Self-Sovereign Identity: Proving Power over Legal Entities

Tim Speelman

July 2020

Self-Sovereign Identity: Proving Power over Legal Entities

by

Tim Speelman

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Tuesday July 21, 2020 at 10:30.

Student number:	4096533	
Project duration:	April 12, 2019 – July 21, 2020	
Thesis committee:	Dr. ir. J. A. Pouwelse,	TU Delft, supervisor
	Dr. J. S. Rellermeyer,	TU Delft
	Dr. N. Tintarev,	TU Delft

An electronic version of this thesis is available at `https://repository.tudelft.nl`.

KVK

TU Delft

Abstract

Self-Sovereign Identity (SSI) is a new paradigm in digital identity systems that puts the end-user in control: no other actor manages, permits or revokes their digital existence. TrustChain is an academic peer-to-peer networking stack supporting SSI. It delivers passport-grade assurance by integrating with Dutch government. However, end-user control requires a programmed *user agent* with a human interface and protocols that enable meaningful communication with issuers and verifiers of identity data. This agent must be inter-operable with a large variety of parties and credentials. TrustChain lacks such an interface and protocols.

This thesis makes three main contributions. First, a theoretical framework is proposed for aligning notions of self-sovereignty across contexts, borders and cultures. It provides more detailed, focused and structured discourse than other work and helps consolidate design efforts. Second, a design project is done in collaboration with the Kamer van Koophandel (KVK). It focuses on 'authorisation by legal entities', a class of identity problems that have no satisfactory solution yet. Third, a generic common 'semantic layer' is prototyped, consisting of a smartphone based user agent and communication protocols. Its wallet-centric approach allows end-users to retrieve their data without leaving the app. The practical value of this prototype is evaluated at a construction site.

The case study shows that the Kamer van Koophandel, like other government institutions, can be a valuable data provider. However, their current legal framework and business model may restrict them. Absence of such vital institutions invites *commercial* parties to close the gap, threatening privacy and independence of end-users.

Finally, this work has three implications for TrustChain. First, attestation metadata must be considered confidential. Second, single-sided public revocation is required to ensure credential actuality without re-issuing. And third, non-interactive verification enables the construction of chains of untrusted issuers. This is a valuable feature as it enables individuals, not just organisations, to issue claims to others.

Preface

A little over a year ago, I had completed my last exam. After an Industrial Design Bachelor, a pit-stop at Mechanical Engineering and a Master's program at Computer Science, I had yet to make another difficult decision. What would be my graduation topic? When Quinten told me about his work, I could not believe my own enthusiasm. The notion of Self-Sovereign Identity (SSI) appealed to me from the start; giving people an unquestionable and autonomous existence in the digital world, like they have in the material world.

Quinten also promised that my project would not be boring and he was right. From the start, my supervisor Johan Pouwelse challenged me to go out and talk to people. This has led to a close and pleasant collaboration with Joost, Saïd and the rest of the Innovation Lab team at the Kamer van Koophandel. Their enthusiasm has inspired me greatly. In parallel, I had the honour of joining the Digicampus SSI committee, working together with Rijksdienst voor Identiteitsgegevens (RvIG), Stichting ICTU and passport provider IDEMIA. I would like to thank Giulietta and Menno for bringing me along on the Digicampus visit to Tallinn, Estonia. In several discussions with the friendly Estonians, I witnessed both an impressive digital government as well as cultural differences that put my project into perspective. I will not forget the inspiring phrase "Self-sovereign identity is like giving a hand grenade to a monkey". Furthermore, I would like to thank those in the Dutch public and banking sector who have invited me to join or host workshops about this topic and learn from their perspectives. I am glad to have met Christopher Allen in person. I also truly appreciate the opportunity that Helga, Will and Niels gave me to do a real world demonstration with my prototype at a construction site.

This project has been quite a learning process. Inspired by the many people I have spoken to and the enormous ambition that is inherent in the Self-Sovereign Identity movement, I struggled to keep focus. I struggled to find something that I could contribute to this movement. I must thank Johan for letting me choke on my own ambition and then put my feet back on the ground, every so often.

I struggled to balance the complex and endless reality with the academic purity that is expected of a Master's thesis. Yet, I felt that a broader consideration of this topic was appropriate. This resulted in the book that lies before you. I am truly grateful to Floor, Helga, Ilse and Reijer for helping me put all of this into writing.

Contents

1 Introduction

Self-Sovereign Identity (SSI) is a new paradigm in digital identity systems that puts the end-user in control [1]: no other actor manages, permits or revokes their digital existence. Phil Zimmermann's revolutionary software, Pretty Good Privacy (PGP), empowered everyone to encrypt e-mail using self-generated keys [2]. Privacy and independence for everyone was the dream, yet the software was used by almost no one [3]. 29 years, this is still just a dream, only with a new name and much more ambition.

The term *self-sovereign identity* does not have a clear definition, however the most referenced description is Christopher Allen's ten principles in his 2016 blog post *The Path to Self-Sovereign Identity* [1], summarised in Table 1.1. The concept has two key characteristics: (1) only the individual holds the *keys* to their identities and (2) actors can only communicate *about* the individual by directly talking *to* the individual (see Figure 1.1). This provides the individual with unprecedented control.

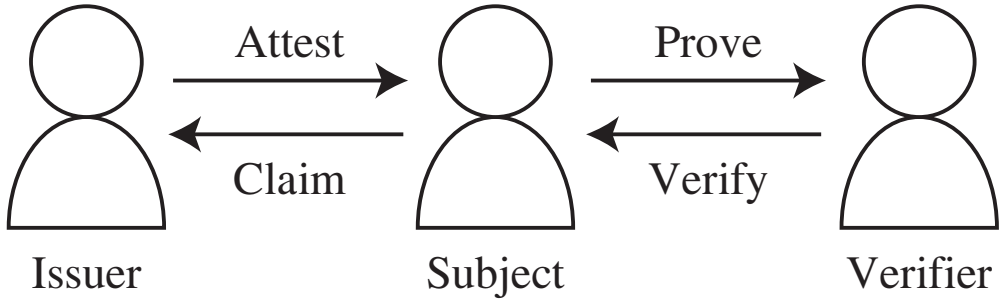


Figure 1.1. Communication in Self-Sovereign Identity. Actors only speak directly to the subject.

1. Existence	Users must have an independent existence.
2. Control	Users must control their identities.
3. Access	Users must have access to their own data.
4. Transparency	Systems and algorithms must be transparent.
5. Persistence	Identities must be long-lived.
6. Portability	Information and services about identity must be transportable.
7. Interoperability	Identities should be as widely usable as possible.
8. Consent	Users must agree to the use of their identity.
9. Minimalization	Disclosure of claims must be minimized.
10. Protection	The rights of users must be protected.

Table 1.1. Self-Sovereign Identity Principles (Christopher Allen) [1].

With public key cryptography, any party can generate a pair of keys: a *public key* to distribute and a *private key* to keep secret. Using the private key, one can *sign* messages, which can be *verified* using the corresponding public key, facilitating *message integrity* and *authenticity*. Vice versa, a message *encrypted* with a public key can only be *decrypted* using the private key, enabling *message confidentiality* [4]. Exclusive possession over the private key is a common way to prove identity [5].

However, for many purposes, a *verifier* wishes to know something *meaningful* about an individual, or *subject*, like their real name, age, education level, etc. The subject can claim anything about themselves, but who will believe them? This introduces the need for an independent trusted party, an *issuer* (e.g. a government, school, social media platform), to assert such traits and *attest* to them. A problem that arises here is that in order to do business with the verifier, the subject depends on the issuer. This impacts the subject's privacy and independence. The self-sovereign identity paradigm reduces the impact of this dependency by having the issuer share a digitally signed credential with the subject. The subject can then use this credential anytime with any verifier that they deem appropriate, without depending on, or informing, the issuer.

From the perspective of the every day user, self-sovereign identity will be a mobile application that holds one's identities (keys) and associated data from different parties. The user can then use this app to share information with all other parties who may need it. A commonly used analogy is that of the physical Wallet and credentials. In every day life, people hold a set of credentials – such as an id-card, credit card, health insurance card, driver's license – in a physical wallet. These credentials are tamper-resistant documents holding some attributes about the subject. They are issued by authorities, banks, insurers and relied upon by the same or other parties for all kinds of purposes. The owner of the wallet controls the credentials and who sees them,

without notifying the issuer or asking its permission. The mobile application is therefore often called a Wallet.

The self-sovereign identity movement has the ambition to create a universal identity infrastructure that allows users to participate in all kinds of identity transactions with just one set of credentials in one Wallet application. This need not imply that the same identity is used everywhere, just that all information can flow through the same system. This ambition is captured in Allen's principle of *interoperability*.

Several projects are currently developing an identity infrastructure in line with this vision, such as uPort [6], Sovrin [7], Jolocom [8], IRMA [9] and TrustChain [10]. Only IRMA and TrustChain are academic works, the others are industry efforts which are mostly explained in white papers or lack proper documentation. No self-sovereign identity framework is in operation today.

1.1 Focus of this Thesis

Since 2017, the Dutch government experiments with TrustChain, one of several projects that are currently developing a novel self-sovereign identity infrastructure [6]–[9]. TrustChain is designed to provide the same legal assurance as the Dutch passport and promises superior scalability over alternatives. The second phase of these experiments has been completed during the time of this thesis project. The result is an Android based application built on TrustChain that uses live facial recognition to authenticate its owner as a Dutch citizen and provide its national identity attributes. The third phase aims to integrate the existing application with third party use cases.

TrustChain lacks a proper semantic layer; a layer that provides meaning to the abstract cryptographic operations performed by TrustChain. The Python library and REST API currently offered by TrustChain do not offer much "control" to the layman. A user agent is required that offers an unambiguous human interface. Furthermore, other parties' agents must be able to communicate their intentions in ways that can be understood by the human end-user. The (user) agents and the protocols that bind them, collectively referred to as the *semantic layer*, must be sufficiently generic to satisfy the principle of interoperability [1]. They must be effective in a wide variety of identity use cases. This thesis contributes to the TrustChain project by applying it to a class of practical use cases and developing a semantic layer in the process.

1.2 Approach and Contributions

In order to develop a semantic layer for TrustChain, first an attempt was made to theorise a generic system that supports a wide range of use cases. However, this appeared to be extremely complex and of limited practical value. In order to achieve a more concrete direction with clear requirements, an incremental strategy was adopted: starting with a narrow focus on one use case, gradually expanding to other cases in future work.

Whilst providing direction, such specific focus may risk *locking-in* the use case. Re-purposing an identity solution built in one context to fit into another has shown to be problematic before [11]. Three measures are taken to avoid this problem and enable future increments to continue this work. First, a theoretical framework for self-sovereignty is proposed that helps consolidate this work with similar design projects. Second, the cryptographic foundation is studied with generic use in mind. And third, the findings from the chosen use case are translated explicitly to a generic semantic layer.

The use case of interest considers *authorisation by legal entities*: is the cleaner authorised to sell the building? This study is performed in collaboration with the *Kamer van Koophandel* (Chamber of Commerce) – the institution managing the Dutch trade register and supporting Dutch entrepreneurs – as part of their mission to establish trust in digital commerce. The study is first framed by applying the proposed theoretical framework and contextualised from business, social and legal perspectives. It then discusses how persons and authorisations may map to pseudonyms and attributes, and whether the Kamer van Koophandel can act as the root identity provider. These findings manifest in a proof of concept for a peer-to-peer mobile app that allows people to authorise one another, named *Zekere Zaken* (Secure Business).

The study of this specific use case provides input to the development of a generic semantic layer on top of TrustChain. It also elicits the end-user's need for a separate Wallet application that provides full control over their identity and safely integrates with which other apps, such as Secure Business. A semantic layer is designed using these insights, which includes a running prototype for three actors – the *issuer*, the *subject* and the *verifier* – and protocols for their communication.

To summarize, this thesis makes three main contributions: a theoretical framework for self-sovereignty, a detailed use case study of *legal entity representation* on SSI, and a prototyped semantic layer on top of TrustChain. The artifacts produced by this thesis can subsequently be used to further apply to other use cases. Hopefully, these contributions enable future work to incrementally expand to coverage of many other use cases.

1.3 Structure of this Thesis

Chapter 2 describes the problem that this thesis will focus on. The *theoretical self-sovereignty framework* is described in chapter 3 and the *cryptographic layer* in chapter 4. The use case *authorisation by legal entities* is studied in chapter 5. Chapter 6 discusses the design considerations for the *semantic layer* and its prototype is presented in chapter 7. The proof of concept of the *Secure Business application* is presented in chapter 8. Chapter 9 presents a real world experiment on a construction site. Finally, conclusions and future work are discussed in chapter 10.

2

Problem Description

The new paradigm of self-sovereign identity puts the end-users in control of their identity (in the form of cryptographic keys) and the data or attributes that are associated with it. It allows them to collect credentials from any connected identity provider, or issuer, and present these to relying parties, or verifiers. Yet, to exert this control, users must have an unbiased view on what issuers offer and what verifiers ask (principle of *transparency* [1]) and some mechanism to enforce their will. A user interface that is in full control of the counter party cannot be assumed to hold the users' best interests at heart. Instead, users need an independent representative, or user agent; a piece of software with a graphical interface that informs them, collects input and protects them from malicious actors. In contrast to other self-sovereign identity developments, TrustChain lacks such user agent.

A significant level of interoperability is required. Whereas identity providers and relying parties could originally communicate about the user directly, SSI dictates that all user-specific information must be passed to the user. This leads to a logical bottleneck, at the agent and the human operating it. It requires a protocol that allows these parties to negotiate the exchange of identity information in such a way that the user understands. This must be sufficiently generic that the user has a consistent experience across all kinds of identity operations [12] and that identity data can be used with as many parties as possible (principle of *interoperability* [1]).

In 1999, Whitten and Tygar provided an explanation to the disappointing uptake of PGP in a usability study titled "Why Johnny can't encrypt" [3]. Next to users lacking a conceptual understanding of the security model, the authors argued following: "People do not generally sit down at their computers

wanting to manage their security; rather, they want to send email, browse web pages, or download software, and they want security in place to protect them while they do those things." As identity is a means to an end, this activity of collecting and sharing identity information should be integrated fluently in external applications, much like modern day banking apps offer an easy payment flow in e-commerce. This involves easy access to existing data sources that house information about end-users, as well as simple means to exchange the necessary credentials with access control mechanisms, be it physical or virtual.

Designing a user agent that works seamlessly across many contexts is out of scope for this thesis. Instead, an incremental approach is adopted, by focusing on a subclass of applications: those involving *legal entity authorisation*. This specific problem can be formulated as:

> **Problem:** verify that a person is authorised to act on behalf of some organisation.

In collaboration with the Dutch Chamber of Commerce, an application is developed that provides high legal assurance, but also convenience, in verifying such authority. This is a complex case that heavily depends on the ability for individuals to digitally identify and sign, making it an appropriate application for the generic semantic layer. The scope is limited to Dutch legal entities, natural persons and legislation. The goal is to answer the following question in an automated fashion:

> **Assertion:** $Q(S, P, L)$: subject S has a power P over legal entity L.

In general, i.e. for any combination of S, P and L in any legal system, answering this is extremely hard, if not impossible. Hence, for high risk transactions, a notary is called in to perform various checks, thereby consulting several registers. He then produces a *legal opinion*, which is an official statement of his findings. For many day-to-day activities, however, such rigor is unnecessary and business can be conducted by mutual trust and the laws of *Power of Attorney* (volmacht). The goal is to prove these authorisations in a self-sovereign manner, using the TrustChain infrastructure and the semantic layer that is developed in this thesis.

3

A Theoretical Framework for Self-Sovereignty

This chapter investigates the notion of 'sovereignty' in the context of digital life. It combines literature with the insights gained over the course of this thesis project, and proposes a new model to debate, analyse, design for and evaluate *digital self-sovereignty*. Such a model can help consolidate and align similar design projects and other discourse, to arrive at a universal solution for digital identity.

The closest alternative to an accepted definition for 'self-sovereign identity' (SSI) is Christopher Allen's ten principles (Table 1.1). It has significant overlap with another well known work published 14 years earlier: Kim Cameron's Seven Laws of Identity [12]. This theoretical framework is inspired on their work and extends it. First, the ground values of *independence* and *privacy* are added, as well as missing principles such as *usability* and *convenience*. Structure is provided by identifying relations between principles and distinguishing *qualities* such as *transparency* from *dimensions* such as *context*, *time* and *human inclusion*. Third, focus is added to the discourse by distinguishing its different aspects visually.

3.1 The Model

The model is visually represented in figure 3.1. Its starting point is the human (bottom), who intends to fulfil some purpose (top) through the system of interest (middle). The pillar describes the system of interest along 8 principles that together shape the sovereignty of the human in that context. The principles are ordered in a hierarchy to illustrate that each quality depends to a certain extent on the layers below it, much like Maslow's hierarchy of human needs [13]. Lining the pillar are two ground values, Independence and Privacy, which are closely related to each of the qualities. The Boundaries of Sovereignty illustrate the question of *authority* that is unavoidable when discussing self-sovereign identity, as explained further in the next section. The remaining sections will explain the details of this model.

3.2 Sovereignty

There is ongoing debate about the appropriateness of the term "self-sovereignty" [14], as it is sometimes mistaken for anarchy: that the individual can fully dictate *who* they are [15], [16]. However, the analogy with state sovereignty is quite fitting, considering a quote from the 1935 book *World Systems Analysis: an Introduction* by Immanuel Wallerstein:

> "Sovereignty is more than anything else a matter of legitimacy [which] requires reciprocal recognition. Sovereignty is a hypothetical trade, in which two potentially conflicting sides, respecting de facto realities of power, exchange such recognitions as their least costly strategy." [17]

Translated to the context of digital space, this implies that an individual should be considered *equal* in dealing with other parties. Instead of being a *client* to their *server*, they interact as peer to peer. In this sense, self-sovereign identity promises to reduce information asymmetries and (thereby) power asymmetries that now exist between individuals and companies [18] (or other actors). The term *sovereignty* is therefore adopted in this manuscript.

3.2.1 The Boundaries of Sovereignty

Sovereignty does not imply anarchy: as states cooperate to accomplish mutual goals, individuals can cooperate with issuers and verifiers to establish trust in their identity. Yet, the misconception of an individual's unbounded freedom over their own identity does spark an interesting debate: what falls under the *sovereignty* of the individual? Is the individual allowed to access all information

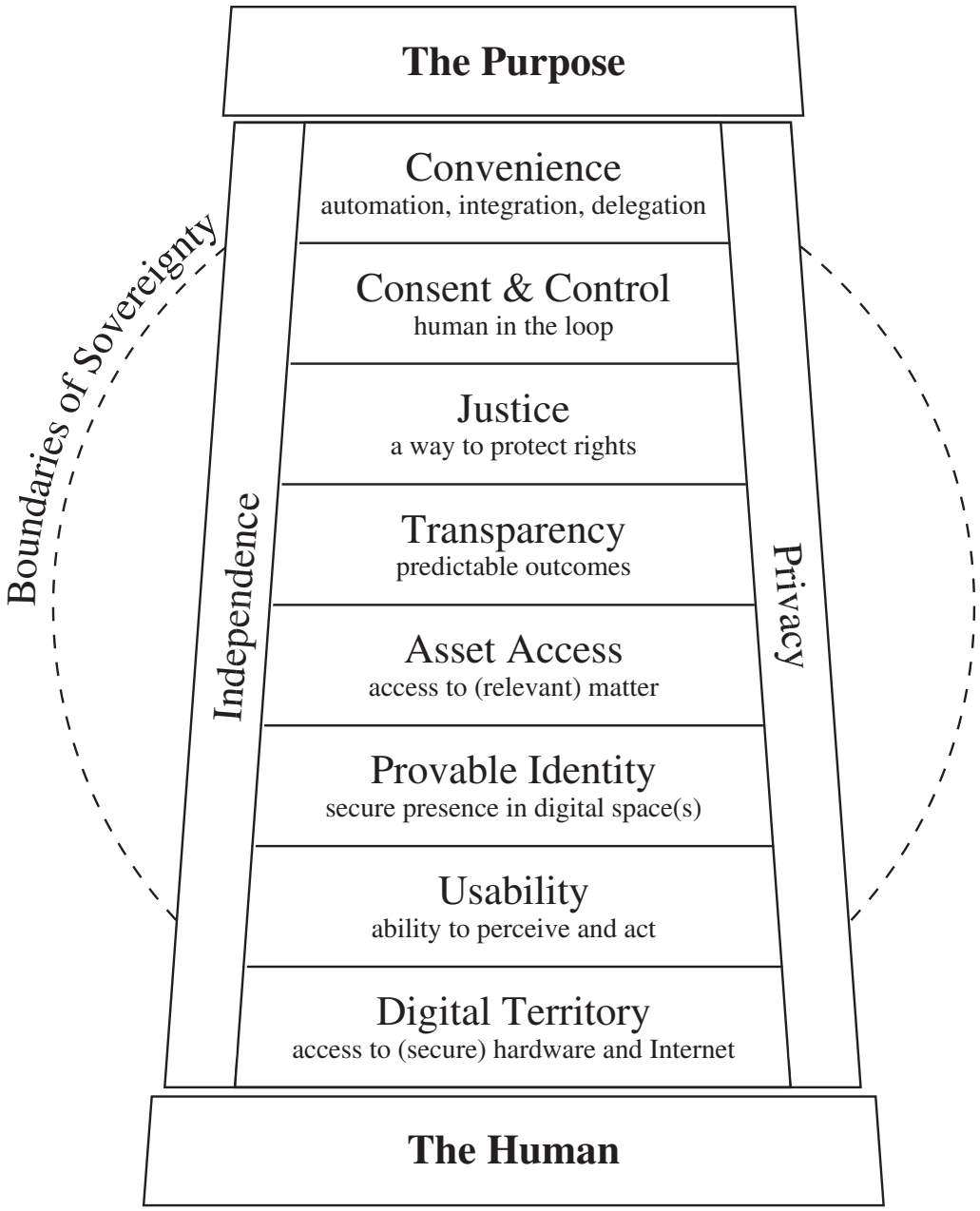


Figure 3.1. Model for Digital Sovereignty.

related to them, regardless of who holds it? Are they allowed to manipulate it? Does the user hold sovereignty over the *content* of identity data, or only over its *use*? It is key to make this debate explicit, rather than bake implicit assumptions into our technologies. Moreover, it is unlikely that a uniform answer will be achieved across context, country and culture, so any 'universal' system must take care of this variability.

3.3 The Human

Self-sovereign identity obviously starts with the self [1]. Hence, the HUMAN is emphasised as the primary object of study. Realisation of digital identity seems a delicate balance between *binding* an individual to their *virtual* appearances and protecting them from it. The appropriate balance then depends on the context: committing fraud online should be punishable in the material world, but getting killed in an online game should not bring the same fate. It needs to be explicitly argued what bond between these entities is desired and why.

Furthermore, as the digital activity has become an indispensable part of human life, and *identity* an indispensable condition for digital activity, self-sovereign identity should be available to all humans [19]. This increases the complexity of the problem, as it involves dealing with the large variety in people's capabilities, norms and values.

3.4 The Value of Independence

In contrast to the physical world, in digital space humans rely on technologies and actors to *be* and *act*. These essential components are created, operated and monitored by often invisible actors forming the (trusted) *middlemen* in many digital transactions. Imagine a classroom full of nosy children passing your written note to a buddy on the other end. Every new dependency introduces threats to one's privacy and autonomy, from Wi-Fi sniffers [20] and 5G network providers [21] to companies and governments that have a monopoly on information and how it is presented [22].

Cameron argues in his third Law of Identity that disclosure of identifying information should be "limited to parties having a necessary and justifiable place in a given identity relationship." [12]. However, such caution must be taken with the entire stack of components and actors. Whenever possible, open standards and open source code should be implemented [1], [23]. Furthermore, one should be careful to rely on governance, legislation and auditing as mechanisms to protect individuals. Even when effective, it simply replaces one corruptible actor with another.

However, also the individual can be corrupted themselves. Norberg et al. have shown that "individuals will *actually* diclose a significantly greater amount of personal information than their stated intentions indicate", for example when offered free products and services [24]. They call this phenomenon the "privacy paradox". Hence, whilst one should strive towards independence, individuals' vulnerabilities must be taken into account.

3.5 The Value of Privacy

Ideally, the individual remains fully anonymous online until they decide to disclose or prove certain aspects about themselves. *Privacy as the Default* is one of the seven Privacy-By-Design principles [25], similar to the *opt-in* directive in the ISO29100 Privacy Framework [26]. This simple privacy measure is motivated on the observation that people hardly ever change the default settings [27]. The extent to which the individual should have agency over their privacy impact is part of the debate on the boundaries of sovereignty (section 3.2.1) as some believe that the average user is not competent enough.

Privacy in relation to self-sovereign identity, concerns not only the identifying or otherwise sensitive data about an individual, but also (metadata of) their online activities, when and with whom. There is plenty of academic literature discussing privacy issues and solutions. Whilst it is not useful to summarise this literature here, it is important to consider the impact on one's privacy at every turn.

3.6 Principles

3.6.1 Principle 1. Digital Territory

Obviously, for the human to be autonomous in the digital world, they must have access to it. This starts with access to the Internet, preferably reliable, free of discrimination and surveillance [1]. However, *access* alone is insufficient for sovereignty, when sensitive and valuable data are stored outside the user's direct control. To be more independent, the individual needs a place to store the data that is rightfully theirs and execute the logic that serves and protects their needs. The term 'territory' is drawn from the *state sovereignty* analogy as it is the tangible subject of authority.

Ownership of a smart phone or other device is a key assumption that virtually every Self-Sovereign Identity project makes. It provides a space to store (part of) cryptographic keys, private data, receipts and other *evidence*. It also enables execution of logic for signing, verifying and data sharing, all on a medium that is physically in possession of the individual. This fundamental building block is also the Achilles' heel of digital sovereignty. If the phone breaks or is lost, the individual loses all their possessions. The accumulation of all this value also makes it an attractive target for theft, hacking or pre-installed back doors. Furthermore, by requiring individuals to wear a smart phone at all times, it may expose people to even more surveillance [28].

3.6.2 Principle 2. Usability

If the individual cannot operate the machine, they cannot perform any digital activities. Hence, in order to include the visually or otherwise impaired, basic accessibility mechanisms must be in place such as specified in the W3 Web Content Accessibility Guidelines [29]. However, *usability* goes beyond the human-computer interface, as the PGP usability study showed that lack of a basic conceptual understanding could compromise the entire purpose of the software [3]. Especially when the individual deals with many actors that all offer their own user interface, simple tasks can become daunting [12]. If the individual does not understand what is going on, cannot reason about it, or is unable to provide the appropriate inputs, they cannot be sovereign. In those cases where an individual simply cannot use digital systems themselves, they must be able to delegate these responsibilities. Especially for this fundamental aspect of *usability*, care must be taken when relying on third parties, as Brolchain et al. argue:

> "[I]nsofar as companies or governments monopolise how and what information is presented to users online, they will have a great deal of influence over how people perceive and interpret the world." [22]

3.6.3 Principle 3. Provable Identity

The individual must have the exclusive ability to prove (1) that they belongs to a *context-unique* identity and/or (2) that they are member of a *group*. This allows them to claim entitlements belonging solely to themselves (such as credentials, currency) or to a particular group (such as discounts for students, premium music) [30]. If possible, the system should favour or default to group membership as it allows the individual to *hide* within the group (or *anonymity set*) which is beneficial to privacy. The *exclusivity* expressed here relates to two concerns: *identity fraud* (no one can falsely claim identity or membership) and *surveillance* (no external party can *infer* identity or membership).

Provable identity is literally and figuratively the *key* to one's online existence: it opens the gates to all *value* that one could access online. It is therefore worrisome that many of our identifiers are *loaned* from identity providers; e.g. phone numbers, e-mail addresses, usernames. Users are required to share their secrets (such as passwords, biometrics, passport copy or hardware tokens) with identity providers to lay claim to these identifiers. This dependency makes them vulnerable in several ways: users may be *banned* from social media platforms when they do not comply with their policies, and may even be tracked when browsing totally different websites [31]. The impact of this dependency is enlarged by identity federation schemes such as *Login with*

Facebook [5]. At the very least, users need sole ownership over their keys and independent mechanisms for authentication.

3.6.4 Principle 4. Asset Access

Currently, many individuals' data are stored in large data warehouses, silos that they cannot access or only *peek* into. Article 15 of the EU General Data Protection Regulation states that a user has the right to access "personal data" [32]. In accordance, Allen's principle of Access states that the individual must always have access to the *data* that concerns them [1]. It is necessary to generalise this principle to an individual's 'assets' which also includes currency and other property. These assets should be available in such a way that data can move freely without losings its value (Allen's principle of Portability [1]), which may be accomplished by using digital signatures, for example. This allows the individual to move or copy it to their own *territory* for protection (similar to why one would save a purchase receipt), and/or reuse them in other situations.

3.6.5 Principle 5. Transparency

Insight in what has happened, is happening and what is about to happen provides the individual with the necessary knowledge to act autonomously. This aligns with Allen's principle of Transparency [1]. If the laws that govern the system are not transparent, the individual cannot see nor predict outcomes. Increasing transparency could hence improve trust. It may, however, also conflict with privacy.

3.6.6 Principle 6. Justice

Each context has its own rules, its own idea of justice which involve concepts such as *entitlement*, *ownership*, *liability*, *responsibility*; e.g. 'money cannot be spent twice'. For an individual to operate in those contexts, it must be assured that justice is served in one way or another. This may be built into technologies like the consensus algorithms 'governing' distributed ledgers, or brought about by trusted authorities. In any case, it is important to think about what *justice* is in the context of interest, how this is enforced and what happens if rules are violated. This is, in a sense, a context-specific form of security.

Unifying systems and contexts on this principle is likely to be the most difficult, as it comes back to unifying conflicting ideas of justice. However, rules will always be baked into systems. These should rather be made explicit beforehand than let the biggest company decide what justice is [33].

3.6.7 Principle 7. Consent and Control

To be fully sovereign, an individual must be able to exert influence over those matters that they have authority over [1], [12]. This starts with asking the individual for consent, but can extend to providing controls. Note that these depend on all the underlying principles: proper consent requires the user to be capable (principle 5) and well-informed (principles 6 and 7) to make the decision at hand [32]. Moreover, it should not be easily forged, and the consent should be enforced.

The individual can only be truly autonomous if they are in control over the relevant matter. Instead of being asked if the they are "OK with taking a left turn", they can now take the wheel themselves.

3.6.8 Principle 8. Convenience

When the user is in control, it may be burdened with a lot of effort: with great power comes great responsibility. Hence, if the individual is free to automate this, integrate it with other systems or delegate it, the burden might be relieved and full autonomy can be established.

3.7 Dimensions of the Problem Space

A prominent characteristic of the self-sovereign identity movement is the ambition to be universal. Instead of use-case specific solutions, it aims for an infrastructure that works across contexts, countries and cultures but also across time and all of human diversity. As argued in the introduction of this thesis, an incremental approach is taken by limiting the problem space. Future consolidation of such focused efforts requires careful coordination, which has motivated the creation of the theoretical framework presented in this chapter. To explore and scope the problem space, it is decomposed into three *dimensions* – Context, Humans and Time – illustrated in Figure 3.2.

3.7.1 Dimension of Context (Interoperability)

Allen' Principle of Interoperability [1] states that an individual should be able to move effortlessly between contexts whilst preserving their identity. That does not imply that all (or any) aspects of their identity need to be exposed in that context, but merely that they *could* prove all of who they are in any situation. Kaliya Young proposes to further divide this context (or *domains* in her terminology) in 16 Domains of Identity [11]. Most notably, Young partitions based on three archetypical power relations – employee to employer,

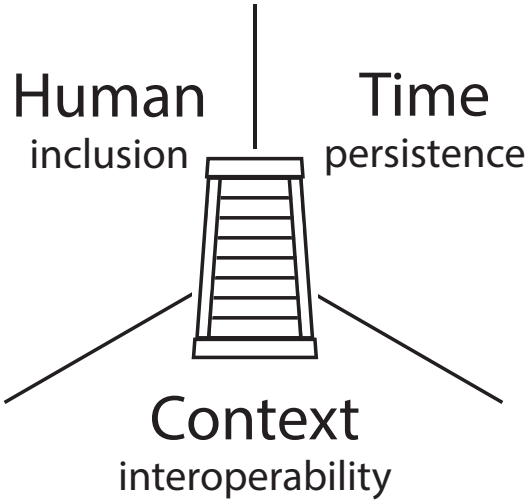


Figure 3.2. Dimensions for Digital Sovereignty Problem Space.

citizen to government and consumer to merchant – as these show very distinct power relationships and characteristics.

3.7.2 Dimension of Time (Persistence)

Allen's Principle of Persistence states that identities should be long lived, at least as long as their user desires [1]. This framework considers time a dimension that is relevant in more qualities of sovereignty, not limited to identity.

3.7.3 Dimension of Humans (Inclusion)

Finding a group of test users that accurately represents the whole of mankind is likely an infeasible task. When developing these systems, one must at least be conscious of the subset of humans they design for, so fall backs and alternatives can be installed for people without smartphones, without fingerprints or even fingers, without education, etc.

4

Cryptographic Layer

This chapter analyses the infrastructure used for building an identity solution – the TrustChain Stack [10] – describing its functionalities, the guarantees it provides and the input it requires. Also its applicability to the identity problem at hand is reviewed. The TrustChain stack consists of:

- IPv8, a peer to peer networking library [34].
- The TrustChain protocol, a personalised distributed ledger [35].
- An application layer that exposes a REST API for identity operations [36].
- A claim model for self-sovereign identity [37].

IPv8 is a network overlay in which peers, distinguished by public keys, can directly set up encrypted communication with other peers by addressing their public key. In contrast to the Internet Protocol [38], which facilitates device-to-device communication, IPv8 allows one peer to directly address another peer, regardless of the device they are operating. This is a much better fit for human to human communication and the digital identity problem.

Its modular architecture ensures that IPv8 can evolve incrementally. It decomposes functionality into different network overlays, also known as communities. These communities are simply groups of peers running the same protocol to accomplish some goal. If they wish to adopt new functionality, peers only have to join another network overlay. This creates an ecosystem that is self-regulating and self-evolving. Three primary functionalities in this networking layer are of interest to the identity problem at hand:

1. Establishing a connection between any two peers.
2. Attesting to a peer's identity claim.
3. Verifying a peer's identity attestation.

Next sections discuss each of these functionalities. Section 4.4 describes the accounting mechanism that IPv8 uses for identity attestations: the TrustChain protocol. As the purpose of this layer is to create trust in identity claims, it must specifically combat *identity fraud*. This is discussed in section 4.5, and section 4.6 specifically addresses the *information withholding* attack. Section 4.7 exposes a vulnerability of this layer when applied to real world identity use cases: metadata leakage.

4.1 Peer to Peer Communication

To set up a connection with a peer, its network address must first be found. The IPv8 overlay *DiscoveryCommunity* executes the peer discovery protocol. All peers together operate a Distributed Hash Table that maps a peers public key to its current network address. It employs *gossiping* to spread this information and also propagate connection requests through the network to ensure that any peer can *ask around for* any other peer, with the help of other peers in the network. IPv8 also solves an additional complication, traversing Network Address Translation (NAT) by employing NAT puncturing. This enables smartphone-to-smartphone communication.

So, setting up communication requires a peer id (a public key or a hash of it). By gossiping this request to other peers in the network, they learn which peers are establishing a connection, which may be a risk to privacy, especially if the real identity of those peers is known.

4.2 Attesting to a Claim

IPv8 distinguishes between *attributes* and *attestations*: a peer *claims* that some statement about his attributes is true. Other peers may provide an *attestation* to that claim, thereby vouching for its truthfulness. The relying party, also known as the *Verifier*, can use these attestations to gain confidence in the truthfulness, or *veracity*, of the claim.

This model supports both *authoritative qualification*, as well as *peer qualification* or a combination thereof. In authoritative qualification only the attestations of particular peers are trusted, such as Certificate Authorities in X.509 Public Key Infrastructures [39]. In contrast, peer qualification networks such as the PGP Web of Trust model [2] allow any peer to attest to an attribute. Trust is computed based on certain metrics such as number of attestations. This thesis focuses on authoritative qualification, because it is present in many of the modern day identity use cases.

Field	Description
Name	The name of the attribute
Timestamp	The time of claim creation
Validity term	The time after which the claim is no longer valid
Proof format	The type of proof for the claim
Proof link	The strong link to the proof for the claim

Table 4.1. Claim Metadata[37].

The IPv8 overlay *AttestationCommunity* provides the protocol for creating and verifying attestations. IPv8 considers attestations a transaction between two parties: the attester, or issuer, and the attestee, or subject. Both parties sign the transaction, which ensures that no identity operation will occur outside the control (and consent) of the subject, but also not without control of the attesting peer. TrustChain explicitly excludes mono-signature support: "TrustChain is based on the assumption that both parties agree on the transaction before signing it, making tampering inherently easy to detect." [10] These signatures offer three qualities: integrity, authenticity and non-repudiation.

Table 4.1 shows the data that is included in the attestation. As with any transaction, attestation blocks are considered public in IPv8. Section 4.7 will discuss the privacy implications of this design choice.

Transactions are captured in a personalised block-chain-like data structure, called TrustChain. Each transaction is captured in a *block* that contains the content of the transaction, the two signatures and also a reference to the preceding block of each party. This creates a personal chain that is entangled with other personal chains. Section 4.4 describes this in more detail.

4.3 Verifying an Attestation

The Verifier may obtain the attestation by requesting it, or through gossiping or crawling of blocks that occurs in TrustChain. The Verifier can check the signature of the transaction to ensure that the attestation was not tampered with and was made by a trusted peer. He can check that the remaining metadata such as timestamp, expiration and attribute name meet his requirements. Note that this can be done without the subject's knowing, let alone control.

The content of the attestation can be anything, such as the plain value of an attribute. However, for any attribute that is considered personal data, this should not be publicly available. For this reason, IPv8 has built-in support for interactive Zero-Knowledge Proofs (ZKPs).

4.3.1 Zero-Knowledge Proofs

In general a Zero-Knowledge Proof is a method whereby one party, the *prover*, can prove to another party, the *verifier*, that he knows some value without revealing that value. In an interactive Zero-Knowledge Proof, the verifier sends a number of challenges to the prover. Only if the prover knows the particular value he claims to know, he can win all challenges and send back a response confirming that.

Instead of a plain attribute value, a ZKP *statement* is provided in the attestation. This statement contains the assertion that the subject *knows* the value, but it does not contain the value itself. Because no private data can be derived from the attestation itself, it can safely be shared with the verifier. As the statement was created together with the *attester*, the verifier can rely on the value to be truthful. Several types of Zero-Knowledge Proofs are possible, the simplest of which is *equality* [40]; the proof can only succeed if the attribute has a specific given value (e.g. *city* equals *Delft*). In contrast, a *range proof* only succeeds if the attribute value lies within a particular range (e.g. *age* is at least *18*) [41] and a *set-membership proof* proves that the attribute value is a member of some set (e.g. *country* in *list of European countries*) [42].

4.3.2 Sharing knowledge obtained in Zero-Knowledge

A successful proof convinces the Verifier that some attribute meets certain constraints (equality, inequality, set-membership, etc.) assured by the attester. However, due to the *Zero-Knowledge* property of Zero-Knowledge Proofs, the Verifier cannot trivially convince any other party of this fact. The intuition is that any party can generate a transcript of a zero-knowledge proof (a series of challenges and responses) that is indistinguishable from a real transcript. Hence, the Verifier cannot convince a third party that the subject successfully proved something. This also makes it impossible for the Verifier to prove to an auditor that it properly verified someone's identity, which may be required by Know Your Customer and Anti-Money Laundering regulations.

In practice however, peers may still trust the verifying party with being honest about the interaction. This is both an opportunity and a threat. The threat being the leakage of private data. For example, to a company sharing private data with advertisement companies, there is no need to actually *prove* the data is correct. It has incentive to be honest about this data because lying could harm the business relationship with the advertiser. The use of Zero-Knowledge proofs must not provide a false sense of privacy, so sharing (or *proving*) of private data should still be minimised.

Existing trust in a verifier can also be considered an opportunity, in fact

minimising leakage of private data by introducing a trusted middleman. By verifying one or more of the subject's attributes in zero-knowledge, the verifier can *validate* that the subject meets some set of criteria. For example, he can verify the attestation from a university that he is a student. He can then issue an *is_student* attestation. Because this attestation comes from a third party, it cannot be directly linked to the university, so the subject gains privacy.

4.4 TrustChain, a personalised ledger

As mentioned in section 4.2, attestations are considered transactions, which are all recorded in a distributed ledger. Specifically, IPv8 relies on a personalised ledger called TrustChain. In other common block-chains, such as Bitcoin [43] and Ethereum [44], peers share a single ledger. Peers hold a full or partial copy of the ledger and over 50% of peers must accept a transaction before it will be appended to the ledger. This is known as global consensus. This feature has dramatic implications for the ledger's speed. Bitcoin's theoretical limit is seven transactions per second.

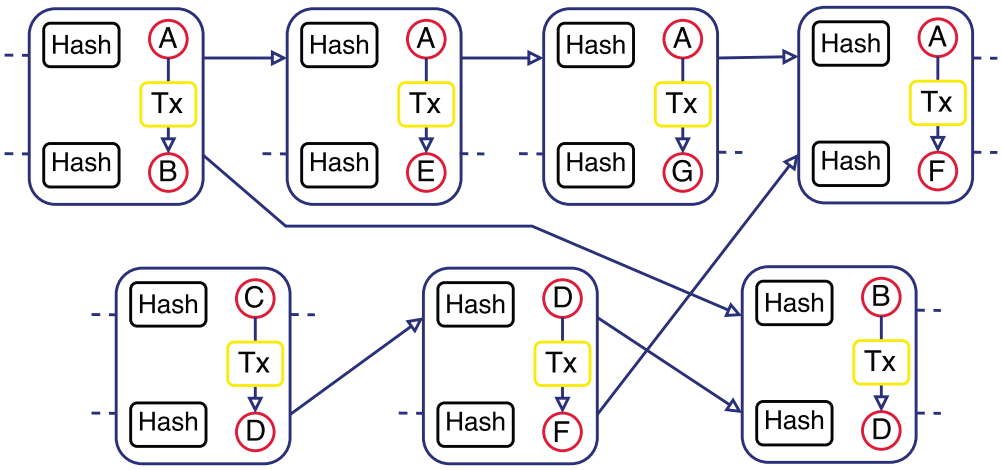


Figure 4.1. The tamper-proof TrustChain data structure to record transactions [35].

TrustChain lacks global consensus, which provides it with superior horizontal scalability. As a consequence, transactions on TrustChain represent local consensus between two actors at a particular time. Figure 4.1 shows TrustChain. Because each block holds a pointer to the previous block of both parties, their chains become entangled. In principle, blocks are only stored by the peers that were involved in those transactions. This maximises horizontal scalability in terms of storage. However, this also allows peers to withhold transactions. This is known as the information-withholding attack or,

from cryptocurrency jargon, *double spending attack*, since hiding a payment could allow a peer to spend that money again. By design, TrustChain is not tamper-proof but tamper-evident. The peer can withhold information, but as the same information is also present at another peer, this may show up later and provide evidence of tampering.

Like many block-chains, TrustChain was developed with economic transactions in mind. It originated from a file sharing project called Tribler. For economic transactions of fungible goods such as bandwidth, money or other economic assets, this can work well. The question is whether TrustChain can be applied to identity information in the same way.

4.5 Identity Fraud

IPv8 uses the TrustChain to "provide the transparency and persistence needed for audit logs and the legal status of the identity system." [37]. Its goal is to expose possible identity fraud. This section examines possible cases of identity fraud and discusses how each of these frauds can be combated.

Consider a scenario where Alice proves to Bob of some aspect about her identity by presenting signed claims made by Chris. In this scenario, *identity fraud* refers to Alice's attempt to convince Bob of a *truth* that is not, or no longer, true from Bob's perspective. *Truthfulness* is considered a subjective property; in the eyes of Bob. Bob relies upon Chris to determine truthfulness of Alice's claim. The following scenarios can be distinguished:

1. Alice convinces Chris to attest something that is not true.
2. Alice forges an attestation by Chris without his involvement.
3. Alice manipulates an attestation by Chris to a different meaning.
4. Alice uses an existing attestation that does not belong to her.
5. Alice hides an attestation, pretending to Bob it does not exist.

Each of these attacks are discussed briefly:

4.5.1 Issuer Scamming

In the first attack, Alice may fool Chris into thinking that she is someone else, or has certain qualifications (e.g. by cheating on an exam). It is up to Chris to establish a level of certainty in the truthfulness of the claim, before making an attestation. Ideally, he communicates his level of certainty to Bob, so the risks of relying upon the attestation are known. This is done in governance or trust frameworks such as the European regulation *e-IDAS* for *Electronic IDentification Authentication and trust Services* [45] and *Digital Identity Guidelines* by the National Institute of Standards and Technology [46].

4.5.2 Attestation Forging

The second attack is mitigated by the use of digital signatures, particularly the authenticity property. In order to forge an attestation, Alice must either break the encryption or steal the private key of Chris. It is up to Chris to protect his private key.

4.5.3 Attestation Tampering

The third attack is also mitigated by use of digital signatures, particularly their integrity property. As the transaction including metadata is signed by Chris, the content of the transaction cannot be changed without Bob being able to notice this. So again, Alice must either break the encryption or steal the private key of Chris.

4.5.4 Identity Binding

The fourth attack relates to identity binding. Suppose Chris provided another peer Dave with a valid attestation, the attack is such that Alice convinces Bob that this attestation applies to her. By design, each identity transaction is bound to its subject by mention of its public key. If Alice could somehow obtain Dave's private key, she can pretend to be Dave, committing identity theft. Dave could also share his identity with her on purpose; consensual impersonation or identity sharing. Finally, Alice could perform a man-in-the-middle attack, convincing Dave to execute the proof to Alice whilst Alice forwards his messages to Bob. These attacks are addressed in the semantic layer.

4.5.5 Information Withholding

The fifth attack is discussed in more detail in the next section.

4.6 Information Withholding

The final attack relates to information withholding (also known as double spending). Recall that by the principles of self-sovereign identity, Alice is in *control* of her data and hence decides whether she shares information or not [1]. In that sense, information withholding would be intentional functionality; a feature, not a bug. This limits the applicability of self-sovereign identity to all those transactions over which Alice has authority. Therefore the attack is specified to "pretending to Bob it does not exist". In general, Alice would hide information when that is beneficial to her, i.e. *negative information*.

4.6.1 Absence of information

A common practical use case of negative information is that of a criminal record. Employers wish to know whether a new applicant has not committed any (relevant) crimes. The applicant is the authority in this information sharing scenario, yet how can he prove the absence of such information? In the Netherlands, the ministry of Justice and Security provides citizens with a *certificate of good behaviour* ("verklaring omtrent het gedrag") which shows that the state is unaware of any relevant criminal records of the subject. So, one method is having a third party attest to the absence of some information.

4.6.2 Updates and Revocations

Another likely scenario is when an attribute value changes. Attestations issued on outdated values of the attribute should no longer be considered valid. In some cases, the actuality of information is essential: a fired employee may abuse their credentials to harm the employer. By the SSI principle of control, the attester cannot force the subject to agree to the new information. Even if the subject *did* sign the information, he can still withhold this from a verifier. Also, the SSI philosophy does not allow the verifier to directly contact the attester, or vice versa, to check if the information is still valid.

Stokkink and Pouwelse offer two ways to support revocation on personalised ledgers such as TrustChain [37]. They distinguish two models: *intent-based* and *active*. The intent-based model records the verification intent as a new transaction on the TrustChain. Suppose the used attestation was made at time T_i, and verified at time T_v, the verifier and subject sign a transaction pointing to the original attestation. If at a later audit it turns out there was a revocation block between T_i and T_v, then this shows the subject has withheld relevant information, hence committed identity fraud.

Because certain use cases need immediate certainty, they cannot rely on later audits to reveal fraud. This problem is tackled by the *active* model, in which the subject simply asks the attester for an updated attestation. It uses a unique challenge given by the Verifier to ensure that the subject cannot replay the signature from an earlier attestation. Note that the attester must be online during this transaction.

4.7 Meta-data Leakage

Transactions in TrustChain are public by design, not only for auditability but also for availability. Peers may hold copies of other blocks they are not involved in to improve the availability of data in case peers go offline. This makes sense

in case of file sharing communities which use transaction history to allocate bandwidth, i.e. don't share files with peers that only leech instead of seed. It would be infeasible to require all peers to be online for such algorithms to operate properly. Yet in the case of digital identity, this argument for availability only holds for public attributes that should be verifiable without involvement of the subject. The primary use case of self-sovereign identity lies in those presentations in which the identity owner actively controls and consents to the sharing, and ensures minimal disclosure to only specific parties.

Does the public nature of attestation metadata conflict with the privacy constraints imposed on the identity problem? Table 4.2 shows an example set of transactions that belong to the same subject. This includes the metadata as explained in Figure 4.1 and the public key of the issuer as shown in Figure 4.1, and also includes revocation blocks and intent-based blocks (as explained in section 4.6.2). Another column mentions the real-world identity of the issuer. The attestation value, or *proof link*, is excluded because it offers no meaningful information here.

Timestamp	**Attribute Name**	**Issuer**
Aug 1, 2010	eligible_to_drive	KRs42.. (Centraal Bureau Rijvaardigheid)
Aug 15, 2010	is_resident	XdaST.. (Delft, Netherlands)
Sep 1, 2010	is_student	AT1da.. (Delft University of Technology)
Sep 12, 2010	has_bank_account	12opT.. (ING Bank)
Dec 31, 2010	insured	IEQW9.. (Insurance4Students)
March 13, 2013	owns_company	YY21x.. (Kamer van Koophandel)
Aug 31, 2013	has_diploma	AT1da.. (Delft University of Technology)
Nov 5, 2013	is_employee	RQ231.. (Creative Media Company)
Dec 31, 2014	insured	pRI12.. (Healthy Insurance)
Feb 9, 2016	is_employee	hK55x.. (Trucking Company)
Jul 7, 2020	has_diploma	AT1da..(Delft University of Technology)
March 1, 2024	is_father_of	RYsW3.. (Rotterdam, Netherlands)

Table 4.2. Example of public metadata of single peer.

Transactions do not include the real world identity of the issuer by default, only its public key. However, for verifiers to be able to trust that the attestation came from a particular source, the (authoritative) source they trust, the link between real world identity and public key must be publicly known. It is hence safe to assume that in the current design of IPv8, all metadata listed in Table 4.2 is public. Whilst the values of the attributes remain hidden, it still reads as a pretty detailed biography. Even without knowing the real world identities of issuers, sensitive information would be revealed. This is a major privacy concern.

4.7.1 Metadata Confidentiality

However, attestation metadata cannot be fully considered private either. The Verifier requires the metadata of the attestation he is to verify. Without it he cannot understand it, nor ensure he trusts the attester. Once this information is disclosed to the Verifier, the system has no way to prevent the Verifier from spreading this information. From this perspective, the creators of IPv8 reason that metadata should not contain any sensitive information and hence be considered public.

As mentioned before, for many meaningful identity applications, the metadata inevitably contains sensitive information. The security model can therefore be expanded to include the Verifier. The subject must trust, or enforce through external measures, that the Verifier will not forward this information to other parties without consent of the subject. Modern data protection regulations already restrict the Verifier's capability to share information. The risk of penalties motivates issuers to comply. If the Verifier is susceptible to these external forces, the attestation information can be considered confidential and be shared only with the verifiers that need it.

5

Authorisation by Legal Entities

This chapter studies the class of use cases related to *Legal Entity Representation*, using the theoretical self-sovereignty framework presented in chapter 3 and maps these to the cryptographic foundation presented in chapter 4. This results in a distributed authorisation management application that enables to distribute power within and across organisations. This study and accompanying design project were performed in collaboration with the Kamer van Koophandel.

5.1 Legal Perspective

This section discusses from a legal perspective under what conditions a person is considered to have *power* over a legal entity, within the context of the Dutch legal system. Derived from legal texts and expert interviews, it considers the following types of power:

1. Registered directors (*functionarissen*) who control a legal entity.
2. Registered Power of Attorney (*volmachten*) granting a natural person full or partial power over a legal entity.
3. Unregistered Power of Attorney based on a shared platform.

5.1.1 Executive Power over Legal Entities

Depending on its form, a legal entity is controlled by one or more *natural* or *legal* persons, called *directors*. This relationship is registered in the trade register (*Handelsregister*), managed by the Chamber of Commerce [47]. Since one legal entity can control another, a hierarchy can be established which always has one or more natural persons as root. Laws, by-laws, and possibly other conditions, determine what power these persons have over each legal entity in the hierarchy. If a legal entity has more than one director, it may be the case that directors cannot individually sign contracts but instead must do so together [48].

To verify a natural person's executive power, one can download an excerpt (*uittreksel*) at the Chamber of Commerce website[1] at the cost of € 2,30 or € 7,50 for a digitally signed certificate [49]. The verifier must then compare the full name and date of birth of the director, as stated on the excerpt, with some form of identification. Note that this applies only to the entities in direct control of the natural person. For entities further down in the hierarchy additional excerpts must be compared. This cumbersome procedure is what notaries offer as a service.

5.1.2 Extending Power over Legal Entities

Depending on their power, executives may choose to distribute power over their employees (vertical authorisation) or to other organisations (horizontal authorisation) by means of explicit or implicit *Power of Attorney* (*volmacht*).

Explicit Power of Attorney. The Chamber of Commerce explicitly registers Power of Attorney using a form[2]. It allows to either grant the subject *full* authority, or restricted by options shown in Table 5.1. To issue a Power of Attorney, the *grantor* (or an authorised representative) must fill out the form and visit one of the offices of the Chamber of Commerce together with the person being granted to submit it. Verification can be done in the same way as with executive control, using excerpts.

Implicit Power of Attorney. The Power of Attorney method is a tedious process both when filing it and when checking it. Suppose a customer checks out at the local grocery store. Before handing over the money, he must check if the cashier actually has Power of Attorney to receive that money. For such situations, Dutch law provides the concept of *implicit Power of Attorney* which

[1] https://www.kvk.nl/producten-bestellen/bedrijfsproducten-bestellen/uittreksels/
[2] Formulier 13 Inschrijving Gevolmachtigde [50]

Restriction Option	Description
By Financial Amount	Maximum amount in Euros
By Act	One or more of the following: *Requesting changes in the Trade Register, Issuing Quotations, Access RDW license plate services.*
By Contract Type	One or more of the following: *Purchase, Sales, Warranty, Lease (Rental), Financing, Software, Maintenance* and/or *a custom description*
By Establishment	Entire legal entity, or specified to a specific establishment (by address).
Other	Any other restriction defined in natural language[a].

Table 5.1. Restriction options for registered Power of Attorney.

[a] As this is inconvenient for automation purposes, the Kamer van Koophandel is developing a semantic model that can replace this free-form text.

means that if one can *reasonably assume* that Power of Attorney *was* granted, one may act on that assumption [51]. The assumption must be based on a *statement* or *behaviour* by the *grantor* – e.g. the store not removing the person from behind the cash register – and take *circumstances* into account.

Sharing Authority Information. The aforementioned methods of explicit and implicit Power of Attorney do not suit all transactions. Whereas the physical space allows customers to easily (yet superficially) assess the identity (face) and attributes (company clothing and placement) to support a *reasonable assumption*, the digital space is by default much less transparent. This has lead to alternative approaches, such as the Dutch eHerkenning system[3], a cross-organisational identity and access management platform. Digital service providers list their online products and services in the central catalogue, along with the *level of assurance* they require. Consuming organisations must purchase a subscription for each participating employee, at an annual cost of around € 5 to € 45 [52], depending on the level of assurance. Executives are granted full access and must divide access rights among their staff or delegate this responsibility. As an employee wishes to use a service, they must log in with their eHerkenning account at the right level of assurance, which was authorised specifically for that service. Since 2020, all organisations that file their taxes digitally are required to pay for an eHerkenning subscription, which has led to much confusion [53].

[3] https://www.eherkenning.nl/

5.2 The Theoretical Self-Sovereignty Framework applied

The goal of this study is to investigate how a TrustChain can be applied to the power over legal entities. The theoretical framework presented in chapter 3 is applied to scope this study.

The PURPOSE of the use case is proving that the subject is authorised to act on behalf of a legal entity. This is primarily of concern to three stakeholders: the subject, the legal entity and the verifier. Note that they need this assurance for their actual purpose: to safely perform the intended (trans)action.

The HUMAN central to this study can play two roles: (1) as the subject that represents the legal entity and (2) as the issuer of power to another subject. In both cases its identity information is of importance to the purpose. Also, in both cases the subject can be held accountable for their actions. This means the human subject and issuer also have a stake in successful authentication, for they may risk legal repercussions otherwise. Note the two involved aspects of the human's identity: a business aspect and a personal aspect. These aspects are as indivisible as the human.

5.2.1 Boundaries of Sovereignty

The BOUNDARIES OF SOVEREIGNTY define the extent of the human's control in these situations. This is delicate in the case of legal entity authorisation, due to several power asymmetries between actors [11]. The boundaries of sovereignty depend on the type of claims that are made and shared.

With respect to one's *power*, this use case shows several power asymmetries which are slightly different. No actor has the sovereignty to authorise oneself[4]. The *trade register* has the authority to grant and revoke a *director*'s authorisation but the directors entitlement to that control is prescribed by law. The *director* however has the freedom to authorise and revoke authorisation of any other employee. The employee does not have the sovereignty to further distribute this power, as this depends on the by-laws of the legal entity.

With respect to one's personal identity, it is not always clear what the boundaries are. This can be observed from a lawsuit by an employer claiming ownership over a departing employee's professional LinkedIn network [54]. The individual's personal identity is however used to authenticate them and hold them personally accountable for their actions. This means one can be personally affected, not only in terms of *privacy* but also in terms of legal repercussions. It is therefore imperative to protect the individual from both vulnerabilities, also against their employer.

[4] One could authorise themselves, but the verifier decides to accept or reject such claim.

5.2.2 Problem Space

Section 3.7 provides a structure to divide the problem space. It makes explicit which situations are covered by the system. In terms of the three dimensions:

- TIME: this depends on the situation, but authorisations may last years and audit logs may also need to be kept several years.
- CONTEXT: any physical or virtual business context that requires an actor to prove some power over a Dutch legal entity.
- HUMAN: any person subject to Dutch law that can understand the notion of authorisations, is capable of handling a smartphone[5] and has a personal smartphone[6].

5.3 Mapping to Pseudonyms and Attributes

This section proposes a mapping from the previously discussed legal perspective to the cryptographic layer presented in chapter 4.

5.3.1 Actors to Pseudonyms

Recall that the cryptographic layer represents actors by *pseudonyms* (key pairs). Which actors need to operate their own pseudonyms? The legal analysis identifies the following actors:

1. Natural persons: directors and (un)authorised personnel,
2. Legal persons registered in the trade register,
3. The Chamber of Commerce (also a legal entity),
4. Implicit: The person or system verifying someone's power.

Every legal entity can only act through natural persons, or systems installed and run by natural persons. Yet, if only natural persons can operate pseudonyms, then the King of the Netherlands would use his pseudonym to authorise government officials, who authorise other officials, who authorise directors, who authorise their staff. All Dutch citizens who trust the King can then verify the power of every person in his hierarchy. It makes sense to simplify this hierarchy by allowing legal entities to operate pseudonyms.

[5] Whilst a universal identity infrastructure must provide for those who are excluded by this definition, it is reasonable within the chosen context.

[6] This may be a weak assumption as employers sometimes provide their employees with phones. This violates the Digital Territory principle, as the phone is not under the individual's sovereign control. For simplicity of this analysis however, it is assumed that the person brings their own device

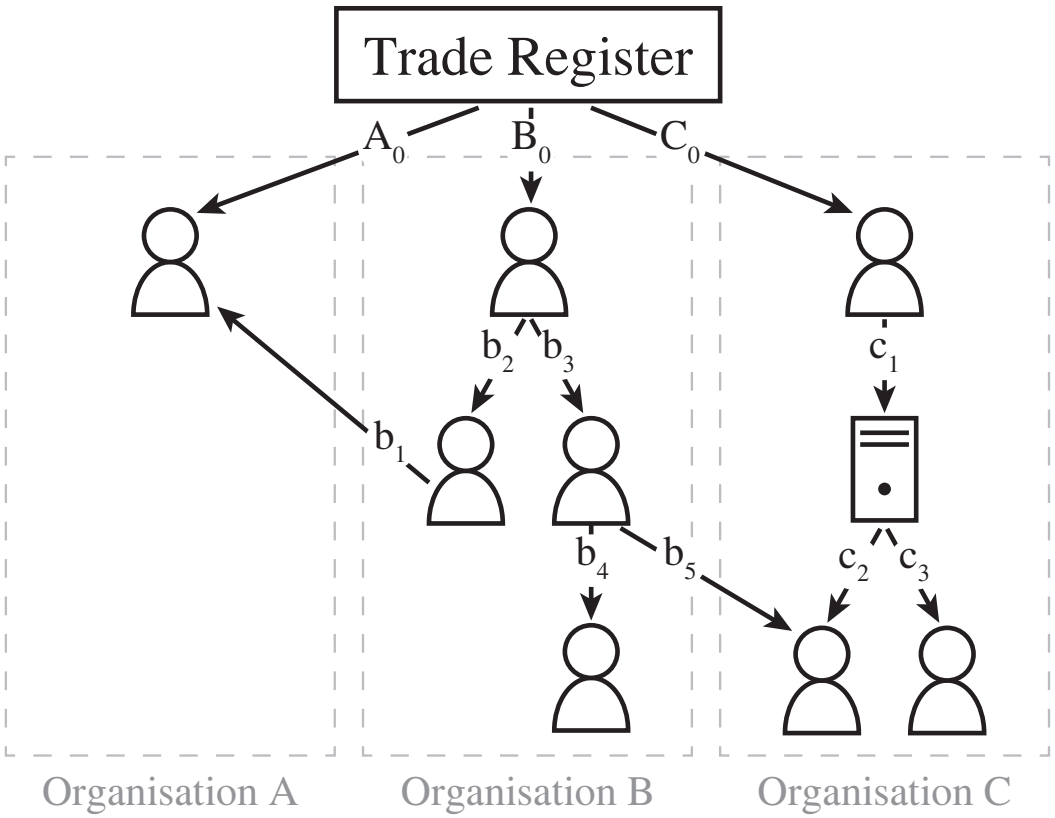


Figure 5.1. Network of Powers. Arrows indicate the assignment of executive power (A_0, B_0, C_0) or extended power $(b_1, b_2, c_1, ..)$.

5.3.2 Powers to Attributes

The cryptographic layer uses *attributes* to describe certain aspects of pseudonyms, and *attestations* to denote that someone *vouches* for that attribute. These constructs can be applied to model the three kinds of powers identified in the legal analysis: full power, registered power of attorney (PoA) and unregistered PoA. The first two powers can be described as $Full(L)$ and $PoA(L, J)$ as shown in Table 5.2. The KVK can attest to these attributes providing trustworthiness.

Attribute	**Meaning**	**Trusted Issuer**
$Full(L)$	subject has full control over a legal entity L.	Chamber of Commerce
$PoA(L, J)$	subject has registered Power of Attorney over L restricted to some jurisdiction J.	Chamber of Commerce
$Auth(L, J)$	subject is authorized to act in name of L within the boundaries of some jurisdiction J.	Any qualified person

Table 5.2. Attributes for Power over Legal Entities.

Directors have the freedom to propagate their power to their staff and other parties in any way they see fit. They may explicitly authorise a manager, or define some organisation-wide policy that determines under what conditions people have *access* to perform certain legal actions (e.g. purchase a car). This does require however that the verifier is aware of the policy and is able to evaluate it.

This problem closely relates to the field of Access Control, which encompasses the "protection of [information] system resources against unauthorised access" [55]. The four most common *models* of access control policy are mandatory (MAC), discretionary (DAC), role-based (RBAC) or attribute-based access control (ABAC) [56] of which ABAC is most expressive and can model all other models. These models could express powers in the following manner[7]:

1. **MAC:** only executives explicitly authorise a person.
2. **DAC:** authorised people can explicitly authorise at their discretion.
3. **RBAC:** only people with certain roles (e.g. managers) are authorised.
4. **ABAC:** only people with certain attributes (e.g. over 21) are authorised.

The RBAC and ABAC models require a policy to be communicated to and evaluated by the verifier, which makes the problem significantly more complex. Conversely, using MAC or DAC allows an unambiguous authorisation to be made from one actor to the next, without additional complexity[8]. Such

[7] RBAC and ABAC could even specify more complex Boolean logic.

[8] If RBAC or ABAC policies are needed anyway, an organisation can install a local policy evaluation mechanism that grants on-the-fly authorisations to staff when their roles or attributes satisfy the policy.

authorisation may be modelled as $Auth(L, J)$ (see Table 5.2), which specifies limitation to the power: the jurisdiction J (so the cleaner is allowed in, but not allowed to sell the building).

5.3.3 Power Ontology and (Partial) Ordering

The verifier needs to know: (1) what power is required and (2) whether the presented attributes provide this power. When this power is divisible and restrictable, thus ordered, an *ontology* and *(partial) ordering* are required that all participants agree on. Consider the ontologies used by the two solutions identified in the legal analysis, e-Herkenning and Registered PoA. Registered PoA breaks down powers into classes with an optional financial restriction; e.g. purchases up to 100.000 euros. e-Herkenning defines authorisation for each specific product and service that a participating organisation offers; e.g. filing quarterly taxes with the tax authorities. So Registered PoA defines a partial ordering and e-Herkenning specifies atomic powers. Whilst e-Herkenning allows more fine grained control, management of so many authorisations becomes very impractical and even dangerous: an employee is stalled as he awaits authorisation, so he borrows the identity from a colleague. Or his superiors simply grant him full access to be done with it[9].

It follows that the best ontology and ordering likely depends on the use case. To improve interoperability, this ontology may be parametric or defined industry-by-industry for example. The specifics are out of the scope of this thesis.

5.3.4 Power Evaluation

Recall from the problem description that – given a subject S, power P and legal entity L – the following assertion must be evaluated:

$$Q(S, P, L) = \text{“subject } S \text{ is has the power } P \text{ over entity } L\text{”} \tag{5.1}$$

The chosen power ontology defines the power P, and the function $\geq$ is defined in a (partial) ordering, hence

$$(Q(S, P', L) \wedge P' \geq P) \rightarrow Q(S, P, L) \tag{5.2}$$

Given this ontology, a set of attributes exists that assigns a power P over a legal entity L to a party S_1, given another party S_2. Let the predicate

$$Pow(S_1, P, L, S_2) = \text{“}S_2 \text{ assigned power } P \text{ over } L \text{ to } S_1\text{”} \tag{5.3}$$

[9] These problems were identified in the KVK Workshop, see section B.1.1

However, this assignment of power is only valid when party S_2 itself has this (or higher) power. Hence the power transition can be formulated as

$$(Pow(S_1, P, L, S_2) \wedge Q(S_2, P', L) \wedge P' \geq P) \rightarrow Q(S_1, P, L) \tag{5.4}$$

This shows the problem is recursive. A base case can be provided if a party is chosen as a trusted root. This requires the following predicate:

$$Trust(S, P, L) = \text{“}S \text{ is trusted to have power } P \text{ over } L\text{”} \tag{5.5}$$

In conclusion, to evaluate the power of a subject S_1, one needs:

- a set of attributes that satisfy $Pow(S_i, P_i, L, S_{i+1})$ for every $i \in \mathbb{N}_{<n}$,
- a trusted root S_n for which hold $Trust(S_n, P_n, L)$ and
- to ensure that all $P_i \leq P_{i+1}$ for all i.

5.4 Verifying Integrity: Trust Model

The evaluation of one's power is only valuable when the integrity of the data can be guaranteed. A verifier can trust[10] the outcome if and only if he trusts that:

1. each participant only attests correct information.
2. each participant only attests to correct pseudonyms.
3. each participant revokes attestations as soon as they are invalid.
4. each participant had sole control over its pseudonym when attesting.
5. the original owner controls the subject pseudonym when verifying.
6. attributes cannot be forged or tampered with.
7. attributes are not outdated.

The first three conditions are a matter of trust based on the assumption that it is in participants' best interest to ensure accurate data. The fourth and fifth requirement are concerns of identity binding as discussed in section 4.5.4. If possession of the private key is considered insufficient authentication, additional methods are necessary to authenticate the participants. The sixth condition is ensured by the cryptographic layer. The seventh condition requires more attention, and will be addressed in section 5.4.2.

[10] Note that whilst the verifier may not in all cases be held liable if these rules are broken, he might still be harmed.

5.4.1 Full-chain verification

The cryptographic layer (currently) only performs interactive verification with subjects, using interactive zero-knowledge proofs. However, the verifier must also verify attestations made to the other participants in the chain who are not actively involved during the verification session. Three alternatives are considered:

1. **Interactive**. The verifier interacts with each participant in the chain. This requires all participants to be online, consent to verification and partake in the interactive zero-knowledge proof.
2. **Passive**. All participants published their attestations to some public storage. The verifier accesses this storage to verify the signature of each attribute. The contents must be public or part of a non-interactive zero-knowledge proof.
3. **Proxy**. All participants forward the chain of attributes every time they authorise a person. The subject can then present all the attributes in the chain, which the verifier can then verify.

5.4.2 Actuality: a Requirement

Whilst by SSI attestations are intended to be used for a longer period, the KVK requires that relying parties should always retrieve the latest information[11]. Ownership of companies changes regularly, and it is imperative that these changes propagate through the power hierarchy as fast as possible, especially considering disgruntled employees may pose a threat to their organisation. Revocation mechanisms of the cryptographic layer were discussed in section 4.6.2. At present, this only offers the option in which a subject fetches a *fresh* attestation. This is problematic in several ways:

1. It creates a single point of failure, the KVK, for all Dutch organisations.
2. It impacts privacy as it leaks times of activity to the KVK.
3. For full-chain verification it requires all participants to be interactive.

What is required is a public register that enables the issuer to revoke credentials at any time, yet without knowing which party checked the revocation register. The creation of such a mechanism is out of the scope of this thesis.

[11] Interviews with KVK staff brought forward that they *intend* for parties to use the most recent KVK information. Hence, certificates do not state an expiration date. However, as this cannot be enforced, many certificates (mainly paper or PDF) are re-used a long time after issuing. A possible cause is the cost and effort of retrieving new versions on every use. A similar concern was raised by Dutch notaries when discussing their possible role as SSI attestors. Their *legal opinion* is the product of a (manual) assessment of several records, which they consider expired after a day, sometimes even hours.

5.5 Issuing the root Legal Entity Certificate

An attribute that states that a subject controls a legal entity is only valuable when a trusted party attests to it. Only the trade register defines the *truth* regarding this relation between natural and legal persons. It would therefore make sense for the KVK to attest to these attributes in an SSI ecosystem. However, the collaboration with the KVK has shown this is not so trivial.

This debate is an important one, as many *sources of truth* may face similar challenges when integrating with an SSI ecosystem. These challenges may motivate the involvement of 'trusted' intermediates (or brokers) who remove the risk and hassle of connecting with the system directly, creating economies of scale which lead to monopolies. As discussed in the Value of Independence (section 3.4), this poses serious threats to the control and privacy of individuals. The considerations are outlined below.

5.5.1 Legal Considerations

The first consideration is whether the KVK has *legal ground* to disclose information to a self-sovereign identity system, as rights and responsibilities of the KVK are prescribed by the law. Whether the law does, or should, provide for this is however out of the scope of this thesis.

Assuming a positive answer, there is another pressing legal issue. In the cryptographic layer, an attestation is directly bound to a pseudonym (by referring to its public key), not to a natural person. By attesting to the pseudonym, the KVK expresses a *belief* that the pseudonym belongs to the person, whereas the KVK should only state *facts*. Note that (commercial) intermediaries such as e-Herkenning are not bound to this legal restriction.

To circumvent this problem, the attribute could be made *conditional* on another *authentication* attribute. Only after the verifier checks has authenticated the subject, may the attribute be used. This moves the authentication responsibility to the verifier. This would be possible with the passport-grade biometric authentication that has been developed in earlier TrustChain experiments. Further legal investigation is needed to ensure that this solution satisfies the law.

5.5.2 Economic Considerations

If the KVK offers attestations as a service, their sales of regular certificates and API requests will likely be affected dramatically. With roughly 3 million registered legal entities, the KVK sold almost 11 million certificates and 27

million API requests in 2018 [57] which would amount to € 90 million[12]. Sales of products and services cover about 45% of the KVK's expenses, making it a very important source of income. This naturally raises the question whether attestations should be free of charge or not. Answering this question is however out of the scope of this thesis.

5.5.3 Technical Considerations

For privacy reasons the trade register is not sorted on, indexed by, or searched by natural persons[13]. This complicates the attestation process as it is not sufficient for the subject to simply identify itself, they must also specify the KVK number of the legal entity that they request an attestation for.

5.5.4 Security and Privacy Considerations

The KVK has expressed concerns that their register may be copied in its entirety and distributed via other so-called *shadow registers* (such as CompanyInfo[14]) that either offer cheaper or free access or additional services on this data. This is not only a threat to the KVK's income, but also a threat to the data quality and actuality, and may enable searching by natural persons.

5.6 Peer to Peer Authorisation Management

Once Executive Power has been attested as an attribute, it can be used by executives to prove their power over an organisation. This is likely sufficient for the over one million (2020) Dutch self-employed entrepreneurs [59]. The remainder of businesses however, almost half a million (2020) with more than one employee [59], may need to delegate their power to staff members or people outside their organisation. The Kamer van Koophandel has expressed interest in a system that allows people to distribute power in a simple manner with high interoperability across organisational boundaries out of the box (see appendix B). This led to a proof of concept for a peer-to-peer mobile application called Zekere Zaken (Secure Business) designed to run on top of the self-sovereign identity infrastructure developed in this thesis. The design considerations are presented in this section, and the proof-of-concept is presented in chapter 8.

[12] Assuming a price of € 2,30 each [49]. Yet, this is likely an over-estimation. The total proceeds from operational activities amounted to 92,4 million in 2018 [57], but this also includes 273.078 new enrolments at € 50 yielding € 13.6 million. Other sources estimate the proceeds at € 45 million (2015) [58].

[13] Handelsregisterwet 2007 [47], article 22:2

[14] `https://companyinfo.nl/kvk-uittreksel/`

Note that the purpose of this design study is not to create a perfect application, but rather to explore implications on the underlying infrastructure.

5.6.1 The ZZ app

All tasks of authorisation management and verification can be performed from one application. The app specifically offers the following functionalities:

- Verify someone's power, or be verified.
- Share power with someone, or receive power.
- Inspect and revoke given powers.
- Inspect and remove received powers.
- Request executive power from the Kamer van Koophandel.

To provide these functionalities, the application needs at least the following components:

- The ontology and (partial) ordering of powers, to enable construction and verification of attributes.
- A mechanism for power evaluation.
- A known root (e.g. KVK), or functionality to pick roots.
- A peer-to-peer communication mechanism.

Whilst it is possible for apps to communicate via a central server, this harms the independence and privacy of actors as it introduces the need for a trusted third party. This application will therefore rely on the communication functionalities offered by the cryptographic layer.

5.6.2 Separation of apps

Whilst it is technically possible to merge this app with the Wallet application, there are many arguments why a separate app would be more appropriate. It may be slightly less convenient to have two apps and switch between them, but the separation of concerns is likely beneficial to overall usability. This avoids clutter with other (personal) attributes and it allows the user interface to be optimized towards the purpose of managing and verifying powers.

A separate app requires integrating with the user's Wallet. As third party applications cannot be trusted to have the user's best interests at heart, the Wallet must have a permissioning system like commonly found in most mobile operating systems[15]. Chapter 6 discusses the implications that this separation of concerns has on the Wallet.

[15] Mobile operating systems such as Android and iOS let users decide the permissions of each application, such as access to camera, GPS, files, etc.

5.6.3 Verification Process

In the verification process, the *subject* proves to the *verifier* that they are authorised for some intended action. This verification may take place in a physical setting where both actors have the ZZ app, or in a digital setting where the subject holding the ZZ app interacts with some web interface.

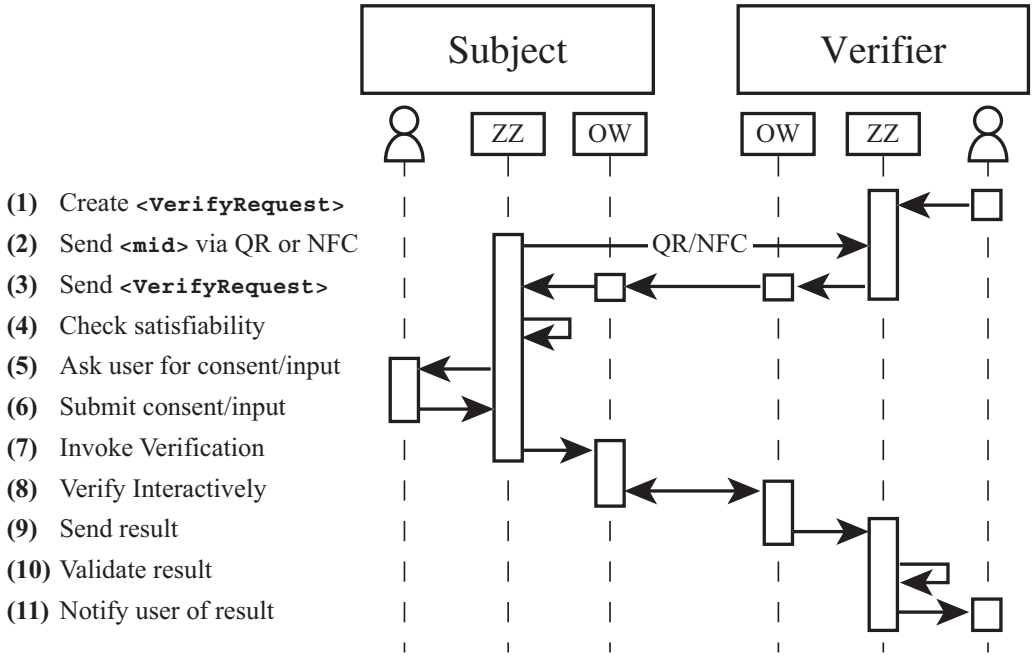


Figure 5.2. A Subject and Verifier performing Verification using both Zekere Zaken (ZZ) and Open Wallet (OW) apps.

Figure 5.2 shows the process of a verifier verifying a subject in a physical setting. First, the verifier defines which power he wishes to check (1). He then asks the subject to share his member ID via QR or NFC (2) which allows the verifier to send his request to this address (3). Note that this communication is proxied via the OW app. The subject's ZZ app checks whether it can satisfy the request (4). If unsatisfiable, it notifies the subject and terminates the process. However, if satisfiable it asks the subject for consent and optionally additional input (5). After providing input (6), the ZZ app constructs the verification response and requests the OW app to execute the verification (7). Assuming the ZZ app is permitted, the OW apps of both parties execute the verification (8) after which the verifier's ZZ app is notified of the result (9). This app then validates the result (10) thereby evaluating whether the subject has the power as defined in step 1. Finally, the verifier is then notified of the result (11).

5.6.4 Authorisation Process

Before power can be proved, it must be acquired. Hence, the *subject* requests[16] the *issuer* to grant them a particular power. In this situation, both actors have the ZZ app, but they need not be in physical proximity.

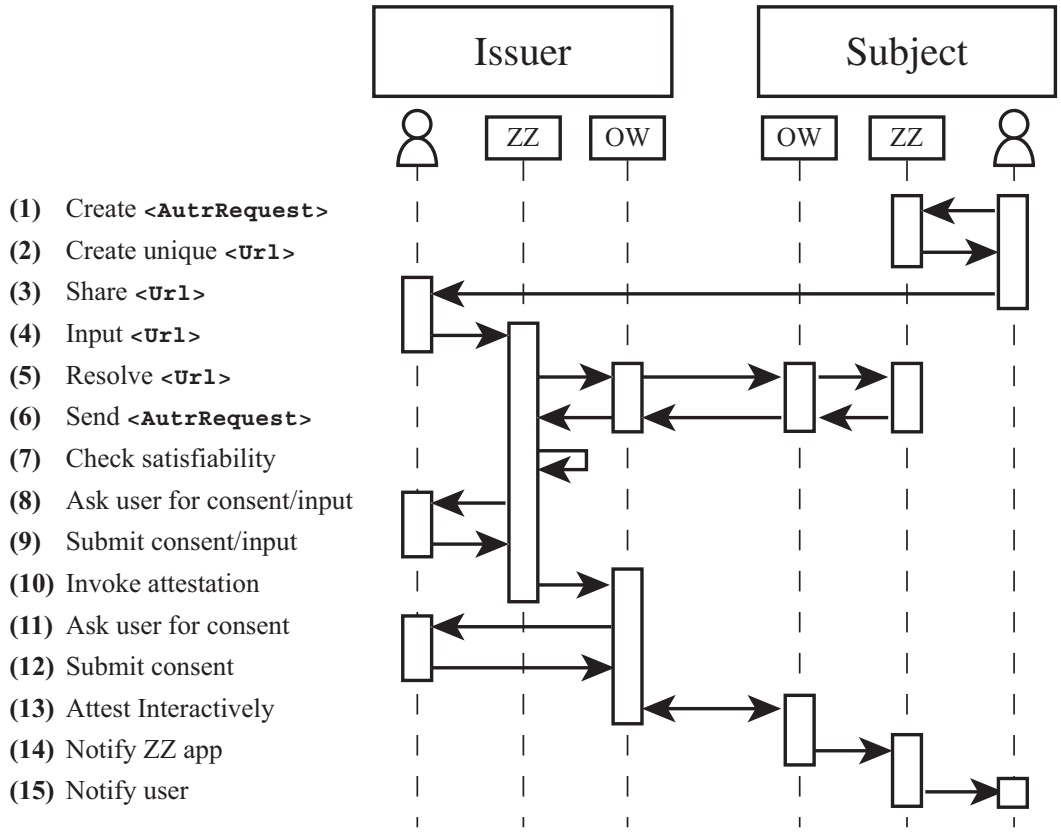


Figure 5.3. A Subject and Issuer performing Authorisation using both Zekere Zaken (ZZ) and Open Wallet (OW) app.

Figure 5.3 illustrates the process of an issuer authorising a subject, per the subject's request. Again, both actors have two apps, Zekere Zaken (ZZ) and Open Wallet (OW). First, the subject defines the required authorisation in the ZZ app (1) which returns a unique URL (2). The user can share this URL with the issuer via any external communication channel (3). As the issuer opens the URL with his ZZ app, his app resolves the URL via the OW app, which ends up at the ZZ app of the subject (5). The subject's app knows the URL as it generated it, so it resolves it to the authorisation request defined in step 1 and returns that (6). The issuer's ZZ app now checks whether it can satisfy the subject's request (7). If unsatisfiable, it notifies the issuer and

[16] This translates easily to a scenario where the issuer initiates (i.e. offers attestation), but this is left out of this analysis.

terminates the process. However, if satisfiable it asks the issuer for consent and optionally additional input (8). After providing input (9) the ZZ app requests the OW App to start the attestation (10). To protect the issuer the OW app now directly requests the issuer whether he consents to this attestation (11). Upon consenting (12), the OW apps perform the attestation (13). Finally, the ZZ app of the subject is notified (14), which in turn notifies the subject (15).

6

Design of a Common Semantic Layer

The findings from the Legal Entity Authorisation use case studied in the previous chapter are now generalized to a semantic layer on top of TrustChain. The first section presents the design of a generic agent that can assume all roles, issuer, subject and verifier. Section 6.2 then discusses design considerations for a generic purpose Wallet. Sections 6.3 and 6.4 then present solutions for issuing and verification.

6.1 Design

The user needs to exert manual control over its identity operations through some independent Graphical User Interface. It is assumed this will primarily be operated on smart phones. To provide issuers like the KVK with automated control, a headless programmed controller is also supported. The model depicted in Figure 6.1 shows the use of a single stack, named "Open Wallet", that provides an Agent which can be controlled both manually and programmatically.

This figure shows the technical context in which the Agent operates. This section will discuss each of the connections, marked *a* through *h*. The subject that owns the Agent can either control it manually via a Graphical User Interface (a, b), or in an automated fashion using software (c).

3rd party applications running on the device may interact with the user's sovereign identity through the Open Wallet Core (d). The user can interact

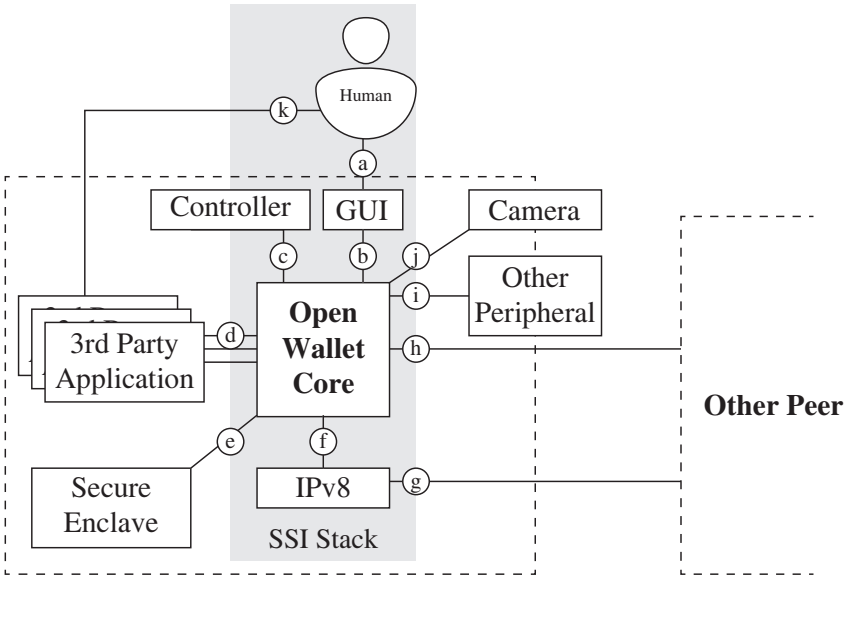


Figure 6.1. Technical Context of the Semantic Layer.

with these apps directly if they offer a Graphical User Interface (k), which requires them to switch between applications. These differ from the programmed controller (c) in terms of authority: the Controller is assumed be a trusted representative of the Subject and hence has full control, whereas 3rd party applications who need a programmable interface are not immediately trusted so their control is limited.

The Open Wallet Core assumes availability of a Secure Enclave (e) for storing keys and private data. On mobile devices it makes use of a camera (j) for scanning QR codes and may use other peripherals (i) such as Near-Field Communication (NFC).

It makes use of the IPv8 library as introduced before (f). The stack is designed in such a way that this library, responsible for the core identity operations such as signing, is only operated by the Open Wallet Core. Finally, each Peer can communicate with other peers running this stack through the low level IPv8 protocol or a higher level Open Wallet protocol.

6.2 Wallet

The case study of chapter 5 showed the need for an independent Wallet application that protects the individual's identity information (for privacy) as well as its signature (for accountability). This section presents the design considerations for this Wallet and elicits new requirements.

6.2.1 Graphical Interface or Headless

The Wallet could run as a background service on the user's device, with 3rd party applications providing use case specific interfaces. This would remove the need for the user to operate another application, and the development complexity of creating a generic user interface that is meaningful for many use case. On the other hand, if every use case needs a separate app it will lead to an explosion of apps or some use cases cannot use the identity system because they cannot afford to build an app. Furthermore, as argued before, the user needs an unbiased view of their identity and 3rd party applications likely hold a conflict of interest. Therefore, some independent user interface is required. The question however remains which capabilities this Wallet must offer, and which can be safely delegated to other apps.

6.2.2 Pluralism of Wallets

It is highly unlikely that a single application can properly serve all people's needs. The beauty of Self-Sovereign Identity is that this is not necessary. As long as actors share the same protocols, many Wallet applications can be built that better suit specific audiences.

6.2.3 3rd Party App Integration

In the case of Secure Business, a specific set of attributes is read and written by the app. The user's signature is used when delegating and receiving power, i.e. when attesting and being attested to. An essential functionality of the Wallet is to protect the user from abuse by 3rd party apps (similar to permissions on mobile operating systems [60]), whilst balancing this with user convenience. Three measures are considered:

1. Provide users with fine-grained permission control over apps, discerning by operation (attesting, being attested to, verifying, being verified) and by attribute name or type.
2. Prompt users for elevated access if 3rd party apps ask for it, especially for sensitive actions (e.g. involving the user's signature).
3. Log what the app has accessed.

Whilst logging should be always enabled, it depends on the application when permissions are granted by default and when the user is prompted for consent. In the case of the Secure Business app, the user is only prompted for elevated access when authorising another person. For the remainder of operations, the app requests during installation the access to the attributes listed in table 5.2 and the passport attributes for legal identification.

6.2.4 Generic Credential Management

Whilst the management of legal entity authorisations may require a separate interface, for many other use cases it may not be necessary. Many users are already accustomed to the *simple* practice of collecting, holding and disclosing credentials such as id cards, credit cards, health insurance cards found in an actual wallet, but also proofs of possession (car, house, land) and other entitlement documents such as certificates and diploma's. These are usually provided by a well-known issuing authority and requested by several actors for a multitude of use cases. Hence, similar SSI developments such as IRMA [9], Sovrin [61] and uPort [6] use the credential analogy as a signed collection of attributes, and a Wallet as a collection of credentials. This is applicable to use cases where the meaning and value of credentials can be understood without additional explanation. The remainder of this chapter will design agents for the issuers and verifiers of such credentials.

6.2.5 Authentication

The analysis of the cryptographic layer revealed the issue of identity binding. The need of additional authentication was subsequently identified by the case study. The problem is that possession of the private key of a pseudonym is only a single authentication factor which means stealing the phone is sufficient to hijack the identity. As assurance level increases with the number of independent authentication factors [5], multi-factor authentication is becoming more and more common factors [62]. The European Union Regulation No 910/2014 on *electronic identification and trust services for electronic transactions in the internal market*, known as eIDAS, specifies three levels of assurance for authentication: low, substantial and high [45]. Both substantial and high require multiple authentication factors to be used.

IRMA uses a 5 digit PIN which is verified by an authentication server before the identity is unlocked [63]. If the registration authority (Stichting IRMA) would require digital or in-person showing of ID during registration, this would amount to *e-IDAS substantial*[1]. TrustChain's experiments with Dutch Government have resulted in a authentication system that performs in-person registration at a municipality desk (which is required for eIDAS high). The subject can later authenticate at any time by capturing a video of their face, following a small challenge (e.g. *turn head left*) to prevent replay attacks. The video is sent to the registration authority (a government server) and compared against the passport photograph. To acquire eIDAS level high however, the private data and keys must be stored on a separate

[1] The eIDAS regulation specifies more constraints, but this is out of the current scope

hardware security module (HSM) which is dedicated hardware that performs cryptographic functions such as data encryption and decryption [64].

These authentication methods unfortunately reintroduce the trusted party, the authentication provider that control whether or not users can use their identity. However, the traditional federated identity paradigm allowed relying parties and identity providers to communicate directly [5], which is a violation to privacy as the identity provider learns all the users activity, and association with relying parties. The SSI approach improves this by letting users (1) manage their own keys and (2) request a new authentication attestation themselves, without any exposing the verifier's identity.

TrustChain's experimental Wallet, here referred to as the RvIG app, has the selfie-based authentication mechanism built-in. Unfortunately the code cannot be open sourced due to security and intellectual property concerns[2]. However, as argued before a Wallet must be open source to ensure its integrity. Therefore the current author proposed to separate the concerns of Wallet and authentication component by means of an open protocol, as shown in figure 6.2. This allows the government, or any other identity provider, to have full control over their authentication mechanism. It also alleviates them from the requirement of building a generic Wallet, and provides Wallet owners with the freedom[3] of picking alternate identity providers. This architectural change was adopted by the work group and is to be implemented in the next phase. The development of this protocol is out of the scope of this thesis.

6.3 Issuing

The case study showed two forms of issuing: (1) the *trade register* issuing executive power to all executives and (2) each *participant* distributing power of some other actors. The main difference between the two is that the former is at much larger scale and can be automated as it draws from a database, whilst the latter is a manual operation based on human insight. The manual operation required development of a custom mobile application, but the first case can be generalized to other issuers.

The KVK issuing procedure (section 5.5) involved a database that maps from an identifier (e.g. *BSN*) to a set of attributes. These attributes can be attested to a pseudonym if it can be authenticated, i.e. mapped to a record in the database. If the user can use the Wallet's verification mechanism, rather than an external login mechanism, to prove that it belongs to the known identifier, this procedure can become very efficient.

[2] This was observed during the author's participation in the project work group.

[3] In terms of technology

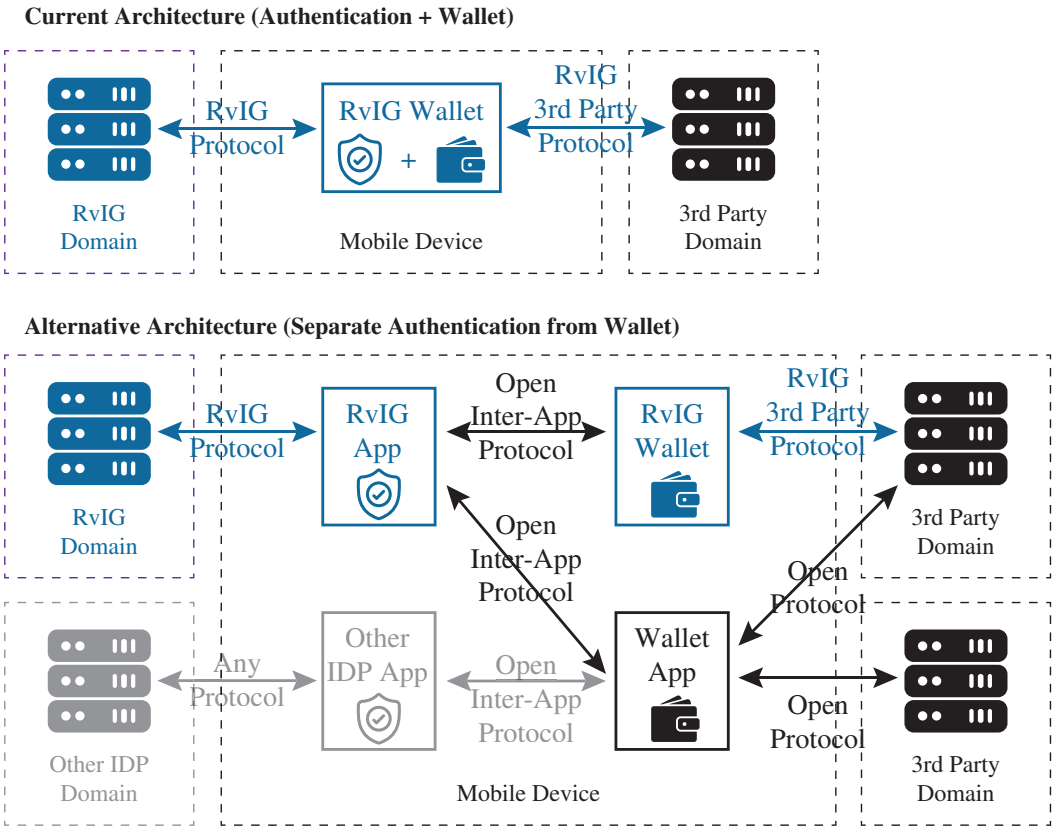


Figure 6.2. Integration between Wallets and Authentication Apps.

Consider Alice, a Wallet user, being asked for credentials. Alice may not possess the information that is asked of her, yet. Her central role brings her the responsibility of collecting information from the right sources in the desired format. A common approach taken by related SSI work is the *portal-based* approach [6], [9], [61]. It allows an organisation to enrich its existing web portal with a QR: scanning it with a Wallet triggers an attestation procedure, providing Alice with attributes. If Alice must look for information she does not yet possess, this approach becomes quite cumbersome. She must first find the source of the desired information, unsure of their SSI-compatibility. She must then log in, find the resource and scan the QR. When self-sovereign identity becomes mainstream, this may grow to be a daily task, making SSI a burden rather than a relief.

Hence, this thesis proposes an alternative approach: *wallet-based* attestation. This allows Alice to collect attributes directly from within her Wallet, by providing a uniform way to interact with data sources. Each available offering is called a *recipe*. Issuers can offer many such *recipes* which provide their clients with different kinds of information. Sections 7.1.2 and 7.6 describe this solution from the user and issuer perspective, respectively.

6.4 Verification

The case study focused on a verification in a physical setting, where two actors both operate the same app. With the proliferation of online activity however, verification may also be required online. Online, SSI can replace not only existing *login* systems, but also save a lot of effort and cost when filling in web-forms at both the user and verifier side, possibly saving 1 billion euros in the Netherlands alone [65][4].

The cryptographic layer already has a protocol for verification, but it is unsuitable for this kind of verification. A verifier, Bob, can only request verification a particular attestation of Alice. This is problematic in two ways. First, it was argued that Bob should not be able to browse Alice's attestations as metadata should be kept confidential, meaning he cannot *pick* a satisfactory attestation and requests its verification. Second, if Alice does not (yet) hold the credentials and Bob cannot communicate to her *what* he wants, then it is impossible for Alice to collect the required credentials.

This which motivates the creation of a higher level protocol on top of IPv8: the *Open Wallet Verification Protocol*. It allows Bob to query Alice's records

[4] The Dutch ombudsman has investigated the Dutch government's practice of communicating with citizens through paper and online forms [66]. Filling in these forms is often a tedious process.

by specifying his requirements in a *VerifyRequest*. It is Alice who then searches her attestations, decides to share and constructs a proper VerifyResponse. The response may also contain certain attribute *values*, a feature that is not supported by IPv8. Furthermore, if she does not *have* the requested attributes, she can attempt to fetch these from a suitable data provider, because she knows *what* is missing. The protocol is depicted in figure 6.3. Note that it distinguishes the Open Wallet from the IPv8 components.

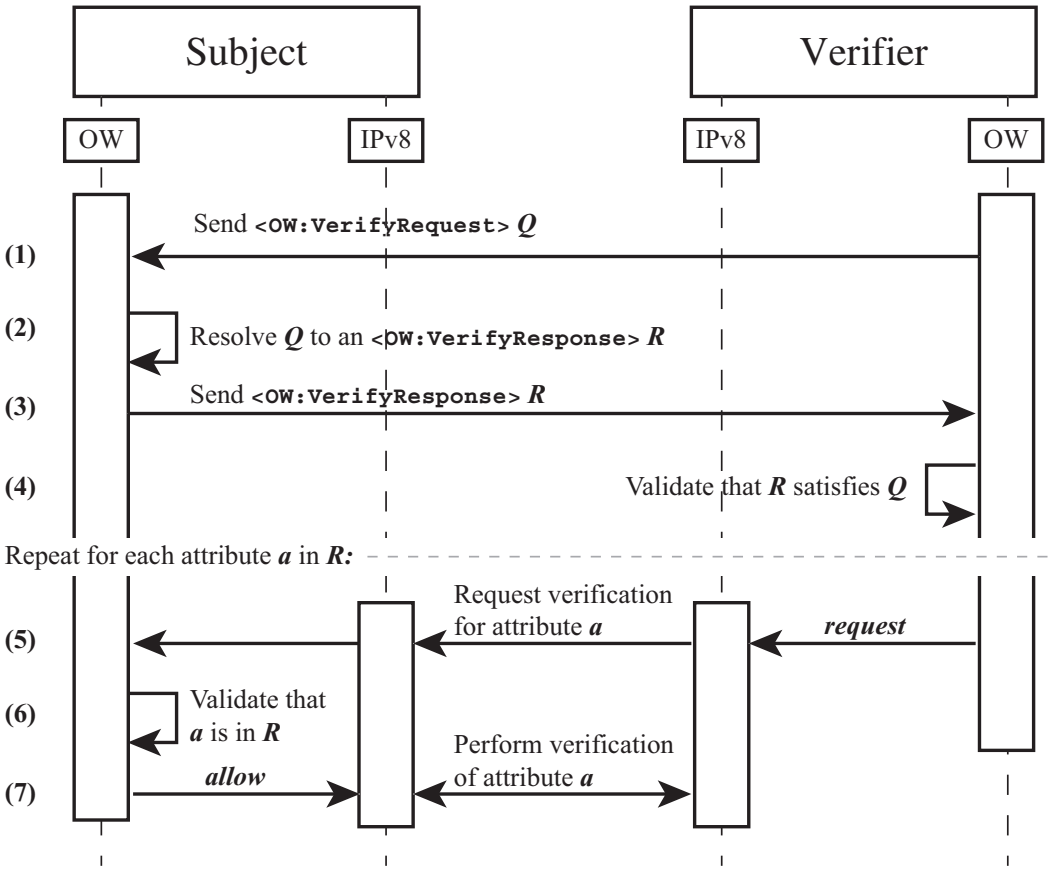


Figure 6.3. Verification Protocol Execution Flow.

7

Infrastructure Prototype

Chapter 4 analysed the peer-to-peer identity layer upon which this thesis builds. Chapter 6 presented the design of a semantic layer as another step towards a Self-Sovereign Identity infrastructure. This chapter presents a proof of concept implementation of this design, called *Open Wallet* (OW).

The major paradigm shift that self-sovereign identity brings is the central role of the subject, Alice. It is therefore only natural to start discussing this prototype from her perspective. Hence, first an Android-based *Wallet* application is presented in section 7.1, that shows how Alice collects identity information from one party and shares it with another.

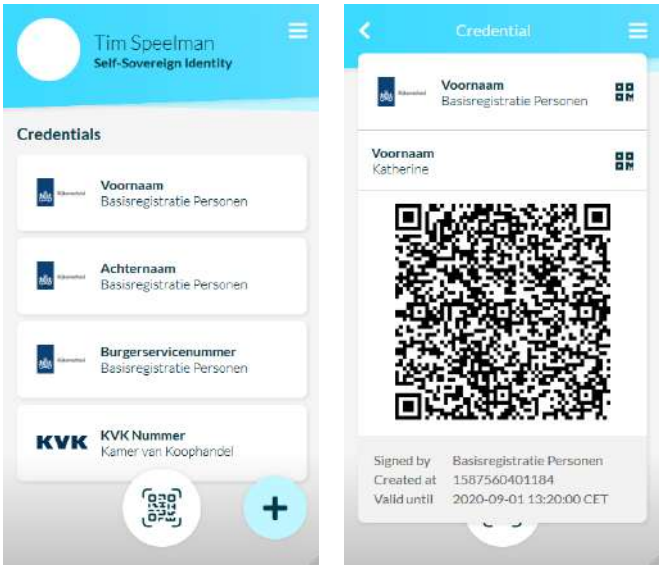
Alice's interaction with other parties introduces the need for protocols and services. Section 7.3 presents the *verification* protocol and is followed by a containerized *verification service*, in section 7.4. Section 7.5 discusses an *attestation* protocol and section 7.6 presents a *recipe-based* attestation service, that provides Alice with convenient data collection.

7.1 Wallet

The primary complicating factor of self-sovereign identity is arguably involving untrained end-users in complex identity operations. As chapter 3 explained, this involves at least properly *informing* the end user and providing the right *controls* that balance freedom with protection. The Wallet prototype presented in this section addresses these concerns.

At the heart of this Wallet are the *assets*: provable identity information, backed by trusted parties. Figure 7.1a displays an overview of this information.

By tapping one of the listed attributes, the user can inspect details as shown in figure 7.1b. The QR code shown can be scanned by another wallet to do simple single-attribute phone-to-phone verifications.



(a) Credential Overview. **(b)** Credential Details.

Figure 7.1. Credential Management in Open Wallet.

The two main procedures, verification and attestation, will first be discussed from Alice's perspective.

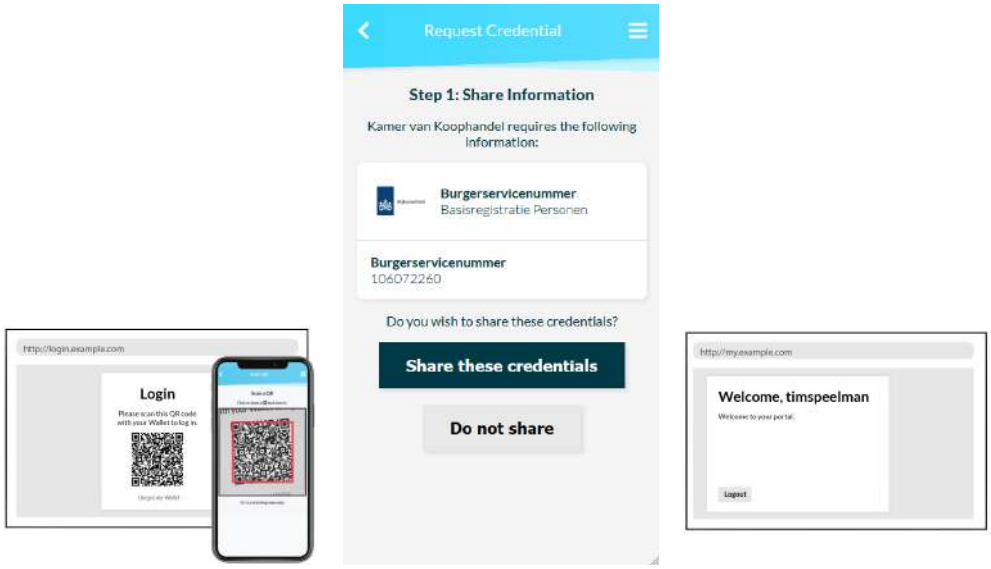
7.1.1 Verification

Suppose Alice visits a website that requires her to authenticate using Open Wallet verification. Her user experience is illustrated in figure 7.2. The website displays a QR code, and prompts Alice to scan this with her Wallet. As her Wallet resolves the URL embedded in the QR code, it is pointed to the Verification Service which returns a verification request. If the Wallet contains these attributes, it asks Alice if she consents to sharing them, as depicted in figure 7.2b. Upon her consent, the Wallet and Verification Service execute the Zero-Knowledge Proofs. If this is successful, Alice is granted access.

The details of this verification protocol are discussed in section 7.3. Section 7.4 explains how this Verification Service can be integrated with just a few lines of code.

7.1.2 Attestation

Figure 7.5 shows the user's perspective on a Wallet-based attestation procedure.



(a) The user scans a QR on the log in page.

(b) The Wallet prompts the user if she consents to sharing information.

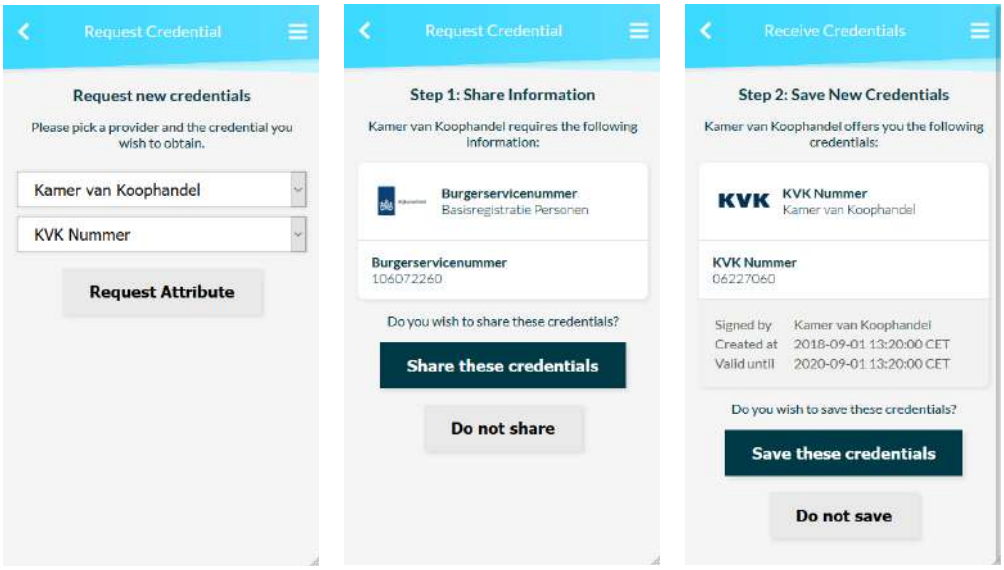
(c) Upon successful verification, the user is granted access.

Figure 7.2. Logging in with Open Wallet verification.

Recipe Selection. Given a list of known services, Alice must be able to select the proper procedure for retrieving the attributes that it needs. The contextual information and the motivation that the user has for retrieving information is specific to the use case at hand. Alice may know beforehand which issuer she needs (e.g. *her university*), but in other cases she may only know the *type* of information she needs (e.g. a *driver's license*, but who issues that?). However, a generic approach is needed that suits all use cases. As the list of known services becomes very large, the query for "diploma" may for example return thousands of schools and universities that issue such diploma, hence Alice must easily be able to narrow it down.

Figure 7.3a shows the recipe selection page of this prototype. It has a single search field that directly queries both the names of services as well as the names of recipes or their underlying attributes. Upon selecting a single recipe, its details become visible and Alice can proceed or decide to pick a different one.

Consent to Verify. Upon selection of the procedure, the Wallet immediately checks which of the requested attributes are already present in the Wallet and satisfy the requirements. If all attributes are present, the Wallet asks Alice whether she consents to sharing these attributes with the service in order to obtain desired data. Figure 7.3b shows an example of this prompt. Upon her



(a) The user can select the desired provider and recipe.

(b) The Wallet prompts the user if he consents to sharing information.

(c) The Wallet asks the user for consent to store the offered attributes.

Figure 7.3. Wallet-Based Attestation.

consent, the Wallet allows verification, shares the requested attestations and requests the promised attributes.

Consent to Attest. Once the Attestation service has verified the attributes, it will query its data source to find the promised attributes. It offers these to the Wallet, which in turn asks Alice whether she agrees with the values and wishes to collect this information.

The attestation procedure that underlies this process is presented in section 7.5. Section 7.6 shows how the attestation service is implemented and can be set up with minimal effort.

7.2 Using IPv8

The peer-to-peer identity solution introduced in chapter 4, named *IPv8*, has been implemented in Python [36]. Among other services, it offers a REST API specifically for attestations[1] and their verification. A detailed overview of this API is included in Appendix C. An abstraction layer is added over this

[1] IPv8 REST API Attestation Endpoint `https://github.com/Tribler/py-ipv8/blob/master/ipv8/REST/attestation_endpoint.py`

API in the form of a client that provides some semantic sugar. It is available on GitHub[2] and presented in detail in Appendix C.

7.3 Verification Protocol

Section 6.4 presented the OWVerification protocol. This section will introduce the message types used for this prototype: the `<OW:VerifyRequest>` and `<OW:VerifyResponse>` message. Subsequently, verifier- and subject-side implementations will be discussed.

7.3.1 The OW:VerifyRequest Message

Bob's verification request contains requirements on the information he wishes to verify. This *question* can take many forms, such as *"are you over 18?"*, *"what is your age?"* or *"are you eligible to buy liquor?"*. Whilst each of these three questions may be answered by the same attribute, they are semantically very different. The first asks for a simple yes/no answer, a range proof that does not disclose the age but merely proves it is within a particular range. The second asks for disclosing the specific age *value*. The third does not ask for a specific attribute, but for any set of attributes that would satisfy the *liquor access policy*; which differs per country and may depend on more than age.

In this prototype, the requests specify a set of *attribute names*, which supports questions in the first or second form. Questions of the third form will be addressed in chapter 8. The request may specify additional constraints on these attributes, explained in more detail hereafter. The data type of the `<OW:VerifyRequest>` message is defined in table 7.1.

Field	Type	Description
attributes	List of `<OW:VerifyReqAttr>`	Required. The required attributes, see table 7.2.
metadata	string	Optional. Any additional metadata.
reason	string	Optional. Human readable reason for verification.
ref	string	Optional. A reference to this request, to be used in the response
return_address	string	Optional. A URL that specifies where to return the `<OW:VerifyResponse>`
subject_id	string	Optional. A member ID of the subject. If unspecified, any subject may respond.
verifier_id	string	Required. A member ID of the verifier.

Table 7.1. The `<OW:VerifyRequest>` data type.

[2] `https://github.com/TimSpeelman/ow-ssi`

Field	Type	Description
attribute	`<Attribute>`	Required. The requested attribute, see table 7.3.
constraints	List of `<Constraint>`	Optional. A list of constraints that this attribute must satisfy.
include_value	boolean	Optional. Defaults to false. If true, the subject must include the attribute value in its response
ref	string	Optional. A reference to the attribute specification in the request.

Table 7.2. The `<OW:VerifyReqAttr>` data type.

Field	Type	Description
name	string	Required. The name of the attribute
format	string	Required. The proof format of the attribute
schema_uri	string	Optional. A URI identifying a schema definition that describes this attribute

Table 7.3. The `<Attribute>` data type.

Constraints. Table 7.2 shows that a requested attribute may be accompanied by a list of constraints. If the Wallet can interpret these constraints and apply them to the available attributes, it is more likely to return a satisfactory result.

If the `<OW:VerifyRequest>` resolves to a satisfactory result and the user consents with sharing the data, the Wallet returns an `<OW:VerifyResponse>` message.

7.3.2 The OW:VerifyResponse Message

The response must either reject the request, or answer it completely. A VerifyResponse is constructed according to the data type specified in table 7.4.

Field	Type	Description
attributes	List of `<OW:VerifyRespAttr>`	The response to each attribute query, see table 7.5). Empty if user did not consent.
consent	boolean	Required. If true, consents to request.
request_hash	string	Required. SHA-1 Hash of IdentityRequest object.
subject_id	string	Required. The member ID of the subject.
ref	string	Optional. A reference to the request this responds to.

Table 7.4. The `<OW:VerifyResponse>` data type.

Field	Value	Description
ref	string	Optional. A reference to the `<OW:VerifyReqAttr>` item listed in the request (see table 7.2).
value	string	Optional. The attribute value, specified if and only if the corresponding `<OW:VerifyReqAttr>` has **include_value** set to true.
hash	string	Required. The attribute hash used in IPv8 that proves this requirement.

Table 7.5. The `<OW:VerifyRespAttr>` data type.

7.3.3 Verifier Side implementation

The verifier must first construct and communicate its request. The request-response interaction may occur over any secure communication channel. Note that channel security is essential when exchanging sensitive attribute values or metadata. This exchange could occur over HTTPS, or over a secure IPv8 channel. Both HTTPS and IPv8 bindings have been implemented.

Upon receiving the `<OW:VerifyResponse>`, the Verifier must ensure that it validates and satisfies the `<OW:VerifyRequest>`. Then it will perform the IPv8 verification in accordance with the negotiated terms and ensure that this verification also yields the expected results. These functionalities have been implemented in the OWVerifier class[3].

7.3.4 Subject Side Implementation

In order to answer an `<OW:VerifyRequest>`, the subject must first retrieve the set of attributes and attestations that satisfy the constraints. To this end, the OWVerifyRequestResolver[4] searches the available attributes for a match, and constructs a valid `<OW:VerifyResponse>` if it succeeds. If it fails, it reports which attributes are missing, or which requirements have multiple solutions. This allows the Wallet to ask for end-user consent, and inform them when the request cannot be satisfied without their help. Upon consent, the OWVerifiee class[5] drives the IPv8 instance to execute verification on the agreed terms.

7.4 A Configurable Verification Service

To realise the full potential of self-sovereign identity, it must be adopted at large scale. This requires low-effort integration with existing services. As a proof of concept, a minimal stand-alone Verification Service is implemented that verifies users according to some configurable specification. The user perspective is

[3] Class `src/ow/protocol/OWVerifier.ts [67]`
[4] Class `src/ow/resolution/OWVerifyRequestResolver.ts [67]`
[5] Class `src/ow/protocol/OWVerifiee.ts [67]`

explained in section 7.1.1. This section presents the implementation of this service.

The first element of this scenario is the Verification Service. It can be be run by using the OW SSI library[6] as npm[7] package, or by running a Docker container with the following command[8]:

```
docker run --mount type=bind,source=/path/to/shared/folder,
    destination=/share owauth
```

This runs the Docker image parameterized by a JSON configuration file. The configuration file describes a set of named *templates* which specify what kind of information to verify. The use of templates allows one service to be used for multiple verification cases. An example configuration is listed in listing 7.1. It simply specifies `<OW:VerifyRequest>` objects.

```
{
    "templates": {
        "delivery_address": {
            "attributes": [{
                "ref": "address",
                "name": "address",
                "format": "id_metadata",
                "include_value": true
            }, {
                "ref": "postal",
                "name": "postal_code",
                "format": "id_metadata",
                "include_value": true
            }],
        },
    }
}
```

Listing 7.1. Example JSON Configuration for Verification Service.

The service contains an internal IPv8 instance and exposes a public HTTP API. If a web application requires verification of the subject's *home address*, the sequence of calls is as shown in figure 7.4. First, the web client requests a verification session from the service, specifying the desired template (1). The Verification Service creates a new session with a UUID and returns this to the web client along with a redirect URL (2). The web client then forwards

[6] This will require running an IPv8 instance as well. An example integration using the library is available at `src/examples/auth-service/example.ts [67]`

[7] Node Package Manager `https://www.npmjs.com/`

[8] A detailed explanation for building and using the docker image is available at `images/auth-service [67]`

this URL to the subject's Wallet, for example by QR code (3), upon which the Wallet calls the Verification Service (4). The service responds with the `<OW:VerifyRequest>` that belongs to this session (5), matching the template specified in step 1. The Wallet and Verification Service now execute the Open Wallet verification protocol (6) as detailed in figure 6.3. In the meanwhile, the web client polls the status of the verification session (7). Once verification is complete, the Verification Service returns the result (8) in two formats: JSON and JSON Web Token (JWT).

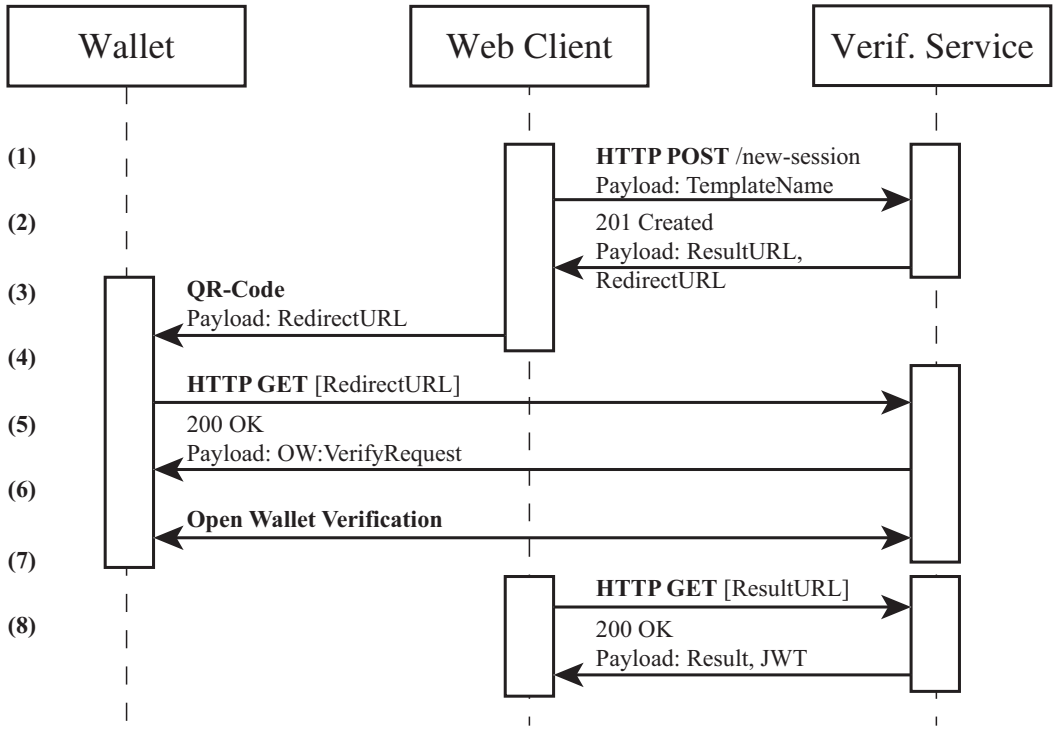


Figure 7.4. Open Wallet Verification using an HTTP Verification Service.

The web client can display information from the plain JSON result to the user. However, when the client needs to prove the integrity and authenticity of the result to any other information system, such as an authorisation server or database, this plain data is useless. This is because the web client runs in an untrusted environment: the end-user's browser. For this case, the same result is wrapped in a JWT, which contains a signature from the Verification Service. Any other party that can verify the signature and trusts this service, can now trust the result. For convenience, the Verification Service serves a small JavaScript library that reduces the web client's efforts to only two lines of code:

```
1 <script src="https://verify.service/client.js"></script>
2 <script>OWVerifyService.verifyByQR(templateName, onResult,
      elementId)</script>
```

This code will put a QR code in an HTML element that redirects the Subject's Wallet as shown in figure 7.2. When the user completes the verification, the `onResult` callback is invoked with the result in both formats, such as in listing 7.2.

```
1 {
2     "success": true,
3     "result": {
4         "address": "Sesamestreet 123",
5         "postal_code": "9876ZX"
6     },
7     "jwt": "eyJhb...",
8 }
```

Listing 7.2. Example Result from Verification Service.

7.5 Attestation Protocol

The IPv8 attestation protocol, detailed in appendix C, enables Alice to send an attestation request to Chris specifying an attribute name and proof format she would like him to attest to. Chris may then complete the attestation by providing some *attribute value* and generating a Zero-Knowledge Proof of it using the given format. IPv8 does not store not transmit any values, which limits its applicability. Alice must be able to exert control over the attribute's value, at the very least in terms of consent. This requires knowledge of the value. This motivates development of an additional protocol: the Open Wallet Attestation Protocol.

7.5.1 The OW:AttestOffer Message

Section 7.1.2 mentioned two ways of data collection: *portal-based* and *wallet-based*. These approaches differ in the way they are started but in both cases the issuer offers the subject to attest to a set of attributes and values. This is specified in an `<OW:AttestOffer>` message as specified in Table 7.6.

The conventional portal-based approach to attestation, explained in section 7.1.2, uses a QR code that embeds or refers to an attestation offer. The `<OW:AttestOffer>` message is sufficient for this approach, as the user's *request* is taken care of by the web application displaying the QR code. However,

Field	Type	Description
attester_id	string	Required. The member ID of the actor that offers to attest.
attributes	List of `<OW:AttestOfferAttr>`	Required. The attributes that will be attested to. See Table 7.7.
ref	string	Optional. A reference to a session or interaction that this offer belongs to.
subject_id	string	Required. The member ID of the subject that will receive attestation.

Table 7.6. The `<OW:AttestOffer>` data type.

Field	Type	Description
attribute	`<Attribute>`	Required. The attribute specifier.
value	string	Required. The value of this attribute.
ref	string	Optional. A reference to some external session.

Table 7.7. The `<OW:AttestOfferAttr>` data type.

wallet-based attestation requires additional messages to be sent between subject and attester. This will be discussed in the next subsection.

7.5.2 Attester Side Implementation

If Alice agrees with the `<OW:AttestOffer>`, she and Chris perform IPv8 level attestation. This translation is made in the OWAttester class[9]. It also ensures that Alice's IPv8 level request does not differ from the offer.

7.5.3 Attestee Implementation

The OWAttestee class[10] translates an `<OW:AttestOffer>` into IPv8 level attestation. It also ensures that Chris's IPv8 attestation responses do not differ from what he has offered and *self-verifies* the resulting Zero-Knowledge Proofs to assert that they work as expected.

7.6 A Configurable Recipe Attestation Service

Wallet-based attestation as presented in section 7.1.2 enables users to collect attestations without leaving their Wallet. This section presents a proof-of-concept implementation of a recipe-based attestation service. The protocol in figure 7.5 allows a Wallet to request attestation of some attributes based on a recipe; a description of attributes offered and an optional verification request

[9] `src/ow/protocol/OWAttester.ts [67]`
[10] `src/ow/protocol/OWAttestee.ts [67]`

that specifies the verification that must occur before the user is granted access to the attestations.

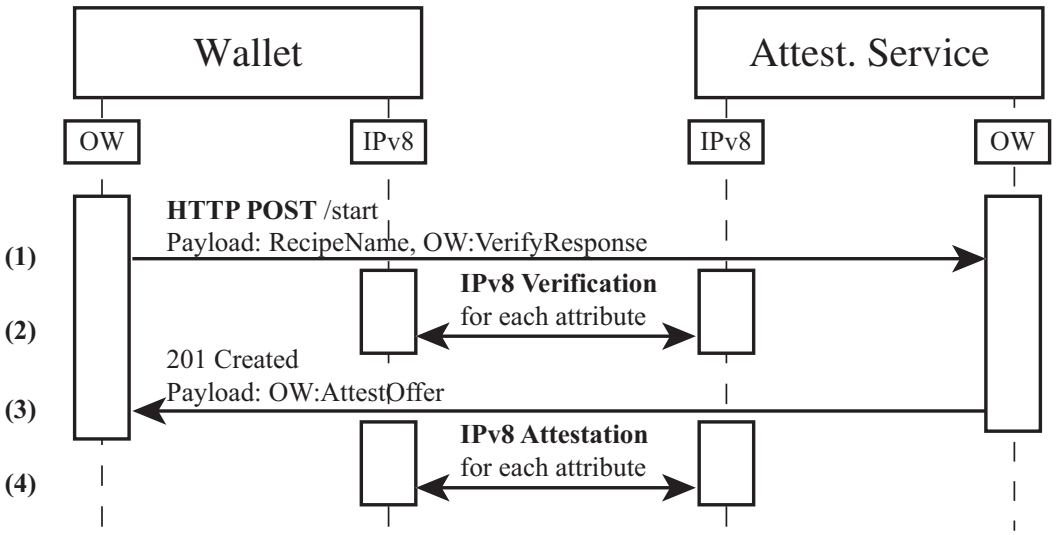


Figure 7.5. Open Wallet Attestation using an HTTP Recipe Service.

An example Recipe is presented in listing 7.3. It offers one attribute (*kvknr*) and demands verification of another attribute: **bsn**. It also mentions the service endpoint URL to which the client must send Recipe Requests and perform the attest-by-recipe protocol. An example of such request is shown in listing 7.4.

```
{
    name: "kvknr",
    service_endpoint: "https://attest.kvk.nl/",
    title: {
        nl_NL: "KVK Nummer",
    },
    attributes: [{
        name: "kvknr",
        format: "id_metadata",
        title: {
            nl_NL: "KVK Nummer"
        },
    }],
    verify_request: {
        attributes: [{
            ref: "bsn",
            name: "bsn",
            format: "id_metadata",
            include_value: true
        }]
    },
}
```

Listing 7.3. Example Recipe.

```
{
    recipe_name: "kvknr",
    verify_response: {
        subject_id: "xads12e+asdasd1v143b135as",
        attributes: [{
            ref: "bsn",
            hash: "adv123vrae...",
            value: "123456789"
        }]
    },
    subject_id: "xads12e+asdasd1v143b135as",
}
```

Listing 7.4. Example Recipe Request.

When the attestation server has received the RecipeRequest including the VerifyResponse, it can start executing Open Wallet verification. Once the verification is completed successfully, the attestation service must gather the promised data. This data can be fetched from any data source, based on the identity information verified earlier. In the example, the data lookup occurs based on the user's BSN. Any result can subsequently be offered to the subject

as attribute by sending an `<OW:AttestOffer>` such as shown in listing 7.5.

```
{
    attester_id: "123qdascsdga3tqrq=+",
    subject_id: "xads12e+asdasd1v143b135as",
    attributes: [{
        ref: "kvknr",
        name: "kvknr",
        format: "id_metadata",
        value: "12345678",
    }],
}
```

Listing 7.5. Example Attestation Offer.

If the user consents to the offered attributes, they can invoke IPv8 attestation. They can now use the newly acquired attestations for any purpose of their liking.

8

Secure Business App Prototype

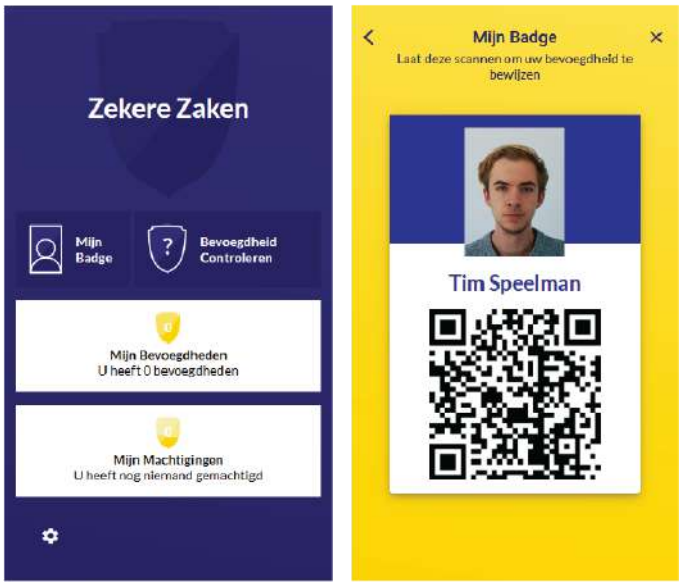
Chapter 5 presented design considerations for a peer-to-peer authorization management system. It addresses the challenge of verifying that a person is authorized to act on behalf of some organisation. This chapter demonstrates a proof of concept of this application, named Zekere Zaken (Dutch for *Secure Business*).

8.1 Presenting a Badge

Figure 8.1a shows the home screen of the application. It is assumed that the most common task is *proving* that one is authorised, which must therefore be the least complex action. Hence, the verifiee only needs to tap the 'my badge' button, which displays the user's *Badge* (figure 8.1b). They can reveal this to the *verifier* so the pair can set up a connection. Three home screen displays three other functionalities:

- **Check Authorisation**. The user can create a new verification request to check whether another user is authorised.
- **My Authorisations**. The user can manage and request authorisations from other people, or directly from the KVK.
- **Given Authorisations**. The user can issue authorisations to others and revoke them.

The following sections describe each of these functionalities in more detail.



(a) Home Screen. **(b)** My Badge.

Figure 8.1. Zekere Zaken Design.

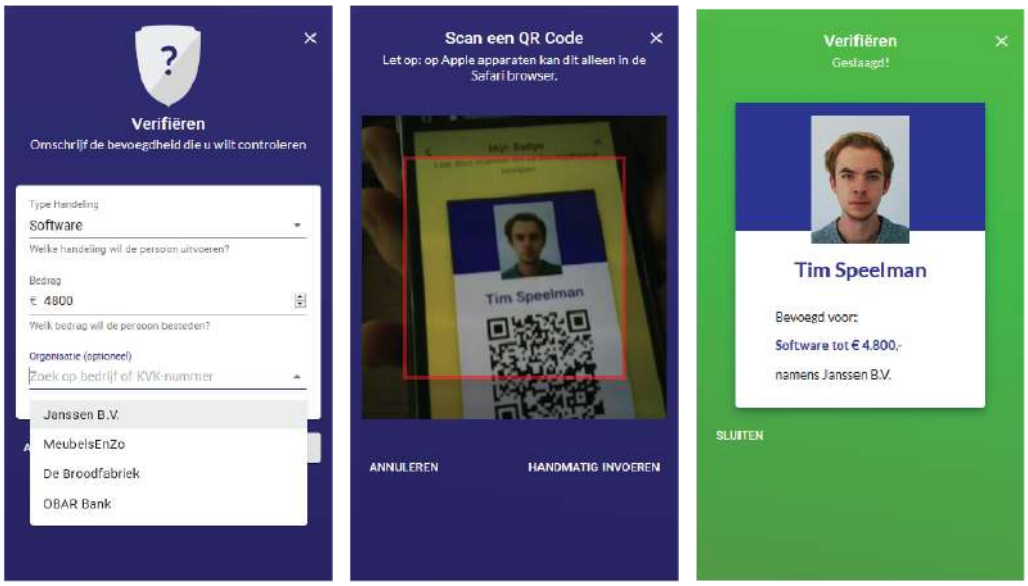
8.2 Checking one's authorisation

The primary functionality of this application is checking one's authorisation. To do so, the user being the Verifier must formulate a verification request; i.e. he must specify which authority he is looking for. Figure 8.2 shows the process of performing such a verification from the perspective of the Verifier in a physical use case. Figure 8.3 shows the user flow of the counter party.

8.2.1 Verifier

Creation of the verification request depends on the information model of authorisations. In this prototype, the user must select a *Type of Action* and a *Maximum Amount* and may optionally specify an *Organisation* as seen in figure 8.2a. The chosen power ontology is a subset from the KVK ontology (table 5.1. If a person wishes to purchase a car on credit (and in name) of an organisation Janssen B.V., the Verifier chooses *Type of Action* to be *Purchase*, the *Amount* to equal the car's selling price and he could specify the Organisation to be Janssen B.V. If the Verifier does not specify the organisation, the buyer may decide this information later on in the process.

After specification of the request, the verifier is prompted to scan the badge of the buyer as in figure 8.2b. The application decodes this into a callback address (a member ID) and a unique reference, which allows the app to send a VerifyRequest to the buyer's agent. The use of the unique reference with an



(a) New Verification. **(b)** Scanning a Badge. **(c)** Verification Succeeded.

Figure 8.2. Zekere Zaken Verifier Flow.

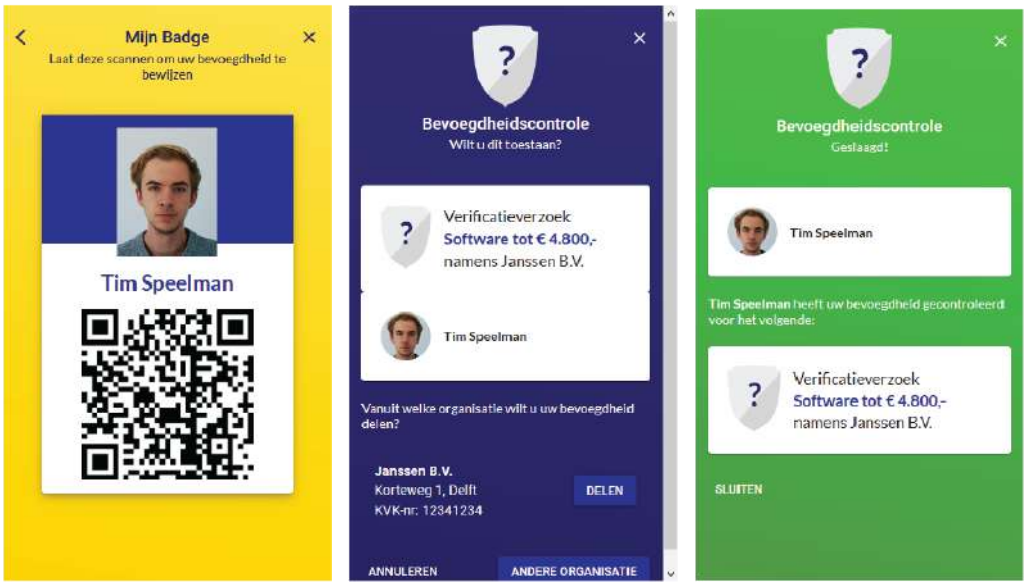
expiration time prevents spamming; only those VerifyRequests specifying a valid reference will be shown to the user.

When the buyer accepts the VerifyRequest, the verification procedure is executed by the wallets as explained in section 7.3. Upon successful verification, the Verifier receives the green light as shown in figure 8.2c which includes the pass-photo and full name of the buyer, including the fact that he is authorised by a particular organisation, as per requirements specified in figure 8.2a.

8.2.2 Verifiee

The Verifiee need only show his badge for the Verifier to scan. This transfers no information other than a rendezvous address and callback reference necessary to establish a connection. After the verification request is sent, the Verifiee sees figure 8.3b. The application immediately checks whether the Verifiee can satisfy the request and displays the option to disclose this information if so. Note that also the personal information of the Verifiee is presented. If the organisation was not specified in the request, the user has the option to choose one here. Upon consent, the two agents execute the verification protocol and the both receive the green light (figure 8.3c).

If the Verifiee does not possess the required power, he can immediately *forward* a request for this power to his superiors, for example via text message. This ad-hoc approach simplifies management of authorisations.



(a) Show Badge to Verifier. **(b)** Received Request. **(c)** Successfully Proven.

Figure 8.3. Zekere Zaken Verifiee Flow.

8.3 Collecting Executive Power

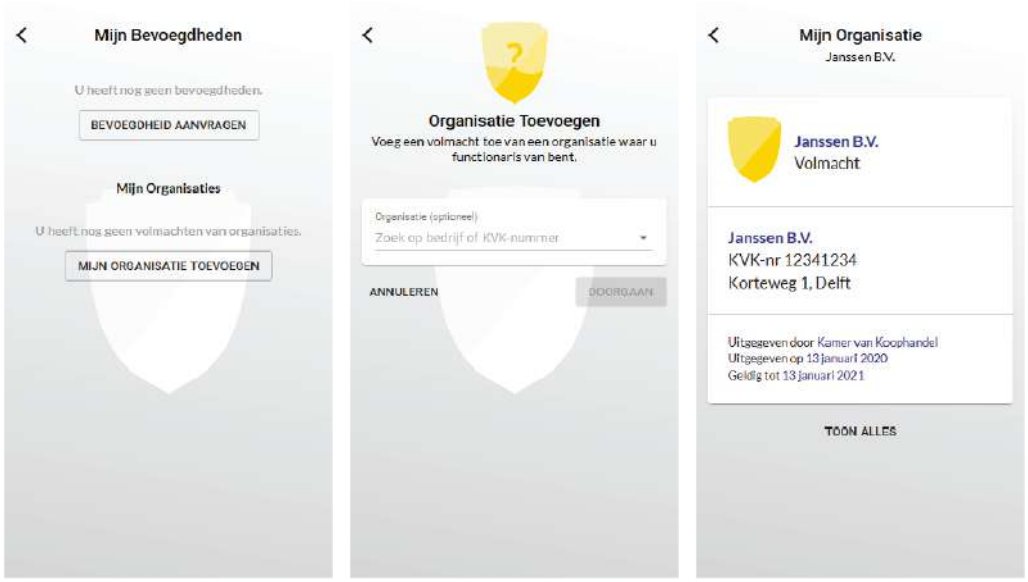
Recall from chapter 5 that, in the Netherlands, the root of legal entity authorisation lies with the *directors*. Hence, before authorising anyone else, first the directors of legal entities must be authorised at the highest level. Figure 8.4 shows how a *director* can request such legal entity control from within the app. Section 5.5 found that this requires the user to specify a particular company as it is not possible to *search* by person. Hence, the user can simply search the KVK database for the organisation he controls, after which the application will execute the issuing recipe. Figure 8.4c shows the resulting authorisation.

8.4 Managing Delegated Power

Once a *director* has root control, they can propagate this to an *employee* or any other natural person. However, this prototype only involves request-based authorisation. Alice requests authority from a director or someone else with the proper authority, Chris.

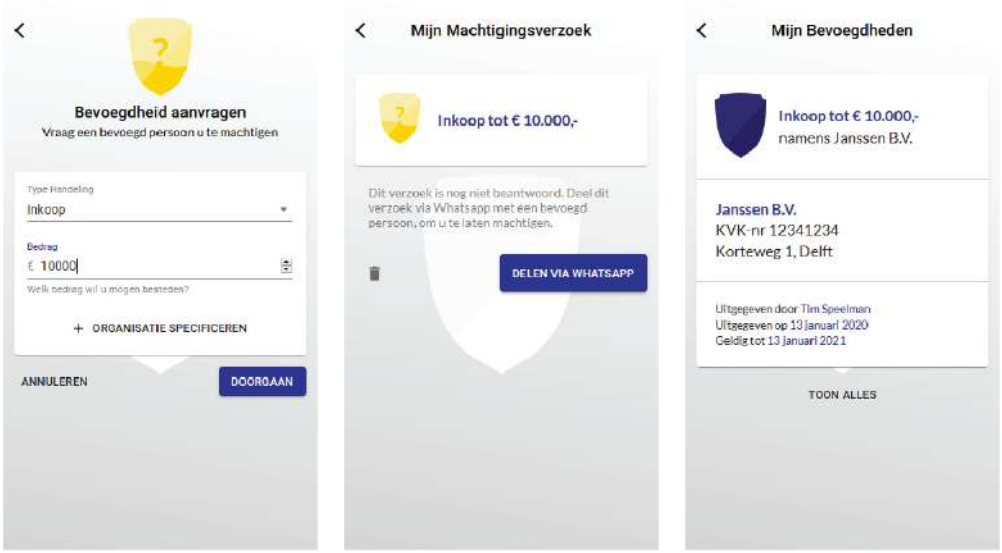
8.4.1 Authorisee

Figure 8.5 illustrates Alice's view on the process: she specifies the desired authority, sends a request-link to Chris via WhatsApp and upon his approval receives her new authority.



(a) Authorization overview. **(b)** Add a Legal Entity. **(c)** Legal Entity control.

Figure 8.4. Adding Legal Entities in Zekere Zaken.

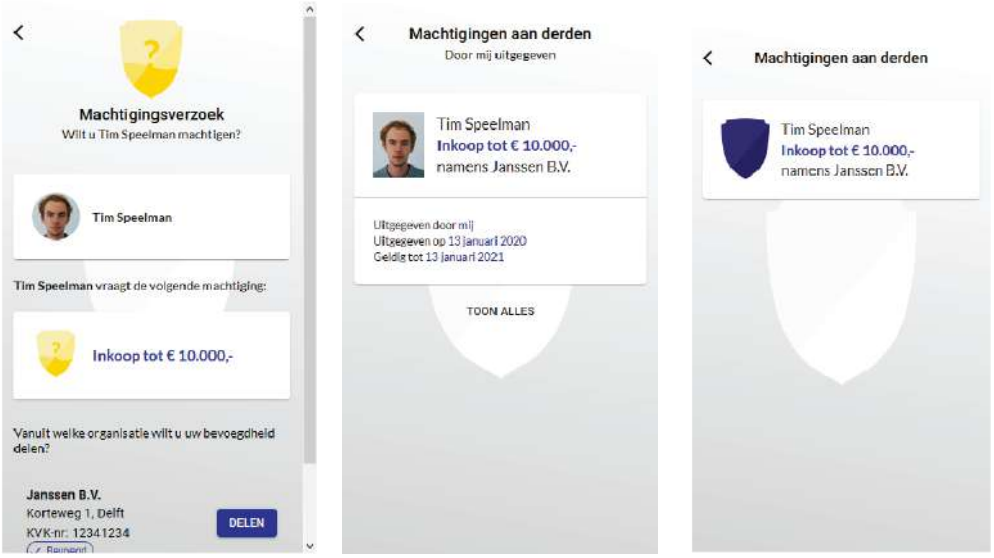


(a) Request Authorisation. **(b)** Share via WhatsApp. **(c)** Granted Authorisation.

Figure 8.5. Requesting authorisation in Zekere Zaken.

8.4.2 Authoriser

Figure 8.6 shows Chris' view: upon opening Alice's link in WhatsApp, his Secure Business app opens showing Alice's authorisation request. If Chris has the proper authority, the app enables him to grant this request. Upon granting the authorisation Chris can inspect details of that authorisation as shown in figure 8.6b and an overview in figure 8.6c.



(a) Receive request. **(b)** Granted Authorisation. **(c)** Granted Authorisation overview.

Figure 8.6. Authorising someone in Zekere Zaken.

9

Evaluation

This chapter describes a real world case study that was performed to demonstrate the applicability of the semantic layer presented in this thesis. In the Kamer van Koophandel kick-off workshop (appendix B), *construction site access* was identified as a relevant application for self-sovereign identity.

This case study took place at a construction site in Nijmegen. Five weeks prior to the on-site demonstration, thousands of euros worth of equipment was stolen at this site in broad daylight, by a man pretending to be authorised. Figure 9.1 shows the first page of the 11 page complaint filed with the police, which described the situation. The claimed to have orders from one of the

POLITIE Kopie

EENHEID OOST-NEDERLAND
DISTRICT GELDERLAND-ZUID
BASISTEAM TWEESTROMENLAND
Telefoon 0900 8844

Proces-verbaalnummer : [redacted]

PROCES-VERBAAL
aangifte

Feit : Gekwalificeerde diefstal in/uit bedrijf/kantoor
Plaats delict : [redacted]
Pleegdatum/tijd : Tussen vrijdag 29 mei 2020 om 17:30 uur en maandag 1 juni 2020 om 16:00 uur

Ik, verbalisant, [redacted], hoofdagent van politie Eenheid Oost-Nederland, verklaar het volgende:

Op dinsdag 2 juni 2020 om 07:47 uur, kwam ik ter plaatse van het misdrijf op de locatie [redacted] Nijmegen, bij een persoon die mij opgaf te zijn:

Figure 9.1. Procès Verbal (redacted for privacy reasons).

Figure 9.2. Unguarded entrance and site manager's office.

subcontractors, borrowed a grinding wheel from some workers on site and used this to grind open several locks. He then stole thousands of euros worth of equipment – including at least 8 drills, 4 jigsaws, 2 construction lights, a levelling line, 7 batteries and chargers, and 33 pieces of hand tools – from six subcontractors at the site.

9.1 Existing Access Control Procedure

Figure 9.2 shows the construction site is fenced, but the entrance of the site is unguarded. Every person is required to register with the *site manager*, located in the yellow cabin next to the entrance. An *inspector* performs regular on-location checks to verify that only authorised personnel is present.

9.1.1 Registration Procedure

Registration first involves legal identification based on a national identity document such as driver's license or passport. These documents are authenticated using a certified system provided by Keesing Technologies[1]. This system is

[1] `https://www.keesingtechnologies.com/automated-id-checking/`

able to verify national and international identity documents, and ensure that these are not expired or revoked. After this authentication procedure, the site manager checks that the individual is authorised. This commonly involves determining that the individual is employed by one of the subcontractors, suppliers, or other companies that have a reason to be present at the location and that they have the proper training and certification. A textual copy of the individual's data is stored in the site manager's system.

9.1.2 Verification Procedure

After one-time registration at the construction site, a dedicated *inspector* performs regular checks to ensure that only authorised personnel is present. Most of these checks are performed by simply recognizing staff, and addressing any person that the inspector does not recognize. This access control procedure failed the day of the theft. It was during the weekend when the inspector and site-manager were not present, but other companies were working. The thief used *social engineering* to pretend his authorisation. The people on site had no simple means to verify the access of this unknown person.

9.2 Self-Sovereign Identity Solution

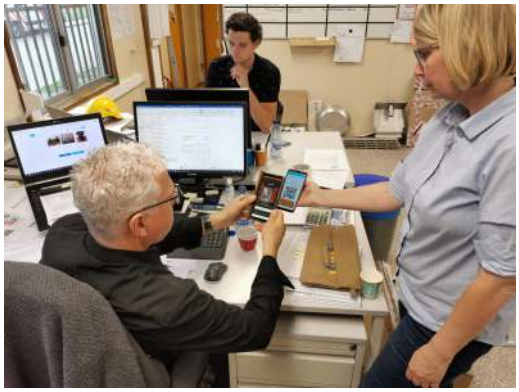
This problem was solved using the self-sovereign identity system presented in this thesis. Instead of a separate application such as Secure Business, this use case was addressed by a small extension to the wallet application. A separate "module" was introduced that provides the users with a focused view on this particular use case and helps specify the verifications and attestations that are needed. This results in a fully peer-to-peer access control mechanism. This section will walk through the three step process:

1. The site manager claims to the inspector that he manages a particular construction site X.
2. The site manager attests to a worker's access to X.
3. The inspector verifies that a worker has access to X, as attested by the site manager.

9.2.1 Trust Establishment

The inspector and site manager first establish a connection by scanning the QR badge that encodes the member id (see figure 9.3b). This allows both parties to store each others identifier under a name of their choosing, very similar to the exchange of phone numbers. The inspector needs to know which

(a) Inspector (left) and site manager (right) establish trust.



(b) Site manager (left) attests to subject (right).

Figure 9.3. Onboarding with the OpenWallet site access module.

identity he can trust to attest to access of construction site X. Therefore, the site manager enters the construction site in his own wallet, so he can manage its access later. Then he shares this information with the inspector. The inspector receives this information and, knowing it came from the manager, accepts it as trustworthy. This concludes the key exchange process.

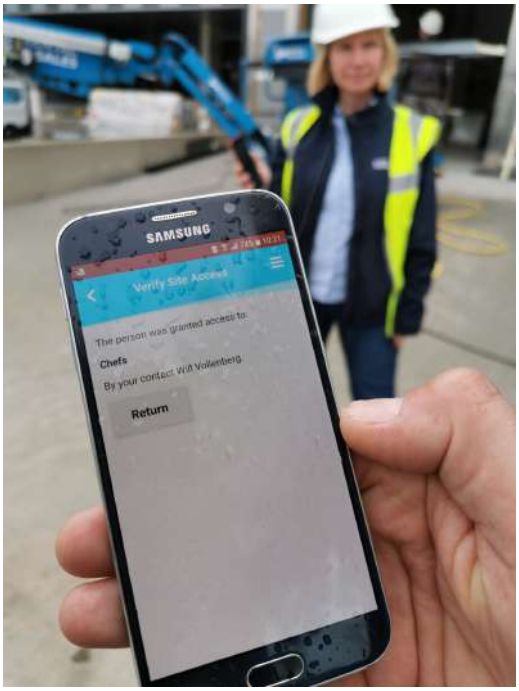
9.2.2 Access Attestation

The site manager performs the regular registration procedure, but concludes by providing the worker with a custom credential that displays her access to the construction site (figure 9.3a). This employs the `<OW:AttestOffer>` message. The connection between worker and site manager can be again set up using the QR badge.

9.2.3 Access Verification

The inspector, or any other person operating a wallet, can now verify the worker by selecting the saved construction site from step 1. The worker need only show her badge, which is then scanned by the inspector's wallet (see figure 9.4a). After this exchange a `<OW:VerifyRequest>` is sent to the worker's wallet. The worker's wallet then resolves this request by checking its database for matching credentials. If it has found one, it prompts its user, the worker, whether she consents to sharing this information. Upon consent, the credentials are returned with an `<OW:VerifyResponse>` message. The wallet finally checks that the verified attestation was actually issued by the trusted site manager. If this is the case, the wallet displays a positive result to the inspector.

(a) Inspector (right) verifies worker (left) is authorised.



(b) Wallet displays positive verification result.

Figure 9.4. Verification with the OpenWallet site access module.

9.3 Results

This case study demonstrated the applicability of the self-sovereign identity system against a real world use case. All participants quickly understood how to use the system and particularly showed interest in the fact that all workers, not just the inspector, would be able to verify one another's authorisation.

10

Conclusion and Future Work

In this thesis, the practical application of TrustChain, a novel self-sovereign identity infrastructure, was explored.

First, a theoretical framework for self-sovereignty was proposed to guide the use case analysis and ease consolidation with future efforts. The framework deviates from other models in several ways. It distinguishes between *ground values* (independence and privacy), *principles* (e.g. control, transparency) and *dimensions* (human, time and context). It adds under-emphasized principles such as *usability* and *convenience* and introduces relationships between the principles. Finally, it introduces the notion of *Boundaries of Sovereignty*, drawing attention to an essential yet underexposed philosophical debate: *what* should the individual actually have control over? This framework is therefore a solid building block, which can be used to consolidate efforts across contexts, borders and cultures.

Second, the general study of the cryptographic layer was performed, which revealed a privacy issue: attestation metadata was unfairly considered *public* and it was argued that this should instead be considered as *confidential* in the relation between subject and verifier. This was also summarily changed and now selective disclosure of metadata has been merged into IPv8[1]. The existing revocation mechanisms of TrustChain are also not sufficient for applications where the issuer is not always online or the subject is not actively participant in the verification, such as with claim forwarding. A mechanism is required

[1] `https://github.com/Tribler/py-ipv8/pull/759`

that allows the issuer to publish revocations without it affecting the privacy of the subject. Whilst it is debatable whether such a power asymmetry between subject and issuer can be considered *sovereign*, it is simply reality in many use cases; governments can revoke passports and driver's licenses without subject consent, employers can revoke authorisations. Supporting single-sided public revocation vastly improves the applicability of this infrastructure.

Third, the Authorisation by Legal Entities case study explored how actors and authorisations can be mapped to pseudonyms and attributes. Both natural and legal persons can be represented by pseudonyms, but for legal persons this requires to manage control over the keys. Furthermore, attributes can represent authorisations but one or more shared ontologies must be defined between the legal entity and the verifier. The case study considered how actors could manage these authorisations in practice. A design was proposed for person-to-person authentication using a mobile application. This design iteration elicited the need for users to have use case specific applications next to their wallet application when this benefits understanding and usability. This separation between *trusted* wallet and third-party apps also ensures that the wallet can stay simple, which is necessary for it to be a commons.

The Kamer van Koophandel holds a key piece of information that, when made available on the SSI network, enables a whole range of opportunities. Unfortunately, the current legal framework and business model of the KVK may restrict the possibility of such integration. These problems likely apply to other (public sector) information sources as well. This invites commercial third parties to act as a middlemen between one or more information sources and the end-user, which poses a threat. By allowing data to flow through, and be correlated by, such parties, the privacy and autonomy of individuals will be harmed. It is therefore best that the ultimate source, the KVK in this instance, supplies the information directly to the network. To at least ease *technical* integration, the proposed semantic layer includes a reusable attestation service that is easy to integrate with existing systems and data sources.

Both the generic theoretical and applied incremental approach led to the creation of a semantic layer on top of TrustChain. This semantic layer involves services for easy integration with existing systems on both the issuer and verifier side. Additionally, it offers a prototype Wallet application. These artifacts can be re-used in future design iterations to discover new use cases, incrementally expanding coverage of the TrustChain infrastructure.

Bibliography

[1] C. Allen, 'The path to self-sovereign identity', *A Life With Alacrity*, 2016.

[2] P. R. Zimmermann, 'The official PGP user's guide', 1994.

[3] A. Whitten and J. D. Tygar, 'Why johnny can't encrypt: A usability evaluation of PGP 5.0', 1999.

[4] W. Diffie and M. Hellman, 'New directions in cryptography', *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[5] M. Benantar, *Access control systems: security, identity management and trust models*. New York, 2006.

[6] D. C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton and M. Sena, 'uPort: A platform for self-sovereign identity', 2016.

[7] D. Khovratovich and J. Law, 'Sovrin: Digital identities in the blockchain era', 2017.

[8] Jolocom, 'A decentralized open source solution for digital identity and access management (whitepaper)', 2019.

[9] G. Alpár, B. Jacobs, W. Lueks and S. Ringers, 'IRMA: Practical, decentralized and privacy-friendly identity management using smartphones', 2017.

[10] J. Pouwelse, 'Trustchain protocol (internet draft)', *Internet Engineering Task Force*, 2018.

[11] K. Young, 'Domains of identity', *Identity Woman*, 2018.

[12] K. Cameron, 'The laws of identity', 2005.

[13] A. H. Maslow, 'A theory of human motivation', *Psychological Review*, 1943.

[14] K. Cameron, 'Let's find a more accurate term than 'self-sovereign identity' – kim cameron's identity weblog', 2018.

[15] T. Ruff, '7 myths of self-sovereign identity', *Medium*, 2018.

[16] M. Schutte. (2016). Schutte's take on SSI principles, GitHub, [Online]. Available: `https://github.com/infominer33/self-sovereign-identity/blob/master/Schutte-on-SSI.md` (visited on 27/02/2020).

[17] I. Wallerstein, *World Systems Analysis: an Introduction*. 1935.

[18] G. Ishmaev, *Open Sourcing Normative Assumptions on Privacy and Other Moral Values in Blockchain Applications*. 2019.

[19] F. Wang and P. De Filippi, 'Self-sovereign identity in a globalized world: Credentials-based identity systems as a driver for economic inclusion', *Frontiers in Blockchain*, vol. 2, 2020.

[20] Q. Xu, R. Zheng, W. Saad and Z. Han, 'Device fingerprinting in wireless networks: Challenges and opportunities', *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 94–104, 2016.

[21] M. Shoebridge, 'Chinese cyber espionage and the national security risks huawei poses to 5g networks', 2018.

[22] F. O'Brolcháin, T. Jacquemard, D. Monaghan, N. O'Connor, P. Novitzky and B. Gordijn, 'The convergence of virtual reality and social networks: Threats to privacy and autonomy', *Science and Engineering Ethics*, vol. 22, no. 1, pp. 1–29, 2016.

[23] B. Jesiek, 'Democratizing software: Open source, the hacker ethic, and beyond', *First Monday*, 2003.

[24] P. Norberg, D. Horne and D. Horne, 'The privacy paradox: Personal information disclosure intentions versus behaviors', *Journal of Consumer Affairs*, vol. 41, no. 1, pp. 100–126, 2007.

[25] A. Cavoukian, 'Privacy by design the 7 foundational principles', 2009.

[26] International Standards Organisation (ISO), 'ISO/IEC 29100:2011 information technology — security techniques — privacy framework', 2011.

[27] E. Johnson, S. Bellman and G. Lohse, 'Defaults, framing and privacy: Why opting in-opting out', *Marketing Letters*, vol. 13, no. 1, pp. 5–15, 2002.

[28] A. Musa and J. Eriksson, 'Tracking unmodified smartphones using wi-fi monitors', presented at the SenSys 2012 - Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems, 2012, pp. 281–294.

[29] World Wide Web Consortium, 'Web content accessibility guidelines (WCAG) overview', Web Accessibility Initiative (WAI), 2018.

[30] J.-H. Hoepman. (2019). The identity crisis (1) – membership vs ownership, [Online]. Available: https://blog.xot.nl/2009/11/04/the-identity-crisis-1-membership-vs-ownership/ (visited on 05/04/2020).

[31] G. Kontaxis, M. Polychronakis, A. Keromytis and E. Markatos, 'Privacy-preserving social plugins', presented at the Proceedings of the 21st USENIX Security Symposium, 2012, pp. 631–646.

[32] European Parliament, Council of the European Union, *Regulation (EU) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (general data protection regulation)*, 2016.

[33] T. L. Dowdeswell and N. Goltz, 'The clash of empires: Regulating technological threats to civil society', *Information & Communications Technology Law*, vol. 29, no. 2, pp. 194–217, 2020.

[34] Tribler. (2020). IPv8 documentation, [Online]. Available: https://py-ipv8.readthedocs.io/en/latest/ (visited on 12/04/2020).

[35] P. Otte, M. de Vos and J. Pouwelse, 'TrustChain: A sybil-resistant scalable blockchain', *Future Generation Computer Systems*, vol. 107, pp. 770–780, 2017.

[36] Q. Stokkink, *GitHub.com/tribler/py-ipv8*, 2020. [Online]. Available: https://github.com/Tribler/py-ipv8 (visited on 12/04/2020).

[37] Q. Stokkink and J. Pouwelse, 'Deployment of a blockchain-based self-sovereign identity', 2018.

[38] University of Southern California Information Sciences Institute, 'IETF RFC0791 - internet protocol', 1981.

[39] ITU, 'X.509 : information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks', 2019.

[40] D. Boneh, E.-J. Goh and K. Nissim, 'Evaluating 2-DNF formulas on ciphertexts', in *Theory of Cryptography*, vol. 3378, Berlin, Heidelberg, 2005, pp. 325–341.

[41] K. Peng and F. Bao, 'An efficient range proof scheme', in *2010 IEEE Second International Conference on Social Computing*, 2010, pp. 826–833.

[42] D. Benarroch, M. Campanelli, D. Fiore and D. Kolonelos, 'Zero-knowledge proofs for set membership: Efficient, succinct, modular',

[43] S. Nakamoto, 'Bitcoin: A peer-to-peer electronic cash system', 2008.

[44] Ethereum. (2020). Ethereum.org, [Online]. Available: `https://ethereum.org` (visited on 01/06/2020).

[45] European Parliament {and} Council of the European Union, *eIDAS regulation (EU) no 910/2014 of the european parliament and of the council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/EC*, 2014.

[46] P. A. Grassi, M. E. Garcia and J. L. Fenton, 'NIST digital identity guidelines: Revision 3', National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-63-3, 2017, NIST SP 800–63–3.

[47] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, *Handelsregisterwet 2007*, wet, 2007.

[48] Kamer van Koophandel. (2020). Tekenbevoegdheid per rechtsvorm, [Online]. Available: `https://www.kvk.nl/informatiebank/tekenbevoegdheid-per-rechtsvorm/` (visited on 22/06/2020).

[49] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, *Financiële regeling handelsregister 2019*, ministeriele-regeling, 2019.

[50] Kamer van Koophandel. (2020). Formulier 13 - inschrijving gevolmachtigde.

[51] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, *Burgerlijk Wetboek Boek 3*, 2020.

[52] eHerkenning. (2020). Leveranciersoverzicht, [Online]. Available: `https://www.eherkenning.nl/leveranciersoverzicht` (visited on 22/06/2020).

[53] L. Marlisa. (2020). 10 vragen over eHerkenning, het inlogsysteem van de Belastingdienst waar ondernemers boos over zijn, Business Insider Nederland, [Online]. Available: `https://www.businessinsider.nl/eherkenning-belastingdienst-inloggen/` (visited on 22/06/2020).

[54] J. A. Mooney, 'Locked out on LinkedIn: LinkedIn account belongs to employee, not employer', 2013.

[55] R. Shirey, 'IETF RFC4949 - internet security glossary, version 2', 2007.

[56] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller and K. Scarfone, 'NIST 800-162 guide to attribute based access control (ABAC) definition and considerations', National Institute of Standards and Technology, NIST SP 800-162, 2014, NIST SP 800–162.

[57] Kamer van Koophandel. (2018). Een nieuwe identiteit, een nieuw DNA. jaarverslag 2018, [Online]. Available: `https://www.kvk.nl/download/KVK_Jaarverslag_2018_tcm109-480173.pdf` (visited on 19/06/2020).

[58] Tweede Kamer der Staten-Generaal, *Regels over het hergebruik van overheidsinformatie (Wet hergebruik van overheidsinformatie); Memorie van toelichting*, officiële publicatie, 2015.

[59] Kamer van Koophandel. (2020). KVK data over de bedrijvendynamiek, [Online]. Available: `https://www.kvk.nl/download/Bedrijvendynamiek-juni-2020_kvk_tcm109-490150.pdf` (visited on 21/06/2020).

[60] A. P. Felt, E. Chin, S. Hanna, D. Song and D. Wagner, 'Android permissions demystified', in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11, Chicago, Illinois, USA, 2011, pp. 627–638.

[61] D. Hardman. (2018). How DIDs, keys, credentials and agents work in sovrin, [Online]. Available: `https://sovrin.org/wp-content/uploads/2019/01/How-DIDs-Keys-Credentials-and-Agents-Work-Together-in-Sovrin-131118.pdf` (visited on 29/05/2020).

[62] S. Harris and F. Maymi, *All In One CISSP Exam Guide*, 7th ed. 2016.

[63] IRMA. (2020). Keyshare protocol · IRMA docs, [Online]. Available: `https://irma.app/docs/` (visited on 29/05/2020).

[64] S. Mavrovouniotis and M. Ganley, 'Hardware security modules', in *Secure Smart Embedded Devices, Platforms and Applications*, K. Markantonakis and K. Mayes, Eds., New York, NY, 2014, pp. 383–405.

[65] TNO, 'Snel en veilig gegevens delen met self-sovereign identity', 2019.

[66] D. Hanse, C. van Vliet, E. ten Berge and B. Wiegel, 'Rapport houd het simpel', 2019.

[67] T. Speelman, *Github.com/TimSpeelman/ow-ssi*, 2020. [Online]. Available: `https://github.com/TimSpeelman/ow-ssi` (visited on 01/06/2020).

[68] Mozilla Developer Network. (2020). Javascript promises, MDN Web Docs, [Online]. Available: `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise` (visited on 23/04/2020).

[69] I. Fette, 'IETF RFC6455 - the WebSocket protocol', 2011.

[70] World Wide Web Consortium. (2015). Server-sent events, [Online]. Available: `https://www.w3.org/TR/eventsource/` (visited on 23/04/2020).

Appendix A

Self-Sovereign Identity Workshops

Self-sovereign identity discussions with the Kamer van Koophandel (KVK) showed wildly more effective when using *physical props* to denote the attributes, pseudonyms and various actors (issuers, verifiers and subjects). Physical proximity suggests their association and movement indicates the flow of data. This inspired the creation of a self-sovereign identity workshop kit, as shown in figure A.1.

Figure A.1. Self-Sovereign Identity Workshop Kit.

This kit contains a set of paper cards depicting generic actors, like *bank* or *school*, and well-known parties like Dutch government and KVK. Other cards depict several personas that allow to reason about individuals in a particular use case. They can be combined with a pseudonym card stating a number (e.g. 215) that represents their digital identity. Finally, a folded 'wallet' card can hold a multitude of attribute cards, like name, age, BSN. This simple tool proved effective at explaining and exploring the complex self-sovereign identity paradigm. The variety in cards supported discussion of a wide range of use cases. It was used in several workshop over the course of this project and is currently used by the Ministry of Internal Affairs and Kingdom Relations as part of their SSI prototype demonstrations.

An immersion into the complex 'messy' reality contributed to the ideas in this thesis. In particular, two opportunities proved invaluable. The collaboration with the Kamer van Koophandel Innovation Lab provided many insights and fruitful discussions. Also, a place in the TrustChain experimentation workgroup involving Stichting ICTU, Rijksdienst voor Identiteitsgegevens (RvIG) and passport manufacturer IDEMIA has been a wealth of inspiration. Furthermore, a wide range of workshops and discussions helped to put the abstract worlds of cryptography and identity into a practical perspective. Some of these workshops are illustrated in the figures below. For reference, workshops were organised at ING bank, Kamer van Koophandel (KVK), Talltech University with Estonian Government and Digicampus, Ministry of Internal Affairs and Kingdom Relations, Ministry of Justice and Security, and at Digicampus in a session with the Secretary-General of the Ministry of Social Affairs and Employment. Other inspirational discussions involved representatives of the Authority of Financial Markets (AFM), The City of The Hague, the Digital Dynamic Attribute Authorisation (DDAA) work group, and the Royal Dutch Association of Civil-law Notaries (KNB).

Figure A.2. Workshop at ING.

Figure A.3. Workshop at KVK.

Figure A.4. Workshop during work-visit in Tallinn, Estonia.

Figure A.5. Workshop at Digicampus with Secretary-General of Ministry of Social Affairs and Employment.

Appendix B

KVK Workshop 'Authorised by SSI'

This appendix includes the report resulting from the kick-off workshop with the Kamer van Koophandel that took place on Wednesday October 30, 2019. It was attended by Saïd Akdim, Bart Bink, Joost Fleuren, Michiel Mayer, Willem Scalogne, Hans Schevers and Tim Speelman (author). The findings presented in chapter 5 and the prototype application in chapter 8 are based on this workshop and other meetings.

The Chamber of Commerce has identified a market need for universal authorisation sharing, which is insufficiently addressed by the current register and may also be out of its scope. The goal of this workshop is to tackle this problem using the Self-Sovereign Identity developed at the TU Delft.

The Chamber of Commerce manages a register that lists the people who are authorised to perform legal acts in the name of specific organisations. This solution is characterized by the following:

- The current scope of these authorisations is usually very wide: allowing the subject all or significant power over the particular organisation.
- The scope can be reduced by specifying it in plain text, hence not machine-readable, or choosing from a limited set of predefined authorisations.
- The process in its current form is time consuming.
- The natural persons are listed by name in a register, which might be considered a violation of privacy.

Figure B.1. KVK Workshop 'Authorised by SSI'.

- The verification of an authorisation requires requesting an extract of the register.
- The register could be considered a single point of failure.
- The resulting authorisation is a legally binding.

The targeted solution should solve the aforementioned drawbacks, whilst maintaining legal value. We took a semi-structured iterative approach in this workshop:

1. Problem analysis and definition.
2. Solution brainstorm and proposals.
3. Evaluation of solution, raising new concerns and understanding.
4. Adjustment of the problem definition.
5. Reiterate.

The 2-hour workshop resulted in the following problem definition, list of concerns and solution proposal. These findings will be applied by building a prototype and subsequently evaluating this with the Chamber of Commerce.

B.1 Problem Definition

Representatives of organisations must be able to prove that they have the proper authorisation to do a particular task. We use a lock-and-key analogy where the product, service or other legal action is a "lock". The authorisation is a "key". The Verifier, offering the lock, should check the following:

1. Does the key fit the lock?
2. Is the key still valid?
3. Is the key issued by (a) a person with a key for which checks 1 through 3 hold, or (b) the Chamber of Commerce?

These authorisations should represent a class of activities that the employee is allowed to do, and not be treated as the company's approval to a particular transaction. This responsibility lies with the relying party.

B.1.1 Does the key fit the lock?

For the first check, we envisioned two approaches:

1. The key is specific to a lock; i.e. a key only fits on one lock.
2. Locks and keys are both defined according to some hierarchical semantics; a key fits all locks that appear under it in the hierarchy.

The first solution is simple but likely results in an abundance of keys, making key management hard. One of the attendees noted that this problem is already present in a centralized authorisation solution called E-Herkenning. Managers, feeling overwhelmed with the amount of keys, simply give their employees all the keys. This causes a security issue.

The second solution allows for more generic keys to be issued (such as: make purchases up to € 50,000) which simplifies issuing, but requires the Issuers and Verifiers to agree upon a hierarchical semantics of authorisations. This problem is currently being investigated by the Chamber of Commerce in a separate project.

We have chosen to focus on the second scenario, but simplify the hierarchy to two elements[1]:

1. `AUTH-FULL` Full power over a company X
2. `AUTH-ENTRY` Access to construction sites in name of company X

Naturally, 2 is contained in 1.

[1] A different ontology was later chosen: *Formulier 13*

B.1.2 Is the key still valid?

The validity of the key can be limited in two ways:

1. Using timestamps, keys can be set to expire. TrustChain is already outfitted with timestamps.
2. A key may need to be revoked by the Issuer, for example in resignation.

As key revocation is out of scope for this thesis, we will not discuss it further.

B.1.3 Does the Issuer have a valid key?

This question makes the problem recursive and creates a hierarchy of key holders. This suits the organisational model where the owner holds all the keys and can selectively share these with employees (vertical authorisation) or across organisations (horizontal authorisation). These employees can subsequently share their key(s) with others.

Upon verification by the Verifier, it will need to recursively check that all Issuers in the chain have a key that fits the lock and is still valid.

The current TrustChain solution only supports identity validation at the subject itself, which means that the Verifier must contact all people in the authorisation chain to verify their own authorisation. This means they must be online and accept the verification, which is impractical. Instead, the Issuers should somehow forward the chain to the Subject, so it can independently present the evidence to the Verifier.

B.1.4 Concerns

The following additional concerns were raised but left out of scope:

1. Would the proposed solution still be legally binding, considering that the authorisations are no longer published by the Chamber of Commerce? This should be addressed by legal experts.
2. Instead of an authorisation for transactions up to a certain amount per transaction, a daily limit could also be of interest. This concern is left out of scope, because it does not fit the current SSI solution as this requires bookkeeping of expenses made by the employee.
3. The authorisation to share keys can be considered an authorisation in itself. This will be left out of scope.
4. When any party in the chain is relieved of his authorisation, the subsequent authorisations could still be considered valid, depending on the context. As it is part of revocation, this will not be addressed.

5. When the chain breaks, it would be very inconvenient if the rest of the chain is invalid. A solution requiring limited manual work may suffice to tackle this. A proposal was made to grant the authority to a "role" instead of a natural person, so as to eliminate the problem of the person leaving the company or changing the role. This comes close to Role Based Access Control, which does not suit all contexts. Furthermore, SSI is made around people and as roles are not separate entities with their own hardware, they cannot be issued to in the current SSI paradigm.
6. The owner of a company may wish insight into all authorisations that are issued by the company.
7. Should an organisation have its own digital identity?

B.2 Proposed Solution Direction

All participants in the chain have a mobile SSI-Wallet application.

1. This Wallet of a Subject holds an attestation of the type `AUTH-FULL` issued by the Chamber of Commerce or `AUTH-ENTRY` issued by another person who also has either attestation.
2. Upon arrival at the construction site, the Verifier asks the Subject for an `AUTH-ENTRY` attestation. If available, the Subject can accept to share this attestation with the Verifier, after which the verification and validity checks are performed (as specified above).
3. If the attestation is not available, the Subject should be able to send an issuing-request by means of a text message to a person who has a valid authorisation. Attendees of the workshop agreed that convenience is of vital importance to the adoption of the solution.
4. The issuing can also be initiated before the verification phase.

B.3 Approach

We will proceed this project in the following fashion:

1. Design explicit User-Flows that visualize the exact interaction as outlined in the previous section.
2. Adapt the current SSI Wallet to suit the above scenario.
3. Evaluate the use in a mock setting.
4. Present the result to the Chamber of Commerce.

Appendix C

IPv8 Client Implementation

This appendix explains the implementation details of the IPv8 API and the Client that uses the API, which is used as a foundation for the applications presented in chapter 7.

C.1 IPv8 API

The peer-to-peer identity solution introduced in chapter 4, named *IPv8*, has been implemented in Python [36]. Among other services, it offers a REST API specifically for attestations[1] and their verification. The functionality of this API can be classified into three categories:

- **Peer Discovery.** Finding peers to attest to, or verify.
- **Attestation.** One peer attests to an attribute of another peer.
- **Verification.** One peer verifies the attestation of another peer.

The next subsections briefly discuss the functionality of this API and finally review the need for higher level communication between peers.

[1] IPv8 REST API Attestation Endpoint `https://github.com/Tribler/py-ipv8/blob/master/ipv8/REST/attestation_endpoint.py`

C.1.1 Peer Discovery API

Recall that peers can only cooperate to perform attestations and verification if they run the same protocol and are connected to each other (section 4.1). Specifically, both peers must be a member of the AttestationCommunity and need to set up a connection before attestation or verification. IPv8 makes use of member IDs, which are the SHA-1 hash of the peers' public keys. The API offers two methods for peer discovery:

1. ListPeers. Lists all peers in the AttestationCommunity currently known to this peer, by member ID.
2. ConnectPeer(m). Starts looking for a peer with the given member ID m. This method does not return any result. If the search is successful, the peer shows up in the ListPeers endpoint. The initiating peer is now also known to peer m.

Only to known peers, those listed in ListPeers, may one send attestation or verification requests. These functionalities will be reviewed next.

C.1.2 Attestation API

The IPv8 attestation service allows two peers to create an attestation on one of the peers (section 4.2). Suppose a peer p_S, hereafter the *Subject*, requests attestation by a peer p_A, the *Attester*[2]. If both peers use the attestation API, the following sequence of calls will enable this attestation procedure:

1. The Subject p_S ensures it *knows* the Attester peer p_A by using peer discovery (section C.1.1).
2. The Subject p_S requests attestation by calling RequestAttestation(p_A, n, f) for an attribute with name n and a proof-format f of its choosing.
3. The Attester p_A observes the new request in the ListAttestationRequests response as a tuple (p_S, n, md).
4. The Attester p_A accepts the request by calling Attest(p_S, n, v), thereby providing the value v of its choosing for the chosen Zero-Knowledge Proof (ZKP). IPv8 does not persist or communicate the value v.
5. The Subject p_S observes the new attestation in the ListAttestations response as a tuple (n, h_A, md, p_S), where h_A is a non-deterministic hash of the newly created attribute.

Once a subject peer has attestations, another peer may request to verify such attestation by referencing the attribute hash h_A as listed in ListAttestations. This is covered in the next subsection.

[2] Here p_S and p_A are referenced by its member ID

C.1.3 Verification API

The IPv8 instance allows two peers to verify an attestation on one of the peers, as discussed in section 4.3. Suppose a peer p_V, the *Verifier*, requests verification of a peer p_S, the *Subject*. The following sequence of calls will enable this verification procedure:

1. The Verifier p_V ensures it *knows* the Subject peer p_S by using peer discovery (section C.1.1).
2. The Verifier p_V requests verification by calling RequestVerification (p_S, h_A, v, f, md) for an attribute with hash h_A of its choosing with corresponding proof format f and testing it for the desired value v. Again, IPv8 does not communicate or persist this value v.
3. The Subject p_S observes the new request in ListVerificationRequests as a tuple (p_S, n).
4. The Subject p_S accepts the request by calling AllowVerification(p_V, n).
5. The Verifier p_V finally observes the result in ListVerificationOutputs. This returns a hash map from attribute hashes (h_A) to a list of tuples (h_V, p) where $h_V = H(v)$ for some hash function H. The result of the ZKP is the probability that p_S actually holds the value v, which is quantified in a probability $0 \leq p < 1$.

It becomes evident that the same parameters are re-used throughout these API calls made by different peers. This calls for coordination, which is discussed.

C.1.4 Coordination

It is up to the agents that consume this API to provide the input and make sense of the output. To perform meaningful attestations and verifications, peers must communicate the following elements on other protocols:

1. *Member IDs* are used throughout these procedures, both to communicate with the correct counter party, as well as for the verifier to understand who the attester is.
2. *Attribute names, proof formats* and optionally *metadata* are the elements of the identity information schema.
3. *Values* may not need to be communicated (for example if the subject must meet certain criteria as *is over 18* or *lives in the Netherlands*), but may be communicated if necessary.
4. *Attribute hashes* are created upon attestation. In order to verify the right information, a Subject and Verifier must agree on which attribute (hash) should be verified.

The agents implemented in the rest of this chapter will exchange this information, when necessary, on higher level protocols.

C.2 IPv8 Client

For convenience, all IPv8 API calls specified in the section C.1 are first combined into a single gateway class named *IPv8API*. This uses a Promise[3] based HTTP client named Axios that works both in browsers and on NodeJS. The class methods map to the API endpoints one-to-one. On top of this API client, the following functionality is added:

- **Events** allow agents to respond to incoming requests and completed attestations or verifications.
- **Asynchronous methods** invoke an action (peer discovery, attestation or verification) and return a promise that resolves when the action yields a result.
- **Request staging** allows agents to *accept* verification or attestation requests before the actual IPv8 request comes in.

This section discusses each in detail.

C.2.1 Events

Agents may respond to incoming requests and need to be notified of completed attestations and verifications. The existing IPv8 implementation does not offer a way to subscribe to such events. One solution is to adapt the IPv8 attestation service to push events to the client, for example by using WebSockets [69] allowing full-duplex communication, or Server-Sent Events [70] for one-way communication.

A much simpler, but arguably less elegant, solution is to poll the existing API endpoints. This leaves the IPv8 implementation untouched. The latter approach was adopted by implementing an *IPv8Service* class that employs hooks which consuming classes can use to "listen" to events, and execute a callback when the event occurs. Specifically, this class offers the following events:

1. Peer Found(m). When attempting to connect to a peer with member ID m by calling ConnectPeer(m) and the peer is found, it will show up in the ListPeers response.

[3] JavaScript Promises are a way to execute code asynchronously[68]

2. Incoming Attestation Request(m, n, md). When a peer m sends an attestation request for attribute named n with metadata md, this will show up in the ListAttestationRequests output.
3. Completed Attestation(n, h_A, md, m). When an attester m completes an attestation we requested, this will show up in the ListAttestations response.
4. Incoming Verification Request(m,n). When a peer m sends a verification request, this will show up in the ListVerificationRequests output.
5. Completed Verification(h_A, h_V, p). When a verification is requested, its result shows up in the ListVerificationOutputs response. When the verification is completed successfully, the probability will reach a certain threshold.

C.2.2 Convenience Methods

To reduce code complexity for an agent consuming this library, three convenient methods are provided that simplify the peer discovery, attestation and verification procedures. This combines the relevant IPv8 method invocations with listening to the appropriate events. All three methods return a Promise that resolves to the relevant outcome if the process succeeds, but rejects if the process times out or fails. These methods are only useful if the response to attestation and verification requests occur within a timeframe that is at most the remaining running time of the agent. In IPv8 it is perfectly fine to answer such request days later, but these methods assume a real-time response.

1. FindPeer(m) checks if a peer with member ID m is present. If not, it sends a ConnectPeer(m) request. Returns a Promise that resolves if the peer was found, or rejects when it has timed out.
2. RequestAttestation(m, n, f) first awaits the result of FindPeer(m). Once the peer with member ID m is found, it requests the peer to attest one of our attributes (with name n and proof-format f). Returns a Promise that resolves to (h_A, md) if the attestation is successful, or rejects when it is not or it has timed out.
3. Verify(m, h_A, v, f, md) first awaits the result of FindPeer(m). Once the peer m is found it requests verification of an attribute with hash h_A on the peer, expecting value v under proof format f. It returns a Promise that resolves to probability p if the verification is successful, or rejects when it is not or it has timed out.

Note that RequestAttestation and Verify only serve the requesting peer, the subject and verifier, respectively. The responding peer still needs to listen

to the incoming request events and respond by invoking the IPv8 API. The next subsection offers convenience for these roles.

C.2.3 Request Staging

This Client is used to develop a system that negotiates attestations and verifications on a higher level protocol. Hence, before the IPv8 attestation or verification is requested, both peers should already have agreed upon the intended specifics (e.g. attribute name, format, value). For these cases, a *Staging* system is created that allows peers to *stage* the requests they expect to receive as a result of earlier negotiations.

The *AttesterService* simply keeps track of a list of templates T where each $t \in T$ is a tuple (n, v, m) consisting of a name n, value v and meant for a peer m. Once a peer m' sends an attestation request $r = (n', m')$, the *AttesterService* compares this request with each template. If $t \in T | t = (n, v, m), n = n', m = m'$ then it will invoke $\textsc{Attest}(m, n, v)$. Otherwise, it will delegate the request to a custom handler if that was specified. The *VerifieeService* performs a similar operation with the verification templates and requests.