DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Analysis and Evaluation of Blockchain-based Self-Sovereign Identity Systems

Martin Schäffner

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Analysis and Evaluation of Blockchain-based Self-Sovereign Identity Systems

Analyse und Evaluation von Blockchain-basierten Self-Sovereign Identity Systemen

Author:	Martin Schäffner
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ulrich Gallersdörfer, M.Sc.
Submission Date:	November 12th, 2019

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

Munich, November 12th, 2019 Martin Schäffner

Acknowledgments

At this point, I would like to thank all those who supported and motivated me during the course of writing this Master's Thesis.

I would like to start off by thanking my advisor Uli Gallersdörfer for his support, remarks, and open communication through the learning process of my thesis. In particular, I would like to thank him for the constructive discussions in the weekly meetings in which the progress of the thesis and future concerns were openly discussed.

I would also like to thank Professor Dr. Florian Matthes for his contribution during the definition of my topic and the opportunity to write my thesis at his chair for Software Engineering for Business Information Systems (SEBIS).
Furthermore, I would like to express my gratitude to my colleagues at Datarella who drew my attention to the topic and supported me during the time of writing my thesis. Their guidance led to valuable ideas and inspiration for this thesis.

Finally, I want to thank my family and friends who were always there for me when I needed them, and never failed to keep me motivated. I would like to especially thank Kaitlyn Wharfield, who acted as a second reader of my thesis and helped me in finding the right words.

This accomplishment wouldn't be possible without them. Thank you.

Abstract

Identity management on the internet has been a problem since its origin, as there is no identity layer implemented. Nowadays, centralized institutions have a lot of power, such as Certifying Authorities (CAs) or identity providers (IDP), which makes them an attractive target to attack and manipulate the system or to control the identifiers of their users or customers. With the rise of blockchain technology, a new concept of identity management is evolving called Self-Sovereign Identity (SSI). This thesis provides an overview of the Self-Sovereign Identity ecosystem, its involved components, and implemented systems based on the DID methods to create a decentralized identity infrastructure on the internet.

The thesis starts with investigating the evolution of online identities, beginning with centralized-, federated-, and user-centric identities, before it eventually evolved into the concept of Self-Sovereign Identity. Afterwards, this concept is presented, including the problems of today's conventional online identities, the purpose of SSI, its principles, and use cases.

In the next chapter, the components of SSI will be described in detail, evaluated, and visualized in a components architecture. These include standards like decentralized identifiers (DIDs), verifiable credentials (VCs), and verifiable presentations (VPs). Further, the concepts of a decentralized public key infrastructure (DPKI) and a decentralized key management system (DKMS) are introduced. Additionally, this thesis deals in detail with the trust infrastructure of SSI.

The second part of this thesis focuses on SSI systems and their underlying DID methods. To provide an overview of existing identity systems, the SSI ecosystem is analyzed on its currently existing DID methods. Based on the presented DID methods, representative DID methods are selected and examined for further analysis and evaluation of the system. To analyze the DID methods and their systems, criteria are defined to emphasize the differences of each DID method. The results from the analysis are then used for evaluating the DID methods.

Contents

1 Introduction

Self-Sovereign Identity (SSI) has evolved into a promising identity concept for the internet, which allows users to keep autonomy over their identifiers and control how personal related information is shared and with whom. It describes a concept with open standards for a privacy-preserving, internet-wide identity layer in a decentralized way. Due to the utilization of blockchains to create, update, and delete decentralized identifiers (DID), no central authority or other intermediaries managing the identifiers are required. Users are in charge of managing their identifiers and hence, can manage them in a sovereign way.

By using newly created standardized technology and processes, it is possible to exchange identity information, such as credentials, online in a trusted way. With the integration of verifiable credentials, it should be possible for users to gather personal information from issuers to merge identity information from the analog- closer to the digital identity.

This thesis focuses on the underlying technologies, processes, and components necessary to establish Self-Sovereign Identity and explains how they interact with each other. Further, already existing DID methods and their associated systems will be analyzed and evaluated based on defined criteria.

1.1 Problem Statement

Conventional digital identities aren't managed by the users themselves, but by central institutions. A digital identity could be an account on a website, or at an identity provider (IDP), like Google, that can further be used to access other websites and online services. Those identities are represented by identifiers like a username or an e-mail address. Besides the fact, that users are highly dependent on the identity providers, they also have to trust them that their data is securely protected against data breaches or that the data isn't used for undesired distribution without the user's consent.

Furthermore, the internet itself relies on a centralized and hierarchical structure in the DNS that the bears risk of manipulation. Certifying Authorities (CAs) verify that domains belong to specific identities and mark this domain as trusted. However, CAs can determine other trusted parties to issue certificates on their behalf.

As a result, the current infrastructure isn't suitable to establish a Self-Sovereign Identity ecosystem. Hence, a new infrastructure is required that allows users to manage their identifiers without the need for a central institution. It is unclear how components interact with each other and how existing SSI approaches differ.

1.2 Research Questions

The following research questions (RQ) will be covered during this thesis:

RQ1 - Which components comprise the architecture for self-sovereign identity?

In order to get a complete picture of the structure of a self-sovereign identity, it is important to identify the implemented components that enable SSI. What is the purpose of each component? How should they interact with each other? To answer these questions, existing and newly created concepts and formats are explained and set in context to each other which leads to a new components architecture model for SSI.

RQ2 - What is the applicability of the integrated components?

Components in SSI have to adopt the tasks that were previously executed by central institutions. Optimally, those tasks are not just simply adopted, but extended to facilitate further use while respecting the principles of SSI. It is therefore important to analyze current features based on their applicability. Do the components meet their requirements? Are there essential features missing? Which additional features could improve SSI? The answers to these questions are derived from their analysis.

RQ3 - Which criteria can be used to analyze DID methods and their systems?

DIDs are derived from DID methods on a specific ledger in a specific way to serve a specific purpose. Besides managing the identifiers in a self-sovereign way, the applicability of a DID highly depends on how it can be used inside the system. In order to identify the differences between DID methods and their respective system, this thesis defines criteria how DID methods can be analyzed.

RQ4 - How do the DID methods and their systems differ based on the criteria?

In order to compare the DID methods, they need to be analyzed on each predefined criterion. Consequently, differences between the DID methods emerge which leads to findings that conclude on how well the DID methods are implemented. The results in each criterion will then be used for the evaluation of the selected DID methods. Additionally, fields of use for each DID method will be suggested.

1.3 Research Approach

This thesis was created by utilizing methods of Grounded Theory. This approach is defined by "a logically consistent set of data collection and analytic procedures aimed

to develop theory" [1]. The goal of Grounded Theory is to formulate a new theory by analyzing different sources of information. An iterative process takes place which alternates between analyzing and interpreting data until no new knowledge emerges.

For this thesis, data was gathered by collecting information from publications of experts in the field, online meetups, academic literature, and whitepapers or technical specifications of SSI systems. Due to the community-culture to open-source information about systems, there is a significant amount of information publicly available. Therefore, Grounded Theory is very suitable for this topic.

For the first research question, the concepts of existing standards or technologies for components, frameworks, and formats are collected, analyzed, and transferred into an architecture model.

The second research question to evaluate the applicability of the components is answered through the contributions of experts' proposals and ideas, and those of the author gained during the research.

Once the big picture is clear, some of the officially registered DID methods are chosen to represent the diversity among them. Information is gathered through reviewing the respective DID method specifications, further documentation provided by the institution, and online forums. These serve as the basis for the analysis and evaluation.

After the representative DID methods are selected, criteria are defined that are suitable for the comparison and analysis of DID methods and their systems, which is the topic of the third research question.

Based on the results of the third research question, the DID methods are evaluated and potential areas of application are suggested.

1.4 Outline

At the beginning of this thesis, the evolution of online identities is reflected. Next, the concept of Self-Sovereign Identity is presented and set against the problems of conventional online identities. Newly implemented standards and their characteristics, as well as decentralized components, that are used for SSI, are presented in chapter 4. Furthermore, concepts and efforts to establish trust in the SSI ecosystem will be discussed. These presented components used in SSI will be set in context and transferred to a components architecture model of SSI. After discussing the architecture, chapter 5 introduces the already registered DID methods and presents a selected subset of DID methods for later analysis and evaluation. In chapter 6, criteria are defined and the selected DID methods are analyzed based on these definitions. The results obtained in the previous chapter are then evaluated in chapter 7, before the thesis concludes in chapter 8.

2 Related Work

The purpose of this chapter is to present related work to this thesis. There are two research categories identified that are relevant for this thesis.

2.1 Work on Self-Sovereign Identity

This chapter contains information about resources focusing on the concept of Self-Sovereign identity. Since this term becomes increasingly more popular, plenty of papers and articles are being published on this topic. This chapter introduces the most important ones, that describe Self-Sovereign Identity out of different perspectives, times, and with a different focus.

A paper that describes Self-Sovereign Identity well is "The Path to Self-Sovereign Identity" [2] by Christopher Allen, who is highly involved in designing and developing decentralized identities and is also a Co-Author of the TLS security standard. His paper summarizes the evolution of online identities, from a centralized identity approach until self-sovereign identity, as well as presenting guiding principles for the SSI ecosystem. As the document was published in 2016, it was considered as the first document that established the term Self-Sovereign Identity and described its contents in an easy and understandable way.

Markus Sabadello holds in his GitHub repository [3] a significant list of projects or other research related to identity on the blockchain. The wide range of topics gives a good overview of projects and work on digital identities. Sabadello is further heavily involved in many SSI-related projects such as the Universal Resolver, the Verifiable Credentials Data Model, or the DID specification.

The Sovrin Foundation published the paper "The Inevitable Rise of Self-Sovereign Identity" [4], in which they explain their understanding of SSI, as well as proposed the importance of Governance [5] and Trust Frameworks [6] for SSI-compatible systems.

The Rebooting Web of Trust (RWOT) events are also worth mentioning. During these repeating meetings, experts in the field of Self-Sovereign Identity and related areas work together to further implement existing standards, such as the DID standard, or design concepts, such as DPKI [7]. They store their findings in the Rebooting Web of Trust Organization in GitHub [8].

2.2 Academic Literature

This chapter focuses on the existing academic literature that covers relevant Self-Sovereign Identity research. This section does not claim to be complete, since there may be other projects or papers that have also done work worth mentioning in this chapter.

A paper that follows a similar approach like this thesis was published in 2017 by Andreas Abraham with the title "Self-Sovereign Identity - Whitepaper about the Concept of Self-Sovereign Identity including its Potential" [9]. He also explained the concept of SSI as well as selected some blockchain-based identity technologies which were analyzed and evaluated, and further analyzed the potential of SSI. His key findings were that Sovrin has the most complete system, while other technologies, such as uPort, Multichain, or Blockstack, fall behind.

Loïc Lesavre, Priam Varin et al. published a draft of the paper "A Taxonomic Approach to Understanding Emerging Blockchain Identity Management Systems" [10] in 2019, in which they presented how blockchain-based identity system are integrated into identity management. They categorized systems into a "taxonomy based on differences in architecture, governance models, and other salient features". Their key findings were that SSI is able to exist without the need for a central institution and that the technologies allow creating SSI-solutions based on them. They eventually conclude that SSI "would become a fundamental architectural component of tomorrow's Internet" [10] once it replaces non-blockchain-based identity management systems.

"A First Look at Identity Management Schemes on the Blockchain" [11] is a paper published by Paul Dunphy and Fabien A. P. Petitcolas, in which they compare existing identity management concepts with blockchain-based ones. They conclude that usability is unknown since the current approaches assume that users are familiar with DLT and cryptographic key management. Additionally, they mention the lack of regulations addressing digital identities, which is challenging for companies in designing identity systems.

Another academic research on the topic was published by Djuri Baars with his paper "Towards Self-Sovereign Identity using Blockchain Technology" [12]. In this paper, a case study was adopted where two parties exchange KYC-attributes after explicitly giving consent. While the actual exchange of attributes was executed off-chain, the exchange action, as well as the signature, were logged on the blockchain. By successfully exchanging attributes, he proved that it is possible to use blockchain technology to exchange KYC attributes. However, sensitive information was stored on the blockchain, which is undesired for productive systems.

Zoltán András Lux, Felix Beierle, et al. performed a full-text search for verifiable credential metadata on distributed ledgers as explained in their paper [13]. They successfully implemented a prototype to find credentials based on input values with a

full-text search engine by maintaining a copy of the Hyperledger Indy ledger for the Sovrin network.

J.S. Hammudoglu, J. Sparreboom et al. faced the question how biometrics can serve to authenticate and establish trust inside SSI systems. As described in their paper "Portable Trust: biometric-based authentication and blockchain storage for self-sovereign identity systems." [14], they developed a biometric-based authentication prototype that is permissionless, autonomous, and open-source. This allows the user to securely store personal information that can only be accessed after successful biometric authentication.

Chameleon-Hashes, or -signatures, are the topic of the paper [15] from Jan Camenisch, David Derler et al. that are non-transferable, and “with only the designated recipient capable of asserting its validity”. This could be a very useful tool to prevent verifiers to store and distribute presented information without the consent of the user.

3 The Evolution of Digital Identity Concepts

The Internet is in a continuous process of change since its founding in 1989 by Sir Tim Berners-Lee [16]. By inventing new protocols and technologies, the evolution of the infrastructure, and users creating content, the internet had a significant influence on the digital transformation of how societies and industries around the world behave and interact with each other. Today, the internet is being used in a wide spectrum of use cases, which makes it indispensable for almost everyone around the world in their everyday life.

Unfortunately, identity management on the internet was always an issue from the beginning. Real-world- and digital identities are currently far from being closely connected. It is difficult to use information and documents of the analog- for the digital identity, and reversed, in a simple and trusted way. The more extensive and important the internet becomes, the more significant becomes the management of digital identities. Users should get more control over how and with whom their personal identity information is used and shared online. This is what Self-Sovereign Identity stands for. The concept of SSI relies on decentralization, where no central authority but the individuals themselves control and manage their digital identity autonomously.

This chapter gives an overview of how identity is understood in the context of this thesis, how online identities have evolved, as well as giving a deep understanding of the concept of Self-Sovereign Identity. In particular, problems of currently common online identities, as well as the purpose of Self-Sovereign Identity, are shown. Moreover, the roles in Self-Sovereign Identity solutions, the guiding principles, the functionalities, as well as use cases for SSI are being demonstrated.

3.1 What is an Identity?

At the beginning of this thesis, it is important to define the term "identity". Identity is a central aspect of our everyday life, in the real world-, as well as in the online world. Due to its versatile applicability, identity exists in various forms which makes identity management very complex. It is important to look at this topic right at the beginning. Therefore, this chapter will serve to define what identity is about, how it is build up, and which aspects of an online identity are important.

Thinking about identity, a person's identity comes to mind first. Among other things, identity is associated with answers about who someone is, how others see someone,

and which characteristics define the person. But identity can be used for more than just persons. Organizations as well as things, especially smart devices, can also have an identity, even if they have different characteristics than a person's. The term "entities" will further be used to describe objects with an identity.

The purpose of an individual's identity is to define and distinguish one individual from others. Although an identity has to be unique and that there is no way of having the same identity asserted to multiple individuals, an identity has to be flexible enough to represent the entities' diversity. In the real world, identity is a combination of feasible and infeasible characteristics. Feasible could be those which describe a person's characteristic. Such as the name, eye color, or gender. Infeasible characteristics could be those which describe social interactions like openness, friendliness, or similar ones.

An identity can be presented in three levels [17], as shown in Figure 3.1.

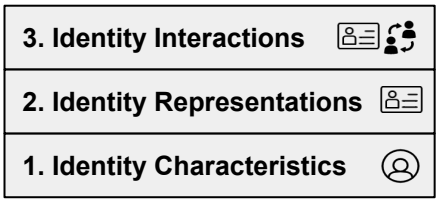

Figure 3.1: The three levels of identity. Inspired by [17].

The first and basic level describes identity characteristics where the name, age, gender, and other characteristics are being described. Those characteristics help to distinguish someone from others and describe who someone is. The second level describes identity representations. These are certificates that are issued by some entity to describe the person and show their affiliation with them, such as a person's ID issued by the state, a club membership card issued by a club, or a credit card issued by the person's bank. The third and final level is for identity interactions. Here, identity representations can be used to interact with others or something which requires a person's identity information, e.g. through identification, authentication, or participation.

Now, that the use of the internet becomes increasingly more important, it also becomes even more necessary to represent the real-world identity online in the best way possible. Unfortunately, the internet wasn't designed with an integrated identity layer resulting in many variations of identity on the application layer at the service providers' side. One identity might have multiple personas for different online services connected to its identity. Personas are characteristics of an identity related to a special environment, e.g. for online shopping, gaming, or business. This results in a significant difference between real-world- and digital identities. It is often complicated to integrate identity characteristics to the digital identity and also share information about the

digital identity in a trusted way.

In 2005, Kim Cameron published a paper “The Laws of Identity” [18] in which he describes in seven laws on how to bring an artificial identity layer to the internet. The following paragraph is derived from his paper.

In this document, Cameron mentioned as the first law that users should be in control over their identity and what they share about themselves. His second law describes minimal disclosure. Only information that is needed should be disclosed, protecting the user from undesired distribution and usage of unrequired information. In the third law, Cameron claims that only parties that have to be involved should be involved. Which parties those are should be defined by the user, so he or she has full control with whom the information is shared. So-called "Directed Identities" are described in the fourth law. These should serve as a single connection to another party that should prevent correlations between multiple connections. Cameron states "Pluralism of Operators and Technologies" as his fifth law where the identity and the corresponding technologies shouldn’t be in the power of one single party but shared about multiple ones. "Human integration" is the sixth law, in which Cameron describes that identity should be able to use and understood by everyone through easy and comprehensible interactions. As the seventh and final law, Cameron states "Consistent Experience Across Contexts". This means that independent with whom information is shared, the user should know that identity parts are shared with others and that processes follow standardized steps to reduce irritation.

Cameron’s “Seven Laws of Identity” already describe many main principles of Self-Sovereign Identity but lack of detailed knowledge on how this should be accomplished. Nowadays, Self-Sovereign Identity can give answers to Cameron’s statements. With SSI, the real world- and the digital identity should be merged more closely to each other. This should be achieved by a standard identity format, that could be used among the internet in a highly interoperable way. Real-world institutions could issue their credentials in a digital and trusted manner, resulting in reducing barriers between the real- and the online world and enabling full online identity functionality that, until now, only exists in the real world. Self-Sovereign Identity could lead to the highly missed trusted identity concept of the internet.

3.2 The Historical Development of Digital Identity Concepts

Managing identities on the internet has been an important task since its early days and still hasn’t been fully resolved until today. Providing identity in a trusted way is complex and has undergone various steps in its evolution. Christopher Allen published a paper "The Path to Self-Sovereign Identity" [2] in which he describes the evolution of

online identities. This chapter describes major steps which have been taken towards Self-Sovereign Identity.

3.2.1 Centralized Identity

In the time before Sir Tim Berners-Lee founded the World Wide Web as it is known today, some US-based universities and some US-government institutions were connected over the first computer-based network, called ARPANET. Established in 1969, it was meant to connect universities that worked together for the U.S. Department of Defense to allow them to exchange information over a long distance. Therefore the ARPANET counts as the predecessor of today's internet [19]. But the original ARPANET's protocols were difficult to scale, so the TCP/IP protocol was implemented and is still fundamental to today's internet. The purpose of TCP/IP was to assign addresses to devices connected with the internet, the so-called IP addresses. This was initially managed over a text file, called "HOSTS.TXT", by one single institution - the Stanford Research Institute [20]. Obviously, this solution wasn't very scalable so it was being replaced by a new concept. The Domain Name System (DNS) was proposed in the early 1980s and is managed by the Internet Corporation for Assigned Names and Numbers (ICANN) as a central institution until today. The ICANN coordinates and manages the DNS as well as the IP addresses.

The next milestone in the history of the internet was the invention of the HyperText Transfer Protocol (HTTP) and the HyperText Markup Language (HTML) by Sir Tim Berners-Lee in 1989. These inventions helped to make computers and the internet easier to use with the ability to create websites to present content and interact with others more intuitively. This user-friendly design made the internet more attractive, especially to private persons. The downside was that visitors of websites couldn't be sure that the website owner was indeed the owner of the website and not a fraud. To solve this problem, a Public Key Infrastructure (PKI), Certifying Authorities (CA), and later the "HyperText Transfer Protocol Secure" (HTTPS) were created. CAs were and still are today, in charge of verifying websites in the same way.

A PKI describes an infrastructure where entities have a public key as an identifier to share among the internet and a corresponding private key to encrypt and decrypt messages. Messages are being encrypted by using the public key, but only the corresponding private key can decrypt the message. CAs manage the public keys, assign identities, and approve that a website is managed by the claimed identity. However, users have to trust the CAs about the correctness of their published data as well as the correct certifying of domains.

3.2.2 Federated Identity

The chapter of federated identity is a small but important one in the evolution of online identities. A central institution provides an identity framework for its users from which a user ID and a password can be created. This combination is used as a valid authentication to access services that trust the identity provider (IDP). The first institution who tried to enable a trusted federated identity was Microsoft in 1999 with "Microsoft Passport". Eventually, the model wasn't successful because the system was inconsistent and lacked in user experience. Moreover, Microsoft acted as a central institution which gave them the power to determine which online enterprise can make use of the identity. Kim Cameron, who worked for Microsoft and was involved in Microsoft Passport, published his paper "The Seven Laws of Identity" [18] in 2005, which cover his lessons learned about online identities.

3.2.3 User-Centric Identity

Before Social Networks became popular, there already was a demand for more control over someone's own identity. Based on this movement, the Internet Identity Workshop (IIW) was founded in 2005. The IIW focused on making the user a central point of identity processes, which is called user-centric identity. If websites require access to a user-centric identity, the user has to give consent. User-centric identities shouldn't be limited to access one service, they should be interoperably usable. Based on the work of the IIW the decentralized authentication system for web applications, OpenID, was created in 2005. Once a user has created a username and a password at an OpenID-provider it could be used at partnered services called relying parties [21]. If a website wanted to use OpenID, it had to have its own protocol to get access.

Due to the lack of standardized API access control for integrating third parties, OAuth was implemented on top of OpenID to solve the problem of authorization in 2009. OAuth enabled the exchange of identity information to third parties while critical data such as authorization information wasn't sent to third parties [22]. With this functionality, popular internet companies like Google or Facebook joined the OpenID program and integrated the technology which resulted in the widespread adoption of the OpenID format. In 2010 OpenID Connect was implemented and ensured a cryptographic authentication process on top of OAuth [23]. Based on authorizations, OAuth enabled confirmation of the authentication. This concept is still the standard in many of today's online identity processes such as authentication using a Google account. It is the most widely-used identity management process because users have to remember only one single username-password combination instead of multiple ones and can avoid repeatable online registration processes.

3.2.4 Self-Sovereign Identity

Unfortunately, some online identity providers such as Facebook or Google have more than a billion users which makes them de-facto centralized institutions and centralized identity providers. They are privileged to other identity providers due to their big amount of users and especially the already existing personal identity information provided in the social networks.

By letting users access websites with these identity accounts, website providers got access to personal related information such as likes, social relationships and preferences which let them target their customers in a more efficient and personalized way. While IDPs, websites, and users benefit from such a type of identity, it comes along with huge trust and privacy issues. The identities are being controlled by the identity providers and may be banned or even deleted which could result in the loss of the identity. Moreover, the personal information of the users may be sold and used in an intransparent way, so that third parties may get access to sensitive information about a user without consent. In summation, the user reveals lots of personal information to use a user-friendly and dependent identity.

Self-Sovereign Identity slowly became a term at the middle of the 2010s when users started demanding autonomy over their identity and control over how their identity information is being shared and used [24]. With the rise of blockchain technology, there is now a possibility to create a digital identity in a fully decentralized way without the need of central institutions. Cryptography should play a central role in this concept to enable pseudonymity and secure connections between organizations and users as privacy becomes an increasingly popular topic among users on the internet. Establishing self-sovereignty on the internet is a big challenge. It has to become independent from central institutions, credentials have to be requested and securely stored, trust among the ecosystem has to be established, and many new processes and formats have to be standardized to make Self-Sovereign Identity an interoperable concept for online identities. Finally, users will have their issued credentials stored in a personal digital wallet, as in the real world today where people have their ID and driver's license in their physical wallet, and identify themselves in a privacy-preserving manner to relying parties.

3.3 The Concept of Self-Sovereign Identity

Self-Sovereign Identity is a concept that could fundamentally change the way identity is handled on the internet. From their early days, the identities were always in the hand of centralized institutions. While in the real world, where people store their identity information in a physical wallet in a decentralized way after getting them

issued, introducing a similar concept wasn't fully possible on the internet so far. The purpose of SSI is to align the online identity processes closer to those in the real world and give the user autonomy about their identifiers.

In the real world, there is decentralized identity management from the perspective of identities. A person is born and gets a birth certificate or other credentials like the social security number issued by governmental- or administrative authorities and keeps them by himself. Those credentials are used to identify the person or state a relationship over the entire life for multiple occasions. Consequently, individuals can identify themselves with the issued credentials. Third parties like the police, landlords or any other party who needs proof of an identity rely on that information and can validate if the person who is presenting a credential is the claimed person based on the presented attributes. This process is under the full control of the identity owner.

3.3.1 Problems of Currently Common Online Identities

Currently common online identities are such, that rely on centralized institutions, such as an e-mail address, a social media profile, or other identities provided by an IDP. Today, most users on the internet use common online identities to get access to services online. These identities can be used either for a single website or for multiple services or websites that accept the identity. For example by using a Google-account for authentication and authorization for Google- as well as the services provided by third parties on their websites. Due to its simplicity, this concept worked well for most of the users while continuously evolved in better usability or improved security, for example by using biometrics or Two-Factor Authentication (2FA) for authentication of the user. However, this concept doesn't fully satisfy the upcoming demand of users to be in control of their own identity. This chapter presents the problems currently common online identities have and show the reasons why a fundamentally new online identity concept needs to be established.

The main dilemma is that the user borrows the account from the IDP which represents the user's identity. While attributes of the account can be changed by the user anytime, it cannot be guaranteed that the provided identity itself stays valid or will not be taken away by the identity provider. This lets users no other choice but to blindly trust the service and their chosen identity provider. The consequences could be substantial. From the user not being able to use this identity for the website and those which depend on it, the identity may also becomes lost and unrecoverable.

Moreover, the user has to trust that the service providers securely store their personal information against theft or undesired distribution of their identity information. The services store personal data on their servers resulting in big data silos. These silos may contain sensitive data about all their customers which makes them an attractive target

for criminals who want to get access to the customers' personal identity information. Unfortunately, personal data is often not secured enough to prevent these things to happen. One of the most famous incidents are the Cambridge Analytica scandal about data improperly passed to third party data analytics companies [25] or the British Airways data breach where hackers stole personal and payment information of around 380,000 customers [26].

The internet has developed in a way where users oftentimes have to create a new online identity every time they want to access new services. The repeated creation of online identities gets annoying quickly, as users have to remember their user IDs and passwords, which becomes unmanageable already after a few times. This either leads to the reuse of passwords, which is easy to remember but highly dangerous for security reasons, or to the use of identity providers by using one master password to access the password vault but fully relying on the centralized identity provider. Once a password has become known, it could get used by others to get access to other online services.

Due to the lack of reusability of already existing identities, online services store their users' personal information on their own servers to determine who their users are. In some cases, the given data may not be correct as users can provide false identity data at registration to hide the real identity. Only for specific services, such as financial services, the identity of the user has to be verified to match the "Know Your Customer" (KYC) and "Anti-Money-Laundering" (AML) regulations. This leads to another problem as the verification of users is very complex and costly. Either users have to provide proof of their identity, like uploading a copy of their ID, or service providers acquire third parties that process the verification of their users. Doing this requires a disproportionately high effort and leads to high costs for verification. Another big problem emerges when facing the authenticity of websites. These gain trust using a public key infrastructure (PKI). By using a public and private key pair, websites can digitally sign contents and documents to prove their authenticity. However, this asymmetric encryption doesn't guarantee the unambiguous assignment of public key and associated identity. Certifying Authorities (CAs) verify correct assignments with certificates to establish trust. Expired or invalid certificates are stored in public revocation registries. However, CAs have to be certified by so-called "root CAs" who manage the certification on the highest level. This means that the trust in the certified websites depends on the trust the root CAs have in their certified intermediate CAs. This leads to centralization and a single-point-of-failure (SPoF). Despite its efficiency, PKI doesn't conform to the decentralized principles of SSI and is therefore inappropriate to use. The solution to this is called "Decentralized Public Key Infrastructure" (DPKI) which will be explained in more detail in section 4.2.

In summation, managing currently common digital identities has its limitations. It is difficult for users to manage their digital identity since it often requires creating new identities for online services, and comes with a lot of effort to memorize user IDs and

passwords while the identity is given into the hands of centralized parties. Users are not the owner of their identity, barely have any control over how their online identity is protected, or how their personal information is used and distributed. Moreover, users cannot trust the authenticity of websites because they can't fully trust the certificates issued by the certifying authorities. Additionally, some online service providers have to establish complicated and expensive KYC and AML processes to verify the user. These problems are difficult to solve in the given environment, so a fundamentally new digital identity concept is needed.

3.3.2 Purpose of Self-Sovereign Identity

Since it is now known that the problems of current digital identities are not easy to solve, it is necessary to have a fundamentally new digital identity concept. This concept should give users full control over their identifiers, reduce the complexity of managing them and increase functionality to enhance online identity to a higher level.

Reaching this goal is the purpose of Self-Sovereign Identity. It is a concept that gives people, organizations, and companies autonomy over their identifiers and full control over how and with whom the data is shared and used. This includes to only reveal the information needed for the third party, known as minimal disclosure, or selectively sharing of information, known as selective disclosure. Additionally, issuing credentials about each other is another main purpose of SSI, which is based on a Web of Trust between the parties. With simple and automated processes and standardized formats, SSI should establish an uncomplicated way for identification with others.

In order to technically implement the purposes, already existing and newly created concepts, formats, and technologies are combined to create standards for online identity management. For instance, blockchain or distributed ledger technologies (DLT) are used to create cryptographic key pairs that represent identities to sign messages or documents that are shared with others. Additionally, new standards, such as decentralized identifiers (DIDs), are created that serve as the backbone for the entire SSI ecosystem. Self-Sovereign Identity should be available to be used all over the internet for multiple different use cases among individuals, companies, and institutions. In order to create global acceptance and interoperability, standardization of processes and formats is required. This promotes the sharing, development, and global reputation of the entire ecosystem.

The most basic feature of Self-Sovereign Identity is identification with the given identifier in a trusted way, without the need for a central authority. Identification is possible by using decentralized identifiers (DIDs) which allow proving ownership of a DID with the public and private key pair associated with a DID. Furthermore, the key pair enables DID owners to sign transactions or documents. By doing so, claims

and credentials, further expressed as "credentials", can be issued or revoked and later verified that a specific public key was used to sign and issue them. Credential holders can present the credentials issued to them to third party verifiers. Verifiers can decide whether they trust a credential, based on their trust in the issuer. The verifiability resulted in the term "verifiable" claim, credential or presentation.

The combination of DIDs and verifiable claims, verifiable credentials, or verifiable presentations can be used for authentication to access or use an online service. Usually, the service provider requests data that is needed to use the service. Otherwise, the user can decide which information should be shared. Additionally, the public and private key pair associated with a DID allows asserting DIDs to things, such as any smart device, and to delegate responsibilities in a trusted way. These functionalities enable a fundamentally new way of using online identifiers and improve online identity processes while also strengthen the power of identity owners over their identifiers.

3.3.3 Roles in Self-Sovereign Identity

The structure of the SSI ecosystem can be described as a Peer-to-Peer model where entities with their independent identity act as a peer and make connections with others. Connections are used so people or organizations can attest information of others by issuing claims or credentials. Roles in SSI are the identity owner, credential issuers, verifiers or relying parties. Every entity can be each role in the system. The roles in SSI are illustrated in Figure 3.2.

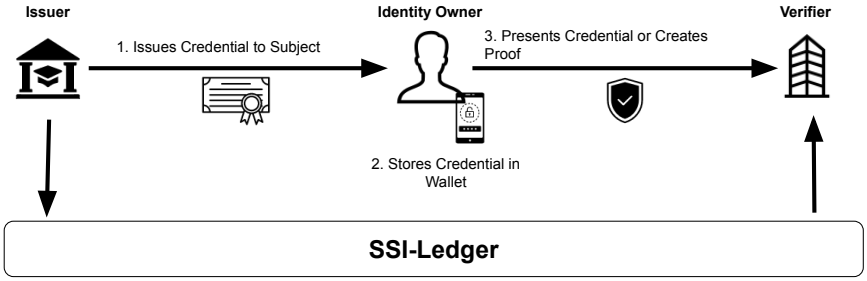

Figure 3.2: Illustration of the roles in SSI with a credential flow.

Issuers give out credentials to make statements to others or even themselves. Since credentials are issued off-chain, the wallet contains all the self-attested information and credentials regarding the identity owner. Issuers could also revoke credentials if qualifications are not fulfilled anymore.

Identity owners receive credentials and store them in a digital wallet that serves as an agent in the SSI ecosystem. Besides the issued credentials, the wallet may contain

further information about the identity owner, called self-attested claims. The identity owner could present entire credentials, parts of them, or combinations of multiple credentials in the form of proofs to verifiers. They should also be able to be presented in a disclosed or undisclosed manner, thus the identity owner has full control which data is shared and how it is used.

Verifiers offer services but need the information to grant access. They request the required information which the identity owner has to give consent to be shared. Publicly available information, like the DID and public key of the issuer, could be looked up in public registries to make sure the credentials were issued by the actual issuer and not forged by others. If the presented information satisfies the requirements, access is granted.

It is important to mention that presentations of credentials can be created and verified without the need to contact the issuer. As in the real world, presenting a credential to others, such as a student ID, doesn't require consent by the issuing university. The owner is in control of how the credential is shared and with whom. The underlying technology is mostly a blockchain or another distributed ledger technology, further expressed as "ledger". While enabling the creation of identities without the need for a central authority, the ledger serves as the root of trust. Information that affects the entire SSI ecosystem is stored here. One important piece of information could be public DIDs or public DID documents. DID methods differentiate what data is written to the ledger. This will be compared in subsection 6.2.2 in more detail.

Another optional component might be an off-ledger backend-storage. Some DID methods use permissioned or publicly available storage, like a private database or the Interplanetary File System (IPFS), to store information such as DID documents off-ledger. IPFS creates content-addressed hashes for the specific information stored on IPFS. Off-ledger backend-storage could also be used to store backup files of the wallet, so they are easy to recoverable in case of loss.

3.3.4 Principles

A fundamental identity concept needs principles that participating parties can orientate on in terms of design their complementary services. Moreover, the principles should serve as a vision for the ecosystem in order to generate trust and interoperability among services. This chapter faces the guiding principles of Self-Sovereign Identity stated by Christopher Allen which will be explained, evaluated, and extended if needed.

As already mentioned in section 3.1, Kim Cameron published a paper "The Seven Laws of Identity" in 2005 in which he described how to bring an identity layer to the internet. Given the fact that Self-Sovereign Identity didn't exist at the time he drafted these principles, it is all the more impressive that most principles were adopted as

guiding principles by Christopher Allen. Allen summarized the principles on a famous paper, “The Path to Self-Sovereign Identity”, in which he states guiding principles from other sources such as Kim Cameron, and the W3C Verifiable Claims Task Force [27]. These ten principles mentioned below serve as guidelines for participants adapting SSI and are derived from Allen’s paper [2].

1. His first principle is "Existence", which describes that every online identity must have a non-digital existence that represents and manages the online identity. Interaction with the identity is mostly managed by humans who also connect information from the real world to the online identity, e.g. certificates.

2. “Control” is Allen’s second principle. Users have to control their identities and should have the power to be sovereign in their decisions and actions they choose to make with their identity. They should be able to decide which data they share with others, for how long, and be able to refer, update or hide the identity.

3. As a third principle, Allen lists "Access", which describes that users must be able to independently access their own data. This principle expresses that no intermediaries or gatekeepers should prevent the user to access his identity or associated credentials. This should make sure that no one can manipulate or distribute the data that doesn’t have access to it.
   However, granting others access to the identity could be a useful feature. As an example, parents could take control and get access over their child’s identity and manage this identity until the child is old and responsible enough to manage it on their own.

4. "Transparency" is Allen’s fourth principle which expresses that the system should be open to participate and open in the governance, while the algorithms should be open-source, free, well-known, and as independent as possible. This leads to increased trust and to continuous improvements since every participant benefits from a better system. Additionally, participants can control the actions of each other and further detect and prevent fraud or malicious actions to happen. This increases fairness and security. However, personal information should be cryptographically hidden from non-authorized participants.

5. As the fifth principle, Allen mentions "Persistence" that addresses the need for identifiers to be available as long as the identity owner wishes it to be. It should be fully usable in the future even though keys and other any attested information may change over time. The users should always retain their identity.

6. An identity needs to be portable. Therefore, "Portability" is stated as the sixth principle which addresses the need to transport information and services about

identity without the need for a third party. This could be achieved by being compatible with different devices to reduce dependence to third parties and increase the user's control over the identity.

7. "Interoperability" is Allen's seventh principle and describes that identities should not be limited to a single service but should be usable on as many services as possible. The identity information should be accessible by services in a standardized way. This should reduce the amount of times where an identity has to be created.

8. Users should always agree with the use of their identity. Allen describes this principle as "Consent". An identity is especially valuable when information about it can be exchanged. The user should always determine which information is shared and with whom by explicitly giving consent about it.

9. "Minimization" describes that the disclosure of credentials should be as minimal as possible. To protect the user's privacy, only the minimal needed information should be requested and shared. Zero-knowledge proofs (ZKPs) allow to cryptographically prove a specific claim or value without sharing the actual information. Using ZKPs in credential presentations is therefore recommended.

10. The last principle Allen states is "Protection", where he expresses that the user's rights should always stand over the needs of a network. To preserve the freedom of the user and to keep the balance in the system, a censorship-resistant, independent, and force-resilient algorithm needs to be run in a decentralized manner

The guiding principles by Christopher Allen were a milestone for Self-Sovereign Identity as he described the principles in detail and gave the ecosystem a direction to where it should develop. However, some principles are difficult to design, since it requires the entire ecosystem to agree on one standard procedure. Interoperability may be an example as there must be a standardized format and procedure among all actors in the ecosystem. Standardization will be discussed in more detail in section 4.1.

Additionally to Christopher Allen's guiding principles, some more principles could be added to that list. One further principle could be privacy-preserving. Even though this is already partly integrated without explicitly saying it, the privacy-preserving design of services plays a key role in Self-Sovereign Identity. Reselling user-related information is a large business on the internet. The desire to gather as much data as possible is currently ubiquitous and mostly an annoying side-effect for most users. Self-Sovereign Identity should not only give full control over the identity to the user, but it should also actively prevent profiling in the network to preserve the privacy of the users. This could be done by encouraging the use of pairwise-pseudonymous identifiers

for each connection, where a special DID is generated for each connection. Third parties wouldn't be able to connect related identifiers of an identity. Pairwise-pseudonymous DID is described in more detail in subsection 4.1.1.

Another principle should raise awareness of possible attacks on the system. What happens if many identities are created to destabilize the ecosystem? How could services protect themselves against DDoS- or any other attack that has the goal to make services unusable? Even though some of these problems are already faced in standards for SSI, having a list of best practices can help developers to offer SSI-compatible services.

In order to adapt SSI on the internet, it is also important to describe how the transition from currently common online identity formats to SSI identities should work. Should identities from centralized organizations such as Google be able to be integrated into the SSI-ecosystem? Who would be in control of them? Would it be the organization or the users itself? If the organization keeps control of its users' identities it would violate the basic principle of self-sovereignty. If the users gain control over their ID, the organization would lose control and power of it which makes offering such an identity worthless to them. Hence, there should be a principle that describes the integration policy of existing identities.

Sometimes the principles contradict themselves, as described by Maciek Laskus in his blog post with the title "The Decentralized Identity Trilemma" [28]. The trilemma faces the problem to find the right balance between privacy-preserving, self-sovereignty and Sybil-resistant for any SSI system. Privacy-preserving describes the ability to utilize and acquire an identifier without revealing personal information. Self-sovereignty describes the ability to control as many identifiers as possible without interference from a third party. Finally, Sybil-resistant describes that "identity is subject to scarcity", meaning it is not possible to create multiple identifiers to manipulate the system. This trilemma is illustrated in Figure 3.3.

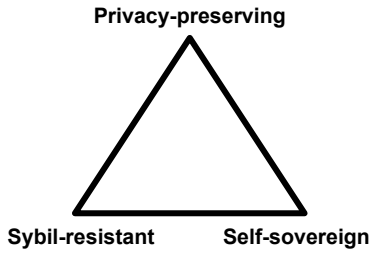

Figure 3.3: The Decentralized Identity Trilemma. Adopted from [28].

If a system wants to be Sybil-resistant, it has to sacrifice self-sovereignty and privacy-preserving because KYC processes are required which limits the ability to create multiple identities and requires to reveal personal information. The second approach

Laskus mentions is the so-called Proof-of-a-Unique-Human-Being. Biometrics are used to prove the existence of a human being without revealing personal information. Due to the lack of reusability of biometrics, the creation of identities is limited here. Another approach would be a system based on a Web of Trust. Everybody can create as many identities as possible without revealing any information about themselves. However, this isn't Sybil-resistant, confirming the decentralized identity trilemma.

As a conclusion, if SSI-compatible systems and DID methods are implemented accordingly to the principles of SSI, there will be an ecosystem that brings many benefits. It would be guaranteed that users enjoy the most possible control over their identity, that the identity can be used over the entire ecosystem, and that personal information is protected against undesired disclosure. Moreover, it strengthens the right of persons to freely connect with others without the control of a central authority. Yet, some principles may not be able to combine, as the decentralized identity trilemma demonstrated. Nonetheless, the list of principles may not be complete yet and could be extended by some more principles in the future to describe the philosophy of the ecosystem better.

3.3.5 Features and Use Cases

Self-Sovereign Identity has a broad range of possible use cases as it could be adapted theoretically everywhere on the internet where identity is involved. Even further, it creates new fields of adoption through its promising and innovative concept. This chapter shows basic features, as well as the potential adoption for industries. Furthermore, an example use case will demonstrate the use cases in a university context.

Features

The most fundamental feature, on which every other use case relies on, is authentication. If an online service requires proof, that the entity claiming to own a DID or public key is, in fact, the owner of it, an authentication process could be initialized, where the user enters the corresponding private key to prove ownership. This might sound very low-functional but could be helpful in decentralized ecosystems, where the parties don't trust each other such as in Bitcoin. Senders could make sure that the receiver's address, in fact, belongs to the receiver without knowing the receiver personally.

The feature for which SSI is most famous for and where most DID methods trying to evolve is the possibility to issue claims and credentials. While claims are simple statements of one party about itself or another party, credentials are collections of claims and are usually more formal, as described in subsection 4.1.2.

On the one hand, credentials are often digital certificates that can also be represented

as physical ones, like an official driver's license or a club membership card. Identity owners mostly request the issuance of the credential at the issuing entity but it is also possible that credentials are issued without request.

On the other hand, a claim can be a simple statement about anything, such as one person's ice cream preference. Others can affirm claims, which can increase the credibility of them, depending on the trust in the affirmators. As well as with claims and with credentials, the identity holder has added digital, trustful information to its digital identity which can be presented to third parties. By selecting the claims or credentials which should be shared out of the available credentials, the verifier gets information about an identity in a trusted, automated, and fully digitized manner. If the requirements satisfy the needs of the verifier, access could be granted, other credentials could be issued or verifier's processes could be triggered automatically.

Use Cases

Self-Sovereign Identity is a concept that is open to and usable by everyone. This includes individuals, companies and also governments. Individuals can benefit from having sole control over their identity, which is the main purpose of SSI itself. They are free to connect with any service they want, without the need for a central institution or having the risk that their personal identity information will be stolen or compromised in any other way.

Companies could use SSI to digitize and automate their processes, which could result in big savings through increased efficiency. Additionally, their employees could use decentralized identifiers and credentials to be authorized for the use of services and the possibility to revoke the authorizations which leads to a more dynamic identity management process.

Moreover, governmental institutions could enable direct connections with its citizens to issue credentials like companies. Governmental processes are oftentimes highly bureaucratic due to high trust restrictions, complex dependencies and lots of manual processes. With its trusted characteristic, credentials such as birth certificates, testimonies of residency, or driver's licenses, and more could slim many internal processes and maybe erase the need for manual processing in a high degree resulting in much quicker processes. This could be particularly useful in emerging markets, where the official infrastructure is not sufficiently developed or difficult to access.

Increased performance and less bureaucracy could also positively affect many business areas such as financial services. People could request credentials from banks or other institutions to prove their capability to get a loan more easily.

However, not every business sector wants to adapt SSI. Making money out of personal-related information of its customers is a large business. Industries that generate and

sell such data, as well as their receivers, could lose their business model. For instance, data from Facebook to recommend products based on the user's interests is used by many e-commerce businesses to better target their customers. The business value will be reduced, if users decide to not share or even create this kind of data or by making the data unlinkable to a real-world identity due to the use of pairwise-pseudonymous identifiers. These may be some reasons why industries won't adapt SSI technology, at least in the near future.

Among the SSI ecosystem, the technology could be used for other use cases as well. As an example, Ocean Protocol uses DIDs to identify data sets that can be used for decentralized data exchange to unlock data for artificial intelligence use cases [29]. It is expected that SSI technology could be used for similar use cases in the future.

University Example

Each entity in SSI can play every role, which is the issuer, credential owner, and verifier. The connection between each entity persists until one party terminates it. The following paragraph demonstrates use cases, where a university plays one role in each of them.

A university can serve as an issuer of credentials. For example, the university could offer to issue a diploma credential that serves as a digital transcript of records. Every grade could be listed as well as the average grade and other related information. This credential could be used for an automated application process at companies where students want to apply for. Issuing digital student IDs as a credential is also a potential use case for a university. This credential could contain information about the matriculation number, course of studies and validity date. Students could use this credential to verify their status as students for online services that give student discounts, for instance.

If the university serves as credential owner, it could prove attributes of itself. In the case that the university only exists online, it could present credentials that show their certification as a university, issued by government authorities. Furthermore, it could present credentials over partnerships with other companies or memberships in other organizations. This creates trust in the online university and potential students might be more willing to apply there.

Last but not least, the university could be a verifier. To get enrolled, many documents need to be presented by the applicants, such as a valid copy of the ID or diplomas of earlier degrees. Requesting this information in credential format would improve the application process since a higher degree of automation can be achieved. Information could be automatically verified and result in instant matriculation or rejection.

4 SSI Components

Solutions enabling Self-Sovereign Identity don't have to be implemented entirely from scratch. There are already many components and standards on which the SSI community has agreed on to make the development of SSI-compatible solutions easier. While some components were already in existence and required adjustment, some components and standards have to be designed from the beginning in order to fulfill the principles of Self-Sovereign Identity.

The combination of already existing and newly created components leads to the component architecture of the SSI ecosystem and to the creation of a Peer-to-Peer architecture where every entity has its own independent identity and is free to connect with others. The purpose of this chapter is to present the components of the SSI ecosystem in detail, describing their purpose, the implementation of the required redesign, as well as their functionality and role. Afterwards, the components are set in context in the new components architecture of SSI.

4.1 Open and Fundamental Standards in SSI

Creating a decentralized ecosystem where multiple parties are interacting with each other requires agreement on how the interaction should take place. The implementation of proprietary solutions is often the easier choice for single actors, as the solution could be integrated into existing systems more smoothly. Nonetheless, if the ecosystem wants to achieve interoperability, the participating parties have to agree on standardized formats and processes. Standardization is a central pillar in the goal of creating a decentralized ecosystem for Self-Sovereign Identity. Even though it may require more effort to connect existing systems of actors to the SSI ecosystem, all participants benefit among other thing from increased efficiency through common processes, better verifiability, and better scalability.

This chapter focuses on the most important standards in the SSI ecosystem. In the first step, decentralized identifiers and their associated functionality will be introduced. Afterwards, claims and verifiable credentials will be defined and their role will be presented. Finally, the concept of verifiable presentations out of claims and verifiable credentials will be put in context, as well as a demonstration of a credential life cycle.

4.1.1 Decentralized Identifiers

Conventional identifiers, like usernames and passwords, are the most common identifiers on the internet. As previously described in subsection 3.3.1, those identifiers don't belong to the users themselves, as they are borrowed from the service provider and are mostly only valid inside the service provider's ecosystem. This leads to a lack of control for the user over the identifier, as well as security risks since the service providers store their users' personal identity information on their storage devices. These centralized data silos are attractive targets for criminals to gain access to this valuable information in order to sell or use them by themselves for criminal actions.

As a consequence, a decentralized system needs decentralized identifiers (DIDs). The DID specification [30] of DIDs was created and published by the W3C Credentials Community Group [31] under the W3C Community Final Specification Agreement (FSA) [32]. The DID specification is still not an official standard yet but is already mature enough to design systems and processes around it.

In the Self-Sovereign Identity ecosystem, entities are represented with decentralized Identifiers (DIDs), which are globally unique, never change, and which can be generated on a DID-compatible ledger. Creating the DID in this way makes the DID independent to central institutions and prevents the DID from being taken away from the identity. A DID represents a portable, digital identity generated and addressed by a public key on a ledger, whereas the DID holder is in possession of the corresponding private key, enabling full control of the decentralized identifier for the DID holder [33]. Not only people can be represented by a DID. It is also possible to create a DID for other entities that might be relevant on the internet, such as a DID for a machine, an organization, or even digital documents. Moreover, so-called decentralized autonomic data (DAD), which contains a DID, can also uniquely identify data items or data streams in the SSI ecosystem, as mentioned in the paper "A DID for Everything" [34] by Shaun Conway, Andrew Hughes, et al. from "Rebooting Web of Trust VII".

A real-world entity can create an unlimited amount of DIDs to use for lifetime persistent connections with others, until it is terminated by one of the parties. Generating new DIDs for each connection could be beneficial to prevent tracking on the side of the providers. The process of creating new DIDs for every connection is also implemented in some of the already existing SSI systems and is called pairwise-pseudonymous DIDs [35]. These DIDs enhance privacy since it prevents linking of identities. If two services want to compare user behavior, all they can relate to is a DID which only exists for their connection. The only information service providers would have stored in the DID, so even if the data is breached, it is useless as it can't be assigned to a real identity. Every pairwise-pseudonymous DID can be automatically generated on the client-side, including a public and a private key.

All the DIDs have to be managed by a real-world entity. This is achieved by storing and controlling them in an agent at the entity's side, such as a digital wallet on a smartphone. How DIDs and their public- and private keys are managed, is described in more detail by explaining DKMS in section 4.3.

The syntax of DIDs creates uniquely identifiable identifiers that serve as identifiers in the Decentralized Public Key Infrastructure (DPKI) in the SSI ecosystem. The syntax of a DID is defined in Figure 4.1.

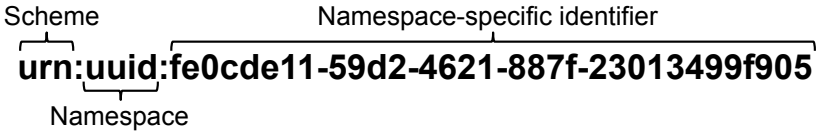

Figure 4.1: The DID syntax. Adopted from [30].

The first part is a scheme as a Uniform Resource Name (URN), followed by a namespace in the form of a Universally Unique Identifier (UUID), and lastly, a namespace-specific identifier that uniquely identifies an entity inside the namespace. By definition, the URN and the UUID must be lowercase. A real and valid DID inside the Sovrin namespace would look like demonstrated in Figure 4.2.

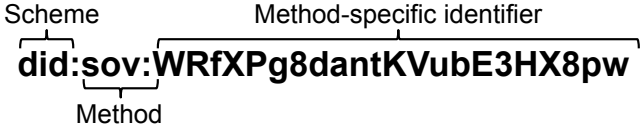

Figure 4.2: An example Sovrin DID. Adopted from [30].

Additionally, it should be possible to directly refer to fragments of the DID, such as a public key listed in the DID document by adding a fragment behind the DID, as shown in Figure 4.3.

Figure 4.3: An example DID with a fragment. Adopted from [30].

Moreover, querying the DID should be able to directly request data from the DID document by adding a query behind the DID, as shown in Figure 4.4.

Figure 4.4: An example DID with a query. Adopted from [30].

Moreover, testnets or other subsystems of the DID method could be accessed by having a subsystem identifier between the method identifier and the method-specific identifier. An example is shown in Figure 4.5.

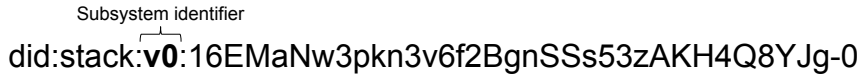

Figure 4.5: An example DID with a subsystem identifier. Adopted from [30].

A DID itself is only an identifier and serves as key in a key-value schema. The value is the corresponding DID document (DDO) which describes valid public keys of the DID, service endpoints, and more, to support interactions with other identity owners. Generating or resolving the DID document of a given DID is called DID Resolution. Since every DID method could create a DID document differently, resolving them is a crucial task to enhance interoperability.

DIDs can be generated in different ways, depending on the underlying blockchain or distributed ledger technology that provides functionality to create a public- and private key pair. The process of creating DIDs is defined in so-called DID methods, which will be explained later in this section, as well as DID documents and DID resolution.

Asserting personal identity information to an identifier is achieved by issuing and receiving verifiable credentials, or creating any other personal information in the form of claims and attesting it to the DID. This could also be achieved by self-attested claims containing the information, which could then be verified by others.

However, there are still some open questions regarding DIDs. As an example, it is still unclear whether DIDs should be opened up to centralized institutions, such as Facebook which controls a massive amount of user identities. This is a dilemma between easily onboarding new users, and finding a deal with the identity providers (IDP) who don't want to give up control over their users' identifiers. The same dilemma is also applicable for higher trustworthy standards like the eIDAS regulation, a highly regulated online identity standard in the European digital single market [36], as Oliver Terbu describes in his paper "Decentralized Identities and eIDAS" [37].

In summation, DIDs are the most important standard of the SSI ecosystem. Every entity and every interaction relies on this standard to uniquely identify an entity. It

enables users to independently create a DID by using a blockchain or distributed ledger and hold the keys under their control to make it self-sovereign. A DID has to be persistent, resolvable to create or get a DID document, cryptographically verifiable through the use of public- and private key pair, and fully decentralized. This standard and its derived processes create a high level of trust. However, there are still some points not fully thought out yet, e.g. how to integrate already existing identities into SSI, similar to the European eIDAS identity standard.

DID Method

Depending on the underlying blockchain or distributed ledger technology, DIDs can be created in different ways. While DIDs that are based on common public blockchains, like Bitcoin or Ethereum, rely on their protocol, proprietary ledgers could solve similar tasks by designing their ledger protocol according to the identity challenges right from the beginning. To sum up, DID methods specify how DIDs are created, read, updated, and deleted within a specific underlying ledger. DID methods don't change the original purpose of a ledger, rather, it extends its functionality, making them capable of interacting with other services using their DID.

In particular, a DID method specification has to define [38]:

1. The DID method name
2. The structure of the method-specific identifier
3. The generation of a method-specific identifier
4. The CRUD operations on a DID and DID document
    a) **C**reate or Register a DID on the ledger
    b) **R**eading or Resolving a DID document
    c) **U**pdating a DID document
    d) **D**eleting or Revoking a DID

Creating a DID could be achieved by creating a cryptographic key pair, or by using an already existing address of a smart contract on the blockchain itself. As an example, Bitcoin Reference, uPort, or ERC725 generate the DIDs by using addresses compatible to their ledgers. In these cases, the address serves as the method-specific identifier.

Reading or resolving a DID depends on the underlying blockchain technology. DIDs may be stored directly on the blockchain. In Bitcoin Reference (BTCR), the DID refers to the block and transaction that contains the sender's address. Otherwise, resolvers could dynamically generate the DID document based on the transaction logs and events on

the blockchain in the same way as the balance of an address is created. An alternative way implemented in BTCR is to store a pointer on the OP_RETURN data field that refers to the DID document in a publicly available backend storage, like IPFS.

Updating a DID document is necessary if the containing information is changed, for instance, by adding another public key. Only valid keys that are registered in the DID document should be able to perform update operations.

Last but not least, operations to delete or revoke a DID make the DID unusable for future interactions. For instance, this could be done by deleting the DID out of the "DIDLedger" smart contract in the SelfKey DID method. There are different ways of revoking a DID depending on the DID method on the underlying blockchain or distributed ledger technology.

There are already some DID methods officially registered in the DID Method Registry of W3C which satisfy the requirements of a DID method specification. Some of the most popular ones are listed in Table 4.1.

Table 4.1: DID methods with example DIDs.

DID Method	**Example DID Syntax**
Sovrin	did:sov:WRfXPg8dantKVubE3HX8pw
Bitcoin Reference	did:btcr:xksa-czpq-qeuw-qcg
ETHR	did:ethr:0x3b0BC51Ab9De1e5B7B6E34E5b960285805C41736
Blockstack	did:stack:v0:16EMaNw3pkn3v6f2BgnSSs53zAKH4Q8YJg-0

The more DID methods that are being integrated into the SSI ecosystem, the more diverse it becomes. Oftentimes, special DID methods are created to serve the purpose of the underlying blockchain. DIDs could serve as a key to the interoperable use of identifiers among the different solutions inside the SSI ecosystem, by using standardized processes on how to interact with the identities.

The DID specification provides requirements for DIDs in order to establish a high level of interoperability [39]. DID methods specification should determine...

1. ... how to provide a unique identifier that isn't already used by another DID method
2. ... how to support the required CRUD operations
3. ... how to describe operations that require a description
4. ... a specific, detailed, and complete enough specification to enable independent implementation
5. ... security and privacy considerations for the DID method.

DID Document

DID documents are the counterpart of a DID. While a DID is an identifier inside the system, a DID document contains information about the DID, such as associated public keys and service endpoints, allowing the decentralized identifier to interact inside the SSI ecosystem in a cryptographically verifiable way. DIDs are the indexes, while DID documents provide the information. Once a DID document is assigned to a DID, it will never be reassigned or reused for someone else.

DID documents are created as a JSON-LD file [40]. By using JSON-LD (linked data), data is transferred in a standardized structure and thus allows two different systems to exchange data in a standardized format that both systems understand. The DID document must be stored on an accessible source. This could be on the blockchain directly, a pointer stored on the blockchain referring to the DID document, or completely off the blockchain on any publicly available storage. A minimal DID Document could look as shown in Figure 4.6.

```
{
  "@context": "https://www.w3.org/2019/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // 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": [{
    // used to retrieve Verifiable Credentials associated with the DID
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

Figure 4.6: An example DID document. Adopted from [41].

The format of the DID document is defined in the DID specification [41]. The DID document begins with the `@context` attribute, which defines the version of the DID document format.
The standard value for this attribute is the URI `https://www.w3.org/2019/did/v1`.

Defining the version of the document serves as a terminology framework which both parties understand.

The second attribute is `id`, which describes the subject of the DID to which this document is associated to. By definition of the DID standard, there is only one value allowed that has to be valid. Additionally, there may be a controller attribute, which defines a controlling entity that is authorized to perform specific services on behalf of the DID. This may be useful if the actual DID subject can't perform actions by itself, for instance, a machine needs someone who controls the machine and performs actions.

The next attribute contains at least one public key. The public key is used for digital signatures, encryption, and other cryptographic operations to enable interactions like authentication. Moreover, public keys are important for the authorization for CRUD operations. Attributes of the public key are its id, type, owner and encoded public key properties, in this case, `publicKeyPem`. If a DID document doesn't contain a public key, it must be assumed that the DID has been revoked and is invalid.

Authentication information is described in the next attribute `authentication`, which is used to cryptographically prove ownership of the DID. This attribute contains authentication types as well as the public key used for authentication. Authentication information isn't required in a DID document, but may be included.

The `service` attribute defines provided service endpoints where others can interact with the DID, such as authentication or further discovery of related information. A service endpoint could be a link, a QR code, or an API for interaction. There may be additional endpoints for different services. However, service endpoints aren't mandatory in a DID document as well. Moreover, there could be a timestamp integrated that could serve for auditing purposes. The timestamp documents the time of the last action, like the creation or update of the DID document.

Last but not least, a JSON-LD signature may be added in order to verify the integrity of the document. Only authorized entities, like the DID owner or its controllers, should be allowed to perform operations on the DID. Unauthorized actions should either be ignored or flagged, to prevent malicious manipulation.

Creating a DID is not always a requirement for creating a DID document. Christian Lundkvist describes in his paper about InterPlanetary Linked Data (IPLD), "IPLD as a general pattern for DID documents" [42], a method to generate a DID based on a DID document. The InterPlanetary File System (IPFS) identifies files through a hash generated from its content. Lundkvist proposes a DID method where the hash of the file represents the method-specific identifier. The main problem is that the DID document must contain the DID, but whenever the DID or the hash of the DID document is inserted into the DID document, the hash changes. Therefore, some sort of a placeholder is inserted like `did:example:this` to enable the correct DID to show the correct DID value. Updated DID documents could be generated that refer to the

original DID.

According to Lundkvist, this procedure could be implemented for pairwise identifiers that only serve for one connection. The DID document and the DID could be sent to the other party for verification, and thus proving ownership of a DID.

DID Resolution

DID resolution describes the process of getting from the DID to its associated DID document. It is the basis for creating connections, initiating interactions, and proving ownership via DID Auth. DIDs should be globally resolvable, allowing others to look up the DID document. Due to the fact that every DID method works differently and has different ways of creating and storing a DID document, resolving them works differently for each DID method.

There are three possible scenarios of creating a DID document out of a given DID. As the first option, the DID method has native support for DIDs and DID documents, meaning that they are stored to the ledger. As another alternative, the DID method only has support for DIDs but not for DID documents, resulting in dynamically creation of the DID document based on transactions. As the third possibility, the DID method neither has support for DID documents nor DIDs. In this case, the public address has to be defined based on the method-specific identifier.

In the DID method Bitcoin Reference (BTCR), there is neither native support for DID documents nor for DIDs. The method-specific identifier in BTCR consists of an encoded pointer to a specific transaction inside a block on the Bitcoin blockchain. An example of a BTCR DID would be `did:btcr:xksa-czpq-qeuw-qcg`. In order to resolve the DID document, the method-specific identifier has to be decoded to get to the storage location of the DID document. Based on the sender's address of the transaction, the resolver looks for connected transactions that contain DID relevant information to dynamically construct the DID document. In order to assert information to a DID document, a second output is created in the OP_RETURN data field where any kind of arbitrary data can be stored, like a URL to an off-chain document or information for service endpoints. The resolver adds or deletes this information to the dynamically generated DID document. This procedure is documented in the BTCR resolver document [43].

In Sovrin, DIDs are natively integrated into the ledger, whereas DID documents have to be dynamically generated using specific transactions that indicate changes to the DID document. To resolve a Sovrin DID document, the resolver goes through the transactions that happened and searches for transactions where the DID was involved. If there are transactions found, the resolver chronologically generates the DID document for the corresponding DID. The resolver extracts specific information from the transaction, decodes it, and adds it to the DID document. If the resolver finished

searching through transactions and reaches the most recent transactions, it is free from doubt, that the DID document is up to date [44].

A DID method that has native support for DIDs and DID documents is Veres One, as described in the Veres One DID method specification [45]. Given the method-specific identifier, the resolver can easily fetch the entire DID document from the ledger. DID Documents can manually be created and stored on the ledger. Therefore, no complicated steps have to be taken, and the DID document can be easily accessed for the ledger.

These were just three examples of how DID documents can be created, utilizing different underlying blockchains or distributed ledgers. Other DID methods follow identical, or similar approaches, and also fall under the three previously presented categories.

In order to establish an interoperable system, a community project of the "Decentralized Identifier Foundation" (DIF) was established to create the universal resolver [46]. It is able to resolve any kind of DID over a generic API that targets the underlying ledgers of a DID method, making it easier to resolve DID methods among the SSI ecosystem.

The universal resolver serves as the intermediary between a service or an application that tries to access a method-specific ledger [47]. Operations requesting information of a DID document could be achieved using a method-specific driver which should be as close to the ledger as possible, for instance by running a node. Unfortunately, this could result in undesired powerful intermediaries in a decentralized system. If a new DID method should be accessible via the universal resolver, it needs to provide a driver to receive requests and to define a way to access the ledger. On the other hand, services and applications should also be as close to the universal resolver as possible to prevent MITM attacks. They could connect to the universal resolver by using a natively linked library or by using an HTTP API. In the best-case scenario, the universal resolver is built into browsers to provide the best usability since they have the highest potential to process requests from services automatically. A model of the universal resolver is demonstrated in Figure 4.7.

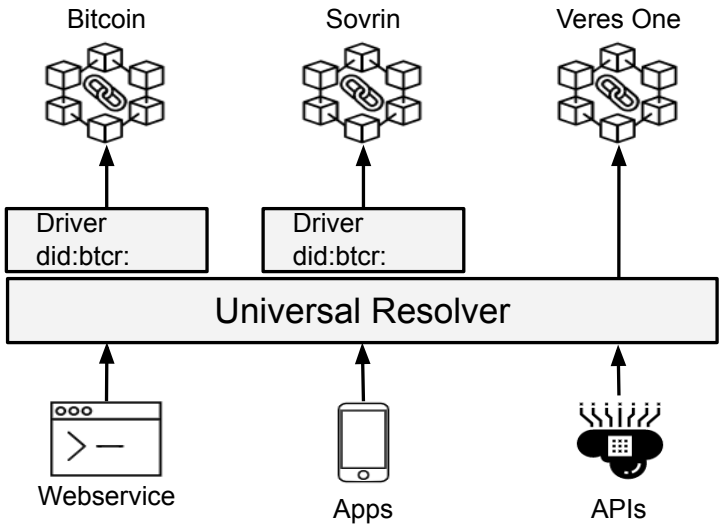

Figure 4.7: An illustration of the DID resolver model. Adopted and extended from [48].

In addition to the functionality of resolving DID documents from DIDs, the resolver bears the potential for further features. For example, specific resources of the DID document could be accessed directly, such as service endpoints, or a specific version at a specific time. Furthermore, frequently requested data could be cached to improve performance. Besides, resolver metadata, like the name of the driver used for resolutions, or method metadata like block height of the DID document could be responded [48].

Similar to the architecture of the universal resolver is the architecture of the universal registrar [49]. The universal registrar provides a set of libraries that make it possible to register, update, and revoke any kind of DID in any kind of target system by invoking specific functions. Creating those transactions may require transaction fees, or contacting a trust anchor to perform the transaction. Eventually, if the operations succeed, a method-specific identifier is returned [50]. Unfortunately, there is only limited documentation due to it currently being under development.

To summarize, resolving DID documents based on given DIDs or method-specific identifiers plays an important role to get further information about a DID. DID resolution works differently for every ledger, however, ledgers can be categorized based on different levels of native support for DIDs and DID documents. DID resolvers have huge potential as they are the perfect access point to implement new functionalities. Moreover, the universal registrar tries to provide a standardized access point to register, update and revoke any kind of DID. Unfortunately, there is only limited documentation about how it should work in detail.

DID Auth

Cryptographically authenticating an identity owner is a central goal in the SSI ecosystem. In order to establish trusted connections, it is crucial that users can prove that they control a specific DID as long as the DID exists. In SSI, this specification is called DID Auth and relies on a challenge-response authentication protocol using the identity's- and relying party's public and private keys, registered in their DID documents. DID Auth can be used by every DID which supports this standard. As previously explained, the DID document contains information about the DID, authentication, and service endpoints which may be necessary to establish a trusted connection between two parties.

Markus Sabadello et al. describe the concept of DID Auth in their paper "Introduction to DID Auth" [51]. There are always three components in a DID Auth process. The first one is an identity owner who controls and is identified by a DID. The second component is a third party to which ownership should be proved. As the third component, there is the challenge-response cycle, where the actual process of authentication takes place. It is designed after the following algorithm:

1. If the DID of the identity owner is known, the relying party sends a cryptographic challenge, e. g. via JSON Web Token, to the provided service endpoints of the identity owner for authentication. The challenge should include a nonce to prevent replay attacks.
   Information about the identity owner's DID isn't required. Hence, why generic challenges can be generated and embedded in a QR code. The sender may or may not include a proof of ownership of their own DID in this challenge.

2. After the identity owner receives the challenge, a response is created. This response contains a proof of control over the DID, such as a signature using a public key registered in the DID document. A response could also be verifiable credentials that were asked for in the challenge. Sending, as well as receiving components may be different for each interaction.

3. Based on the information in the response, the third party can validate that the identity owner controls the DID by resolving the DID document and validate the public key.

In fact, there exist multiple ways to interact between the identity owner or the third party. The third party, for instance, could directly send an HTTP POST to the DIDs service endpoint for authentication if the DID is known. Otherwise, a generic QR code could provide a request to prove ownership. Receiving the challenge and sending

a response from the identity owner's side, as well as the third party's side, can be processed from different devices which have dependencies. For instance, the service endpoint in the DID document could be an agent that notifies the identity owner while the response is sent over the identity owner's wallet. Compared to traditional identity authentication protocols like OAuth, DID Auth only has two parties involved instead of the identity provider as the third party.

There can be different formats used in DID Auth, with the most common being the JSON Web Token (JWT) [52]. This format encodes claims as a JSON object which can be cryptographically signed and transferred to other parties. It is an established and mature data format, which has proven to be ideal for authentication processes. An additional format is JSON-LD, which is especially suitable for verifiable credentials, as it provides the same signature functionality as a JWT. Additionally, HTTP could be used as well but is complicated as it would require a signature specification for HTTP which adds an HTTP header with the relevant information. Having JWT or JSON-LD as data formats provides the simplest and smoothest way to send information required for DID Auth, as presented by Markus Sabadello in the SSI Meetup "Introduction to DID Auth with Markus Sabadello" from July 11th, 2018 [53].

Authentication is often associated with biometric authentication to log in to services or unlock data. In SSI, biometrics don't play a major role but can be implemented as well. The most common integration of biometrics is to unlock a wallet or an agent where the private keys of public keys to a DID are stored. However, if biometric information has to be used to prove ownership of a DID, biometric data has to be part of the DID document. A service provider could then offer biometric authentication where the resulting value has to match the value in the DID document. Even if an individual's biometrics remain consistent, it is still a possibility that they could change as a result of an accident, for example. Consequently, the DID document needs to be updated to align with the changed biometric.

To summarize, a successful DID Auth interaction creates trust between both parties and serves as a basis to form a connection for further interactions, such as issuing credentials or any other data exchange. DID Auth is doing this by relying on the challenge-response authentication protocol, where signatures or any other form of proofs are cryptographically created. Instead of the conventional authentication process with an identity provider involved, DID Auth only needs the identity owner and a relying party. All the required information to validate the proofs can be found in the identity owner's DID document. Designing a DID Auth process is up to the parties themselves, as long as the challenge-response authentication protocol is used.

4.1.2 Claims and Verifiable Credentials

The Self-Sovereign Identity ecosystem was created not only to have sole control and autonomy about an identifier but also to enable actors to make attestations about themselves or others. This could be done by attesting claims or issuing verifiable credentials. The Verifiable Credentials Working Group (VCWG) of the W3C has created a data model [54] covering claims and verifiable credentials which defines a standard format and procedure on how claims and credentials should be issued or used. This chapter gives an overview of claims and credentials, the difference between them, and how they are structured.

Claims

According to the VCWG, claims can be used to make a statement about a subject and can be described as a subject-property-value relationship as shown in Figure 4.8 [54]. For example, a claim could be made about someone being a student at a certain university.

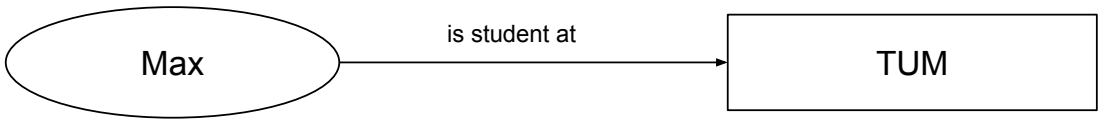

Figure 4.8: An example for a claim represented as directed graph. Adopted from [54].

It is worth mentioning that claims have a more informal nature than credentials. Claims can be made by anyone about anybody and anything. A student could claim, for example, that he has the best grade point average in his course without explicitly knowing it. Therefore, this claim can naturally not be trusted. In order to assert a higher level of trust in the claim, it can be verified by third parties. All other students that have a worse grade point average could verify the claim based on their knowledge. Unfortunately, it is technically not possible to mark a claim as incorrect. This would be useful if another student had a better grade point average to state the claim as incorrect, based on his knowledge. In this case, the only way to trust this claim is to have it verified by the university or a professor of the course, as they know whether this student has the best grade point average or not.

This is a concept of making claims verifiable. Others could verify claims, but it is up to the third party as to whether the verifier can be trusted or not. In a claim-based Web of Trust (WoT), introducing contestability would increase the trust in the system, but it could also lead to harassment and trolling.

Verifiable Credentials

In contrast to a claim, a verifiable credential is a more formal certification, as it comes in a similar form of a physical document. It contains at least one, but oftentimes more, claims and comes with a signature when the credential is issued. A digital credential is highly needed to manage online identities in a simple way. Physical credentials often have to be digitized in an unnecessarily complex process in order to use them online, e.g. creating a scan of the credential and uploading it. Furthermore, physical credentials have some disadvantages, as they are easy to fake, can be lost or damaged, expensive to create, and disclose more than is needed.

With the use of digital verifiable credentials, the disadvantages of physical credentials become advantages of verifiable credentials. This is intended to achieve by integrating digital claims, cryptographic signatures, or minimal disclosures. Some of these advantages might be, that the identity owner can decide which information should be disclosed in which way, that processing of information could happen automatically, higher scalability, or increased trust by digital and verifiable signatures. The data model of a verifiable credential may consist of up to three parts, as illustrated in Figure 4.9.

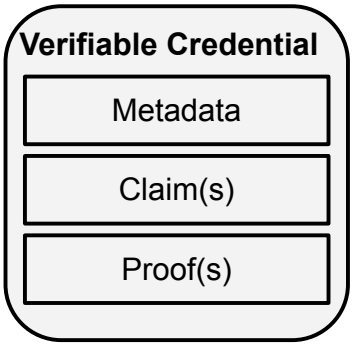

Figure 4.9: An illustration of the verifiable credential data model. Adopted from [54]

First, a verifiable credential may contain metadata that describes the properties of the credential, such as a credential identifier, a public key of the issuer, or a timestamp. These metadata may be signed by the issuer. The main content of a verifiable credential is at least one or a set of claims. It's often the case that each attribute of the verifiable credential is a single claim. Therefore, only the required claims have to be disclosed, whereas other claims remain untouched. This is an important feature that creates a higher level of privacy and a big advantage of verifiable credentials. By signing the verifiable credential in the third part, a cryptographical proof is generated to demonstrate who issued it. The proof contains a type, like RsaSignature2018, a

timestamp of the creation date and time, the nonce, the signature value and the public key of the issuer. This information is important for third parties in order to verify the presented data.

After receiving a credential, the credential holder stores it inside a wallet on a personal device, such as a smartphone. This wallet can be used to form connections with other participants, where credentials could be used for authentication or to prove authorization to use a service. Which information of a credential goes on the ledger also highly depends on the DID method. In general, native information of verifiable credentials should not be written on the ledger due to its immutable nature. However, identifiers of credentials could be stored there, which is useful for revocation registries but also requires someone to spend transaction fees, depending on the underlying ledger.

In order to prevent impersonation of verifiable credentials, there has to be a secret to prove that the credential was issued to one specific owner. This is achieved by introducing a "Link Secret", which can be any form of characters in any length that only the identity owner knows. They transform the "Link Secret" and send it to the issuer who embeds it inside the proof. When the proof is shared with the verifier, it doesn't invalidate the signature of the issuer but makes it able to verify by a third party. The signatures in combination with the "Linked Secret" prove that the credential was indeed issued to the credential holder, as explained by Dmitry Khovratovich in his paper "Anonymous Credentials" [55].

Integrating verifiable credentials into the daily business of verifiers brings plenty of benefits. To begin with, verifiers can save money due to the complicated and costly verification processes not being needed anymore. In order to satisfy KYC regulations, the verifiers can request a verifiable credential to verify the user. A further advantage is that they don't have to collect personal information. They save the DID and maybe other metadata of their users, so there is no need for data silos for service providers anymore. The data that would be stored on the verifier's side is useless for online criminals because the information is either encrypted or a pseudonym. Last but not least, verifiable credentials allow a much faster, less complicated processing of personal information and require fewer authentication processes as before.

Unfortunately, there are currently only a few regulatory aspects of using verifiable credentials. For instance, it is unclear whether cryptographically proofs are valid for processing or how disputes should be processed.

Verifiable Presentations

Presenting information from one or more claims or verifiable credentials to a verifier is called verifiable presentation. It is possible to either present the credential directly

or derive data formats from verifiable credentials that are cryptographically verifiable. The subject, to whom this presentation refers to, is the same in most cases. The verifiable presentation has a similar data model as the verifiable credentials, as shown in Figure 4.10.

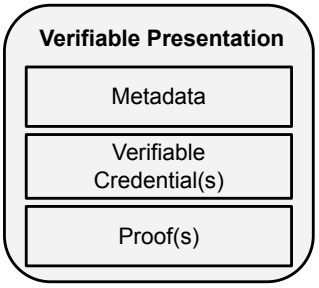

Figure 4.10: An illustration of the verifiable presentation data model. Adopted from [54].

At first, there might be metadata to describe information about the presentation, such as a type or terms of use. The main content comprises the information from the verifiable credentials, which by themselves contain the metadata, relevant claims, and proofs of each involved verifiable credential.

Furthermore, the verifiable presentation is signed and thereby generates a digital proof. As defined in a verifiable credential proof, the presentation contains information about the type, the date and time of creation, the nonce, the signature value and the public key of the presenter. The proofs could either be created by JSON Web Signatures (JWS) [56] to sign a JSON Web Token (JWT) [52], by Linked Data Signatures [57], or by Camenisch-Lysyanskaya ZKPs [15].

Eventually, the verifier can look up the public keys and DIDs of the signing parties to verify the authenticity of the presented data. It's up to the verifier whether those signing parties can be trusted or not. Additionally, the presenter could add a validity or revoke the presentation.

One key tool for verifiable presentations are Zero-knowledge proofs (ZKPs), which enable enhanced privacy to keep the balance between personal privacy and institutional integrity. This is achieved by cryptographically proving a statement in a way where the original data is hidden. One of the most common ways to describe Zero-knowledge proofs and their benefits is to verify the age. If a verifier wants to make sure that the user is over 18, only the proof that the user is indeed over 18 is required, not the actual date of birth. In this way, the privacy of the user is respected and the verifier got the information needed in a trusted way. Consequently, the information a verifier gets is also less sensitive, which makes it less attractive for digital criminals to get access to

the data. In SSI, ZKPs are used primarily for digital proving of statements, as well as for authentication.

Credential Lifecycle

In a credential life cycle, there are three entities involved. The issuer who issues and revokes the credential, the identity owner or credential holder who stores the credential in a wallet and presents containing information to others, and the verifier, to whom credential information is presented to verify the data. Additionally, the ledger is also part of the credential life cycle since they contain revocation registries and may be used to register public DIDs. The credential life cycle is illustrated in Figure 4.11.

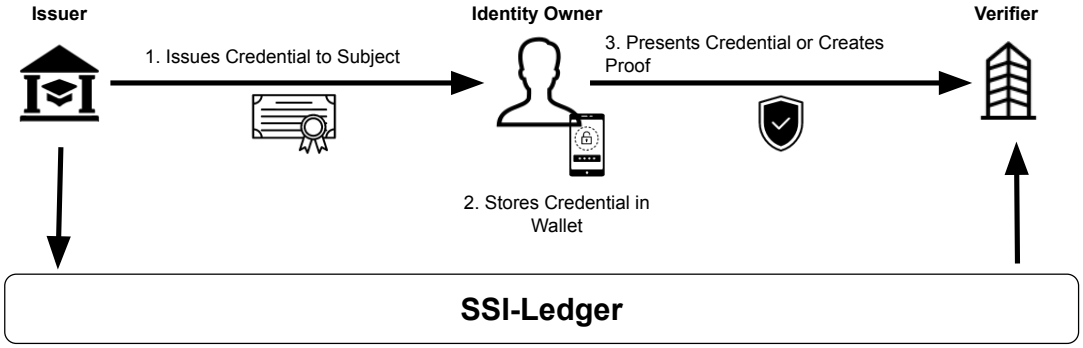

Figure 4.11: An illustration of a credential lifecycle.

Credential Issuing

Offering and issuing a credential is the first step in the lifecycle of a credential. This step could be different for every use case and also depends on the ledger used. In most cases, the issuer already has information about a person or thing with an identity and is, therefore, able to offer a credential. For instance, an issuing party, such as the local citizen office, already has information about who lives in their district and could offer a credential that confirms this information to others. On the other side, there may be dependencies to other credentials where the third party doesn't have to store information about an identity but only grants access to a service. A process to issue a credential could look like the following steps:

1. The issuer registers its DID on the ledger. This makes it easier for others to find the issuer and verify its credentials. This is an optional step and isn't required or possible in many SSI systems.

2. The issuer writes a public entry of the credential definition to the ledger. The credential definition contains information about the attributes' properties as well as metadata that describe the credential. The credential definition helps verifiers to directly request specific attributes. This step is also Sovrin-specific and may not be required in other DID methods.
3. (Optional) An identity owner requests a credential at an issuing party. In order to make sure the identity owner is eligible to receive the credential, the identity owner has to present other credentials to the issuer, such as proof that a specific minimum age is satisfied. The credential request is also not mandatory. Sometimes, a credential could be offered by the issuer as well.
4. The issuer creates a credential that contains information in the form of claims about an identity owner and signs the credential with its private key.
5. The issuer issues the credential to the identity owner over one of the provided service endpoints.
6. The identity owner stores the verifiable credential in a wallet on a personal device. From now on the identity owner can also be called credential holder.

Credential Presentation

The identity owner has full control over how this credential is used after it is stored inside a wallet. It can be used for further forming of connections or to present to third parties that require this information.

1. In order to make use of a credential, a connection with verifiers, also called relying parties, has to be formed.
2. The verifier requests data in order to grant access to a provided service or to issue another credential.
3. The credential holder selects the required information out of one or more credentials or may generate a proof from the selected data.
4. The data or the proof will be presented or disclosed to the verifier.
5. The verifier checks if the data or the proof satisfies the requirements, looks up the public DID information on the ledger to verify the signatures, and checks whether the credential is listed on a revocation registry.
6. The verifier accepts the data or the proof access is granted, another credential is issued, or further data exchange can be processed.

Credential Revocation

Even though credential holders have full control over how these are shared and used, the issuer always has the option to revoke a credential. If the requirements that are linked to a credential are no longer satisfied, the issuer revokes the credential. The identifier of the credential or any other form to distinguish one credential to another will be published on a revocation registry. This registry is stored on the ledger and can be checked by verifiers if a credential presented to them is revoked.

4.2 Decentralized Public Key Infrastructure

Establishing trust in the internet is very difficult to achieve. Over the years, the internet has become more secure, e.g. with the integration of SSL and TLS. Users can be sure that no unauthorized intermediary can read or retrieve the information they send over the internet by using SSL or TLS to create encrypted connections between the parties. Nonetheless, this doesn't mean that it is entirely safe. It is important to have a closer look at the protocols and certificates that are meant to establish trust on the internet.

The most common way to identify anything on the internet is to use a Public Key Infrastructure (PKI). Among other things, it is used to establish a secure connection on the Internet through the digital signing of documents or e-mails. This is achieved by the use of a pair of public- and corresponding private keys to encrypt and decrypt information. Public keys serve as the identifier of the actor's identity in the PKI and have to be unique. These could be URLs in the Domain Name System (DNS), IP addresses for devices on the internet, or e-mail addresses.

One problem with PKIs is that they require centralized registries and registrars which are managed by Certificate Authorities (CAs), resulting in a centralized and hierarchical trust infrastructure. If an online service wants to be registered under a special domain in the DNS, the CAs lend the domains to them. This could lead to data ownership problems or credibility issues due in part to the CA being the owner of the domain and not the online service itself. CAs are accountable for managing identities and preventing losses through insufficient protection of entrusted data. This results in the investment of lots of money and time for the CAs in security purposes, to keep this centralized trust infrastructure running.

Singing X.509-based certificates for domains is one purpose of CAs. Once a domain gets certified, it will appear as trusted in most browsers. Due to the great number of existing domains and the resulting afforded maintenance, it is possible for so-called "root CAs" to give their right to sign and certify domains to intermediary CAs which they trust. If one of those intermediaries falsifies or fraudulently certifies a domain, then the user believes the domain is trustful, while it actually may not. As a result, the

users have to trust the CAs since there is no other institution that checks the correctness of a CA and its certificates.

Centralization always comes along with a single point of failure or a single point of control and is, as a consequence, a lucrative target for online criminals. Man-in-the-middle (MITM) attacks are one example of how centralized systems can be attacked. A successful intervention could lead to the fraudulent issuing of certificates, which happened in 2017 when hackers successfully took control of a Brazilian bank's DNS server [58] by issuing a wrong certificate. Therefore, a successfully executed attacked root- or intermediate CA could result in massive problems or financial losses and bring down all the trust in the system.

These systems were established in times where no decentralized approach was possible, and the data had to be stored in centralized databases in the "HOSTS.TXT" file managed by the Stanford Research Institute [59]. Now that decentralized systems have become increasingly popular, there is a way to resolve the problems of PKI. The solution is the "Decentralized Public Key Infrastructure" (DPKI), according to the "Rebooting Web of Trust Group" and their published paper on DPKI in 2015 [7].

With DPKI, no central authority has to create and manage the keys of actors because it can be done by the users themselves in a decentralized way. This could be achieved by using blockchain- or any other decentralized ledger technology. Users can generate a key pair on the client-side, without the need for a central authority. With the structure of a key-value storage, the identifiers, which serve as keys, can be assigned to data that is able to be globally readable. The value of the identifier is always linked to the identifier's most recent public key to prevent criminals from successfully comprising the integrity and security mechanisms of the system. The system creates trust by agreeing to a global status with the help of a consensus mechanism. To follow the rules of this system, an integrated reward scheme should incentivize the validators to do so. To improve trust, internal processes are open-source, transparent, and immutable.

The identifiers in the SSI ecosystem are the decentralized identifiers (DIDs). The given format of a DID is highly influenced by DPKI since it makes a DID uniquely identifiable among multiple ecosystems like a blockchain. DPKI itself is not a blockchain but a protocol that is applied in order to securely access and leverage blockchain technology. One potential design aspect could be that every blockchain gets its own DPKI Top-Level-Domain (TLD), e.g. `.bit` for the Bitcoin-, or `.eth` for the Ethereum blockchain. It is unclear whether or not it is even required since TLDs usually make identifiers human-readable. Yet, a part of the SSI community says this isn't a desirable feature due to privacy reasons, as mentioned in an RWOT7 document by Greg Slepak [60].

To summarize this chapter, DPKI adapts the concepts of PKI for the challenges of Self-Sovereign Identity and its fully decentralized approach of identity. Instead of a centralized identity provider, a user can create an identifier using blockchain technology

without the need for a central authority. There are no Certifying Authorities anymore, as the data on the blockchain serves as a central root of trust on which all actors agree on. Moreover, users can cryptographically sign their data with their public and private key. As a conclusion, DPKI provides a set of functionalities that enable a decentralized approach to identify subjects in the SSI ecosystem.

4.3 Decentralized Key Management System

Managing identities or assets stored on the internet is an essential and difficult task for users. Losing a key often results in complicated key recovery protocols, or even in the loss of the identity if there was no backup made. For centralized services, an integrated key- or username and password management is often given, so that a lost password can be easily reset. However, this means that the service provider has full control over its user's keys. In the SSI ecosystem, the DIDs are generated in a decentralized manner. Following this decentralized approach, decentralized identities require a decentralized key management system (DKMS). This chapter will give an overview about what DKMS is, how the architecture looks like, as well as emphasizing the key recovery and key revocation functionalities.

DKMS is an open and emerging standard which describes how users can manage their DIDs and public and private key pairs in an interoperable way. It is especially important for wallets, in which the users store their DIDs and private keys and for the SSI agents that read and write from those wallets. The purpose of agents is to define how to connect and communicate with services in a secure way, especially for authentication, managing authorizations, and sharing data with others. In the best-case scenario, agents and wallets fully adapt DKMS as their standard to achieve a higher degree of openness and interoperability, which increases usability and flexibility for the users. Furthermore, users never have to worry about security, privacy or vendor lock-in when using a standardized wallet.

DIDs are designed to persist and to represent addresses of public keys in a ledger. While the address should never change to retain the existing connections, it should be possible to generate a new key pair for the existing DID. Michael Lodder proposes in his paper "Recommendations for DKMS" [61] that users have a master private key linked to the identity to authenticate the user's actions and subkeys for each connection to others. If a public key of a DID becomes known, it should be changed, which is called rotating keys. This increases security without the need to reconnect with other parties.

Another task for DKMS is allowing users to have multiple DIDs inside a wallet or an agent. Utilizing pairwise-pseudonymous DIDs for each connection with another

party would lead to an amount of as much DIDs as existing connections, that have to be managed by DKMS.

DKMS should preserve privacy, allow endpoints to manage their own secrets, and provide a method for establishing trust for securely distributing public information. Endpoints can be implemented in the form of decentralized identifiers (DIDs) and any keys used by that DID in DID documents are also called DID description objects, which provide anonymity for identities and their keys.

4.3.1 DKMS Architecture

DKMS follows a special architecture as it implements three layers between the identity owner and the underlying ledger, as shown in Figure 4.12. The basis of DKMS is a public ledger where DIDs can be created. On the other side, there is the identity owner who controls its identity over a so-called edge agent with an integrated edge wallet to it. An edge device could be a smartphone or a notebook. Between the Edge Agent and the public ledger is the edge cloud. The cloud agent is a service that coordinates messages between the different levels and could also have a wallet integrated. The cloud agent is useful to store information for an edge agent if it is not connected.

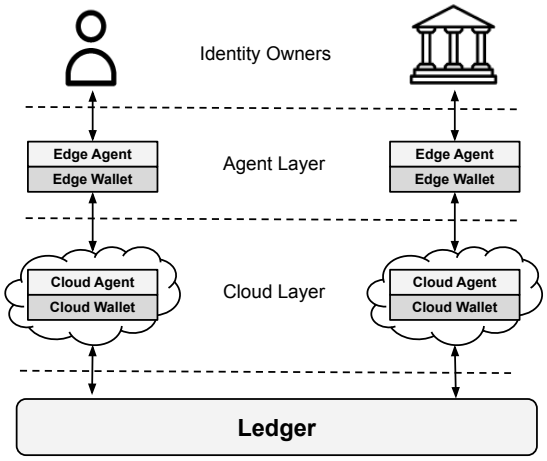

Figure 4.12: The architecture of DKMS. Adopted and extended from [62]

The architecture of DKMS comprises of three layers. The first one comprises of a ledger as the basis. The blockchain serves as the single source of trust, so users can trust the data it contains. To protect the system from unauthorized manipulation, only the user's public key or master public key must be allowed to interact with its initially written data. Lodder also recommended enabling delegation of ledger interaction to

third parties on behalf of the user, using certificates or signed keys by the user [61].

Users need online devices with an interface that allows them to interact with the SSI ecosystem. These devices are called edge agents and serve as the storage for private keys for DIDs and management of the user's identities and are defined in the second layer of the DKMS architecture. Having this information stored directly at the user's agent and not on a cloud agent increases security as it is harder for criminals to get access to it. The most convenient usable devices for an agent are mobile devices because of maximum control for the user and availability since it's almost constantly by the user's side and a high uptime. A mobile device is able to have a wallet installed on it to store the user's keys or the location of the others' endpoints. Furthermore, using a mobile device makes it is easier to create connections through scanning codes of the other party. Moreover, it can be used to generate link secrets to prove that a credential was issued to the user to obtain and issue credentials, as well as store digital assets like tokens.

Edge devices may not always be online to interact with others and only have reduced computing power. These problems can be resolved using cloud agents in the third layer of DKMS, which have their own cloud wallets. Their purpose is to sign or prove actions or documents and send messages around to the ledger or other cloud agents. Furthermore, cloud agents allow users to have multiple wallets across different devices which increases flexibility. However, different keys should be used as it could be unsafe to share keys across wallets since if one wallet gets hacked, other wallets can be accessed the same way. Additionally, cloud agents can be used for backups and recovery in case a wallet is lost or hacked. Due to the importance of edge- and cloud agents, it is important to standardize as much of it as possible to increase interoperability in the SSI ecosystem.

4.3.2 Key Recovery

One big problem with giving control of their identity to the user is that people forget their keys or lose their devices. Therefore, key recovery is an important task for DKMS. The entire recovery process requires that the users make backups of their wallets. DKMS should provide the functionality to recover wallets in a user-friendly way.

Users should make backups of their wallets regularly and store the encrypted backup on the cloud agent or any other secure digital storage. To bring the information to a new edge agent, the edge agent connects with the cloud agent, downloads a copy of the wallet backup and decrypts it using a special recovery key. Decryption could be achieved in two ways; offline recovery and social recovery. Both of them use cloud agents to store an encrypted backup file for the wallet.

Offline recovery describes the use of a seed value which was generated at the initial

wallet creation. The seed must be stored somewhere else, for example, by being written on a piece of paper that is securely stored, but accessible. However, people tend to forget where the recovery keys are stored which is an issue of usability.

Social recovery could use sharding of the recovery key as a recovery mechanism. Their key is divided into multiple pieces which can be shared with trusted parties, such as family and friends. These encrypted artifacts can be stored at the trustees' wallets. The problem with social recovery is that users tend to forget who the trustees are or that the contact might get lost. In this case, users could be permanently asked to list the trustees or provide the functionality to add new ones. The recovery key artifacts can be shared with multiple trustees, so only a subset of all trustees are required to recover the recovery key.

With people forgetting their keys, where the recovery key is stored, or with whom they shared the recovery key, technology should implement better recovery mechanisms resulting in easier ways to recover the keys. However, the increased risk due to a higher chance of unauthorized access has to be respected, as it shouldn't be too easy for criminals to recover and access keys.

4.3.3 Key Revocation

Another crucial task of DKMS is to allow key revocation. If a public key becomes known, it is rendered useless and should be revoked. A new one has to be added and take over that functionality. Revoking keys also comes with the problem that verifiers might not be aware of which keys can be trusted. To create a level of transparency, Lodder suggests implementing accumulators that manage revoked keys in a revocation registry [61]. Verifiers could quick-check those registries to find out if the provided key is on the list of revoked keys and consequently, not trusted for establishing a connection.

4.3.4 Creating Connections By Using DKMS

Issuing and verifying credentials uses all levels of DKMS. In order to be able to issue and receive a credential, a connection between the two parties has to be formed. This could be initiated from the issuer's, as well as the identity owner's side. In the following example, the identity owner initiates the connection process.

At first, the edge agent could directly go to the issuer's service endpoint, if known or ask the cloud agent to look up the issuer's DID and the required service endpoint, if the endpoints aren't known. Then the edge agent sends an encrypted message to the issuer's cloud agents endpoints, which pushes the message further to the edge agent. From there on, an introduction protocol will be processed. The issuer can look

up the DID document of the identity owner, get its public key, and send an encrypted message back to the identity owner. After the two parties have agreed, a connection between them is formed. Since there is no interaction with the ledger itself, the process of issuing credentials is high performing.

In order to initiate a process, either the DID of the issuer has to be known or a potential introduction service on the cloud agent level has to be processed to receive the DID or endpoint information of the issuer. Unfortunately, there is no introduction service already implemented. The DID must be publicly discoverable on the issuer's side, for instance with a QR code on the website, or directly implemented into the website's metadata.

The edge- or cloud agents could also provide the functionality to create pairwise-pseudonymous identifiers or even pairwise endpoints, which are only used for this single connection. This improves privacy on both sides.

In summation, DKMS is necessary to establish interoperability among the SSI ecosystem. It contains standards that define how keys should be managed, stored and protected and how interaction among the SSI ecosystem could be established. Moreover, it recommends approaches for key recovery as well as the key revocation. Unfortunately, these concepts aren't binding to SSI actors. It cannot be guaranteed that systems follow this approach, even if the ecosystem would benefit from it. Additionally, there could be lots of features added. Furthermore, it bears the risk of centralization at the cloud layer, if one cloud wallet provider gains monopoly-like status due to better services.

4.4 Trust Infrastructure

Trust plays an important role in the SSI ecosystem since it has a decentralized infrastructure where trust isn't established by a central institution. For this reason, achieving accountability and having reliable and tamper-proof information is essential for achieving trust in the Self-Sovereign Identity ecosystem and also one of its biggest challenges. While in centralized systems, where the user has to trust one central institution, the community needs to agree on formats, standards, and common processes in decentralized systems. Current methods of establishing trust requires a fair amount of effort. Verification processes become slow and expensive, especially when data has to be verified by multiple parties. Information from only one identity information system is often insufficient to establish trust, particularly in decentralized systems where identity attributes could be self-attested [19]. Here, the users don't know each other and have no registry to look up if another party can be trusted. Additionally, the users have to trust that the information in the system is protected against fraud, digital profiling, or any form of attack. The solution is to establish trust with technology, achieved by

standardized, open-source and transparent processes in a decentralized manner, like the accurate management of DIDs or signatures. Managing the identities is brought from centralized institutions to the user itself.

The purpose of this chapter is to describe how trust is established in the SSI ecosystem by emphasizing the role of the blockchain or distributed ledger technology, the concept of a Web of Trust, as well as governance and trust frameworks for decentralized systems.

4.4.1 The Role of Blockchain and Distributed Ledger Technologies

Blockchain and other distributed ledger technologies (DLT) enable users to store data in an immutable, transparent and distributed manner without the need for a central authority. These characteristics make blockchain and DLT fit for creating an advanced identity management ecosystem in a decentralized manner which satisfies the principles of SSI.

Although the used ledgers may have different characteristics, they all share the characteristic of serving as the root of trust on which all participants agree on. This makes ledgers the underlying element of the core infrastructure in Self-Sovereign Identity systems.

Blockchains are mostly characterized by openness and transparent processes. Openness means that everyone is free to join the system while open-source means that code and processes are publicly visible and auditable. This makes it possible for anyone to participate by running a node to verify transactions. Transparency means that the data flow is also publicly visible which makes fraudulent actions easier to detect. Despite the transparency, the privacy of the individual is protected. In the SSI ecosystem, users are identified by their decentralized identifier (DID) backed with at least one or multiple public- and associated private keys using a decentralized public key infrastructure (DPKI) as already described in section 4.2. Blockchains enable the creation of DIDs and registration inside the SSI ecosystem without revealing any personal information.

To establish trust, those DIDs could be registered on the ledger, so that others can see which public key is associated with it, which is especially useful for issuers whose credentials are issued with this specific DID. Having DIDs registered on the ledger will be used to verify signatures on data such as a verifiable credential. Entries that are published on the ledger are stored in an immutable way and publicly visible. The ledger must prove that any changes made its data are only made by those that have been authorized. One way to manage unauthorized requests is to ignore or flag the entry by the system. If anyone else would be able to change the data, the security of the system is compromised, which should be prevented by the underlying protocol. By doing so, blockchain enables trust in connections and other actors without the need for a centralized identity management system.

The information that could be stored on the ledger may differ within each SSI-system but is mostly information that is relevant to maintain the web of trust, such as DIDs, credential definitions, revocation registers or even non-sensitive information that describes an identity better. Revocation registries play an important role in terms of security. Here, other actors could check whether a DID is still active or not. Additionally, credential issuing parties could store revocation registries with the revoked credentials to present already revoked and invalid credentials. To preserve privacy, credentials are not stored on the ledger.

A big criticism of blockchain or DLT is the lack of performance or high transaction fees. This may be a problem for some DID methods applying on public blockchains like Bitcoin Reference (BTCR) on the Bitcoin blockchain. However, the main message traffic in the SSI ecosystem happens directly between the agents. Only minor and mostly non-time-sensitive processes, like a DID registration, interact with the blockchain itself. Creating the public and private keys doesn't happen on the blockchain itself but on the devices of the participating parties. The process of issuing a credential happens entirely off the ledger. As a result, performance is therefore not a big factor.

In summation, the ledgers serve as the root of trust in the SSI ecosystem. It enables users to create and register DIDs in a decentralized way. Processes and data are traceable, documented, and publicly visible on the blockchain. The characteristics of blockchains, such as openness and immutability increase trust in the system resulting in a trustless environment, where users don't have to trust other actors but trust the underlying technology.

4.4.2 Web of Trust

There are different models of creating trust inside an ecosystem. One trust model is the hierarchical trust model. Here, there are entities which can create identifiers or add value, such as a public and a corresponding private key, to it and verify the identity with a given verification process. As a result, verified entities could get a certificate by the verifying authority which leads to a hierarchical structure in the system. Usually, verifying parties have an official license to act as a verifier in the system. This model is highly used in already existing ecosystems, like the DNS, where CAs certify domains and give their rights to other intermediate CAs to act on their behalf. Disadvantages of this model are the risk of providing a single point of failure and being prone to MITM attacks where attackers get control of or manipulate a verifier and use its power to manipulate the system. Yet, the centralized approach doesn't conform with principles of SSI where decentralization is one key principle.

The second model of creating trust is the "Web of Trust" (WoT). It describes a trust model in which all actors are equal without the need of a centralized entity and gives

actors the option to verify others or things. The more actors are participating and verifying, the more trustworthy the system becomes. If one actor gets verified by five others, this actor is more likely to be trusted than an actor who only has one verification. Not only is the amount of verifications important, but also who verifies can make a significant difference. As an example, a student claims he or she has a diploma from a specific university and get this claim verified by 20 of his or her friends. Simultaneously, another student claims to have a degree at the same university and gets this claim verified only by the university itself. It is more likely that the student verified by the university is more trusted than one verified by his friends.

Web of Trust in PGP

In 1991, Phil Zimmerman published the "Pretty Good Privacy" (PGP) concept [63]. It is used for an e-mail based asymmetric encryption and signing of data. The e-mail addresses served as the public key of an identity and had equal rights among all entities, whereas one identity, or peer, could act as publisher and verifier. PGP tried to answer questions of whether a public key actually belongs to the given identity and whether a public key belongs to someone trusted to certify other public keys [64]. E-mail addresses became more trusted once they got verified by one or multiple others. Due to the main focus on verification on PGP, a group from "Rebooting Web of Trust" published an article "Rebranding Web of Trust" , where they pointed out that the WoT is more about verification than trust, leading them to suggest rebranding it to a more fitting "decentralized key validation system" [65].

PGP can be technically described as a directed graph model for verification, as illustrated in Figure 4.13.

Figure 4.13: An illustration of a trust chain as a directed graph model.

A verified B, B verified C, and C verified D, and so on, resulting in dependencies and a verification structure formed like a web-structure under the assumption that e-mail addresses are used for verification and object of verification multiple times. Given the example above, a third party doesn't trust actor D. D was verified by C, so the third party can reevaluate the decision depending on the credibility of actor C. If the third party trusts actor C, it's more likely to trust actor D. However, if the third party doesn't trust actor C as well, the third party could follow the verification path back until the

root verification of actor A about actor B. If the third party only trusts actor A, it is likely that the trust is transitively transferred up to actor D.

PGP was an early decentralized trust approach for identities but wasn't successful for a number of reasons, as described by Henry Story in an article [66]. One issue was that e-mail addresses, which represented the identity, were under the control of institutions that provide e-mail addresses under their domain. Therefore, the concept was under the power of centralized institutions by design. Another issue it failed was due to the lack of functionality. Only a few attributes, namely the e-mail address, name, and a picture could be added and verified, while it also lacked in usability. More disadvantages of the classic WoT are also described by Ouri Poupko in his paper "A Trustless Web of Trust – Establishing Identity Uniqueness" [67]. One of them is that trust is subjective. Just because actor A trusts B doesn't mean that actor B is trustful in general. Actor A and B might be criminals and artificially create trust between them to seem more trusted to others. As a result, others don't know whether actors A and B are indeed trustful. Moreover, verifying others might lead to a cluster of identities or groups with common characteristics that are distinct from others which reduces the usability for an entirely accessible WoT. Last but not least, trust could also be gained more easily, due to better skills or political knowledge to encourage others to verify you more often. However, the idea of having full control over an identity and the ability to verify claims was a positive response for PGP, eventually leading to SSI.

Having a PGP-like web of trust isn't the only way to get trusted connections, as mentioned in the paper "Peer-to-Peer Degrees of Trust 1.0" by Harrison Stahl et al. from the Rebooting Web of Trust VII [68].

Another way could be a trust structure similar to a social network. If actor A is connected with actor B, and actor B is connected with actor C, there is a high chance that actor A wants to connect with actor C. This could also work for credentials or any other sort of connection like a recommendation system. It could be recommended for actor A to connect with actor C because the already connected actor B is connected with actor C.

The concept of a WoT could be, and is already, adapted to modern SSI systems. Replacing e-mail addresses with DIDs and data with public and private key pairs follow this approach of a WoT. Verification could be done by looking up the public keys of signatures and comparing it with the signer's public keys listed in the DID document, where all public keys are listed. If the issuer is trusted, the presented claim or credential can be trusted, too.

The Concept for a Modern Web of Trust

Trust in SSI is established by adopting the Web of Trust from PGP. It is established by making the DIDs transparently accessible and verifiable, for example by writing the DID on the ledger or by exposing the DID to web pages. This makes it easier to verify signed documents or issued credentials and can lead to verification chains and consequently to a Web of Trust.

However, it still needs some optimization to align with today's standards and processes. The paper "Rebranding the Web of Trust" focuses on the adaptability of the WoT to today's SSI and suggests separating entities and actions [65].

The main finding of the paper is that since identity and verification topics should be addressed by the entities themselves, actions should address processes between entities. Therefore, an entity- and action model are defined in the document. The entity model describes the identity itself, as well as verification- and authentication processes surrounding the identity. On the outside of the entity model, attributes define the identity, such as a person's name. These attributes can be verified by others.

The action model defines how entities interact with each other in five sequential steps. Those steps comprise deciding which data should be published, displaying the data and the expected result, actual processing, interpretation of the result, and finally comparison of the expected to the actual result.

Additionally to the five sequential steps, there could be another step added called "Manipulation of Feedback". In this step, an automated system can take and adjust the data according to specific rules previously defined. The result of this model could minimize bridging problems between entities by discouraging negative feedback or maximize bridging problems depending on the created WoT. The interaction between identities based on the action model is illustrated in Figure 4.14.

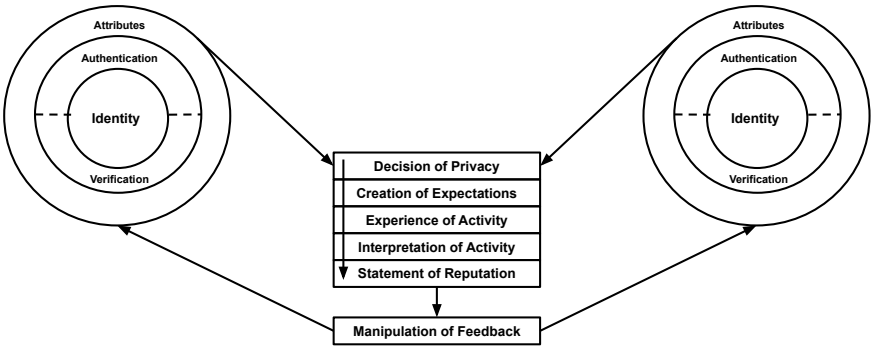

Figure 4.14: An illustration of the interaction model between two identities using the identity- and action model. Adopted from [65].

In this modernized concept of the WoT, trust will come from the perceptions of other actors. It will increase in their interactions with others as well as the observation of interactions between others. It has the potential and is designed to not repeat the mistakes from the original WoT used in PGP. Using the entity- and the action model could result in a better-defined process flow between the interacting entities.

There could be additional features implemented to increase trust in the system, such as a browser add-in that displays embedded DIDs on a website of an entity. This could maybe include categories of trust that indicate users the trust level of a website. Yet, it raises the question of how the levels or categories are defined in a decentralized way.

Furthermore, it is unclear whether this concept will be successful since it is not yet implemented in this way. It may need some modification but this will be clear once it is actively used. Another problem that could occur are Sybil-attacks. These attacks describe when fake IDs try to attack the integrity and functionality of the system as they actively share false information. The decentralized identity trilemma by Maciek Laskus describes discusses how it is difficult for any SSI system to achieve the right balance between privacy-preserving, self-sovereignty, and being Sybil-resistant [28]. This problem could be solved using governance and trust frameworks inside an SSI-System and is investigated in subsection 4.4.3.

4.4.3 Governance and Trust Frameworks

Decentralized systems have to manage themselves as there is no central authority which makes decisions for the system. The actors inside a system have to manage it solely by themselves, which requires business, legal and technical policies. These are written down in a governance model for the associated decentralized system which acts as a constitution for the system. Among other things, the governance model defines which roles exist and which rights and obligations these groups have. Moreover, it defines how decisions are made inside the system, which processes exist, and how they work. Every actor inside the system has to agree on this model for using it. The purpose of a governance model is to establish trust in the system by having transparent processes and a clear listing of rights and obligations. Governance models play a huge role in the SSI ecosystem. SSI solutions are either built upon an already existing blockchain or created a new one and where a governance model is required around the system. In governance, multiple layers of participation for stakeholders with varying incentives and knowledge are necessary. Governance models should be well thought out, otherwise, something like the DAO hack [69] could happen. In this example, users made use of inconsistent governance and the resulting technical opportunity to repeatedly transfer value out of the system into their own accounts.

A trust framework is something less administrative as it defines actions and rights to

establish trust inside a system. If more than two participants are involved in the use of a digital asset, there must be a trust framework behind it. This guarantees that the relying party can trust the information provided by one of the other parties.

There are two types of trust frameworks involved. To begin with, there is the general trust framework that is applicable for the entire ecosystem and contains basic processes, like defining who is eligible to write information on the ledger or how decisions are made. SSI-compatible systems have different characteristics, like being built upon already existing ledgers, or built on ledgers designed to support SSI, it is difficult to agree on one basic, global trust framework among these methods. However, the SSI ecosystem should be as standardized as possible. That's why Sovrin published the "Sovrin Governance Framework" [5] which can be used by other methods as well as basic trust framework. An alternative approach is implemented by Advogato [70], who uses seed nodes to generate trust scores for their members. Even though this violates the entirely decentralized approach, it is one way to control SSI systems.

Next, there are domain-specific trust frameworks that apply only to a specific field. These specify who issues which credential and which policies in order to achieve the desired level of trust. Furthermore, there may be additional rules and guidelines which are only applicable within a specific domain. For instance, terms and conditions define how digital credentials of a club membership can be used. Actors define their own schemas for their special purpose without contradicting the basic trust framework. Standardized schemas will become more established as more actors will participate over time.

Trust frameworks aren't limited to SSI solutions. In fact, the real world needs trust frameworks too, in order to align their processes with the digital ones, e.g. accepting digital credentials in the real world world. One project which tries to achieve this is the Pan-Canadian Trust Framework (PCTF) [71]. It describes a set of criteria and specifications to ensure that all jurisdictions in the private sector abide by a common agreed-upon set of rules to trust and accept each other's digital identities. This framework is still under development and faces different challenges like aligning KYC and AML processes of banks with those of the PCTF.

In conclusion, governance models and trust frameworks are necessary to give an SSI solution business, legal and technical policies while trust frameworks define how processes are designed to establish trust. Trust frameworks can be separated in general- and domain-specific trust frameworks. Once the SSI ecosystem reaches common acceptance, it will be demonstrated if the modern WoT concept is successful or whether it needs some rework. Moreover, it is important for offline services to align with digital services using trust frameworks to bridge the gap between the real- and online world.

4.5 SSI Components Architecture

In this chapter, an architecture model will be created based on the previously introduced components. The purpose of this architecture is to put the components in context and to visually present the implications between each component. The components architecture model is shown in Figure 4.15.

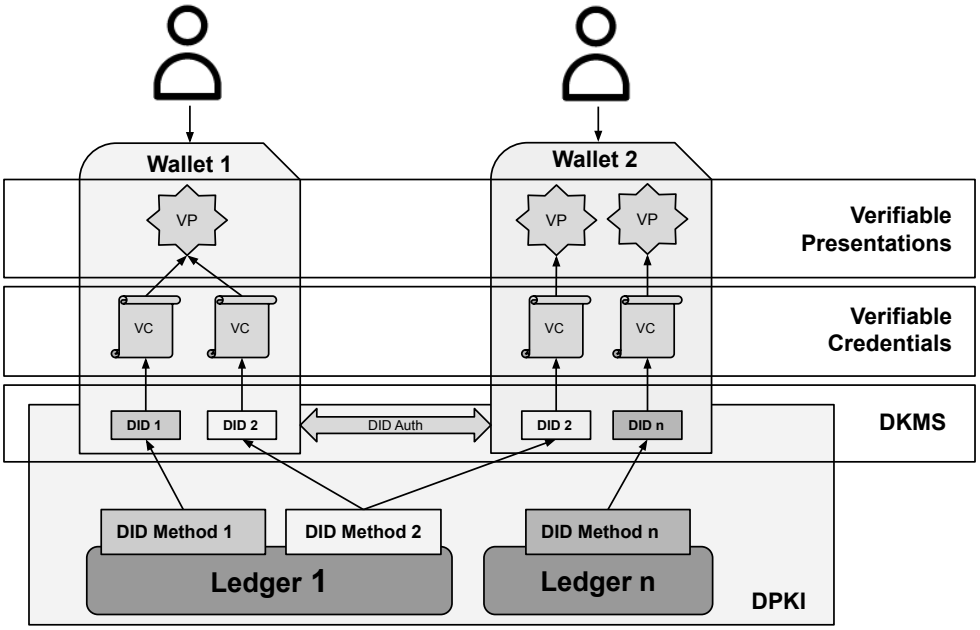

Figure 4.15: The components architecture of SSI. Adopted and extended from [72].

This components architecture comprises of four layers. The first one is DPKI in which different ledgers with different DID methods provide DIDs for an entity to make it publicly identifiable on this level. The DID standard is the basic and underlying standard at this level. By using a DID, every entity is uniquely identified in the SSI ecosystem. These identifiers allow users the creation of persistent connections with others to interact, such as issuing or requesting verifiable credentials while the DID is fully under the control of the entity. DIDs are defined in DID methods, which are ledger-based and define operations to create, read, update, and delete DIDs in a decentralized way without the need of a central authority. Information, such as public keys or service endpoints, are described in the corresponding DID document of a specific DID. The DID document may be dynamically resolved on the ledger, or stored in a publicly accessible off-chain storage.

The second layer is defined by the Decentralized Key Management System. A

DID is an identifier that has one or multiple public keys. Managing those keys in a standardized way is the task of DKMS. Besides managing keys, it also defines how the actors inside the system interact with each other or with the underlying distributed ledger. DKMS is crucial for the design of wallets, in which identity holders have their personal identity information and credentials stored. Wallets could also serve as endpoints for the DID Auth process, which describes the processes of authentication and proving ownership of a DID by using public and corresponding private keys. By proving ownership, connections are considered trusted and can be used for verifications and other processes that require a proof of ownership.

The third layer is defined by verifiable credentials, which are digital certifications such as a driver's license, diploma, or proof of residency. These credentials enable the attestation of further personal information about others or oneself. Every entity could issue, hold, or verify credentials.

The fourth and final layer is defined by verifiable presentations that can be created out of claims and verifiable credentials. They serve to present personal identity information in a trusted way to third parties, revealing only as much information as required, to preserve the identity owner's privacy.

Wallets are integrated into every level of the components architecture. They store the DIDs and may have interfaces to the ledgers for interaction. It is also possible to manage the keys of DIDs inside the wallet. Further, it stores the verifiable credentials and optimally allows creating verifiable presentations and share them with others. Finally, this information is all controlled by a real-world identity that is represented by the containing DIDs, verifiable credentials, or verifiable presentations.

The Web of Trust is not included in the components architecture, as it is included in every layer. The ledger serves as a root of trust in the first layer. In the second layer, trust comes mainly from DID Auth, while verification of verifiable credentials and -presentations have their own trust features implemented.

4.6 Evaluation of the SSI Concept

This chapter described in detail all the components that make Self-Sovereign Identity the new identity concept of the internet. Providing autonomy over an identity, enabling full control over personal information, and how personal information is shared and used are the main goals of SSI. Based on the implementation of standards and integration of other concepts, this goal is reachable and therefore a promising concept that could change the way people manage their online identities in a fundamental way.

Introducing decentralized identifiers as the central element is an essential step in order to create self-sovereignty for any entity in the SSI ecosystem. At least one public

and private key allows the user to cryptographically sign data and other information. This feature enables users to identify themselves, form connections and interact with others, issue, receive, and present verifiable credentials, as well as verifying others, and more. Hence, DIDs enable users to have full autonomy about their identifiers and therefore achieves the first goal of SSI.

Allowing users to have control over their data and how it is shared and used requires more components. Standardizing DIDs in its format among every DID method helps to create and establish a DPKI to find and distinguish one identity from another. Additionally, DPKI doesn't require central institutions that might have too much power and could control the ecosystem. DID documents play an important role because they contain data to authenticate or connect with a DID, such as the included public keys and the provided service endpoints. This information is required in order to successfully create interactions between different users.

Furthermore, standardization could increase interoperability between different DID methods. One important tool is the universal resolver that resolves DID documents of registered DID methods based on the given DID. This functionality lowers the barriers for cross-ledger interactions and therefore brings the DID methods closer together. One important part of interoperability is also DKMS as it describes how DIDs interact with each other and the ledger, as well as providing useful tools for key management, such as key recovery mechanisms.

Claims and verifiable credentials help to lift the SSI ecosystem to a higher level. Based on the functions of DIDs, like proving ownership and creating connections, claims and verifiable credentials help to create trusted attestations from one party about another or to verify claims of others. This enables identity owners to assert personal identity information to their identifier and to use them for interactions where trusted identity information is required.

SSI also brings an economic and usability benefit to actors inside the SSI ecosystem. For the users, it is easier to connect their online identity closer to the offline identity through the integration of verifiable credentials that allow them to have a digital version of a physical credential. It is possible to store the online identity in a digital wallet on one of their personal devices, making them easily accessible. Additionally, users can decide with whom they connect, which data is shared, and for how long information is presented to others. Hence, the second goal of SSI is achieved.

For companies, the main goal is to reduce costs and make processes more automated. Due to the verifiable and trusted nature of digital information in SSI, many manual steps to verify data can be fully automated. A student who applies at a university could get feedback about his application right after it was submitted instead of waiting for university employees to check the application manually. Furthermore, data silos can be highly reduced or eliminated since users just have to prove that they are eligible to

access the service by providing information that is pseudonymous or encrypted. Hence, it lowers the value for attackers to gain access to this information and companies don't have to store personal identity information at all. Last but not least, the concept of SSI conforms with the General Data Protection Regulation (GDPR), which is a huge benefit for SSI since it's currently difficult for companies to conform to this regulation of managing user data.

Even though the concept of Self-Sovereign Identity is quite well thought-out and brings its users to the center of actions, it is unclear whether users will adopt this new identity concept. First of all, it raises the question of whether people are responsible enough to have their online identity under their full control. People tend to forget or lose passwords and keys and it is debatable whether key recovery functionalities are good enough. In the worst case, people lose access to their DID and therefore lose all personal associated information with it. Manually reconnecting with other parties would then be the only solution, which should be prevented as it requires much time and effort.

Before people actually start utilizing SSI systems, it is unclear how they will get motivated to use a self-sovereign identity instead of conventional identity types that require usernames and passwords. In order to use the system, users have to decide which DID method will be their preferred one, which wallet suits them best, and how quickly they get used to the processes of this new type of identity. Transforming an online identity with usernames and passwords to a concept with predefined pseudonyms and public- as well as private keys could be a significant challenge for the majority of people.

Moreover, there is still a lot of regulatory uncertainty in terms of accepting digital signatures or verifiable proofs, which serve as fundamental trust components in SSI in combination with the underlying ledger. Those attestations would be more widely used and accepted if legal regulators would describe the terms of use.

The SSI ecosystem is not intended to act as a distinct parallel identity concept on the internet, but as a complementary one to existing systems. Therefore, it is important to define a policy to integrate existing systems. What if a centralized institution like Facebook wanted to provide DIDs for its users? Should Facebook still be in control of its users' identifiers, even if that violates the principle of self-sovereignty? If not, how can those users be onboarded?

Governance plays a very important role when it comes to the design of an SSI-compatible system as it defines policies within a system. It is important that governance is well thought-out, respects the principles of SSI, and doesn't leave too much power to central authorities inside the system. Having centralized components could lead to a single point of failure or unwanted user behavior which bears high risks.

Central tools, such as the universal resolver, have lots of power since using them makes it easier for actors to interact with each other. Unfortunately, it has to be

ensured that those tools don't have too much power so that monopolies can't be generated. Power should always be separated among some or multiple actors, e.g. having different models of universal resolvers with slightly different functionality. Similar to implementations of blockchain clients in the strongest systems, like Geth and Parity for Ethereum.

Another problem is that once data has been sent out, it's almost impossible to ensure that the receiver hasn't persisted it somewhere for later use or sale. One approach to solve this problem is using chameleon-hashes, as proposed by Jan Camenisch, David Derler et al. in their paper "Chameleon-Hashes with Ephemeral Trapdoors And Applications to Invisible Sanitizable Signatures" [73]. The chameleon hash is an undeniable commitment of the signer to the contents of the signed document but doesn't allow the receiver to disclose the contents of the signed information to any third party without the signer's consent.

To summarize, the SSI ecosystem is still in its beginning as it is constantly developing and only a small number of SSI-compatible systems already exist. Therefore, there still needs to be further research to solve problems, such as interoperability between the ledgers or between different SSI-compatible system, which governance model suits best, or if the web of trust is working as desired.

5 Introduction to DID Methods

The Self-Sovereign Identity ecosystem already contains some DID methods which implemented SSI differently from each other. Some of them are adapted on already existing ledgers, while others rely on ledgers which are explicitly designed for specific identity use cases. The purpose of this chapter is to provide a detailed analysis of already existing DID methods and their respective systems where the DIDs can be used. First, an overview of the existing DID methods in the SSI ecosystem will be given, followed by a well-founded selection of DID methods to be analyzed more precisely.

5.1 DID Methods Overview

The source of already existing DID methods is the DID Method Registry of the W3C, as of its status of August 2nd, 2019 [74]. This document contains all known DID methods and their associated DID method specifications. The W3C is closely connected to several identity projects and the development of standards, such as the DID specification [41]. It also defines the requirements to be registered as a DID method on the DID Method Registry. In order to register a DID method, the following requirements have to be satisfied:

1. There exists at least an experimental version of the DID method
2. The DID method has to be publicly available and has to have a DID method specification according to the DID Specification of W3C [41].
3. In order to link the DID method specification to the entry in the DID Method Registry, a pull request with the associated DID method specification has to be performed.

If there is an updated version of a DID method or its associated specification, deprecated entries will be taken off the list. Additionally, it is important to point out that the listed DID methods can still be under development and therefore may not be documented very well. Table 5.1 contains all DID methods that are suitable for scientific analysis and assessment, and considered for a more detailed analysis. The table has four columns, which represent the name of the DID method, its prefix, status, and the type of underlying ledger.

Table 5.1: DID methods listed in the DID Method Registry [74]. Accessed on August 2nd, 2019.

DID Method Name	Prefix	Status	DLT or Network
ABT DID Method	did:abt:	PROVISIONAL	Proprietary Ledger
Alastria DID Method	did:ala:	PROVISIONAL	Proprietary Ledger
bryk DID Method	did:bryk:	PROVISIONAL	Proprietary Ledger
ICON DID Method	did:icon:	PROVISIONAL	Proprietary Ledger
InfoWallet DID Method	did:iwt:	PROVISIONAL	Proprietary Ledger
JLINC Protocol DID Method	did:jlinc:	PROVISIONAL	Proprietary Ledger
Metadium DID Method	did:meta:	PROVISIONAL	Proprietary Ledger
Ocean Protocol DID Method	did:op:	PROVISIONAL	Proprietary Ledger
Ockam DID Method	did:ockam:	PROVISIONAL	Proprietary Ledger
Ontology DID Method	did:ont:	PROVISIONAL	Proprietary Ledger
peer DID Method	did:peer:	PROVISIONAL	Proprietary Ledger
lifeID DID Method	did:life:	PROVISIONAL	Proprietary Ledger
Sovrin DID Method	did:sov:	PROVISIONAL	Proprietary Ledger
TM DID Method	did:ttm:	PROVISIONAL	Proprietary Ledger
Veres One DID Method	did:v1:	PROVISIONAL	Proprietary Ledger
Vivvo DID Method	did:vvo:	PROVISIONAL	Proprietary Ledger
Weelink DID Method	did:wlk:	PROVISIONAL	Proprietary Ledger
BTCR DID Method	did:btcr:	PROVISIONAL	Bitcoin
ION DID Method	did:ion:	PROVISIONAL	Bitcoin
Blockstack DID Method	did:stack:	PROVISIONAL	Bitcoin-fork
erc725 DID Method	did:erc725:	PROVISIONAL	Ethereum
ETHR DID Method	did:ethr:	PROVISIONAL	Ethereum
N/A	did:dom:	PROVISIONAL	Ethereum
Jolocom DID Method	did:jolo:	PROVISIONAL	Ethereum
SelfKey DID Method	did:selfkey:	PROVISIONAL	Ethereum
Emtrust DID Method	did:emtrust:	PROVISIONAL	Ethereum-Fork
TangleID DID Method	did:tangle:	PROVISIONAL	Other Public Ledgers
IPID DID method	did:ipid:	PROVISIONAL	Other Public Ledgers

Out of the original 32 entries of the DID Method Registry, only 28 of them are listed in Table 5.1. Three of the four not included entries define a DID specification, which is a format for DID method specifications. One of them is the official DID specification that describes the standard format of a DID inside the SSI ecosystem. All DID method specifications listed in the table build upon this standard. Therefore, analyzing other DID specifications are not relevant in this thesis. The fourth not listed entry defines the deprecated uPort DID method (did:uport:) which was succeeded by the ETHR DID method that is listed in Table 5.1. Therefore, the uPort DID method won't be discussed in further detail.

The table also shows that all DID methods are still in a "provisional" status. This means that changes can still be made to the code. However, some DID methods might be further developed than others. Since this ecosystem develops very quickly, some information might already be deprecated at the time when this thesis is being submitted or published.

All listed DID methods in Table 5.1 can be categorized based on their underlying ledger. Those categories are listed and explained in Table 5.2.

Table 5.2: Categories of underlying ledgers and their respective explanations.

Ledger Category	**Explanation**
Bitcoin-based	The DID methods rely on the public Bitcoin blockchain.
Bitcoin-forked	The DID methods rely on a fork of the public Bitcoin blockchain.
Ethereum-based	The DID methods rely on the public Ethereum blockchain.
Ethereum-forked	The DID methods rely on a fork of the public Ethereum blockchain.
Proprietary Ledgers	The DID methods rely on proprietary ledgers which were implemented for SSI.
Other public ledgers	The DID methods rely on already existing, publicly available, and DLT-like technologies.

Figure 5.1 shows the DID methods visualized in a pie chart based on the predefined ledger categories.

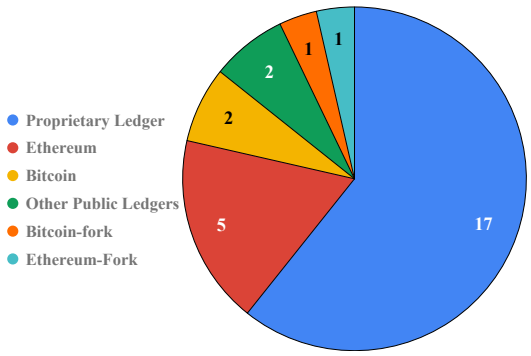

Figure 5.1: Amount of DID methods for each ledger category illustrated in a pie chart.

It is fairly apparent that DID methods based on proprietary ledgers are in the majority with a total of 17 of 28, or 60,7% of all DID ledgers. The next largest portion is Ethereum with a total of 5 out of 28, or 17,9%. Bitcoin-based ledgers and DID methods based on other public ledgers follow with each a total of 2 out of 28, or 7,2% of all ledgers. Finally, Bitcoin-forked and Ethereum-forked each have a total of 1 out of 28, or 3,6% of all DID ledgers.

More than 60% of the DID methods are built on proprietary ledgers. It is worth mentioning that already existing public blockchains weren't initially designed to establish Self-Sovereign Identity. Their main purpose from the beginning was to transfer value from a sender to a receiver, as well as to provide and make use of smart contracts in the case of Ethereum. One of the main reasons to develop an SSI component inside already existing blockchains is their huge user base. Those users could make use of their already-existing public- and private key pairs to create a DID and participate in the SSI ecosystem.

5.2 Selected DID Methods

For this thesis, only a selected amount of DID methods will be analyzed and evaluated in more detail. The DID methods and their systems are selected based on the general progress in development, available documentation, the relevance for potential users, like in Bitcoin, interesting use cases based on the DID, such as in Blockstack, its multiple different ways of adoption on a single ledger, as on Ethereum, or the different ways of creating DIDs on proprietary ledgers such as Veres One and Sovrin. This chapter presents DID methods that play a significant role in the SSI ecosystem and are representative of others not included. These DID methods will be further analyzed and assessed in chapter 6.

5.2.1 Bitcoin-based

The Bitcoin blockchain is the oldest existing blockchain and has the highest user base. Therefore, it is considered as the most trustful in terms of future use and persistence, as well as it is the easiest to provide services to a high amount of potential users. Since it is mainly designed to transfer value, it barely provides ways to implement additional logic like anchoring data. However, the main action of systems adapting to the Bitcoin blockchain could be processed without blockchain interaction by using a second-level protocol, which would make the system faster and cheaper since transaction fees could be set to a minimal value by default. Another advantage of the Bitcoin blockchain is that everyone can set up and run a node, or use block explorers to check and validate transactions due to its permissionless and public nature.

Bitcoin Reference

Bitcoin Reference (did:btcr:) is a DID method based on the public Bitcoin blockchain and was designed by Christopher Allen, Kim Hamilton Duffy, Ryan Grant, and Dan Pape [75]. It provides basic DID functionalities, such as the creation of DID documents and regular authentication mechanisms by using the DID Auth authentication concept. BTCR focuses on anonymous or pseudo-anonymous identities and minimal personal information disclosure. A DID is created and updated by sending a new transaction on the Bitcoin blockchain. Updates to a DID, which are affecting the DID document, are therefore publicly visible and traceable, whereas information about the location of the DID document could be referenced in the transaction OP_RETURN data field.

In contrast to other DID methods, the method-specific identifier doesn't represent an identity but a transaction inside the Bitcoin blockchain. This is achieved by TxRef technology, as described in the Bitcoin Improvement Proposal (BIP) 136 [76]. TxRef refers to transactions inside a specific blockchain, e.g. mainnet or testnet, that must already be confirmed. It encodes the given chain, block height and transaction index where the transaction is located. By doing so, the method-specific identifier is created. An example of a DID inside BTCR looks is shown in Figure 5.2.

did:btcr:xz35-jznz-q6mr-7q6

Figure 5.2: An example of a BTCR DID

Creating and updating a DID requires a transaction on the Bitcoin blockchain. Figure 5.3 illustrates the process of creating a DID and asserting information to the DID document.

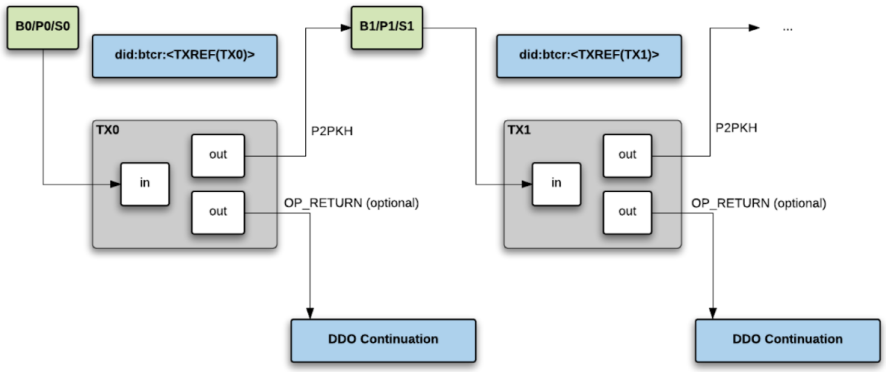

Figure 5.3: The process of creating a BTCR DID. Taken from [75].

In order to create a DID and the associated DID document, two key sets have to be

created - one for sending (B0/P0/S0) and one for receiving (B1/P1/S1), where "Bi" represents the Bitcoin address, "Pi" the public key, and "Si" the signing key or private key. The sender B0 creates a transaction to the receiver B1, where the OP_RETURN data field in a second output contains the link to the DID document. Signing this transaction reveals the public key of the sender. After signing and receiving confirmation of the transaction, a TxRef encoded value represents the method-specific identifier and the DID is created. If the DID should be updated, this process has to be repeated with Bitcoin address B2 with the latest used Bitcoin address as the sender - in this case, B1.

Transactions to create a DID could be performed with or without the OP_RETURN data field. If no OP_RETURN data field is inside the transaction, the DID document is generated with default values of the transaction. If an OP_RETURN data field exists, it must have an HTTP URL included that refers to the DID document [75]. This could be achieved by storing the DID document on a publicly available storage, such as IPFS. Otherwise, it could be possible to store information relevant to merge into existing DID documents by the BTCR resolver.

In order to read information about a DID, the DID document has to be resolved, as described in the BTCR Resolver paper published by Kim Hamilton Duffy, Christopher Allen, et al. [43]. Given the DID, the resolver checks whether the transaction has an output value for newer transactions. If there is an output value, then the resolver has to go further along with the transactions in the output value. If there is no further transaction documented, the latest DID document is the most current one and is being used for interactions between actors inside the Bitcoin blockchain. A DID is considered as revoked or deleted if a transaction is signed without an OP_RETURN data field.

DID construction can be performed with or without the OP_RETURN data field, while the following transactions must contain one. The data field must contain an HTTP URL pointing to a DID document that specifies the keys and service endpoints. The DID resolver must be able to return the referenced DID document.

With the use of the transaction output index, the DID resolver can check whether a DID was being updated or revoked. The resolver checks the OP_RETURN value to find the latest DID document and resolves it. If there is no OP_RETURN value, the resolver considers the DID to be deleted or revoked.

The BTCR method is freely accessible to every Bitcoin address holders. The logic of BTCR is implemented only on the blockchain itself and not on a complementary system. Since this is the only well-known DID method that operates this way, it is important to take a closer look at this DID method.

Blockstack

The Blockstack system comes with a DID method (did:stack:) based on a fork of the public Bitcoin blockchain, as described in the Blockstack DID method specification [77]. As of October 2018, Blockstack relies on the "Stacks Blockchain v1", which resulted from a Bitcoin hard fork at block 547921 [78]. The Stacks blockchain is customized for the Blockstack system to allow developers to build decentralized and scalable apps that accept a Blockstack identity.

The main purpose of Blockstack is to enable a domain name system to access provided decentralized applications (dApps) in Blockstack's dApp marketplace. In order to access the platform, the user has to create an identity with a username to create a namespace. The username is then linked to the current public key of the user and points to the user's data storage. Therefore, the username's state can only be utilized or modified by the holder of the corresponding private key to the user's public key. The usernames are created in a decentralized naming layer inside the Blockstack system, called the Blockstack Naming System (BNS) whereas the creation and changes to the ID are documented on the blockchain.

Blockstack implements the standard Bitcoin consensus mechanism to the Stacks blockchain, which is the Proof-of-Work consensus mechanism, on the Bitcoin blockchain. In combination with PoW, Blockstack adds a Proof-of-Burn [79] (PoW) where miners "burn" their cryptocurrency to show their intention to participate in the mining process [80]. Since the hard fork, the newly implemented "Stack" token can also be used for the Proof-of-Burn mechanism, instead of Bitcoin. This combination of two consensus mechanisms is called the "Tunable Proof" mechanism and is used for leader election to validate blocks. By doing so, Blockstack wants to securely bootstrap its new blockchain and slowly transform to the native PoW mechanism. Among other things, this structure has the benefit to create higher throughput, since it is not coupled with the transaction throughput of the underlying blockchain. Additionally, anyone can become a leader since the burning of cryptocurrencies doesn't require mining hardware. This loose coupling with the underlying chain makes it easier to switch the underlying blockchain.

Blockstack provides the option to have on-chain DIDs as well as off-chain DIDs, also called on-chain names or off-chain names respectively, as described in the Blockstack DID method [77]. The following paragraphs are derived from this document. On-chain DIDs are written directly to the underlying blockchain which requires two transactions - `NAME_PREORDER`, which creates a hash based on the username, and a `NAME_REGISTRATION` transaction, which registers the name in the naming system [81]. If the `NAME_REGISTRATION` transaction was successful, the name is assigned to an indicated on-chain address generated by the BNS. The method-specific identifier is created out of the name's hash and the index parameter. The addresses are

encoded with different version bytes, based on the name type, on-chain or off-chain, or mainnet or testnet. An example of a Blockstack DID looks is shown in Figure 5.4.

did:stack:v0:16EMaNw3pkn3v6f2BgnSSs53zAKH4Q8YJg-0

Figure 5.4: An example of a Blockstack DID

The DID that was created for a specific name can be referenced by the name owner's address and the number of names associated with the address at the time of initiating the DID. The number of names is shown in the last number of the DID, also called index parameter. If three names were asserted to the address during DID creation, the index parameter would be a 2, since counting starts with 0.

Besides on-chain DIDs there are also off-chain names, or subdomains which need on-chain DIDs to get their transactions processed on the underlying ledger. An off-chain name is structured similarly to an on-chain name and is owned by an address with a public and private key pair. The main difference is that off-chain names don't have direct access to the ledger. In order to store the off-chain name on the ledger, identity owners have to submit a JSON in an HTTP POST with their operations, like the creation of a DID, that has to be processed by on-chain names. Transactions of off-chain names are instantiated and stored outside the Blockstack ledger in a Blockstack-specific Peer-to-Peer network, called "Atlas". In order to provide the same security properties as for on-chain DIDs, transactions from off-chain names are encoded and collected in batches which will be hashed and then stored on the underlying ledger. Additionally, on-chain names propagate signed transactions from the off-chain names. Even though on-chain DIDs have to instantiate the off-chain names to the ledger, they don't have control over off-chain names and can't perform updates on the name initiated by themselves, since the off-chain name's private key is never in the hand of the on-chain name.

By definition, each name in Blockstack has its own URL. This URL has to point to a well-formed DID document, which contains information on the Blockstack name, such as account information, APIs, or public keys and in the Blockstack system. The DID document must be in the form of a signed JSON Web Token (JWT), so the resolvers can read and validate it properly. This signature is important in order to authenticate the DID document as it reveals the DID owner's address.

Resolving a Blockstack DID requires a node that has full access to all transactions on the Stacks blockchain. Additionally, the node has to store the "Atlas" network that holds the registered off-chain nodes. By performing a GET request with a specific Blockstack name, the resolver at the node returns the name's method-specific identifier. Based on this response, a JSON object is returned with a public key field from the hash

to the DID's address as well as a document field containing the DID document.

In order to update a Blockstack DID, the underlying name has to be transferred to a new address with a NAME_TRANSFER transaction which results in an update of the DID's public key. Off-chain names have to send the new DID via the on-chain name that initially instantiated the off-chain DID. In this case, the on-chain name processes the NAME_TRANSFER transaction on behalf of the off-chain name. To update the DID document storage location, the on-chain name has to send a NAME_UPDATE transaction to the underlying blockchain. Off-chain names have to propagate this transaction to any on-chain name.

Since DID documents are stored outside the blockchain and only contain a link to the DID document, all previous DID documents become invalid once the public key of the DID document changes.

Deleting a DID works by deleting the name in the naming system of Blockstack. This is done by executing a NAME_REVOKE transaction for a specific name. If the name doesn't exist in the naming system anymore, this means that the corresponding DID was effectively deleted. Off-chain names have to construct and broadcast a new transaction with a modified address that look false, like an address only consisting of 1s. By doing so, the address isn't resolvable in "Atlas" anymore and also effectively deletes the DID.

By forking the Bitcoin blockchain, Blockstack implemented additional functionality to the Stacks blockchain and connected it with its integrated naming system and other databases like "Atlas". In case of a successful attack on the Stacks blockchain, the entire Blockstack system can be transferred to another ledger and the CRUD operations have to be updated based on the underlying ledger.

5.2.2 Ethereum-based

The Ethereum blockchain is a public, permissionless blockchain like Bitcoin. It is also meant to transfer value in the form of Ether from one address to another and further allows to implement processing logic by utilizing smart contracts for decentralized applications (dApps). Smart contracts are represented with an address and can, therefore, be seen as a valid identity in Ethereum. Smart contracts could contain their own rules for managing ownership, which makes it attractive to implement SSI-compatible solutions. Like the Bitcoin blockchain, the Ethereum blockchain has a huge amount of active users that could benefit from an implemented identity concept.

uPort

One of the biggest projects to enable decentralized identities in the SSI ecosystem is the uPort project with the implemented ETHR DID method [82] (did:ethr:). DIDs can be managed by invoking functions on the "EthereumDIDRegistry" smart contract [83], further referenced as ETHR DID Registry.

Decentralized identifiers implemented using uPort libraries rely on the ETHR DID method [82], which allows any Ethereum smart contract or other addresses managed by a person to become a valid identity, as described in the ERC1056 documentation [84], on which the ETHR DID Registry smart contract is derived. Off-chain addresses don't have to be registered on the blockchain, which means that there may be no transaction fees for the creation of off-chain addresses, as registration could be done by others by invoking a function on the smart contract that addresses the owner's address.

Every DID allows only one address that represents the owner controlling the identity. The owner of a DID is by default the address who invoked the ETHR DID Registry smart contract but ownership can be transferred to someone else. Furthermore, the smart contract provides functions to authorize a delegate to perform operations on behalf of the owner.

In order to create a decentralized identifier in uPort by using the ETHR DID method, a regular Ethereum address, which can be generated off-chain, has to be owned by an identity. If the address is the sender of a transaction that invokes a function, or an address is used as the input value, an ETHR DID is created. The combination of the ETHR DID prefix and the Ethereum address is the method-specific identifier of the ETHR DID method. An ETHR DID is shown in Figure 5.5.

did:ethr:0xE6Fe788d8ca214A080b0f6aC7F48480b2AEfa9a6

Figure 5.5: An example of an ETHR DID

Figure 5.6 shows how the ETHR DID Registry smart contract is involved in the identity process.

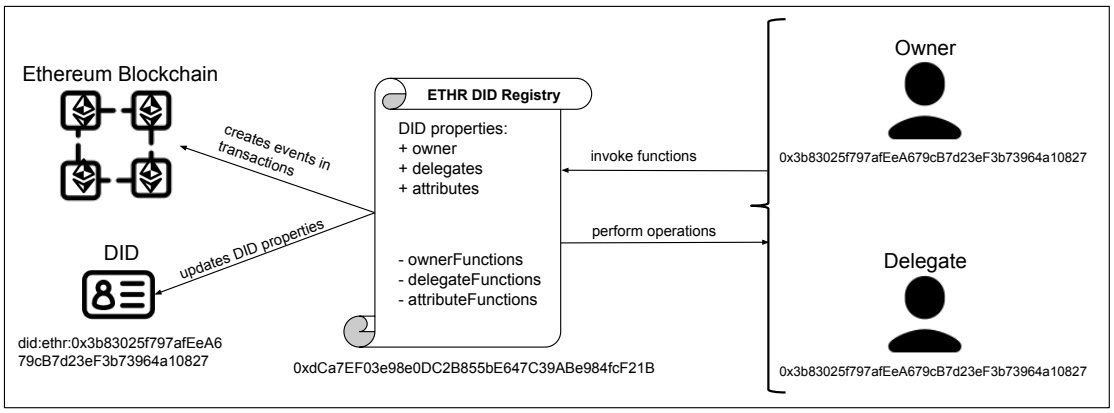

Figure 5.6: An interaction model of the ETHR DID Registry smart contract

Firstly, every DID has owner-, delegates-, and attribute properties to describe the representing identity. These properties are managed by set-, get-, and revoke functions, defined in the ETHR DID Registry. The DID is created at the moment an address invokes a set-function for the first time. Further changes to the DID can be made by invoking provided functions.

Additionally, delegated addresses can perform actions on behalf of an identity. Therefore, the smart contract contains further functions to set the duration of the validity of a delegate, as well as looking up, adding, and revoking a delegate on-chain and off-chain. For this case, functionality to read, set, and revoke attributes is provided. All functions regarding delegates and attributes, as well as change of the address owner are eventually documented as `DIDDelegateChanged`, `DIDAttributeChanged`, or `DIDOwnerChanged` events in transactions on the blockchain [85]. The functions could be manually invoked over the smart contract interface in Etherscan [86].

Only owners or registered delegates can invoke change-functions. Every time a change-function is invoked, a transaction is created that contains an event that documents and describes the performed change. These events serve as input for the DID resolver to create the DID document.

Offline-functionality is enabled by external management by using delegates that are authorized by the owner. The owner could have key rotation and key specification done by delegates on the owner's behalf by delegating singing to externally, managed key pairs. This could also result in delegating transaction fees to the delegate.

To create a DID document for an address, the resolver has to look up the address owner, followed by iterating over transactions in the Ethereum blockchain by the DID's address to find the events which document changes to the DID.

Updating a DID document results from invoking the presented functions that result in previously mentioned events. The events are linked through a `previousChange` attribute that refers to the last block where it was changed. During resolving of the DID, the resolver checks the events and constructs the DID document based on the documented changes. The resolver updates the DID document once one of these events is documented on the blockchain.

If an ERC1056 needs to be revoked or deleted, there are two ways of achieving it. If no interactions have happened on the smart contract so far, only the private key has to be deleted from the controlling storage device. If the smart contract was utilized, the value of its owner has to be set to `0x0` indicating the smart contract is without an owner. By invoking the revoke functions previously mentioned, delegates and attributes can be revoked, which will trigger an event that the resolver reads and adapts the changes to the DID document.

uPort is not just a single identity solution, it further provides open-source tools and protocols in a large-scale development library. These libraries allow other developers to implement their own DID-compatible applications based on uPort artifacts. The developed identity- and messaging protocols create an interoperable identity layer for the internet. A key feature of uPort is the possibility to integrate addresses that are already in use by the user by importing the public and private key pair to the wallet. The goal of uPort is to provide a full set of decentralized identity tools and protocols on the Ethereum blockchain, where the user is in full control of its identifiers and also enables credential management between actors.

In detail, uPort provides libraries that define the interaction between two parties on the client-side, as well as on the transport layer. First, there is "uPort Connect" [87], which defines interactions with an identity over a web browser using predefined workflows to request and send data from and to a wallet. Additionally, it defines processes how to interact with the Ethereum blockchain. The second library uPort provides is "uPort transports" [88]. This library defines communication logic by sending messages between clients in the uPort system. Furthermore, the "uPort credentials" [89] library defines how credentials and claims are being requested, issued, and how information can be disclosed.

uPort also makes use of already existing formats, such as JSON Web Tokens (JWT) [52] for signing and verifying signatures for any DID compliant identity. Moreover, uPort uses the ETHR DID resolver to resolve identities from Ethereum addresses according to the ERC1056 specification [84]. The ETHR DID registry [83] is a smart contract that allows on-chain and off-chain resolving and management of DIDs, as well as the JSON RPC [90], to interact with the blockchain. These libraries and the smart contract already do most of the work for a developer, which makes it easy to implement and gain experience.

To summarize, uPort uses the ETHR DID Registry smart contract to manage DIDs. Users with addresses that aren't registered on the blockchain yet could still become a valid DID if others use their address as input value by invoking the smart contract. Further, delegation and revoking of permissions is possible with this contract. With the combination of the provided libraries, uPort is a very interesting approach to provide Self-Sovereign Identity to the Ethereum blockchain.

Before the uPort project used the ETHR DID method, they had another implementation to interact with the Ethereum blockchain, described in the uPort DID method that is now deprecated. The main concept was similar to the current one as the uPort DID utilized a central proxy smart contract to perform actions. However, the ETHR DID method has improved interoperability and is more scalable and privacy-preserving than the initial uPort proxy smart contract [91].

ERC725

In 2017, Fabian Vogelstetter proposed a new standard for managing on-chain identity on the Ethereum blockchain by introducing a proxy smart contract concept, known as ERC725 [92] (did:erc725:). This concept enables standard key management functions, such as an owner proving control of ownership of a DID as described in the ERC725 DID method specification [93], from which the following paragraphs are derived. Additionally, ERC725 is closely connected to ERC735 which is a concept to attest claims to ERC725 DIDs.

The ERC725 identity concept describes a smart contract as a valid identity that can hold keys in order to sign actions, e.g. transactions, as well as claims either attested from third parties or self-attested. Additionally, it enables proxy functionality as it provides methods to execute logic for identity management that the owner can invoke remotely. The smart contract is represented by a standard Ethereum address.

Another functionality of an ERC725 smart contract is the ability to add multiple keys and other smart contracts to control the identity for authentication or signing of claims. Additionally, if a contract is managed by multiple parties, multiple keys could serve to define access logic and permissions for each key individually [94].

Besides the ERC725 concept for identity smart contracts, another complementary proposed standard is ERC735 [95]. This concept provides functionality to add, remove and hold claims made about an ERC725 identity. These claims could be attested by third parties or by the ERC725 identity itself. The information contained in claims could be exchanged among other parties to verify or authenticate. Third-party verifiers could then decide, whether they can trust the issued claim based on the trustworthiness of the issuer.

The claim has to be requested from issuers who sign a message containing the

identity's address, the claim topic, and optionally some metadata. The issuing identity could be the same as the requesting identity, which would result in a self-attested claim. The claims are then stored inside the identity's ERC725 smart contract.

One popular project that adapts the ERC725 concept is "Origin Protocol", who's goal is to enable a Peer-to-Peer marketplace on the blockchain [96]. In this case, claims must be verified before a smart contract will be executed.

A DID in the ERC725 DID method will be identified by the address of the ERC725 smart contract with the prefix "did:erc725:", as described in the ERC725 DID method specification [93]. Optionally, the network might be between the method-specific identifier and the prefix. If there is no network included, the default network is the Ethereum mainnet. An ERC725 DID is shown in Figure 5.7.

did:erc725:2F2B37C890884242Cb9B0FE5614fA2221B79901E

Figure 5.7: An example of an ERC725 DID

Figure 5.8 presents how the ERC725 smart contract is involved in identity management.

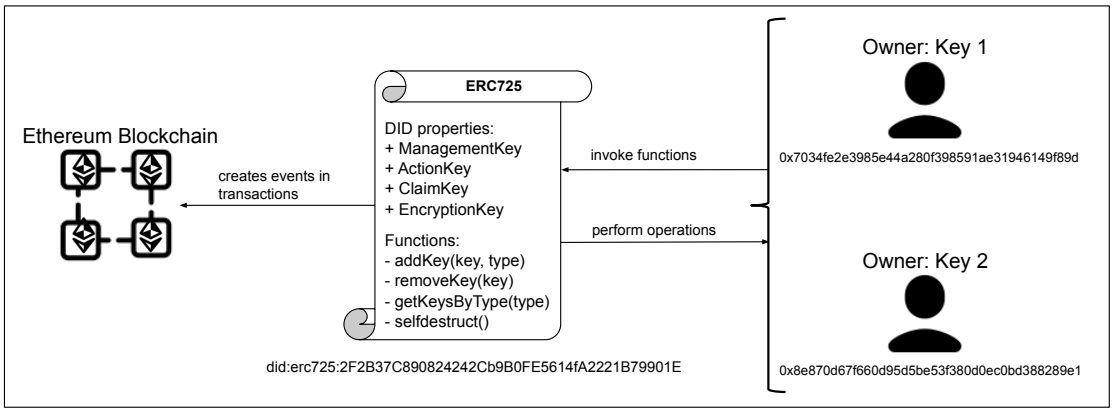

Figure 5.8: An interaction model of an ERC725 smart contract

Firstly, the ERC725 DID method extends the `@context` notation in the DID document with terms for supported key types `MANAGEMENT`, `ACTION`, `CLAIM`, and `ENCRYPTION`. These are used for authorization and delegation. This is being done by invoking provided functions in the smart contract. Depending on the authorizations, other addresses can invoke these functions and perform operations they are authorized for.

The ERC725 smart contract is created by deployment on the Ethereum blockchain. The resulting DID identifies the smart contract.
To resolve an ERC725 smart contract and to create a DID document, `getKeysByType` has to be invoked for each supported key type.
For each key type, the public key has to be added to the respective key type in the contract. As an example, for adding a management key, the public key is added to the `ERC725ManagementKey` part of the DID document. The same procedure is similar for `ACTION` -, `CLAIM` -, and `Encryption` keys.
In order to update the DID document, the `addKey` or `removeKey` function on the smart contract has to be invoked.
By invoking the `selfdestruct()` function, the code, and storage of the smart contract will be removed from the Ethereum state, which marks the DID as revoked.

In contrast to uPort and the ERC1056 smart contract, an ERC725 is the DID, while ERC1056 provides the functionality to change and manage properties of a DID. ERC725 allows others to perform operations on their behalf. Furthermore, ERC725 solves this by authorization of keys in the DID document instead of invoking functions implemented to the smart contract. Additionally, ERC725 is closely connected to the ERC735 concept, that enables managing and making claims associated with an identity.

SelfKey

Another Ethereum-based DID method is Selfkey (did:selfkey:), managed by the SelfKey Foundation. It is implemented in the SelfKey-system which provides the possibility to manage a user's identifiers in a wallet using a SelfKey DID that gives the user access to a distributed marketplace for financial services. SelfKey published a whitepaper [97] where the concept and processes of SelfKey are described. The attributes of an identity are represented as claims, whereas SelfKey provides integration of pictures or scans of official documents where the information could be then parsed out and eventually be used to selectively disclose information to verifiers. These claims have to be attested by a third party verifier within a KYC process. Additionally, the SelfKey wallet provides the possibility to manage a stack of crypto in a portfolio, e.g. the SelfKey-specific "KEY" coin, based on the ERC20 token, that can be used for the financial offers provided in the marketplace.

As described in the SelfKey DID method specification [98], SelfKey provides the "DIDLedger" [99] smart contract where operations, such as to create and update a DID, can be performed. The smart contract allows determining a controller, which is by default the creator of the DID. Only the controller of a specific DID can invoke functions to a respective DID on the "DIDLedger" smart contract. Figure 5.9 shows a

SelfKey DID.

did:selfkey:0x58a9d3f14916f6ba75fa57032c4627497b62fa0105f60142b03c3ebab74b7e15

Figure 5.9: An example of a SelfKey DID

The DIDLedger smart contract is structured as illustrated in Figure 5.10. While the other fields are self-explaining, it is important to point out that the metadata attribute is used for the resolver.

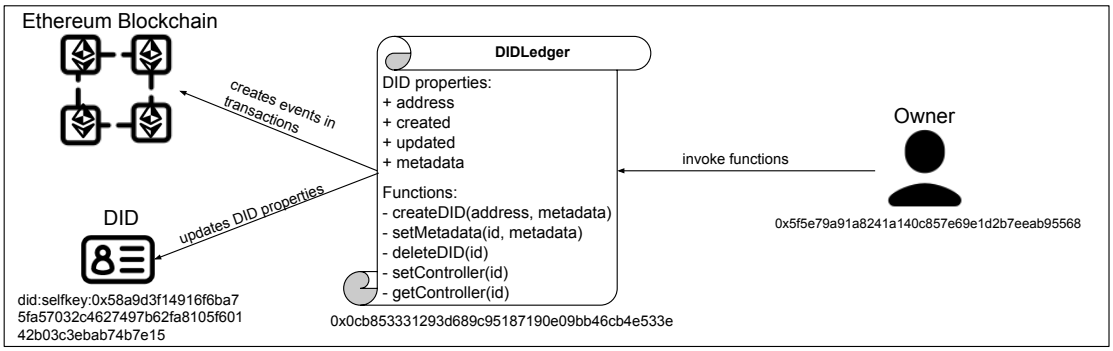

Figure 5.10: An interaction model of SelfKey's "DIDLedger" smart contract

To create a DID, a transaction to invoke the function `createDID` has to be sent to the "DIDLedger" smart contract, where the sender of the address becomes the controller of the DID. Any Ethereum address can invoke this function. In order to resolve a SelfKey DID, transactions on the "DIDLedger" smart contract with a given sender address and resulting events, have to be observed on the Ethereum blockchain to dynamically construct the DID document, based on the events. Currently, no service endpoints are supported in SelfKey.
Updates to the DID are processed by invoking either the `setMetadata` or the `setController` function on the "DIDLedger" smart contract to either change the metadata or to change the controller of the DID. The ledger contract also provides the `deleteDID` -function to permanently delete the DID. These functions can only be invoked by the controller of the DID.

Further, it is possible to assert claims to a DID. The provided claim functionality relies on the ERC780 claim management concept [100], where claims can be issued about Ethereum identities.

SelfKey created this DID method because the identities of its customers used in SelfKey can be easily managed using the "DIDLedger" smart contract. Managing the DIDs over one smart contract makes it easy for SelfKey to let the users manage their

DIDs. Furthermore, self-attested claims can be made about an identity but they have to be verified by a third-party verifier in order to use financial services.

5.2.3 Proprietary Ledgers

This chapter contains DID methods that are based on a proprietary ledger. The two DID methods presented in this chapter rely on ledgers that were specially designed to allow Self-Sovereign Identity management. With this characteristic, the ledgers were designed to support any identity-related feature, making it easier and more efficient to manage the identities.

Sovrin

While other DID methods try to enable Self-Sovereign Identity on already existing public blockchains, the Sovrin DID method (did:sov:) tries to achieve this goal with a purpose-built public-permissioned blockchain solution. This entirely new setup of a system requires the Sovrin Foundation, among other things, to design the system from scratch, as well as defining the system governance, trust frameworks, and a privacy policy. In order to accomplish all these tasks, the Sovrin Foundation was created by gathering competences to try to get the best possible solution.

The main goal of Sovrin [101] is to give users full control and autonomy about their identity, which should be globally usable among the SSI ecosystem. Based on this identifier, the user is able to issue, receive, and present verifiable credentials according to the corresponding standard by the Verifiable Credentials Working Group [54]. Additionally, "privacy by design", which means that privacy-preserving technology is integrated during the design of the system, prevents data leaks and protects the privacy of each actor inside the system in the best way possible. This is achieved by introducing pairwise-pseudonymous DIDs for every connection of a user, minimal disclosure using Zero-knowledge proofs at presenting personal identity information to verifiers, and keeping such private identity information off the ledger.
Due to its permissioned nature, the system requires a governance framework [5] that defines business, legal, and technical policies inside the system and a trust assurance framework [6] that defines how trust is established and how the system can be trusted by all stakeholders.

The Sovrin ledger is run and maintained by Sovrin Stewards who run validator nodes and can read and write from and on the ledger. Stewards are trusted partners of the Sovrin Board of Trustees who run a node and have the right to write information to the ledger, according to the Sovrin Governance Framework [5] and are therefore considered trust anchors.

The Sovrin protocol entirely relies on open standards and open-source code, which is all defined inside the Hyperledger Indy project. This project aims to provide best practices for a decentralized online identity [102] and was collaboratively developed for the Sovrin project but can be adapted by other identity systems as well. The underlying consensus algorithm is called "Plenum" and is based on Redundant-BFT which tolerates faulty nodes during consensus agreement.

The following paragraphs are derived from the Sovrin DID method specification [103]. A Sovrin DID has the prefix "did:sov:". A valid Sovrin DID is shown in Figure 5.11.

did:sov:CYQLsccvwhMTowprMjGjQ6

Figure 5.11: An example of a Sovrin DID

To create a DID, the user has to provide an unused DID and a corresponding document that will be stored on the ledger. Due to its permissioned nature, getting access requires verification. For this reason, a trust anchor party serves as a trusted entity and has the right to verify and sign transactions of users, effectively granting access to the system. After the DID document is signed by the trust anchor, a transaction containing the DID document has to be submitted to the ledger. If the transaction is well-formed, contains a signature of a trust anchor, and the DID is still unregistered, the transaction succeeds and the DID is registered in the Sovrin ledger.

A DID document in Sovrin is directly stored on the ledger, therefore a simple transaction has to be processed, returning the DID document of a given DID. This request can be sent by anyone. Updating a DID requires the owner, or controller, to send an NYM transaction containing the new DID document. Only actors owning or controlling a key in the authentication part of the DID Document can perform updates. To revoke or delete a key, the verification key of the DID document has to be set to a null value, which permanently disables the user to operate in the network because authentication is impossible without the key. Besides deletion, there is also the possibility to revoke the authentication key, which prevents the DID from being further used but keeps all the records of previous transactions before the revocation.

To summarize, Sovrin utilizes the Sovrin ledger to enable users to keep control over their identifiers and associated data. The system is backed by an extended governance model and a trust assurance framework that establishes trust in the system, its actors, and processes. Sovrin also has a well-designed credential ecosystem that allows organizations to issue credentials to identity owners that could merge their online-identity closely with their real-world identity by digitally gathering the credentials. Presentations of these credentials could be disclosed by generating ZKPs to preserve the user's privacy. This enhanced functionality results from the implementation of the new,

and open-sourced Hyperledger Indy ledger, where these functionalities are initially implemented. Therefore, this DID method is very well suited for further analysis in this thesis.

Veres One

Another project that doesn't rely on an already existing public ledger but implemented its own is Veres One (did:v1:). The following paragraphs are derived from the Veres One project summary [104]. Veres One is a project to enable Self-Sovereign Identity that was founded by Digital Bazaar [105] and operated under the Veres One organization. The goal of this project is to provide an identity system that is accessible and usable by everyone and enables creation and administration of DIDs (did:v1:) in a self-sovereign manner. Unlike Sovrin, Veres one runs a public, permissionless system, allowing everyone to participate as an equal entity. In the future, Veres one should allow users to issue, hold, and present verifiable credentials that conform with the standard of the Verifiable Credentials Working Group at W3C [54]. Open standards and open-source code play an important role to make the identity system as interoperable, trustful, and highly functional as possible. At the moment this thesis is written, Veres One is not released and only runs a testnet where DIDs can be created. The following paragraphs are taken from Veres One documentation about its concept.

Respecting users' privacy is an essential aspect of the Veres One project, as only DIDs are stored on the ledger, and selective and minimal disclosure are supported. Due to its public and permissionless characteristic, the ledger of Veres One is open for everyone to run a node and read the information included. The containing "Continuity" consensus algorithm [106] should enable a Byzantine-Fault tolerant system [107] and a leaderless elector collaboration, where leaders are dynamically determined which events should be included in the network. Dynamic, leaderless election prevents a permanent single point of failure but also increases the chances that one leader might be more secure than others. To include an event, 2/3 of all leaders have to agree on [108].

The governance model of Veres One is mission-driven, focusing on making the project as successful as possible. Due to the fact that everyone can run a node, an incentive scheme has to be deployed to reward node operators and other stakeholders. Veres One does not intend to create an own token, but charges fees for creating and updating DIDs if an accelerator node is used to process these operations. These fees are then used to spend node rewards and to pay maintainers of the system.

There are multiple parties involved in the system, with the head of the project being the Board of Governors. Their task is to ensure the proper execution of the governance and the collection and distribution of funds among the system. Maintainers are in charge of maintaining the software, such as fixing bugs or implementing new features.

Nodes are responsible for providing computational and storage resources that together build the node network. Users are described as entities which can create, use, and update their identifiers. These operations can be done by so-called accelerators, which charge a small fee for processing these operations. These fees are then collected and distributed among nodes and maintainers [109].

As a result, these concepts should enable Veres One to be fast, cost-effective, and more privacy-preserving than already existing blockchain-based identity systems. Unfortunately, Veres One is still in development and so far has only provided a very limited testnet.

The following paragraphs are derived from the Veres One DID method specification [45], which describes two forms of DIDs inside the system. The first is a cryptonym-based identifier, which is a SHA256 hash of a public key. These identifiers don't have to be registered on the ledger and could be used as unregistered pairwise-pseudonymous identifiers. If they are registered on the ledger, they have to provide an authentication key. A cryptonym-based identifier has the additional prefix "NYM" before the method-specific identifier, resulting in a DID that could like shown in Figure 5.12.

did:v1:nym:4jWHwNdrG9-6jd9I7K1si3kTRneNwftZV9m6rkrAfWQ

Figure 5.12: An example of a Veres One DID of the type "NYM"

The second version of a DID supported in Veres One is an UUID-based identifier for entities that want to store metadata on the ledger, like revocation lists. A UUID-based identifier could look like shown in Figure 5.13.

did:v1:uuid:804c6ac3-ce3b-46ce-b134-17175d5bee74

Figure 5.13: An example of a Veres One DID of the type "UUID"

There are two ways of creating DIDs in Veres One. The first is to pay an accelerator who processes the DID creation, which takes as long as the network comes to a consensus, usually a few minutes. The second way is to submit a creation request to the network and perform a PoW, which takes a few hours but doesn't require a fee. The same policy also counts for updating a DID document. Either pay an accelerator to perform an update operation of the DID or process the operation by yourself.

Veres One provides an alternative way to let users control their identifier on top of a proprietary ledger. In contrast to Sovrin, it follows a permissionless approach that allows anyone to run a node, and therefore requires less governance. In addition, it has a different approach to support DIDs and credentials which makes it interesting for further analysis.

5.2.4 Further DID Methods

In this chapter, further DID methods will be presented that were not selected for detailed analysis but are still worth mentioning.

Another DID method that adapts the Bitcoin blockchain is the Identity Overlay Network (ION) [110] which implements the Sidetree protocol [111]. The Sidetree protocol enables storing of multiple transactions on the Bitcoin blockchain by summarizing and adding them as a single transaction. This results in much higher throughput than the Bitcoin blockchain alone would be able to process. Nodes of ION collect transactions in chronologically ordered batches, which are then stored on the Bitcoin blockchain as well as on IPFS [112]. Unfortunately, detailed information about the processes of ION is scarcely available, as the project is still in the development stages. Therefore, ION will not be addressed in more detail in this thesis.

The DID method TangleID [113] applies the concepts of Self-Sovereign Identity to the Tangle, the ledger of IOTA. It optimizes "Masked Authenticated Messaging" [114], which serves as a second-layer protocol for key management and related features across the Tangle. Encrypted messages are signed by "Merkle tree-based signature schemes" [115]. The root of the Merkle Tree is used as the ID of the channel, and since a single tree only lasts for a short period, each message contains the root of the next Merkle Tree.

The Interplanetary Identifiers (IPID) DID Method [116] supports DIDs on IPFS. It utilizes the Interplanetary Linked Data (IPLD) suite to reference data as an identity. The method-specific identifier is created from Content Identifiers (CIDs), which result from the content hash of the data. IPLD provides a high degree of interoperability, as it makes IPLD a valuable structure for DID documents and may be complementary with other DID methods.

Alastria [117] is a Spanish project that attempts to provide a legally binding identity solution for Spanish citizens. Based on a proprietary ledger, Alastria aims to provide a DID that can execute smart contracts to access services legally. It further intends to issue credentials, as described in their DID method specification [118].

Metadium [119] forked the Ethereum blockchain to develop an SSI system. Besides DIDs, their system consists of a claims and achievements structure that allows making attestations and triggers a reward system when claims are used for any type of interaction. Metadium also provides a wallet and the META-coin that serves as native currency. Furthermore, an extensive governance model establishes trust and secures the system.

Ocean Protocol is a decentralized network for artificial intelligence data and services. This network addresses actors with plenty of artificial intelligence data and provides a marketplace for them to sell the data [29]. Other parties, for instance, those who

perform analytics on this data, could purchase the data. Ocean Protocol, therefore, serves as a marketplace. They involve DIDs to identify data packages, as further described in their DID method specification [120].

The Peer DID method [121] aims to provide technology that enables the exchange of DIDs without recourse to a central authority. This means that only parties that have a connection to each other should be able to resolve the other's DID. Consequently, only related personal information, such as a DID, is visible to authorized parties. Peer DIDs are supported on any ledger that allows implementing Peer's generation algorithm and protocols. Therefore, Peer DIDs aren't limited to one ledger, but can be adopted on multiple ones.

This thesis focuses on DID methods and systems that are widely implemented and in the best case fully open-sourced. Closed source, as well as DID methods and systems in early development stages, are not suitable for extensive research and therefore not relevant for this thesis. This applies to the following DID methods and systems: ArcBlock [122], Bryk [123], Dominode [124], Emtrust Wai [125], ICON [126], Infowallet [127], JLINC [128], Jolocom [129], Ockam [130], Ontology [131], LifeID [132], TokenTM [133], Vivvo [134], and Weelink [135].

6 Analysis of DID Methods

In this chapter, the selected DID methods and their associated systems from chapter 5 will be analyzed and compared based on criteria. These are selected to emphasize the differences in the approaches to establish Self-Sovereign Identity. These criteria serve as the basis to highlight, compare, and ultimately evaluate the differences in DID methods and their systems, where the respective DIDs can be used.

The analysis and evaluation takes place according to the following scheme. First, the criteria and optionally subcriteria are defined and their possible characteristics are presented. The next step is to show the respective DID methods' characteristics and compare them to each other. These results from the analysis are then used for evaluation in chapter 7.

6.1 Status

This criterion describes the current status of the DID methods and their systems. The two characteristics are either "released", when the DID method is ready to use, or "not released" in the case it isn't. The purpose of this chapter is to give an overview about the development status in the SSI ecosystem and help to evaluate the different methods concerning their current status.

BTCR fully relies on the public Bitcoin blockchain, which has been active for more than 10 years and already provides methods to enable SSI. For this reason, BTCR is entirely ready to use in its current state. It is possible to create a DID based on Bitcoin transactions and following to prove ownership of a DID. However, there are still some additional features under development, such as the integration of verifiable credentials.

Blockstack successfully forked the public Bitcoin blockchain and implemented additional features to its resulting Stacks blockchain. Even though the "Stacks blockchain V1" is already released, additional features like proper smart contracts for the use inside the Blockstack system are currently under development and will assumingly be deployed in further versions of the Stacks blockchain.

The ETHR DID method specification with its ETHR DID Registry smart contract [136] and the uPort libraries are already implemented and available on the public Ethereum blockchain. Therefore, ETHR is also considered "released".

The ERC725 concept describes a smart contract that is still a proposed standard. However, ERC725 is already able to be implemented on the Ethereum blockchain. It is expected that there are still some changes in the making to improve the identity smart

contract. As a consequence, ERC725 is considered "released" as well, as it is already possible to create and interact with an ERC725 smart contract.

SelfKey's "DIDLedger" smart contract is also deployed on the public Ethereum blockchain and is able to be used as well [99]. There is no information available about future changes or additions to the DID method. Therefore, SelfKey is also considered "released".

Veres One currently only has a testnet available, where it is only possible to create a DID. As of now, there is no schedule available when the mainnet should be launched. Therefore, Veres One is considered "not released", as there is no mainnet currently available.

Sovrin released its public network [137] to public issuers in March 2019. The system is already extensive with many privacy-preserving tools, like pairwise-pseudonymous DIDs or the ability to create ZKPs from existing credentials. Even though there are still some features in development, the system is ready to use and is therefore considered "released".

6.2 System Design

The second criterion is about analyzing the system design of each DID method and the surrounding identity system. The purpose of this chapter is to give an overview of how identity systems are designed, integrated into the underlying blockchain, as well as which functionalities are supported.

6.2.1 Level of Integration

The level of integration can be described as a first-level integration, when all the activities and processes take place directly on the ledger. Alternatively, it can be described as a second-level integration when another protocol on top of the ledger is implemented, which is where the main action takes place and only a limited amount of data is forwarded to the ledger. A great example of a second-level protocol would be the lightning network [138]. It enables users to create payment channels on a layer on top of the underlying Bitcoin blockchain, which results in almost instant processing and minimal transaction fees since there are only two parties involved in the process. In this channel, multiple transactions can occur but become effective as one transaction transferring the delta of their balances on the blockchain when the channel is closed and the updated balances are broadcasted to the blockchain.

The BTCR DID method takes place on the Bitcoin blockchain itself. There is no other system required besides a publicly available off-chain storage to hold the DID document. Therefore, it is considered to be fully first-level integrated. A second-level

protocol would also not be effective since the link in the OP_RETURN data field refers to the latest DID document. If there are many changes to the DID document planned, it would make sense to not publish the DID document in a new transaction until its final status. Every new indexing of a changed DID document requires a transaction in the Bitcoin blockchain and therefore transaction fees, which should be avoided as much as possible.

Blockstack consists of two components, that are relevant to analyze the level of integration. The first is the Stacks blockchain, which keeps track of the naming transactions, like NAME_PREORDER, NAME_REGISTRATION, and NAME_UPDATE. The second is the "Atlas" Peer-to-Peer network which implements a gossip protocol, a procedure or process of computer peer-to-peer communication, and manages the nodes to avoid issues with joining and leaving of nodes inside the network [80]. Furthermore, the “Atlas” Network is considered an "extended storage subsystem for the Stacks blockchain" [80]. It is used for interactions with other applications such as the Blockstack Naming System, which doesn’t need direct interaction with the blockchain and allows to create Off-Chain DIDs. Due to the fact that applications interact with the “Atlas” Network and not directly with the Stacks blockchain itself, and that "Atlas" stores off-chain names, Blockstack is considered second-level integrating.

uPort and ERC725 use smart contracts that are deployed on the Ethereum blockchain itself to create and manage DIDs. Since these smart contracts allow identity management for each user directly on the blockchain, these DID methods are considered first-level integrated. However, identity systems can be built on top of these DID methods, as demonstrated by SelfKey who implemented the so-called "DIDLedger" smart contract for its identities. Even though the SelfKey network consists of multiple other applications like the marketplace, interactions with the DID always have to be processed by using the smart contract. Therefore, it is considered first-level integrating.

The Sovrin- and the Veres One ledger were built entirely for managing and using identities. In Sovrin, even credentials and revocation registries may be documented on the ledger. All processes are highly integrated into communicating and interacting with the ledger, therefore no second-level protocol is required for identity management and both systems are considered first-level integrating.

Six out of seven DID methods are considered first-level integrating. Identity-relevant operations are performed directly on the ledger and don’t use an additional protocol. The reason for this is the design of the systems, which allows identity management directly on the blockchain. Second-level integration becomes more important when an actual system or business is built around the identity. To give an example, Blockstack and Selfkey, have a system built around the ledger to manage a DID. While Blockstack adopted a second-level protocol to enable Off-Chain DIDs, SelfKey directly interacts with the ledger. The second-level protocol requires further designing and

implementation and leads to increased complexity. However, it also brings advantages, like additional functionality to the system. As a result, second-level protocols increase performance, scalability, and may reduce the number of transactions with the ledger or interaction costs. A second-level protocol is necessary when additional functionality needs to be implemented, in which the underlying ledger doesn't support the required features. In this case, only Blockstack makes use of a second-level protocol.

6.2.2 Ledger Interaction

This chapter faces the question of which data is being written on the blockchain in each of the DID methods.

In BTCR, the entire DID management will be processed on the Bitcoin blockchain. A Bitcoin transaction with an OP_RETURN output data field linking to the most recent DID document is required. Every time changes to the DID document are made, a new transaction is required to index the new DID document.

In Blockstack, six interactions need to happen with the Stacks blockchain to fully manage a DID from the creation until its revocation. The first interaction is the `NAME_PREORDER` transaction, which creates a hash of the username of the identity. The second transaction, the `NAME_REGISTRATION` transaction, reveals the hash and the name to all BDS nodes and assigns a name to an initial public key hash and zone file hash. The third transaction is the `NAME_UPDATE` transaction. It changes the name's zone file hash [139], which manages the mapping of the name and the address. `NAME_TRANSFER` is the fourth transaction. It changes the name's public key hash in the case the private key is changed or the name is transferred to someone else [139]. The fifth transaction, `NAME_REVOKE`, changes the name's zone file hash to `null`, thus preventing the name from being resolved. The sixth and final transaction, `NAME_RENEWAL`, extends the validity of a name after it has expired, for about an additional two more years [139].

In uPort, only a valid Ethereum address with public and private keys is necessary to create a DID. ETHR relies on a smart contract and provides functions to create and manage DIDs where events are created for each change-transaction happening. This transaction happens when owner-, delegates- or attribute information has changed [84]. Each change event links to the block of the previous event. Therefore, every interaction that is not a read-operation with the identity is written on the ledger. Credential management is managed entirely off-chain.

For ERC725, the deployment of the smart contract is registered on the blockchain as well as changes to the DID inside transactions. ERC735 represents a smart contract, so every credential management step requires interaction with the underlying ledger, as well.

In Selfkey, the creation of a DID is achieved by a transaction to the "DIDLedger" smart contract that manages the DIDs. Every change to the DID is stored on the "DIDLedger" contract. As it occurs in uPort, the transactions help the resolver generating the DID document.

Veres One currently only supports the creation of a DID. In the future, it should be possible to perform updates to the DID. Both of these operations should be stored within a transaction on the Veres One ledger. Unfortunately, there is no more information available as of now.

Sovrin provides a document [140] that describes in detail which data is written on the ledger. In Sovrin, DIDs and their associated DID documents can be stored on the ledger if the identity owner wants to be publicly verifiable. However, the identity owner can keep their DID private by not storing it on the ledger. Furthermore, a schema definition is written on the blockchain that defines which attribute data types and formats could be used for claims inside credentials. Based on the schema definition, the actual credential definition describes an issuer-linked subset of the attribute data types from the schema definition connected with the attribute-specific verification keys of the issuer. This enables an issuer to reuse a schema and furthermore allows a verifier to look up the credential definition to verify a received proof. Additionally, revocation registries of the respective issuer are stored on the ledger as well. Verifiers can easily check whether a received credential or proof of a credential is still valid by checking the revocation registry with the "cryptographic accumulator" number within a Zero-knowledge proof.

To summarize, the DID methods mainly document transactions that serve to create or manage the DIDs on the blockchain. However, there are always transaction costs each time ledger interaction takes place. An advantage of uPort and the ETHR DID method is that the addresses are already valid DIDs even without a previous transaction with the blockchain while others must first create them through a transaction. Credential management is used in the form of ERC735 and ERC780 smart contracts in the ERC725 or SelfKey DID method, for which result in transactions for each interaction. uPort solves the problem by managing credentials via wallets and issuing them via JSON Web tokens, and thus do not need to interact with the blockchain. In Sovrin, credentials are also issued via JWT and do not require any ledger interaction. However, an issuer can store revocation registries on the blockchain, serving as a source of trust for verifiers.

Low interaction with the ledger has the advantage that only low transaction costs are incurred. Furthermore, no personal information is stored on the blockchain, which improves privacy.

6.2.3 Functionality

There are multiple ways to use a DID in the SSI ecosystem. This criterion breaks down features inside the selected DID methods and compares different possible implementations. In detail, the chapter describes the possibility to prove ownership as the fundamental feature, followed by the feature to delegate operations. Last but not least this chapter analyzes if there is credential management implemented.

Proving Ownership

The first feature is proving ownership of a DID. This criterion is crucial as it establishes trust in a trustless ecosystem between two parties. By using DID Auth, users can cryptographically prove control over a private key to a corresponding public key of a DID, as described in section 4.1.1. DID Auth fully relies on the DID specification and gets the information about service endpoints, which are required to initiate a DID Auth process, from a DID's DID document.

Since every DID method provides an algorithm to resolve the DID to its DID document, DID Auth is possible in each of the selected DID methods as long as the correct service endpoints and keys are included in the DID document.

Delegation

The second feature is delegating control or responsibilities of one DID to another, which performs operations on behalf of the identity, such as the signing of documents or authentication. This feature is useful as a remote control allows others to manage a DID that belongs to entities that can't control the DID by themselves, like digital objects, children, or representatives of an organization. Furthermore, it could be used to let others pay gas costs for the creation or interaction of a DID. Not included in analyzing the delegation feature in this thesis is the possibility to change ownership, as this is possible in every DID method. This functionality is limited to operations that others perform on behalf of the DID owner.

In BTCR, only the holder of the public and private key of the most recent transaction to the DID can perform updates to the DID. By design, it is impossible that another public key can perform operations on behalf of the current public key inside BTCR. Therefore, BTCR doesn't support delegation.

Blockstack doesn't explicitly allow delegation, but it allows users to register multiple names to an address that is represented with a DID. This name could then perform operations under the name of a DID. Theoretically, the initial DID owner could transfer responsibility to other names.

uPort explicitly allows delegation of a DID in its identity smart contract, as there are functions implemented to add, look up, and revoke a delegate [83]. The addresses listed as delegates in the DID document can perform operations on behalf of the identity owner. The delegation permission may also have a validity limit that erases the status of a delegate after a specific amount of seconds, defined during delegate addition.

The purpose of ERC725 is to serve as a proxy smart contract for other entities. Hence, it allows delegating control to other entities that can process authentication, verification, or any other implemented contract interaction.

SelfKey utilizes the "DIDLedger" smart contract to manage the DIDs of the SelfKey-users. The setController method in the "DIDLedger" smart contract manages the ownership of the DID and doesn't allow delegates to control the DID [99], as DIDs are used to access financial services, which have to respect KYC regulations.

Veres One plans to manage the delegation of operations to third parties with authorization capabilities to perform specific actions with the DID, as described in the Veres One DID method specification [45].

Sovrin provides three different types of relationships as defined in the Sovrin Glossary [141]. A distinction is made between "Delegation", "Guardianship", and "Controller", as illustrated in Figure 6.1.

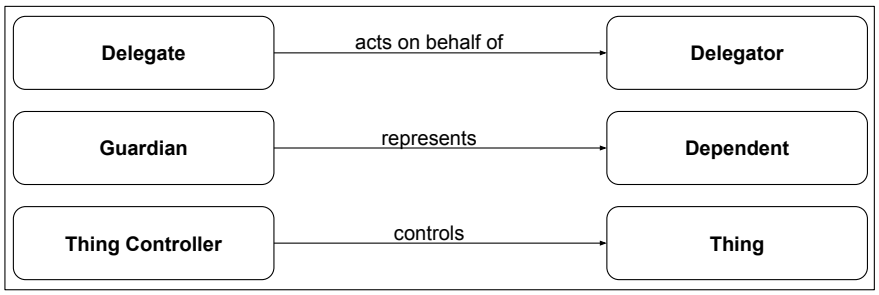

Figure 6.1: The Delegation Types in Sovrin. Adopted from [141].

Delegation is understood as a transfer of responsibilities to another party where both parties have control over their private keys. The delegate can perform operations with its own private key on behalf of the delegator. In order to define a delegate, a delegator credential is issued which can then be used to prove authorization of responsibilities or capabilities to third parties.

The difference between delegation and guardianship in Sovrin is that the dependent individual who is represented by the guardian doesn't control his own private key since it is controlled by the guardian. Thus the guardian is in control of two private keys, one for its own use and one for the delegate. Providing the private key to a guardian

requires a high degree of trust and a formal contract that is defined in the real world between those two parties. However, in some cases, there is no formal contract required since there is already a trusted relationship, such as a parent that acts as guardian to the child's identity or a representative of an organization. In these cases, a guardian credential could be issued which could then be used to prove authorization.

A controller is an identity that controls private keys, wallets, or agents of a thing and therefore enables the identity owner to permanently have full control over it. A thing can be distinguished between passive things that don't have their own private key, like digital documents, and active things, such as computing devices like a smart car. Again, identity owners could self-issue a "Thing Controller Credential", in order to prove authorization to others. Even though these concepts for delegation, guardianship, and controller aren't developed yet, they provide an interesting approach to design transfer of responsibilities inside the Sovrin ecosystem. Additionally, there needs to be an extensive regulatory frame around delegation that covers all the possibilities that could happen with delegation.

Support of Credentials

Another important feature that adds great value to the SSI ecosystem is the support of verifiable credentials. As previously described in subsection 4.1.2, a credential is an attestation of one entity about itself or another that contains one or more statements in the form of claims.

In BTCR, there is currently no way to manage verifiable credentials. However, Kim Hamilton Duffy, co-author of BTCR, mentions in her article "BTCR DID Method Updates" [142] the integration of verifiable credentials as one of the next features to implement in BTCR.

Even though Blockstack forked the Bitcoin blockchain to implement further features, verifiable credentials are not yet or soon to be implemented to the Blockstack ecosystem.

With the credentials library, uPort provides a way to issue and manage credentials. Using this credential library, developers can easily create individual credentials and integrate them into their SSI-compatible application. However, these credentials don't fully conform with the verifiable credential standard.

ERC725 itself doesn't provide credential functionality, as it only serves as a proxy identity contract. However, with the addition of ERC735, claims about an ERC725 identity can be managed and verified. These verified claims can then be used for multiple purposes, such as authentication or proving of statements.

SelfKey also provides the option to attest claims to the identity by utilizing the ERC780 Claims Registry smart contract [100]. These claims are used to attest identity information, such as the name or address, that is required for the KYC and AML

processes. These claims that contain identity information are self-attested but have to be verified by an identity verification provider, as described in the SelfKey whitepaper [97]. The verified claims can then be used to gain access to financial services on the SelfKey marketplace.

Veres One claims on its website to support verifiable credentials in the system [143]. However, at the time this thesis is being written, only the creation of a DID is possible inside a testnet. Unfortunately, there is no sign that credentials are already implemented in Veres One. Moreover, there is no further description on how credentials can be used inside Veres One.

In Sovrin, verifiable credentials play a central role inside the system, which was opened up for credential issuers in March 2019 [144]. Sovrin supports self-attested as well as verifiable credentials that are attested from one entity over another and which fulfill the standard proposed by the Verifiable Credential Working Group of W3C [54]. Credentials are defined by using a credential schema and a credential definition which makes it easier for third-party verifiers to request specific data. Issuers can issue credentials with the personal identity information inside to identities in the given credential definition. Credential holders can present those credentials either in native- or in the form of a ZKP to third party verifiers. ZKPs can be generated based on the claims inside credentials and can also summarize claims from different credentials in one proof.

In summation, there is a wide diversity of features among the selected DID methods. While every DID method supports the authentication feature, four out of the seven have added some sort of delegation feature that allows transferring responsibilities to other entities. The design of the ETHR and ERC725 smart contract allows adding delegates to the contract, whereas Veres One plans to manage delegations with the issuing of capabilities to others. Sovrin also has a very extensive model for remote control of a DID in the form of delegation, guardianship, and control implemented. However, this feature is still in development. While verifiable credentials aren't supported in BTCR or Blockstack, the Ethereum-based DID methods either provide a library to manage credentials, as in the case of uPort, or the ERC735 concept was defined to attest claims to ERC725 identities. SelfKey provides the possibility to attest ERC780 claims but requires a third-party verification service to verify those. Veres One intends to allow credential management but is still in development. However, Sovrin provides already implemented credential feature that conforms the verifiable credential standard provided by W3C. This standard is important to leverage interoperability for credentials. The more systems support this standard, the more cross-ledger interaction can take place.

6.3 Management and Governance

Although the DID methods allow independent creation and management of DIDs, the systems must be managed and governed to define basic rules for the system. This criterion describes how the DID methods and their implemented systems are managed and governed. The purpose of this chapter is to give an overview of how SSI systems can be managed and which conclusions result from different management approaches. For this thesis, two types of management approaches are defined in the next paragraph.

A system can either be managed by the community itself or by an institution. Community-driven DID methods follow a structure where everyone in the community has equal rights, can make proposals, and can be involved with the decision-making process. Hence, the system is considered fully decentralized.

Institutional-driven DID methods and systems are managed by one or a few sets of institutions and often come with a hierarchical structure with different rights and responsibilities for each level. Usually, the highest hierarchical role makes decisions that affect the entire system. This type of organization requires a more sophisticated governance model that covers business, legal, and technical policies for all stakeholders.

The BTCR DID method requires no further implementation on the blockchain. The already existing OP_RETURN data field can be used to attach 40 bits of data like a URL that refers to a DID document on publicly available storage outside the blockchain. BTCR was published by the W3C Credentials Community Group [31] in an attempt to bring SSI to the Bitcoin blockchain and to further develop the BTCR resolver. However, the Bitcoin community seems to be divided in terms of using the OP_RETURN data field [145], as Bitcoin was only initially intended to transfer value. To save storage capacities the size of the initial 80 byte-sized OP_RETURN data field was reduced to 40 bytes in 2014 [146], after proposed and accepted by the community. Changes to the Bitcoin blockchain can be submitted via a Bitcoin Improvement Proposal (BIP).

Blockstack was funded, created, and developed by the Blockstack Public Benefit Corp. Besides the possibility to have on-chain- as well as off-chain names, there is no hierarchical structure in the Blockstack ecosystem. Anyone is free to become an on-chain name. Leader election to create and validate blocks is implemented via the "Tunable Proof" mechanism, which is a combination of Proof-of-Work and Proof-of-Burn. Blockstack also provides the opportunity to propose "Stack Improvement Proposals" (SIP) [147] which proposes changes to the codebase or adds functionality. However, Blockstack decides which features will be integrated to the Blockstack system. Therefore, Blockstack is considered institutional-driven.

The Ethereum blockchain natively provides the possibility to create smart contracts that have specific characteristics and can be used for specific purposes. The project team from uPort implemented the ERC1056 smart contract that serves as the basis for

the ETHR DID Registry smart contract. uPort itself isn't a closed system but a set of libraries built on top of the ERC1056 smart contract inside the Ethereum blockchain that enables management of identifiers and claims. Developers and other participants can access the tools and libraries published by uPort and modify them if necessary. Through the high integration to the Ethereum blockchain and its community, the ETHR DID method is considered community-driven.

A further DID method that is entirely community-driven on the Ethereum blockchain is ERC725, which was proposed by Ethereum developer Fabian Vogelstetter. The ERC725 concept can be implemented and used by everyone participating in Ethereum. Additionally, users can easily run and test the smart contract by themselves and further propose changes. By allowing users to do so, the system could improve over time, as people attempt to make it better and eventually resulting in best-practice processes, which may be published in another DID method based on Ethereum.

SelfKey enables users to manage their identifiers inside the SelfKey ecosystem, which is built on top of the public Ethereum blockchain. This ecosystem is managed by the SelfKey Foundation which implemented a smart contract for its identities and claim management [148]. The SelfKey Foundation develops and manages the entire ecosystem by itself, and is therefore considered institutional-driven. However, users of Selfkey can control their identifiers according to the principles of SSI in a sovereign manner. Besides having validation Ethereum nodes [97], there is no further information about roles inside the SelfKey ecosystem or a governance model.

The Sovrin Foundation legally represents the Sovrin identity system and is responsible for its maintenance as defined in the Sovrin Governance Framework [5]. The head role of the Sovrin Foundation is the Board of Trustees who have to approve changes to the system, such as changes to the governance model. Underneath the Board of Trustees are the stewards who act as trusted entities of the Sovrin Foundation. Their task is to operate the validator nodes to maintain the ledger and therefore enjoy writing permissions on the ledger. On November 2nd, 2019 the Sovrin Network consists of 71 Stewards [149], to achieve decentralization. In order to become a steward, the Sovrin Board of Trustees has to grant permissions to the respective organization after an extensive review process [150]. The Board of Trustees makes essential decisions in the system, even though they are transparently distributed. Therefore, the Sovrin identity system is considered institutional-driven.

Last but not least, the identity project of Veres One is managed by the Veres One Community Group of W3C [151] which is free to join by everyone. The head of the project is the Board of Governors who ensures stability and economic operation as well as approves or declines decisions made by the Veres One Community Group. Nodes secure the computational and operational power of the network. Due to the project's permissionless nature, anyone can set up and run a node [108]. Moreover, anyone

can apply to become an accelerator node, which has privileged access rights on the ledger [152]. Users can pay the accelerator node to create and register their DID instead of submitting a request with an associated Proof-of-Work. Unfortunately, there is no further information about the underlying ledger, which may come from its unfinished status. Due to the fact that the Board of Governors manages the Veres One network, it is considered institutional-driven.

The analysis has shown that three identity systems are community-driven while the other four are institutional-driven. The community-driven ones are often highly connected with the underlying ledger and therefore closely coupled with that ledger's governance. Community-driven identity systems have the advantage that users can directly come up with proposals and that there is already an underlying blockchain where solutions can build upon. Moreover, direct feedback of the other community members could result in diverse responses that could improve the system or change given boundaries like reducing the size of the OP_RETURN data field. However, if everyone has the possibility to make proposals it can quickly become confusing. Additionally, no further governance is required as the systems are designed within the given barriers and provide functionality.

Institutional-driven identity systems require a governance model if a high level of decentralization is intended. Sovrin and Veres One defined business, legal, and technical policies in their governance models which makes the entire system more trusted as well as better structured. The tasks and responsibilities are clearly defined for each role. Even though Blockstack is institutional-driven and does not provide a governance model, they also provide the possibility to submit "Stacks Improvement Protocols (SIPs). It is worth mentioning that an extensive governance model backed by some sort of Board of Directors reduces complexity and streamlines responsibilities. Only a few individuals have to inform themselves and make a decision, while others have the possibility to make proposals to those decision-makers. This reduces the time for making a decision.

In summary, community-driven identity systems tend to work best when they are closely coupled to the underlying ledger, as it is in BTCR, uPort, or ERC725. In particular, the uPort project shows that enough effort in the community could lead to a well-designed system with useful tools. Since the community of the underlying ledger aims to improve the ledger, they are motivated to make it as versatile as possible, which is well-demonstrated in uPort.

On the other hand, if a system is built on a proprietary ledger or adapts an own business, an institutional-driven approach suits better as tasks and responsibilities can be better coordinated and the Board of Directors can provide a vision and streamline processes towards it. The policies of the system should be defined in a governance model on which all participants have to agree on.

6.4 Establishment of Trust

Having trust in a system is essential in a decentralized system. Therefore, the DID methods and their system have to establish trust in order to be successful. This chapter faces how trust is established in the DID methods and their corresponding systems. In order to get a better overview, the trust infrastructure is analyzed separately from the level of trust inside the system. By doing so, this chapter takes a closer look at the trust components in each system.

6.4.1 Trust Infrastructure

Due to the decentralized approach described in DPKI, there is no longer a central institution that creates trust in the identifiers. Thus, the trust in the identifiers is outsourced to the users in a decentralized manner, as described in section 6.4. However, if a system is built around the identifiers, trust has to be established in the system itself.

For the BTCR-, uPort-, and ERC725 DID methods there is no additional trust component built in the DID method. Trust is established with the consensus mechanism of the underlying ledger, in this case, PoW, and to make the code open-source for everyone interested to review. Verifying documents and signatures or using the DID Auth protocol helps to prove and verify the integrity of the DID.

SelfKey is also designed so that users can create and sign transactions directly from their wallet on the Ethereum blockchain. There are no further trust components implemented. To access the marketplace of SelfKey, personal information needs to be verified by a third-party verifier. This information doesn't affect other DIDs but is necessary to conform to KYC and AML regulations.

In the Blockstack system, trust is established through the "Tunable Proof" consensus mechanism where miners have to burn cryptocurrency and do a PoW. The burning of cryptocurrencies shows the intention to participate in the mining process. Anyone can register new usernames in the BNS, by either deploying it on the Stacks blockchain directly or by using a node connected by an on-chain name to register an off-chain name. The trust system between these two parties also relies on mutual trust that the transaction is processed. However, updates to names can be processed by any on-chain name which reduces the coupling with an off-chain name.

Veres One and Sovrin are systems specially designed for enabling the creation and management of decentralized identifiers using an underlying blockchain. Inside this system there are different roles defined with different purposes that require a certain level of trust. These roles, tasks, and responsibilities are described in the following sections. The trust model of Veres One is illustrated in Figure 6.2.

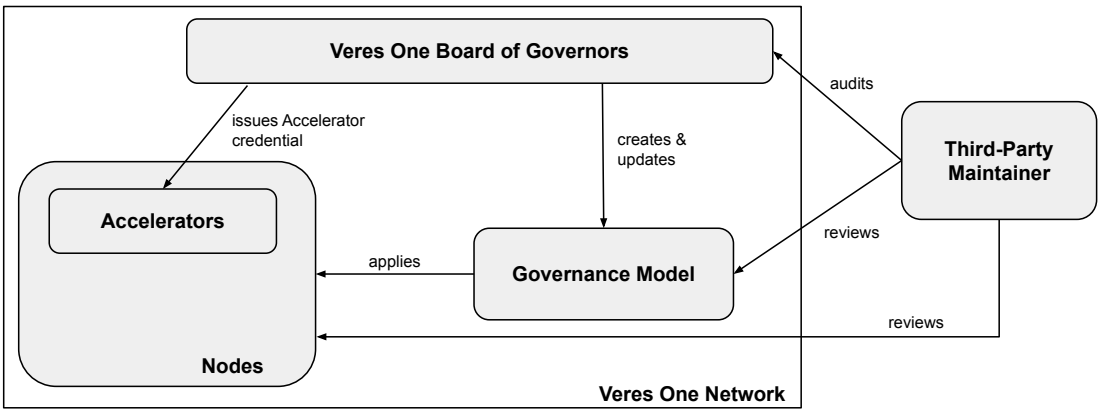

Figure 6.2: The trust model of Veres One

In Veres One, anyone can become a node due to Veres One's permissionless characteristic. The nodes who run a replica of the ledger may participate in the leader election process to determine which events should be included in the ledger. Those leaders are randomly chosen [108] which makes it harder for attackers to successfully take over a node and manipulate data inside the ledger. Additionally, the consensus is Byzantine-Fault tolerant, so it tolerates up to 1/3 of faulty electors. Another role in Veres One is the "Accelerator", which has the privilege to create and update DIDs and charge a specific fee from the user. Those privileged roles cannot be executed by anyone, and therefore require an approval process. A member of the Veres One Project checks provided information on the accelerator application and may eventually approve the application. The new accelerator will be sent an accelerator credential by the Veres One Project [152]. This also means, that users have to trust in the correct assessment of the approver, which is comparable with blindly trusting the CAs in the DNS.

To increase trust and security to the ledger, the Veres One Board of Governors hires "Maintainers" who should maintain the Veres One Software. Maintainers aren't a part of the Veres One project, so that they can act independently. But as the ledger doesn't exist yet, this governance concept is largely hypothetical.

Like Veres One, Sovrin also was designed from scratch to enable an identity system. The described trust model in Sovrin is illustrated in Figure 6.3.

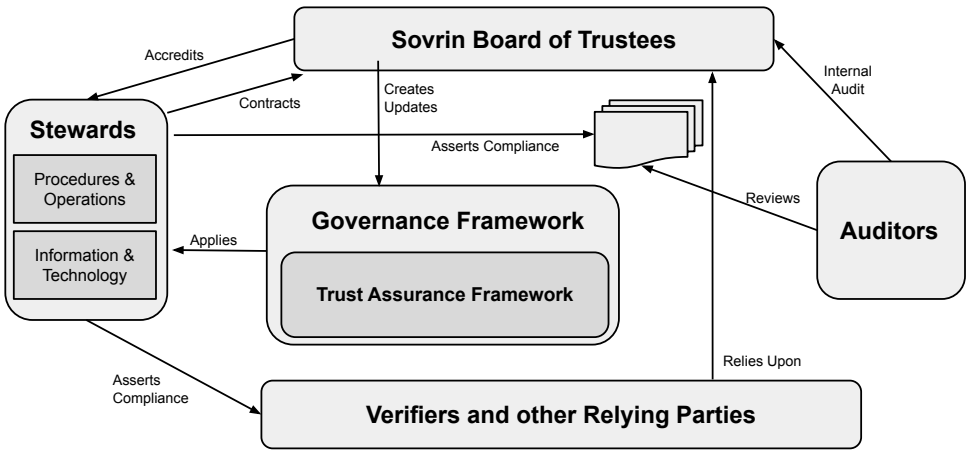

Figure 6.3: The trust model of Sovrin. Influenced by [6].

Roles inside the system and their responsibilities are defined in the Sovrin Governance Framework [5] and in the Sovrin Trust Assurance Framework [6]. The Sovrin Foundation, represented by the Sovrin Board of Trustees and the Stewards, can assert trust in any of the trust elements, for instance by signing contracts or agreements. The Board of Trustees issues the policies of the Governance Framework and also approves or suspends Stewards from the network. Approved stewards can then process transactions to and from the ledger but also have to undergo internal audits by internal and third-party auditors. Verifiers and other relying parties rely on the trust in the system established by the Sovrin Board of Trustees and the information provided by the Stewards.

This analysis has shown that only the proprietary Self-Sovereign Identity-compliant systems defined an extensive trust infrastructure. The other five DID methods and their systems rely on the trust provided by the ledger itself, through their permissionless design and their fault-tolerant consensus algorithms. This kind of trust infrastructure for the proprietary ledgers is necessary for effective decision making in a network. The trust framework tries to achieve decentralization and transparency by implementing the business, political, and technical regulations defined in the respective governance model. Thus, the user can verify that security standards are met by the operating instances and publicly available documentation about decision processes.

6.4.2 Level of Trust

Sovrin issued an entire document on how trust should be established inside an SSI ecosystem with the Sovrin Trust Assurance Framework [6]. This document is derived

from the "AICPA Trust Services Criteria" [153] based on the "COSO Internal Control-Integrated Framework" [154]. According to this Framework, the following trust criteria play a leading role in establishing trust inside a system.

1. **Availability**: Information and systems are available for operation and used to meet the entity's objectives.
2. **Security**: Information and systems are protected against unauthorized access, unauthorized disclosure of information, and damage to systems that could compromise the availability, integrity, confidentiality, and privacy of information or systems and affect the entity's ability to meet its objectives.
3. **Processing Integrity**: System processing is complete, valid, accurate, timely, and authorized to meet the entity's objectives.
4. **Confidentiality**: Information designated as confidential is protected to meet the entity's objectives.
5. **Privacy**: Personal information is collected, used, retained, disclosed, and disposed to meet the entity's objectives.

In this chapter, the DID methods and their systems are evaluated with the given trust criteria.

Availability

The first criterion is availability. In each of the selected DID methods, an underlying ledger and smart contract implemented on the ledger serve as the root of trust to store transactions to manage the DID to create and resolve DID documents to get further information. Due to the decentralized approach, the ledger is always available in every selected DID method and system. However, in BTCR the resolver doesn't directly resolve the DID document. The resolver gathers the data from a file outside the Bitcoin blockchain that was referred by a link to the DID document in the OP_RETURN data field. It may be possible, that this DID document isn't available as the storage location may have changed or is not publicly available anymore.

Another system worth explicitly mentioning is Blockstack. The system is designed with the Stacks blockchain as the basis and a Peer-to-Peer Network "Atlas" on top of it. This network provides a registry of off-chain names and additional functionality without having to directly interact with the blockchain. Since "Atlas" is a Peer-to-Peer network, it is also always available through the integration of many peers that store a copy of the ledger, as long as someone is maintaining the network.

The availability of claims is given where the claims are issued directly on the ledger, as with ERC725 and SelfKey. The availability of claims and credentials depends on the implemented wallet used in uPort-systems and Sovrin. As a conclusion, the selected DID methods fully satisfy the criterion of availability.

Security

The most fundamental security mechanism that exists in all ledgers is that only public keys listed in the corresponding DID document are eligible to perform operations to the specific DID. Therefore, public keys must be added by the initial identity owner, or in the case of ECR725, where the creator of the contract is initially set as the owner.

Furthermore, the underlying consensus mechanism should prevent faulty entries from being added to the blockchain. For Bitcoin and Ethereum, this is the Proof-of-Work mechanism, which makes it almost impossible to manipulate entries in the data. Veres-One and Sovrin rely on the Byzantine-Fault tolerant consensus mechanism that tolerates 1/3 of faulty nodes during consensus. Moreover, Blockstack implemented the "Tunable Proof" consensus mechanism, which is a combination of PoW and PoB. The more cryptocurrency will be burned, the more likely it is for a node to mine the block. This might be a security issue because a miner could burn disproportionately high amounts of cryptocurrency to have higher chances to mine a block and insert faulty data. But even then, the transaction has to be accepted and confirmed during the leaderless and dynamic election of nodes.

In order to secure personal identity information, the wallets containing this information have to be secured against attacks. The user might be the weak piece in the chain as he or she does not securely store the private key or has weak authentication options, allowing criminals to get access to the private key easily or to successfully attack the personal device.

Moreover, for the proprietary ledgers, the system must be protected from undesired behavior through a governance model. There shouldn't be a way for criminals to take advantage of incomplete governance that makes it possible to bypass defined processes and by doing so, erasing the trust in the system.

As a conclusion, the systems appear to be satisfying the security criterion. However, some systems aren't defined as a standard or are still in development, so there might be additional features coming. Furthermore, sometimes security vulnerability may not be recognized until it is too late. Therefore, the community should be actively looking for security vulnerabilities and fix them before criminals make use of it.

Processing Integrity

The third trust criterion is processing integrity, which is given in all DID methods and systems due to their underlying ledger technology. Only well-formed transactions signed by the identity owner are valid for processing. They will be submitted and then verified and mined to blocks by the miners or stewards. Changing the data after they are confirmed on the blockchain is nearly impossible and can only be changed by another transaction. If off-chain names in Blockstack want to be registered on the Stacks blockchain, they have to define a well-formed transaction and promote this transaction to an on-chain name to process it. The on-chain names have no possibility to manipulate the data as the signature of the off-chain name becomes invalid. The off-chain name has the possibility to let the transaction be processed by another on-chain name.

Confidentiality

Confidentiality is a very important topic in Self-Sovereign Identity doesn't face the DID methods directly, but more the systems above it. Claims and credentials can be issued with a validity date, so that it is only valid within a defined timespan and needs to be reissued afterwards. Moreover, credentials or presentations of the credentials presented to third parties have to be processed in a confidential way. This means that only the information required should be presented with the use of selective and minimal disclosure using ZKPs, which currently only Sovrin does. Once presented, the verifier has to make it clear how the data will be used. The credential holder could also add a validity date to the presentation or revoke it once it is not necessary anymore. It is difficult to evaluate the selected DID methods and systems to this criterion because DIDs are initially created to be able to sign credentials, and the credentials themselves have to be managed by the wallets and other agents. However, Sovrin appears to be ahead as they have built-in functionality to derive ZKPs from claims and credentials, as described in a comment on ZKPs by Sovrin [155].

Privacy

Last but not least, privacy is another criterion to establish trust. By design, there is no personal information added to any ledger. Key pairs and resulting DIDs can be created entirely without revealing personal information and serve as a pseudo-anonymous identity. The only information that can be derived from the blockchain is the DID document, which may contain service endpoints that allow authentication. Even though all the transactions are publicly visible, they cannot be referenced to one real-world identity without further knowledge. However, obtaining further knowledge might be easy by reviewing the transaction history to determine where the value

initially came from. Popular exchanges where fiat currency is traded follow KYC and AML regulations and often serve as the first access point to get cryptocurrency. Since the trades and outgoing transactions have to be stored by the exchanges, it may be possible to assert a real-world identity to an address and DID. Therefore, BTCR-, ETHR-, and ERC725 identities might not be as privacy-preserving as it seems. SelfKey allows users to create a DID without further identification but then requires KYC and AML information if the user wants to use a financial service provided in the SelfKey Marketplace.

In Blockstack, no personal information is stored on the blockchain, but it can be attached to the name profile. Names, social media accounts, and similar personal information can be added which might be useful to access applications with the Blockstack identity. How those apps handle privacy is not in the hands of Blockstack anymore, as they just provide an identity to access the service and are not in charge of managing the presented information to the apps. Nevertheless, if the user doesn't add extra personal information to the Blockstack name profile, the privacy characteristic is the same as in Bitcoin or Ethereum.

One thing sets Sovrin apart from other systems is the use of pairwise-pseudonymous identifiers where unique identifiers are created for every connection. This prevents others to identify correlations of the identity owner to other services, since the identifier is used only once. The transactions of one identity owner are not traceable among multiple connections. As in the other DID methods, no personal data such as private DIDs or credentials, are stored on the ledger. However, a public issuer can write its DID on the ledger to make it easier for identity owners and relying parties to verify the DID. Moreover, Sovrin intends to use selective and minimal disclosure by default, which only reveals necessary information to the verifier, enhancing privacy inside the system. Additionally, Sovrin allows issuers to have a revocation registry for each credential definition and contains a long number called "cryptographic accumulator" that is stored on the ledger. Furthermore, the credential holder can create a proof of non-revocation to prove the credential is valid. A third party verifier can then check whether the credential is still valid with the proof of non-revocation and the cryptographic accumulator number.

As a conclusion, privacy by design plays an important role in SSI. DIDs are created in a decentralized way without revealing personal information. However, if transactions were created in Bitcoin or Ethereum, it is possible to follow the transactions to the initial transaction. If the user exchanged fiat currency against cryptocurrency, the exchange might be able to identify the owner due to KYC and AML regulations. While Veres One only provides technology to create and update a DID, Sovrin considered lots of privacy-related topics by using pairwise-pseudonymous DIDs, Zero-knowledge proofs to verify a credential, and selective disclosure to preserve the identity owner's privacy.

6.5 Fee Structure

This chapter considers the fee structure of the selected DID methods and their systems. It describes what types of fees are incurred, how high they are, and to whom they are addressed. Furthermore, it is analyzed if and how a separate token is used.

In BTCR the only fees occurring are the transaction fees in the form of gas costs for creating a new transaction. The average transaction fees on October 12th, 2019 are $0.66 for the next available block. The currency in use is Bitcoin (BTC), which is the standard currency for the Bitcoin blockchain.

Blockstack forked the Bitcoin blockchain and established the “Tunable Proof” consensus mechanism which is a combination of PoB and PoW. Additionally, Blockstack introduced the Stacks token (STX), which serves with Bitcoin as the underlying currency to pay fees for registering digital assets or pay miners to execute smart contracts and transaction fees [156]. A current transaction fee would be STX in the value of around 0.0002 BTC [157], which is worth around $1.67 on October 12th, 2019. Additionally, costs for mining a block sum up with the amount of cryptocurrency burned during the Proof-of-Burn mechanism and the price of the energy needed during PoW. The higher the value of burned coins, the higher the chance to mine the block.

ETHR, ERC725, and SelfKey all rely on the public Ethereum blockchain. Therefore, gas costs occur for every interaction with the Ethereum blockchain or the respective smart contract. ETHR and ERC725 natively interact with the Ethereum blockchain and pay regular gas costs, which is on average about 1 GWEI or $0.004 on October 12th, 2019. SelfKey however, charges an extra fee significantly higher than regular gas costs. on October 12th, 2019 the creation of a DID costs around 0.000831 ETH, or 831000 GWEI, or $0.153. Veres One doesn’t have an own token implemented and uses payments in USD to charge fees. Fees apply when registering a DID ($0.89), updating a DID ($0.25), and issuing a credential, where 3% of the price is charged to the customer [109]. The fees are then distributed among the network nodes, maintainers, non-profit organizations, such as communities inside Veres One or legal departments, savings for open source projects, and the founders. Sovrin does not have a token for the Sovrin network yet but intends to launch its token, as indicated in the "Sovrin Protocol and Token Whitepaper" [158]. Right now, fees are charged in USD. Fees are charged for writing a DID ($10), a credential schema ($50), credential definition ($25), an entire revocation registry ($20), and a revocation update ($0.10) to the ledger, as described in the "Public Ledger Fees" document [159].

Fees for writing to or interacting with the ledger should be as close to zero as possible, especially regarding regular users that have to actively decide to create a DID by themselves. High transaction fees make it unattractive for users to actively participate. While those DID methods that rely on a public, permissionless blockchain

like Bitcoin and Ethereum, are bound to the common transaction fees, proprietary ledgers can define their own fee policy. Veres One charges fees comparable to those of the others, whereas Sovrin has significantly higher fees for writing data to the ledger. It is doubtful that this helps to acquire new users and customers. But on the other side, Sovrin is the most advanced system out of all presented.

7 Evaluation of DID Methods

In this chapter, the selected DID methods and their systems will be evaluated. The evaluation takes place based on how the general concept of SSI is adopted and on the results of the analysis criteria in the previous chapter. Furthermore, the DID methods are suggested for specific users according to their characteristics.

Bitcoin Reference

BTCR had the challenge to bring SSI to the relatively static but very popular Bitcoin blockchain with very limited functionality besides transferring value from one party to the other. The optional OP_RETURN data field of an outgoing Bitcoin transaction can store arbitrary information and enables the establishment of an SSI approach. This data field holds a reference to the DID document that is resolved by the BTCR resolver and is required to establish a connection with others. Due to the limitations of the Bitcoin blockchain, BTCR has only limited functionality and requires transaction fees in the form of gas costs for creating and updating a DID. However, BTCR allows users to independently create identifiers, authenticate themselves, and prove that they have control over the private key to a corresponding address. Personal information isn't involved, as the user enjoys the pseudo-anonymity of Bitcoin. Therefore, BTCR conforms with the concept of Self-Sovereign Identity but is only useful for authentication.

BTCR only has relevance to single entities, such as individuals or organizations, who are using the Bitcoin blockchain to transfer value in the form of Bitcoin.

Blockstack

The goal of Blockstack is to establish a naming system on the Bitcoin blockchain. Due to the limited functionality of the Bitcoin blockchain, Blockstack decided to annually fork it to create the Stacks blockchain. This ledger has improved customized functionality for Blockstack, which increases scalability and enables the implementation of a new consensus mechanism. The "Tunable Proof" consensus mechanism is a combination of "Proof-of-Burn" and PoW that drastically reduces mining efforts without an increased risk of manipulation since the miner has to burn cryptocurrency to increase the chances to mine the block. The forked blockchain allows Blockstack to integrate entirely new systems like the "Atlas" or "Gaia" system on top of the Stacks blockchain. Besides enabling users to anchor their usernames on the Stacks blockchain, off-chain names can be stored outside the blockchain in the Peer-to-Peer "Atlas" system. To sum up, Blockstack enables users to create a self-sovereign identity on a fork of the Bitcoin

blockchain that aligns with the concept of SSI. Consequently, Blockstack is more functional than BTCR, has relatively low and stable transaction fees, is already used by many users, and already relatively mature. However, Blockstack users can add a significant amount of personal information to it, which may be shared by accessing other applications and therefore might not be as privacy-preserving as intended by the SSI concept.

The Blockstack identity can be used to transfer value inside the Stacks blockchain and more importantly, to access several decentralized applications provided in the Blockstack browser that accept this special identity. Therefore, using the Blockstack identity is attractive for users that want to access these decentralized applications with a self-sovereign identity. Among all presented DID methods, Blockstack's is the most applicable due to the use of a Blockstack ID to access several decentralized apps.

uPort

The Ethereum-based project uPort allows users to create a self-sovereign identity on the Ethereum blockchain. uPort is not a system itself but provides a stack of libraries to manage identities utilizing an ERC1056 smart contract. This combination of an ERC1056 smart contract and the provided libraries allows users to develop own systems allowing integration of self-sovereign identity. The main advantage of uPort is that the ERC1056 smart contract allows offline-functionality and delegation that gives high potential for SSI solutions based on this method. By default, an ERC1056 smart contract provides the functionality to manage the owner, delegation, and attribute management. To manage a DID, a regular Ethereum address that is authorized in the DID document can perform actions to a DID without the need of being previously registered on the blockchain. Other addresses can serve as delegates who adopt responsibilities and perform operations on behalf of the DID owner.

Developers can already use existing libraries and either adapt an ERC1056 smart contract or make use of the existing ETHR DID Registry smart contract to manage identifiers in their system. Furthermore, uPort also provides the "credentials"-library that enables the management of customized credentials. However, uPort doesn't provide a solution to establish further trust, such as revocation registries, a trust framework or an extensive DKMS model.

In summation, the uPort project provides many useful tools for developers to build their own systems. Identities can be managed either with an ERC1056 smart contract or the already existing ETHR DID Registry that also allows offline creation of DIDs. Especially useful is the integration of verifiable credentials to the system. It is worth mentioning that there is no communication with the ledger during the interaction with other identities, such as credential exchange. However, the success of a uPort-based

system relies on the system design and its integrated governance model.

ETHR is especially relevant for developers who want to create their own SSI-supporting system. uPort provides plenty of libraries that can be freely used to create such a system. Furthermore, already existing addresses can be easily converted to an ETHR DID, which makes it easier to onboard users. Last but not least, regular Ethereum users can use the ETHR DID Registry to manage their identifiers. Nonetheless, there is no real publication that allows using an ETHR DID.

ERC725

The ERC725 proxy smart contract also enables identity management on the Ethereum blockchain. It is a community-driven concept that allows creating and updating a DID on Ethereum. The smart contract represents the identity and provides the functionality to manage the identity with delegation functionality. There are no further libraries needed in order to interact with an ERC725 smart contract. However, claims to an ERC725 identity can be added through the ERC735 claim concept, which is directly integrated into the Ethereum blockchain. Therefore, this DID method is closely connected to the Ethereum blockchain, but with limited functionality compared to the uPort project. However, it still provides the basic functionality to provide a self-sovereign identity to the Ethereum blockchain and allows users to authenticate, sign documents, and make attestations about each other in the form of claims.

ERC725 is relevant for any user of the Ethereum blockchain that needs basic identity functionality, or for developers who want to create a system based on the ERC725 smart contract.

SelfKey

SelfKey is an SSI-compatible system that allows users to create a DID which can then be used to access a digital marketplace for financial services. SelfKey utilizes the "DIDLedger" smart contract that is designed similarly to ERC1056 but with less functionality. SelfKey DIDs are registered on the Ethereum blockchain and managed with this smart contract that provides basic functionality for the SelfKey system. The system supports the integration of claims that are used to describe the person, such as the name, address, or other personal information. These claims have to be verified by a third-party verifier, which may lead to a bottleneck inside the system.

In summary, SelfKey DIDs are limited to the SelfKey system and aren't usable outside of it. There is only limited functionality implemented and the SelfKey DIDs are mainly used to access financial services. Nonetheless, SelfKey fulfills the concept of SSI as it provides the user with full control over their DID. Using SelfKey DIDs is only relevant

to users inside the SelfKey system to access financial services in its marketplace.

Veres One

The Veres One project tries to establish a DID method based on the proprietary, public, permissionless Veres One ledger. DIDs should be created by either paying an accelerator or by manually doing a PoW and promoting it to the ledger. The Veres One project consists of an extensive governance model that defines the internal business, legal, and technical policies, including funding. Moreover, credential management should also be possible in the future. Unfortunately, by the time of this thesis, there was only a testnet available with the possibility to create a DID. There is still a lot of conceptual and development work to do in order to achieve the goal of providing a self-sovereign identity on a proprietary ledger. Hence, the final scope of Veres One is still not fully determined.

This DID method might become relevant for users who don't want to create a DID on a well-established blockchain like Bitcoin and Ethereum.

Sovrin

Sovrin is an identity system that also relies on the proprietary "Hyperledger Indy" project that serves as the ledger. The Sovrin ledger is a public, permissioned ledger managed by the Sovrin Foundation and its stewards, and is already ready for use. Besides an extended DID management model, Sovrin also provides functionality for issuing and managing credentials. The system provides lots of privacy-preserving functionalities like default pairwise-pseudonymous DIDs for each connection or generating ZKPs based on the user's credentials. By default, DIDs are not written to the ledger to preserve the users' privacy. Additionally, even though Sovrin is managed by the Sovrin Foundation, the governance model decentralizes the power to multiple parties in an open and transparent manner, which increases trust in the system. Sovrin is also designed to easily provide credential management for companies. Based on a credential schema and a credential definition, issuers and verifiers can easily issue or verify credentials to or of users. Moreover, revocation registries allow centralized management of credential revocation for one issuer. This makes it easier for verifiers to check the validity of a presented credential. Additionally, Sovrin provides many concepts that describe processes like delegation in a very detailed way.

The system is relevant not only for individuals but also for companies and organizations that search for a well-thought-out identity system that meets their needs for identity management. The more participants involved in the system, the more attractive it gets for others to join.

In summation, the Sovrin system is an extensive, well-thought-out, and currently the most complete self-sovereign identity system available. It fully implements identity features and is similar to real-world identity management. However, Sovrin charges relatively high fees to write on the ledger, which might be difficult to convince users and companies to onboard to the system. Also, the provided documentation isn't easily approachable, such as the delegation concepts that are very extensive but lack in detailed processes of how it should be achieved. Additionally, Sovrin doesn't provide an approach for DKMS and seems to be very difficult to set up or review the entries on the ledger.

During the analysis, it has also emerged that the DID methods and their systems are limited to their respective ledger. There is no approach to enabling cross-ledger interoperability, which may result in many distinct identities. One of the main principles of Christopher Allen is to establish interoperability, which would allow the identity to be as widely usable as possible, among many different ledgers. This is only partly achieved since DIDs are tailored for the use of the respective system, and because the ledgers generally do not allow cross-ledger interaction. As a result, some parts of the components architecture, like DKMS, are not well thought out or still missing. The organizations providing SSI solutions aren't capable of making the connections needed to enable broad adoption.

Furthermore, the usage of a DID is highly dependent on the used wallet that manages it. Additionally, wallets are important for storing and managing credentials but unfortunately aren't sufficiently considered by the systems. In contrast to the ledgers themselves, wallets can be designed to support different ledger technologies. Therefore, the ability to use a DID also depends on the used wallet, since DIDs should be easy to import into wallets. An specification for an interoperable wallet spec could solve this problem.

8 Conclusion and Future Work

This chapter summarizes the results from the previous chapters. Further topics will be presented that can be further analyzed in future research projects.

8.1 Conclusion

In this thesis, the ecosystem of SSI was analyzed in detail. The components of SSI were extensively analyzed and evaluated, as well as existing DID methods and their systems.

In the beginning chapters, the evolution of online identities was summarized. It emerged, that the online identities evolved from a centralized to a user-centric approach before ultimately the user himself can manage his or her identity by using a self-sovereign identity. Subsequently, it was found that the current architecture of the Internet is incompatible with the concept of SSI and that new components must take on the tasks of central institutions.

To answer the first research question using grounded theory methods, a component architecture with four layers for self-sovereign identity was designed in chapter 4. The underlying ledger serves as a source of trust and enables the implementation of DID methods. From these, DIDs can be created that represent the identity of the user. DIDs, verifiable credentials, and verifiable presentations are stored in wallets that can be used for authentication between users by using the DID Auth protocol. This architecture consists of multiple trust-enhancing mechanisms and processes to create a Web of Trust that aims to outsource the trust between users to the technology.

In respect to the second research question, one result of the analysis of the components was that wallets play a much bigger role than initially expected. Unfortunately, DKMS can not fully take up this role as exact procedures and specifications are not sufficiently defined. Consequently, a standard for wallets would help promote interoperability and usability. However, DPKI, which results from the DID standard, as well as credentials and presentations, are already well-designed and easy to understand. It is uncertain whether the web of trust in its current form is sufficient since it follows a promising approach but only the extensive use of SSI will show whether it is good enough or requires further work.

For the DID method analysis seven representative DID methods and their systems were selected. Those are Bitcoin Reference (BTCR), Blockstack, uPort, ERC725, SelfKey, Sovrin, and Veres One.

Analyzing the DID methods and their systems on their status, system design, trust architecture, management, and fee structure has highlighted their differences, served

as criteria to emphasize differences between the DID methods. One finding was that they are very much based on underlying ledgers and the functionality depends heavily on them if no second-level protocol is built on top. As a result, Sovrin can provide more features than others through the development of Hyperledger Indy. It also shows that governance models are needed to foster trust in the systems and to target their development. Furthermore, fees are often only occurring during ledger interaction in the form of gas costs, while proprietary ledgers have fixed fees for performing operations. Especially Sovrin charges significantly higher fees for ledger interaction.

Another result was that individuals benefit the most from the DID methods and their systems, as they are now able to manage their digital identity in a sovereign way. However, uPort and Sovrin are also interesting for companies or institutions to either build an SSI-compatible system or make use of well-designed infrastructure to issue and manage credentials.

8.2 Future Work

This thesis served to provide an extended and detailed overview of the Self-Sovereign Identity ecosystem. However, there are still topics worth to further investigate that were not explored, since it would go beyond the scope of this thesis. Therefore, this chapter provides starting points for further research.

One criticism mentioned in the thesis was the lack of interoperability between the DIDs. Currently, DIDs are only usable while interacting with their native ledger. To achieve interoperability, it is important to figure out how cross-ledger interaction can be processed and prevent having multiple distinct identities that are only usable on their native ledger. Utilizing IPFS and IPLD to enhance interoperability might be an interesting approach, as it can act as a bridge between did methods and chains due to its neutrality. Furthermore, it would be interesting to find out which approaches are currently in research or development to solve this problem.

Another point that was not very well designed is the use of agents and wallets for the user. Agents and wallets are the edge to the online identity and are crucial for the success of Self-Sovereign Identity. Many identity systems, which might also have a registered DID method, already provide their users with a wallet to manage their identifiers. An extended analysis could investigate different styles of wallets and their functionalities. Based on this information, suggestions or a framework can be derived on what wallets should be able to do, such as provide the functionality to create ZKPs for presentations.

As already mentioned in the thesis, the adoption of SSI needs to be incentivized for users and service providers. This raises the question of how the adoption of SSI

could be initiated. Does there need to be a top-down approach, that forces the users to use SSI or does the community demand the use of SSI? This comes along with the challenge to make SSI as simple to use as possible, so onboarding of users as an entry barrier can be reduced. The simpler it is for the user, the better it will get adopted. Moreover, those interactions need to be legally binding.

Conventional identities are already heavily integrated into the everyday life of each user. Lots of personal information is already connected to the existing identifier. Therefore, it raises the question of how conventional identities or identity formats can be integrated into the self-sovereign identity world, such as the eIDAS identity format.

It would also be interesting to see how much SSI is already involved in the blockchains. Therefore, future research projects could focus on which SSI-relevant data is currently written to different blockchains. To do so, the respective ledger can be analyzed according to the resolving algorithms provided in the DID methods specification to find out which data was written to the blockchain. For instance, the OP_RETURN data field of transactions in Bitcoin could be analyzed by which data it contains and how high the percentage of SSI-relevant transactions is.

Missing regulatories are also a big problem, since SSI doesn't have the status of a legally binding identity. A great topic for further work would be to figure out what needs to be done to make SSI legally binding and especially if SSI with the use of ZKPs is suitable to solve KYC and AML problems.

Last but not least, SSI is a concept that gains its value through actual development and execution. Therefore, specific use cases covering real-world examples that would benefit of SSI could be implemented using one or multiple DID methods to design systems able to use SSI. Based on these implementations, it could be analyzed how the system performs and where missing components need to be designed and integrated.

List of Figures

List of Tables

Bibliography

[1] K. Charmaz and L. L. Belgrave. "Grounded theory." In: *The Blackwell encyclopedia of sociology* (2007).

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

[3] M. Sabadello. *GitHub - Peacekeeper/Blockchain Identity*. URL: https://github.com/peacekeeper/blockchain-identity.

[4] A. Tobin and D. Reed. *The Inevitable Rise of Self-Sovereign Identity*. Tech. rep. 2017. URL: https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf.

[5] The Sovrin Foundation. *Sovrin Governance Framework V2 Master Document V1*. Tech. rep. 2019.

[6] The Sovrin Foundation. *Sovrin Trust Assurance Framework V1*. Tech. rep. 2019. URL: https://sovrin.org/wp-content/uploads/Sovrin-Trust-Assurance-Framework-V1.pdf.

[7] D. Dorje, C. Lundkvist, P. Kravchenko, J. Nelson, M. Sabadello, G. Slepak, N. Thorp, and H. T. Wood. *Decentralized Public Key Infrastructure*. Tech. rep. 2015. URL: https://github.com/WebOfTrustInfo/rwot1-sf/blob/master/final-documents/dpki.pdf.

[8] Rebooting Web of Trust. *Web of Trust Info*. URL: https://github.com/WebOfTrustInfo.

[9] A. Abraham. "Whitepaper Self-Sovereign Identity." PhD thesis. 2017.

[10] L. Lesavre, P. Varin, P. Mell, M. Davidson, and J. Shook. *A Taxonomic Approach to Understanding Emerging Blockchain Identity Management Systems*. Tech. rep. 2019. URL: https://doi.org/10.6028/NIST.CSWP.07092019-draft.

[11] P. Dunphy and F. A. P. Petitcolas. *A First Look at Identity Management Schemes on the Blockchain*. Tech. rep. 2018. URL: https://arxiv.org/ftp/arxiv/papers/1801/1801.03294.pdf.

[12] D. Baars. *Towards Self-Sovereign Identity using Blockchain Technology*. Tech. rep. 2016. URL: https://essay.utwente.nl/71274/1/Baars_MA_BMS.pdf.

[13] Z. A. Lux, F. Beierle, S. Zickau, and S. Göndör. *Full-text Search for Verifiable Credential Metadata on Distributed Ledgers*. Tech. rep.

[14] J. S. Hammudoglu, J. Sparreboom, J. I. Rauhamaa, J. K. Faber, L. C. Guerchi, I. P. Samiotis, S. P. Rao, and J. A. Pouwelse. *Portable Trust: biometric-based authentication and blockchain storage for self-sovereign identity systems*. Tech. rep. 2017. URL: http://arxiv.org/abs/1706.03744.

[15] J. Camenisch and A. Lysyanskaya. *A Signature Scheme with Efficient Protocols*. Tech. rep. URL: https://groups.csail.mit.edu/cis/pubs/lysyanskaya/cl02b.pdf.

[16] World Wide Web Foundation. *History of the Web*. URL: https://webfoundation.org/about/vision/history-of-the-web/.

[17] D. Gisolfi, M. Patel, and R. Radulovich. *Decentralized Identity Introduction*. Tech. rep. 2018. URL: https://www.ibm.com/downloads/cas/OPEQYEL7.

[18] K. Cameron. *The Laws of Identity*. Tech. rep. 2005. URL: https://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf.

[19] Internet Society. *Building Trust*. URL: https://www.internetsociety.org/issues/trust/.

[20] C. Liu and P. Albitz. *DNS and BIND*. 1999. URL: http://web.deu.edu.tr/doc/oreily/networking/dnsbind/ch01_02.htm.

[21] OpenID Foundation. *What is OpenID?* URL: https://openid.net/what-is-openid/.

[22] OAuth Core Workgroup. *OAuth Core 1.0*. URL: https://oauth.net/core/1.0/.

[23] OpenID Foundation. *OpenID Connect*. 2010. URL: https://openid.net/connect/.

[24] T. M. Tongue. *What is "Sovereign Source Authority"?* 2012. URL: https://www.moxytongue.com/2012/02/what-is-sovereign-source-authority.html.

[25] B. Logan. "Facebook suspends data-analysis firm Cambridge Analytica from platform." In: *Business Insider Deutschland* (2018). URL: https://www.businessinsider.de/facebook-suspends-cambridge-analytica-strategic-communication-laboratories-2018-3.

[26] R. V. Raman. "British Airways customer data stolen." In: *Business Insider* (2018). URL: https://www.businessinsider.com/british-airways-customer-data-stolen-2018-9?IR=T.

[27] W3C Credentials Community Group. *Verifiable Claims Task Force*. URL: https://w3c.github.io/vctf/.

[28] M. Laskus. *Decentralized Identity Trilemma*. 2018. URL: http://maciek.blog/dit/?cookie-state-change=1571321373068.

[29] Ocean Protocol Foundation. *Technical Whitepaper*. Tech. rep. URL: https://oceanprotocol.com/protocol/#papers.

[30] W3C Credentials Community Group, D. Reed, M. Sporny, M. Sabadello, D. Longley, C. Allen, and R. Grant. *Decentralized Identifiers (DIDs) v0.13*. 2019. URL: https://w3c-ccg.github.io/did-spec/.

[31] W3C Credentials Community Group. *W3C Credentials Community Group*. URL: https://www.w3.org/community/credentials/.

[32] W3C. *W3C Community Final Specification Agreement (FSA)*. URL: https://www.w3.org/community/about/agreements/fsa-deed/.

[33] D. Reed and M. Sporny. *DID Primer*. URL: https://github.com/WebOfTrustInfo/rwot5-boston/blob/master/topics-and-advance-readings/did-primer.md.

[34] S. Conway, A. Hughes, M. Ma, J. Poole, M. Riedel, S. M. S. P. D, and C. Stöcker. *A DID for Everything*. 2019. URL: https://github.com/WebOfTrustInfo/rwot7-toronto/blob/master/final-documents/A_DID_for_everything.pdf.

[35] Curity Identity Server. *Pairwise Pseudonymous Identifiers*. URL: https://curity.io/resources/operate/tutorials/advanced/ppid/.

[36] The European Parliament and of the Council. *eIDAS REGULATION (EU) No 910/2014*. Tech. rep. 2014. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0910.

[37] O. Terbu. *Leveraging eIDAS for DID*. 2018. URL: https://github.com/WebOfTrustInfo/rwot7-toronto/blob/master/topics-and-advance-readings/leveraging-eidas-for-did.md.

[38] M. Sabadello, D. Reed, and M. Sporny. *DID Specification - DID Methods*. URL: https://w3c-ccg.github.io/did-spec/#did-methods.

[39] W. C. C. Group, D. Reed, M. Sporny, M. Sabadello, D. Longley, C. Allen, and R. Grant. *Interoperability in Decentralized Identifiers (DIDs) v0.13*. URL: https://w3c-ccg.github.io/did-spec/#interoperability.

[40] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström. *JSON-LD 1.0*. URL: https://www.w3.org/TR/json-ld/.

[41] W3C Credentials Community Gro, D. Reed, M. Sporny, M. Sabadello, D. Longley, C. Allen, and R. Grant. *DID Document in Decentralized Identifiers (DIDs) v0.13*. URL: https://w3c-ccg.github.io/did-spec/#did-documents.

[42] C. Lundkvist. *IPLD as A General Pattern For DID Documents*. 2018. URL: https://github.com/WebOfTrustInfo/rwot7-toronto/blob/master/topics-and-advance-readings/ipld_did_documents.md.

[43] K. H. Duffy, C. Allen, R. Grant, and D. Pape. *BTCR Resolver*. 2018. URL: https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/btcr-resolver.pdf.

[44] Decentralized Identity Foundation. *Sovrin Resolver*. URL: https://github.com/decentralized-identity/universal-resolver/tree/master/drivers/sov.

[45] M. Sporny, D. Longley, and C. Webber. *Veres One DID Method 1.0*. URL: https://w3c-ccg.github.io/didm-veres-one/.

[46] D. I. Foundation. *Universal Resolver*. URL: https://uniresolver.io/.

[47] D. I. Foundation. *Universal Resolver implementation and drivers*. URL: https://github.com/decentralized-identity/universal-resolver.

[48] M. Sabadello. "DID Resolution: Given a DID how do I retrieve its document?" In: *SSI Meetup* (). URL: https://ssimeetup.org/did-resolution-given-did-how-do-retrieve-document-markus-sabadello-webinar-13/.

[49] Decentralized Identity Foundation. *Universal Registrar*. URL: https://uniregistrar.io/.

[50] Decentralized Identity Foundation. *Universal Registrar API Documentation*. URL: https://github.com/decentralized-identity/universal-registrar/blob/master/docs/api-documentation.md.

[51] M. Sabadello, K. D. Hartog, C. Lundkvist, C. Franz, A. Elias, A. Hughes, J. Jordan, and D. Zagidulin. *Introduction to DID Auth*. Tech. rep. URL: https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-spring2018/blob/master/draft-documents/did_auth_draft.md.

[52] Internet Engineering Task Force. *JSON Web Token (JWT)*. 2015. URL: https://tools.ietf.org/html/rfc7519.

[53] M. Sabadello. "Introduction to DID Auth for SSI." In: *SSI Meetup* (). URL: https://ssimeetup.org/introduction-did-auth-markus-sabadello-webinar-10/.

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

[55] D. Khovratovich. *Anonymous Credentials*. Tech. rep. 2016. URL: https://github.com/hyperledger-archives/indy-anoncreds/blob/master/docs/anoncred-usecase1.pdf.

[56] D. Goleman, R. Boyatzis, and A. Mckee. *JSON Web Signatures*. Tech. rep. 2019.

[57] D. Longley, M. Sporny, and C. Allen. *Linked Data Signatures 1.0*. URL: https://w3c-dvcg.github.io/ld-signatures/.

[58] A. Greenberg. "An Unprecedented Heist Hijacked a Brazilian Bank's Entire Online Operation." In: *WIRED* (2017). URL: https://www.wired.com/2017/04/hackers-hijacked-banks-entire-online-operation/.

[59] G. Breckenridge. "A Brief History of Digital Identity - Mattereum - Humanizing the Singularity." In: *Medium* (). URL: https://medium.com/humanizing-the-singularity/a-brief-history-of-digital-identity-9d6a773bf9f5.

[60] G. Slepak. *DIDs in DPKI (Decentralized Public Key Infrastructure)*. 2018. URL: https://github.com/WebOfTrustInfo/rwot7-toronto/blob/master/topics-and-advance-readings/dids-in-dpki.md.

[61] M. Lodder. *Recommendations for DKMS*. 2017. URL: https://github.com/WebOfTrustInfo/rwot5-boston/blob/master/topics-and-advance-readings/dkms-recommendations.md.

[62] D. Reed. *Decentralized Key Management System*. 2017. URL: https://github.com/WebOfTrustInfo/rwot4-paris/blob/master/topics-and-advance-readings/dkms-decentralized-key-mgmt-system.md.

[63] P. Zimmermann. *Why I Wrote PGP*. 1999. URL: https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html.

[64] P. Zimmermann. *PGP User's Guide, Volume I: Essential Topics*. 1994. URL: https://web.pa.msu.edu/reference/pgpdoc1.html.

[65] S. Appelcline, D. Crocker, R. Farmer, and J. Newton. *Rebranding the Web of Trust*. 2015. URL: https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust/blob/master/final-documents/rebranding-web-of-trust.pdf.

[66] H. Story. "Why did the PGP Web of Trust fail?" In: *Medium* (). URL: https://medium.com/@bblfish/what-are-the-failings-of-pgp-web-of-trust-958e1f62e5b7.

[67] O. Poupko. *A Trustless Web-of-Trust – Establishing Identity Uniqueness*. 2018. URL: https://github.com/WebOfTrustInfo/rwot7-toronto/blob/master/topics-and-advance-readings/trustless-web-of-trust.md.

[68] H. Stahl, T. Capilnean, P. Snyder, and T. Yasaka. *Peer to Peer Degrees of Trust a white paper from Rebooting the Web of Trust VII*. Tech. rep. URL: https://www.prnewswire.com/news-releases/botnet-detection-market-worth-11911-million-usd-by-2023-681515921.html.

[69] F. Klint. "A $50 Million Hack Just Showed That the DAO Was All Too Human | WIRED." In: *WIRED* (2016). URL: https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/.

[70] *Advogato - Trust Metric*. URL: https://web.archive.org/web/20170627230829/http://www.advogato.org/trust-metric.html.

[71] Government of Canada. *Pan-Canadian Trust Framework Overview*. URL: https://canada-ca.github.io/PCTF-CCP/overview/pctf-overview.html.

[72] D. Reed. "The Story of Open SSI Standards." In: *SSI Meetup* (). URL: https://ssimeetup.org/story-open-ssi-standards-drummond-reed-evernym-webinar-1/.

[73] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. *Chameleon-hashes with ephemeral trapdoors and applications to invisible sanitizable signatures*. Tech. rep. 2017.

[74] D. Reed, M. Sporny, and W3C Credentials Community Group. *DID Method Registry*. Tech. rep. 2019. URL: https://w3c-ccg.github.io/did-method-registry/.

[75] C. Allen, K. H. Duffy, R. Grant, and D. Pape. *BTCR DID Method*. URL: https://w3c-ccg.github.io/didm-btcr/.

[76] veleslavs. *BIP 136: Bech32 Encoded Tx Position References*. URL: https://github.com/bitcoin/bips/pull/555.

[77] Blockstack. *Blockstack DID method specification*. URL: https://github.com/blockstack/blockstack-core/blob/master/docs/blockstack-did-spec.md.

[78] J. Nelson. *The Launch of the Stacks Genesis Block*. URL: https://blog.blockstack.org/the-launch-of-the-stacks-genesis-block/.

[79] Blockstack. *Blockstack FAQs*. URL: https://docs.blockstack.org/faqs/allfaqs#what-is-leader-election-and-proof-of-burn.

[80] M. Ali, J. Nelson, A. Blankstein, R. Shea, and M. J. Freedman. *Blockstack Technical Whitepaoer*. 2019. URL: https://blockstack.org/whitepaper.pdf.

[81] Blockstack. *Register a name*. URL: https://docs.blockstack.org/core/naming/register.html.

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

[83] uPort. *ETHR DID Registry*. URL: https://github.com/uport-project/ethr-did-registry.

[84] J. Thorstensson. *ERC1056: Lightweight Identity*. URL: https://github.com/ethereum/EIPs/issues/1056.

[85] uPort. *ETHR DID Resolver*. URL: https://github.com/decentralized-identity/ethr-did-resolver.

[86] *ETHR DID Registry on Etherscan*. URL: https://etherscan.io/address/0xdca7ef03e98e0dc2b855be647c39abe984fcf21b.

[87] *uPort Connect*. URL: https://developer.uport.me/uport-connect/index.

[88] *Uport Transports*. URL: https://developer.uport.me/uport-transports/index.

[89] *uPort Credentials*. URL: https://developer.uport.me/uport-credentials/index.

[90] *JSON RPC - Ethereum Wiki*. URL: https://github.com/ethereum/wiki/wiki/JSON-RPC.

[91] E. Mentie and uPort. "Goodbye uPort DIDs, Hello Ethr-DIDs." In: *Medium* (). URL: https://medium.com/uport/goodbye-uport-dids-hello-ethr-dids-ea2e80256f54.

[92] F. Vogelstetter. *ERC725: Proxy Account*. URL: https://github.com/ethereum/EIPs/issues/725.

[93] M. Sabadello, F. Vogelstetter, and P. Kolarov. *ERC725 DID Method Specification*. URL: https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/topics-and-advance-readings/DID-Method-erc725.md.

[94] J. Santos. "First impressions with ERC 725 and ERC 735 — identity and claims." In: *Hackernoon* (). URL: https://hackernoon.com/first-impressions-with-erc-725-and-erc-735-identity-and-claims-4a87ff2509c9.

[95] F. Vogelstetter. *ERC735: Claim Holder*. URL: https://github.com/ethereum/EIPs/issues/735.

[96] S. James. "Origin DApp gets ERC 725 Identity, Transaction Steps, Escrow, and Reviews." In: *Medium* (). URL: https://medium.com/originprotocol/tech-update-dapp-gets-erc-725-identity-transaction-steps-escrow-reviews-298a7c91b7a5.

[97] The SelfKey Foundation. *SelfKey Whitepaper*. Tech. rep. 2017. URL: https://selfkey.org/wp-content/uploads/2017/11/selfkey-whitepaper-en.pdf.

[98] The SelfKey Foundation. *SelfKey DID Method Specification*. URL: https://github.com/SelfKeyFoundation/selfkey-did-ledger/blob/develop/DIDMethodSpecs.md.

[99] SelfKey Foundation. *SelfKey "DIDLedger" Smart Contract*. URL: https://github.com/SelfKeyFoundation/selfkey-did-ledger/blob/develop/contracts/DIDLedger.sol.

[100] J. Thorstensson. *ERC780: Ethereum Claims Registry*. URL: https://github.com/ethereum/EIPs/issues/780.

[101] *Sovrin - Website*. URL: https://sovrin.org/.

[102] Linux Foundation. *Hyperledger Indy – Hyperledger*. URL: https://www.hyperledger.org/projects/hyperledger-indy.

[103] M. Lodder, D. Hardman, and The Sovrin Foundation. *Sovrin DID Method Specification*. URL: https://sovrin-foundation.github.io/sovrin/spec/did-method-spec-template.html.

[104] *Veres One - Summary*. URL: https://veres.one/summary/.

[105] Digital Bazaar. *Solutions for the next generation Web*. URL: https://digitalbazaar.com/.

[106] Digital Bazaar. *Web Ledger Continuity Consensus Protocol*. URL: https://github.com/digitalbazaar/bedrock-ledger-consensus-continuity.

[107] X. Hao, L. Yu, L. Zhiqiang, L. Zhen, and G. Dawu. *Dynamic practical byzantine fault tolerance*. Tech. rep. 1999. DOI: 10.1109/CNS.2018.8433150. URL: http://www.pmg.csail.mit.edu/papers/osdi99.pdf.

[108] *Veres One - Nodes*. URL: https://veres.one/network/nodes/.

[109] *Veres One - Funding*. URL: https://veres.one/network/funding/.

[110] Decentralized Identity Foundation. *Identity Overlay Network (ION)*. URL: https://github.com/decentralized-identity/ion.

[111] Decentralized Identity Foundation. *Sidetree Protocol Specification*. URL: https://github.com/decentralized-identity/sidetree/blob/master/docs/protocol.md.

[112] A. Simons. "Toward scalable decentralized identifier systems." In: *Microsoft Tech Community* (). URL: https://techcommunity.microsoft.com/t5/Azure-Active-Directory-Identity/Toward-scalable-decentralized-identifier-systems/ba-p/560168.

[113] IOTA Foundation. *TangleID DID Method Specification*. URL: https://github.com/TangleID/TangleID/blob/develop/did-method-spec.md.

[114] P. Handy. *Introducing Masked Authenticated Messaging*. URL: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e.

[115] ImperialViolet. *Hash based signatures*. 2013. URL: https://www.imperialviolet.org/2013/07/18/hashsig.html.

[116] J. Crunch. *IPID DID Method*. URL: https://did-ipid.github.io/ipid-did-method/.

[117] *Alastria - Website*. URL: https://alastria.io/en/.

[118] Alastria. *Alastria DID Method Specification (Quorum version)*. URL: https://github.com/alastria/alastria-identity/wiki/Alastria-DID-Method-Specification-(Quorum-version).

[119] *Metadium - Website*. URL: https://www.metadium.com/.

[120] Ocean Protocol Foundation. *Ocean Protocol DID Method Specification*. URL: https://github.com/oceanprotocol/OEPs/tree/master/7/v0.2.

[121] O. Deventer, C. Lundkvist, M. Csernai, K. Den Hartog, M. Sabadello, S. Curren, D. Gisolfi, M. Varley, S. Hammann, J. Jordan, L. Harchandani, D. Fisher, T. Looker, B. Zundel, and S. Curran. *Peer DID Method Specification*. URL: https://openssi.github.io/peer-did-method-spec/index.html.

[122] ArcBlock. *ArcBlock DID Method Specification*. URL: https://arcblock.github.io/abt-did-spec/.

[123] M. Allende, S. Murcia, F. Munhoso, and R. Cessa. *Bryk DID Method Specification*. URL: https://github.com/bryk-io/did-method/blob/master/README.md.

[124] Dominode. *Trusted Professional Identity Solutions for a Global Economy*. URL: http://dominode.com/.

[125] Halialabs Pte Ltd. *EmTrust Wai DID Method Specification*. URL: https://github.com/Halialabs/did-spec/blob/gh-pages/readme.md.

[126] ICON Foundation. *ICON DID Method Specification*. URL: https://github.com/icon-project/icon-DID/blob/master/docs/ICON-DID-method.md.

[127] Raonsecure. *Infowallet DID Method Specification*. URL: https://github.com/infowallet/did_method/blob/master/did_method.md.

[128] V. Grey. *JLINC DID Method Specification*. URL: https://did-spec.jlinc.org/.

[129] C. Cunningham and Jolocom. *Jolocom DID Method Specification*. URL: https://github.com/jolocom/jolocom-did-driver/blob/master/jolocom-did-method-specification.md.

[130] Ockham. *Ockham DID Method Specification*. URL: https://github.com/ockam-network/did-method-spec/blob/master/README.md.

[131] Ontology. *Ontology DID Method Specification*. URL: https://github.com/ontio/ontology-DID/blob/master/docs/en/DID-ONT-method.md.

[132] lifeID Foundation. *LifeID DID Method Specification*. URL: https://lifeid.github.io/did-method-spec/.

[133] Token.TM. *TokenTM DID Method Specification*. URL: https://github.com/TokenTM/TM-DID/blob/master/docs/en/DID_spec.md.

[134] Vivvo Application Studios. *Vivvo DID Method Specification*. URL: https://vivvo.github.io/vivvo-did-scheme/spec/did-method-spec-template.html.

[135] Weelink. *Weelink DID Method Specification*. URL: https://weelink-team.github.io/weelink/DIDDesignEn.

[136] uPort. *EthereumDIDRegistry Smart Contract*. URL: https://github.com/uport-project/ethr-did-registry/blob/develop/contracts/EthereumDIDRegistry.sol.

[137] *Sovrin Main Net Indy Network Browser*. URL: https://sovrin-mainnet-browser.vonx.io/.

[138] M. Loeb and D. H. Holding. "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments." In: *Perception and Psychophysics* (2016). ISSN: 15325962. DOI: 10.3758/BF03205969. URL: https://lightning.network/lightning-network-paper.pdf.

[139] Blockstack. *Manage BNS Names*. URL: https://docs.blockstack.org/core/naming/manage.html.

[140] A. Tobin. "Sovrin: What Goes on the Ledger?" In: (2017). URL: https://www.evernym.com/wp-content/uploads/2017/07/What-Goes-On-The-Ledger.pdf.

[141] The Sovrin Foundation. *Sovrin Glossary V2*. Tech. rep.

[142] K. H. Duffy. "BTCR DID Method Updates." In: *Medium* (). URL: https://medium.com/@kimdhamilton/btcr-did-method-updates-d0fd14386139.

[143] *Veres One - Credentials*. URL: https://veres.one/credentials/.

[144] The Sovrin Foundation. *Sovrin Network Now Ready for Digital Credential Issuers*. URL: https://sovrin.org/sovrin-network-now-ready-for-digital-credential-issuers/.

[145] almel. *Explanation of what an OP_RETURN transaction looks like*. URL: https://bitcoin.stackexchange.com/questions/29554/explanation-of-what-an-op-return-transaction-looks-like/29555#29555.

[146] jgarzik. *script: reduce OP_RETURN standard relay bytes to 40 by jgarzik · Pull Request #3737 · bitcoin/bitcoin · GitHub*. URL: https://github.com/bitcoin/bitcoin/pull/3737.

[147] *Stacks Improvement Protocol*. URL: https://github.com/blockstack/blockstack-core/tree/develop/sip.

[148] SelfKey. *SelfKey Claim Registry*. URL: https://github.com/SelfKeyFoundation/selfkey-claim-registry/blob/develop/contracts/SelfKeyClaimRegistry.sol.

[149] *Sovrin Stewards*. URL: https://sovrin.org/stewards/.

[150] G. Zimmerman and D. Reed. *Becoming a Sovrin Steward A Briefing from the Sovrin Foundation*. Tech. rep. 2016. URL: https://www.evernym.com/wp-content/uploads/2017/07/Becoming-a-Sovrin-Steward.pdf.

[151] *Veres One Community Group*. URL: https://www.w3.org/community/veres-one/.

[152] *Veres One - Accelerators*. URL: https://veres.one/network/accelerators/.

[153] AICPA Assurance Services Executive Committee. *Trust Services Criteria*. 2017. URL: https://www.aicpa.org/content/dam/aicpa/interestareas/frc/assuranceadvisoryservices/downloadabledocuments/trust-services-criteria.pdf.

[154] J. S. Mcnally. *The 2013 COSO Framework and SOX Compliance*. Tech. rep. 2013. URL: https://www.coso.org/documents/COSO%20McNallyTransition%20Article-Final%20COSO%20Version%20Proof_5-31-13.pdf.

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

[156] M. Ali, J. Nelson, A. Blankstein, R. Shea, and M. J. Freedman. *Stacks Token Economics*. 2019. URL: https://blockstack.org/tokenpaper.pdf.

[157] Blockstack. *Add Bitcoin Gas*. URL: https://docs.blockstack.org/org/wallet-use.html#add-bitcoin-gas.

[158] P. J. Windley, D. Reed, and The Sovrin Foundation. *A Protocol and Token for Self-Sovereign Identity and Decentralized Trust*. 2018. URL: https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf.

[159] The Sovrin Foundation. *Public Ledger Fees*. Tech. rep. 2018. URL: https://sovrin.org/wp-content/uploads/2019/05/Public_Ledger_Fees_Writes_Definition_010519.pdf.