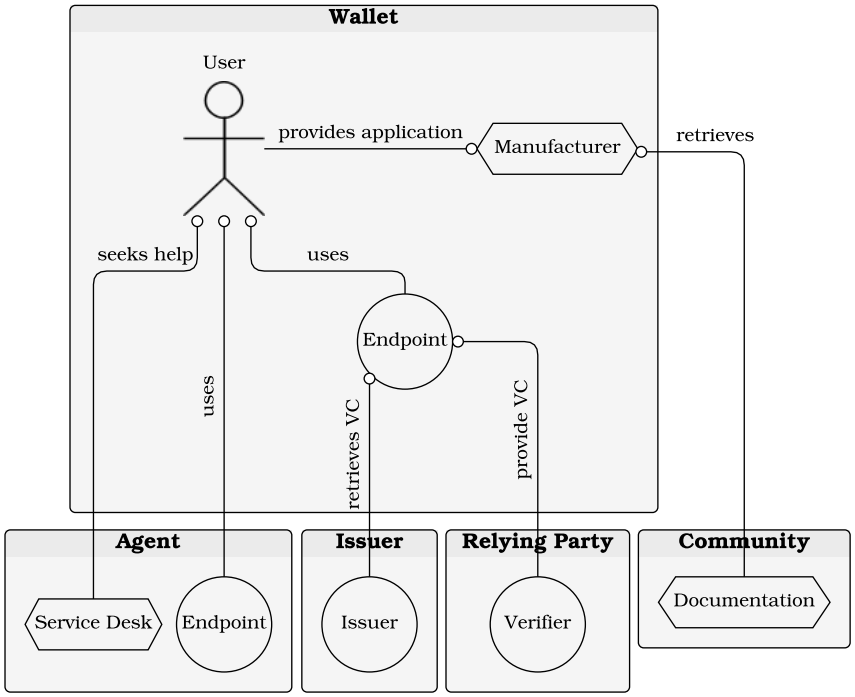

Figure 4.10: Organizational relations of the wallet domain

In order to facilitate communication between wallets within a LAN or even an ad-hoc wireless personal area network (WPAN), the wallet needs to support three phases of communication in the W2W protocol:

1. **Neighbour discovery** of other wallets via a suitable communication standard, e. g., Wi-Fi, Bluetooth, or NFC, each of which has its own methods for peer discovery on ISO/OSI layers one and two. For TCP/IP-based communication protocols, a standard port for the wallet's API must be specified. After the discovery of the device, this port can be used to initiate a connection. Non-TCP/IP-based systems must adapt a similar approach, e. g., through magic headers in the exchanged packets.

2. **Mutual authentication** of the discovered wallets to prevent impersonation or MITM attacks. According to Burrows–Abadi–Needham logic (BAN logic) [27], the steps to ensure a secure connection must contain verification of the message's origin, freshness, and the origin's trustworthiness. Those properties can be achieved with certificate-based [140] and password-based [153] authentication schemes. In the case of SSI, the certificate-based approach is better suited, as each identity is represented by a public/private key pair.

3. **Information exchange** using the previously determined common interfaces at the user's request and the other user's clearance. In this step, the validity of credentials can be established by verifying the contained claims and attestations.

Outside of a LAN environment, additional communication functions are necessary. Those functions can be part of the W2A protocol:

- **Identity discovery** is necessary if the wallet needs to contact an identity that it cannot directly reach. To do so, it needs a lookup protocol to find the appropriate endpoint to contact the other identity. As part of standardization efforts, the DID resolution protocol [159] aims to achieve just that.

- **Message relaying** is a primary use case for an agent, as it provides a permanently connected endpoint, whereas the wallet may be offline for extended periods. The agent stores messages intended for the wallet and relays them either through a push or pull model to the wallet.

4.3.2.4 Functional Model

A wallet's functional domains are primarily the management of credentials, keys for any owned identities, and communication with other wallets and agents through a unified user interface, as depicted in Figure 4.11.

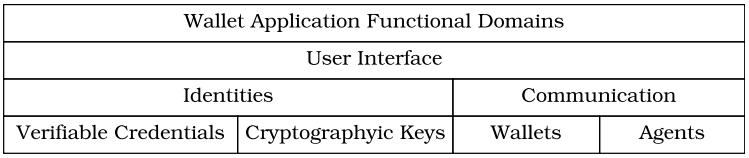

Figure 4.11: The breakdown of the main functional domains of SSI wallets

The user interface of a wallet application must provide methods to manage identities. This includes creating, updating, and deleting personas and their corresponding keys and credentials. To initiate exchanges of identity information or to synchronize wallets, the user interface must allow the user to contact other wallets or agents and react to received requests from other wallets or agents. If the wallet supports some form of automated exchange of credentials or keys, the conditions and rules for those exchanges must also be configurable from within the wallet.

One challenge with SSI systems is that there will be no single definitive and universal system. Instead, the wallet must adapt to different configurations with system-specific communication protocols for other wallets and agents. Those configurations should be provided by the wallet's supplier or be defined by the user. The performance of the wallet can be measured by the number of supported systems, its security, and its usability.

The prevention of data loss through a robust backup system of both cryptographic keys and credentials is extremely important. This system must retain security guarantees while allowing the wallet's device to break and be restored. The only parts of the identity that cannot be recovered by needing to re-issue VCs are the private keys. Those must be backed up securely or held in multiple wallets. Generating the keys by utilizing a passphrase, as shown in [110], is also possible. For some identities, even peer-based recovery systems [12] may be helpful. In those systems, several trusted peers can be used to recover a lost identity.

4.3.2.5 Security Management

From the security management's perspective, the most important aspect of a wallet is to keep the private keys secure and confidential. Depending on the operating system the wallet is running on, different approaches may exist to implement adequate protections, e. g., using secure enclaves, biometrics, physically unclonable functions (PUFs), or basic encryption. As the wallet is also responsible for generating public/private key pairs, the random number generators and algorithms used for this step need to be sufficiently strong, as well.

At a minimum, the wallet should be protected with a password or passphrase that is required to access its contents and operate with the contained identities. Going one step further would include MFA for accessing the wallet. For example, the wallet could also be secured by key files on external hard drives, biometrics, smart cards, or HMAC-based security tokens.

Further risks to the wallet's security may arise from interfacing it with other wallets or agents, leading to all kinds of possible exploit scenarios. Proper security by design is essential to limit the necessity of frequent updates to the wallet application, especially when deployed on IoT platforms, where updates are notoriously slow to roll out.

Metrics to track the security of a wallet are:

- The **frequency of updates** supplied by the wallet's developer.
- In addition to the update frequency, the developer's **reputation** has to be monitored. This can also include checking for adequate certification. Especially for eID standards exists that regulate, for example, eID-cards or electronic international passports. For example, ISO/IEC 14443 might need to be used or adapted if wallets are to be used in similar use cases.
- Monitoring **hardware security** features, i. e., TPMs, and potential vulnerabilities, is essential to ensure the wallet stays as secure as intended.
- The wallet's developer's **business model** can also be used to indicate the wallet's security. Especially an ad-supported model, which includes ads into the wallet, seems highly problematic, as those ads are processed by the same code responsible for keeping the wallet's secrets.

4.3.2.6 Data Protection Management

Like the security of key material, data protection management focuses on the involuntary disclosure of credential information instead of key material. This involuntary disclosure may happen because of insecure software or user interface design and social engineering.

As a result, data protection management must consider secure storage options and sufficient user interface design to prevent or warn users of potentially unwanted disclosures. User interface design lessons can probably be learned from how app permissions are managed on modern Apple's mobile operating system (iOS) and Android smartphones. This problem may not be as important for IoT devices because the device's identity should not contain protected PII. Additionally, the nature of IoT device credential exchanges requires more automated rules of when, where, and to whom credentials are shared.

4.3.2.7 Conclusion

Besides the backend infrastructure provided by the DLT, the wallet is the core component of SSI. It has to handle security-sensitive private keys and data protection-sensitive PII credentials. As the central point of contact for users of SSI systems, the user experience of the wallet is essential for the success and large-scale adoption of SSI. For IoT applications, the wallet may not be as visible as for personal use, but it is still important for the device's security and integrity of the provided data.

4.3.3 Agents

Agents are the identity proxies for any entity's wallet to interact with other entities outside the local LAN. It is secondary whether an agent operates on behalf of a person, a company, or an IoT device. Its primary functions and provisions are always the same. They provide persistent endpoints and implement the necessary protocols for communication with other agents. Additionally, they may store some information on behalf of the entity they represent, either for synchronization between different methods of accessing the agent (e. g., from a smartphone app or desktop app) or to release information to other agents.

An agent is comparable to a mail server, which receives, stores, forwards, and sends mail for the user. In this comparison, the wallet would be the user's mail client. Like a mail server, most people cannot set up and run their own SSI agent. This difficulty has also affected OIDC as a user-centric identity management system, where users could theoretically host and use their own identity server. In the end, only very few do so, and, in the case of OIDC, even fewer trust the self-signed assertions provided by those who do. This results in a few large corporations acting as identity providers. The latter problem, however, does not exist with SSI agents, as they usually do not self-issue VCs.

4.3.3.1 Information Model

The agent manages the user's identity connections with other identities. It provides a persistent endpoint to store and forward messages intended for the user and their wallet, which might not always be online. Consequently, the primary management aspects of the agent concern establishing and maintaining connections between wallets and agents. Additional features can include synchronization between different wallets and the automatization of tasks.

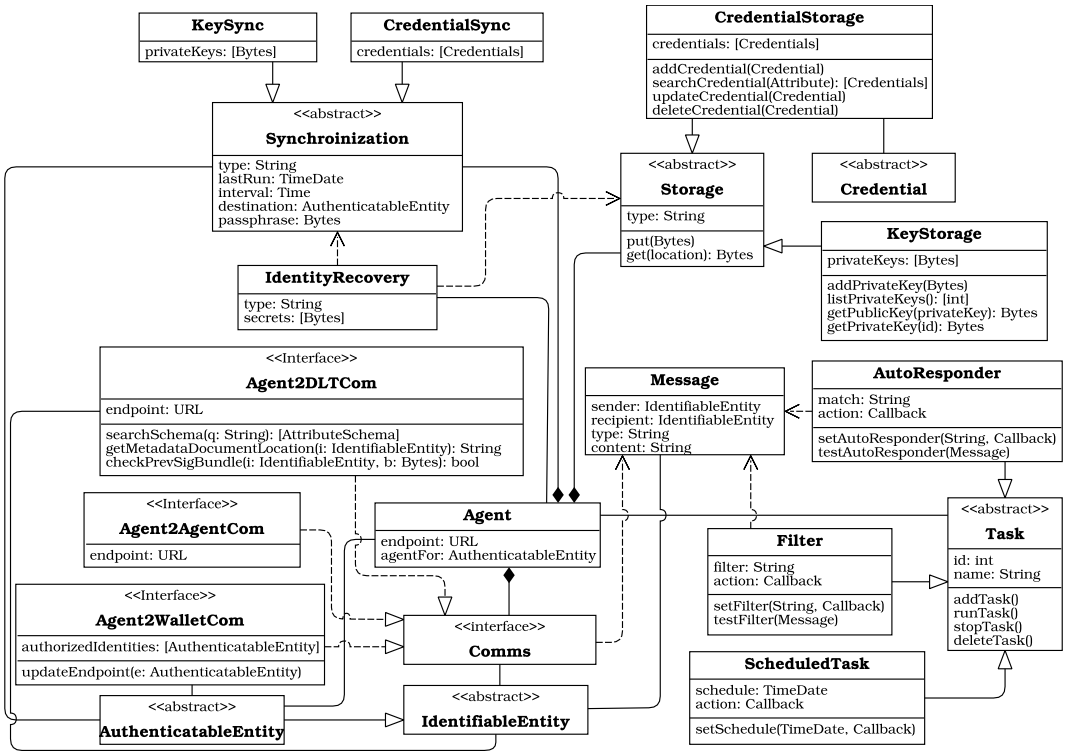

Figure 4.12: Agent information model

An agent itself consists of different subcomponents, which are also depicted in Figure 4.12, some of which are required (**bold**) and some are optional (*italic*):

- **Wallet Interface** (`Agent2WalletCom`): Provides an interface for wallet applications to communicate with the agent. This subcomponent is required to fulfill the agent's primary purpose as a persistent wallet endpoint.
- **Agent Interface** (`Agent2AgentCom`): The agent interface allows the identity's wallet to communicate with other agents and their identities through a transport network like the Internet. It is therefore required for exchanging messages with other identities.
- *Distributed Ledger Interface* (`Agent2DLTCom`): To retrieve metadata, the agent needs to be able to access the distributed ledger. This component is optional, as the lookup can also be done directly by the wallet application. However, including it in the agent may allow for preferable outsourcing of used storage space and mobile data usage.
- *Credential storage*: As a backup and synchronization mechanism, the agent may also act as a personal credential storage. This would allow the identity to recover credentials affected by data loss or theft of the wallet without having to re-issue them. Because the credentials contain PII, this feature should be optional.
- *Private key synchronization* (`KeySync`): As a comfortable method of synchronizing private keys between multiple wallet applications, the identity holder's agent, as a central point of contact for this identity, might facilitate the synchronization process. This component is (and should be) optional, as the agent is generally a proxy not directly controlled by the identity holder. The identity holder might not want to trust the agent's operator to store a copy of the private keys.
- *Identity recovery*: In combination with the credential storage and private key synchronization, the agent can also assist in recovering access to a lost identity. Proper care must be taken to prevent abuse and identity takeovers. Counter-measures might include forced waiting periods, notifications, and peer approval.
- *Tasks*: The agent can perform specific tasks for the user. Those tasks may be filtering and spam protection, automatic responses for defined identities, or scheduled operations, like cleaning up unused connections.

4.3.3.2 Organizational Model

Referring to the agents' comparison with mail servers, the organizational structure and relations play out similarly. The agent provides the user a service. As service provider, it must provide a service desk to handle the users' issues. To provide the service as required, the agent must follow the standards and documentation provided by the SSI community. If standardization progresses and a standard for SSI agents is developed, this will also become relevant. Both sides of the organizational relations are shown in Figure 4.13.

With its central position within the SSI system, the agent is connected to most other organizational units. In its primary role as the persistent endpoint of the user's wallet, the agent is connected to issuers, relying parties, and TGWs. Here it serves as a middleman to receive or present VCs on behalf of the user. The agent is also connected to the distributed ledger to help vet connections to issuers and relying parties. Information from the distributed ledger can also be used by the agent to provide additional information about the semantics of received VCs.

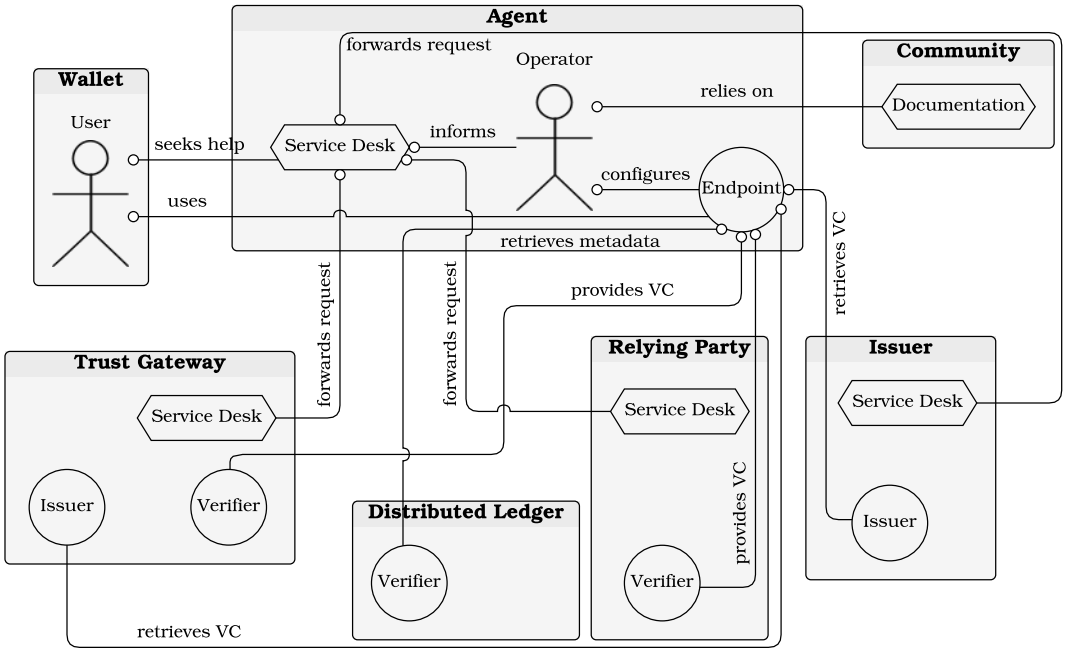

Figure 4.13: Organizational relations of the agent

The agent is run by an operator, as opposed to the user's personal wallet. This operator is usually part of a corporation that provides SSI agent services. Within the decentralized SSI system, it is often tricky for the user to determine where and how something went wrong. As a central point of contact for the user, a professional agent is also in an excellent position to provide a service desk. The agent has information about failed exchanges and can steer the user in the right direction or even relay the user's service requests to the right issuer, relying party, or TGW.

4.3.3.3 Communication Model

An agent is identified by an URL provided by the identity holder. In the case of a public identity, the agent's location is stored in a public repository, where the authenticity of the location can be checked by validating a signature. For private identities, which are usually only disclosed to a single entity, the agent's location is provided during the first exchange. Using the same individual agent endpoint across multiple identities can introduce a way to correlate and track those.

The core operation of the agent is initiating connections with other agents and exchanging credential information. Figure 4.14 shows the sequence of exchanged messages to establish initial contact between two agents of SSI-capable entities.

The initial connection between two entities' agents, where the target identity does not have a published public metadata file, works slightly differently and without the involvement of a distributed ledger. All required metadata information must be exchanged directly at first contact via an out-of-band method, as shown in Figure 4.15. This does not impact the further verification or issuance of credentials, as those credential definitions are again stored on a distributed ledger.

While both previous exchanges between agents are the most interesting, as they provide the foundation for reliable message exchange within an SSI system, further communication protocols must be defined for the agent's synchronization service. As a single user may use multiple wallet devices, synchronization conflicts may arise if wallets are not constantly online. For credentials, however, due to them

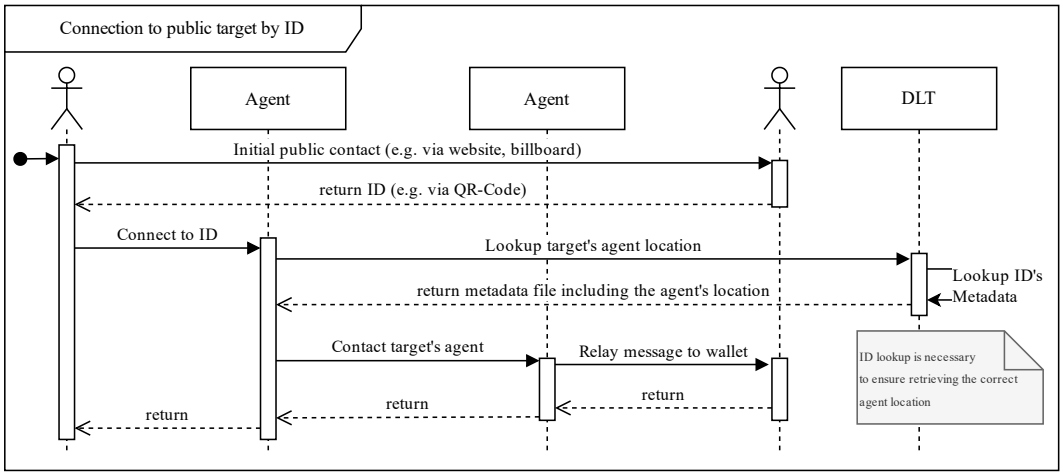

Figure 4.14: Sequence diagram of the connection establishment to a public SSI identity and agent

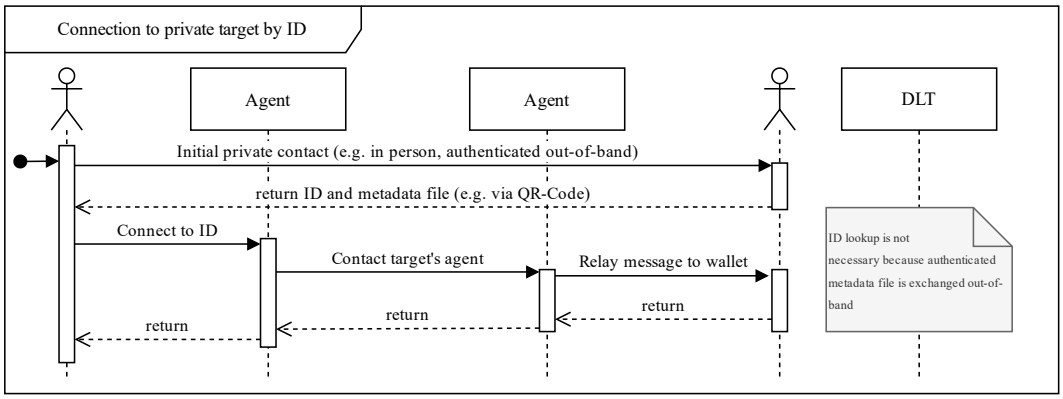

Figure 4.15: Sequence diagram of the connection establishment to a private SSI identity and agent

being signed by a third party during their issuance, the most recent version can be determined easily for each combination of identity, attribute, and issuer. Prior versions of the same credential should no longer be valid.

4.3.3.4 Functional Model

The functional domains of agents consist of forwarding messages between the wallets of different users and syncing the contents of a user's wallet. Additionally, the automatization of recurring tasks may be configured at the agent to reduce the complexity and workload of the user's wallet. As an extension of the user's wallet, most functional management operations are directly done by the user. Some exceptions, such as a spam filter or abuse prevention and account recovery, may apply. Figure 4.16 provides a hierarchical breakdown of the different functional domains.

Agent Functional Domains				
User Managed				Provider Managed
Credential Presentation	Wallet Synchronization	Task Automatization	Account Recovery	Spam/Abuse Prevention

Figure 4.16: Functional domains of an SSI agent

The user-managed credential presentation and wallet synchronization functions are exposed to the users' wallet applications. In order to allow the user a free choice of wallet applications, a standardized protocol for these management operations is required. Task automatization can be controlled either through the wallet or through a web-based interface of the agent. The reason for allowing both is that task automatization may be very individual to the agent's primary purpose, and a standardized protocol that works with any wallet is rather unlikely to be established.

Account recovery involves both the user requesting the recovery, and the agent's provider, verifying the request's legitimacy.

Last but not least, spam and abuse prevention may be influenced by the user but are primarily managed by the provider. This includes checking and updating denylists, running heuristic analyses, or forming trust relationships with other agents.

4.3.3.5 Security Management

As an always-online component of the SSI system, the agent's security is extremely important. Keeping the availability of the agent high has to be prioritized, as failure to do so would render large parts of the SSI system inoperational. For its primary purpose of forwarding messages from and to wallets, confidentiality is important, but the agent should never possess plain text messages anyway. Thus, a lack of confidentiality on the agent may leak metadata, like sender and receiver, but not actual message contents. Similarly, the integrity of messages is handled end-to-end between the wallets, and the agent should – by design – never be in a position where it could impersonate or modify messages.

Security management becomes more difficult the more features are implemented by the agent. If credentials or keys are stored for synchronization between wallets, those need to be protected particularly well. In this case, the agent basically acts as an online password safe.

Automated tasks add another level of problems for securing the agent. Depending on how this automation is implemented, it could be abused for spamming or credential theft.

4.3.3.6 Data Protection Management

The agent is an SSI system's most visible aspect of a personal identity, as it provides a point of contact. Actual identity-identifying information in the form of identifiers is kept private due to the use of unique identifiers for each connection. However, using the same agent with different identities may allow correlation between those identities, depending on how many identities use the same agent.

As an analogy from email systems using the Google Mail server to send mail does not reveal much, as many users have one or more Google Mail addresses. Instead, using a private mail server, just the fact that a custom mail transfer agent (MTA) is used can provide identifying metadata.

4.3.3.7 Conclusion

The agent is an integral part of the SSI system, providing the identity wallets' persistent connectivity to the rest of the network. In most cases, where the availability of the individual wallets cannot be guaranteed or where wallets should always be able to be sent messages, one cannot go without agents. Only in those circumstances, the wallet may also inherit the agent's responsibilities.

To reduce the complexity of the wallets, the agent also acts as an abstraction layer, allowing the easier development of wallets, which increases the wallet's security, which is extremely important. A flexible agent implementation also allows for migrating or switching between different IAM systems, as it can provide connections to different networks transparently.

4.3.4 Relying Party

Relying parties, service providers, or verifiers are all common names for entities consuming assertions about identities. As such, they are the driving factor creating a need for easy but secure identification across many domains. They require the identification and authentication of entities and want to verify the correctness of any attributes associated with the entity. Traditionally all of those tasks are performed by the RP itself, but usability drawbacks, operating cost, and scalability issues push for more streamlined approaches like FIM and, ultimately, SSI.

In an SSI system, the RP can request and verify an entity's assertions. Those assertions are provided to the entity by independent third parties, the issuers. RPs no longer need to check self-asserted statements, but can outsource verifying statements to IdPs, AAs, or issuers. Those may be in a better position to assess specific assertions. An RP might request some credentials while also acting as an AA for others.

4.3.4.1 Information Model

The primary purpose of RPs is the consumption of entities' assertions to provide the entities with a specific service. An assertion in the form of a VC, as it is used with SSI, consists of two parts: the asserted attribute and the issuer's signature. Handling of these attributes is done via the RP's `CredentialReceiver`, which handles the SSI-specific details. Those components and the following ones are displayed in Figure 4.17.

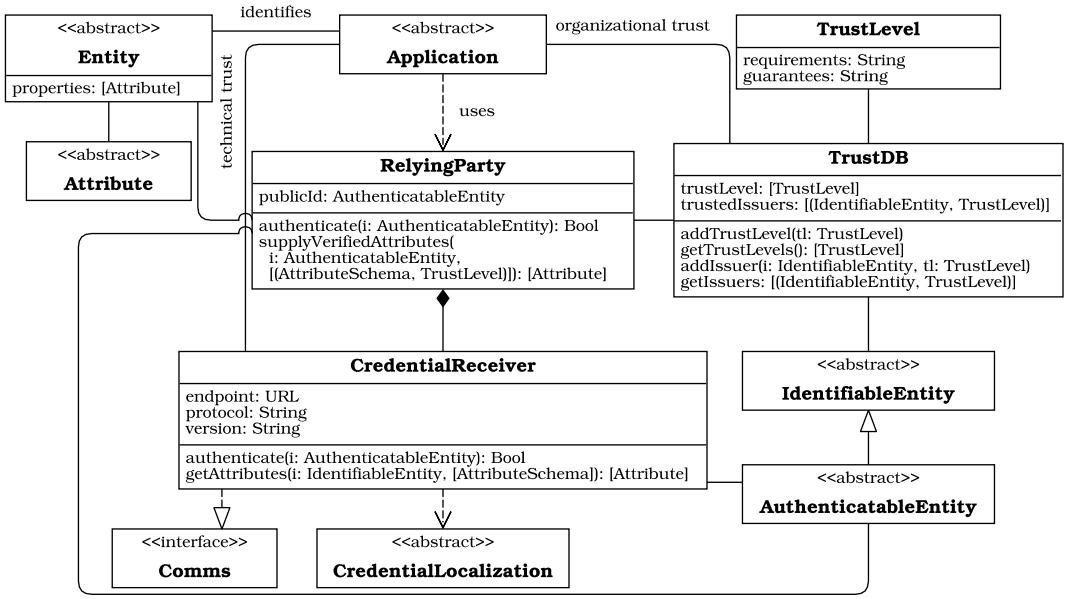

Figure 4.17: Overview of the RP's information model

Attributes can be expressed in various syntaxes and semantics, making them difficult to process automatically. Thus, meta-information is required to properly interpret and process the large set of possible assertions, which can be used in an open SSI ecosystem. Standardization of a specific set of attribute formats is unlikely to work, as this kind of standardization has already shown to be challenging for FIM systems. A special localization service can provide the RP with automated rules to convert attributes into a form that is understood by the application.

As the number of possible issuers within an SSI system is not limited to a known group of IdPs, quantifying the reliability of assertions by a specific issuer is not easy. Verifying the correctness of the issuer's signature alone is not sufficient. Instead, where direct trust relationships are limited by scalability efforts, the level of assurance provided by a specific issuer must be deduced with the help of a system similar to the federations of FIM. The federations of SSI can be as small as only two entities and can be built and removed dynamically. They are managed by the RPs using a trust database, which describes the federation's terms and conditions.

4.3.4.2 Organizational Model

The RP uses its verifying service to identify, authenticate, and authorize access to its own services. To do so, the necessary organizational connections to other SSI components are shown in the organizational model of the RP in Figure 4.18. The RP's position between the issuers and users can introduce new responsibilities (i. e., selecting possible issuers, determining which issuers to trust, or reacting to incidents at specific issuers) compared to self-run IAM and FIM.

The relying party has two actors. The operator is using the information provided by the community to run the verifying endpoint. They are also responsible for reacting to information provided by the RP's service desk to fix any potential issues with the service. The trust manager has to decide which issuers are sufficiently trusted to accept their VCs in authorization decisions. To do so, they are connected to the distributed ledger and their corresponding issuers' trust manager. Especially the trust managing aspect may overwhelm small and medium-sized RPs. As a result,

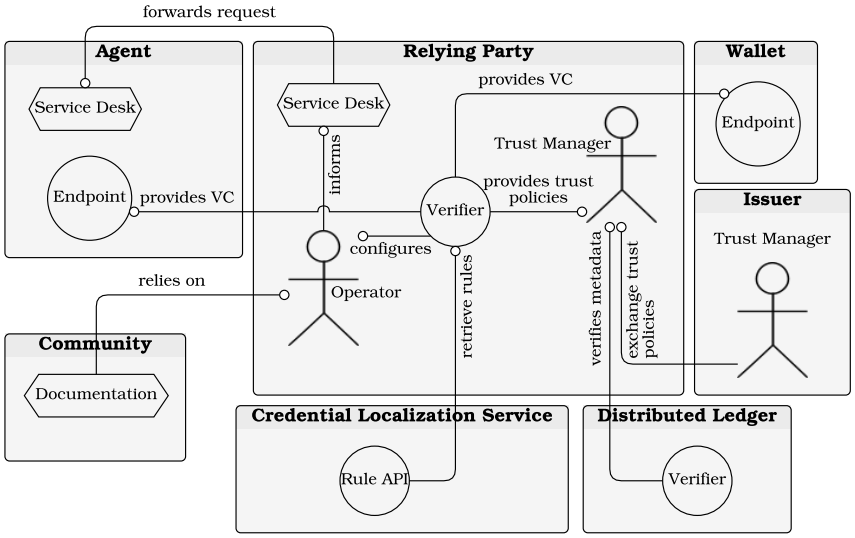

Figure 4.18: Overview of the RP's organizational model

those RPs may outsource the trust managing services to a specialized provider. The result of out-sourcing the trust management part is similar to FIM systems, where the federation operator is responsible for entering into a contract with all other participants.

4.3.4.3 Communication Model

Due to the general messaging infrastructure of SSI, all communication between the RP's customer and the RP is done between the RP's endpoint and the customer's wallet. There may be intermediaries providing continuous availability to the RP and the customer, e. g., through the help of one or more agents. Direct interaction between the RP and the issuer is not required or specified by the SSI design. All necessary information about the issuer should be available on the distributed ledger. As a result, the communication necessary for identification and credential exchange between the wallets follows the protocol described in Section 4.3.2.3, and communication with the user's agent is done through a standardized protocol, which is described in Section 4.3.3.3.

Additional connections between the RP and other components are described in their respective sections. The connection to the CLS and its API is described in Section 4.3.6.3.

4.3.4.4 Functional Model

At its core, the RP's function is to deliver a service to the customer or the entity in contact with the RP. This function can be of varying complexity and may be able to be completed nearly instantly or take multiple days to process. Configuration of the provided service is highly individual and dependent on the service itself.

Regarding SSI, the consumption of VCs is the most important function of an RP. This function requires more thought than with more traditional IAM systems, as basically, anybody can act as an issuer of VCs with SSI. The main decision, therefore, is which issuer to trust with their assertions. This decision must be integrated into the RP's risk management, marketing strategy, and know-your-customer requirements. Allowing any issuer provides the easiest access for users but is probably detrimental to data quality standards.

Configuration of the credential consumption process can be done in three ways: allow listing of known good issuers, deny listing of known unreliable issuers, or outsourcing the issuer management to a third party. This third party can act like a federation operator in FIM to provide a framework for issuers to get recognition. As multiple such federation-like organizations can be formed within the whole SSI ecosystem, each federation can focus on specific industries of attribute assertion (e. g., banking, workplace, sports, gaming).

4.3.4.5 Security Management

SSI allows the RP to externalize some parts of its traditional IAM duties. However, security should still be considered carefully for each individual RP. Using SSI, novel security risks are introduced which may not be suitably covered by existing security management practices.

SSI-related assets that need to be considered for security management-related risk management are:

- **Distributed ledger connector**: Enabling access to the SSI ledger through the `Comms` interface is necessary to perform authentications and check revocations. Without it, the availability of the IAM system is severely limited.
- **Localization engine**: Localization of credentials is an essential part of allowing a wide variety of users to access the RP's services. Incorrect localization rules may result in unintended authorization choices.

To protect the confidentiality of the users' data and the data associated with the service, the user's authentication has to be done securely. Important aspects include using secure cryptographic protocols and tried and tested implementations. The choice of authentication mechanism is likely dependent on the mechanisms supported by the SSI system, so the choice to support a specific SSI system should include a reliable and strong authentication process. Access control is usually in the hands of the RP, which bases its decision on the authentication's result and additional information, e. g., attributes supplied by VCs. Due to the multitude of credential formats expected to be used in a global SSI system, localization of credentials may be required for some RPs. Extra care has to be taken to translate and adapt credentials to the RP's systems to ensure correct access control decisions.

The same aspects for authentication and access control must be considered for protecting the integrity of the RP's data. The integrity of the IAM data is more difficult to assess as it may originate from many issuers. Suppose the RP cannot determine the integrity of the issuer's claims or limit the number of issuers to a manageable level. In that case, they require assistance from a (specialized) federation or PKI.

By including external components (like the distributed ledger of an SSI system) into the services of an RP, the RP's own availability is partly dependent on those components to function. However, for a limited time, authentication and verification of credentials can be done without an active connection to the distributed ledger, as the metadata on the ledger usually only changes slowly. In this regard, the RP's availability is not impacted by availability disruptions of the distributed ledger.

Close monitoring of authentications and issuers to detect anomalies, like increasing authentications using very new issuers or using uncommon credentials, might be necessary to detect potential misuse. Therefore, a reporting system providing comprehensive metrics around the authentication and authorization process is essential.

4.3.4.6 Data Protection Management

As the consumer of user-provided attributes, the RP is obligated to handle all shared data with proper care. In accordance with data protection laws, each RP should determine which attributes are needed for what duration and where to store them, if necessary. Depending on the use case, storage of attributes may not be necessary at all. During each authentication, the user can easily re-transmit the attributes required for authorization. However, some information (e. g., shipping and billing addresses) will likely be required to be stored, depending on the service offered by the RP, local laws, and due diligence requirements.

Even more care must be taken if the RP requires or wants to share the data of its users with other organizations, whether directly for services provided or not. In an SSI system, the user should be in control of passing their verified credentials to each organization. Working around this principle by allowing information sharing between organizations without user involvement should be strictly limited to cases where it is absolutely necessary.

A neat way to inform the user of the processing of their attributes by other organizations in the RP's processes is to specify the organizations and the release of those attributes directly in the credential request. With the EU's GDPR, each request for personal information must be coupled with a reason for why this data is required. Those reasons should also be included in the credential request. This allows the user to more clearly see why certain attributes are required and revisit those reasons later if the user's agent records the requests.

As each RP's credential request is replied to directly by the user with a corresponding credential response, the user must actively confirm the attributes' transmission. Through this system, the RP can prove the user's consent to the transmission as the uniquely created response.

If some or all of the attributes provided by the users are stored at the RP, processes need to be established to continually check their validity. The RP must also establish a process to delete no longer required personal information. Deleting personal information may be coupled to a period of inactivity by the user or the user's active request for the account's deletion. In both cases, deletion of the data should be possible, not only in the active database but also in any backups or replications that have been made.

4.3.4.7 Conclusion

As a core component to any IAM system, the RP and its new SSI capabilities are included in the architecture. The RPs have a strong incentive to support SSI but face challenges adapting to the system. For already existing RPs, the decision to adopt SSI requires the choice of the most suitable path to take:

- **New service**: One way to protect existing systems and pave the way to adopting SSI is to set up a separate new service, which does not interface with the old service's IAM and delivery systems. New customers can then use SSI, while old customers can keep the system working the way they are used to. Over time, if SSI catches on, the user base will automatically move to the SSI-enabled service, and the legacy system can be shut down eventually. In the meantime, however, two systems need to be operated.
- **Complete migration**: A less customer-oriented approach is to specify a date at which IAM systems are switched over from traditional IAM to SSI. This approach forces users into the new system but prevents the RP from operating two systems simultaneously.

- **Dual-Stack operation/transition**: A hybrid approach would allow users to keep using both IAM systems concurrently with the same service. This allows users to try the SSI authentication system and switch back if they are uncomfortable with it. For the RP, this approach requires the most attention to detail, as interfacing two systems will create challenges and interdependencies which are difficult to overcome.

4.3.5 Issuers

In traditional IAM systems, the issuers of attribute certificates are called IdPs or attribute authorities. With SSI, the identity is provided by the user and only augmented with individual verified attributes in the form of VCs. Therefore, the name IdP is no longer fitting. Instead, the entities issuing VCs are called issuers.

4.3.5.1 Information Model

Issuers are the natural counterpart to RPs, which are dependent on the issuers' attribute data sources. For an issuer to be trustworthy, they need to be able to guarantee the correctness of the VCs they are issuing to a defined degree (i. e., assurance or trust level). In most cases matching users to attributes requires direct contact with the user, as the issuance of VCs is mainly a task of digitizing already existing paper documents, at least at first. For this task, the issuer must either have access to the original documents or act as a notarization service. The latter can transform basically any document to VCs. An example of an issuer accessing the original documents might be a government office creating a VC for an eID or a bank asserting a customer's bank account number.

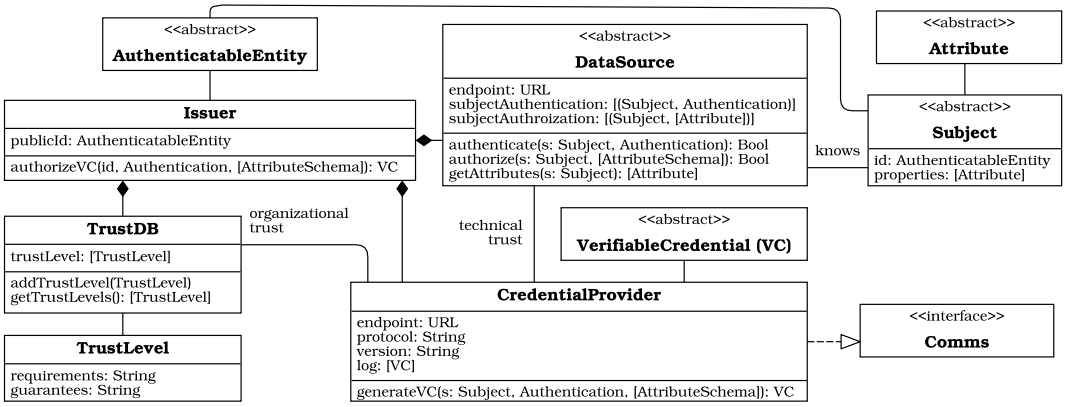

Figure 4.19: Overview of the issuer's information model

As shown in Figure 4.19, the issuer comprises *data sources*, a *credential provider*, and a *trust database*. The data source is where the issuer can obtain qualified attribute statements for entities. Those entities, if correctly authenticated and authorized to do so, can then use the credential provider to obtain VC. The trust database indicates at which trust level VCs can be provided. The RP will use its own trust database to evaluate if the indicated trust level is achieved.

4.3.5.2 Organizational Model

The organizational model of the issuer is similar to the RP's one. A schematic overview is provided in Figure 4.20. In general, the issuing service is used to provide the user's wallet or agent with VCs. To do so, the issuing endpoint is run by an operator supported by the community's documentation.

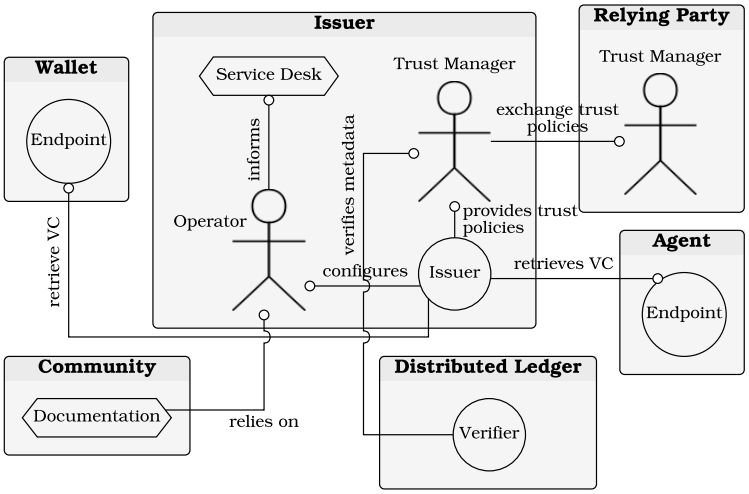

Figure 4.20: Overview of the issuer's organizational model

To assure the relying parties, which ultimately rely on the accuracy and validity of the issued VC, the issuer's trust manager sets up policies that describe who can be issued VC. Additionally, they deposit information about the syntax and semantics of the VC on the distributed ledger.

4.3.5.3 Communication Model

Communication between the issuer and the VC holder is standardized by the selected SSI system. In most cases, the holder would be the same entity as the VC's subject. Both are represented either by their agent or directly with their wallet. The exchange protocol needs to be highly standardized to allow a multitude of agents to support the system, which in turn allows more users to access the issuer. Management information is exclusively exchanged via the distributed ledger (i. e., links to metadata files).

The protocol shown in Figure 4.21 provides a method for issuing VCs in three distinct phases:

1. In the **preparation phase**, the issuer publishes their identifier and metadata (including a long-lived public key $k_{i_{pub}}$) to the ledger. This is a one-directional trust anchor, as every other entity can retrieve the metadata and check the issuer's assertions within the issued VCs.
2. During the **exchange phase**, the subject (or any entity acting on their behalf) can request a VC with attributes known to the issuer. During the exchange, the subject identifies and authenticates with the issuer. The issuer generates a new key pair k_n to sign the new VC. Only the public key component $k_{n_{pub}}$ is signed with the issuer's key $k_{i_{pub}}$ to link the issuer and the new key pair.
3. After the VC's validity period, the **purging phase** sees the issuer invalidating the no longer valid VCs by publishing the key pair k_n, which includes the private key used to sign the VC. As a result, similar to the OTR protocol [20], any expired VC cannot be proven not to be counterfeit. The subject needs to refresh the VC before it expires, to always be able to prove the contained attributes.

This system does not include a method for the issuer to revoke VCs. Revocation and checking revocation would require the issuer to publish revoked credentials to the ledger or the RP to query the issuer to verify the VC's validity. The first

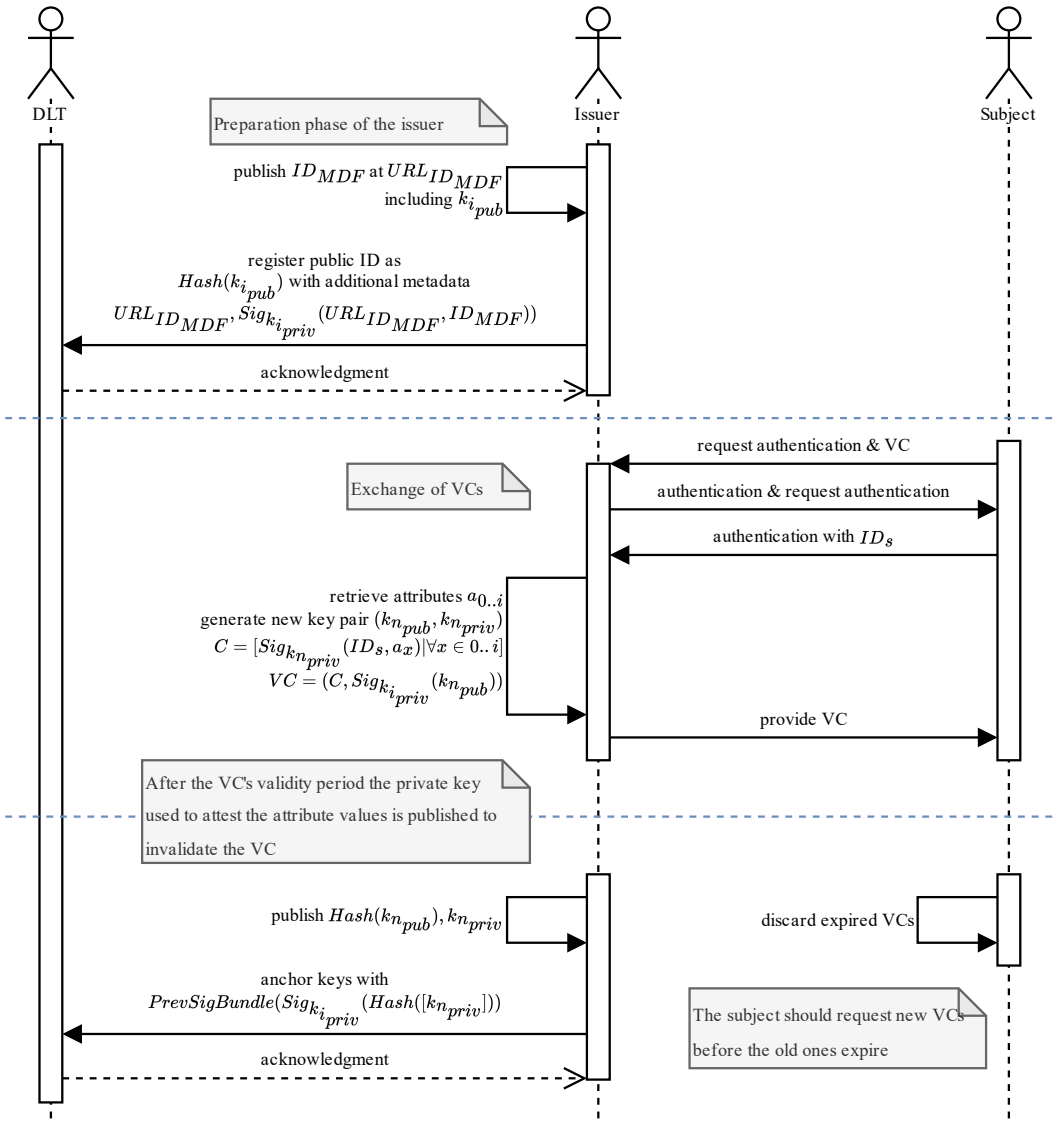

Figure 4.21: Communication model for issuing a VC

solution can potentially create a permanent connection between a specific VC, its subject, and the issuer. The latter solution can debilitate the desired separation of the issuer and RP. Both solutions result in undesirable properties.

For claims with attributes that specify "holder can access system x" or similar statements, the holder might want to delegate the permission to another entity. In this case, the holder becomes an issuer. As a prerequisite to this delegation, the original issuer must also include an attribute stating "holder can delegate permission". If both attributes are part of the same claim in a VC, the issuer can create a new VC to extend the permission to another entity. Both the original and the new VC can be used together to prove access permission by the new holder. This system can also be advanced further by specifying in more detail under which conditions delegation whether possible and if chaining delegations should be allowed.

4.3.5.4 Functional Model

Functional design decisions depended highly on the type of attributes the issuer creates VCs for. To differentiate between the different types, VCs are categorized by their data source and data type. The data source describes whether a VC is a digital copy of an already existing physical document or if the VC only exists as a digital assertion.

- **Digitized document**: Transferring physical documents into a digitized version can be done by an issuer acting as a notary. Depending on the complexity of the document, this process must be designed with utmost care. Decisions on the granularity of attributes extracted from a document and the presentation in a VC must be taken.

  For example, a school report contains the individual disciple's marks, a final aggregated mark, and a pass or fail indication. As it is, this report cannot be presented in parts (e. g., presenting only the marks of STEM disciplines). However, a VC could be used that way if the individual data is encoded appropriately. During the transformation process – no matter if it is done by hand or using automated tools – errors might be introduced. The potential for errors increases with the depth of detail that is mapped.

- **Digital document**: A purely and originally digital document is more easily presented as a VC, as the individual components are already available digitally, which makes transcription errors much less likely. However, purely digital data usually only exists with the issuing authority, and they would need to act as an issuer. This requirement may slow down the issuance of VCs.

Independent of the form of the original data, the issuer must decide whether they store the original documents after they have issued the VC. Depending on the issuer, some might need that data permanently anyway, but especially notary services need to specify storage duration and deletion policies. Traceability requirements require notary issuers to keep copies of the original documents. However, as long as the user has access to the original documents, they could always redigitize them if doubt about their correctness arises, and the issuer would not have to store the original. Notary services are usually not free, and issuers might therefore charge the user for issuing VCs. Determining the cost of the service will need to factor in the difficulty of generating the VC, the duration the VC will be valid for, and the required research to provide the intended LoA.

4.3.5.5 Security Management

As the source of high-quality attribute information within a specific domain, an issuer's data repository is a prime target of attackers seeking to steal the data contained. SSI tries to prevent the aggregation of precious private information at a specific entity, but qualified authorities of one domain still need to store and process such data. Additionally, highly trusted issuers are also prone to being tricked into creating bogus VCs. This is similar to tricking PKI CAs into signing certificates for domains that an attacker does not legitimately control. Security management procedures should be oriented at existing best practice approaches for CAs (e. g., ISO/IEC 27099 or BSI TR-03145-1) and IdPs (e. g., the Sirtfi framework [15, 156]).

Metrics to monitor issuing of VCs for potentially security-relevant irregularities, specifically for SSI, are early re-issuing of VCs. VCs need to be regularly re-issued because of limited validity periods. However, if the validity is still sufficiently long, a re-issuing request might indicate the abuse of an identity's stolen private keys.

If the keys of an identity are ever potentially compromised, they need to be rotated. In case there is the possibility of maliciously issued VCs or messages before the compromise is detected and fixed, the old keys must also be revoked, and the involved parties be notified. To do so, schemes from CAs, like revocation lists or OCSP, can be adapted. As the RP is only querying for the validity of the issuer's signing key, the request does not leak information about the user.

4.3.5.6 Data Protection Management

The data repository at an issuer is most critical regarding data protection management if the repository contains PII or otherwise confidential or non-public information. However, an issuer's job in protecting this information is easier than that of a classic FIM IdP, as the issuer should ever only reveal and assert the information to the entity it belongs to. The issuer does not have to determine which SPs it deems trustworthy. This burden is moved to the user receiving the VCs. However, some responsibility in educating the user on the proper usage of the VC and the risks of disclosing information to third parties may be required.

4.3.5.7 Conclusion

Issuers are an integral part of an SSI system, and their trustworthiness is essential for RPs to adapt them as sources of high-quality identity data. Management decisions play an important factor in the successful operation of an issuer, especially because the underlying technology is relatively new and under rapid development.

4.3.6 Credential Localization Service

A new component introduced by the architecture is the credential localization service. Similar services have already been described for use in FIM-based federations [148] but have not been used in productive federations yet and have not been described in detail for SSI.

In SSI, the many diverse users, issuers, and RP can introduce different credentials. Standardization approaches are unlikely to succeed, especially as there will be different approaches and requirements for VCs worldwide. The proposed localization service should not only translate from one VC schema to another but also provide adaptions to locally understood scripts, traditions, and legal requirements.

There are two possible ways to implement a localization service.

1. The credentials are passed by the user to the service, which localizes them and creates a new VC of the localized version referencing the original VC as the source. The resulting VC is passed back to the user. The localization service acts as both an RP and an issuer.

2. More prudent of the principle of data economy, the second approach prevents the localization service from seeing the actual values being localized. Instead, it provides a schema according to which the localization can be achieved. This schema can be requested by the RP and used to process the received VCs locally. The user's data is not processed outside the RP.

In this architecture – also depicted in Figure 4.22 – the second option is chosen for localizing credentials, as it is more in line with SSI's data protection and self-determination principles. The first option also does not differ too much from how an actual AA works, so if a service like this is required, it is not difficult to set up.

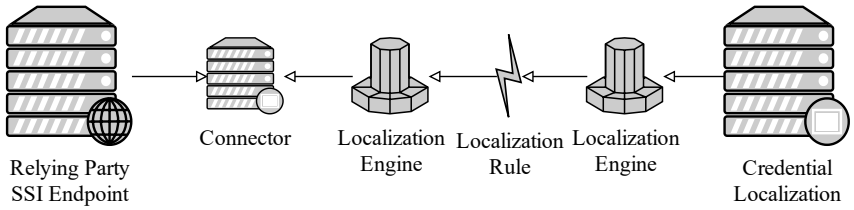

Figure 4.22: Overview of the CLS

4.3.6.1 Information Model

Core components of the CLS are the localization rules, which describe how a specific credential schema can be transferred to another schema. The rules involved can be pretty simple or more involved (maybe requiring a combination of credentials), and this flexibility must be expressed in the rules design. Figure 4.23 displays the CLS's information model and its components.

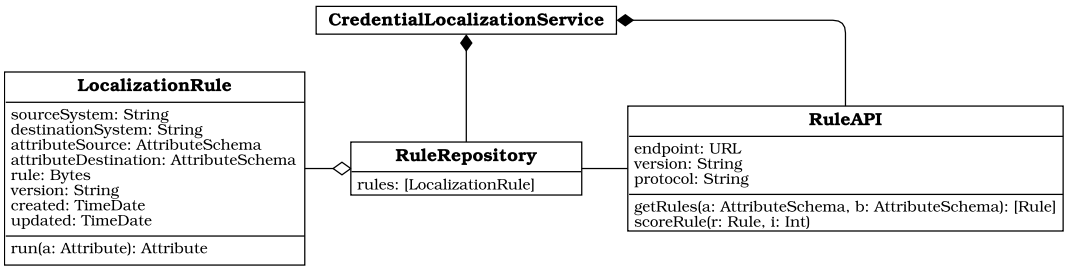

Figure 4.23: Information model of the CLS

In addition to storing the rules, the rules must also be kept up-to-date if they should not be sufficient or incomplete. As the CLS does not observe the actual localization process, as it is only run on the RP's side, a feedback loop is essential to determine the effectiveness of the provided rules.

4.3.6.2 Organizational Model

The CLS only has direct business relations with individual RPs, which subscribe to specific localization services, and to the community. Those connections are depicted in Figure 4.24. Run by its operator, the CLS provides an API to which the RPs can subscribe. The API can then be used by the RP to retrieve fitting localization rules.

To identify and fix any problems, which could arise during the localization process, the RP's and CLS's service desks are also connected. The end user should never have to contact the CLS separately. To react to new VC schemata and steer the direction of commonly used ones, the CLS might also participate actively in the SSI community and its documentation of VCs and processes.

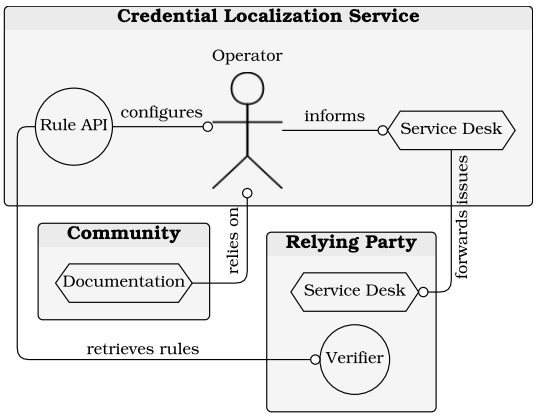

Figure 4.24: Organizational model of the CLS

The CLS can be operated either as-a-service or within the decentralized infrastructure. Decentralized operation is possible because the CLS does not need to access any attribute data, and the respective conversion rules and scripts can be exchanged using DLT. If implemented as a decentralized infrastructure, each RP is responsible for retrieving and applying the correct rules. Using a service to do so allows RPs to outsource this work and responsibility. A hybrid approach is also possible where easy localizations, i. e., reformatting dates, are done using rules stored in a decentralized repository and more specific localization rules, i. e., localizing a university's certificate to an organization's internal format, are provided by commercial service providers.

4.3.6.3 Communication Model

In contrast to the main SSI operations, the distribution and processing of localization rules can be designed individually or follow a (pseudo-)standard. Individually designed processes allow for more flexible options but also require the installation of custom connectors at the RP. There may be universal or application-specific localization services with different requirements and domain-specific features. A standardized but extensible approach could be integrated with the RP's SSI software, thus easing the deployment of the rules.

The basic query for retrieving rules from a localization service is shown in Figure 4.25. As the application of the rule is performed on the RP's systems, the localization service does not get any direct feedback about the applicability or performance of the rules. A feedback system is therefore important to provide the

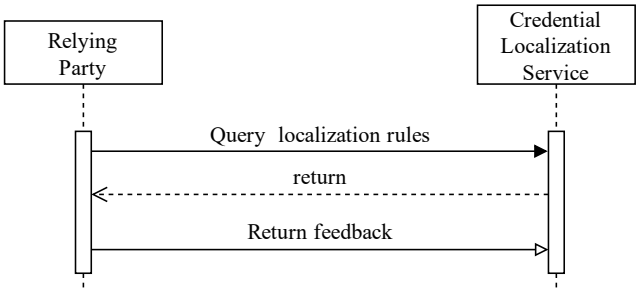

Figure 4.25: Primary communication sequence of the CLS

localization provider with information to improve their service. Details about the potentially personal information contained in the credentials the rules were applied to must not be shared with the localization service, however.

4.3.6.4 Functional Model

The CLS provides an interface to RPs where they can query localization rules for specific credential schemata.

Failures in credential localization can lead to a service not being usable. Thus their reliability is critical. Most credentials will either be understood directly by the RP, or the necessary localization rules are known to work well. Rarely used or seen variants of credentials that need to be localized on the fly are more complex to localize. It is the job of a good CLS to provide those.

Providing an inaccurate or plain wrong localization rule is more severe than the RP not being able to provide its service to a user. This might lead to improper authorization decisions. Depending on the automation grade at the RP those errors might also be difficult to detect if they are not reported by the users.

After installing and configuring a localization rule, it must be kept up-to-date. Those updates may be pushed to or regularly queried by the RP.

Downloading and using the rules may require a fee or subscription. From the perspective of privacy preservation, running the localization rules at the RP is definitely the better choice. If the fee or subscription should be based on a per-authorization or per-user metric, the CLS can hardly verify if the RP provides correct numbers. Resulting “flat rate” prices might be pretty steep for small RPs. Development of the rules, especially if no suitable localization already exists, is the primary added value of a CLS and the area of work where they can set off against their competitors.

4.3.6.5 Security Management

Using localization rules may require executing more or less complex algorithms on the RP’s side. If rules are poorly designed, this might introduce vulnerabilities at the RP. Furthermore, integrating localization services into an RP, including eventual dynamic payment, rule discovery, and update processes, might result in security issues. The CLS must develop its rules and integration in a way that does not threaten the security of the RP. To detect any issues with the rules, they should be created with unit tests of the complete code and, better yet, test-driven development (TDD). Using the tests, issues at the RP can be prevented.

The CLS itself must also be wary of its security to protect its intellectual property, the integrity of ts rules, and constant availability. In general, the CLS should follow security guidelines and standards applicable to any web service provider depending on the kind of RPs their working together with.

4.3.6.6 Data Protection Management

As described in this architecture, the CLS does not directly process PII. Still, as the localization process might be fairly complex, it might consult external resources (e. g., current lookup tables) and might leak some data or at least metadata this way. As a result, the design of localization processes must take into consideration what kind of data is being processed and where potential leaks can occur.

4.3.6.7 Conclusion

A CLS is not essential for the operation of an SSI system, but it can alleviate some of the potential downsides of SSI that come with the freedom to create credential definitions without central oversight.

4.3.7 Trust Gateways

Like the CLS, the goal of the trust gateway is to broaden access to the SSI system. They provide a trusted passage for moving VCs from one SSI system to another and help in connecting different IAM systems. To do so, they basically act as a cloud-based wallet and manage VC for users without appropriate wallets or agents of their own. A schematic overview of a TGW is shown in Figure 4.26. With multiple SSI systems, there might be chains of TGWs connecting multiple distributed ledgers. While a TGW offers accelerated access and ease of deployment, they are also concentrators for potentially highly personal information. Their design and deployment must be done with utmost care. An overview of other TGW systems, as they are, for example, used in FIM, is described in Section 3.2.3.

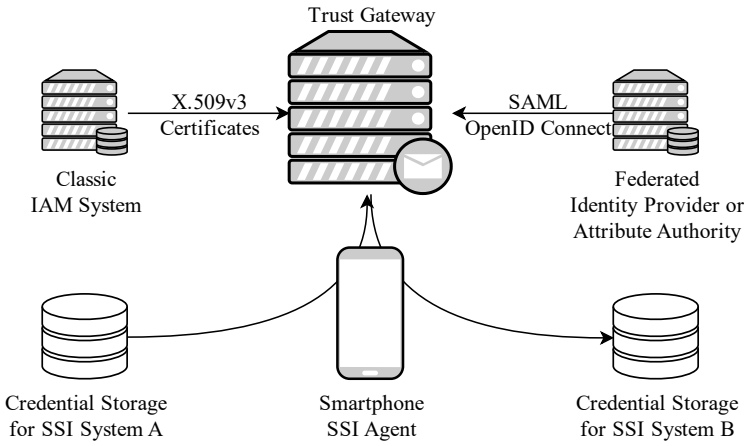

Figure 4.26: Overview of the TGW's connections

The TGW, as a new component to SSI systems, should provide better usability when starting an SSI system. If multiple SSI systems are developed concurrently, VCs can be moved between the networks without additional modifications to the RPs. In the long run, the aim should always be to use TGWs as little as possible and instead rely on direct integration of SSI and CLSs.

4.3.7.1 Information Model

The TGW has two primary components that need to be managed: the source IAM system's interface and the destination IAM system's. Both are linked via a connector component, which performs the necessary adaptions. This simple model is shown in Figure 4.27. The source or destination IAM systems can be classic IAM systems or SSI systems. However, for use with SSI, at least one system should be connected to an SSI system.

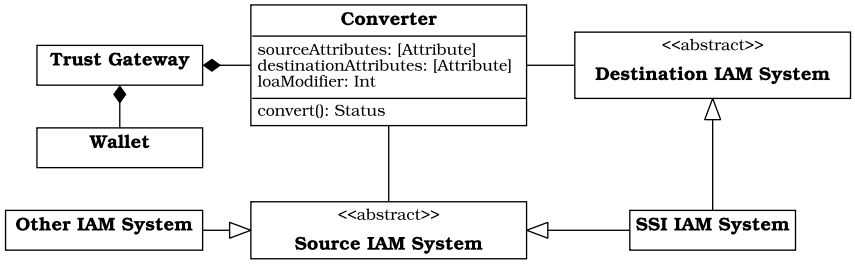

Figure 4.27: The TGW's information model

Figure 4.27 also shows that the conversion takes a few parameters, such as the list of input and output attributes. The TGW might not be able to convert every input attribute, so the output list of attributes might contain fewer attributes than the input list. Additionally, the break in the direct connection between the issuer and RP reduces the attributes LoA. How significant this reduction is must be determined by the participating entities.

Additionally, the TGW also acts as a wallet for users who do not have a dedicated wallet application. This requires more trust in the TGW but allows non-SSI users to access VCs issued to them.

4.3.7.2 Organizational Model

The TGW is essentially an issuer and relying party, where one of those parts has strong ties to organizations outside a specific SSI community. They may be operated independently, for example, as a notary service. Figure 4.28 displays the TGW's organizational connections other entities. This figure also clearly shows the resemblance between issuers (Figure 4.20) and relying parties (Figure 4.18).

Trust management is an essential step for the TGW. However, this conversion of authentication statements of different systems is not only challenging on a technical but also on an organizational level. The trust manager must negotiate policies for attribute exchanges within two separate IAM systems. If problems arise, good communication between the service desks of the involved parties is essential. The affected users must be guided to the right point of contact, as it might not always be obvious to the user where an error originates and who is responsible for its remediation. Additionally, changes must be communicated ahead of time to minimize potential problems.

The greatest difficulty for a TGW, however, is providing sufficient trust. While one issuer or IAM system might provide data with a high LoA, a relying party might no longer view the attributes' LoA as high after it passes through a TGW, and for a good reason. As a result, TGWs must either adhere to at least comparatively high-security standards as the original issuer and fulfill the same regulatory or legal requirements, including direct contracts or be only used for relatively inconsequential attribute conversions. A third option might exist when converting an attribute

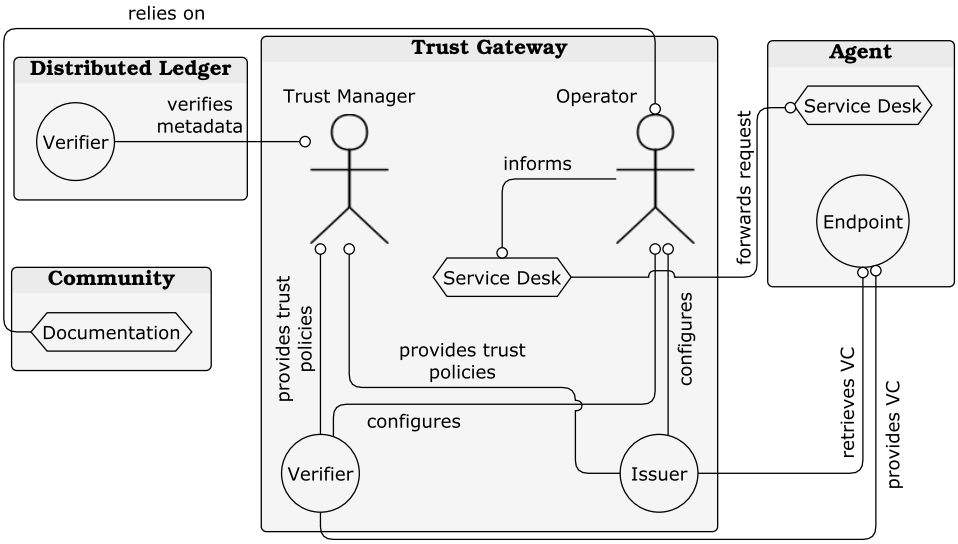

Figure 4.28: The TGW's organizational model

statement without breaking the original issuer's signatures and encryption is technically possible. This could be the case when the TGW proxies assertions between two technically identical but organizationally separated systems (i. e., two SAML federations or two SSI systems with different DLTs). In this case, the VC could be processed without translation, but the necessary metadata must be provided.

The trust manager's operator has to run both the endpoint for the verifier and the issuer. To provide good service, the service desk informs the operator about potential problems. The TGW's service desk is notified about issues by the user's agent's service desk.

4.3.7.3 Communication Model

As the TGW is connected to (at least) two different IAM systems, its effort for handling authentication is increased in relation to participants who only act in one IAM system. Assertions from one system must be understood, parsed, and verified correctly and then be transformed into assertions within another system. The two primary options for communication interactions for this are shown in Figure 4.29. If the user cannot directly provide the required attributes to the relying party, the relying party might suggest using a TGW by redirecting the user there. The choice of TGW can be made depending on the relying party's capabilities and the type of the user's authentication system.

In the first case, shown in Figure 4.27, the RP does not support SSI, but the user has SSI credentials they would like to use. This case requires the TGW to be set up with the RP to use another IAM protocol, e. g., SAML or OIDC. The TGW can then verify the credentials provided by the user and authenticate the user.

The reverse is more difficult. Suppose the user can only supply a non-SSI authentication method. In that case, the TGW must be set up to act as an RP and perform the authentication with the user's identity provider. As can be seen in Figure 4.29, this involves more communication and redirections between the TGW and the IdP. After the TGW has received the necessary authentication information, it creates an SSI credential and passes it to the user. This credential can then be used directly at the original RP.

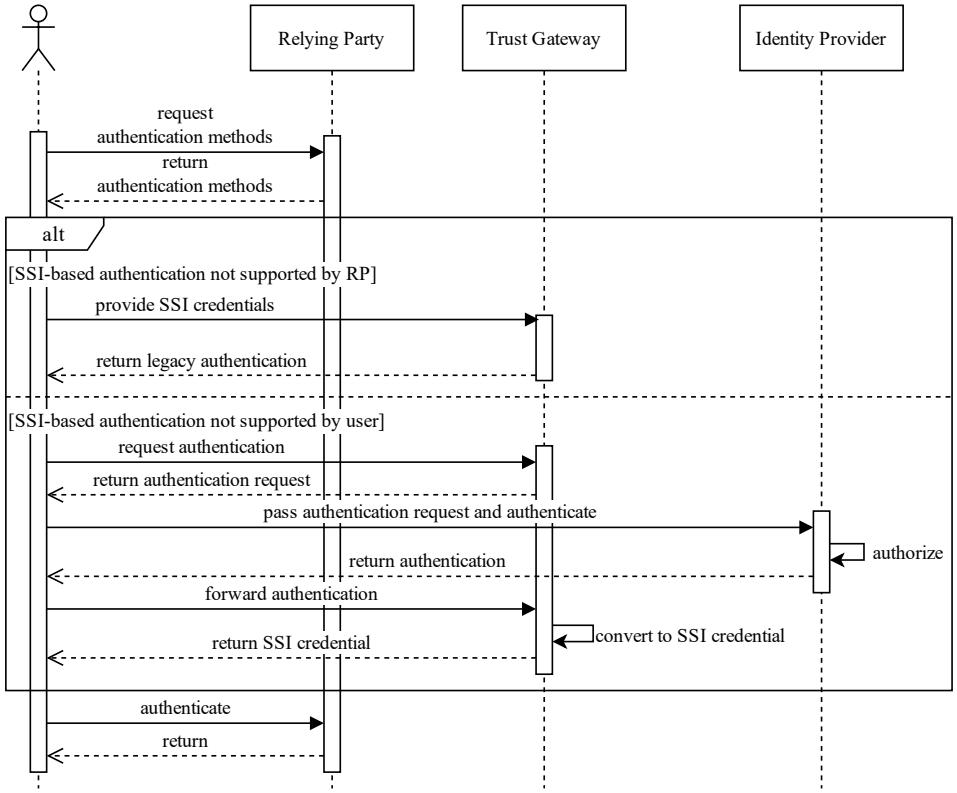

Figure 4.29: The TGW's communication model

Figure 4.29 shows an abstract method for implementing a TGW that should be possible to implement with any IAM protocol. The tricky part of this is designing the interactions for specific protocol combinations in a way that discloses as few as possible details to the TGW. This is not possible with RPs that use non-SSI techniques, as centralized systems cannot be routed via a TGW, and federated systems usually have defined communication paths between IdP and RP. However, an SSI-based RP could use a VC that is designed to allow the TGW to translate an assertion from a non-SSI system to a VC without breaking the end-to-end encryption and signature. This requires the IdP to know about the RP as specified by the IAM protocol. For example, with SAML, the RP must be included in the IdP's metadata, and the RP must know the IdP's public keys. The TGW can then issue a SAML request on behalf of the RP and transparently convert the SAML assertion by the IdP into a VC for the user. This method can similarly work for other protocols, such as OIDC and SAML, as shown in Section 3.2.

Other factors to consider after the TGW was used to perform a conversion are that, as the connecting part, the TGW has to ensure that changes from one system affecting the other are communicated and acted upon. Examples of critical information that needs to be propagated are security incidents, detected misuse, and revocations.

4.3.7.4 Functional Model

Functional areas of a TGW are similar to those of RPs, issuers, and CLSs. On one side, the input assertions must be processed and verified; on the other, valid credentials must be returned to the user.

If any irregularities or faults arise in the process, the TGW must be able to identify and direct the user to the right point of contact. Any problems might originate at the TGW, the RP, the issuer, or the user. Localizing them can be a major struggle. Consequently, good communication between those parties and the user is required as part of a proper incident response process.

Similarly to the CLS configuration, changes in the connected providers or the TGW may severely affect their service. A protocol for planning, testing, and performing configuration changes is important. Depending on the entity initiating a change, it must be communicated to the other participants, who need sufficient time to adapt.

As a service provider for the RP, issuer, and users, the TGW has many choices to charge for their service. Due to the close relationship and organizational ties with the RP, a business relationship for accounting purposes is also conceivable. Charging the users would probably only be accepted in very few situations, as they would usually already be charged for using the RP's service, and charging multiple times for using "the same" service would lead to a loss of users. The issuer has a similar incentive as the RP to expand its user base and prolong the relevance of its service, especially if SSI is indeed the future of IAM.

The performance of a TGW can be measured by the number of RPs and issuers connected through the gateway. Additionally, the service's number of users can provide information about its relevance and ease of use. The usage numbers, however, will be significantly lower for a TGW which can issue VCs, which it usually only does once or rarely per user, and a TGW that is required for each authentication.

4.3.7.5 Security Management

The integrity of the TGW's systems is essential to be considered trustworthy. If an entity could get VCs or authentication from the TGW, which are not based on verified and correct attributes, the credibility of the TGW's assertions is destroyed.

Similarly, the confidentiality of the conversions' contents is essential for users to trust and use the service. Even small violations of confidentiality can result in sustained mistrust and reluctance to use the service.

Availability is the least important of the classic security management goals. If the RP is SSI-capable, users should only access the TGW once to convert their attributes. If the TGW is required for every authentication with an RP via a particular IAM system and if those authentications make up a significant portion of the RP's users, the availability of the TGW is increasingly essential.

As the TGW is both issuer and RP, it should adhere to the security management provisions of both. The issuer's key security management points are described in Section 4.3.5.5, and the RP's in Section 4.3.4.5.

4.3.7.6 Data Protection Management

Depending on the application's goal and scope of the TGW, it might encounter a lot of users' PII. In this case, proper protections must be established. This includes informing the users about gathered data, especially if it is stored, stating the reason for the storage, and describing options to delete the data and the consequences of doing so. It should not be necessary for the TGW to pass the gathered data to other third party services for cross-checking. If such needs arise, they should be handled within the SSI system, where the user is actively asked to get the needed confirmation from the third party via a VC.

4.3.7.7 Conclusion

A TGW is required to kick-start an SSI system and increase the number of available VCs, making it more useful and attractive for users and RPs. Even fully established SSI systems can profit from TGWs as a means of transferring VCs from one SSI system to another. In this constellation, they are an essential part of the SSI paradigm allowing users to keep control of their data and quickly move between SSI systems if necessary.

The position of a TGW in between two IAM networks poses unique challenges in trying to keep both sides synchronized. This combination of general importance and tricky execution necessitates a closer description of TGWs.

4.3.8 Community

The community is responsible for choosing a path when adapting to new technological changes, modified requirements or new use cases, or new threats to the system. All the components listed in this section can only work together towards the common goal of providing a user-friendly and rich ecosystem for high-quality SSI if the community or leadership of the organization managing this ecosystem is acting in each participant's interest. The actual amount of influence granted to an organization managing the SSI system or individual community members can vary between systems. Using one universal distributed ledger for an SSI system is risky, as some changes may lead to a fractured community. Examples of this have been observed for both major blockchains. First, the split between Bitcoin and Bitcoin Cash or Bitcoin Gold [187] and the fork of Ethereum and Ethereum Classic [187] were due to fundamental differences about how to progress in the future within the respective communities. In contrast to those examples, the value of an SSI system is not purely measured through transactions and unspent outputs. Instead, the users' VCs are only loosely attached to the blockchain and could retain their usability even without it.

The SSI space is currently highly dynamic. Not only do enthusiasts and organizations compete for recognition, but governments have also shown interest in the technology, as we have collected in [143].

The main idea is to model the organization and community for an SSI system after the Internet's example. The Internet is an inherently diverse and distributed system, which faces a multitude of technological and organizational challenges, but still is extremely successful. As with a successful distributed ledger, parting from the Internet to build a better system is very unlikely to succeed.

The primary function of the community is to structure and consolidate efforts made by individual community members into a consistent bigger picture. As part of this effort, the main tasks are:

- **Communication** inside and outside the community, attracting and onboarding new members, sharing the community's vision, and encouraging participation.
- **Standardization and documentation** of used techniques and protocols together with vendors and developers to build a stable basis for SSI, which is easy to adopt and extend.
- **Security management** to define and implement processes to react to security incidents affecting the whole SSI system.
- **Interoperability** to keep track of other developments for IAM and SSI, in particular.

4.3.8.1 Information Model

A community managing an SSI system is usually broken into multiple parts that attend to different aspects of the system. Technical discussions can be held in working groups tackling a specific problem and providing feedback to a technical committee. Similarly, organizational topics can be covered by other committees. The exchange and discussion are mostly done publicly to allow any interested party to participate. Users of the system do not necessarily have to be active in the community but need to be informed of important news and changes. An overview of how a community could assemble is shown in Figure 4.30. However, communities may be structured differently and still be effective and efficient.

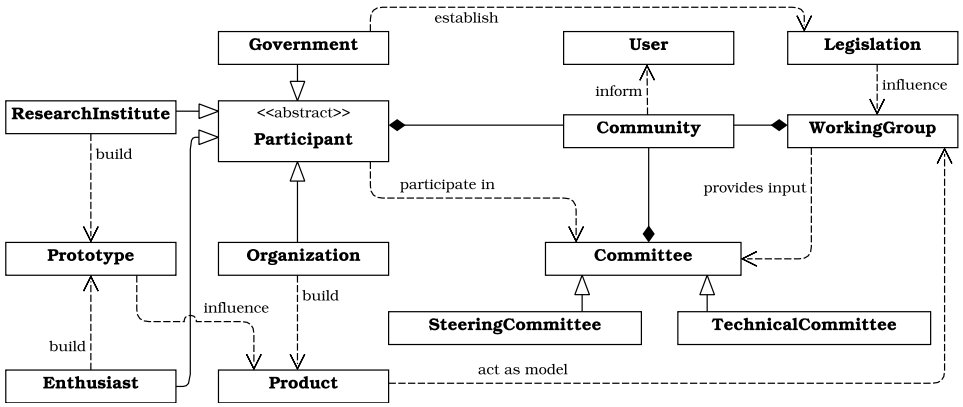

Figure 4.30: The community's information model

The community shown in Figure 4.30 comprises three main components: participants, committees, and working groups. Participants can be classified as individual or loose groups of enthusiasts, for-profit or non-profit organizations, or government employees or agencies. Each of those brings different interests, skills, time, and money into a community. The involvement of government parties is especially interesting, as they might be able to influence or establish legislation that could affect DLT and SSI systems. Enthusiasts, research institutions, and organizations bring expertise for developing and evaluating prototypes and actual products. Those can be part of a feedback loop that affects the development of working groups, which provide input to the committees that make the actual decisions.

4.3.8.2 Organization Model

The organization of a community for a large ecosystem is developed throughout the maturing process of the SSI system. The Internet, as arguably the largest distributed system in use today, is controlled by several governing bodies: *Internet Corporation for Assigned Names and Numbers* (ICANN), *Internet Engineering Task Force* (IETF), *Organization for the Advancement of Structured Information Standards* (OASIS), W3C and *Internet Governance Forum* (IGF) whose respective areas of control have also shifted through the Internet's development. While certainly smaller today, the potential for growth of an international SSI system might be to the scale of the Internet. The community ties to the other components primarily via the provided documentation, as described for the components in the previous sections and shown in Figure 4.31. Within the community, the documentation is built by the community's governance framework and standardization efforts.

Each of those actors has different expectations and means to influence and advance the system. As a result, each actor can take on a number of different roles. The distribution of those roles needs to be established by the organization.

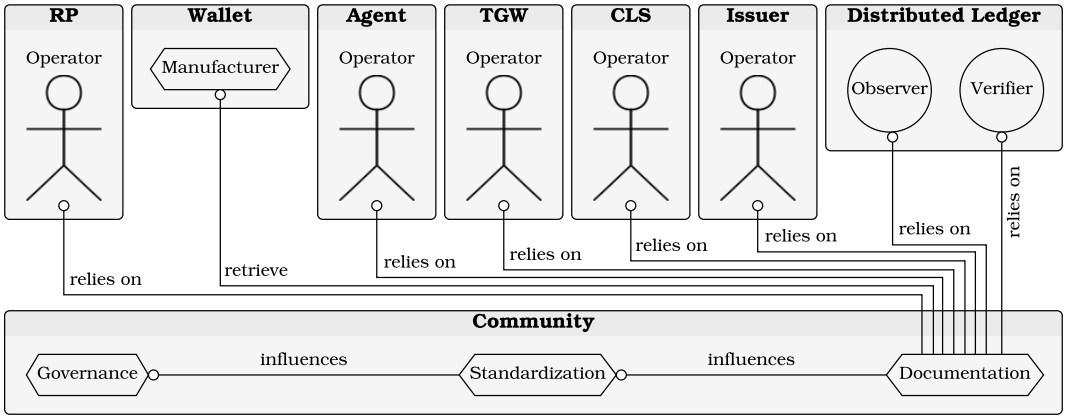

Figure 4.31: An exemplary community's organizational model

4.3.8.3 Communication Model

Communication within a larger organization or community is always challenging. Participants join from all over the world, speaking different languages, living in different time zones, and so on. Except for a few important meetings and conventions, most exchanges are done asynchronously via mailing lists.

4.3.8.4 Functional Model

Regardless of the community not being a technical system, it should perform its functions orderly. As such, it needs clearly defined processes and guidelines for handling any arising faults. To detect faults that are not reported, a monitoring system is required to detect irregularities and help debugging known problems. In a distributed system, such monitoring is challenging to implement, as no centralized instance is participating in the individual authentication processes, and it is not in the users' interest to report meta-information about their transactions. The number of systems and expectedly differing software implementations and versions further complicate finding faults. As a result, those individual issues must be primarily handled by the affected parties, and the community as a whole must focus on the general protocol and DLT infrastructure.

Similarly, the configuration of systems cannot be dictated by the community for all participants. A reference implementation might provide guidelines and hints for setting up other systems, however. This reference implementation should be used as a testing target by others who want to test their setups to ensure interoperability of the systems and configurations. Detailed documentation of the reference system (in multiple languages) is required to allow others to perform their evaluations. Feedback from those tests can refine the reference implementation and protocols without exposing actual users to those tests.

A long-term successful project also needs funding. Funding concepts range from government-subsidized models, popular for eID systems, over membership fees, to transaction-based systems. Most free systems are built on top of existing blockchains, like Bitcoin or Ethereum, and utilize the fact that the blockchain rewards miners. Which solution is preferable for which community cannot be determined in an abstract concept but must be discussed and chosen by the community.

4.3.8.5 Security Management

At this large scale, the organization of a system cannot effectively respond to security incidents. The security incident response is, therefore, part of each individual participant's duties. However, the organization should prevent security incidents by focusing on security by design and security by default approaches when deciding about supported technologies and promoting specific implementations.

Security incidents not isolated to specific participants may require coordination through the greater community. To facilitate the disclosure and handling of security incidents between many organizations, the Sirtfi framework provides guidance, as described in Section 4.3.5.5. At a community level, the biggest threat is a large-scale disruption of the availability or integrity of SSI services. Such a security incident probably affects the distributed ledger, either corrupting it and making it unusable or injecting false information. Confidentiality is less of a concern at the community level, as most parts of the SSI system are openly readable.

4.3.8.6 Data Protection Management

Like security management, data protection management is the responsibility of the individual participants. The organization running the SSI community can only support the individual efforts by providing documentation, best practices, and the possibility to communicate issues to other participants. Government agencies should be informed of serious data protection incidents in case of an incident and depending on legislation. They can then help estimate the impact and plan to inform affected users.

Data protection becomes even more important in an international setting as individual countries demand access to users' data. The SSI system should be designed so that the SSI organization cannot weaken the security of the individual connections between users, AAs, and RPs.

4.3.8.7 Conclusion

The community driving and using the SSI system is an integral part of the system itself. This necessitates its inclusion into the core component selection for the SSI architecture. As with any larger system, many aspects will have to be developed and refined over time. However, quite a few lessons learned can be transferred from larger FIM federations and general IT service management (ITSM) best practices.

4.3.9 Component Dependencies

The components described in Section 4.3 interact to provide a functioning SSI system. During the description of the components, some aspects could not be predetermined. However, decisions at one component can also affect others. To assess those connections, the dependencies between the components are explored in this section. As dependencies might not only exist within the SSI system, another generic entity "External" is introduced. External components or influences might be legislation or regulation for specific sectors. Figure 4.32 provides a rough overview of the resulting number of dependencies between the components. The width of the connecting arrows indicates the number of dependencies.

The dependencies themselves are explored in detail in the following sections. Each dependency is identified by an ID of form Dx, where x is the number of the dependency. Those IDs can be used to track repeated dependencies at different components.

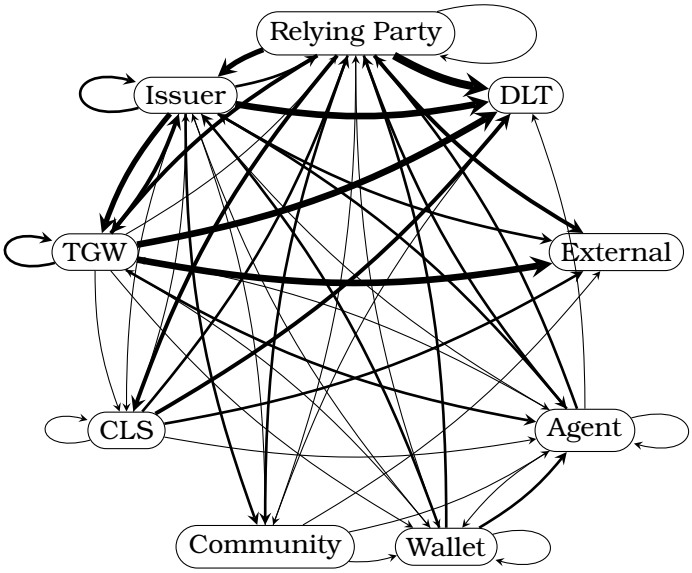

Figure 4.32: Overview of the components' dependencies

4.3.9.1 Issuer

The SSI system is useless without any issuers, so its dependencies are explored first. The information model described in Section 4.3.5.1 shows that the issuer needs to implement the interface `Comms`. This interface is also implemented by the wallet, agent, and RP, as visible in Figure 4.33. The TGW, by extension, a wallet, also indirectly implements this interface. This connection shows dependency D1 concerning a common protocol to exchange VCs with those three components.

In the issuer's information model, the issuer also uses a `TrustDB` to manage trust relationships with other components. The resulting dependency D2 mirrors this by showing a dependency on RPs, TGWs, as well as the distributed ledger, the community, and external entities.

The interface for data sources is not connected to a specific component and is ignored at this stage. It does not lead to any dependencies on other components. Instead, each issuer must individually manage its data sources.

Issuer and agent have a bidirectional dependency, as they rely on each other to issue VCs and make them accessible for the user. To support the issued VCs with technical and non-technical descriptions, the issuer relies on metadata managed through the distributed ledger. This DLT connection results in multiple direct dependencies, which will partly be recurring for other DLT-connected entities:

- A method for retrieving and verifying the genesis file to verify the integrity of subsequent states of the DLT. This is expressed in dependency D3, a straightforward dependency to the community, which must provide provisions to obtain the genesis file.

- Provisions for creating a public identity by issuing a unique identifier and attaching a metadata file. The dependency D4 shows that the issuer relies on the DLT to publish and manage their public identity documents.

- Retrieve VC schemata that others have published to reuse them instead of creating a new one. This dependency D5 has connections to any component which might publish VC schemata, including external entities. It also connects to the DLT, which stores and manages the schemata.
- Publish the keys of no longer valid VCs. If no suitable VC schema exists, dependency D6 describes how the issuer can publish their own VC schema. To publish a VC schema, the issuer only relies on the DLT.
- Publishing and retrieving VC schemata is not enough. The entities must agree on common schemata that can be used in general, in large groups, or at least in smaller federation-style communities. The corresponding dependency D7 affects any component of such a constellation.

If the issuer should have issued a no longer valid VC, the VC's signing keys are published as described in Section 4.3.5.3. As shown by dependency D8, the issuer only depends on the DLT to publish this information.

Last but not least, the issuer also relies on external parties, as the issuance of VCs might be regulated by laws or sector-specific regulations in some instances. Dependency D9 describes these external dependencies, which must also be considered and re-evaluated constantly.

4.3.9.2 Relying Parties

The RPs share some of the issuers' dependencies. They rely on the DLT to get the genesis file (D3), they may need to create a public identity to be discoverable (D4), they need to retrieve VC schema definitions (D5), and they possibly also need to publish their own VC schema definitions (D6).

Because of the VC localization design described in Section 4.3.6.3, which performs VC localization on the RP's side, the RP has a direct dependency on the CLS. This is also shown in the RP's information model depicted in Figure 4.23.

4.3.9.3 Agents

The agents are working on behalf of the users' wallets and must perform preliminary validation and filtering before advancing SSI messages to the wallets. This procedure is described in Section 4.3.3.3. As verifying requests and offers requires access to the DLT, the agents depend on following the DLT by obtaining the genesis file (D3). To exchange and understand the VCs, a common exchange protocol (D1) and VC schema definition (D7) are also required. In the case of the agent, the common exchange protocol includes not only the communication to issuers and RPs. It also includes communication with other agents, the W2A protocol described in Section 4.3.2.3, and TGWs.

4.3.9.4 Wallets

Wallets are the components that users directly interact with and access the SSI system through. Many operations can be done directly between wallets, but some advanced features – especially those involving accessing the DLT – depend on the wallet communicating with an agent. This communication is described in Section 4.3.2.3. To facilitate this communication with other wallets and agents, the wallet depends on the W2W and W2A protocols combined in dependency D1.

Retrieving VCs from an issuer and presenting VCs to an RP can be done without interacting with the DLT. The wallet never needs to contact the DLT and can perform its basic SSI operations always and anywhere without connectivity to the Internet. As the issuer can also be a TGW, the wallet also depends on those.

4.3.9.5 Trust Gateway

As the TGW can be both an issuer and an RP, it has the union of both of their dependencies and then some. The dependencies inherited by the issuer and RP are the connection to the DLT for retrieving the genesis file (D3), registering a public identity (D4), and retrieving VCs schemata (D5). As part of the TGW's operation, it must constantly check whether the credentials it forwarded have been revoked in the source IAM system and invalidate the revoked VCs in its wallet. As issuer and RP, the TGW must adhere to a common VC exchange protocol (D1) for issuing and verifying credentials. The definitions of those VCs need to be agreed upon with issuers, RPs, agents, and the CLS (D7).

Besides the general external dependency on laws and regulations, the TGW has some unique dependencies if it can bridge non-SSI systems to the SSI network (D9).

4.3.9.6 Credential Localization Services

The CLS must connect to the DLT to keep up-to-date with the defined VC schemata. To do so, it needs to retrieve the genesis file (D3), follow the DLT, register a public identity (D4), and understand the used VC schema definitions (D5, D7). Besides, the CLS only needs to supply localization rules to the RPs that allow changing one VC into another, as described in Section 4.3.6.3. External dependencies to laws and regulations might arise if the CLS has to keep certain guarantees for the provided rules (D9).

4.3.9.7 Distributed Ledgers

As a central point in the SSI system, the DLT is depended upon by many of the systems' components. The ledger itself has few dependencies on others, however. This is because the ledger is a networked program that should only perform a limited set of pre-programmed operations. It must be operated and maintained (D10), as part of the community's tasks.

As DLT technology is increasingly regulated and as the basis for potentially highly trusted digital identities, the community is also responsible for any necessary provisions and adaptions.

4.3.9.8 Community

The community – as the name suggests – is reliant on its participants. As a result, it is heavily dependent on active participation by members of all other components (D11), especially by the operators and developers of issuers, RPs, agents, and wallets. Additionally, external inputs can also be of value to prevent the community from focusing too much on itself.

4.3.9.9 Summary

The dependencies found in the previous sections show which components rely on which other components. Figure 4.32 provides an overview of the number of dependencies between the components. The strong dependency of many components on the DLT emphasizes its importance. Those dependencies also highlight that the DLT must be as stable as possible, as any changes that would require updates or modification of relying components could break the function of the whole system.

The components' dependencies and their directions are summarized in Table 4.1. Each row shows the sum of all dependencies for any component and a checkmark to indicate individual relations between components and components.

Table 4.1: Summary of the components' dependencies

ID	Dependencies by components	*Issuer*	*Verifier*	*Agent*	*Wallet*	*TGW*	*CLS*	*DLT*	*Community*	*External*
	Issuer	2	3	1	1	4	1	4	4	4
D1	Common VC exchange protocol			✓	✓	✓				
D2	Dynamic trust relations through federations		✓			✓		✓	✓	✓
D3	Retrieve Genesis file								✓	
D4	Register public identity							✓		
D5	Retrieve VC schema	✓	✓			✓		✓	✓	✓
D6	Publish VC schema							✓		
D7	Common VC schema definition	✓	✓			✓	✓		✓	✓
D9	Laws & Regulation									✓
	Verifier	3	1	2	1	3	2	5	2	3
D1	Common VC exchange protocol			✓	✓	✓				
D2	Dynamic trust relations through federations	✓				✓		✓	✓	✓
D3	Retrieve Genesis file							✓		
D4	Register public identity							✓		
D5	Retrieve VC schema	✓	✓			✓	✓	✓	✓	✓
D6	Publish VC schema							✓		
D7	Common VC schema definition	✓		✓			✓			
D9	Laws & Regulation									✓
	Agent	1	2	1	1	1	0	1	0	0
D1	Common VC exchange protocol	✓	✓	✓	✓	✓				
D3	Retrieve Genesis file							✓		
D7	Common VC schema definition	✓	✓							
	Wallet	2	2	1	1	1	0	0	0	0
D1	Common VC exchange protocol	✓	✓	✓	✓	✓				
D7	Common VC schema definition	✓	✓							
	Trust Gateway	3	2	2	1	2	1	4	0	4
D1	Common VC exchange protocol	✓	✓	✓	✓	✓				✓
D3	Retrieve Genesis file							✓		
D4	Register public identity							✓		
D5	Retrieve VC schema							✓		
D7	Common VC schema definition	✓	✓	✓			✓			✓
D9	Laws & Regulation									✓

Continued on next page

Table 4.1: Summary of the components' dependencies (Continued)

	Credential Localization Service	1	2	1	0	0	1	3	0	2
D3	Retrieve Genesis file							✓		
D4	Register public identity							✓		
D5	Retrieve VC schema							✓		
D7	Common VC schema definition	✓	✓	✓			✓			✓
D9	Laws & Regulation		✓							✓
	DLT	0	0	0	0	0	0	0	1	0
D10	Operation								✓	
	Community	1	1	1	1	0	0	0	0	1
D9	Laws & Regulation									✓
D11	Participation	✓	✓	✓	✓					

4.4 Reference Architecture

A complete reference architecture can be obtained by combining the individual components' descriptions from the previous section. To prevent repetitions, only new aspects concerning the combination of components and the systems as a whole are described.

In Section 4.3, all components are described using an information model. The individual information models contain repeated elements that are part of multiple components' information models. A merged "big picture" information model is generated using overlapping components in Figure 4.33. In this figure, each component is color-coded to show the element's originating component. For elements that are part of multiple components, the first component sets the element's color. Figure 4.33 shows that no single component is entirely independent of others, similar to the component dependency overview of Figure 4.32. When implementing an SSI system, this view can help estimate individual implementation decision's influence.

Comparable to the information model, the organizational model of individual components also contains some overlap. All direct relations of components are shown in each component's organizational model and are not repeated here. Similarly, the communication and functional models, as well as the security management and data protection considerations are not combined in the reference architecture. They are component specific, and a combined view adds no benefit.

4.5 Integration

Integrating the components described in Section 4.3 into a new or an existing IAM system requires careful planning. This section should provide an overview of aspects that need to be considered.

4.5.1 Starting a New SSI System from Scratch

From a technological point of view, starting from scratch is the easiest option when deploying a new SSI IAM system. There are no dependencies on legacy components or data that has to be migrated to the new system. However, this also provides the largest hurdle for starting a new IAM system, as everything has to be built anew, and many established systems might be easier to set up, use, and manage.

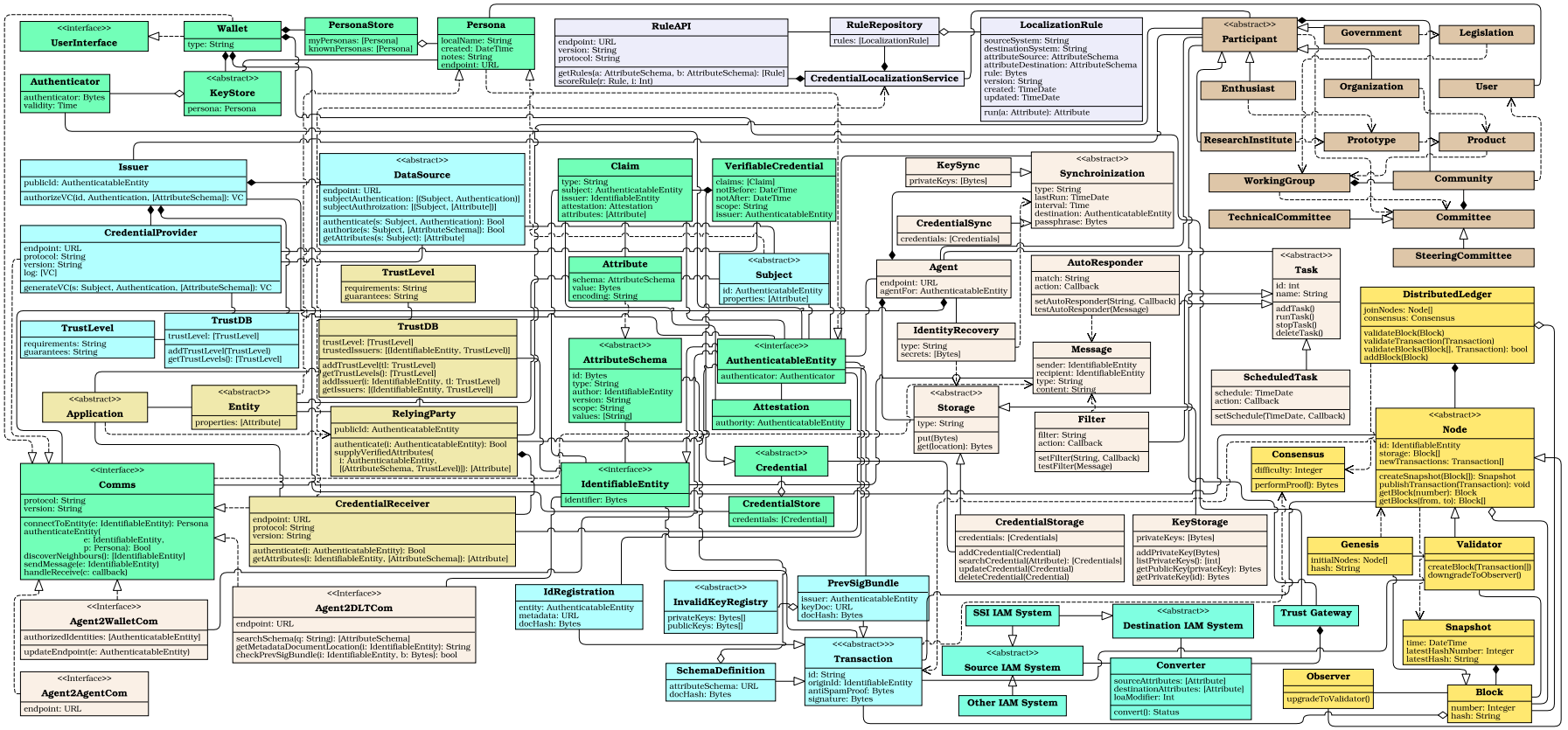

Figure 4.33: Complete overview of the information model. The colors indicate the components' origin: DLT gold, wallet light green, agent linen, RP pale gold, issuer aqua, CLS near white, TGW aquamarine, and community vanilla. Detailed information about the components can be found in the respective parts of Section 4.3.

This description of the integration ignores the TGW and CLS because they are not needed when setting up a new IAM system. They rather help to migrate data from older systems, which will be described in Section 4.5.2.

A precondition for building a decentralized IAM system is having an interested community of participants. Those participants must agree on the system's goals and technological and organizational parameters. As Section 4.3.9.9 shows, the component which most other components depend on is the DLT. Therefore, in the order of components to set up, the DLT takes first place. The wallet takes second place, as it is necessary to store the cryptographic keys for SSI operations securely. Third and fourth place go to the issuers and relying parties. It is advisable to have one issuer running before setting up a relying party, as the RP depends on the issuers' VCs and cannot be tested without them.

With those four components in place, user interaction via a wallet application must be tackled. Additionally, agents can be provisioned to streamline the whole system and improve its usability.

4.5.1.1 DLT Setup

When deciding about building a new SSI system, choosing the right DLT is essential. Once fixed, major changes to the DLT can hardly be implemented if the SSI system is running. The systems' fundamental characteristics are determined by the ledger type. Usually, different versions of DLT are distinguished by who can read the ledger and who can write to the ledger. The most common combinations of those two settings are public/permission-less ledgers (e. g., Bitcoin and Ethereum) or private/permissioned (e. g., Hyperledger-based). Another option is not to use any DLT system but to rely on a PKI-based system.

The primary use case for using DLT over PKI is to have a traceable history for any value written to the ledger. This is useful if disputes arise and decisions need to be justified. Additionally, a public/permission-less distributed ledger can guarantee that individual identities are actually self-sovereign. Any dependency on a permission system or CA increases the risk of being limited in publishing public identities and other metadata. At the same time, public/permission-less systems cannot fulfill the requirements of the GDPR and, as such, carry a risk for node operators [85, 174]. Figure 4.34 summarizes those options in a decision tree.

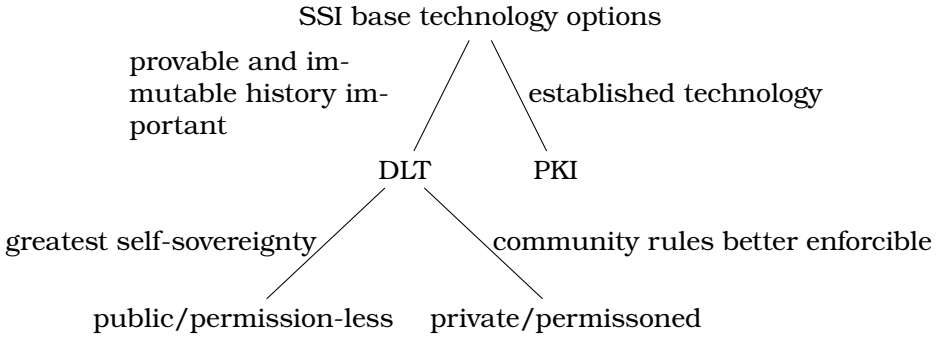

Figure 4.34: Decision tree for choosing the base technology for SSI

If a DLT-based approach is selected, a suitable consensus protocol needs to be found. For public/permission-less ledgers, the choices are primarily PoW and proof of stake (PoS). More efficient quorum-based protocols can be used for private/permissioned ledgers. In the case of a PKI-based system, the consensus

about transaction ordering needs to be solved by either using a hierarchical approach (i. e., domain name system (DNS)) or random identifiers with a low chance of collisions (i. e., PGP fingerprints).

The primary risk to the DLT system is an unauthorized modification that allows the issuance of forged VC or results in a DoS of entities using the system.

Regardless of the base technology used, the ledger's data must be stored by multiple nodes to ensure redundancy and availability. If DLT is used, the integrity of the data is straightforward to check. Starting from the genesis transaction, all further transactions can be checked for correctness. Any later modification of a transaction would be easy to detect.

The integrity of non-DLT systems is more difficult to guarantee and depends on the individual technologies used.

Confidentiality of the system's nodes is, except for a private/permissioned ledger and all private keys, not important. The data stored in those systems is usually designed to be available publicly and thus does not necessitate any special precautions.

In order to prevent PII from being stored on the DLT, transactions should be designed in a way that prevents accidental or deliberate storage of PII. This can be done by only allowing signatures, hashes, and URLs to be stored. The main content is then kept off-ledger at the URL's location, which can be removed and is not mirrored across all nodes. As a major drawback, this also limits the long-term availability of the data.

4.5.1.2 Wallet Setup

The wallet is essential to store cryptographic keys and perform a minimal set of SSI operations to connect to agents, issuers, and relying parties. It is highly dependent on the available hardware and operating system and its capabilities.

On most modern platforms that run full-fledged operating systems, there are provisions to store cryptographic keys in a specially protected environment. Some of those secure elements may also be hardware-based, providing even better protection against an attacker compromising a device or trying to retrieve the keys with physical access.

Backups of those keys are also important, as a complete loss of the private keys can render an identity useless. It is important that the backup process is as secure as the key storage to not introduce an easy attack vector.

- The most straightforward option is to export an **encrypted backup** file, which can be imported again if needed. This option might not be suitable if it is required to keep the keys protected by a hardware security module (HSM) at all times.
- Another solution would be to generate the base identity from a **physical copy** or **printout**. Entirely offline, this backup would be hard to steal and would not suffer from random disk failures.
- Another option is to have **no backup**. This might be necessary if an IoT device's keys are associated with non-modifiable hardware, i. e., a PUF, and this hardware is damaged. In this case, a repaired device will be assigned new keys, and the device's identity needs to be updated in the DDo.

4.5.1.3 Issuer Setup

The setup described in this section is intended for integrating a permanent issuing service. A self-issued credential by a user's agent might not require the agent to implement all the provisions described here. How those systems can be integrated is described in Section 4.5.1.5.

An issuer is composed of three main components:

- The **data source** is the repository that contains information that can be provided to users as a VC. The data source must be kept well secured, as it can contain sensitive PII, process data, or trade secrets.
- The **DLT connector** is required to connect, add, and verify transactions on the DLT. It is a component necessary for managing the public identity of the issuer, the repudiation of expired VCs, and the schemata of used VCs.
- User-facing the **issuing frontend** connects to agents and wallets to issue VCs to the users.

Those three components should be placed in separate network domains, with the data source being in a closed-off network and the DLT connector issuing frontend in separate DMZ networks connected to the Internet. A schematic of this setup and the firewalled connections between the network segments is depicted in Figure 4.35.

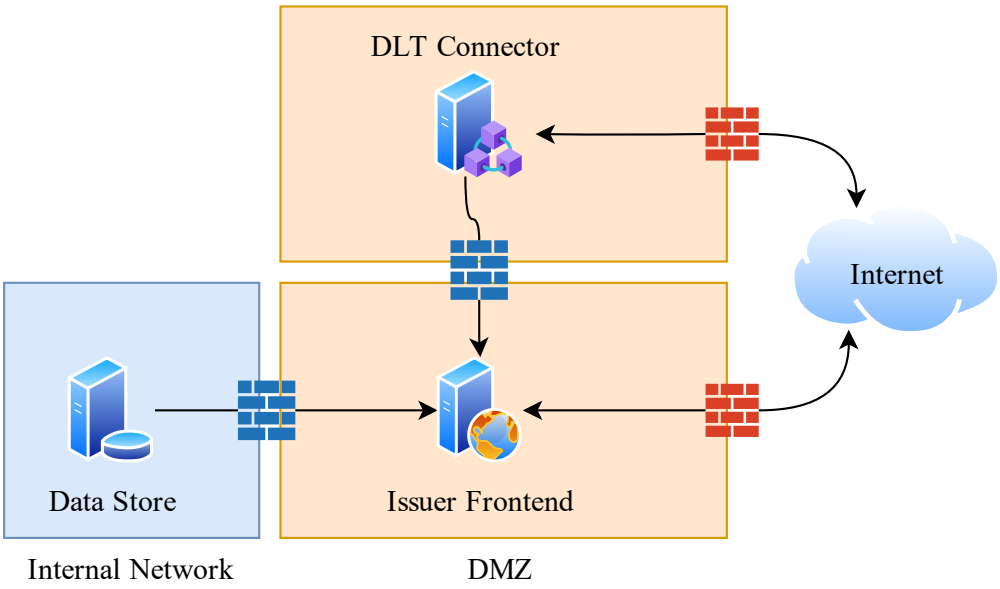

Figure 4.35: Network integration of the issuer

To establish a base for technical trust with other system entities, the issuer must publish its unique identifier and associated metadata to the distributed ledger. The metadata must contain at least the issuer's public key(s), but additional information like a description and a URL to the issuer's website is usually helpful. Additionally, the combination of identifier, metadata, and time stamp must be signed and added to the distributed ledger.

With the identifier and metadata in place, the issuer can form organizational trust relations with other participants. This trust can be divided into the following categories:

- **Institutional trust** is provided by well-known institutions, for example, governments or corporations. They can issue VCs generally recognized as well-verified and trusted without direct contractual provisions (like physical ID cards). The biggest challenge to an issuer with this trust model is preventing impostors from tricking services into accepting VCs. EU government agencies might use qualified certificates [157] for this task.
- Most issuers will not have the benefit of being basically universally trusted. Those can join together with peers in their field in a **federation**. This requires them to exchange legal details over data quality and usage but provides a robust framework.
- Both of the previous options require substantial organizational management. As a hassle-free option, issuers can issue VCs using a **best-effort** approach. Those VCs might only be useful within an organization or group of friends but are fundamentally the same credential as the others.

4.5.1.4 Relying Party Setup

As with the issuer described in the previous section, this section targets permanently set up relying parties. How an agent can also act as a relying party and check VCs is described in Section 4.5.1.5.

A relying party can either require authenticated information about an entity or ensure it had previous contact with it. In both cases, the IAM component is placed in front of the application, as shown in Figure 4.36. The service's database is specially protected within the internal network, and the DLT connector is placed in another network, all separated by firewalls.

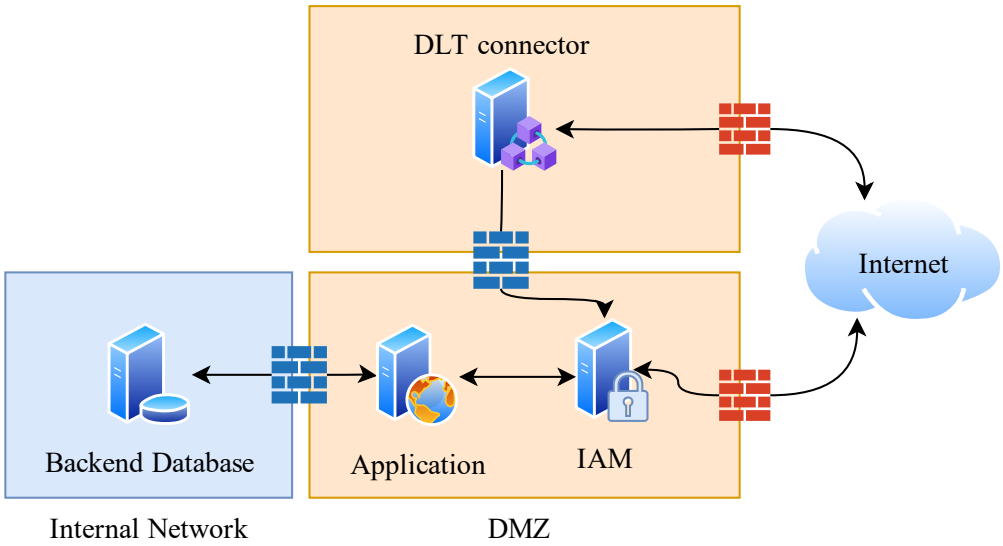

Figure 4.36: Network integration of the relying party

The RP also has to register a public unique identifier if it wants to be recognizable by the other participants. This allows it to prove its identity to others in a mutually authenticated exchange and prevents impersonations. The technical trust is established the same way as for the issuer. An identifier and associated metadata are published to the DLT, where it can be retrieved from, and changes to it be tracked by the users' agents.

Organizational trust for the RP depends on the kind of issuers the RP wants to interact with and the guarantees it needs to provide for its services. An RP might just trust the VCs issued by individual known institutions or must join a federation. Joining a federation involves explicit contracts with the other participants. The use of issuers that issue VC according to the best-effort method is rather unlikely, with the exception that the issuer is internally known.

4.5.1.5 Agent Setup

Agents play an important role within the SSI system. They connect users' wallets persistently to the system but are usually not run directly by the users. As such, the users need to trust their agents with handling connections to issuers and RPs.

An agent, as described in Section 4.3.3, can support various features. In its simplest form, the agent only passes messages between wallets and issuers or RPs. More advanced features like automated tasks, synchronization of VCs and keys, or identity recovery add to the effort of running an agent. Similarly, the target audience can shift depending on features but also because of strategic decisions (e. g., is the agent available for free or limited to specific users). Determining the desired feature set and scope is, therefore, the first necessary step.

Depending on the agent's features, the setup can be handled like a web application. If it can be used to synchronize confidential or private data, however, the setup needs to consider adequate protections. This can be achieved by the separation of data stores and ensuring all data is stored with sufficient encryption. Ideally, private data, like VCs, are encrypted client-side and never accessible to the agent. If the agent requires access due to some automatization process, only the strictly necessary VCs need to be stored. Similarly, users' keys should be protected to the same level as an online password safe would provide.

4.5.2 Migrating IAM Systems to SSI

The other components presented in Section 4.3 are added if an existing IAM system is migrated to SSI. In particular, the CLS and the TGW make using previous identities and attributes with SSI easier.

4.5.2.1 Credential Localization Service Setup

The CLS's main challenge is providing reliable and trustworthy localization rules. This requires it to be securely set up and managed and being organizationally well-connected to its partners. The most important goal of IT security for the CLS is the protection of the rules' integrity. Integrity protection must be achieved at the CLS itself and while transferring the rules to the customers.

Availability and confidentiality also need to be considered as they allow the CLS to operate its business and keep the trade secrets it wants to sell. The setup must keep those goals in mind.

4.5.2.2 Trust Gateway

The TGW is even more challenging regarding integrity and confidentiality requirements than the CLS. It needs to process and convert data that might consist of PII. As described in the TGW's description in Section 4.3.7, the TGW is a mix of a cloud-based wallet, an issuer, and an RP. As a result, the TGW must include the recommendations for those components described in Sections 4.5.1.2, 4.5.1.3, and 4.5.1.4.

4.5.3 Integrating SSI in the Scenarios

The use cases selected for integrating the SSI concept are from the three main scenarios presented in Section 2: Web Apps, IoT, and eID. Each of those scenarios is developed with different characteristics in mind.

The web app scenario is modeled after current OIDC-based IAM systems, which are primarily operated by large corporations and social networks. While those larger corporations are the main issuers of credentials in such a system, they benefit by allowing any other web-based application to use the VCs issued by them. This allows the system to be public and allows anybody to participate. With web services as the main issuers and consumers of VCs, the system is built around web technology and features a browser-based wallet. The prevalent method for proving website authenticity and keeping transmitted data confidential is using TLS certificates based on a PKI. The SSI system can be integrated into this setting without the need for new infrastructure in the form of a dedicated DLT.

For the IoT scenario, the premise is a closed system between manufacturers, operators, and data processors. The data handled here is less PII and more generic sensor readings and assertions of manufacturing processes. This scenario uses private/permissioned DLT technology as a basis for SSI. The wallets are primarily designed for embedded systems. To enable a large number of businesses to participate, this system also integrates CLSs and TGWs.

The last scenario looks at highly personal and highly reliable identity information used for e-government and eID. It is also based on DLT but utilizes a public/permissioned ledger to provide transparency while keeping the ledger efficient. As the data managed in this scenario is extremely sensitive, the wallet should use trusted compute modules or HSMs. To migrate data from existing systems, certified portals are used as TGWs. This scenario should mirror the plans set forth by the EU with the new revision of eIDAS [150].

Table 4.2 summarizes the different aspects selected for those scenarios from the components' perspectives. A goal of the chosen characteristics is to cover a wide variety of realistic and possible cases for the application of SSI.

Table 4.2: Overview of the different aspects of integrating SSI into the selected scenarios

	Scenario		
Perspective	Web Apps	IoT	eID
DLT	PKI	Private/Permissioned	Public/Permissioned
Issuer	Corporations	Manufacturer	Government
Relying Party	Web Applications	Data Processors	Government Agencies
Wallet	Browser	Embedded	App
Agent	Web Service	Cloud Service	Certified Provider
CLS	none	Service Provider	none
TGW	none	Service Provider	Certified Portal
Community	Public	Sector Specific	Public-private Partnership

4.5.3.1 Web Apps

As this concept is reusing the existing PKI of the web, it does not need a new DLT backend. The main problem with re-using the web's PKI for issuing SSI VC is that the certificates used by websites tend to be valid for short periods. For regular web browsing, the switch to a new X.509 certificate would result in the VCs no longer being able to be verified. Especially for short-term certificates, like the 90 day validity period of LetsEncrypt's certificates [1], this poses a major issue. This forces the relying party to first figure out which certificate was in-use and valid at the time the VC was issued and then check its validity against this certificate. Using certificate transparency (CT) logs, which share similarities with DLTs, is possible. Revoking certificates reliably is a difficult problem and topic of multiple standards like CRLs [39] and the OCSP [163]. However, revocation is necessary, and using short-lived certificates limits the impact of not being able to reliably communicate revocations to end devices.

For SSI, revocation of the issuing keys is a different problem, described in Section 4.3.5.4. The issuing keys are usually long-lived, and the public keys can be published by the issuer at a well-known location. Their authenticity and integrity is protected by the short-lived X.509 certificate used to secure the website serving those. As a result, the lookup of the certificate requires an Internet connection, but the use case for web pages requires that anyway.

The issuer's VCs are stored by the users in wallets, which their web browser provides. Those work similarly to password safes, which are already part of most major browsers or operating systems.

The relying parties can request VCs, and the user's wallet helps select the right credential and provides information about what data is shared with whom. To validate the received VC, the RP must retrieve the verification keys from the issuer or have them already cached. This does leak some metadata which could be prevented with fully decentralized storage for those keys, but the metadata does not include the user's identity.

Agents are used to relay messages to the users in their absence. Those messages can notify the user of revoked or renewed VC or contain general messages.

4.5.3.2 IoT

IoT applications can utilize private/permissioned DLTs to manage devices and their digital identities. While the ledger should never contain confidential or private information, the content of metadata like credential schemata and public identifiers can already give away details about participants, project goals, and timelines. In this case, a private ledger can offer some protection for participants of confidential projects by preventing insights of others into this metadata.

The architecture for using private/permissioned DLT is also already in use in some industry projects where supply chain tracking applications are using DLT [45, 136]. An example of this was Maersk and IBM building a tracking platform for tracking shipping containers [183], which has since been retired [36]. Similar frameworks for blockchain integration in supply chains do not use SSI and instead rely on traditional IAM methods [113]. However, the expertise gathered in deploying and running DLT-based systems provides a good foundation to integrate SSI.

Issuers and relying parties in this scenario are usually web services. The entities which need to be identified and associated with attributes are IoT products. An important part of the system is setting up and deploying the IoT devices. For this setup to scale to thousands of devices, the manufacturer must be involved and prepare the devices as part of the manufacturing process. The devices are assigned

their machine identity with the help of TPMs or PUF technology. They are also issued a VC, which guarantees they are an original product of the manufacturer. To include such a device in an operation, the operator can, as part of the procurement, issue their own VC asserting that the device has been taken into operation. Using both VCs, the device can participate in the operator's system.

Products without any processing power and digital storage (i. e., tires, wood planks, or food) are not included as entities in this scenario. Their progress within the supply chain can, however, be tracked through identifiers by the machines that process them. The IoT device's messages can be verified by the VC from the manufacturer, stating its capabilities, and the operator's VC, describing its position and use case.

As the managed entities primarily are IoT devices, this scenario poses the challenge that the device itself cannot act without an operator and is owned by a person or legal entity. As a result, the entity's identity must be able to be managed by the operator on behalf of the owner. This management functionality must be built into the IoT device's wallet. A hierarchy of permission delegations from the device's owner to the operator can be used to implement this management functionality through SSI itself. If the device is to be serviced by a subcontractor, the operator can allow the subcontractor to manage the device by issuing a corresponding VCs. The collection of all VCs of an IoT device and its messages can be used to model a digital twin.

Because many IoT devices are not directly connected to the Internet, they may only be reachable through a gateway. If this gateway is reliably connected to the Internet, it can forward the devices' messages to a cloud-based agent. With different projects being run on a private DLT system, it is especially important to have a CLS to localize VC between projects. As the information from one project to another is not directly visible, it is harder to re-use existing standards, and slightly differing schemata will evolve. The same applies to TGWs, as the closed architecture does not immediately encourage using one single DLT. TGWs have to fill the gaps.

4.5.3.3 eID

The private/permissioned DLTs suggested in the previous section for IoT are less suitable for eID. To foster transparency and provide the highest possible interoperability, everybody should be able to read the schema definitions and public identities defined on the ledger. As shown in [143], many approaches to mobile government IDs and eIDs exist worldwide. Some of them already rely on SSI technology. However, it would add much value to eIDs, in general, to use those identities even across borders.

Within the EU [157] already provides provisions for a common exchange format for eID and rules and regulations for building and running eID services. The planned revision of eIDAS [150] already points to using an EU-wide system similar to SSI. In this system, identity issuers are primarily government organizations or private companies contracted by governments. The number of issuers is, therefore, relatively low. To issue reliable VCs for the eID scenario, those issuers must comply with strict security and data protection rules. The certified adherence to those rules can be expressed through a VC that guarantees a certain LoA.

Relying parties can be government as well as private services. The high quality and assurance of the identity information available in this system require special precautions to prevent misuse or identity theft. Users must, therefore, always be sure which service provider they interact with, and impersonations must be prevented as effectively as possible. This kind of protection also must carry over to the wallets and devices where the wallets are stored. Storage of the wallet's keys in a TPM is a possible solution for this.

The attribute schemata for use in an eID scenario can be implemented easily, as laws govern the initial number of attributes. This eliminates the guesswork for all participating parties, but some differences between countries can still exist. To translate specific attributes from one language to another or to adapt attributes to local formats, the use of CLSs is essential. Combined with TGWs, which can import identity data from existing eID systems, this allows for a smoother transition to using SSI.

Already established trust services, like a PKI system, can be integrated into an eID system. The PKI can be used as trust anchors for identities on the SSI system, as all participants already know how to validate signatures from the PKI, and the entities have already been carefully vetted before being issued their PKI certificates.

4.6 Assessment

The assessment will revisit the requirements posed in Chapter 2 and listed in Section 2.6. Requirements are considered in order of their importance, as determined in their respective descriptions and grouped according to their category. The source scenario is not factored into the assessment.

Each requirement can be determined to be either:

- *Completely fulfilled*, if all aspects of the requirement are met,
- *fulfilled*, if the most important aspects are met,
- *partially fulfilled*, if some aspects could not be met,
- *not fulfilled*, if the concept cannot provide the required aspects, or
- *overruled*, if the requirement is determined to be infeasible to achieve but mitigating factors have been provided.

At the end of the chapter, Table 4.3 visualizes the results similarly to the comparison in Section 3.6.7 and compares them to the results from exploring the state-of-the-art with SSI from Table 3.2.

4.6.1 Essential Requirements

- **Identification** (SAT3): Any entity within the SSI system can be identified by its unique identifier. This requirement is *completely fulfilled*.
- **Authentication** (SAT1): Any entity can be authenticated using its unique identifier and the public-private key pair associated with its identity. This requirement is *completely fulfilled*.
- **Authorization** (SAT2): By utilizing VCs, attributes can be asserted to an entity. Those can be used in arbitrary complex authorization decisions by the RP. This requirement is *completely fulfilled*.
- **Identity provisioning** (SAT4): The protocol for establishing a connection between an entity, represented by the identity in its wallet, is shown in Section 4.3.2.3. This requirement is *completely fulfilled*.
- **Trust establishment** (SAT5): Through the use of DLT or PKI systems, as described in Section 4.3.1, all participants of the system can reliably verify assertions made by public entities. Establishing trust between all entities is not possible without further constraints of the system. This requirement is *fulfilled*.

- **Automated integration/registration** (SAT7): To support automated discovery and connection to entities in the vicinity, the discovery protocol of the used network link is used in combination with a common endpoint as described in Section 4.3.2.3. This requirement is *completely fulfilled*.

- **Message delivery services** (SAT9): SSI itself cannot be used to force the wallet to notify an entity if they received their message. As a result, this requirement is *overruled*. To implement this feature, where necessary, either the wallet or the agent, which is used, must be implemented with this functionality.

- **Credential establishment** (INF1): The first step of the protocol for interacting with other entities establishes how each entity can authenticate with the other using their identity credentials. This protocol, including mutual authentication and verification of VCs, is described in Section 4.3.2.3. This requirement is *completely fulfilled*.

- **Digital identification** (INF3): Each entity can be identified by its unique identifier, which can easily be printed as a QR code, sent via NFC, or communicated with similar technologies. This requirement is *completely fulfilled*.

- **Physical identification** (INF4): Physical identification of a digital identity depends on the capabilities of the devices in question and the data available. This requirement is *fulfilled*.

- **Neighbour discovery** (INF6): A protocol for discovering entities within a specific network or region is described in Section 4.3.2.3. This requirement is *completely fulfilled*.

- **Identity data set** (INF10): Due to the flexible application of VCs, any set of attribute data can be expressed with them. The identity data set can be defined as a VC schema, as shown in Section 4.5.3.3. This requirement is *completely fulfilled*.

- **Identity de-provisioning** (CON1): As VCs are designed to be issued with short validity periods, as described in Section 4.3.5.4, any VC stored outside the subject's control is automatically invalidated regularly. From the subject's point-of-view, the data can be deleted in the wallet to de-provision the identity. This requirement is *completely fulfilled*.

- **Access controls** (SEC1): Access decisions can be based on the values stored in VCs. This requirement is *completely fulfilled*.

- **Credential revocation** (SEC2): Revoking credentials is not feasible in a secure and privacy-focused system. This requirement is therefore *overruled*. The mediating factor is that VCs are only issued for limited validity periods, like X.509 certificates.

- **Mutual authentication** (SEC3): Impersonation of other entities is a primary attack vector in many (web-based) attacks. The protocol described in Section 4.3.2.3 shows how both entities can authenticate mutually to prevent impersonations. This requirement is *completely fulfilled*.

- **Security by default** (SEC4): By default, communication and VC exchanges are peer-to-peer. No private information needs to be disclosed publicly or be written to the DLT. In fact, doing so should be made deliberately hard. This requirement is *completely fulfilled*.

- **Security by design** (SEC5): The concept relies on well-known methods of encrypting and signing messages through public-private key cryptography. Information stored on the distributed ledger is similarly signed and integrity protected by well-known hashing functions. This requirement is *completely fulfilled*.

- **Tamper-evident** (SEC7): The use of PUF for managing IoT devices' identities is discussed in Section 4.3.2.5. This allows any tampering to be detected, as it is highly likely to destroy the device's identity. This requirement is *fulfilled*.

- **Off-the-record (OTR)** (SEC12): The privacy of the exchange of information between the identity holder and the RP supports repudiation towards third parties. This is achieved through short validity periods of VC and the publication of the private keys used to sign them, as first introduced by the OTR protocol [20] and adapted to SSI in Section 4.3.2.3. Because of the delay between showing the VC and it being invalidated, this requirement is just *fulfilled*.

- **Privacy by default** (DAT1): Private information is always stored in the user's wallet by default. Depending on the implementation, agents might allow the user to store the PII elsewhere, but by default, it is in the wallet under their control. This requirement is *completely fulfilled*.

- **Privacy by design** (DAT2): With no direct connection between the issuer of PII and the RPs, the user (or agents on their behalf) is always involved in an exchange and in control of the information flow. This requirement is *completely fulfilled*.

- **Protected application storage** (DAT5): As shown in Section 4.3.2.5, the confidential data required to be stored on the entity's wallet can be protected through the use of TPM systems. However, even those can be misconfigured, ill-designed, or otherwise broken. This requirement is *partially fulfilled*, and further research into the use of external hardware security devices is necessary.

- **Reliability** (ROB1): The reliability of this concept is provided by not having a single point of failure. Even if the DLT breaks down and cannot be continued, the last valid state allows most operations to continue as usual while services are restored. This requirement is *completely fulfilled*.

- **Offline authNZ** (ROB5): Communication between two entities can be done via any local network, i. e., offline without the Internet. In this setup, no information stored on the DLT can be used, but basic authentication and authorization (authNZ) can be performed. Information stored on the DLT is shown in Section 4.3.1.1. This requirement is *completely fulfilled*.

- **Scalability** (ROB6): The concept relies on generating public-private key pairs as the basis for the entities' identities. Those can be generated at each entity without much effort. The DLT is only used sparingly to store information about public identities, attributes, and schemata, as described in Section 4.3.1.1. Thus, it is not limiting the scaling of the number of participants. This requirement is *completely fulfilled*.

4.6.2 Important Requirements

- **Access delegation** (SAT6): Delegation of access permissions in the form of VC is possible through credentials that allow being passed on by entities. The basics of delegation are described in Section 4.3.5.3. An example of defining such credentials is described in Section 4.5.3.2. This requirement is *completely fulfilled*.

- **Identity data set matching** (SAT8): The task of adapting and localizing VCs is described in Section 4.3.6.3 and used in the RP's workflow as described in Section 4.3.4.3. This requirement is *fulfilled*, as it can be supported by the system but relies on the CLS as an additional component.

- **Documentation** (INF2): The essential aspects of the concept have been described in this chapter. This does not fulfill the aspect of having complete application documentation for developers, operators, and end-users, but documentation in this style is out of scope. This requirement is *partially fulfilled*.

- **Product specification** (INF5): This requirement can be fulfilled in multiple ways. Either the entity provides a VC that proves the entity is operating for a specific organization, or the organization can provide a customer with a VC listing all the entities' identifiers that are employed by them. This requirement is *completely fulfilled*.

- **Agreement monitoring** (INF8): Representing and monitoring agreements is not directly supported by the concept. The short-lived nature of the VC requires constant re-issuing of VC, which shows commitment to a contract. In many cases of edge computing and IoT, this will be sufficient. This requirement is *partially fulfilled*.

- **Capability exchange** (INF9): A service's capabilities can be shown through either self-issued VC or even be asserted by other entities through VCs issued by them. This requirement is *completely fulfilled*.

- **Level of assurance** (INF11): The level of assurance of an issuer can be asserted through a VC, either self-issued or issued by another vetting entity. An example of how VCs can be structured for this use case is provided in Section 4.5.3.3. This requirement is *completely fulfilled*.

- **Credential recovery** (CON2): Due to the need to regularly re-issue VC, losing a set of VCs is not dramatic. The loss of the identity's root private keys, however, is an irrecoverable problem. Section 4.3.2.4 provides solutions to facilitate backups of the identity's key data. This requirement is *fulfilled*.

- **Digital twin** (CON3): A reliable digital identity for IoT is essential for establishing digital twins, and SSI, as described in Section 4.5.3.2, can provide those. However, digital twins in itself are part of the application's implementation and not the IAM system. As a result, this requirement is *not fulfilled*.

- **Content verification** (CON4): Providing VCs proving certain assumptions about an entity in combination with the entity's generated data is shown in Section 4.5.3.2. As a result, this requirement is *completely fulfilled*.

- **Netsplit/Join** (CON5): Re-synchronization between segmented DLT partitions depends on the used consensus protocol of the ledger, as described in Section 4.3.1. Other entities do not need to synchronize but can continue to access the ledger as usual. As a result, this requirement is *completely fulfilled*.

- **Once-only** (CON6): The greater goal of once-only, not requesting information about the user, which is already stored at another government agency, can be achieved by storing all user data with the user and querying it as needed. If this actually fulfills the legal requirements of once-only legislation, is not clear at this time. This requirement is *partially fulfilled*.
- **Multi-factor authentication** (SEC6): MFA on a local database, like the wallet, is fairly limited in options that do not rely on another trusted third party. However, options for including MFA in SSI are described in Section 4.3.2.5. This requirement is *completely fulfilled*.
- **Delegation parameters** (SEC8): If an entity issues a delegation based on a received VC, it acts as an issuer. This particular case of issuing VCs is described in Section 4.3.5.3. This requirement is *completely fulfilled*.
- **Secure de-provisioning** (SEC9): Any stored VCs will become obsolete after their validity period making secure deletion of those easy. The identity's private keys, however, must be disabled securely. As described in Section 4.3.2.5, the keys should be stored in a TPM where secure deletion can be achieved. This requirement is *fulfilled*.
- **Secure setup** (SEC10): The secure setup procedure established by manufacturers of IoT devices can include issuing VC to the organization or individual buying their product, as shown in Section 4.5.3.2. This requirement is *fulfilled*.
- **Tamper-resistant** (SEC11): Tamper resistance can be achieved by utilizing TPM hardware, as described in Section 4.3.2.5. This kind of hardware, however, is not universally available or accessible, so it cannot always be used. This requirement is *fulfilled*.
- **Trust service providers** (SEC13): Building or including a PKI infrastructure into a decentralized SSI system using VC or X.509 certificate is possible, as shown in Section 4.5.3.3. Those can designate certain issuers as especially trusted within a specific domain. This requirement is *fulfilled*.
- **GDPR** (DAT3): While a concept in itself cannot be compatible with specific legislation and determining legal compliance takes actual legal review, the concept shown in this section should be possible to implement in a way that achieves GDPR compliance. This requirement is *partially fulfilled*.
- **Correlation resistance** (DAT6): Metadata leaked by the issuance or presentation of VCs is limited to the metadata produced by the used transport protocol (i. e., TCP/IP or HTTP). This requirement is *fulfilled*.
- **External tracking resistance** (DAT7): All data exchanged for SSI is encapsulated in the used transmission protocol in an encrypted form. An outside observer cannot deduct which entity is communicating via SSI. However, devices can still be tracked using other identifying network information, for example MAC, addresses. This requirement is *fulfilled*.
- **Internal tracking resistance** (DAT8): Observers that are part of the SSI system but not part of a specific exchange do not learn any more information than outside observers. This requirement is *fulfilled*.
- **Approachability** (ROB3): From a user's perspective, an SSI app's authentication process is not too dissimilar to using password-safe apps. The scanning of QR codes to connect with websites or terminals is also pretty common. This requirement is *fulfilled*.

- **Accessibility** (ROB2): The accessibility of this concept is not limited to specific devices, applications, or operating systems, except for requiring specialized TPM hardware for added security with highly reliable PII like government-issued eID. This requirement is *fulfilled*.
- **Usability** (ROB4): The concept is fairly focused on exchanging VCs between issuers, users, and RPs. An actual implementation's usability will differ based on implemented and added features. This requirement is *fulfilled*.
- **Platform independence** (ROB7): The requirements necessary to run the shown SSI concept consist of being able to sign and verify public-private key signatures and communicate with other entities through an interface. Any hardware supporting those two can use the concept. This requirement is *completely fulfilled*.
- **Transitive trust** (ROB8): Scaling trust for connection lengths larger than two hops is difficult. Any entity b can prove to another entity c that it received a VC from a third entity a. This already requires entity c to trust entity a issued the VC correctly. Entity c can now issue a VC to itself or another entity that it observed the VC from entity a at entity b. This solves transitive trust on a technical level, but how to reliably interpret it on an organizational level cannot be answered. This requirement is *partially fulfilled*.
- **Communication protocol independence** (ROB9): VCs exchanges can be performed over a variety of transport protocols. This requirement is *completely fulfilled*.
- **Resource efficiency** (ROB10): In [70] it was shown that the necessary operations for SSI can be used on very low-powered microcontrollers using LoRa®. This requirement is *completely fulfilled*.
- **DDoS protection** (ROB11): The basic credential exchange and authentication is handled between two entities directly. There is no central component that can be attacked via DDoS. This requirement is *completely fulfilled*.
- **Standalone authNZ** (ROB12): Similar to the **Offline authNZ** (ROB5), authentication and credential exchanges can be done with limited or outdated information. This requirement is *completely fulfilled*.

4.6.3 Optional Requirements

- **Service discovery** (INF7): Service discovery can be implemented on top of the SSI concept but is not part of this concept. This requirement is *not fulfilled*.
- **Multiple identities** (DAT4): The concept does not limit the number of identities. Each entity can manage multiple identities for different scenarios or use one identity, as each connection to another entity cannot be correlated and acts like a separate identity. This is described in Section 4.3.2.3. This requirement is *completely fulfilled*.

Table 4.3: Comparison of the requirements met by the concept

Category	Requirement	Concept	Importance
Satisfaction	SAT1: Authentication	*	↑
	SAT2: Authorization	*	↑
	SAT3: Identification	*	↑
	SAT4: Identity provisioning	*	↑
	SAT5: Trust establishment	+	↑
	SAT6: Access delegation	*	↕
	SAT7: Automated integration/registration	*	↑
	SAT8: Identity data set matching	+	↕
	SAT9: Message delivery services		↓
Information	INF1: Credential establishment	+	↑
	INF2: Documentation	~	↕
	INF3: Digital identification	*	↑
	INF4: Physical identification	+	↑
	INF5: Product specification	*	↕
	INF6: Neighbour discovery	*	↑
	INF7: Service discovery	-	↓
	INF8: Agreement monitoring	~	↕
	INF9: Capability exchange	*	↕
	INF10: Identity data set	*	↑
	INF11: Level of assurance	*	↕
Consistency	CON1: Identity de-provisioning	*	↑
	CON2: Credential recovery	+	↕
	CON3: Digital twin	-	↕
	CON4: Content verification	*	↕
	CON5: Netsplit/Join	*	↕
	CON6: Once-only	~	↕
Security	SEC1: Access controls	*	↑
	SEC2: Credential revocation		↑
	SEC3: Mutual authentication	*	↑
	SEC4: Security by default	*	↑
	SEC5: Security by design	*	↑
	SEC6: Multi-factor authentication	*	↕
	SEC7: Tamper-evident	+	↑
	SEC8: Delegation parameters	*	↕
	SEC9: Secure de-provisioning	+	↕

Continued on next page

Table 4.3: Comparison of the requirements met by the concept (Continued)

	SEC10: Secure setup	+	↕
	SEC11: Tamper-resistant	+	↕
	SEC12: Off-the-record (OTR)	+	↑
	SEC13: Trust service providers	+	↕
Data Protection	DAT1: Privacy by default	*	↑
	DAT2: Privacy by design	*	↑
	DAT3: GDPR	~	↕
	DAT4: Multiple identities	*	↓
	DAT5: Protected application storage	~	↑
	DAT6: Correlation resistance	+	↕
	DAT7: External tracking resistance	+	↕
	DAT8: Internal tracking resistance	+	↕
Robustness	ROB1: Reliability	*	↑
	ROB2: Accessibility	+	↕
	ROB3: Approachability	+	↕
	ROB4: Usability	+	↕
	ROB5: Offline authNZ	*	↑
	ROB6: Scalability	*	↑
	ROB7: Platform independence	*	↕
	ROB8: Transitive trust	~	↕
	ROB9: Communication protocol independence	*	↕
	ROB10: Resource efficiency	*	↕
	ROB11: DDoS protection	*	↕
	ROB12: Standalone authNZ	*	↕

Key: ↑ essential, ↕ important, ↓ optional, * completely fulfilled, + fulfilled, ~ partially fulfilled, - not fulfilled, not applicable

Chapter 5

Prototype Implementation and Application

Contents

This chapter utilizes the concept developed in Chapter 4 to realize SSI by implementing software and applying it to a subset of the scenarios described in Chapter 2. The parts of the concept that need to be implemented are discussed in Section 5.1. Applying the implementation to the respective scenarios is described in Section 5.2. The selection of scenarios follows the decision made in Section 4.5.3, where this subset was selected based on maximizing coverage of requirements and realistic use cases. The main application areas are the Internet in Section 5.3, IoT in Section 5.4, and eID in Section 5.5. The implementations shown here are proof-of-concept to show the concept's plausibility. However, the implementations may contain security-critical bugs and are not intended for use in production environments.

5.1 Selection of Components to Implement

The concept developed in Chapter 4 consists of eight core components for SSI. Those are described in Section 4.3. However, not all of those components are new. For some, there are already existing implementations. As a result, the focus of the prototype implementation is to implement new or significantly changed components, as described below.

- The **DLT** does not need to be implemented to show this concept, as there is already a diverse ecosystem of DLT implementations. The exact DLT used should not matter for the concept's implementation and should also work with non-DLT systems like PKI. Each of the DLT variants or the PKI has different arrangements for their security and data protection management, which have to be evaluated for the respective application with the help of the concept.
- An **issuer**'s necessary provisions for issuing VCs must be implemented. As the concept differs from other SSI concepts, this part cannot be re-used from existing implementations. The need to constantly re-issue VCs affects the implementation, API, and organizational processes.
- As the issuer uses a custom implementation described in the previous item, the **relying party** must also be implemented to be compatible with the VCs issued. On the organizational level, the RP also needs to adapt, especially regarding offline authentication and the non-existence of revocation of VCs.
- For best fit, the **wallet** should always be implemented for a particular operating system or hardware. Where necessary, a wallet is implemented for each prototype.
- Similar to the wallet, the **agent** and its characteristics must be adapted heavily for each application and target audience. A generic version is also implemented here to show the interfacing with the other components.
- As a completely new component to SSI, the **credential localization service** is an optional component only necessary for larger SSI environments. No prototype of this component is planned, as evaluation requires already working large-scale deployments. The integration of a CLS is conceptually shown in Section 6.2.
- The **trust gateway** is also a new component, and a prototype faces problems similar to the CLS. Therefore, it is not implemented but its conceptual application is shown in Section 6.2. It is especially relevant as an example of how the conceptualized SSI system can interface with other IAM systems and how a gradual migration might be done.
- As the last component, the **community** holds special significance for the long-term success of the SSI system. In the scope of this work, a community can not be implemented.

Reproducing the components' implementation results can be achieved by re-implementing some of those components using the concept as a reference and the descriptions from Section 5 as blueprints. All implementations are integrated into Docker environments to allow the reproducing of the prototypes' setup. As the components can not be used to evaluate their suitability, they must be examined in a suitable environment. Therefore, Section 5.2 discusses the scenario selection to test the implemented components.

5.2 Selection of Scenarios to Apply the Implementation to

Due to the number of scenarios, not all can be implemented as prototypes. Those chosen are paired with the components selected for implementation in the previous Section 5.1. The resulting implementations are shown in Table 5.1. It contains the same text as Table 4.2, detailing the characteristics of the components in specific scenarios but highlights which of the concept's components are implemented in which prototype.

Table 5.1: Components selected for implementation in the three prototypes

	Scenario		
Perspective	Web Apps	IoT	eID
DLT	PKI	Private/Permissioned	Public/Permissioned
Issuer	Corporations	Manufacturer	Government
Relying Party	Web Applications	Data Processors	Government Agencies
Wallet	Browser	Embedded	App
Agent	Web Service	Cloud Service	Certified Provider
CLS	none	Service Provider	none
TGW	none	Service Provider	Certified Portal
Community	Public	Sector Specific	Public-private Partnership

Key: implemented, partially implemented, not implemented

Issuing and verifying VCs is necessary for showing anything related to SSI. Therefore, issuer and RP are implemented for each scenario. In the web application scenario, the issuer is implemented for a fictive issuing corporation that utilizes the VCs directly as an RP to authenticate and authorize users. The IoT scenario manages IoT devices' access to the network through SSI. Therefore, this scenario focuses less on VC issuing and verifying but more on basic IAM. The eID scenario prototype utilizes existing SSI systems to show how SSI could work for eID.

A wallet is implemented for each scenario to show and test different approaches. The web application scenario utilizes a browser-based wallet build as a complete prototype. The IoT scenario utilizes a cloud-based wallet, and the eID scenario's wallet is a smartphone application. The latter is implemented as a prototype without integrating trusted compute modules, the former as a hybrid of cloud-based and embedded wallet. The agent that would be part of each scenario is only implemented in the IoT scenario. There, it is a central component and acts as a cloud-based wallet, similar to a TGW.

5.3 Web Applications

With X.509 certificates being prevalent in most websites today, its use as a backend for SSI would provide easy adoption. In this scenario, the issuers are websites, for example, social networks or other corporations, that provide their users with an option to log in and transfer personal attributes via SSI. The users manage their identities, accounts, and VCs in a browser extension or dedicated application. The community of this scenario is the general public on the Internet.

A system with similar design goals called BrowserID was developed by the Mozilla Foundation under the name Mozilla Persona. A closer description of this system is provided in Section 3.1.3. This system suffered from some security problems [58], which were fixed, but the system still did not gain any significant traction and sits on GitHub, abandoned for several years. As a result, the complete toolchain used in this project is outdated, and getting it to run on current software versions is not feasible. Still, the idea to utilize HTML 5 local storage and browser plugins to store

credentials issued by websites in the form of JSON web tokens (JWTs) is not too far from the idea of an issuer providing the user with VCs they can present to any RP.

The web application utilized to implement the prototype is based on the scenario described in Section 2.5.1. It is a simple publish/subscribe platform where users can post messages and retrieve messages about specific topics or keywords. This section describes the technical components and modifications made to implement a similar system for SSI in Section 5.3.1. Further, organizational processes are outlined in Section 5.3.2. To complete this prototype implementation, Section 5.3.3 summarizes the development and highlights key areas where the system could be improved.

5.3.1 Technical Components

To match the scenario closely, user identification, authentication, and authorization must be based on SSI workflows. The web application's main "useful" features are storing and displaying users messages using a basic web frontend. This frontend needs to communicate with the backend to recognize users, receive their messages, and store and retrieve them for others.

The prototype application, therefore, needs to operate as RP and issuer at the same time. For easier integration, it is essential that most of the implementation for RP and issuer is re-used and core functions are split into modules. The most critical module developed for this use case is the `LibWebSSI` library, which provides SSI functions to issuers and RPs.

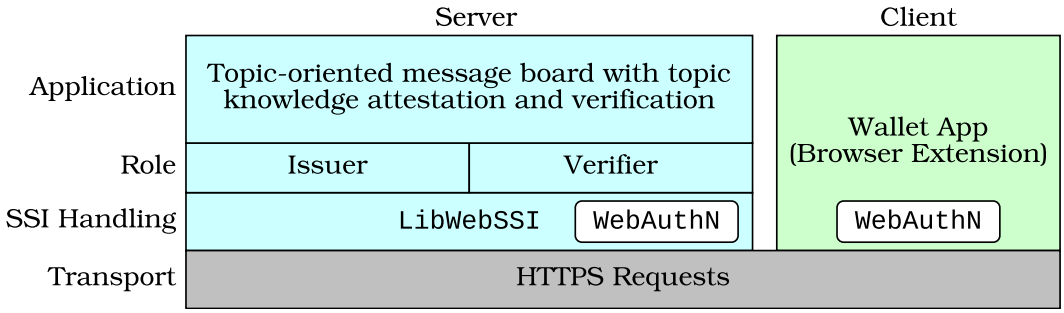

Figure 5.1: Overview of the implemented components for the web-based SSI prototype

The implemented components are shown schematically in Figure 5.1. All components displayed in aqua are implemented using the Scala 3 programming language. The wallet app (displayed in light green) is implemented as an extension for the Chrome web browser using JS. As a basis for modern browser-based authentication, the WebAuthn [103] standard is utilized. WebAuthn is a standard integrated into many modern browsers and provides an API to facilitate public key-based authentication between users and web pages. It interfaces with security devices like FIDO-enabled Yubikeys and works with recent versions of Android as an authentication source. As a transport for communicating between the wallet, the issuer, and the RP, HTTP `GET` and `POST` requests are used. The messages are encrypted and signed by the participants, but they should be passed via HTTPS instead of plain HTTP for added security. More details regarding the individual components are provided in the following sections.

5.3.1.1 Web Application

The web application is implemented using Scala 3 and the `cask`[1] framework. Both being written in Scala 3, this setup integrates the `LibWebSSI` library with the webserver. It also provides a direct setup for the prototype without needing an external web server, i. e., Apache2 or nginx. However, a web server is required to act as TLS reverse proxy if access to the application must be possible via HTTPS.

The application exposes the following endpoints:

- `/`: This endpoint shows the main page and, depending on the visitor's status, a connection or post composition form. The visitor's login status is tracked using a server-side session and browser cookies.
- `/connect`: To log in to the page, the user must connect with their web SSI-enabled wallet. The connection endpoint initiates the WebAuthn authentication workflow by generating a JSON formatted `publicKeyCredentialCreationOptions` configuration object. This object contains the challenge to be signed by the user and information about eligible cryptographic methods and authentication devices.
- `/confirm`: A challenge generated by the connection endpoint needs to be signed, and the resulting `PublicKeyCredential` is returned via this endpoint following the WebAuthn workflow. If the server can verify the challenge, a new user session is started, and the user is logged in via a browser cookie.
- `/preview`: After composing a new post, it is processed by the server to extract the contained topics and generate requests for VCs presentations. Those are returned with the post's preview, and the SSI wallet can notify the user about the option to include VCs.
- `/submit`: The previewed post with any VCs provided by the user is submitted to this endpoint. If the VCs can be verified, the post is added to the post list, displayed on the website's main page.
- `/vt/<value>`: The website can also issue VCs, e. g., for knowing about the website and SSI. This endpoint creates a VC for the `topic-knowledge` credential schema with the value specified in the URL.

The function of the `/connect` and `/confirm` endpoints are detailed further in the next section, as the application relies heavily on the `LibWebSSI` library to process requests to those. As part of the prototype nature of this application, it only persists posts and users in memory. Each restart of the application results in the same base state.

5.3.1.2 SSI Library `LibWebSSI`

As a core part of the prototype, the SSI library handles all operations related to creating and verifying connections between entities and creating and verifying VCs. It is developed using Scala 3 utilizing, a modern programming language with the option of including numerous Java-based libraries.

The `LibWebSSI` library implements four essential functions for SSI on the web:

- **Establishing connections:** To establish a new connection, the connecting user provides their user and display name. The library generates a unique ID for each user they must provide for subsequent authentications. Additionally, a random 128 byte challenge is created that must be signed by the user's authenticator. The user's ID and challenge are stored temporarily while awaiting the response.

[1] `https://com-lihaoyi.github.io/cask/index.html`

- **Issuing verifiable credentials:** The library provides abstractions to represent attributes and schemata for VCs. Using a connected user's identity and the issuer's private key, the library generates a JWT containing the claims and values. The issuing instant and expiration date are managed through the JWT.
- **Verifying verifiable credentials:** As VCs are represented through JWTs, the verifying process first checks the validity of each JWT. The library does not automatically search for the issuers' metadata. Instead, it expects to be provided with a list of issuing entities. This allows the library to remain stateless and leave the task of managing known issuers to the calling application.
- **Retrieve issuer metadata:** While the library does not store the issuers' metadata, it provides a function to search and retrieve it.

Cryptographic operations are implemented using the Java cryptography API and the Bouncy Castle API[2] because no Scala 3 native alternatives offer the same functionality and well-tested security. All operations requiring the processing of JWTs utilize the JWT Scala[3] library. Processing of JSON files is primarily done using ujson[4]. Additionally, all concise binary object representation (CBOR) encoding and decoding is done through borer[5].

The library manages entities of three abstraction levels. Any `Entity` is identified by its `ldid` locally and self-identifies with its remote `rdid`. Using two identifiers for one entity ensures decentralized unique identification of pairwise connections. The other entities extend the base entity, first adding authenticatability by storing the entity's public key for an `AuthenticatableEntity` and then adding a private key for the `AuthenticatingEntity`, which can authenticate to other entities. Listing 5.1 depicts how those entities are implemented in Scala 3.

```
class Entity(
    val ldid: String,
    val rdid: String
  ):
  def canEqual(a: Any): Boolean = a.isInstanceOf[Entity]

  override def equals(that: Any): Boolean = that match
      case that: Entity =>
        that.canEqual(this) && (that.ldid equals this.ldid)
      case _ => false

class AuthenticatableEntity(
    ldid: String,
    rdid: String,
    val authenticator: PublicKey,
    var authenticated: Boolean = false,
  ) extends Entity(ldid, rdid)

class AuthenticatingEntity(
    ldid: String,
    rdid: String,
    authenticator: PublicKey,
    val authenticatee: PrivateKey,
  ) extends AuthenticatableEntity(ldid, rdid, authenticator, true)
```

Listing 5.1: Entity representation within `LibWebSSI`

[2] https://www.bouncycastle.org/java.html
[3] https://jwt-scala.github.io/jwt-scala
[4] https://com-lihaoyi.github.io/upickle/#uJson
[5] https://github.com/sirthias/borer

Another essential part of the library is its handling of attributes, attribute schemas, and VCs. For transport, they are always encoded as JWTs, but internally they are represented as implemented with Scala 3 in Listing 5.2.

```
case class Attribute(
    schema: AttributeSchema,
    value: Array[Byte],
  )

case class Attestation(

  )

case class Claim(
    typ: String,
    subject: AuthenticatableEntity,
    issuer: Entity,
    attestation: Attestation,
    attributes: List[Attribute],
  )

case class VerifiableCredential(
    claims: List[Claim],
    scope: String,
    notBefore: DateTime,
    notAfter: DateTime,
  )
```

Listing 5.2: Attribute representation within `LibWebSSI`

When interfacing with the library's API, the caller must provide those objects where applicable.

5.3.1.3 Browser Extension

The browser extension is used as an intuitive and seamless interface for the users to manage their connections and VCs on the Internet. An extension mimics the long-term goal of having a browser-builtin feature for managing SSI identities on the Internet. The extension must protect the credentials from unauthorized access and provide an interface for receiving, presenting, and general management operations. It can also be used to synchronize the identity data with other browser instances of the user to allow cross-device usage. The extension is built for `Chromium`-based browsers using the Chrome Extension API.

As it is explicitly not recommended [37] by the developer documentation for Chrome browser extensions to store confidential data like passwords and private keys, the extension utilizes the browser's built-in capability of WebAuthn for handling the identity keys. However, the JWT-based VCs are stored in the extension's storage and, if enabled, utilize the synchronization mechanism to make them available on all browser instances of the same user.

To fulfill the basic requirements, the extension implements the following three operations:

- **Establishing a connection:** When activated by the user, the extension injects a small JS script into every webpage to determine if it contains an element with the ID `web-ssi-connect`. If it does, further JS code is loaded into the page, and the connect button inside the extension's popup window is enabled. When pressed, the extension loads the connection challenge, signs it using the provisioned private key, and returns the signed message as JWT to the

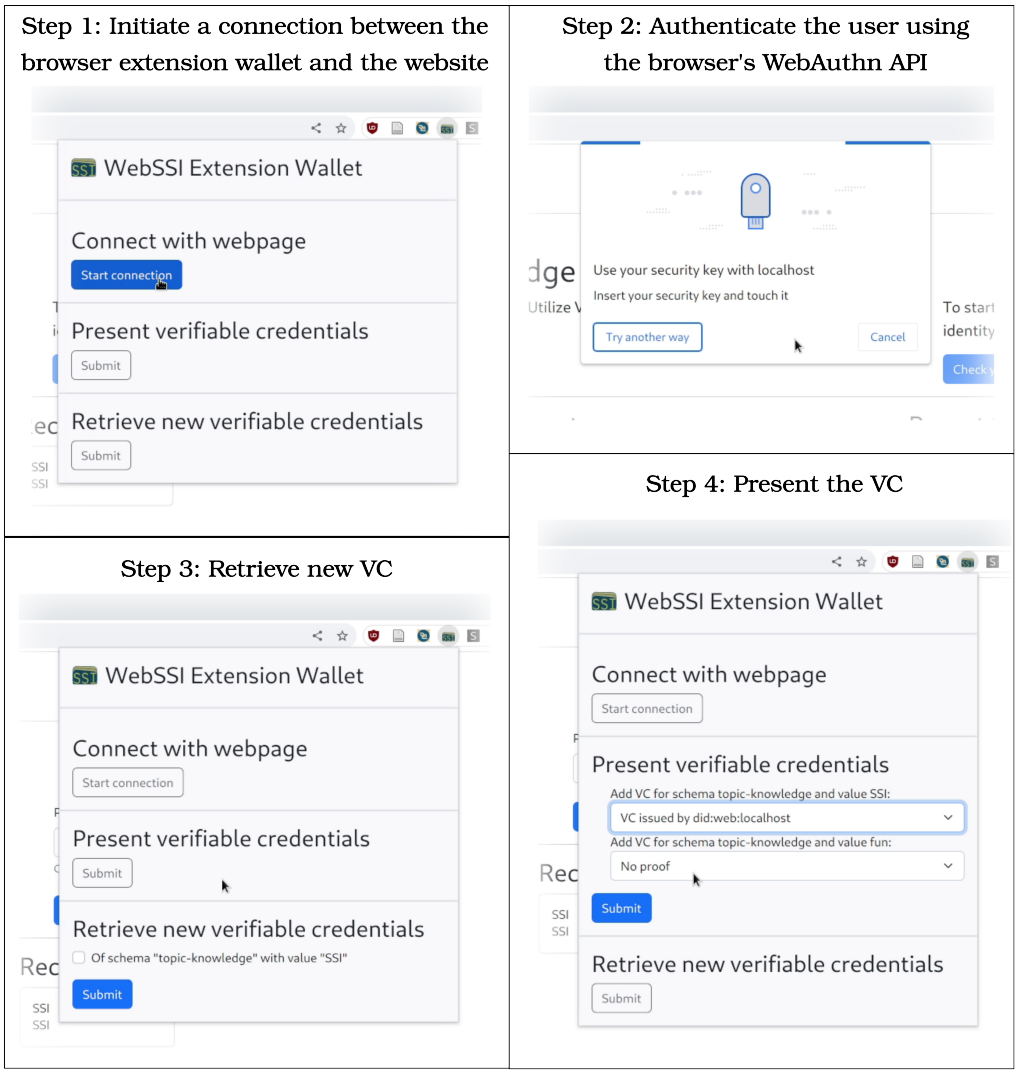

Figure 5.2: Screenshots of the prototype's UI for establishing a connection and exchanging VCs with the browser extension wallet

server. If the server can verify the response correctly, the server handles creating and managing the user's session. The same workflow can be used for first-time and repeated connections.

- **Retrieving verifiable credentials:** Similar to the connection establishment process, the extension uses some JS code to determine the presence of elements with a custom HTML tag name `web-ssi-credential-offer`. This tag allows multiple offers to be placed and found by the script. The tag also contains additional information like the VC schema, attribute value, and URL to retrieve the offered VC. This information is passed to the extension's popup window, and using checkboxes, the user can select which VCs they would like to retrieve.
- **Presenting verifiable credentials:** The presentation of VCs is also triggered by a small piece of JS that scans the page for forms with `input` fields containing the following data attributes: `data-web-ssi-verification-type`, `data-web-ssi-verification-schema`, and `data-web-ssi-verification-value`. An ID cannot be used in this case, as multiple requests may be part of the same form. The information from the data attributes is used to build a list in the extension's popup window, prompting the user to select which verification requests to answer using which VC from the extensions repository. If the user selects some VCs, those are input into the form fields and sent to the server for verification if the user submits the form.

To keep the overhead of the extension as low as possible, no JS is automatically added to each page to look for the signature elements of an SSI-enabled web page. Instead, the page has to notify the user of the possible use of web-based SSI, and when opening the extension's popup window code is injected into the page to retrieve necessary information. A minimal example of this is shown in Listing 5.3 for enabling the "connect" button.

```
function injectConnection() {
  return document.getElementById("web-ssi-connect");
}

// Activate connect button
chrome.tabs.query({active : true, currentWindow : true}).then((tab) => {
  chrome.scripting.executeScript(
      {target : {tabId : tab[0].id}, func : injectConnection},
      (injectionResult) => {
        if (injectionResult[0].result) {
          // Enable the connect button in the popup window
          connectButton.disabled = false;
        }
      });
});
```

Listing 5.3: Determining if the page supports web-based SSI for the extension's user interface

The extension utilizes the `jrasign`[6] library for all cryptographic operations, including signing and verifying JWTs. For the WebAuthn workflow, the `webauthn-json`[7] library is used. It handles interfacing with the browser's WebAuthn implementation and supports the otherwise cumbersome conversion between binary data, which is Base64-encoded in a JSON string, and the WebAuthn API's expected `unsigned int ArrayBuffer`. Using WebAuthn allows authenticating via Yubikeys and other FIDO-enabled hardware tokens, as well as Android smartphones. Thus, the solution does not require additional passwords or passphrases to manage the identity's keys.

[6] https://github.com/kjur/jsrsasign
[7] https://github.com/github/webauthn-json

5.3.2 Organizational Processes

The Internet is intended and built to be decentralized and resilient. As such, website operators, infrastructure providers, governments, and users operate independently and in their own interests. For handling IAM securely, the lifecycle shown in Section 2.2, transformed into requirements in Section 2.5.1, and conceptualized in Section 4.3 needs to be supported. Most of the requirements for secure authentication are fulfilled by using WebAuthn, as described in Section 3.1.1. However, for using VCs on top of the authentication layer, some additional, mainly organizational, processes need to be addressed:

- **Renewal:** To renew VCs from the issuer, the user's browser extension, acting as the user's agent, must contact the issuer at regular intervals. If the refresh interval is chosen such that the pool of VCs does not run out before they are due to be refreshed, the issuer cannot deduce how many have been used. If the user does run out of usable VCs before they would expire, an early request for new VCs might indicate to the issuer that the user is relying on those VCs heavily. To prevent DoS attacks on the issuer, the issuer might also rate-limit the number of VCs issued per entity to a manageable number. The extension offers the user an option to define a minimum number of VCs to keep ready at any moment. If the supply drops below, new VCs are automatically requested.

- **Revocation:** The scheme described in Chapter 4 argues why explicit revocation cannot work in a privacy-preserving and reliable way. The only way to revoke an issued VC is to wait for it to expire and prevent re-issuing it. Depending on the necessary security level, the VC's expiry time must be chosen. The implemented web service issues VCs that expire after 90 days, the same as the LetsEncrypt X.509 certificates, which inspires this model.

- **Trust:** Facilitating trust between different entities on the Internet is a difficult problem, which originates from the many diverse, decentralized, and international participants. Only a few entities are globally trusted, evident by the large number of root CAs in most operating systems and browsers. Agreements between different websites on which VC they want to accept will need to be established on a case-by-case basis. As the prototype web page is both issuer and RP, brokering trust is unnecessary here.

5.3.3 Summary

As the first prototype, the implemented components show how an SSI system can be built from scratch. Instead of using existing IAM projects or libraries and only utilizing common browser technologies like X.509 certificate and JWT, a system similar to the ones shown in Section 3.3 was built. This system also shows that SSI not necessarily requires DLT.

While the prototype for web apps demonstrates SSI operations in a browser, further development needs to recognize the challenge of handling private data within this environment. Most of the storage accessible for websites and extensions is unsuitable, as its confidentiality, integrity, and availability can not be guaranteed. As a result, private data would need to be stored in an external password-safe-like component, or the wallet would need to be integrated more tightly with the browser itself. With larger adoption of SSI concepts, both solutions seem possible.

5.4 IoT Sensors

IoT applications can differ significantly between use cases. As a result, three individual scenarios were developed to deduce the requirements in Sections 2.5.2 to 2.5.4. The prototype developed for evaluating the concept of Chapter 4 focuses on the second scenario based around IoT sensors, described in Section 2.5.3. IoT sensor networks and their large-scale deployment is an area of active research that is accompanied by practical products and standardized protocols.

The prototype aims to show the applicability of the SSI concept described in Chapter 4 for one IoT platform. Key areas of consideration are the integration with the least number of changes, especially to the many devices that are potentially already deployed and scalability. The necessary additions and changes are described from a technical point of view in Section 5.4.1 and from an organizational point of view in Section 5.4.2.

5.4.1 Technical Components

The prototype is designed to use a private permissioned distributed ledger, as described in the concept's Section 4.5.3.2. As described in the presentation of the IoT scenarios in Section 2.5, there are many aspects to consider for IAM in IoT. The described scenarios differentiate between IoT devices, sensors, and networks. For the prototypical implementation, a setting containing aspects of all three of them is chosen: a LoRaWAN®-based IoT platform primarily focused on IoT sensors, but also supporting general IoT devices. This platform is built on the *The Things Stack (TTS)*[8] network server. Other similar servers, like *Chirpstack*[9] also exist. However, the largest open LoRaWAN® network, *The Things Network* (TTN), is built with TTS, making it the primary candidate for integrating SSI into a general-purpose IoT architecture.

Another proposal to combine LoRaWAN® and blockchain for IAM by [158] has been shown in Section 3.4.2. This approach differs by not replacing the join server with DLT but using SSI as part of the join server's protocol. The usual IAM setup of LoRaWAN® is described in detail in Section 3.4.1. Integrating an SSI cloud agent into the LoRaWAN® TTS is described in Section 5.4.1.1, and the complete process of transmitting data from IoT sensors with SSI is shown in Section 5.4.1.2.

5.4.1.1 The Things Stack Modifications

Initially, the root keys must be configured in the TTS's management interface or with an external join server. When extending an existing protocol stack like LoRaWAN® to support SSI aspects, care must be taken to keep changes to the necessary minimum. This way, onboarding new devices is more manageable, and utilizing the modified protocol alongside the original one remains possible. For this reason, end devices are excluded from protocol changes and keep functioning as before.

The base software for this prototype is TTS, which implements LoRaWAN® `1.0.X` and `1.1`. TTS also includes an interoperability server that can configure different join servers for different device IDs. This interoperability is independent of the roaming features introduced with LoRaWAN® `1.1` and thus works for all LoRaWAN® versions.

[8] `https://www.thethingsindustries.com/stack`
[9] `https://www.chirpstack.io`

The device IDs consist of a network ID and a host ID, similar to IP addresses. It can – just like an IP address – be expressed in a CIDR-like notation. The interoperability server of TTS can be configured to forward network messages of certain devices' networks to specific servers. For example, the configuration shown in Listing 5.4 specifies that all devices whose device IDs start with `abcd` should be handled by the join server configured in the file `./js/js.yml`.

```
join-servers:
  - file: './js/js.yml'
    join-euis:
    - 'abcd000000000000/16'
```

Listing 5.4: Example interoperability configuration for TTS

The file describing the join server more closely is shown in Listing 5.5. This listing shows the necessary configuration to join devices using the SSI-enabled join server developed for the prototype. All configuration options are described in TTS documentation [101].

```
scheme: 'https'
fqdn: '172.20.0.1'
port: 8080
protocol: 'BI1.0'
paths:
  join: 'joinRequest'
  app-s-key: 'appSKeyReq'
tls:
  root-ca: './ca.pem'
  certificate: './ca.pem'
  key: './key.pem'
```

Listing 5.5: Example configuration for a specific join server for TTS

For simplicity reasons and because most devices currently use it, the LoRaWAN® `1.0.X` family of protocols is chosen for the prototype. It could, however, be adapted to also support version `1.1`.

With the interoperability server of TTS configured, the SSI agent must implement the join server's protocol to generate join responses for the end devices and application session key requests by the applications. Those two endpoints are also defined in the configuration of Listing 5.5.

The response to a join request has to be structured, as shown in Figure 5.3. The whole response is the physical layer payload, and the contained medium access control (MAC) payload controls the end device's access to the LoRaWAN® network. Constructing this message has to be implemented by the agent. The differences between LoRaWAN® `1.1` and `1.0.X` are minuscule for the join response and do not affect the construction of the message. Which of the versions the response is intended for can be indicated with the first bit of the `DLSettings` parameter. It must be noted that the payload and message integrity code (MIC) are encrypted with the AES *decrypt* operation. This is done to allow end devices to only implement AES encryption. They use it to encrypt their regular messages and "decrypt" the join response by encrypting the *decrypted* plaintext.

The resulting physical payload is sent by the SSI agent to the network server using a JSON-encoded message. Using parts of the response's payload and the application key the end device and the agent generate the application session key and the network session key. The end device uses those keys for any further messages according to the LoRaWAN® protocol. The agent server stores the application session key to pass it to the application when requested.

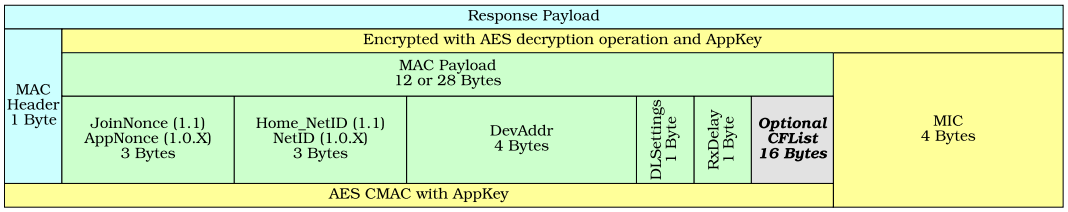

Figure 5.3: Structure of the LoRaWAN® join response message returned by the join server and passed to the end device [121, 122]

To get the current application session key, the application needs to query the join server. The endpoint for this is specified in the interoperability configuration of Listing 5.5 if an external join server – like the SSI agent server – is used. The application session key is returned as part of a JSON-encoded message via HTTP, as shown in Listing 5.6. Transmission of this message should be secured using TLS or an additional key encryption technique. The application receiving the data via TTS must also request this key. The answer is also JSON-encoded and displayed in Listing 5.6.

```
{
  "SenderID": "ABCD000000000000",
  "ReceiverID": "000000",
  "PHYPayload":
    ↪ "20B762B56E4E19F306D2FD205F6AE25A372285F9036E36A31B9A6533B5CA2B22F8",
  "Result": {
    "ResultCode": "Success"
  },
  "Lifetime": 1000000,
  "SessionKeyID": "9B77073050F7C56FF81B791C619EC2D7"
}
```

Listing 5.6: JSON-encoded response of the SSI agent to the TTS interoperability server's forwarded join request

```
{
  "SenderID": "ABCD000000000000",
  "ReceiverID":"000000",
  "Result": {
    "ResultCode":"Success"
  },
  "KEKLabel":"",
  "AESKey":"42078A219C0FFD6C1F8FBC59E2B0CA0B",
  "SessionKeyID":"A33B5DAD0B80615A52FE1B3F5BEF4B0C"
}
```

Listing 5.7: JSON-encoded response of a LoRaWAN® application's application key request to a join server

5.4.1.2 SSI Process

The prototype implemented for the IoT scenario acts as an SSI cloud wallet and stores the device's keys. Integrating into the TTS system allows the SSI-managed devices to join a LoRaWAN® network. Additional VCs managed through the Hyperledger Indy DLT can be used to publish and prove identity attributes of the devices, i. e., their operator, location, or maintenance log. A combination of this setup and the lightweight asymmetric key signatures developed for LoRa® in [70]

can ensure that the messages of an end device can be published with a signature binding it to the device's SSI identity. The resulting process is schematically shown in Figure 5.4.

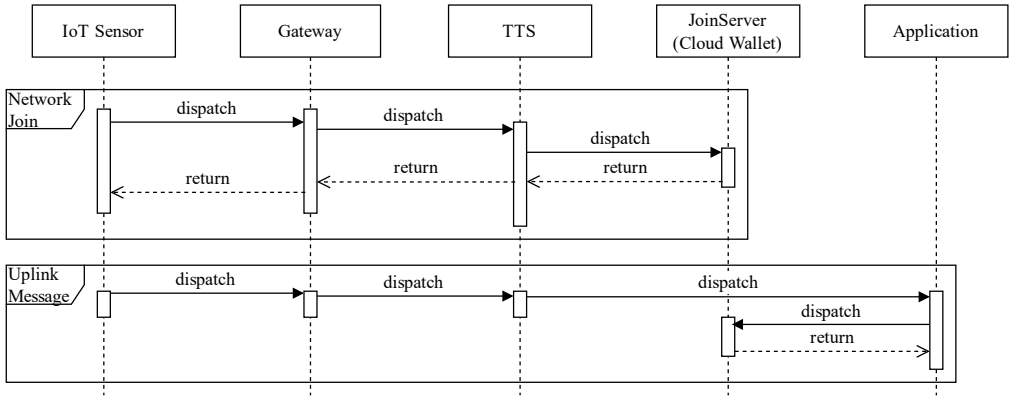

Figure 5.4: Sequence diagram of an SSI-enabled IoT device's measurements via message queuing telemetry transport (MQTT)

5.4.2 Organizational Processes

With SSI, instead of relying on one platform to handle the IoT device's information, this information is managed by the device's operator's SSI wallet. To do so, SSI features need to be added to the regular workflow of TTS. The changes are part of the registration shown in Section 5.4.2.1,and the modifications for data processing steps are described in Section 5.4.2.2.

5.4.2.1 Registration

For use with TTS, the end devices have to be registered with their device IDs to an application within TTS. To utilize the SSI features, the device must be configured to use an external join server, allowing the TTS to use the SSI cloud agent as a join server. The device's data can then be associated with the VCs managed through the cloud agent's API.

5.4.2.2 Data Processing

The data captured by the TTS network and forwarded to the application can be processed as regular with TTS. Any VCs associated with the data, either by the end device itself or any intermediary handler, can be verified using the workflow from Hyperledger Indy.

5.4.3 Summary

The integration of the developed SSI cloud wallet as a LoRaWAN® join server shows the interaction of SSI with existing IoT network infrastructures. It shows that existing protocols can be retrofitted with SSI technology without introducing breaking changes or requiring major updates to the network. Instead, a seamless transition and dual-stack operation can be achieved by routing SSI and non-SSI authentication requests to different join servers using different `join-euis`.

While this prototypical implementation shows SSI integrated with the LoRaWAN® join process, it does not include the handling of VCs, as this is already included in the web application and eID implementation. Using VCs would work the same as in the other implemented scenarios.

5.5 Electronic Identification (eID)

Strong and secure IAM is especially important for eGovernment and eID applications. As described in Section 2.5.6 and shown with the overview of existing approaches in Section 3.5.1, current solutions are very country-specific and sometimes require special smart cards. SSI can unify the landscape in an international context that necessarily exists on the Internet.

The prototype described in this section describes a real-world use case discussed with the German ministry for digitalization in Bavaria *Bayerisches Staatsministerium für Digitales (StMD)* within the context of the projects DISKURS and DISPUT. The demonstrator developed for those projects recreates a use case for cross-state eID. Within this use case, a fictional Bavarian resident, *Paula*, wants to move to the state of *North Rhine-Westphalia* (NW). To be able to work there as a teacher, she must provide proof of her successful state examination. In the workflow using FIM, Bavaria and NW both have a state-run IdP and SP, combined as a *dipole* that residents of one state can use to access the eID services of the other. Figure 5.5 showcases this original workflow.

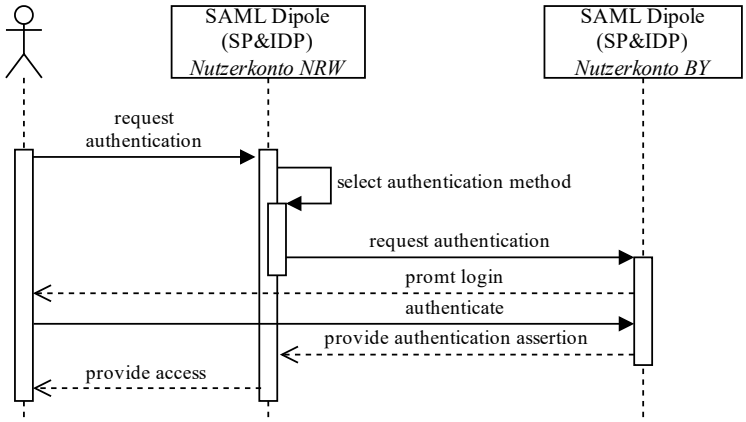

Figure 5.5: Workflow for cross-state eID using an FIM dipole architecture

The main goal of the demonstrator is to showcase how this workflow would change with the introduction of SSI. This has been achieved by recreating the use case in a dedicated environment based on Hyperledger Indy and Aries. Additionally, a mobile application has been developed that interfaces with the DLT components through *Hyperledger Aries Cloud Agent - Python* (ACA-PY) and shows what a Bavarian eID smartphone app could look like. Further technical components are described in Section 5.5.1, and organizational challenges found during the project are discussed in Section 5.5.2.

5.5.1 Technical Components

Similar to the prototype for IoT, the eID prototype is based around the DLT of Hyperledger Indy. It uses the ACA-PY library to connect to the ledger and manage a cloud wallet. In this prototype, an application for Android smartphones is developed to serve as a user interface.

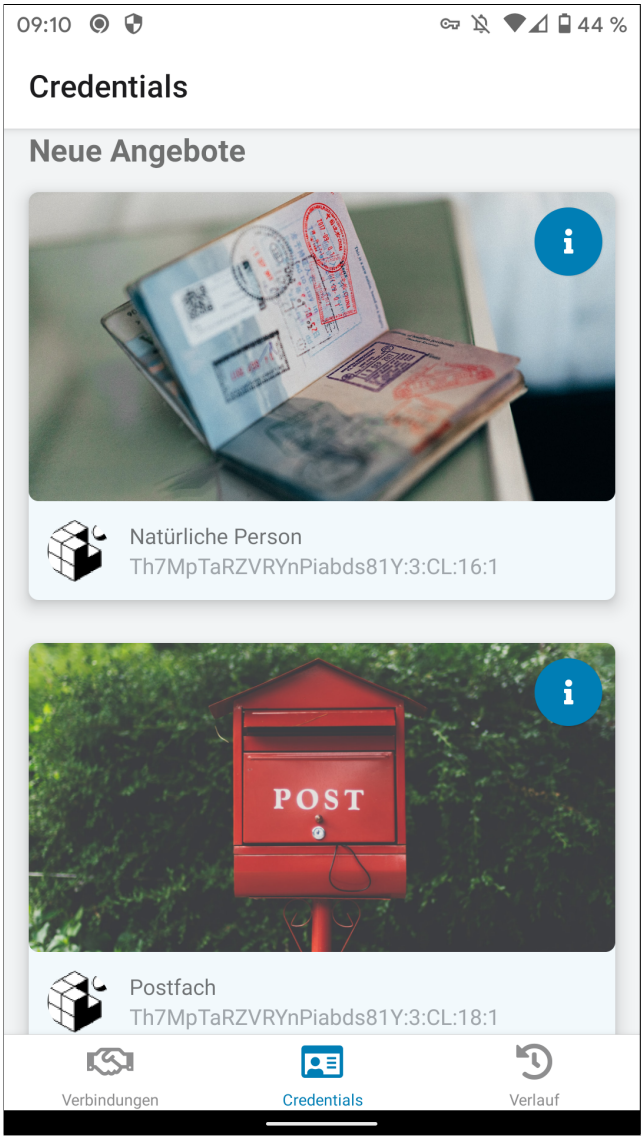

Figure 5.6: Screenshot of the eID prototype wallet application

A screenshot of the app's credential screen is shown in Figure 5.6. The app's text is in German, as the project was done in Germany. The app implements all necessary operations to showcase the SSI workflow of connecting to other entities by scanning a QR code. All interactions are performed using the API of a private DLT instance running Hyperledger Indy. The app is developed using the React Native[10] framework and tested on an Android smartphone. Due to React Native, the app could also be directly used on iOS.

The app is programmed by defining the seven necessary UI screens as React Native `.tsx` files. Those contain the logic for the screens' primary functions as Java code and the layout defined as HTML. Each of the three main tabs for displaying connections, credentials, and history is defined in its own `.tsx` file. It query the respective ACA-PY API to display the results as individual cards.

Additionally, screens for displaying details of connections and credentials are defined to show details for a specific credential or connection. Those screens also offer methods for deleting the entry. The two remaining screens are the camera screen used to scan and process QR codes, and an error screen displayed if a requested screen can not be loaded.

The wallet's backend is provided by the ACA-PY API. Using a cloud agent's API to perform SSI operations is the wallet prototype's biggest limitation to being a genuine SSI wallet. However, the frontend matters more in this case, as it should be usable for basically all citizens with access to smartphones, and the backend for an SSI eID system should – similar to cryptographic operations – utilize an SSI library and special protected storage like TPMs, anyway.

The demonstrator's server components are managed through a *Docker Compose* file. Figure 5.7 provides an overview of the services. The containers are shown as components with their exposed ports as circles and mounted volumes as file-shaped rectangles. All containers are connected via the backend network displayed as a pentagon.

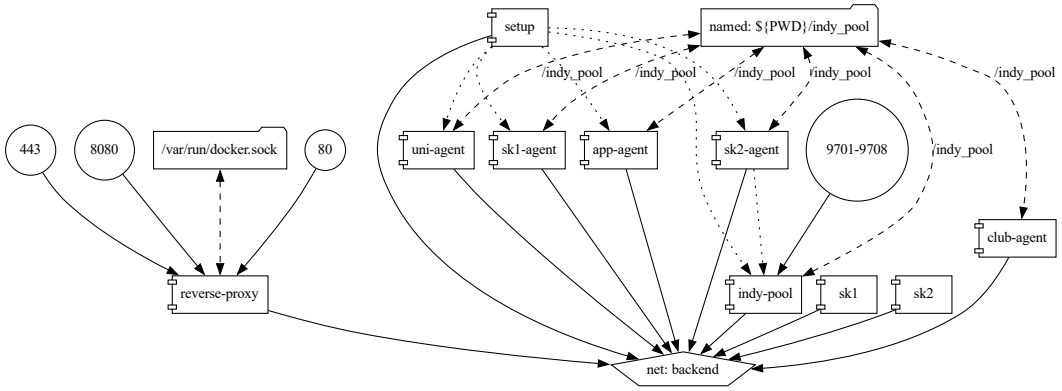

Figure 5.7: Overview of the dockerized setup of the eID demonstrator environment

The DLT components are implemented using a Hyperledger Indy pool. A local demonstration and testing network of four validator nodes started from a customized genesis file, is used to run this pool. The pool's genesis file is required by the agents, which can access it via a common, read-only mounted volume. Web services for the two issuers have been implemented using *Django*[11] and are started as individual Docker containers. The RP's front end utilizes ACA-PY agents for all SSI operations. So for each RP, a corresponding agent container is also started. The

[10] `https://reactnative.dev`
[11] `https://www.djangoproject.com`

issuers do not have a web frontend for this demonstration. Their credentials are issued automatically at the start of the network by a helper setup container. Each issuer, however, is represented by their agent, also run as an individual Docker container.

All of those containers are connected using a *Traefik*[12] reverse proxy. It is used to route HTTP requests to the right services, as all services' hostnames are resolved to localhost using `/etc/hosts` entries.

5.5.2 Organizational Processes

While the technical details of the eID implementation do not differ much from the other scenarios, the organizational restraints are more pronounced. As described in Section 3.5, eID is governed by national and international legal frameworks that need to be considered and conformed to. While integration into organizational processes was not part of the DISPUT project, some preparations have been discussed and implemented. Those affect the management of VCs, as shown in Section 5.5.2.1, and security management, as described in Section 5.5.2.2.

5.5.2.1 Verifiable Credential Management

As discussed for IAM federations in Section 3.2, larger federations and inter-federations suffer from a more and more reduced set of commonly available attributes. The eID scenario has the advantage over self-organizing federations because the core attributes necessary are described by legislation. On an international scale, the EU also describes required attributes for cross-border eID, thus providing both issuers and RPs with a well-defined set of usable attributes.

For this prototype, the attribute definitions given by the German BSI were translated to credential definitions usable with Hyperledger Indy using the process described in Section 4.5.2. The source attribute definitions were given as XML schema definitions (XSDs) and were translated into the corresponding JSON-LD syntax. An example of a translated definition is shown in Listing 5.8.

The original XSD's translation into a syntax for VCs allows both attribute descriptions to be as similar as possible and aids in simplifying the translation of actual attributes from one system to the other. Migration or a side-by-side deployment of eID systems is, therefore, easier to setup.

5.5.2.2 Security Management

The decentralized security management process envisioned for SSI is described in Section 4.3.8. As the prototype is only used for demonstration, integration with security management processes was not part of the project and could not be tested. However, the restricted participants of an eID deployment would allow for easier security management than a completely open IAM system, as the participants are primarily government agencies bound by strict regulations.

5.5.3 Summary

The prototype developed for eID was able to show the basic suitability of SSI. Implementing the Hyperledger Indy and Aries frameworks was straightforward, and integration with prototypical web services was relatively easy. This kind of connection of an agent system to a service is close to what would be implemented in an actual deployment. In a live scenario, however, the security management aspect needs to be focused. Primary actions to be taken include securing the agent using

[12] `https://doc.traefik.io/traefik`

```
{
  "@context": {
    "@version": 1.1,
    "@protected": true,
    "id": "@id",
    "type": "@type",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "eid": "http://bsi.bund.de/eID#",
    "NaturalPerson": {
      "@id": "http://bsi.bund.de/eID#NaturalPerson",
      "@context": {
        "@version": 1.1,
        "@protected": true,
        "id": "@id",
        "type": "@type",
        "Place": {
          "@id": "eid:Place",
          "@context": {
            "@version": 1.1,
            "@protected": true,
            "id": "@id",
            "type": "@type",
            "street": { "@id": "eid:PlaceStreet", "@type": "xsd:string" },
            "city": { "@id": "eid:PlaceCity", "@type": "xsd:string" },
            "state": { "@id": "eid:PlaceState", "@type": "xsd:string" },
            "country": { "@id": "eid:PlaceCountry", "@type": "xsd:string" },
            "zipcode": { "@id": "eid:PlaceZipcode", "@type": "xsd:string" }
          }
        },
        "DocumentType": { "@id": "eid:DocumentType", "@type": "xsd:string" },
        "ICAOCountry": { "@id": "eid:ICAOCountry", "@type": "xsd:string" },
...
```

Listing 5.8: Part of the resulting JSON-LD syntax after translating an XSD attribute definition for VCs

an intrusion detection system (IDS) and defining a process for updating the agent and the corresponding components. During the testing period, multiple vulnerabilities were discovered, patched, and updates needed to be done. The resulting service interruptions would need to be avoided in a real setup.

The wallet app implemented and used also lacks basic security features necessary to protect the user's privacy. A proper implementation would need to utilize native platform features to store confidential data in secure environments. Additionally, the UI design needs to be evaluated more thoroughly, as shown by exploiting the human factor of operating wallet apps.

Chapter 6

Evaluation

Contents

After conceptualizing a system for SSI in Chapter 4 and implementing selected essential parts in Chapter 5, this chapter evaluates whether the implementation according to the concept could achieve the major design goals. The concept itself is assessed in Section 4.6, and most requirements are determined to be covered completely.

This evaluation focuses on a technical exploration of the individual implementations in Section 6.1 and the combination of the implemented prototypes in Section 6.2. The latter part is especially interesting for forming the envisioned comprehensive IAM system. Section 6.3 summarizes the evaluation and provides a graphical overview of the fulfillment in Table 6.1.

The evaluation criteria are defined in Section 4.6, where the concept is evaluated. Each requirement can be evaluated as *completely fulfilled*, *fulfilled*, *partially fulfilled*, or *not fulfilled*. To reach *completely fulfilled*, all aspects of the requirement need to be handled, and the prototype must be able to show that it works reliably. A *fulfilled* requirement needs to be implemented in the prototype and must cover most, but not all, edge cases. *Partially fulfilled* is achieved by a requirement implemented in the prototype but missing relevant functionality. Requirements that are not implemented or do not work as expected are classified as *not fulfilled*.

6.1 Prototype Implementations

Chapter 5 describes the implementation of three prototype SSI systems developed for Scenario 1, Scenario 2b, and Scenario 4. The following sections describe how each of those prototypes fulfills the requirements from Section 2.5.

6.1.1 Web Applications

The first prototype implemented utilizes a non-DLT system to build SSI with web technologies to explore its suitability for Scenario 1. The following paragraphs explain the chosen fulfillment levels in detail. Table 6.1 shows this description summarized.

As prescribed by the concept, the implementation follows the requirements of **Security by design** (SEC5) and **Security by default** (SEC4), as well as **Privacy by design** (DAT2) and **Privacy by default** (DAT1). The requirements for security are fulfilled by utilizing standard libraries for encryption and signatures. However, the implementation knowingly disregards the recommendations and best practices around storing sensitive information in browser extensions, and therefore requirement **Security by design** (SEC5) is downgraded to partially fulfilled. The privacy-related requirements are fulfilled completely, as the implementation does not share any information without the user's explicit consent and protects transmitted information using state-of-the-art technology. As a result, the implementation partly fulfills the requirement **GDPR** (DAT3). To attest a higher fulfillment level, a legal evaluation would be necessary. For SSI, however, no such evaluation exists to date.

Requirements **Identity provisioning** (SAT4), **Identification** (SAT3), and **Authentication** (SAT1) are completely fulfilled using WebAuthn. Each entity uses two DIDs for identifying itself to and being identified by others. A pair of these local and remote DID are combined into a unique connection. The step of establishing credentials – described by requirement **Credential establishment** (INF1) – is also completely fulfilled by WebAuthn. Related to **Identity provisioning** (SAT4) is the requirement **Multiple identities** (DAT4). It can be completely fulfilled by using multiple profiles in the browser, thus separating the extension's local storage for identity data.

The use of WebAuthn allows the system to build on standard authentication technology implemented by many web browser providers. Using HTTPS for the communication between the browser extension and the website allows for requirement **Mutual authentication** (SEC3) to be fulfilled.

The next basic action of IAM, authorization, is described by requirement **Authorization** (SAT2) and built with its own implementation. This implementation entails both the server's and the user's browser side. The implementation is integrated into the browser as an extension for Chromium-based web browsers and stores all necessary key material and VCs. As a result, the requirement is completely fulfilled.

The demonstrator web application's implementation also shows how to enforce access controls, which are described by requirement **Access controls** (SEC1). This can be done by utilizing the implemented library to check the validity of the VCs and fulfills the requirement. The implementation is documented in Chapter 5.3, fulfilling requirement **Documentation** (INF2).

As all identity data is stored in the browser extension's local memory, requirement **Identity de-provisioning** (CON1) can be achieved by deleting this storage. This step is not entirely fulfilled as it could be improved by providing methods for removing individual parts of the identity.

To facilitate synchronization of the identities between multiple browsers and to prevent data loss, requirement **Credential recovery** (CON2) can be achieved by activating a synchronization provider. This synchronization stores the browser's data, including the extension's data, in the cloud. Such services are available from many browser manufacturers.

Requirements **Accessibility** (ROB2) and **Approachability** (ROB3) are fulfilled by using the browser extension as a very unobtrusive way of providing SSI in the browser. Other parts of the login workflow are only modified slightly and resemble the use of a password safe. As a result, requirement **Usability** (ROB4) is completely fulfilled. Application of the prototype within an experimental setup also shows its reliability, fulfilling requirement **Reliability** (ROB1).

The requirement **Credential revocation** (SEC2) is not implemented, as all design options for this requirement were determined to be unreliable by the concept. The requirement for **Multi-factor authentication** (SEC6) is not fulfilled by WebAuthn, as the user's authenticator is viewed as the only authentication factor, even if it is protected by additional authentication, e. g., a personal identification number (PIN) or fingerprint.

6.1.2 IoT

The prototype developed for Scenario 2b integrates an SSI agent into the existing LoRaWAN® implementation TTS. A visual representation of the prototype's fulfillment of the requirements is depicted in Table 6.1. The following paragraphs explain the results of the evaluation.

The IoT implementation extends the existing TTS and adds SSI features. The architectures of TTS and LoRaWAN® are designed with requirements **Privacy by design** (DAT2), **Privacy by default** (DAT1), and **GDPR** (DAT3), as well as **Security by design** (SEC5) and **Security by default** (SEC4) in mind. Respective descriptions of TTS and LoRaWAN® are provided in Section 3.4.1. The prototype extends the TTS with SSI as an additional method of authenticating IoT devices. Requirements DAT2 and DAT1 are marked as "not applicable" in this evaluation, as the implementation does not directly affect users' privacy guarantees. Instead, requirement DAT3 is marked as fulfilled, as the implementation does not rely on users' personal information, and the managed identities are device identities.

An SSI identity's lifecycle begins in the prototype by creating the device in the TTS web-based interface and marking it as an externally managed device. Devices are identified by their standard LoRaWAN® device ID, as specified in and completely fulfilling requirement **Identification** (SAT3). Any join requests received by TTS are routed to the prototype's join server to handle exchanging the necessary keys. This completely fulfills requirement **Identity provisioning** (SAT4). Within the LoRaWAN® (join) protocol, each device can only have one identity. As a result, requirement **Multiple identities** (DAT4) cannot be fulfilled.

The prototype's join server completely fulfills requirement **Authentication** (SAT1) by authenticating the device via its application key. The keys for authentication are established by the user registering the device in the TTS and by connecting the prototype SSI agent to an SSI network. Deploying the initial keys needs to be done manually and – while not especially scalable – completely fulfills requirement **Secure setup** (SEC10). However, this procedure only partially fulfills requirement **Credential establishment** (INF1), as two independent configuration steps and locations are involved. Removing keys as part of requirements **Identity de-provisioning** (CON1) and **Secure de-provisioning** (SEC9) is fulfilled, as removing the keys from the SSI system immediately removes the sensor from the network.

Requirement **Credential recovery** (CON2) is out of scope for the prototype and the scenario. The sensor can only be issued new credentials if there is suspicion that the previous credentials have been compromised or the sensor has lost data, including the keys. Requirement **Credential revocation** (SEC2) is not implemented in the prototype. A possible solution is redeploying new keys to the device

and SSI backend. While using secure keys is sufficient in most situations, the prototype cannot easily be extended to utilize MFA and cannot fulfill requirement **Multi-factor authentication** (SEC6).

This setup fulfills requirement **Access controls** (SEC1), but only allows for binary decisions. Different levels of access rights are not possible due to the design of the LoRaWAN® join protocol, which only discerns between allowing and denying access to the network. Similarly, requirement **Authorization** (SAT2) is fulfilled but limited to yes/no decisions. Performing authentication or authorization offline, as described by requirement **Offline authNZ** (ROB5), is not in this prototype's scope, as it only handles the device's join process into the network. The prototype does not implement requirements **Access delegation** (SAT6) and **Delegation parameters** (SEC8). The LoRaWAN® protocol includes limited delegation to other network providers from version 1.1. This roaming functionality also partially fulfills requirement **Transitive trust** (ROB8), even though it is a LoRaWAN® feature independent of the prototype's implementation. The same process also allows requirement **Trust establishment** (SAT5) to be partially fulfilled.

The prototype cannot fulfill requirements **Digital twin** (CON3), **Digital identification** (INF3), **Physical identification** (INF4), and **Product specification** (INF5). Each of those must be handled at the application level, which the prototype does not include in favor of only handling joining devices to the LoRaWAN® network. The focus on LoRaWAN® and TTS also prevents requirements **Platform independence** (ROB7) and **Communication protocol independence** (ROB9) from being fulfilled.

However, the prototype successfully shows that it is possible to use SSI in a resource-efficient (requirement **Resource efficiency** (ROB10)) and scalable (requirement **Scalability** (ROB6)) way. Requirements **Tamper-evident** (SEC7), **Tamper-resistant** (SEC11), and **Protected application storage** (DAT5) can only be partially fulfilled by the prototype. The used devices do not contain specialized hardware TPM to protect their keys. However, outages can be detected within the network and successfully manipulating the devices to impersonate them is difficult. LoRaWAN® can fulfill requirement **Correlation resistance** (DAT6).

Due to the limited feature set of the implemented prototype, requirement **Usability** (ROB4) can only be partially fulfilled. More complete fulfillment would require a more straightforward configuration of devices and identities, e. g., through a dedicated web interface. Similarly, requirements **Accessibility** (ROB2) and **Approachability** (ROB3) can only be partially fulfilled. Documentation of the prototype is provided in Section 5.4 to fulfill requirement **Documentation** (INF2). Additionally, the reliability of the prototype was shown in the experimental setup, which fulfills requirement **Reliability** (ROB1).

6.1.3 eID

Developed to fulfill the requirements of Scenario 4, the prototype described in Section 5.5 utilizes DLT and state-of-the-art SSI technologies to showcase SSI in an eID setting. The complete list comparing the scenario's requirements and the prototype's implementation is depicted in Table 6.1. The following paragraphs describe the reasons for the respective evaluation.

In an IAM system with high-quality personal information, like eID, completely fulfilling requirements **Privacy by design** (DAT2), **Privacy by default** (DAT1), and **GDPR** (DAT3) is essential. The prototype does this by utilizing the well-researched backend of Hyperledger Indy / Aries. Similarly, the prototype fulfills requirements **Security by design** (SEC5) and **Security by default** (SEC4). Due to the backend's

new and constantly changing nature, those requirements cannot be completely fulfilled. A more stable version of the backend software and protocol would be necessary.

Identification, as demanded by requirement **Identification** (SAT3), happens via connections between two entities, facilitated through pairwise DIDs. This completely fulfills the requirement. The process of provisioning an identity (requirement **Identity provisioning** (SAT4)) is also completely fulfilled, as any individual can create their digital identity using a smartphone app. Multiple identities are possible, but support from applications to handle those are limited and not shown in the prototype. Requirement **Multiple identities** (DAT4) is therefore only partially fulfilled. The two eID-specific requirements, **Identity data set** (INF10) and **Identity data set matching** (SAT8), are not fulfilled by the prototype. Implementing those would be possible in any actual use of a setup similar to the prototype by ensuring that during issuing of an identity's attributes, all required attributes are present. Also, the eID-specific requirement **Once-only** (CON6) can be fulfilled by SSI if legislation agrees.

Requirement **Authentication** (SAT1) is completely fulfilled by the protocol chosen for the prototype. The steps to establish connections also completely fulfills requirement **Credential establishment** (INF1) and partially fulfills requirement **Off-the-record (OTR)** (SEC12). Not possible, however, is the usage of MFA so requirement **Multi-factor authentication** (SEC6) cannot be fulfilled. Users' authorization is done according to the VCs' attributes and values. This completely fulfills requirement **Authorization** (SAT2) and also completely fulfills requirement **Access controls** (SEC1). Authorization decisions can also include LoAs as described by requirement **Level of assurance** (INF11). While the prototype does not explicitly use LoAs and a decentralized system would need overlying infrastructure to manage LoAs, LoAs can be expressed as VCs and the requirement INF11 can be partially fulfilled. Requirement **Trust service providers** (SEC13), which can only be partially fulfilled by issuing trust certificates as VCs, could also interface with specifying LoAs.

Connections formed between entities can be broken by deleting the keys to the pairwise DIDs. Further, VCs are stored within the user's wallet and can be safely deleted there. Both points together fulfill requirement **Identity de-provisioning** (CON1). Handling lost credentials, as described by requirement **Credential recovery** (CON2), is impossible in the prototype setup and not fulfilled. Comparable to the physical world, a lost wallet must be replaced by re-issuing all contained VCs. Requirement **Credential revocation** (SEC2) is discussed within the Hyperledger Community and is possible in some instances, but the prototype does not showcase or utilize those features and does not fulfill the requirement.

The prototype could show that the system is usable and fairly easy to understand. This completely fulfills requirement **Usability** (ROB4). It is also accessible, to the degree that a smartphone is needed, and fulfills requirements **Accessibility** (ROB2) and **Approachability** (ROB3). Within the prototype setup the system could also show that it can perform reliably and fulfills requirement **Reliability** (ROB1). Large-scale tests of similar systems are underway and seem to support this evaluation. Utilizing a well-documented protocol for this prototype, the protocol's description and the own implementation's details fulfill requirement **Documentation** (INF2).

Not implemented in the prototype and still an open problem is described by requirement **Message delivery services** (SAT9). Delivering messages to entities within the SSI system and proving their delivery is not widely implemented. The requirement cannot be fulfilled.

6.2 Combination of the Implemented Prototypes

Individual IAM solutions are plentiful. The prototypes described in Chapter 5 for web-based, IoT, and eID scenarios are also mostly standalone systems. Ideally, the prototypes could be combined to form a large-scale identity ecosystem. A successful combination of the scenarios can show the bridging potential of SSI. This section evaluates the suitability to achieve this, following the concept from Chapter 4.

For this purpose, a combined scenario is assembled. In this scenario, an IoT sensor publishes data via an IoT WAN to a web-based platform. The platform stores and processes the data and provides a web-based interface for users to download the data. This interface is accessible to users after authenticating with an eID.

The challenges of combining the prototypes to cover this combined scenario are described in more detail in Section 6.2.1. Afterwards, solutions using the concept's components CLS and TGW are presented in Section 6.2.2 and Section 6.2.3. The theoretical evaluation of this combined system is performed in Section 6.2.4.

6.2.1 Challenges

With the prototypes implemented individually, the main challenge is interfacing the three prototypes' VCs:

- On a technical level, the **syntax** of VCs differs between the eID prototype's DLT-based VC and the web platform's X.509 certificate-based VC.
- Within each of the prototype systems, the **semantics** between VC attributes can differ.
- From an organizational point of view, **trust** between the parties within the respective scenarios and their prototypes cannot be guaranteed.

A graphical representation of this combined scenario and the three main challenges is shown in Figure 6.1. The authentication of the IoT devices via SSI needs to be associated with the data they produce outside the prototype's TTS. To extend the scope of where the IoT device's identity can be shown to be the origin of its data, a bridge between both prototypes is required. This bridge needs to be able to (unidirectionally) transform syntax and adapt semantics. A process that requires exceptional trust in the correctness and confidentiality of both sides.

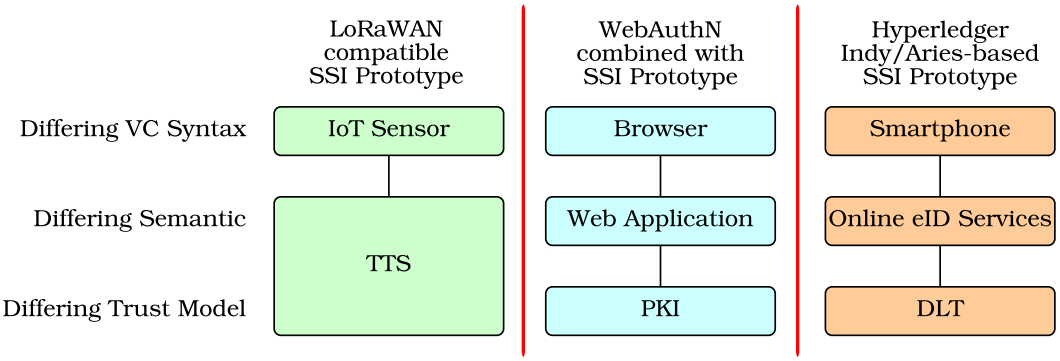

Figure 6.1: Overview of the combined evaluation setup

Between the eID prototype and the web app prototype, similar conversions are necessary. However, as both systems have fully functioning SSI wallets and VCs, there is no need for an entity to join data and identities. This part only requires

syntax and semantics to be aligned for use within the other prototype's system. Trust in this conversion being done correctly is also very important, but easier to verify, with VCs available on both sides.

Those challenges are alike the problems described for FIM in Section 3.2: Similar but differently implemented IAM systems need to communicate with each other to allow users to move between them. As FIM and its challenges are part of the inputs to the design of the concept for SSI described in this work, it should account for those challenges and provide components to solve them.

6.2.2 Use of the Credential Localization Service

According to the concept, the challenge of connecting the web-app prototype to the eID prototype can be tackled by introducing a CLS. The CLS, as described in Section 4.3.6, is responsible for providing SPs with the rules necessary for translating credentials to the target platform. In the case of the combined scenario, both SSI prototypes use differing concepts for representing and storing VCs. As a result, the CLS needs to be able to offer translations between those different kinds of VC, namely VCs from Hyperledger Indy and JWT. To do so, the following steps need to be passed by the SP, which in this case is the web app:

1. **Validate the VC in the source system.** Before continuing, the web app must ensure that the VC the user provides is valid. To do so, the web app must be part of the eID prototype's SSI system.
2. **Get a suitable localization rule from the CLS.** With the source VC validated, its contents must be adapted to "fit" the local SSI system.
3. **Localize the VC's contents.** Using the localization rule obtained from the CLS, the web app can convert the VC's assertions to the corresponding JWT assertions.
4. **Create a VC in the target system's syntax and add the localized assertions.** A new JWT VC must be built to be processed by the local SSI system.
5. **(Self-)sign the VC.** As the new ephemeral VC, in JWT form, must pass the full validation of the local SSI system, it needs to be signed. The keys for this signature can be kept completely local.
6. **Pass the VC to the internal IAM system.** The resulting JWT VC should be processed as any other by the local SSI system.

The concept's description focuses on Step 2, the localization rules, and their exchange. Conceptually, this step is evaluated in Section 4.6. To evaluate the suitability of the CLS in this scenario, the other steps need to be included in the evaluation in Section 6.2.4.

6.2.3 Use of the Trust Gateway

The challenge of bringing the IoT prototype's data into the web app is solved using a TGW. As described in Section 4.3.7, the TGW can be used to import data from the TTS LoRaWAN® system. A TGW provides an additional SSI agent and wallet acting as a proxy. This is required in the combined scenario because IoT devices cannot directly issue VC for their published data. Implementing IoT device-issued VC is considered and shown in [70] but not included in this scenario to keep full LoRaWAN® compatibility.

The TGW is connected to TTS as an application, on the one hand. This way, it can receive the IoT sensor's data directly via an API and obtains the application key that ensures within the TTS that the data is authentic (see Section 5.4.1).

On the other hand, the TGW can utilize the LibWebSSI library developed for the web app prototype to provide each IoT sensor with its own identity. These identities can then be used to create JWT VCs for the IoT sensor's data and upload it to the web app.

In between those two connectors, the TGW has to perform similar syntactic and semantic conversions as the CLS in Section 6.2.2. On its own, this step is evaluated in Section 4.6. Together with the complete workflow of the combined scenario, the evaluation is detailed in Section 6.2.4.

6.2.4 Evaluation

Introducing CLS and TGW to the prototypes does not hinder the fulfillment of most requirements. The following paragraphs discuss which requirements are *possible* and which are *impossible* to fulfill in the scenario of combining the implemented prototypes. The results are also added to the previous evaluation results shown in Table 6.1.

Requirements **Privacy by design** (DAT2) and **Privacy by default** (DAT1) are possible. The CLS and TGW are new software components but are as privacy-preserving as possible. The same is true for requirements **Security by design** (SEC5) and **Security by default** (SEC4). As there are no legal challenges or decisions for requirement **GDPR** (DAT3) yet, it is undecided if it can be fulfilled.

The addition of the CLS does not affect requirements **Identification** (SAT3), **Identity provisioning** (SAT4), or **Multiple identities** (DAT4). Neither does the TGW, so the requirements are still possible to be fulfilled. Requirements **Identity data set** (INF10), **Identity data set matching** (SAT8), and **Once-only** (CON6) originate from the eID scenario and can be fulfilled with the CLS. Those requirements can therefore be fulfilled in the combined scenario. If the TGW were used for bridging eID to the web app, requirement **Once-only** (CON6) would not be possible to fulfill, as the TGW essentially stores another copy of the user's attributes in another SSI wallet.

Fulfilling requirements **Credential establishment** (INF1), **Authentication** (SAT1), and **Mutual authentication** (SEC3) is still possible. The additional components do not change this. The TGW, however, requires establishing additional credentials. Retiring identities, described in requirements **Identity de-provisioning** (CON1) and **Secure de-provisioning** (SEC9), also remains possible.

Necessities for IAM described through requirements **Access controls** (SEC1) and **Authorization** (SAT2) are possible to implement. They are also part of the implemented prototypes. Impossible, however, is the fulfillment of requirement **Offline authNZ** (ROB5). This is because querying for CLS rules and communicating with the TGW requires network connections to those third parties. Delegating access, as described by requirements **Access delegation** (SAT6) and **Delegation parameters** (SEC8), is impossible and not implemented in any prototype. Important for the success of the system, requirements **Trust establishment** (SAT5) and **Transitive trust** (ROB8) are possible, particularly because CLS and TGW can be used to broaden the trust network. To improve building this trust network, it is also possible to implement requirements **Level of assurance** (INF11) and **Trust service providers** (SEC13). Equally possible to implement is the requirement **Platform independence** (ROB7) and **Communication protocol independence** (ROB9).

The addition of new components always risks bloating the system. In this case, requirements **Resource efficiency** (ROB10), **Scalability** (ROB6), and **Correlation resistance** (DAT6) are still possible to fulfill. Especially, ROB6 can be improved by

allowing different SSI systems to work together. Similarly, requirements **Usability** (ROB4), **Accessibility** (ROB2), **Approachability** (ROB3), and **Reliability** (ROB1) remain possible to achieve.

The remaining requirements have not been evaluated for the combined prototype's scenario for the following reasons. Requirements **Credential recovery** (CON2) and **Multi-factor authentication** (SEC6) have not been implemented for any prototype. As a result, they cannot be shown in the combined scenario. Requirements **Digital twin** (CON3), **Digital identification** (INF3), **Physical identification** (INF4), **Product specification** (INF5), **Protected application storage** (DAT5), **Tamper-evident** (SEC7), **Tamper-resistant** (SEC11), and **Secure setup** (SEC10) are IoT-specific and thus not evaluated for this combination of prototypes. They also have not been implemented for the IoT prototype, as they are more related to the application level than the IAM operation. Requirement **Documentation** (INF2) cannot be evaluated as there is no actual implementation and documentation for this scenario. The concept deliberately chose to exclude requirement **Credential revocation** (SEC2), so it is impossible to implement using this concept. Similarly, requirement **Message delivery services** (SAT9) is not evaluated, as it is also excluded from the concept and implementations. Requirement **Off-the-record (OTR)** (SEC12) is not evaluated, as it is not part of each prototype.

6.3 Summary

The evaluation performed in this chapter shows that the three standalone prototypes can fulfill many of the requirements gathered from the initial scenarios. To summarize fulfillment of requirements, any fulfillment (i. e., *completely fulfilled*, *fulfilled*, or *partially fulfilled*) is counted as "fulfilling" the requirement. The nuances are described in the respective evaluations above and displayed in Table 6.1. Additionally, only requirements not classified as *not applicable* are counted. Using this simplification, the following fulfillment statistics are achieved:

- The concept (Section 4) fulfills 24/24 essential, 30/31 important, and 1/2 optional requirements.
- The web app prototype (Section 5.3) fulfills 13/14 essential, 5/7 important, and 1/1 optional requirements.
- The IoT sensor network prototype (Section 5.4) fulfills 14/17 essential, 12/19 important, and 0/1 optional requirements.
- The eID prototype (Section 5.5) fulfills 13/15 essential, 8/11 important, and 1/2 optional requirements.
- The combination of prototypes (Section 6.2) is possible for 16/18 essential, 13/17 important, and 1/1 optional requirements.

The fulfillment of the concept, and the prototypes for the web app and eID scenario are excellent. The prototype for the IoT scenario is lacking fulfillment of multiple essential and important requirements. This is because those additional missing requirements are generally needed for the scenario but are not necessary, and thus not implemented, for the prototype's scenario. Combining the prototypes is possible and can fulfill most of the requirements.

Table 6.1: Evaluation of the requirements' fulfillment in each scenario's prototype, as well as in the combined setting

Category	Requirement	Concept	Web-App Prototype	IoT Sensors Prototype	eID Prototype	Combined Scenarios	Importance
Satisfaction	SAT1: Authentication	*	*	*	*	p	↑
	SAT2: Authorization	*	*	+	*	p	↑
	SAT3: Identification	*	*	*	*	p	↑
	SAT4: Identity provisioning	*	*	*	*	p	↑
	SAT5: Trust establishment	+		~		p	↑
	SAT6: Access delegation	*		-		i	↕
	SAT8: Identity data set matching	+			-	p	↕
	SAT9: Message delivery services				-		↓
Information	INF1: Credential establishment	+	*	~	*	p	↑
	INF2: Documentation	~	+	+	+		↕
	INF3: Digital identification	*		-			↑
	INF4: Physical identification	+		-			↑
	INF5: Product specification	*		-			↕
	INF10: Identity data set	*			-	p	↑
	INF11: Level of assurance	*			~	p	↕
Consistency	CON1: Identity de-provisioning	*	+	+	+	p	↑
	CON2: Credential recovery	+	-	-	-	i	↕
	CON3: Digital twin	-		-			↕
	CON6: Once-only	~			+	p	↕
Security	SEC1: Access controls	*	*	+	*	p	↑
	SEC2: Credential revocation		-	-	-	i	↑
	SEC3: Mutual authentication	*	+	~	-	p	↑
	SEC4: Security by default	*	+	+	+	p	↑
	SEC5: Security by design	*	~	+	+	p	↑
	SEC6: Multi-factor authentication	*	-	-	-	i	↕
	SEC7: Tamper-evident	+		~			↑
	SEC8: Delegation parameters	*		-		i	↕
	SEC9: Secure de-provisioning	+		+		p	↕
	SEC10: Secure setup	+		*			↕
	SEC11: Tamper-resistant	+		~			↕
	SEC12: Off-the-record (OTR)	+			~		↑

Continued on next page

Table 6.1: Evaluation of the requirements' fulfillment in each scenario's prototype, as well as in the combined setting (Continued)

	SEC13: Trust service providers	+			~	p	↕
Data Protection	DAT1: Privacy by default	*	*		*	p	↑
	DAT2: Privacy by design	*	*		*	p	↑
	DAT3: GDPR	~	+	+	+		↕
	DAT4: Multiple identities	*	*	-	~	p	↓
	DAT5: Protected application storage	~		~			↑
	DAT6: Correlation resistance	+		+		p	↕
Robustness	ROB1: Reliability	*	+	+	+	p	↑
	ROB2: Accessibility	+	+	~	+	p	↕
	ROB3: Approachability	+	+	~	+	p	↕
	ROB4: Usability	+	*	~	*	p	↕
	ROB5: Offline authNZ	*				i	↑
	ROB6: Scalability	*		+		p	↑
	ROB7: Platform independence	*		~		p	↕
	ROB8: Transitive trust	~		~		p	↕
	ROB9: Communication protocol independence	*		-		p	↕
	ROB10: Resource efficiency	*		+		p	↕

Key: ↑ essential, ↕ important, ↓ optional, * completely fulfilled, + fulfilled, ~ partially fulfilled, - not fulfilled, p possible, i impossible, not applicable

Chapter 7

Conclusion

Contents

Security and scalability of IAM systems are hindered by username/password combinations being the prevalent method for identification and authentication. SSI promises to allow decentralizing IAM to achieve both security and scalability. This work explores SSI for various scenarios and shows how to implement SSI systems for three exemplary settings. To highlight the most critical contributions, Section 7.1 provides a summary of the previous chapters. Section 7.2 uses the initial research questions posed in Section 1.2 and provides succinct answers and references to the relevant sections. Last but not least, Section 7.3 discusses topics suitable for further research.

7.1 Recapitulation

IAM presents multifaceted challenges in system design, implementation, and operation. Those challenges are often technical, especially when scaling systems to larger user counts or requiring interoperability between systems. Organizational challenges arise once the technical solutions are found. This work describes SSI, a new IAM concept, and its design, implementation, and application in various scenarios. The focus of this description is technical but organizational challenges are considered, where they are expected or presumed. This section revisits the chapters, summarises their content, and highlights exciting discoveries, new contributions, and potential applications beyond this work.

Developing and describing a new iteration of IAM systems requires careful examination of existing systems and understanding of their strong points and deficits, as well as understanding changed or new requirements of current applications of IAM. To gather the most relevant requirements for a broad spectrum of applications, **Chapter 2** examines existing IAM concepts, namely CIM, FIM, and SSI. On top of those existing systems, application scenarios for the Internet, IoT, cloud and edge computing, and electronic identities are described. Each scenario describes a typical use case for its respective domain. However, the IoT scenario is broken up into three sub-scenarios focusing on IoT devices, less capable and more battery-constrained IoT sensors, and complete IoT networks.

- **Scenario 1** describes a fairly generic web application with a text-based publish/subscribe system for registered users. From an IAM point of view, this application is similar to many real-world applications on the Internet today. Therefore, the resulting requirements fit most web applications that require users' sign-in.
- Even though the IoT scenarios cover a similar topic, each of the following scenarios is described as its independent scenario.
  - The first **Scenario 2a** introduces requirements relevant to IoT devices. Those devices are Internet-connected products that can be found everywhere (e. g., in households, businesses, or manufacturing).
  - A specialization of IoT devices in the form of IoT sensors is used as a model for deducing requirements in **Scenario 2b**. Those sensors are heavily restricted in their hardware capabilities, connection speeds, and latency. They are designed to be run on battery power for many months and even up to multiple years. This scenario introduces requirements to handle those constraints in resource usage. Being mindful of resource usage can also be beneficial in the other scenarios.
  - **Scenario 2c** broadens the view from individual IoT devices within their respective environments and finds requirements necessary to build complete networks of IoT devices. Those networks can consist of minimal sensors, more capable devices acting as routers, and gateways connected to the Internet. Organization and communication contribute additional requirements in this setting.
- **Scenario 3** discusses the requirements necessary for cloud and edge computing. Many of the requirements of this scenario are similar to those of the preceding scenarios. However, negotiation of capabilities and monitoring agreements are new aspects that are found to be necessary in this scenario, and supporting those features through IAM design has its own challenges.
- **Scenario 4** explores the realm of eID and introduces requirements set by government agencies and legislation. The eID space is one where SSI concepts can provide many benefits, partly due to the existing systems being difficult to use. However, this scenario also results in the most non-technical requirements, as legislation highly regulates everything. Parts of this scenario are also applied in a project with the German *Bayerisches Staatsministerium für Digitales* responsible for eID in Bavaria and part of leading the German eID initiative.

On their own, the requirements resulting from those six scenarios are too many and lack structure to form a concept and implementation. To fix this and to focus on the most important requirements, they are categorized as either *essential*, *important*, or *optional*. Combined, the scenarios yield **25** essential, **31** important, and **3** optional requirements. This set of requirements is then used throughout this work but provides value beyond. For example, individual requirements can be re-used for a specific scenario, similar to one described here, or as a whole, as a reference for covering a wide range of IAM needs.

Using the determined requirements, **Chapter 3** explores the current state-of-the-art of IAM and SSI in research, implementations, and standardization. As IAM is a very well-researched topic, the requirements are used to narrow down relevant results. To include sources from all relevant aspects, a graphical representation of the analyzed topic is used to visualize its place within the dimensions of identity, access, and management.

- The **identity** dimension is split into device, personal, and organizational identities.
- **Access** describes which identification, authentication, and authorization processes are covered.
- The aspect of **management** is split into the main characteristics of centralized, federated, or self-sovereign.

Most combinations are covered by at least one regarded source. From the number of state-of-the-art descriptions, the following selection of projects – deemed most relevant – are compared to the requirements.

- **FIDO 2** is the project developing the WebAuthn standard, which is implemented by most major browsers and provides password-less authentication on the web. Its wide adaption makes it highly relevant for any authentication system on the Internet. As such, it is used in the prototype for Scenario 1.
- **Mozilla Persona** was a project aimed at bringing decentralized identities to the Internet through browser-based identity stores. This project serves as inspiration for the concept and implementation of the prototype for Scenario 1.
- **Dynamic FIM** is an extension of regular FIM where federations can form dynamic groups with use case-specific rules around required attribute formats, validity, and usage. Some variations of dynamic FIM describe and employ a trust broker component to facilitate trust between those federations. This approach is adapted in the concept to describe the TGW and CLS components.
- **Hyperledger Indy** is the base of many currently operating SSI systems. Indy is a private/permissioned DLT implementation and specialized for IAM operations. Together with the Hyperledger Aries protocol stack, it offers a complete architecture for SSI and is the go-to solution for many successful projects. This architecture is used in the prototype implementation of Scenario 4.
- Inspecting the current **German eID** systems is required to find the challenges hindering its widespread adoption and use.
- For IoT-specific IAM, the protocol **LoRaWAN®** was chosen. It is used for large IoT star networks with very low-power sensor devices reporting data through gateways to the Internet. Its security architecture shows how to manage device identities for extremely low-power devices. The prototype for Scenario 2b adds SSI characteristics to LoRaWAN®.

The output of this chapter is a comparison of existing approaches and technologies with the requirements of Chapter 2. This comparison shows clearly that there is no solution fulfilling all – or even close to all – requirements. For the established IAM concepts, this chapter provides a concise overview of technologies and implementations that can be of value to other publications. The subject of SSI, however, is currently under constant development and evolution and thus may only represent the state-of-the-art for a short time.

The core contribution of this work is described in **Chapter 4**. This chapter shows a concept for building an SSI system specifically designed to be flexible enough to adapt to the scenarios' different characteristics. Additionally, the concept is designed to be interoperable even if adapted to a specific scenario in order to achieve universal interoperability of identity and attribute exchanges.

To provide flexibility, the concept describes its components individually and combines them to form the reference architecture. The components used in the concept are the following.

- The backend used to store metadata, which the IdP or the federation would ordinarily store, is kept in a **distributed ledger**. The metadata describes which identity attributes exist and how they should be interpreted based on the context. As an alternative to DLT, the prototype for Scenario 1 also demonstrates how PKI can be used.
- To store any identity information that could be considered PII, the **wallet** is designed to keep the user in immediate control of this information.
- Communication between entities might not always be possible directly. In those cases, **agents** offer their users a persistent endpoint that can be used to send and receive messages, even if the wallet is not online when the message is sent.
- The **relying party** is the entity receiving and processing VCs that contain the users' attested attributes. Relying parties are sometimes also called SPs or verifiers.
- The **issuer** attests the users' attributes and provides them with VCs. Contrary to the RP/SP/verifier, an issuer differs from an IdP, as they only provide attribute attestation and do not provide an entire identity. In this regard, they are more similar to a fully independent version of FIM AAs.
- Because of the decentralized nature of SSI, the **credential localization service** is required and introduced as a new SSI component to help attributes be understood across different use cases.
- As a gateway to other SSI or non-SSI systems, the **TGW** is a newly described component for SSI. It is vital to increase the chances of adoption by making older IAM systems accessible and enabling SSI without requiring the world to agree on one particular flavor immediately.
- As the last component, the **community** is a more abstract component which is vital to fill the technical framework of the other components with life. The community must also work together to develop the ecosystem further, discuss new proposals, and handle security incidents.

The concept also provides templates for building new SSI systems or migrating existing IAM infrastructure to SSI. Both templates are applied to a subset of the scenarios to determine the best way of making them SSI-capable while using the concept. The scenarios are chosen to cover as many different settings as possible while avoiding unnecessary repetition. The chosen scenarios are used throughout the further chapters and are partially implemented as prototypes.

- The first scenario chosen is **Scenario 1**. As respective components, the DLT is replaced with the Internet-prevalent PKI. It is assumed that issuers are mainly corporations and RPs are any web application. The user's browser is chosen as their wallet application, and possible agents are additional web services. This scenario does not require CLS or TGWs and is managed by a public community.
- As the second scenario, **Scenario 2b** is chosen. It features IoT sensors and uses a private/permissioned DLT. Issuers are sensor manufacturers, and RPs are data processors. The devices' wallets are embedded into the respective sensors. The connection between the sensors and data processors may be augmented by agents, which run as cloud services or simple gateways. CLS and TGWs are included in this scenario and run by specialized service providers. The community is sector-specific, which also motivates why translation services for connection to the Internet or other sectors are required.

- To complete the selection of scenarios, **Scenario 4** is chosen because of its different scope. To fulfill the public interest, this scenario uses a public/permissioned DLT. Issuers and RPs are primarily government agencies. The high standard for confidentiality and integrity requires the use of TPMs in combination with the wallet software. Specially certified providers provide agents. A CLS is not required due to the high standardization of attributes, but TGWs are provided by certified portal providers. The community is built by public-private partnerships.

Concluding the chapter, a first evaluation of the concept's design with the requirements of Chapter 2 results in fulfilling 24/24 essential, 30/31 important, and 1/2 optional requirements. This indicates that the concept covers nearly all needed requirements. The individual components, their descriptions, and the complete concept are supposed to be usable beyond this work. Component descriptions can help with the design, and implementation and the reference architecture provides a detailed, helpful characterization to show complete contemplation of own systems.

To prove the concept's soundness, **Chapter 5** describes how the three selected scenarios are implemented as prototypes following the concept's description. The implementations adopt the options chosen in the concept's integration section, summarized above.

- As the first prototype, **Scenario 1** is implemented in three parts. The prototype is based on PKI with X.509 certificate to sign and verify the VCs. VCs are stored in the first newly implemented part, an SSI wallet as an extension for Chromium-based browsers. This extension integrates with websites to allow users to obtain and show VCs. Those VCs are encoded using the web standard JWT. Authentication is done via another web standard WebAuthn, already available in most major browsers. The second implemented part is a web application, which simultaneously acts as an issuer and RP. Users can post messages on this web application and add proof of their knowledge about the topic. To keep the prototype concise, attestation of topical knowledge can also be obtained from the same page. In the third part, a library `LibWebSSI` is implemented, which handles all SSI operations and can be used to add SSI to different front ends.

- A second prototype is created to match **Scenario 2b**. This prototype integrates SSI features into the LoRaWAN® system provided by TTS. To do so, the LoRaWAN® join server is extended to manage the sensors' identities using SSI principles but without modifying the LoRaWAN® communication between sensors and gateways. This prototype forgoes implementing a PKI or DLT backend, as PKI is shown in the first and DLT is used in the third prototype.

- The third prototype implementation tackles **Scenario 4** and implements a smartphone wallet, multiple issuers, and RPs. This prototype uses Hyperledger Indy as the backend DLT and Hyperleger Aries as the protocol. The wallet is implemented using the cross-platform development framework React Native and uses the `ACA-PY` framework to handle SSI operations.

The choice of technology shown in the implemented prototypes can be an inspiration for implementing actual SSI products. The prototypes highlight implementation aspects that work well with SSI, e. g., using PKI and integrating SSI with WebAuthn or adding SSI functions to existing LoRaWAN® infrastructure. However, other findings show where improvement needs to be made, e. g., securing the users' wallets, especially in the web browser but also for IoT and smartphones.

The implemented prototypes are evaluated against the requirements from **Chapter 6**. This evaluation shows that the individual prototypes fulfill most requirements. Missing requirements can be added on an application level and do not affect core IAM functionality.

- The web-app prototype fulfills 13/14 essential, 5/7 important, and 1/1 optional requirements. The credential revocation is missing as an essential requirement, which has been deliberately excluded from the concept and thus could not be fulfilled. Further missing important requirements are credential recovery and MFA. As WebAuthn handles authentication, those could be achieved but are omitted as they are not unique to this prototype.
- The IoT sensor network prototype fulfills 14/17 essential, 12/19 important, and 0/1 optional requirements. This prototype also omits credential revocation and does not implement the requirements for digital and physical identification of IoT devices. Both requirements can be implemented on an application level if needed. The set of important requirements not implemented also contains credential recovery, MFA, and further application-specific requirements. As a missing optional requirement, this prototype could not fulfill the multiple identities requirement without changing the LoRaWAN® protocol. It was thus omitted.
- The eID prototype fulfills 13/15 essential, 8/11 important, and 1/2 optional requirements. In addition to the omitted credential revocation, this prototype does not specify an identity data set as an essential required feature. Defining a meaningful set of required attributes must be done at an organizational level. An additional important but missing requirement is the identity data set matching requirement. It is not implemented, as no identity data set has been defined. Message delivery services, as an optional requirement, are not implemented, as they are application-specific.

The evaluation discusses a combined scenario to cover the two components, CLS and TGW, which are not part of the prototypes. In this combined setting, all prototypes are connected to depict a workflow where data from the IoT sensor is sent to a web platform that offers users log-in via eID. As a result of this combination, it is argued that 16/18 essential, 13/17 important, and 1/1 optional requirements can be implemented with the concept. Here, the impossible requirements are a subset of the ones of the prototypes. Credential recovery is not possible, as it is excluded from the concept. MFA, credential recovery, and offline authentication and authorization are also not specified and thus impossible to implement according to the concept. Application-level requirements, such as access delegation and delegation parameters, are not specified on the SSI level.

Even though not all requirements are shown in the prototypes, the core IAM mechanics are. The prototypes and the combined scenario feature issuing and verifying VCs and basic handling of VCs within the users' wallets. Furthermore, the combined scenario also presents how the new components of CLS and TGW can be utilized to ease the migration and deployment of SSI-based IAM systems.

7.2 Revisiting the Research Questions

Eleven research questions have been formulated at the beginning of this work in Section 1.2. This section provides the answers based on the work's contribution.

1. **Which techniques enable SSI to provide usability, freedom, and privacy guarantees to users of web services and IoT devices, which conventional IAM systems (e. g., CIM or FIM) cannot provide?**

SSI achieves better usability through the usage of mostly smartphone or browser-based wallets. All relevant solutions shown in Section 3.3.2 use apps, and the prototypes developed for this work also use smartphone-based (Section 5.5) or browser-based apps (Section 5.3). Those wallets can authenticate with other entities (e. g., websites) without managing individual username and password combinations. Instead, a simple PIN code or biometric authentication (e. g., fingerprint or face scan) unlocks the wallet app to access all the users' identities. Large-scale adoption of a similar system (e. g., through WebAuthn) is not foreseeable. However, ease of use also poses dangers, as shown in the master's thesis of Moriz Teuschel, whose research highlights the subject's necessity for future development. Increased freedom through simpler data portability, as shown by combining the prototypes in Section 6.2 and using VCs in very distinct settings. SSI can only marginally improve privacy guarantees, as it is still up to the services receiving VCs to handle the containing data safely. The improvement of privacy arises from the possibility of viewing one's transaction history and knowing exactly which VCs have been shared with which verifier.

2. **How is the type of security offered by distributed ledgers (i. e., consensus on transaction ordering and prevention of double spending in a network of untrusted peers) useful to IAM systems in general and SSI in particular?**

   The primary benefit is independence from other service providers. As long as parties are interested in continuing the ledger, it will persist. With every active participant responsible for continuing the ledger, the cost is shared, and only little dependency on others is created. This and the subsequent standardization through a common API for accessing the ledger and representing VCs is a tangible benefit over federated or centralized systems.

3. **Which characteristics of a distributed ledger are required to be suitable for a privacy-conscious IAM system, and which of the currently available blockchains and distributed ledgers possess those characteristics?**

   It is shown through the development of the web prototype in Section 5.3 that SSI can function without DLT. However, as discussed in RQ 2, there are benefits to using DLT. If DLT is used, those benefits can only exist if the DLT is publicly readable, can scale from zero to hundreds of transactions, and be efficient with energy consumption and storage consumption.

4. **How can a process be defined to securely and permanently bind a digital identity (digital twin) to a (real world) entity?**

   The process for enrolling entities in an IDM system depends on the concrete use case. Enrollment is described by registering an identity and establishing credentials in Section 2.2. In an eID use case, the only secure and permanent option of associating a digital identity to one citizen is by deriving the digital identity from an official identity document, e. g., an ID card. For IoT devices, attached markers (i. e., QR codes or serial numbers) can be used to identify devices, but they can be copied or modified easily. A secure and permanent method uses PUFs included in the device's hardware. In the scenario of web users, where the primary aspect is recognizing repeated visits and offering customized experiences or data, a unique identifier combined with a secret is usually sufficient.

5. **Which methodologies enable a distributed ledger to manage the identities of many web services and a growing number of IoT devices?**

Scaling SSI happens at the user, the participating issuers and verifiers, and the infrastructure (e. g., the DLT). A wallet application for all personas can aggregate the user's identities at a central place, similar to a password safe. Both solutions increase the secure scalability of maintaining accounts and connections to many entities, such as websites. For issuers and verifiers, an SSI system has minimal effect on the scalability of their systems. They still need to administrate their local databases for their users' data. Scaling the infrastructure with DLT is one of the key challenges. As the concept described in Chapter 4 only utilizes the DLT as a universal repository for attribute definitions, only a few transactions for creating or updating attribute definitions are necessary. However, the DLT should provide provisions to prune outdated data without violating integrity guarantees.

6. **How can standardization enable organizations to use DLT to trust each other's assertions and data quality guarantees on an international scale?**

   The immediate trust between two entities cannot be formed by the technological solutions primarily developed in this work. Instead, this always requires contractual agreements. The new standards developed for SSI (e. g., for DIDs or VCs) offer advantages over established IAM standards (e. g., SAML, OIDC, or WebAuthn). Section 4.3.8 describes basic needs of an SSI community. In addition, the standardization for SSI displayed in Section 3.3 shows a strong unifying tendency, which – though also technical – eases shaping agreements.

7. **Which organizational approaches can ensure that a distributed ledger for SSI can be used by many parties for different use cases and with different implementations?**

   From an organizational point of view, it is essential to keep a ledger focused on one or a few specific tasks. This can be seen implemented in architectures that utilize multiple ledgers for different aspects of their system.

8. **How can an IAM system for SSI be implemented on a distributed ledger?**

   Implementation of an IAM system on a distributed ledger alone is not possible. Some solutions shown in the state-of-the-art in Section 3.3 do run as *Smart Contracts* or *Chaincode*, but they need to include necessary features as discussed in Section 3.3.3. Utilizing distributed ledgers as part of an IAM system is possible, as shown in Section 4.3.1.

9. **How does an interface to a decentralized identity management system need to be designed to be able to use it as a replacement for classic IAM systems?**

   The basic operations required for IAM are determined in Section 2.2. Those operations need to be exposed by an interface to the SSI system. As part of the three implementations described in Chapter 5, interfaces for the web (Section 5.3), IoT (Section 5.4), and eID (Section 5.5) have been described.

10. **Which adjustments must be made to use the self-sovereign IAM system on very constrained (i. e., connectivity, power usage, memory, computational power) IoT devices?**

    With the example of LoRaWAN®, a very low-power IoT WAN has been adapted without any changes to the end devices. A system for IoT mesh systems has been developed in [70]. This architecture requires IoT devices to perform more cryptography (i. e., public key cryptography). As a result, for any IoT system where IoT devices need to verify identities and assertions independently, efficient public key signature and encryption methods need to be implemented.

11. **How can SSI be used in an international eID system, like it is proposed by the EU's eIDAS?**

    SSI can provide a strong unifying aspect to the various IAM systems deployed today. The result is adapting eID systems for interoperability with one SSI wallet API and preventing having to build proxies or adapters for the many systems used around the EU. This chance to provide an EU-wide system is also reflected in the new revision of eIDAS [150].

7.3 Outlook on Future Work

This work and related research for SSI shows that the technical challenges can be solved for most applications. The only requirement chosen to be excluded from this work is **Credential revocation** (SEC2). State-of-the-art research is shown in Section 3.3.3.1, but no universal procedure has been developed yet. Further research might be capable of finding a solution for credential revocation that does not centralize the system and works reliably across different SSI implementations. Additional remaining technical challenges of SSI systems involve the following aspects:

- Securing VCs and corresponding private keys is a hardware (i. e., TPM) and software challenge. This is described by requirement **Protected application storage** (DAT5) and is conceptually easy to fulfill, but none of the prototypes implemented a solution that could fulfill this requirement. Additional work must be done to make secure hardware features easier and more accessible on all platforms.
- An additional requirement, which was not implemented in the prototypes, is **Multi-factor authentication** (SEC6). To secure the identities stored in the wallets, access control to the wallets needs to be strengthened, including adding provisions for MFA.

Organizational challenges similar – or even exacerbated – to those of FIM exist with the SSI systems. Those challenges are explored by the projects trying to build real-world usable SSI systems. The more successful, i. e., those running for multiple years, are still fairly centralized in their organization. Much of this is related to legal issues and financial responsibilities. Exploring the non-technical aspects of providing and legally running a world-wide backend for SSI services is still a very open field. Some of the most pressing challenges include the following:

- The idea of SSI strives for decentralized IAM accessible to anyone. In reality, the currently successful systems use private/permissoned DLT, incur hosting and management costs, and need a way to monetize their IAM platform, at least from a long-term perspective. How can systems which require investment to start, run, and maintain be decentralized and open?
- A decentralized approach to a critical IAM infrastructure, like SSI aims to be, is susceptible to security incidents that need handling. This handling is difficult to orchestrate in a decentralized architecture and requires the development of new and improved approaches.

Last but not least, technical and organizational challenges cannot be solved if users do not accept the new IAM system. Designing a successful product that is quickly adopted by a relevant number of users is very demanding. Additionally, the user must be capable of using the SSI product to their advantage. A first step in this direction is studying users' behavior with the new powers of sharing high-quality identity information from their smartphones. Further questions regarding UI are:

- What are the effects of using SSI and wallet-based IAM systems for the general population? Especially the effects on people with limited or no access to modern smartphones need to be considered, as they should not be excluded by introducing a new IAM paradigm.
- This and most related work focus on personal identities while designing applications for SSI wallets. However, the interaction with IoT devices and user-friendly management of their VCs needs to be improved to allow actual products to use SSI technology.

Based on this selection of open questions around SSI, there are still plenty of opportunities for additional research and contributions. This work provides a base SSI concept helpful in continuing with those questions.

Bibliography

[1] Josh Aas et al. “Let’s Encrypt: An Automated Certificate Authority to Encrypt the Entire Web”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 2473–2487. ISBN: 978-1-4503-6747-9. DOI: 10.1145/3319535.3363192.

[2] Moussa Aboubakar, Mounir Kellil, and Pierre Roux. “A Review of IoT Network Management: Current Status and Perspectives”. In: *Journal of King Saud University - Computer and Information Sciences* 34.7 (July 2022), pp. 4163–4176. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2021.03.006. (Visited on 06/28/2023).

[3] Andreas Abraham et al. “Revocable and Offline-Verifiable Self-Sovereign Identities”. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. Dec. 2020, pp. 1020–1027. DOI: 10.1109/TrustCom50675.2020.00136.

[4] Nazrul M. Ahmad et al. “Improving Identity Management of Cloud-Based IoT Applications Using Blockchain”. In: *2018 International Conference on Intelligent and Advanced System (ICIAS)*. Aug. 2018, pp. 1–6. DOI: 10.1109/ICIAS.2018.8540564.

[5] Ifteher Alom et al. “Dynamic Management of Identity Federations Using Blockchain”. In: *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. May 2021, pp. 1–9. DOI: 10.1109/ICBC51069.2021.9461128.

[6] Patricia Arias Cabarcos et al. “Enabling SAML for Dynamic Identity Federation Management”. In: *Wireless and Mobile Networking*. Ed. by Jozef Wozniak et al. IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer, 2009, pp. 173–184. ISBN: 978-3-642-03841-9. DOI: 10.1007/978-3-642-03841-9_16.

[7] Siddhartha Arora. “National E-ID Card Schemes: A European Overview”. In: *Information Security Technical Report* 13.2 (May 2008), pp. 46–53. ISSN: 1363-4127. DOI: 10.1016/j.istr.2008.08.002.

[8] Katja Assaf et al. “Prison Break: From Proprietary Data Sources to SSI Verifiable Credentials”. In: *Advanced Information Networking and Applications*. Ed. by Leonard Barolli. Lecture Notes in Networks and Systems. Cham: Springer International Publishing, 2023, pp. 355–366. ISBN: 978-3-031-28451-9. DOI: 10.1007/978-3-031-28451-9_31.

[9] Mikaël Ates et al. “Interoperability between Heterogeneous Federation Architectures: Illustration with SAML and WS-Federation”. In: *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*. Dec. 2007, pp. 1063–1070. DOI: 10.1109/SITIS.2007.148.

[10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A Survey". In: *Comput. Netw.* 54.15 (Oct. 2010), pp. 2787–2805. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2010.05.010.

[11] C. Warren Axelrod. "IAM Risks during Organizational Change and Other Forms of Major Upheaval". In: *Digital Identity and Access Management: Technologies and Frameworks*. Ed. by Raj Sharman, Sanjukta Das Smith, and Manish Gupta. Hershey, PA, USA: IGI Global, 2012, pp. 1–18. ISBN: 978-1-61350-498-7. DOI: 10.4018/978-1-61350-498-7.ch001.

[12] Mehmet Aydar et al. "Private Key Encryption and Recovery in Blockchain". In: *arXiv:1907.04156 [cs]* (June 2020). arXiv: 1907.04156 [cs]. URL: http://arxiv.org/abs/1907.04156.

[13] Mark C. Ballandies, Marcus M. Dapp, and Evangelos Pournaras. "Decrypting Distributed Ledger Design—Taxonomy, Classification and Blockchain Community Evaluation". In: *Cluster Computing* 25.3 (June 2022), pp. 1817–1838. ISSN: 1573-7543. DOI: 10.1007/s10586-021-03256-w.

[14] Elaine Barker et al. *A Framework for Designing Cryptographic Key Management Systems*. Tech. rep. NIST Special Publication (SP) 800-130. National Institute of Standards and Technology, Aug. 2013. DOI: 10.6028/NIST.SP.800-130.

[15] Tom Barton et al. "A Security Incident Response Trust Framework for Federated Identity (Sirtfi)". In: (Dec. 2015). URL: https://refeds.org/wp-content/uploads/2016/01/Sirtfi-1.0.pdf (visited on 04/20/2021).

[16] Diana Berbecaru and Antonio Lioy. "On the Design, Implementation and Integration of an Attribute Provider in the Pan-European eID Infrastructure". In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. June 2016, pp. 1263–1269. DOI: 10.1109/ISCC.2016.7543910.

[17] BIS Research. *Global Biometric Authentication and Identification Market in 2020 and 2026, by End User (in Million U.S. Dollars) [Graph]*. Feb. 2022. URL: https://www.statista.com/statistics/1299001/biometric-authentication-and-identification-market-by-end-user/ (visited on 06/24/2023).

[18] Board of Trustees. *Sovrin Foundation Amended and Restated Bylaws*. Jan. 2018. URL: https://sovrin.org/wp-content/uploads/Sovrin-By-Laws-V1.pdf (visited on 02/01/2021).

[19] Flavio Bonomi et al. "Fog Computing and Its Role in the Internet of Things". In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. MCC '12. Helsinki, Finland: Association for Computing Machinery, Aug. 2012, pp. 13–16. ISBN: 978-1-4503-1519-7. DOI: 10.1145/2342509.2342513.

[20] Nikita Borisov, Ian Goldberg, and Eric Brewer. "Off-the-Record Communication, or, Why Not to Use PGP". In: *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society - WPES '04*. Washington DC, USA: ACM Press, 2004, p. 77. ISBN: 978-1-58113-968-6. DOI: 10.1145/1029179.1029200.

[21] Carsten Bormann, Mehmet Ersue, and Ari Keränen. "Terminology for Constrained-Node Networks". In: Request for Comments 7228 (May 2014), p. 17. DOI: 10.17487/RFC7228.

[22] Pelle Braendgaard and et al. *Ethr-DID Library*. June 2020. URL: https://github.com/uport-project/ethr-did (visited on 06/25/2020).

[23] Bundesamt für Sicherheit in der Informationstechnik. *Requirements for Smart Card Readers Supporting eID and QES Based on EAC*. Feb. 2020.

[24] Bundesamt für Sicherheit in der Informationstechnik. *Technische Richtlinie TR-03160-1 Servicekonten*. Oct. 2019. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03160/tr03160_node.html (visited on 06/13/0202).

[25] Bundesamt für Sicherheit in der Informationstechnik. *TR-03121-2 Biometrics for Public Sector Application*. 2011. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03121/TR-03121-2_Biometrics_2_3.pdf?__blob=publicationFile&v=3 (visited on 07/23/2020).

[26] Bundesgesetzblatt. *Gesetz über Personalausweise und den elektronischen Identitätsnachweis vom 24. Juni 2009*. June 2009. URL: https://www.bmi.bund.de/SharedDocs/downloads/DE/gesetzestexte/eperso.html (visited on 06/30/2023).

[27] Michael Burrows, Martín Abadi, and Roger Needham. "A Logic of Authentication". In: *Acm Transactions on Computer Systems* 8 (1990), pp. 18–36.

[28] Vitalik Buterin. *A Next-Generation Smart Contract and Decentralized Application Platform*. 2014. URL: https://www.weusecoins.com/assets/pdf/library/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf (visited on 07/19/2016).

[29] Kim Cameron. *The Laws of Identity*. 2005. URL: http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf (visited on 10/27/2018).

[30] Marian Carcary et al. "Exploring the Determinants of IoT Adoption: Findings from a Systematic Literature Review". In: *Perspectives in Business Informatics Research*. Ed. by Jelena Zdravkovic et al. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2018, pp. 113–125. ISBN: 978-3-319-99951-7. DOI: 10.1007/978-3-319-99951-7_8.

[31] Ann Cavoukian. "Privacy by Design: The Definitive Workshop. A Foreword by Ann Cavoukian, Ph.D". In: *Identity in the Information Society* 3.2 (Aug. 2010), pp. 247–251. ISSN: 1876-0678. DOI: 10.1007/s12394-010-0062-y.

[32] David W. Chadwick. "Federated Identity Management". In: *Foundations of Security Analysis and Design V: FOSAD 2007/2008/2009 Tutorial Lectures*. Ed. by Alessandro Aldini, Gilles Barthe, and Roberto Gorrieri. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 96–120. ISBN: 978-3-642-03829-7. DOI: 10.1007/978-3-642-03829-7_3. (Visited on 03/12/2022).

[33] Richard Chang and Vitaly Shmatikov. "Formal Analysis of Authentication in Bluetooth Device Pairing". In: *FCS-ARSPA'07* 45 (2007).

[34] Bharat S. Chaudhari, Marco Zennaro, and Suresh Borkar. "LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations". In: *Future Internet* 12.3 (Mar. 2020), p. 46. ISSN: 1999-5903. DOI: 10.3390/fi12030046.

[35] Rowdy Chotkan, Jérémie Decouchant, and Johan Pouwelse. "Distributed Attestation Revocation in Self-Sovereign Identity". In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. Sept. 2022, pp. 414–421. DOI: 10.1109/LCN53696.2022.9843323.

[36] Christian Kjærgaard-Winther. *A.P. Moller - Maersk and IBM to Discontinue TradeLens, a Blockchain-Enabled Global Trade Platform*. Nov. 2022. URL: https://www.maersk.com/news/articles/2022/11/29/maersk-and-ibm-to-discontinue-tradelens (visited on 07/03/2023).

[37] Chrome Developers. *Chrome Developers Extension API Reference: Chrome.Storage*. URL: https://developer.chrome.com/docs/extensions/reference/storage/ (visited on 06/19/2022).

[38] Conor P. Cahill et al. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2. 0*. Mar. 2005.

[39] D. Cooper et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Tech. rep. RFC5280. RFC Editor, May 2008, RFC5280. DOI: 10.17487/rfc5280.

[40] George Coulouris et al. *Distributed Systems: Concepts and Design*. 5 edition. Boston: Pearson, May 2011. ISBN: 978-0-13-214301-1.

[41] Council of European Union. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union* L 119 (May 2016), pp. 1–88. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (visited on 10/01/2018).

[42] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. "Goal-Directed Requirements Acquisition". In: *Science of Computer Programming* 20.1 (Apr. 1993), pp. 3–50. ISSN: 0167-6423. DOI: 10.1016/0167-6423(93)90021-G.

[43] Sumit Singh Dhanda, Brahmjit Singh, and Poonam Jindal. "Lightweight Cryptography: A Solution to Secure IoT". In: *Wireless Personal Communications* 112.3 (June 2020), pp. 1947–1980. ISSN: 1572-834X. DOI: 10.1007/s11277-020-07134-3.

[44] Vikram Dhillon, David Metcalf, and Max Hooper. "The DAO Hacked". In: *Blockchain Enabled Applications: Understand the Blockchain Ecosystem and How to Make It Work for You*. Ed. by Vikram Dhillon, David Metcalf, and Max Hooper. Berkeley, CA: Apress, 2017, pp. 67–78. ISBN: 978-1-4842-3081-7. DOI: 10.1007/978-1-4842-3081-7_6.

[45] Mario Dobrovnik et al. "Blockchain for and in Logistics: What to Adopt and Where to Start". In: *Logistics* 2.3 (Sept. 2018), p. 18. DOI: 10.3390/logistics2030018.

[46] Drummond Reed. *Decentralized Key Management System*. Apr. 2017. URL: https://github.com/WebOfTrustInfo/rwot4-paris (visited on 05/21/2019).

[47] Drummond Reed et al. *Decentralized Identifiers (DIDs) v1.0*. Aug. 2021. URL: https://w3c.github.io/did-core/ (visited on 08/11/2021).

[48] Paul Dunphy and Fabien A.P. Petitcolas. "A First Look at Identity Management Schemes on the Blockchain". In: *IEEE Security Privacy* 16.4 (July 2018), pp. 20–29. ISSN: 1558-4046. DOI: 10.1109/MSP.2018.3111247.

[49] eIDAS Technical Sub-group. *eIDAS SAML Attribute Profile*. Aug. 2016.

[50] Rachid El Bansarkhani and Jan Sturm. "An Efficient Lattice-Based Multisignature Scheme with Applications to Bitcoins". In: *Cryptology and Network Security*. Ed. by Sara Foresti and Giuseppe Persiano. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 140–155. ISBN: 978-3-319-48965-0. DOI: 10.1007/978-3-319-48965-0_9.

[51] Abdulmotaleb El Saddik. "Digital Twins: The Convergence of Multimedia Technologies". In: *IEEE MultiMedia* 25.2 (Apr. 2018), pp. 87–92. ISSN: 1070-986X, 1941-0166. DOI: 10.1109/MMUL.2018.023121167.

[52] EU. *A Digital Single Market Strategy for Europe*. May 2015. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52015DC0192 (visited on 12/11/2019).

[53] Geovane Fedrecheski et al. “Self-Sovereign Identity for IoT Environments: A Perspective”. In: *2020 Global Internet of Things Summit (GIoTS)* (June 2020), pp. 1–6. DOI: 10.1109/GIOTS49054.2020.9119664. arXiv: 2003.05106.

[54] Adrienne Porter Felt et al. “Measuring HTTPS Adoption on the Web”. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1323–1338. ISBN: 978-1-931971-40-9. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt.

[55] Haonan Feng et al. “A Formal Analysis of the FIDO UAF Protocol.” In: *NDSS*. 2021.

[56] Md. Sadek Ferdous and Ron Poet. “Dynamic Identity Federation Using Security Assertion Markup Language (SAML)”. In: *Policies and Research in Identity Management*. Ed. by Simone Fischer-Hübner, Elisabeth de Leeuw, and Chris Mitchell. IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer, 2013, pp. 131–146. ISBN: 978-3-642-37282-7. DOI: 10.1007/978-3-642-37282-7_13.

[57] Ana Juan Ferrer, Joan Manuel Marquès, and Josep Jorba. “Towards the Decentralised Cloud: Survey on Approaches and Challenges for Mobile, Ad Hoc, and Edge Computing”. In: *ACM Computing Surveys* 51.6 (Jan. 2019), 111:1–111:36. ISSN: 0360-0300. DOI: 10.1145/3243929.

[58] Daniel Fett, Ralf Küsters, and Guido Schmitz. “An Expressive Model for the Web Infrastructure: Definition and Application to the Browser ID SSO System”. In: *2014 IEEE Symposium on Security and Privacy*. May 2014, pp. 673–688. DOI: 10.1109/SP.2014.49.

[59] FIDO Alliance. *FIDO Alliance FAQs*. July 2022. URL: https://fidoalliance.org/faqs/ (visited on 08/04/2022).

[60] Andreas Freitag. *A New Privacy Preserving and Scalable Revocation Method for Self Sovereign Identity – The Perfect Revocation Method Does Not Exist Yet*. Nov. 2022. DOI: 10.48550/arXiv.2211.13041. arXiv: 2211.13041 [cs].

[61] Ruti Gafni and Dudu Nissim. “To Social Login or Not Login? Exploring Factors Affecting the Decision”. In: *Issues in Informing Science and Information Technology* 11 (Jan. 2014), pp. 57–72. DOI: 10.28945/1980.

[62] Pedro Garcia Lopez et al. “Edge-Centric Computing: Vision and Challenges”. In: *ACM SIGCOMM Computer Communication Review* 45.5 (Sept. 2015), pp. 37–42. ISSN: 0146-4833. DOI: 10.1145/2831347.2831354.

[63] Simson Garfinkel. *PGP: Pretty Good Privacy*. ”O’Reilly Media, Inc.”, 1995. ISBN: 978-1-56592-098-9.

[64] Samson Kahsay Gebresilassie et al. “Distributed, Secure, Self-Sovereign Identity for IoT Devices”. In: *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. June 2020, pp. 1–6. DOI: 10.1109/WF-IoT48130.2020.9221144.

[65] Geldwäschegesetz - GwG. *Gesetz Über Das Aufspüren von Gewinnen Aus Schweren Straftaten § 11*. URL: https://www.gesetze-im-internet.de/gwg_2017/__11.html (visited on 06/12/2020).

[66] Arthur Gervais et al. “On the Security and Performance of Proof of Work Blockchains”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 3–16. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978341.

[67] Giovanni Buttarelli. *Preliminary Opinion on privacy by design*. May 2018. URL: https://edps.europa.eu/data-protection/our-work/publications/opinions/privacy-design_en (visited on 10/22/2019).

[68] Michael Grabatin and Wolfgang Hommel. “Blockchain-Basiertes Föderiertes Identity Management Am Beispiel von Ethereum Smart Contracts”. In: *Sicherheit in Vernetzten Systemen 24. DFN-Konferenz, 2017, Hamburg, Februar 14-15, 2017*. Hamburg: DFN / Universität der Bundeswehr München, Fakultät für Informatik, INF 2 - Institut für Softwaretechnologie, Professur: Hommel, Wolfgang, 2017, B1–B16.

[69] Michael Grabatin and Wolfgang Hommel. “Reliability and Scalability Improvements to Identity Federations by Managing SAML Metadata with Distributed Ledger Technology”. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei: IEEE, Apr. 2018, pp. 1–6. ISBN: 978-1-5386-3416-5. DOI: 10.1109/NOMS.2018.8406310.

[70] Michael Grabatin and Wolfgang Hommel. “Self-Sovereign Identity Management in Wireless Ad Hoc Mesh Networks”. In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. May 2021, pp. 480–486.

[71] Michael Grabatin, Wolfgang Hommel, and Michael Steinke. “Policy-Based Network and Security Management in Federated Service Infrastructures with Permissioned Blockchains”. In: *Security in Computing and Communications*. Ed. by Sabu M. Thampi et al. Communications in Computer and Information Science. Springer Singapore, 2019, pp. 145–156. ISBN: 9789811358265.

[72] Michael Grabatin et al. “A Matrix for Systematic Selection of Authentication Mechanisms in Challenging Healthcare Related Environments”. In: *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*. SAT-CPS ’21. New York, NY, USA: Association for Computing Machinery, Apr. 2021, pp. 88–97. ISBN: 978-1-4503-8319-6. DOI: 10.1145/3445969.3450424.

[73] Michael Grabatin et al. “Improving the Scalability of Identity Federations through Level of Assurance Management Automation”. In: *9. DFN-Forum Kommunikationstechnologien*. Gesellschaft für Informatik eV, 2016.

[74] Paul A Grassi, Michael E Garcia, and James L Fenton. *Digital Identity Guidelines*. Tech. rep. NIST SP 800-63-3. Gaithersburg, MD: National Institute of Standards and Technology, June 2017, NIST SP 800-63–3. DOI: 10.6028/NIST.SP.800-63-3.

[75] Paul A Grassi et al. *Digital Identity Guidelines: Authentication and Lifecycle Management*. Tech. rep. NIST SP 800-63b. Gaithersburg, MD: National Institute of Standards and Technology, June 2017, NIST SP 800–63b. DOI: 10.6028/NIST.SP.800-63b.

[76] Paul A Grassi et al. *Digital Identity Guidelines: Enrollment and Identity Proofing*. Tech. rep. NIST SP 800-63a. Gaithersburg, MD: National Institute of Standards and Technology, June 2017, NIST SP 800–63a. DOI: 10.6028/NIST.SP.800-63a.

[77] Paul A Grassi et al. *Digital Identity Guidelines: Federation and Assertions*. Tech. rep. NIST SP 800-63c. Gaithersburg, MD: National Institute of Standards and Technology, June 2017, NIST SP 800–63c. DOI: 10.6028/NIST.SP.800-63c.

[78] Andreas Grüner et al. “Quo Vadis, Web Authentication? – An Empirical Analysis of Login Methods on the Internet”. In: *Advanced Information Networking and Applications*. Ed. by Leonard Barolli. Lecture Notes in Networks and Systems. Cham: Springer International Publishing, 2023, pp. 471–479. ISBN: 978-3-031-28694-0. DOI: 10.1007/978-3-031-28694-0_45.

[79] Ruth Halperin and James Backhouse. "A Roadmap for Research on Identity in the Information Society". In: *Identity in the Information Society* 1.1 (Dec. 2008), pp. 71–87. ISSN: 1876-0678. DOI: 10.1007/s12394-008-0004-0.

[80] D. Hardt. *RFC6749 – The OAuth 2.0 Authorization Framework*. Internet Engineering Task Force (IETF), Oct. 2012.

[81] Heinz-Gerd Hegering, Sebastian Abeck, and Bernhard Neumair. *Integrated Management of Networked Systems: Concepts, Architectures and Their Operational Application*. Morgan Kaufmann, Aug. 1999. ISBN: 978-1-55860-571-8.

[82] Wolfgang Hommel. "Architektur- und Werkzeugkonzepte für föderiertes Identitäts-Management". PhD thesis. Ludwig-Maximilians-Universität München, July 2007. URL: https://edoc.ub.uni-muenchen.de/7300/.

[83] Wolfgang Hommel et al. "Level of Assurance Management Automation for Dynamic Identity Federations Based on Vectors of Trust". In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 39.3-4 (Jan. 2017). ISSN: 1865-8342, 0930-5157. DOI: 10.1515/pik-2016-0003.

[84] Qinwen Hu, Muhammad Rizwan Asghar, and Nevil Brownlee. "A Large-Scale Analysis of HTTPS Deployments: Challenges, Solutions, and Recommendations". In: *Journal of Computer Security* 29.1 (Jan. 2021), pp. 25–50. ISSN: 0926227X. DOI: 10.3233/JCS-200070.

[85] Andries Van Humbeeck. "The Blockchain-GDPR Paradox". In: *Journal of Data Protection & Privacy* 2.3 (Mar. 2019), pp. 208–212. URL: https://ideas.repec.org/a/aza/jdpp00/y2019v2i3p208-212.html.

[86] Hyperledger White Paper Working Group. "An Introduction to Hyperledger". In: *Linux Foundation: San Fransisco, CA, USA* (2018).

[87] "IEEE Draft Standard for Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 4: Enhancements For Transit Links Within Bridged Networks". In: *IEEE P802.11ak/D4.0, March 2017* (Jan. 2017), pp. 1–108.

[88] "IEEE Standard for Ethernet". In: *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)* (Mar. 2016), pp. 1–4017. DOI: 10.1109/IEEESTD.2016.7428776.

[89] "IEEE Standard for Information Technology– Local and Metropolitan Area Networks– Specific Requirements– Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPAN)". In: *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)* (June 2005), pp. 1–700. DOI: 10.1109/IEEESTD.2005.96290.

[90] "IEEE Standard for Low-Rate Wireless Networks". In: *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)* (Apr. 2016), pp. 1–709. DOI: 10.1109/IEEESTD.2016.7460875.

[91] International Civilian Aviation Organization (ICAO). *Doc 9303 – Machine Readable Travel Documents – Part 3: Specifications Common to All MRTD*. 2015. URL: https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf.

[92] International Telecommunication Union. "Recommendation M. 3010". In: *Principles for a Telecommunication Management Network [Z]* (Feb. 2000).

[93] Mehzabien Iqbal, Abu Yousha Md Abdullah, and Farzana Shabnam. "An Application Based Comparative Study of LPWAN Technologies for IoT Environment". In: *2020 IEEE Region 10 Symposium (TENSYMP)*. June 2020, pp. 1857–1860. DOI: 10.1109/TENSYMP50017.2020.9230597.

[94] A. K. M. Najmul Islam, Matti Mäntymäki, and Marja Turunen. "Why Do Blockchains Split? An Actor-Network Perspective on Bitcoin Splits". In: *Technological Forecasting and Social Change* 148 (Nov. 2019), p. 119743. ISSN: 0040-1625. DOI: 10.1016/j.techfore.2019.119743.

[95] *ISO/IEC 24760-1:2019 IT Security and Privacy – A Framework for Identity Management – Part 1: Terminology and Concepts*. ISO/IEC JTC 1/SC 27, May 2019. URL: https://www.iso.org/standard/77582.html.

[96] *ISO/IEC 27002:2013*. Oct. 2013. URL: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/45/54533.html (visited on 08/23/2019).

[97] ISO/IEC 7498–4. *Information Processing Systems — Open Systems Interconnection — Basic Reference Model — Part 4: Management Framework*. Nov. 1989.

[98] Ivonne Thomas and Christoph Meinel. "From Domain-Based Identity Management Systems to Open Identity Management Models". In: *Digital Identity and Access Management: Technologies and Frameworks*. Hershey, PA, USA: IGI Global, 2012, pp. 19–38. ISBN: 978-1-61350-498-7. DOI: 10.4018/978-1-61350-498-7.ch002.

[99] Nima Jafari Navimipour and Farnaz Sharifi Milani. "A Comprehensive Study of the Resource Discovery Techniques in Peer-to-Peer Networks". In: *Peer-to-Peer Networking and Applications* 8.3 (May 2015), pp. 474–492. ISSN: 1936-6450. DOI: 10.1007/s12083-014-0271-5. (Visited on 07/20/2023).

[100] Joe Andrieu, Sunny Lee, and Nate Otto. *Verifiable Claims Use Cases*. June 2017. URL: https://www.w3.org/TR/verifiable-claims-use-cases/ (visited on 05/10/2019).

[101] Johan Stokking and Roman Volosatovs. *LoRaWAN Backend Interfaces Interoperability*. July 2022. URL: https://www.thethingsindustries.com/docs/reference/interop-repository/ (visited on 07/25/2022).

[102] John Bradley et al., eds. *Client to Authenticator Protocol (CTAP)*. June 2022. URL: https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-errata-20220621.html (visited on 08/04/2022).

[103] Michael B. Jones, Akshay Kumar, and Emil Lundberg, eds. *Web Authentication: An API for Accessing Public Key Credentials - Level 3*. Apr. 2021. URL: https://w3c.github.io/webauthn/ (visited on 06/26/2022).

[104] Bastian Kemmler and Dieter Kranzlmüller. "Redefining the Cloud Based on Beneficial Service Characteristics - A New Cloud Taxonomy Leads to Economically Reasonable Semi-Cloudification". In: *Proceedings of the 5th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER*, SciTePress / INSTICC, 2015, pp. 135–144. ISBN: 978-989-758-104-5. DOI: 10.5220/0005446401350144.

[105] Olga Kieselmann, Nils Kopal, and Arno Wacker. "A Novel Approach to Data Revocation on the Internet". In: *Data Privacy Management, and Security Assurance*. Ed. by Joaquin Garcia-Alfaro et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 134–149. ISBN: 978-3-319-29883-2. DOI: 10.1007/978-3-319-29883-2_9.

[106] John Klensin and Peter Saint-Andre. *RFC 8141: Uniform Resource Names (URNs)*. Tech. rep. Apr. 2017. URL: https://tools.ietf.org/html/rfc8141.

[107] Hermann Kopetz. "Internet of Things". In: *Real-Time Systems*. Real-Time Systems Series. Springer, Boston, MA, 2011, pp. 307–323. ISBN: 978-1-4419-8236-0. DOI: 10.1007/978-1-4419-8237-7_13.

[108] Nikita Korzhitskii and Niklas Carlsson. "Revocation Statuses on the Internet". In: *Passive and Active Measurement*. Ed. by Oliver Hohlfeld, Andra Lutu, and Dave Levin. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 175–191. ISBN: 978-3-030-72582-2. DOI: 10.1007/978-3-030-72582-2_11.

[109] Robert Krimmer et al. "Exploring and Demonstrating the Once-Only Principle: A European Perspective". In: *Proceedings of the 18th Annual International Conference on Digital Government Research*. Dg.o '17. Staten Island, NY, USA: Association for Computing Machinery, June 2017, pp. 546–551. ISBN: 978-1-4503-5317-5. DOI: 10.1145/3085228.3085235.

[110] Taisya Krivoruchko, James Diamond, and Jeff Hooper. "Storing RSA Private Keys In Your Head". In: *2006 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)*. Dec. 2006, pp. 129–138. DOI: 10.1109/PRDC.2006.58.

[111] Herbert Kubicek and Torsten Noack. "Different Countries-Different Paths Extended Comparison of the Introduction of eIDs in Eight European Countries". In: *Identity in the Information Society* 3.1 (July 2010), pp. 235–245. ISSN: 1876-0678. DOI: 10.1007/s12394-010-0063-x.

[112] Dennis Kügler and Ingo Naumann. "Sicherheitsmechanismen für kontaktlose Chips im deutschen Reisepass: Ein Überblick über Sicherheitsmerkmale, Risiken und Gegenmaßnahmen". In: *Datenschutz und Datensicherheit - DuD* 31.3 (Mar. 2007), pp. 176–180. ISSN: 1614-0702, 1862-2607. DOI: 10.1007/s11623-007-0066-4.

[113] Dennis Lamken et al. "Design Patterns and Framework for Blockchain Integration in Supply Chains". In: *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. May 2021, pp. 1–3. DOI: 10.1109/ICBC51069.2021.9461062.

[114] Peter Landrock. "X.509". In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg. Boston, MA: Springer US, 2005, pp. 669–669. ISBN: 978-0-387-23483-0. DOI: 10.1007/0-387-23483-7_462.

[115] Jason Law and Lovesh Harchandani. "Scaling a BFT Consensus Protocol for Identity". In: *ID2020 Rebooting the Web of Trust Design Shop* (May 2016). URL: https://github.com/WebOfTrustInfo/rwot2-id2020 (visited on 01/20/2021).

[116] Paul J. Leach, Michael Mealling, and Rich Salz. *A Universally Unique IDentifier (UUID) URN Namespace*. Tech. rep. IETF Network Working Group, July 2005. URL: https://tools.ietf.org/html/rfc4122.

[117] Ulrike Lechner, ed. *LIONS Monitor : Resilience and Digital Sovereignty in Organizations*. 1st edition. Neubiberg: Universität der Bundeswehr München / Universität der Bundeswehr München, Fakultät für Informatik, INF 8 - Institut für Schutz und Zuverlässigkeit, Professur: Lechner, Ulrike, 2023. ISBN: 978-3-943207-67-5 978-3-943207-68-2.

[118] Huang-Chen Lee and Kai-Hsiang Ke. "Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation". In: *IEEE Transactions on Instrumentation and Measurement* 67.9 (Sept. 2018), pp. 2177–2187. ISSN: 1557-9662. DOI: 10.1109/TIM.2018.2814082.

[119] Gabriel M. Lentner and Peter Parycek. "Electronic Identity (eID) and Electronic Signature (eSig) for eGovernment Services – a Comparative Legal Study". In: *Transforming Government: People, Process and Policy* 10.1 (Jan. 2016), pp. 8–25. ISSN: 1750-6166. DOI: 10.1108/TG-11-2013-0047.

[120] Sin Kuang Lo et al. "Analysis of Blockchain Solutions for IoT: A Systematic Literature Review". In: *IEEE Access* 7 (2019), pp. 58822–58835. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2914675.

[121] LoRa Alliance. *LoRaWAN® 1.0.4 Specification Package*. 2020. URL: https://lora-alliance.org/resource_hub/lorawan-104-specification-package/ (visited on 07/17/2022).

[122] LoRa Alliance. *LoRaWAN® 1.1 Specification*. 2017. URL: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf (visited on 10/16/2018).

[123] Manu Sporny, Dave Longley, and David Chadwick. *Verifiable Credentials Data Model 1.0*. Mar. 2019. URL: https://www.w3.org/TR/verifiable-claims-data-model/ (visited on 05/08/2019).

[124] Markus Sabadello et al. *Introduction to DID Auth*. July 2018. URL: https://github.com/WebOfTrustInfo/rwot6-santabarbara (visited on 05/21/2019).

[125] Larry Masinter, Tim Berners-Lee, and Roy T. Fielding. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. IETF Network Working Group, Jan. 2005. URL: https://tools.ietf.org/html/rfc3986.

[126] Lucica Matei and José Luis Vázquez-Burguete. *Permanent Study Group: Public and Nonprofit Marketing : Proceedings*. Matei Lucica, 2012. ISBN: 978-973-709-612-8.

[127] Friedemann Mattern and Christian Floerkemeier. "From the Internet of Computers to the Internet of Things". In: *From Active Data Management to Event-Based Systems and More*. Springer, 2010, pp. 242–259.

[128] Roman Matzutt et al. "A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin". In: *Financial Cryptography and Data Security*. Ed. by Sarah Meiklejohn and Kazue Sako. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2018, pp. 420–438. ISBN: 978-3-662-58387-6. DOI: 10.1007/978-3-662-58387-6_23.

[129] Roman Matzutt et al. "Thwarting Unwanted Blockchain Content Insertion". In: *2018 IEEE International Conference on Cloud Engineering (IC2E)*. Apr. 2018, pp. 364–370. DOI: 10.1109/IC2E.2018.00070.

[130] R. J. Mcwaters et al. "A Blueprint for Digital Identity the Role of Financial Institutions in Building Digital Identity". In: *World Economic Forum, Future of Financial Services Series*. Aug. 2016, pp. 1–108.

[131] millsd. *Introducing BrowserID: A Better Way to Sign In*. Blog. July 2011. URL: https://web.archive.org/web/20130128201115/http://identity.mozilla.com/post/7616727542/introducing-browserid-a-better-way-to-sign-in (visited on 08/04/2022).

[132] Anita Mittal. *Catalog of Technical Standards for Digital Identification Systems*. Tech. rep. Washington, D.C.: World Bank Group, Sept. 2018. URL: http://documents.worldbank.org/curated/en/707151536126464867/Catalog-of-Technical-Standards-for-Digital-Identification-Systems (visited on 07/02/2020).

[133] *Mozilla/Persona*. Mozilla. July 2022. URL: https://github.com/mozilla/persona (visited on 08/04/2022).

[134] Alexander Mühle et al. "A Survey on Essential Components of a Self-Sovereign Identity". In: *Computer Science Review* 30 (Nov. 2018), pp. 80–86. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2018.10.002.

[135] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Oct. 2008. URL: http://www.cryptovest.co.uk/resources/Bitcoin%20paper%20Original.pdf (visited on 07/19/2016).

[136] Niels Hackius and Moritz Petersen. “Blockchain in Logistics and Supply Chain: Trick or Treat?” In: *Digitalization in Supply Chain Management and Logistics: Smart and Digital Solutions for an Industry 4.0 Environment. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 23*. Ed. by Thorsten Ringle, Christian M. Kersten, and Wolfgang Blecker. Berlin: epubli GmbH, 2017, pp. 3–18. ISBN: 978-3-7450-4328-0. DOI: 10.15480/882.1444.

[137] Torsten Noack and Herbert Kubicek. “The Introduction of Online Authentication as Part of the New Electronic National Identity Card in Germany”. In: *Identity in the Information Society* 3.1 (July 2010), pp. 87–110. ISSN: 1876-0678. DOI: 10.1007/s12394-010-0051-1.

[138] *Overview of Pre-Notified and Notified eID Schemes under eIDAS*. Apr. 2023. URL: https://ec.europa.eu/digital-building-blocks/wikis/display/EIDCOMMUNITY/Overview+of+pre-notified+and+notified+eID+schemes+under+eIDAS (visited on 06/25/2023).

[139] Eleni Panopoulou et al. “Stakeholder Community for Once-Only Principle”. In: *Public management review* 6 (2004), pp. 21–53.

[140] Chang-Seop Park. “On Certificate-Based Security Protocols for Wireless Mobile Communication Systems”. In: *IEEE Network* 11.5 (Sept. 1997), pp. 50–55. ISSN: 1558-156X. DOI: 10.1109/65.620522.

[141] Moritz Platt et al. “The Energy Footprint of Blockchain Consensus Mechanisms Beyond Proof-of-Work”. In: *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. Dec. 2021, pp. 1135–1144. DOI: 10.1109/QRS-C55045.2021.00168.

[142] Daniela Pöhn. “Architektur und Werkzeuge für dynamisches Identitätsmanagement in Föderationen”. PhD thesis. Ludwig-Maximilians-Universität München, Nov. 2016. URL: https://edoc.ub.uni-muenchen.de/20203/ (visited on 04/24/2019).

[143] Daniela Pöhn, Michael Grabatin, and Wolfgang Hommel. “eID and Self-Sovereign Identity Usage: An Overview”. In: *Electronics* 10.22 (Nov. 2021), p. 2811. ISSN: 2079-9292. DOI: 10.3390/electronics10222811.

[144] Daniela Pöhn, Michael Grabatin, and Wolfgang Hommel. “Modeling the Threats to Self-Sovereign Identities”. In: *Open Identity Summit 2023*. Ed. by Heiko Roßnagel, Christian H. Schunck, and Jochen Günther. Bonn, Germany: Gesellschaft für Informatik e.V., 2023, pp. 85–96. DOI: 10.18420/OID2023_07.

[145] Daniela Pöhn and Wolfgang Hommel. “Management Architecture for Dynamic Federated Identity Management”. In: *Computer Science & Information Technology (CS & IT)*. Academy & Industry Research Collaboration Center (AIRCC), May 2016, pp. 211–226. ISBN: 978-1-921987-51-9. DOI: 10.5121/csit.2016.60617.

[146] Daniela Pöhn and Wolfgang Hommel. “Proven and Modern Approaches to Identity Management”. In: *Advances in Cybersecurity Management*. Ed. by Kevin Daimi and Cathryn Peoples. Cham: Springer International Publishing, 2021, pp. 421–443. ISBN: 978-3-030-71381-2. DOI: 10.1007/978-3-030-71381-2_19.

[147] Daniela Pöhn and Wolfgang Hommel. “Universal Identity and Access Management Framework for Future Ecosystems”. In: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 12.1 (Mar. 2021), pp. 64–84. DOI: 10.22667/JOWUA.2021.03.31.064.

[148] Daniela Pöhn, Stefan Metzger, and Wolfgang Hommel. "Géant-TrustBroker: Dynamic, Scalable Management of SAML-Based Inter-federation Authentication and Authorization Infrastructures". In: *ICT Systems Security and Privacy Protection*. Ed. by Nora Cuppens-Boulahia et al. IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer, 2014, pp. 307–320. ISBN: 978-3-642-55415-5. DOI: 10.1007/978-3-642-55415-5_25.

[149] Andreas Poller et al. *Electronic Identity Cards for User Authentication—Promise and Practice*. Text. Jan. 2012. DOI: 10.1109/MSP.2011.148.

[150] *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL Amending Regulation (EU) No 910/2014 as Regards Establishing a Framework for a European Digital Identity*. 2021. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A52021PC0281 (visited on 01/09/2022).

[151] Benedikt Putz and Günther Pernul. "Detecting Blockchain Security Threats". In: *2020 IEEE International Conference on Blockchain (Blockchain)*. Nov. 2020, pp. 313–320. DOI: 10.1109/Blockchain50366.2020.00046.

[152] Qahhar Muhammad Qadir et al. "Low Power Wide Area Networks: A Survey of Enabling Technologies, Applications and Interoperability Needs". In: *IEEE Access* 6 (2018), pp. 77454–77473. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2883151.

[153] Maninder Singh Raniyal et al. "Passphrase Protected Device-to-Device Mutual Authentication Schemes for Smart Homes". In: *SECURITY AND PRIVACY* 1.3 (2018), e42. ISSN: 2475-6725. DOI: 10.1002/spy2.42.

[154] Mohammadreza Rasolroveicy and Marios Fokaefs. "Performance Evaluation of Distributed Ledger Technologies for IoT Data Registry : A Comparative Study". In: *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. July 2020, pp. 137–144. DOI: 10.1109/WorldS450073.2020.9210358.

[155] Drummond Reed, Jason Law, and Daniel Hardman. "The Technical Foundations of Sovrin". In: (Sept. 2016), p. 26.

[156] REFEDS. "A Security Incident Response Trust Framework for Federated Identity (Sirtfi) Version 2". In: (28 J ULY 2022). URL: https://refeds.org/wp-content/uploads/2022/08/Sirtfi-v2.pdf (visited on 07/05/2023).

[157] *Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on Electronic Identification and Trust Services for Electronic Transactions in the Internal Market and Repealing Directive 1999/93/EC*. Aug. 2014. URL: http://data.europa.eu/eli/reg/2014/910/oj/eng (visited on 12/11/2019).

[158] Victor Ribeiro et al. "A Fault-Tolerant and Secure Architecture for Key Management in LoRaWAN Based on Permissioned Blockchain". In: *IEEE Access* 10 (2022), pp. 58722–58735. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3179004.

[159] Markus Sabadello and Dmitri Zagidulin. *Decentralized Identifier Resolution (DID Resolution) v0.2*. Ed. by Markus Sabadello and Dmitri Zagidulin. Feb. 2022. URL: https://w3c-ccg.github.io/did-resolution/ (visited on 03/12/2022).

[160] Natsuhiko Sakimura et al. "Openid Connect Core 1.0". In: *The OpenID Foundation* (2014), S3. URL: https://openid.net/specs/openid-connect-core-1_0-final.html (visited on 06/25/2023).

[161] Salah Machani et al., eds. *FIDO UAF Architectural Overview*. Oct. 2020. URL: https://fidoalliance.org/specs/fido-uaf-v1.2-ps-20201020/fido-uaf-overview-v1.2-ps-20201020.html (visited on 08/04/2022).

[162] Sampath Srinivas et al., eds. *Universal 2nd Factor (U2F) Overview*. Apr. 2017. URL: https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html (visited on 08/04/2022).

[163] S. Santesson et al. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. Tech. rep. RFC6960. RFC Editor, June 2013, RFC6960. DOI: 10.17487/rfc6960.

[164] Mahadev Satyanarayanan. "The Emergence of Edge Computing". In: *Computer* 50.1 (Jan. 2017), pp. 30–39. ISSN: 1558-0814. DOI: 10.1109/MC.2017.9.

[165] Mahadev Satyanarayanan et al. "The Case for VM-Based Cloudlets in Mobile Computing". In: *IEEE Pervasive Computing* 8.4 (Oct. 2009), pp. 14–23. ISSN: 1558-2590. DOI: 10.1109/MPRV.2009.82.

[166] Rüdiger Schollmeier, Ingo Gruber, and Michael Finkenzeller. "Routing in Mobile Ad-hoc and Peer-to-Peer Networks A Comparison". In: *Web Engineering and Peer-to-Peer Computing*. Ed. by Enrico Gregori et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2002, pp. 172–187. ISBN: 978-3-540-45745-9. DOI: 10.1007/3-540-45745-3_16.

[167] Abubakar-Sadiq Shehu, Antonio Pinto, and Manuel E. Correia. *A Decentralised Real Estate Transfer Verification Based on Self-Sovereign Identity and Smart Contracts*. July 2022. DOI: 10.48550/arXiv.2207.04459. arXiv: 2207.04459 [cs].

[168] Clay Shirky. "What Is P2p... and What Isn't". In: *The O'Reilly P2P Conference*. 2000.

[169] Mahesh Shirole, Maneesh Darisi, and Sunil Bhirud. "Cryptocurrency Token: An Overview". In: *IC-BCT 2019*. Ed. by Dhiren Patel et al. Blockchain Technologies. Singapore: Springer, 2020, pp. 133–140. ISBN: 9789811545429. DOI: 10.1007/978-981-15-4542-9_12.

[170] Soraya Sinche et al. "A Survey of IoT Management Protocols and Frameworks". In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 1168–1190. ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2943087.

[171] Karen R. Sollins. "IoT Big Data Security and Privacy Versus Innovation". In: *IEEE Internet of Things Journal* 6.2 (Apr. 2019), pp. 1628–1635. ISSN: 2327-4662. DOI: 10.1109/JIOT.2019.2898113.

[172] Gabor Soos et al. "IoT Device Lifecycle – A Generic Model and a Use Case for Cellular Mobile Networks". In: Aug. 2018, pp. 176–183. DOI: 10.1109/FiCloud.2018.00033.

[173] Statista. *Statistics Report about Password Security*. 2023. URL: https://www.statista.com/study/109492/password-security/ (visited on 06/24/2023).

[174] Unal Tatar, Yasir Gokce, and Brian Nussbaum. "Law versus Technology: Blockchain, GDPR, and Tough Tradeoffs". In: *Computer Law & Security Review* 38 (Sept. 2020), p. 105454. ISSN: 0267-3649. DOI: 10.1016/j.clsr.2020.105454.

[175] Technical Commitee: ISO/IEC JTC 1/SC 17. *ISO/IEC 14443-4:2018*. June 2018. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/35/73599.html (visited on 07/17/2020).

[176] Technical Committee ISO/IEC JTC 1. *ISO/IEC 29115:2013*. Apr. 2013. URL: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/04/51/45138.html.

[177] Technical Committee: ISO/IEC JTC 1/SC 17. *ISO/IEC 7501-1:2008*. Sept. 2008. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/04/55/45562.html (visited on 07/17/2020).

[178] Technical Committee: ISO/IEC JTC 1/SC 37 Biometrics. *ISO/IEC 19784-1:2018 Information Technology – Biometric Application Programming Interface – Part 1: BioAPI Specification*. Apr. 2018. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/08/70866.html (visited on 07/23/2020).

[179] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. Tech. rep. Federal Information Processing Standard (FIPS) 186-4. U.S. Department of Commerce, July 2013. DOI: 10.6028/NIST.FIPS.186-4.

[180] National Institute of Standards and Technology. *Security Requirements for Cryptographic Modules*. Tech. rep. Federal Information Processing Standard (FIPS) 140-3. U.S. Department of Commerce, Mar. 2019. DOI: 10.6028/NIST.FIPS.140-3.

[181] Andrew Tobin. *Sovrin: What Goes on the Ledger?* Sept. 2018. URL: https://www.evernym.com/wp-content/uploads/2017/07/What-Goes-On-The-Ledger.pdf (visited on 06/18/2020).

[182] Andrew Tobin, Drummond Reed, and Phillip J Windley. *The Inevitable Rise of Self-Sovereign Identity*. Sept. 2016. URL: https://www.evernym.com/wp-content/uploads/2017/07/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf (visited on 01/19/2021).

[183] TradeLens. *TradeLens Blockchain-Enabled Digital Shipping Platform Continues Expansion with Addition of Major Ocean Carriers Hapag-Lloyd and Ocean Network Express*. July 2019. URL: https://www.maersk.com/news/articles/2019/07/02/hapag-lloyd-and-ocean-network-express-join-tradelens (visited on 03/13/2022).

[184] Transforma Insights. *Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2021, with Forecasts from 2022 to 2030 (in Billions) [Graph]*. July 2022. URL: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (visited on 06/24/2023).

[185] Michal Trnka and Tomas Cerny. "Identity Management of Devices in Internet of Things Environment". In: *2016 6th International Conference on IT Convergence and Security (ICITCS)*. Sept. 2016, pp. 1–4. DOI: 10.1109/ICITCS.2016.7740343.

[186] Twitter, Inc. *Overview*. 2020. URL: https://developer.twitter.com/en/docs/basics/authentication/overview (visited on 07/27/2020).

[187] Dejan Vujičić, Dijana Jagodić, and Siniša Ranđić. "Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview". In: *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*. Mar. 2018, pp. 1–6. DOI: 10.1109/INFOTEH.2018.8345547.

[188] Arno Wacker et al. "A Fault-Tolerant Key-Distribution Scheme for Securing Wireless Ad Hoc Networks". In: *Pervasive Computing: Second International Conference, PERVASIVE 2004, Linz/Vienna, Austria, April 21-23, 2004. Proceedings 2*. Springer, 2004, pp. 194–212.

[189] Arno Rüdiger Wacker. "Key Distribution Schemes for Resource-Constrained Devices in Wireless Sensor Networks". PhD thesis. 2007.

[190] Yvonne Wilson and Abhishek Hingnikar. *Solving Identity Management in Modern Applications: Demystifying OAuth 2, OpenID Connect, and SAML 2*. Berkeley, CA: Apress, 2023. ISBN: 978-1-4842-8260-1 978-1-4842-8261-8. DOI: 10.1007/978-1-4842-8261-8.

[191] Ibrar Yaqoob et al. “Mobile Ad Hoc Cloud: A Survey”. In: *Wireless Communications and Mobile Computing* 16.16 (2016), pp. 2572–2589. ISSN: 1530-8677. DOI: 10.1002/wcm.2709.

[192] Hakan Yildiz et al. “Connecting Self-Sovereign Identity with Federated and User-centric Identities via SAML Integration”. In: *2021 IEEE Symposium on Computers and Communications (ISCC)*. Sept. 2021, pp. 1–7. DOI: 10.1109/ISCC53001.2021.9631453.

[193] Thomas Zefferer and Peter Teufl. “Leveraging the Adoption of Mobile eID and E-Signature Solutions in Europe”. In: *Electronic Government and the Information Systems Perspective*. Ed. by Andrea Kő and Enrico Francesconi. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 86–100. ISBN: 978-3-319-22389-6. DOI: 10.1007/978-3-319-22389-6_7.

[194] Fan Zhang et al. “DECO: Liberating Web Data Using Decentralized Oracles for TLS”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 1919–1938. ISBN: 978-1-4503-7089-9. DOI: 10.1145/3372297.3417239.

[195] Ming Zhao et al. “A Comprehensive Study of RPL and P2P-RPL Routing Protocols: Implementation, Challenges and Opportunities”. In: *Peer-to-Peer Networking and Applications* 10.5 (Sept. 2017), pp. 1232–1256. ISSN: 1936-6450. DOI: 10.1007/s12083-016-0475-y. (Visited on 07/20/2023).

[196] Gary Zimmerman and Drummond Reed. *Becoming a Sovrin Steward*. 2016. URL: https://www.evernym.com/wp-content/uploads/2017/07/Becoming-a-Sovrin-Steward.pdf (visited on 01/19/2021).