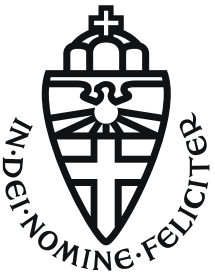
Master thesis
Computing Science – Cyber Security



Radboud University

Measures against over-asking in SSI and the Yivi ecosystem

Author:
Job Doesburg
job.doesburg@ru.nl
s4809327

First supervisor/assessor:
Prof. dr. B.P.F. Jacobs
bart@cs.ru.nl

Second assessor:
Dr. H.K. Schraffenberger
hanna.schraffenberger@ru.nl

October 13, 2023

Abstract

Self-sovereign identity (SSI) is a growing new paradigm for online identity management (IdM). In current implementations of SSI ecosystems, any verifier is able to request any attribute from a user and get it disclosed as long as the user approves this. Users are thus in full control over the disclosure of their own data. While this control has major benefits for interoperability and availability of data, it also burdens users with the responsibility to properly protect this data, which is a responsibility many users might not be capable of bearing.

During disclosure of data in SSI, users must make their own decision on whether a request for data disclosure is legitimate and made by a trustworthy party, whereas in other forms of IdM, the identity provider (IdP) has a responsibility as gatekeeper. The lack of this gatekeeper role could enable phishing, but even more subtle, also enable over-asking: parties requesting disclosure of more data than strictly required. The increased availability (unsiloing) of data in SSI further contributes to this. Considering users potentially being unaware or ignorant of protecting their own data, or simply subject to a power imbalance when a party asks for data disclosure, we argue that over-asking could be a more significant problem in SSI than in other forms of IdM. Users might unfairly expect a privacy-friendly platform (that SSI platforms typically are advertised as) to not allow them to violate their own privacy, while this is not the case. This thesis gives an analysis of the problem of over-asking in SSI ecosystems and considers multiple approaches to prevent it.

As a partial solution, we introduce the concept of *protected attributes* that can only be requested by selected authorised requestors. This offers extra protection, which could be useful for specific, highly sensitive attributes (like the Dutch BSN or attributes from one's DNA). For a broader definition of over-asking (not limited to highly sensitive attributes), a general registration procedure for verifiers must be implemented, requiring a central governing authority to certify disclosure requests. While the administrative costs of operating such a governing party can be high, a system based on open public self-registration that anyone (e.g. democratic bodies and interest groups) can audit could significantly reduce the costs, while still offering decent protection against the most significant forms of over-asking.

While most of our findings are applicable to general SSI systems, specifically, the Yivi ecosystem is considered. The goal is to take a pragmatic approach that does not require too many changes to the existing infrastructure, does not complicate the adoption of Yivi for all parties involved (especially verifiers) and keeps the administrative load minimal while putting responsibilities at the responsible parties. Specifically, TLS-based verifier authentication and scheme-based authorisation of disclosure requests are considered, as they can be implemented with relative ease without introducing the hassle of key management for verifiers. This approach, however, does not scale well to a global scale, which can only be solved by fundamentally changing Yivi's scheme infrastructure.

Ultimately, we conclude that SSI systems should give users more handholds to prevent them from over-sharing their data, which requires both a technical, organisational and UX design approach to the problem.

Contents

Chapter 1

Introduction

Self-sovereign identity (SSI) is a growing new paradigm for online identity management (IdM). It is a claim-based, decentralised and user-centric approach to identity management, where rather than identity providers (IdPs) providing information about an individual to relying parties (after authenticating the user), those parties (*issuers*) issue their claims on the subject in the form of credentials to the user itself, who stores them in a 'wallet' and can then disclose those credentials (or parts of them, which are called *attributes*) to relying parties (*verifiers*) directly.

This approach has notable privacy benefits because it allows users to control their own identity without the IdP being involved in every interaction. Specific cryptographic features allow them to share only the information necessary for a given interaction in zero-knowledge.

While several implementations of SSI wallets have been around for several years, their adoption has not yet become widespread. With the European Commission announcing its EU Digital Identity initiative in 2020, mandating all its member states to make interoperable SSI wallets available to their citizens by 2025, it is to be expected that in the upcoming decade, this technology will gain more traction. More users will possess an SSI wallet containing their credentials, and more issuers and verifiers will appear.

In the current leading paradigm for identity management on the internet, *federated* identity management, centralised identity providers register a user's online identities and provide data to relying parties. Examples of such identity providers could be parties like Google and Facebook, but also a government via an eID(AS) solution like the Dutch DigiD. Users authenticate to such a central identity provider, which in its turn provides identity information to the relying party.

However, these identity providers are not always transparent about what data they share and how they use the (meta)data they collect *themselves* about these sessions. When using Google to authenticate towards an e-commerce store, Google also learns the user is visiting that store. IdPs thus form a *privacy hotspot*.

The architecture of SSI systems solves this problem, with the identity provider (issuer) not having an active role in the disclosure of data. Users possess their own data and can control it all by themselves. Only they can decide which data they want to share with whom.

To our knowledge, all current implementations of SSI ecosystems are designed so that every verifier in the system is allowed to *request* every attribute of a user. Of course, users will have to approve this before data is actually disclosed; that is one of the foundational principles of SSI wallets. But when a user decides to proceed with information disclosure, there are no limiting mechanisms in place that might prevent information from being shared. In principle, any party can ask for any information about a user and receive it when the user approves the disclosure. This makes SSI wallets a universal identity solution, with the user in total control.

We used to consider this a feature of SSI wallets, but in this thesis, we will argue that this can also be a privacy risk.

1.1 Problem statement

On the one hand, the user-centric nature of SSI, with great availability and portability of data, is what makes it so appealing, as it gives users full control over their own identity and data. In this regard, these features of an SSI ecosystem can be considered a privacy benefit compared to previous architectures, where third-party IdPs control a user's identity.

On the other hand, however, giving users complete control over their identity also shifts the *responsibility* of *protecting that data* to the user itself. This is a responsibility many users might not be capable of bearing. Though intuitively, shifting this responsibility to the primary stakeholder (the user) might seem like a good idea, it is not always clear whether users are capable of protecting their own data. The examples where users fail to protect their own privacy are numerous (e.g. cookie popup dialogues, privacy configurations and social media, browser settings, phishing emails, et cetera), both when they are being misled (i.e. phishing or impersonation) or *nudged* (i.e. with *dark design patterns* [1]–[3]), but even when they are not (plain ignorance or disinterest). As such, SSI systems can be considered a privacy risk, too.

1.1.1 Trustworthiness of verifiers

The problem with users being solely responsible for determining the trustworthiness of SSI verifiers is also presented by Chadwick et al. [4]: "Requiring users to know which verifiers to trust is very similar to asking users to know which websites to trust, even when they have not visited them before. Browsers and web PKI now facilitate this process by using trusted third parties, namely X.509 Certification Authorities (CAs) [...]. Web browsers indicate if a secure TLS session has been established to the domain validated web site by displaying a lock icon next to the web site's URL. Something similar will be needed for SSI [...] to enable human users to determine if a verifier is trustworthy or not." [4].

This viewpoint mainly considers a phishing scenario with possibly malicious

parties impersonating a legitimate party, trying to trick users into disclosing data to them. While this is undoubtedly a risk, the problem can also be more subtle.

1.1.2 The myth of true self-sovereignty

Credentials in SSI wallets are considered to be privacy-friendly because of features like unlinkability, transparency, selective disclosure and control over disclosure. Users are *self-sovereign* because all information exchange takes place via the user's wallet and only after explicit consent by the user ('*Do you agree with sharing the following attributes with party x? (y/N)*'). Users are thus in control, as no data is exchanged without the user agreeing. Though this system is indeed beneficial to one's privacy, it is far from perfect.

There are multiple reasons why just asking for user consent is not sufficient. For example,

- users might be unaware or ignorant (possibly due to a cognitive bias causing them to not act rationally) of the sensitivity of certain attributes;
- there might be a power imbalance between the user and the relying party (requestor).

For example, many users are unaware that their citizen registration number (BSN, in the Netherlands) is a protected credential of which the processing, by law, is restricted to only selected (authorised) parties. The average user is not aware of this, nor will they realise the sensitivity of that attribute.

Or consider the, perhaps extreme, case of one's DNA. People might intuitively feel that their DNA is a sensitive piece of information. However, when asked to share (parts of) their DNA profile, they might not oversee the consequences, especially not when, for example, they are offered a discount on their next order or when a potential future employer is requesting specific attributes during an application process.

Simply hitting the '*yes, proceed*' checkbox can thus not be considered actual consent in the context of the whole transaction between the user and requestor as (legal) entities. One could merely consider it a confirmation of being informed that data is being disclosed, not as freely given and well-informed consent.

Users can be unaware, ignorant, or subject to a power imbalance. This results in either a significant knowledge, or power asymmetry. One could argue that this should result in a duty of care, and value driven design of SSI wallet should carefully take these factors into account [2], [3].

Informed consent and the GDPR

The European General Data Protection Regulation (GDPR) is one of the most significant pieces of privacy and data protection legislation. In order to process personal data, the GDPR, among others, requires parties to have a particular *processing ground* for that data. These grounds, among others, can be *consent* and *legal* or *contractual obligations*. When the processing ground is consent, Recital 32 requires that the consent is "*freely given*, *specific*, *informed* and *unambiguous*" [5].

The form in which consent is given is not specified by the GDPR; only does it require that a "*clear affirmative act*" gives the consent. Naively, one might think that hitting the 'yes, proceed' checkbox in an SSI wallet is a clear affirmative act and, thus, sufficient to meet the requirements of the GDPR for consent as a processing ground. The GDPR, however, also requires consent to be freely given and well-informed. We argue that this is not always the case in the context of SSI wallets.

Instead, we argue that the 'yes, proceed' button merely provides *transparency* and the *ability to object*: by requiring the user to press the button, the user is explicitly informed about the fact that data is going to be shared (and which data exactly), and the user is able to abort the session in case they disagree with it. Not objecting, however, does not imply free and well-informed consent.

The user might not be aware of the *consequences* of sharing the data. The wallet interface simply does not offer the ability to inform the user in all detail about all related aspects e.g. purpose limitation, necessity and minimality.

Moreover, the user is not necessarily free to choose *not* to share the data. After all, not sharing the data simply means that the user might not be able to proceed with the transaction they were trying to perform.

Consider, for example, the scenario where a user is performing an (online) job or visa application which requires the user to disclose specific, possibly sensitive, attributes. Refusing to share these attributes might mean the user is not able to apply. In this case, the power imbalance between the user and the requestor renders consent provided via an SSI wallet as a processing ground invalid.

1.1.3 Expectations and responsibilities

As we have seen, in reality, users are far less *self-sovereign* than the SSI technology might imply at first glance. Meanwhile, the promise of SSI to increase one's privacy could further strengthen the ignorance of users towards the topic; users might *expect* a privacy-friendly system to prevent them from violating their own privacy, even when, strictly speaking, they are making that decision themselves. Users might still impute a responsibility towards the platform (the wallet provider), issuers or parties encouraging the use of the wallet (like a government or a relying party).

While one could argue about the *moral* responsibility, we argue that *pragmatically*, it is not feasible to expect users to bear full responsibility for their own privacy, for the reasons mentioned above. The general impression is that users simply are not capable of making well-informed decisions about their own privacy, especially not in the context of a (technically and cryptographically) complicated digital environment where there is also a power imbalance and where usage of the platform is encouraged by a government.

In addition to the earlier mentioned problems, there might thus also be a mismatch in expectations and responsibilities regarding the protection of users against disclosing data that they should not disclose. One could consider that this further contributes to SSI being a potentially dangerous technology.

1.1.4 Factors contributing to over-asking

As identified, in current SSI systems, users could end up sharing sensitive personal data with an untrustworthy party without being aware of it, even without the system (their wallet) giving them tools to be aware of this. And even if a wallet were to prevent this, ignorance or a power imbalance could still lead to data disclosure to a seemingly trustworthy party but that still is not authorised to process that data.

This phenomenon is referred to as *over-asking*: relying parties (or verifiers) asking for more (or more sensitive, or sometimes higher assurance[1]) information than actually required for the purpose they are being disclosed for.

Several factors further contribute to this problem.

Unsiloing of data First, the problem of over-asking is exacerbated by another consequence of SSI wallets that is sometimes called *unsiloing* of data: making data more readily available for disclosure. In an SSI system, all data about a user is available in a single place: the user's wallet. With all users having their attributes readily available for disclosure, the barrier for other parties to ask for these attributes also lowers. Parties offering services that previously would not require users to identify themselves might just start requiring this simply because it became easier to do so[2]. And parties might ask for more information they would not require previously, just because the data became available.

No gatekeepers While over-asking can also occur in federated identity management, here, the identity provider (IdP) has a responsibility in establishing whether the requesting party should receive the data[3] and can be held accountable. Laws and regulations can be put in place to enforce this and protect users. IdPs, thus, to some extent, have the role of gatekeeper. This is not the case in a user-centric SSI system, at least in the current implementations. Previously, as a service provider, you had to convince an identity provider to disclose data to you; now, you just have to convince the user (which, obviously, is much easier considering the reasons described in section 1.1.2).

Loss of context-awareness Additionally, in federated IdM, users might associate a specific identity provider with a specific context (e.g. Google for online games, DigiD for government services). When users are asked to use their national eID to log in to an online game, they might be intuitively more reluctant to do so, as they might not want to share their government-issued identity information with an online game platform. In an SSI system, however, this association between an IdP (and the data it provides) and its context is lost, as all credentials are stored in a single place and disclosed in the same manner. This makes users even less aware of what data they are sharing with which party.

As such, SSI could be seen as an enabler of *sphere transgression*, a term introduced by Sharon [7] to describe the process of Big Tech involving itself in increasingly more *Spheres of Society* [8], where data flows from one context to

[1] For example, when self-asserted data would suffice, as discussed in [6].
[2] This phenomenon in itself is actually called *over-identification*.
[3] At least in theory, they should do so.

another, potentially violating one's privacy. We will further discuss this in section 1.1.5.

While intuitive from a privacy standpoint, the user-centric nature of SSI wallets can be an appealing feature, it also makes them vulnerable. This all makes the problem of over-asking even more prominent. This is also why over-asking was listed as one of the dangers of SSI wallets by the Dutch Ministry of the Interior[4] [9].

Issuer perspective and interest

Finally, the problems mentioned above might also actually hinder the adoption of SSI wallets. The fact that any party can ask for any attribute might discourage issuers from making information available in an SSI system. This is because, as an issuer, when issuing credentials to a user, you also lose control over the data. The data gets out of your sight and becomes vulnerable in a certain way: the data might be disclosed to other parties and be (ab)used in different contexts than it was intended for.[5]

From a legal perspective, it is unclear to what extent which party is responsible for protecting data residing in an SSI wallet. Though it might not be unreasonable to argue that issuers indeed do not have a *legal* responsibility over data stored in a user's SSI wallet, it might not be unreasonable for parties to still experience a *moral* responsibility either.

For certain parties, economic arguments could play a role, too. For example, the Dutch Chamber of Commerce charges a fee for excerpts from their registry (both for traditional excerpts and for issuing SSI credentials in Yivi). Currently, they charge the user for this, but potentially, a *verifier-pays* business model could be more fitting. We will further discuss such economic arguments in section 3.5.2.

At least, a case can be made that under certain circumstances, issuers might also have an interest in restricting access to specific data they would issue in an SSI system.

Encrypted attributes

This is why, in history, we have seen cases where parties decided to issue specific attributes only in encrypted form. That is, the attribute's value is ciphertext, and the issuer only distributes the key to parties they want to have access to the data.

For example, the Dutch VZVZ (*Vereniging van Zorgaanbieders voor Zorgcommunicatie*, or *Association of Healthcare Providers for Healthcare Communication*) has, experimentally and for a short time, been issuing an encrypted form

[4]Apart from over-asking, *over-identification* is mentioned as a separate problem. Where over-asking involves asking for more information (more attributes) than strictly required, over-identification involves requiring identification at all where this would not be necessary. For this thesis, we consider both problems as the same problem, as they are both related to the trustworthiness and legitimacy of a request for data disclosure.

[5]Ironically, issuers experience the same problem as data subjects regarding contextual integrity of their data.

of a user's BSN as '*healthcare code*'.[6] The reason for encrypting the BSN is that they did not want to be held accountable for technically making the BSN available to any other party requesting it.

This way of acting, however, stands perpendicular to other fundamental principles of SSI wallets, like transparency towards the user (a property called *WYSIWYS, What You See Is What You Share*, in [3]), which is why the credential was never accepted in the Yivi production scheme.

1.1.5 Keeping data in context

Nissenbaum has defined privacy in terms of contextual integrity: the ability of a user to keep information in its context [10]. Privacy is violated when personal data from one context flows to another. This has become one of the most popular definitions of privacy.

There are different ways to think about what we should consider for this *context*. A schoolbook example would be the case of a person's medical dossier being disclosed to their employer. This loss of contextual integrity would clearly violate one's privacy. Similarly, we also would not want our medical doctors to know the details from our personnel file. The medical and employment contexts, considered as categories of personal data, should thus not be mixed. This closely relates to Walzer's Spheres of Justice [8].

Maintaining contextual integrity thus highly relates to separating such different categories of data. There are, however, also more subtle cases. For example, when one's email address is disclosed for the purpose of ordering in an online store but is later used for advertising purposes, this could also be considered a break of contextual integrity (even if it is done by the same party). Contextual integrity thus does not only relate to the data itself but also to aspects of the *metadata*, such as the *purpose* that the data has been disclosed for and the party that received it. Such aspects should thus also be considered part of the data's *context*.

We can also express this in (GDPR-)terms like *purpose binding* and *data minimisation*, or the more general (legal) concepts of *necessity* and *sufficiency / proportionality*. Data should have a defined purpose, only be used for that purpose and minimised accordingly; when data is not necessary, it does not have a purpose and thus should be removed or not even be asked for in the first place. The GDPR requires data processors to make data subjects (users) aware of these concepts because, arguably, this information is required for users to make an informed decision on whether they agree with the processing of their data.

In order to maintain privacy as contextual integrity, it is essential that users are well aware of the context of a request for data disclosure: who will be receiving the data and for what purpose (and potentially, what will happen with the data after sharing, how long will it be stored and why is this data strictly required)?

The *context* can be considered as an abstract concept that contains some of the aforementioned aspects, but one can also try to give it a more formal definition.

[6] `https://github.com/privacybydesign/irma-demo-schememanager/blob/master/vzvz/Issues/healthcareidentity/description.xml`.

For example, in section 2.3 we will discuss decision models based on the context of a request for data disclosure, and in section 3.3.4 we will further discuss the importance of context binding.

1.2 Our contribution

As we have argued, the absolute user control that characterises SSI wallets can be considered a privacy benefit but also a privacy risk. Several factors contribute to the problem of over-asking, making it a more prominent danger than before in other forms of IdM. We argue that, at least to some extent, an IdM system should prevent data that *should not be shared* from actually *being shared*.

In this thesis, we give a detailed analysis of the problem of over-asking in SSI ecosystems and try to find solutions to this problem. For this, we aim to find solutions that enable users to make informed decisions about whether a party and their request for data are authentic, trustworthy and legitimate.

Our work is aimed specifically at the Yivi ecosystem, which is a concrete implementation of an SSI ecosystem. When proposing solutions, we take a pragmatic approach with a focus on making minimal changes to the current architecture and implementation of the Yivi ecosystem that can be implemented in the short term, as we believe this topic must be addressed with pragmatism. However, we believe many of our more general findings can also apply to other ecosystems, as the problem of over-asking is not unique to the Yivi ecosystem.

We consider two cases of over-asking. First (in chapter 3), we consider the specific case of sensitive attributes being requested by unauthorised parties. Then (in chapter 4), we consider general cases of over-asking, where the context of the request is more relevant.

Though the Yivi ecosystem is a great way to securely (decentrally) store credentials and let the user control whom they are shared with, some credentials can be so sensitive that just being in control is insufficient. We argue that in certain situations, issuers could have some interest, possibly a (moral) responsibility, to prevent users from sharing those credentials, either as a duty of care or for legal or economic reasons. For this purpose, we propose a solution for protecting specific highly sensitive attributes in the Yivi ecosystem by introducing so-called *protected attributes*.

For the second case, we propose a broader solution for general protection against over-asking based on certified disclosure requests and self-registration of requestors.

The challenge is to do this in a way that is compatible with the existing Yivi ecosystem, does not require too many changes to the existing infrastructure, does not complicate the adoption of Yivi for all parties involved and keeps the administrative load minimal while putting responsibilities at the responsible parties.

Concretely, we do not aim to introduce a dedicated PKI, as we do not want to introduce the hassle of key management for requestors. Instead, our proposal relies on TLS, which is already used in the Yivi ecosystem. Authorisation

happens via the existing schemes, which are simple, open and transparent.

In the end, we argue that our proposals are feasible and do not require too many changes to the existing Yivi ecosystem, but does not scale well to a large number of requestors. With some changes to the way schemes currently work, this problem can be mitigated to some extent (section 5.3), but ultimately, a more fundamental overhaul of the Yivi infrastructure would be required to implement a dedicated PKI.

Additionally, we argue that users might wrongly, though not unreasonably, assume an SSI ecosystem is a safe environment in which privacy-violating behaviour is impossible. For users to be capable of bearing the responsibility of protecting their identity data, we argue that SSI wallets should give users more handholds to prevent them from over-sharing their data. This does not only entail technical or organisational measures but also measures related to usability aspects of SSI wallets. As also described by Schraffenberger and Jacobs, "privacy by design must include careful UX design" [3]. In line with their work, we present a number of possibly alternative, non-technical approaches to mitigate over-asking in section 5.1 and section 5.2.

Reading notes

The problems that are discussed in this thesis are very interdisciplinary. While the intended scope of this thesis is mainly technological, it is impossible not to address the many legal, ethical or governance-related aspects. In fact, solutions to the problem addressed cannot be merely technological but must always be interdisciplinary. However, the author of this thesis does not have an extensive background in these fields. Whenever such aspects are discussed, we try not to go into too much depth and not make strong statements.

Also, while reading this thesis, we often use IdM and SSI terminology. This means that terms like *issuer* and *identity provider*, or *verifier* and *relying party* or *service provider*, are used somewhat interchangeably. Especially the terms *requestor* and *verifier* are used both to indicate the same party, however, the term *requestor* is used as a party requesting the disclosure of information while the wallet might not have initiated disclosure yet, while we use *verifier* when data is actually being disclosed.

Chapter 2

Related work

A significant challenge, maybe the foundational one, for user-centric claim-based IdM like SSI is to establish a trust relation between verifiers and issuers. Verifiers need to trust the claims that users present to them (i.e. the credentials that they disclose) are valid and issued by a trusted issuer.

On a cryptographic level, this is typically achieved by (blind) signatures and zero-knowledge proofs, maintaining unlinkability, selective disclosure and other cryptographic features. On an administrative level, this challenge entails determining whether a specific issuer should be trusted in the ecosystem.

There are multiple approaches, either with a centralised trust framework, a decentralised one (i.e. via a blockchain), or even hybrid approaches. While in some systems, issuers need to be registered at a central administrative party, in other systems, anyone can be an issuer and the trust relation is established via a web-of-trust or, as we see most often, a blockchain is used as a source of truth to establish consensus, guarantee integrity and for timestamping.

Typically, these design decisions of a system are founded in fundamental and sometimes philosophical or political assumptions about society, (public) administration and governance. While centralised infrastructures tend to be based on a continental European concept of governance, with an important role for central governing bodies that establish a base identity for users and organisations, a more Anglo-American concept of governance results in decentralised infrastructures without such central governing bodies and trust is established in a different way. As such, there are different approaches to the concept of (issuer) trust in SSI.

While most academic focus is on the trust from verifiers to issuers, to our knowledge, there is no extended literature on the reverse: **how can users trust verifiers?**

From a cryptographic standpoint, this is not unexpected. While features like selective disclosure and unlinkability of credentials are essential in the trust relation between verifiers and issuers, they are not required – even undesirable in the relation between users and verifiers. Users should be able to maintain a certain level of anonymity in the SSI system, but verifiers (or actually, at

this point, requestors) should never be anonymous, so users should always be certain about with whom they are sharing their data. This certainty, however, can easily be achieved with default cryptography.

While trust between users and verifiers is implicitly touched upon in more high-level descriptions of SSI ecosystems, the first academic work explicitly mentioning the problem we address was recently published in 2023 ([4]). And while it is getting more attention in recent implementations, especially in the EU Digital Identity consortia, a thorough analysis of the topic is lacking.

2.1 Privacy and identity management

In 2005, Cameron described seven Laws of Identity as general principles that "explain the successes and failures of digital identity systems" [11]. Though not explicitly using these terms, he bases his laws mainly on experiences with centralised and federated identity management systems. Meanwhile, he already mentions the need for a "[...] unifying identity metasystem [to] allow digital identity to become loosely coupled" [11] with a user-centric design and SSI-like features. He also acknowledges that "the emergence of a single simplistic digital identity solution as a universal panacea is not realistic" because users experience the internet in many different contexts, and parties want to prevent spillover of digital identity between these contexts [11]. For example, he argues that "it will be a cultural matter whether, for example, citizens agree it is 'necessary and justifiable' for government identities to be used in controlling access to a family wiki", and that even when a cross-sector agreement on a single simple identity system could be made in one country, extending it internationally would be impossible [11]. With this paper, he actually presents some of the foundational problems that the SSI paradigm tries to solve.[1]

Specifically relevant to our research, Cameron mentions the concept of *Justifiable Parties* in one of his laws: "Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship." [11]. Meanwhile, Cameron states that "[this] law is not intended to suggest limitations of what is possible, but rather to outline the dynamics of which we must be aware" [11]. Simply put, the identity system should make users aware of to whom they are disclosing their identity and help users decide if this is justifiable (but not necessarily prevent certain actions).

This is what we see in many federated IdM implementations. First, the IdP determines that the relying party is an authorised party that can request certain data (which often involves the relying party registering itself and making agreements with the IdP beforehand). Then, still, the identity provider displays to the user which party forwarded them and what data is about to be shared. The identity provider thus has an important role in deciding whether a party or

[1] In fact, his paper is based on experiences with the product 'Microsoft Passport', a federated-like IdM system. Later, Cameron would work on 'Windows CardSpace', a user-centric IdM system that looks even more like current SSI systems in its architecture. Later, CardSpace would be discontinued and replaced by U-Prove, a system with cryptographic features to maintain anonymity, which we can consider to be an actual SSI system.

request for information is justifiable, first by enforcing their own policies, and then, by informing the user about this and asking for their consent.

2.1.1 Identity management towards of SSI

Allen describes the history of identity management paradigms in 4 phases [12]:

1. Centralised identity: administrative control by a single authority or hierarchy
2. Federated identity: administrative control by multiple, federated authorities
3. User-centric identity: individual or administrative control across multiple authorities without requiring a federation
4. Self-sovereign identity: individual control across any number of authorities

In 2003, Jordan et al. made a first step towards our current concept of SSI by suggesting a "civil society approach" to "persistent online identity" that "gives individuals a high level of control over how their profile is used", for what they call the Augmented Social Network [13]. They describe problems and shortcomings of then-existing (federated) identity solutions, hinting at a user-centric design.

Jøsang and Pope in 2005 [14] describe a user-centric IdM architecture driven by usability concerns about federated IdM on the growing internet and aiming to simplify the infrastructure. They mention *entities* having multiple *identities*, consisting of a number of characteristics or identifiers. Their proposal includes a Personal Authentication Device (PAD) containing all credentials of a user, which we currently would refer to as a wallet.

Cameron [15] also proposes a user-centric identity framework with a similar architecture, mentioning different kinds of claims on user attributes, incorporating principles of attribute-based credential systems.

While the concept of user-centric IdM did gain traction, actual implementations never really got popular. As Allen summarises it, "[...] final ownership of user-centric identities [remains] with the entities that register them", making it that "being user-centric isn't enough" [12].

The concept of self-sovereignty is mentioned in blog posts by Loffreto about the societal concept of self-proclaimed, or self-determined identities, as opposed to of nationally registered identities [12], [16]–[18]. This idea is reflected by Allen's principle that in SSI, any user must be able to *exist* in the system without the need for a third party [12]. This has significant consequences for the trust framework of systems. From this thought, the concept of SSI became closely related to blockchains as decentralised consensus protocols, being able to register a user's existence without relying on a trusted third party.

Blockchain and SSI

Consequently, existing SSI implementations are based on the idea of Decentralised Identifiers (DIDs)[2] as developed by the World Wide Web Consortium (W3C), which are typically registered in blockchains. This enables users to always register an identity in the system without a central authority being able to prevent it or ban them from the system. Graglia et al. state: "Skeptics have sometimes described blockchain as a hammer looking for nails. Blockchain for SSI is just the opposite; not a case of a hammer looking for nails, but of a nail finding its hammer." [19]. As such, some parties even argue that only blockchain-based systems are truly self-sovereign. Giannopoulou, on the other hand, defines SSI as *blockchain-adjacent*, but not *blockchain-dependent* [20].

Attribute-Based Credential systems and SSI

Meanwhile, developments on Attribute-Based Credential Systems (ABCs) [21]–[23], as proposed by Chaum [24], [25], addressed privacy issues with existing identity systems and became suitable as a complete alternative to existing IdM [26]. These developments focused on privacy-by-design concepts, like partial identities/identifiers, unlinkability and selective/minimal disclosure, which were also being adopted by concepts about SSI.

While the paradigm shift from central and federated IdM, via user-centric IdM, to self-sovereign identity was mostly an ideological one about authority and autonomy, developments towards ABCs were mostly cryptographically inspired by technologies like zero-knowledge proofs and blind signatures, aiming to protect privacy and offer anonymity. The first can be considered top-down, from socio-technological ideas about how the concept of identity should be reflected online, while the latter can be considered bottom-up approaches aiming to build an identity system from technologies for cryptographic anonymity.

Most current implementations can be seen as a form of both, though the exact mathematical properties with regard to the degree of anonymity that they deliver, differ per system. Notably, DIDs typically seen in blockchain-based SSI systems are incompatible with the unlinkability property that ABCs typically implement. However, we will not elaborate on such differences for the rest of this thesis.

2.1.2 Refining SSI properties

Currently, Allen's Ten Principles of Self-Sovereign Identity [12], as a complementary counterpart to Cameron's Seven Laws of Identity [11], are by most considered as the founding principles of SSI: existence, control, access, transparency, persistence, portability, interoperability, minimisation and protection.

In his principles, the topic of privacy is treated in terms of user control, user consent and minimisation of data disclosure (zero-knowledge proofs and selective disclosure). Allen states that "[...] an identity system must balance transparency, fairness, and support" and focuses on protecting the user's autonomy [12]. Notably, Cameron's concept of justifiable parties is not explicitly

[2] `https://www.w3.org/TR/did-core/`

reflected in these principles. The rationale behind this seems to be that because users themselves are in control, there is no need for any party to decide on justifiability (which is a feature); the user determines this themselves.

We argue, however, that in order to protect a user's autonomy, the system must support the user in determining whether a party (or better, a disclosure request) is justifiable. Allen's principles do not state such functionality.

While Cameron's and Allen's principles are leading and most cited, they are not formally defined. Ferdous [27], in 2019, tries to perform a structured academic analysis on them in order to create a common understanding of the concept of SSI. He tries to give mathematical definitions, and he uncouples definitions from the system's architecture. This results in a more formal taxonomy of the different system principles.

Additionally, there have been attempts to refine these principles. López [28], for example, extends Allen's guiding principles to 16, more specific system principles. The topic of privacy with regard to the trust relation between users and verifiers is implicitly touched upon by López, but ultimately he only argues that regulations should be updated. He does not mention a role in this for the system itself.

Naik and Jenkins again revise and extend the set to first 15 [29], and then 20 guiding principles [30] and try to align them with the European General Data Protection Regulation (GDPR) [31]. In this process, however, they merely focus on the specific blockchain that uPort[3] [32], [33] and Sovrin[4] [34], two popular SSI implementations, use and how this affects GDPR compliance. A more holistic analysis of GDPR compliance involving the system's role in helping the user inform to whom they are disclosing data is lacking.

In 2022, Pattiyanon and Aoki tried to analyse the compliance of SSI system properties, resulting in a set of 42 properties that result from different laws, regulations and standards [35]. They conclude that "while the consent property prohibits PII [(Personally Identifiable Information)] disclosure without the user's consent, it is not fully consistent with GDPR article 5.1.(b) due to the lack of a restriction on further processing. We believe this is an opportunity to enhance the SSI system's security and privacy through compliance" [35]. Their detailed overview of properties as a product of their research [36], however, is inconsistent on this topic. They still seem only to mention user consent.

As a conclusion so far, while Cameron does mention the concept of justifiable parties in his original laws of identity, none of the papers specifying requirements for SSI systems explicitly mentions methods to help users determine whether a verifier's request is justifiable.

2.1.3 Establishing trust in SSI verifiers

Only in June 2023, Chadwick et al. addressed the topic of trust in verifiers [4]. Specifically, they extend an existing trust framework for issuers, TRAIN [37],

[3] A project that is now split into Serto (`https://www.serto.id`) and Veramo (`https://veramo.io`).

[4] `https://sovrin.org`

which uses ETSI TS 119–612 trust lists [38] based on DNS, for trust in verifiers as well. For this, they build upon the existence of trust domain administrators that define trust lists.

With regard to the underlying problem, Chadwick et al. describe a conceptual model of the interaction between user and verifier. Two primary trust questions are relevant [4].

1. Are the holders connected to the verifiers that they think they are?
2. Is the verifier a responsible entity i.e. can it be trusted with the personal identifying information (PII) that it asks for?

As stated by Chadwick, the first question is a technical one and can, in online scenarios, easily be answered by existing technologies, like TLS. The second question, however, is about administrative trust and poses problems that are less simple to solve.

Similar to how browsers implement TLS but need to know which Certificate Authorities (CAs) to trust, this challenge also exists for SSI wallets. "We cannot expect the user or wallet to be familiar with all the SSI trust domains in the world, or worse still, which trust domain administrators should be trusted. So should we expect wallets to come pre-configured with the list of trust domain administrators? [...] The implication of this solution is that the wallet software provider becomes the ultimate arbiter of which trust domain administrators are deemed to be trustworthy." [4]

Chadwick also presents a possible solution, comparing this problem with the problems the first web browsers faced in the 1990s. "Web browser suppliers initially each configured their own (different) list of trusted CAs into their browsers and allowed users to add and remove root CAs from this list. Then some of the browser suppliers switched to using the CA trust lists provided by the operating system providers. The CAB-Forum was initiated in 2004 in order to provide a set of rules that CAs should follow in order to become trusted CAs that could be added to such trust lists. Clearly, SSI is just at the start of this journey and is at a comparable stage that X.509 web PKI was at during the 1990s." [4]

It is important to realise that the challenge of determining trustworthiness is greater than the problem we see in browsers. In browsers, CAs are only used to determine if communication is happening to the server that belongs to the domain name that is displayed.[5] It does not provide an answer to the question of trustworthiness and over-asking. That requires an in-depth analysis of compliance with (national) data protection regulations, like the European GDPR [4].

For the leading existing SSI ecosystems, no trust infrastructure for verifiers exists at all. Solving this problem is thus far from straightforward, and there are many steps to take.

[5] In certain sectors, e.g. banking, the CA can also verify the underlying entity offering the service.

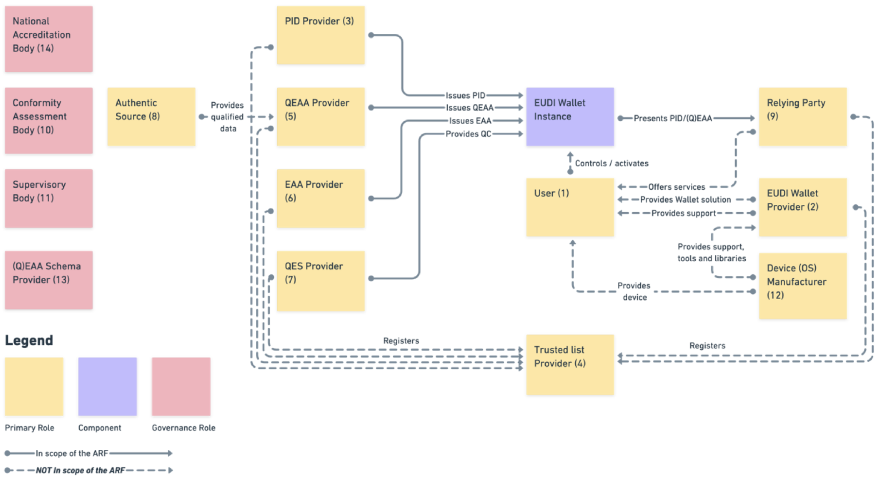


Figure 2.1: Overview of the EUDI Wallet roles [39]

2.2 EU Digital Identity

As presented in section 2.1.1 and section 2.1.2, SSI should be considered as a paradigm to identity management, and no widely accepted specification of a system's architecture and behaviour exists yet. While implementations do exist, standards are being developed, and interoperability is being striven for, this is not yet achieved. However, as mentioned before, in 2020, the European Commission initiated its EU Digital Identity (EUDI) project to create a single European SSI system. By the scale of this initiative, it is interesting to analyse the design decisions that are being made about this product, as they will most definitely become guiding in the industry.

As of now, the technical details of this system are not strictly established, but over a period of multiple years, different proofs-of-concept are to be developed and used in experimental settings to learn about the matter and further define the specifications. In 2022, the first version of the EUDI Architecture and Reference Framework (ARF) [39] was published, a technical document containing requirements about the architecture of the EUDI system. While the current version does not contain details on all matters, it does mention a few interesting observations concerning the topic of this thesis.

Most importantly, the ARF specifically mentions that "mutual authentication" between the wallet and verifier could be required for specific sessions. And while it does explicitly mention that mutual authentication does not need to be mandatory for every transaction, it does mention that some trust framework is required for verifiers. Any further technical details, however, are lacking.

Currently, the Dutch government is developing a demo wallet application in an effort to contribute to the development of the EU wide system[6].

[6]See `https://edi.pleio.nl`.

2.2.1 EUDI Trusted Lists

While the ARF does not specify details on this trust framework, the document does contain some hints. In the description of roles in the ecosystem, also displayed in Figure 2.1, the ARF mentions a primary role for Trusted List Providers (TLPs). Both verifiers (called *relying parties* in the ARF), as well as issuers (called *providers*) and wallet providers, are registered at these TLPs. Though this registration is positioned not in the scope of the ARF, it does mention that: "when used, Trusted Lists need to provide a registration service for the relevant entities, maintain a registry and enable third party access to the registry information. The terms and conditions of entities to become registered are for each registrar to determine unless specified in, e.g., sectoral rules." [39] However, the ARF states that "further precisions to the specifics about how the trusted lists could be implemented will be brought later." [39]

It is important to note that the TLPs in the ARF are considered to have a primary role in the system and not a governance role, such as a National Accreditation Body, Conformity Assessment Body, Supervisory Body or (Q)EAA Schema Provider. It thus seems like an active role is envisioned and that this role will not simply be assigned to a government (Member State). This implies a design where relying parties (and providers) first need to be registered in a trusted list in order to be accepted in the system.

Meanwhile, regarding relying parties, the ARF states that: "To rely on the EUDI Wallet, Relying Parties need to inform the Member State where they are established and their intention for doing so. Relying Parties need to maintain an interface with the EUDI Wallet to request attestations with mutual authentication" [39]. This implies a design where relying parties do not only need to be registered at a TLP but also at the member state itself (while there is no specific role specified for member states in Figure 2.1). The ARF is thus inconclusive on the exact design and states that "Trusted Lists can be implemented in different ways" [39].

2.2.2 eIDAS Trusted Lists

Despite the inconclusiveness in the ARF, it is possible to get a grasp of a plausible infrastructure design. The ideas of Trusted Lists seem to be inspired by the existing eIDAS regulations that are implemented by national eID systems, which also knows Trusted Lists of Trust Service Providers. In this system, every EU Member State publishes an EU-EEA Trusted List (which are all contained in the List of Trusted Lists, LOTL), containing a number of Trust Service Providers that are based in that Member State. These can be both public and private parties and are a form of Certificate Authorities that, in their turn, publish a list of their (root) certificates for different purposes. For example, KPN B.V. is a Dutch trust service provider with a number of certificates used for creating signatures or timestamps. Any party compliant with eIDAS should trust these certificates for their purpose.

The trust service providers are regularly audited by Conformity Assessment Bodies, for which there are standardised procedures in the EU.

For eIDAS, a complete trust infrastructure thus already exists, and despite the

ARF not specifying this yet, it makes sense to simply extend this architecture to the EUDI system as well. There are, however, challenges.

The eIDAS system is nowhere near as popular (with regard to the number of involved parties) as an EU-wide SSI system could potentially become. In order to be certified for eIDAS, extended assessments need to be performed. For all SSI issuers (providers), this might still be feasible, but requiring similar certification for all relying parties wishing to use the system might be unfeasible.

For example, something as fundamental as a digital infrastructure for establishing the identity of relying parties (as legal entities) might not yet exist in all European member states. And many relying parties might not be willing to pay for the costs of certification. There could also be a scenario in which not all relying parties need to be certified by a trust service provider, but only under certain conditions. It thus remains interesting to see how the EUDI system rules will be established in the near future.

2.3 Decision models for over-asking in SSI

Eisenlohr also addresses the topic of over-asking and over-identification in his master thesis [6]. He starts with the notion that determining whether a request is justifiable is not an objective decision (that, for example, could be automated). Therefore, he argues that central authorities should be designated with the task of deciding on this.

"They [...] need to very carefully evaluate if a perceived risk really warrants collecting the attributes an organisation wants to request [...] and carefully design decision models for every possible context" [6]. He then (non-exhaustively) compares different candidates as authorising parties based on values and criteria he defines.

Eisenlohr thus does not state these central authorities should simply accept or reject data requests, but instead, they should create *decision models*. These are formalised models that describe under which circumstances (which context definition for a request) a wallet should decide to allow[7] or deny[8] a data disclosure request. Such decision models can be more expressive than just allowing or denying a certain request and can take into account more of the context of a request.

As implementation for the aforementioned decision model, designed by the authorising party, Eisenlohr proposes a system of decision rules. Here, he builds upon research by Anciaux et al. [40] and Ardagna et al. [41]. Anciaux et al. propose a system of disclosure rules based on allow-listing (stating under which situations certain data is allowed to be disclosed), while Ardagna et al. propose a system of sensitivity labels and categories based on deny-listing (stating when data is too sensitive to be disclosed given a certain context). Verifiers acquire so-called *authorisation certificates* that allow them to request certain data from a governing authority based on a specified decision model.

[7] That is, *conditionally* allow, while still asking for the user's agreement.
[8] That is, either abort with an error or show a warning to the user.

While Eisenlohr does present a proof of concept implementation (inspired by OIDC with W3C Verifiable Credentials[9]) of a system with decision models, even supporting complicated decision models, he does not specify how such decision models could be implemented in a real SSI system's infrastructure with actual authorising parties. Also, it is unclear to what extent Eisenlohr's system is practical in such an infrastructure. This topic that we address in this thesis is thus still open for further research.

2.4 Verifiable Credentials and Verifiable Presentations

Over the past decade, multiple implementations of SSI systems have been developed. As interoperability is a key principle of the SSI paradigm, the need for standardisation of interfaces and protocols between systems and components has been recognised. Verifiable Credentials[10] (of which Verifiable Presentations are a key component) are an example of an open standard that has been developed by the W3C.

Verifiable Credentials (VCs) are designed to be a universal data model for representing credentials in different SSI/ABC systems. The data model is based on JSON-LD and is designed to be interoperable. It is flexible, allowing for different trust models, cryptographic algorithms and signature formats. The standard also defines a data model for Verifiable Presentations (VPs), which are used to present Verifiable Credentials to verifiers. Verifiers should be able to verify the Verifiable Presentation and determine whether the holder's credentials are valid and satisfy the verifier's requirements.

Regarding trust, the Verifiable Credentials standard does not define any trust model. The standard states: "The data model detailed in this specification does not imply a transitive trust model, such as that provided by more traditional Certificate Authority trust models. In the Verifiable Credentials Data Model, a verifier either directly trusts or does not trust an issuer. While it is possible to build transitive trust models using the Verifiable Credentials Data Model, implementers are urged to learn about the security weaknesses introduced by broadly delegating trust in the manner adopted by Certificate Authority systems." [42]

2.4.1 Verifiable Presentation Requests

The W3C standard defines a standardised, universal and interoperable data model for the most important interactions in SSI systems. Verifiable Credentials offer a standardised way of representing credentials during issuance. Verifiable Presentations offer a standardised way of presenting credentials to verifiers during a disclosure session. The W3C, however, has not defined a standard to *request* Verifiable Presentations from a holder, despite this being a key interaction in SSI systems. The VC standard does not specify in what form a verifier should request a VP from a holder. This is somewhat intentional, as the standard is designed to be implemented in existing non-SSI-default protocols and

[9] `https://openid.net/sg/openid4vc/`
[10] `https://www.w3.org/TR/vc-data-model/`

systems (like OpenID Connect), which already have their own way of requesting credentials.

The W3C Community Group has, however, published a draft for Verifiable Presentation Requests[11]. Additionally, the Decentralised Identity Foundation has created a pre-draft for Presentation Exchange (PEX)[12].

Apart from offering a way to define which credentials a verifier requests, these proposals also offer a way to define the reason for the request (as an attempt to include at least a part of the context), which the holder can use to determine whether they want to disclose their credentials to the verifier. This could be a valuable privacy feature, helping the holder to determine whether the request is legitimate (though, obviously, simply stating the reason does not prevent secondary use).

None of these draft standards, however, mention any form of a trust model for verifiers, which would be required for wallet implementations to determine whether a request is legitimate. Any implementations of this (like in Eisenlohr's thesis [6]), are not directly tied to the credential itself but only to the presentation exchange protocol that uses the credential, which is not specific to SSI. There is, thus, no universal solution that tries to solve the problem we address in this thesis. This underlines the immaturity of the industry on this topic and the need for further research.

2.5 Yivi

Yivi, formerly called IRMA[13], is designed as an open-source implementation of an attribute-based credential system, based on the IBM Idemix scheme [22], which was one of the first ABCs designed. It is a minimal implementation of Idemix in the sense that it does not implement all of Idemix's cryptographic features, such as deanonymising organisations or one-show credentials. The core implementation started as a smartcard application but was soon rewritten for smartphones. Currently, the Yivi ecosystem has been in production for several years and is being maintained by the Privacy By Design Foundation[14] (PBDF) and SIDN[15], the organisation responsible for the `.nl` top-level domain.

In the Yivi infrastructure, users have an app on their smartphones that stores all credentials issued to them. Issuers and verifiers both run a dedicated backend server that interacts with the app to perform issuance and disclosure protocols. For issuers, this server is configured with a private key for signing credentials upon issuance. For verifiers, no such key is required. Anyone can be a verifier in the infrastructure.

For further details about the Yivi architecture, we refer to Yivi's technical documentation[16].

[11] `https://w3c-ccg.github.io/vp-request-spec/`
[12] `https://identity.foundation/presentation-exchange/`
[13] IRMA was renamed to Yivi due to legal trademark reasons in 2023.
[14] `https://privacybydesign.foundation`
[15] Stichting Internet Domeinregistratie Nederland, `https://www.sidn.nl`
[16] `https://irma.app/docs/what-is-irma/`

2.5.1 Yivi scheme as trust anchor

Yivi uses a simple scheme as its trust anchor. This scheme[17] contains information about all issuers and credential types in the system, i.e. it contains the public keys from all issuers that are allowed to issue credentials in the system and a description of the form and meaning of these credentials. The client app fetches scheme updates regularly. The scheme is published (both on a web server and in a GitHub repository) by the Privacy by Design Foundation (hence, it is called the `pbdf` scheme) and is signed by SIDN. As such, SIDN is a trusted third party in the ecosystem.

Though there is currently only one production scheme, the underlying Yivi protocol does not require this per se. Multiple schemes can exist. The currently available production built for the client app that is available to users is configured for the `pbdf` production scheme, but in theory, other parties could host their own scheme as well. For example, there is a demo scheme available for testing purposes (and the production built for the app even has a hidden feature available to enable this scheme).

Theoretically, anyone could set up its own trust anchor for Yivi, defining its own scheme with its own issuers, credentials and public keys. The client app, however, is currently not designed to support custom schemes.

2.5.2 Yivi as SSI system

Upon registration of a new wallet, the Yivi client app generates a new key pair for the user. All credentials that are later issued to this wallet are bound to this private key (though not directly, as we will explain next).

For practical reasons, the Yivi protocol involves a keyshare server. The keyshare server also stores a second private key upon registration of an app and protects it with a user PIN. The private key of the server, which is only unlocked after presenting the correct PIN, together with the private key securely stored on the phone, together form the secret key that is actually used for cryptographic operations, like issuance and disclosure of credentials. Requiring a PIN and storing the partial secret key on the keyshare server disables brute force attacks on the user's PIN and allows users to revoke their app in case of loss. The cryptographic protocol is designed in such a way that the keyshare server only learns *when* the user uses their app but does not learn anything about the interactions themselves.

Users can opt to register an email address at the keyshare server (for notification purposes), but further registration is anonymous and uncontrolled. The keyshare server is run as a separate service, and, in theory, anyone could run their own keyshare server.[18] Issuers and verifiers do not interact with the keyshare server; only the app does. As such, anyone can register themselves an unlimited amount of times by creating a secret key, and as such, one could view Yivi as an SSI system according to Allen's principles [12].

[17] `https://privacybydesign.foundation/attribute-index/en/pbdf.html`

[18] An interesting observation is that in the available app, the keyshare server to use is actually specified in the scheme. This could be separated as well in order to give users more freedom in this, as the keyshare server does not need to be globally specified by the trust anchor.

2.5.3 Requestor scheme (pretty verifiers)

In the Yivi ecosystem, any party on the internet can set themselves up as a verifier. Upon interaction with a verifier (at the start of a disclosure session), the Yivi app simply displays the hostname of the server the app is interacting with when asking for consent. For most people, however, the hostname can look ugly or confusing because it highly depends on the technical architecture. For example, `yivi.auth.prod.example.com` could be a regular hostname for *Example BV*. And when a party would outsource its infrastructure to a different party (as we will discuss in more detail in section 3.3.4), `auth.yivi-broker.com` could even be a legitimate hostname, which could be even more confusing.

Moreover, as also mentioned by Chadwick et al. [4], displaying the hostnames results in the same problem as browsers face, with malicious parties *typosquatting* domain names to impersonate well-known parties.

As a countermeasure against this, SIDN offers verifiers the (paid) service of including them in their so-called requestor scheme. SIDN offers this service as *pretty verifiers*. In this scheme, a pretty name for the verifier can be included, as well as custom messages specific to this verifier (e.g. making the app display '*Example BV asks you to login*' instead of '*yivi.auth.prod.example.com wants you to disclose the following attributes: ...*'). This removes possible confusion for users of this verifier, replacing ugly hostnames with a more human-readable name. Typo-squatting, however, is still possible. Typo-squatted domains will not be displayed as a pretty verifier, but while non-pretty-verifiers still exist in the ecosystem and users are used to seeing hostnames at all, users could potentially still consider them as trustworthy.

The requestor schemes are not based on dedicated public keys, like the standard issuer schemes, but simply use the server's hostname. Because Yivi is expected to run over HTTPS (in production), these hostnames are authenticated (using TLS).

The requestor scheme is treated similarly to the regular issuer scheme. The client app fetches scheme updates regularly, is published by the Privacy by Design Foundation (and in a GitHub repository) and is signed by SIDN, making them the trusted authority.

The requestor scheme functionality is not included in the Yivi documentation, but Schraffenberger has written a blog post [43] about it.

2.6 Existing authorisation mechanisms for data disclosure

The challenge of restricting relying parties' access to personal data issued by identifiers is not limited to SSI wallets. A very similar challenge is found in e-passports, or eMRTDs (Machine Readable Travel Documents), as specified by the ICAO (International Civil Aviation Organisation) [44]. This is an example of a successful global standard that practically all passports implement (at least to a certain degree).

Compliant passports contain a digitally signed copy of a person's most essential

identity information issued by the government: their name(s), sex, nationality and date of birth, and often a photo or even fingerprints. This data is stored in several data groups (DGs); DG1 contains so-called MRZ data (e.g. name, sex, nationality, date of birth), while DG2 contains a photograph, DG3 could contain a fingerprint, et cetera.

Different parties, mainly at airports, should be able to read the information stored in these data groups and verify if the person carrying the document is indeed the document holder and if that person is indeed a citizen of the country that issued the document. Passports could, in that sense, be considered a decentral, user-centric IdM solution (similar to SSI), carrying a selected number of credentials issued by the government.

Regarding the disclosure of data (reading from a data group), multiple protocols exist. While we will not discuss the details in this thesis, an important observation is that on most passports, reading the data in DG1 or DG2 does not require any authentication. Cryptographic protocols are in place to ensure physical access to the document is in place and prevent MITM or replay attacks, but those protocols do not authenticate the identity of the relying party reading the data. Anyone is able to read the most elementary data that is stored on a passport (similar to the physical document).

For other data groups that contain more sensitive information, however, like biometric data such as fingerprints or iris scans, relying parties (referred to as *terminals* in this context by the ICAO) do need to authenticate themselves further. This is accomplished by a so-called Terminal Authentication (TA) protocol as a part of what is called Extended Access Control (EAC). This is basically a simple challenge-response mechanism in which the terminal also provides a certificate issued by the government, of which the root key is stored in the passport. Only after this, the passport will allow reading of the restricted data groups. Essentially, this is very similar to an X.509-like PKI, yet in a different form that is optimised for the computational limitations of smartcards.

2.6.1 Germain eIDs

Another important observation is that, while most passports allow for the unauthenticated[19] reading of DG1 and 2 of a passport, this is not the case for *all* passports. Specifically, German eIDs have implemented additional measures to protect those data groups. Concretely, German eIDs *always* require terminal authentication in order to read any data group of the document, not only for the more sensitive data groups [45].

While, on the one hand, this measure prevents possible unauthorised parties from reading personal data, it remains questionable how effective this measure really is. As already mentioned, other protocols ensure physical access to the document for communication with the document to be possible at all, and the MRZ data is also physically printed on the document. Yet, this approach *does* require all parties that require cryptographically signed identity information to register themselves at the government and request a certificate for this purpose.

[19] That is, without requiring EAC.

Meanwhile, this requirement does complicate the technical infrastructure for terminals. With regular e-passports, any device can read the contents of DG1 and DG2, but for German e-passports, these terminals need to be certified and configured with a dedicated certificate. While this has not been researched, one could argue that this more complex infrastructure is an important reason for the German eID(AS) system, which is based on reading these passports, lacking adoption [46].

To use the Germain eID(AS) system, users need to present their e-passport to a USB NFC reader or their (NFC capable) phone. In itself, this is a standard solution that many countries implement. For German eIDs, however, devices first need to get hold of a certificate allowing them to read the data on the passport. For regular citizens wishing to read the contents of their own documents, an automated procedure to register your reader at a government service is implemented, but still, this is a complicated step that makes users drop out. For actual relying parties wishing to use the data read from a passport, there is a more complicated registration procedure, requiring them to set up dedicated eID Servers with HSMs (Hardware Security Modules). This makes it an expensive process.

We consider this a good example to address the importance of easy adoption of a system for verifiers. When the costs of registration in the system are too high or key management is too complicated, parties might simply not use the system.

Chapter 3

Protected attributes

Previously, we have identified that in current SSI system designs, any party can request disclosure of any attribute from a user. This is a problem because requestors may not always be entitled to receive these attributes (think of the Dutch BSN, certain health care credentials). Currently, users are expected to identify when this is the case themselves and simply abort the session in such cases. However, in section 1.1, we have presented several reasons why this is not a realistic scenario.

One approach to solve this problem is with a system that controls which requestors can access (request) specific attributes. Requestors will need to acquire authorisation to be able to request these attributes. We will call such attributes with access control **protected attributes**.

As we will see, protected attributes are especially suitable for simple scenarios where specific (sensitive) attributes require additional protection while the data is in a user's wallet. The approach is centred around attributes and, thus, around the issuer.

It is probably desirable for the authorisation rules of this system to *supersede* the user's consent so that the user is not able to disclose attributes to requestors that are not authorised to access them, even if the user would want to do so, because of the reasons mentioned earlier (ignorance, unawareness or coercion).[1] Meanwhile, of course, the user should still be able to deny access to authorised requestors to access certain attributes, so this system should not *replace* the user's consent.

First, we will discuss several approaches to different system designs for authentication and authorisation (section 3.1, 3.2 and 3.3), considering multiple parties for deciding on these authorisations (section 3.4) and related aspects and design decisions (section 3.5). In section 3.6, we then propose a specific solution for the Yivi ecosystem. Finally, in section 3.7, we make some critical remarks and highlight possible limitations (and solutions to these limitations).

[1] In section 3.5.3, we will discuss this matter in more detail.

3.1 System designs

There are different approaches to designing a system's infrastructure with access control and authorisations. Multiple system designs can be interesting:

1. a simple (X.509-like) **dedicated public key infrastructure (PKI)**, where requestors receive a certificate from a certificate authority and prove their identity to the user/wallet with a simple challenge-response mechanism. This PKI, in its turn, can be implemented in different ways:
   (a) For every protected attribute, a dedicated root certificate is defined. Requestors receive a dedicated certificate from this root for every protected attribute they are authorised to access.
   (b) A single trusted certificate authority issues certificates that include which protected attributes a requestor is authorised to access.
   (c) A single trusted certificate authority issues certificates on the identity of a requestor. Those are used to authenticate a requestor. Additionally, other methods are in place to authorise requestors to request certain protected attributes.
2. rely for authentication on a **lower-level protocol** (like TLS) and implement methods for authorisation based on these lower-level established identities.
3. requestor authorisation can be **attribute-based** too: issuers of protected attributes could, in addition to issuing AttributeX to users, issue an MayAccessAttributeX to requestors that they need to disclose upon requesting AttributeX. In this scenario, verifiers would become users with a wallet, too.

Requestor anonymity The latter option (attribute-based), we do not consider valuable. The unique features of this design would be anonymity and unlinkability *for the requestor*, which, as mentioned in chapter 2, we consider undesirable. For a user to give informed consent on attribute disclosure, the requestor should **not** be anonymous.

PKI certificate revocation For all dedicated PKI-based methods, note that the PKI should also implement revocation mechanisms so that certificates can be revoked when a requestor is no longer authorised to access a specific attribute. This can be challenging because the revocation mechanism should not be *online*: the wallet should not leak to any party *when* an attribute is requested. Hence, no revocation lists can be retrieved *at the time* of request. This should instead happen in the background or in the case of Yivi, during communication with the keyshare server.

This is also true for option 2 that relies on a lower-level protocol. Here, authentication of the requestor in the lower-level protocol should not leak to any party *when* an attribute is requested.

3.2 Authorisation methods

In system designs 1a and 1b, authorisation is included in the certificate infrastructure itself: the certificate itself forms the authorisation. For 1c and 2, however, the requestor's identity is only *authenticated*. In those architectures, an additional authorisation system is required to determine if a requestor is authorised to access a specific attribute based on the already established (authenticated) identity. This can be implemented in different ways:

1. **inside the credential**, for every protected attribute, apart from its value, the wallet also stores a list of requestors that are authorised to access it. This list is generated during the issuance of the credential and is thus signed by the issuer. When a requestor requests an attribute, the wallet checks if the requestor is on the list.
2. every wallet keeps a **central scheme** that contains a list of authorised requestors to access a certain attribute.
3. **issuers** expose a service to **announce** a current list of authorised requestors for a certain attribute, for example, by exposing some web API that returns a list of authorised requestors for a certain attribute.

3.2.1 Accumulator-based methods

So far, all authorisation methods described above use allow-lists of authorised requestors. This approach, however, obviously is not scalable. The size of this list could become very problematic for attributes like the Dutch BSN, which should be protected, yet would have a large number of authorised requestors.

Instead of storing a list of authorised requestors, we can use accumulator-based methods to be more efficient.

A similar concept is also used to implement revocation in Yivi (though there, a deny-listing approach is used): so-called revocation servers keep a list of issued credentials and can mark credentials as revoked. During disclosure, wallets can efficiently prove non-revocation (non-membership).

For requestor authorisation, a similar setup could be used. Using accumulators, instead of storing a list of requestors, a single accumulator value is stored that describes a set of requestor IDs. When a requestor requests an attribute, the wallet checks if the requestor is included in the accumulator. For further explanation on accumulators, we refer to work by, among others, Baldimtsi et al. [47].

Accumulators can, in principle, replace allow-lists in all three aforementioned authorisation methods: an accumulator value could be stored inside a credential or in a central scheme. The most fitting implementation, however, would be to implement accumulators in an issuer-announced authorisation architecture with an authorisation server that exposes a current authorisation value. We discuss the implementation later in section 3.6.3.

Though it is possible with this method to let verifiers *anonymously* prove that they are authorised (proof membership of the accumulator), this is not required for the authorisation system, as the requestor should not be anonymous (as

discussed earlier in section 3.1). We would only use accumulators for efficiency, not for anonymity.

3.2.2 Privacy-preserving issuer-announced authorisations

The aforementioned issuer-announced authorisation method, when implemented naively, leaks information to the issuer (the issuer knows when a requestor is requesting an attribute). However, this communication can be made unlinkable using the same infrastructure as for attribute revocation. Issuers could broadcast authorisation updates, and wallets could, in the background, regularly check for updates, updating their local authorisation lists. This way, the issuer does not know when a requestor is requesting an attribute.

3.3 Comparing different system designs

In the previous sections, we have listed several approaches for implementing access control for protected attributes. These architectures have different characteristics and trade-offs. In this section, we compare these architectures on several criteria. An overview of the comparison is given in Table 3.1 at the end of this section.

3.3.1 Modifiability

Requirement 1 (Modifiability) *Authorisations can easily be updated after issuance of the credential (e.g. new organisations can be added to or removed from the set of authorised parties without the user having to get a new credential issued).*

On the one hand, one could argue that authorisations should be immutable in order to preserve purpose limitation: after issuance of a credential that is restricted from being accessed by a specified set of entities, it should not be possible to suddenly add new entities to this set. One could thus argue that modifiability of authorisations should be undesirable.

On the other hand, there are many scenarios where authorisations should be modifiable. Consider the BSN example: new organisations would need to be added to the list of authorised requestors all the time because of the sheer number of organisations that are authorised to access the BSN. If authorisations are not modifiable, the user must get a new credential issued every time a new organisation is added to the list of authorised requestors. This would be impractical and bad for usability, especially considering users' expectations of the platform. Therefore, we argue that for most scenarios, the practicality of modifiable authorisations outweighs the argument of preserving purpose limitation.

An in-credential authorisation system is not modifiable, as changing authorisations requires re-issuance of the credential. Issuer-announced authorisations (via a web API) are obviously easily modifiable. Scheme authorisations are also modifiable but do require scheme updates, so modification can be expensive.

3.3.2 Scalability

Requirement 2 (Scalability) *The number of authorised requestors can be large without affecting the system's performance.*

Using an allow-list of authorised requestors is not scalable, making only a dedicated PKI (with direct authorisation) suitable for large-scale usage. Implementing the accumulator-based method as described in section 3.2.1, however, solves the scalability issue for other methods (see also section 3.3.5). Additionally, depending on the exact use case, for some credentials, the number of authorised requestors could be so low that scalability does not need to be a problem.

3.3.3 Granularity

Until this point, we have discussed cases where a requestor is authorised to access a certain attribute or not. For example, a government agency might be authorised to access a user's BSN, but a webshop is not. However, there are more granular authorisation schemes possible. For example, medical professionals might be authorised to access medical data for *their own* patients only. There could thus be scenarios where the fact that a requestor is authorised to access a specific attribute cannot be globally determined but should differ per individually issued credential.

Requirement 3 (Granularity) *Authorisations can differ per issued credential (e.g. a requestor is authorised to access a specific attribute for one person but not for another).*

This scenario can only be implemented using authorisation method 1 with the authorised requestors stored inside the credential. The other methods do not allow for this level of granularity, as authorisations are globally defined. For this reason, this property has an inherent conflict with the property of transparency: if authorisations are transparent, they are globally defined and thus cannot differ per credential.[2]

Arguably, this level of granularity might not be necessary for most use cases. Whenever such a fine-grained level of access control is required, an SSI ecosystem might not be the best system for data processing whatsoever. For completeness of this comparison, however, we do include this property in the list to describe the expressiveness of all possible solutions properly.

3.3.4 Authorisation context

So far, we have described authorisation as a binary concept: either a requestor is authorised to access a certain attribute or not. As also concluded in Eisenlohr's thesis [6], however, authorisations could be seen as (possibly complicated) decision models that consist of decision rules, taking many inputs. In practice, there could thus be more complicated scenarios in which it is essential to define the context of an authorisation (as we have also mentioned in section 1.1.5).

[2]Strictly speaking, it could be possible to implement multiple sets of globally defined authorisations, essentially allowing for a certain degree of granularity, but this would be very unpractical. Practically, this would be the equivalent of defining the same credential type multiple times, each time with different authorisations.

We will describe this problem with a very practical example of so-called *brokers*.

Brokers in Yivi

Within the Yivi ecosystem, some (relying) parties do not implement their infrastructure themselves. Instead, they outsource it to a third party acting on their behalf, which we will call a broker.

Consider the case that organisation A outsources its authentication infrastructure to broker X. Organisation A should be authorised to request a specific attribute, and thus, broker X is added to the authorised requestors. Now consider Organisation B, which also outsources its infrastructure to broker X. Even though Organisation B is not authorised, they could, if broker X were not to prevent this themselves, use the authorisation for Organisation A to request attributes they are not allowed to request. The responsibility for properly using the authorisations is thus placed at broker X rather than the authorising party itself.

We see the same problem in a different context without brokers, too. Consider a large retail company using SSI for both its Human Resources department and its retail activities. For the Human Resources activities, it could be very well possible that this company is authorised to process the BSNs of its employees. For their retail activities, though, they obviously are not. The fact that an organisation is authorised to access certain attributes can thus be very context-dependent.

In order to prevent this type of problem, authorisations should not only cover the authorised party itself but also the broader context of the authorisation should be defined: for what purpose can the data be requested?

This context could be defined in natural language but could also be defined more formally, for example, using a set of other attributes that should be present in the credential, together with a more formally defined decision model.

For a context in natural language, the wallet application could simply display the description of the context to the user, who can then decide whether to share the attribute or not (e.g. '*Company X is asking to access your BSN* ***for Human Resources purposes***').

We argue that such a context in natural language, describing the purpose of data processing and the type of transaction that the user is performing, should always be included and displayed to the user for verification. This would alert the user if an authorisation is (ab)used out of context.

Though the context here cannot be automatically verified nor enforced by the wallet application, it does provide the user with the information needed to make a more informed decision and misuse of an authorisation (issued for a different context) can be noticed by the user. A screen with a notification as described above, would probably alert regular retail customers of the company that something is not right, even though they might not be aware of the sensitivity of the requested attribute itself.

Additionally, we could try to define a more formalised decision model for authorisation more formally, so the wallet application could automatically verify the context of the request. In the example above, we could, for example, require the presence of a credential in the wallet stating that company X is indeed the user's employer. However, we consider further research into this topic to be out of the scope of this thesis.

Requirement 4 (Context definition) *For every authorisation, it is possible to define a context description.*

Implementation of this property makes proving authorisation more complex than just checking membership of a specific requestor ID in an allow-list. For PKI-based methods, the authorisation context could be included in the certificate. For accumulator-based methods, the only implementation would be to make a mapping of a whole requestor ID and context definition and include that in the accumulator. Requestors would need to announce the context for which their requests are authorised, and the wallet application would then need to verify if that context definition, together with the requestor ID, is actually included in the accumulator. We consider this implementation not elegant, but it is the only way to implement this property using accumulator-based methods.

3.3.5 Transparency

Requirement 5 (Transparency) *Authorisations can be publicly reviewed (e.g. anyone can see who is authorised to access a protected attribute).*

In certain circumstances, it could be desirable for authorisations to be transparent (e.g. for governance reasons or when commercial third parties are the authorising party). In a transparent system, anyone should be able to verify which parties are authorised to access specific protected attributes.

How important transparency is, depends on the entity responsible for determining (cryptographically signing) the authorisations and to what extent this party can be trusted. We will discuss this in more detail in section 3.4.

Transparency and accumulators In an accumulator-based system, though the accumulator value can be public, it is impossible to enumerate the list of authorised requestors solely based on the accumulator value. To make this transparent, the issuer would still have to publish the list of authorised requestors in some way so that anyone can verify that the list of authorised requestors is indeed valid and corresponds with the accumulator value. Not every wallet would need to actually process this list, though, so this would not break scalability.

3.3.6 Overview

A comparison of the different authorisation methods in different access control system designs on the aforementioned properties is listed in Table 3.1. We refer to the previous text for a more nuanced description of the properties.

Method	Modifiability	Transparency	Granularity	Scalability	Context
Dedicated PKI[3]	++	−	−	++	++
In-credential	−	−	+	−	+
In-credential (with accumulators)	−	−	+	+	−
Central scheme	+[4]	++	−	−	+
Central scheme (with accumulators)	+	+	−	+	−
Issuer announced	++	+	−	−	+
Issuer announced (with accumulators)	++	+	−	+	−

Table 3.1: Comparison of different access control system designs

3.4 Authoriser candidates

In the previous sections, we have discussed different methods of authorisation of requestors: how to define that a requestor is authorised to request a protected attribute. We have not explicitly discussed *who* this party should be: who decides which requestors are authorised to request a protected attribute and how they should make these decisions. This section will briefly discuss the different options for the authorising party regarding this solution and relevant considerations related to this topic.

This quickly touches upon legal, ethical, economic and governance topics. For example, the question of rights and obligations towards personal information (who *'owns'* personal information) has been listed as one of the four ethical questions of the information age [48]. However, we will not go into too much detail on these aspects, as this is out of scope for this thesis.

Obviously, not all choices are compatible with all system designs, but for most architectures, different decisions can be made. We will not elaborate on the exact way of implementing the procedures.

3.4.1 Issuer as authoriser

Perhaps the most obvious choice for the authorising party, at least from a technological perspective, is the issuer of the credential. As an issuer, you already have a lot of duties and responsibilities in the SSI system, and thus, it would be logical also to include the authorisation of requestors in these. The issuer has the most knowledge about the data and might thus be the most suitable party to decide who should be authorised to access the attributes in the credential.

Moreover, one could argue that the issuer also has the most interest and even perhaps a (legal) responsibility or duty to ensure that the attributes they issue are adequately protected. As a data controller/processor, under the GDPR, you have obligations to protect the data adequately. SSI wallets are no exception to this, and one could argue that this obligation still applies when the data is

[3]Specifically, the methods where authorisation also happens via the PKI, being 1b and 1a as discussed in section 3.1. Method 1c does not use the dedicated PKI itself for authorisation and thus does not fall under this category for the purpose of this comparison.

[4]Requires a scheme update, which is feasible but does not scale well.

issued and resides in a user's SSI wallet. Defining authorisations would be a way to enforce this.

For certain kinds of attributes, especially those where the issuer has a specific interest in protecting the data as well, the issuer would be the best candidate for authoriser. However, for many other (more general) kinds of attributes, the issuer might not be willing to take on this responsibility.

3.4.2 Scheme manager (wallet provider) as authoriser

The issuer is not the only party that could be made responsible for the authorisation of requestors. At least in the Yivi ecosystem, the scheme manager is responsible for defining the scheme and (by that means) allowing parties (issuers) access to the system and could also be assigned this role. The scheme manager already has some form of gatekeeper role in the ecosystem and, for example, also decides on which credentials can be issued and their form and meaning (though the wishes of the issuer are generally followed).

Arguments in favour of this approach are that the scheme manager is considered to be a trusted, neutral party. No economic incentives should be involved in the scheme manager's role, and thus, they are less likely to abuse their power. This is especially important in ecosystems where the issuer is a commercial party, as they might be tempted to authorise only requestors willing to pay for it. We will discuss this topic in more detail in section 3.5.2.

Apart from that, depending on the system design, the scheme manager could have better knowledge about the requestors/verifiers in the ecosystem and thus be better suited to decide who should be authorised.

However, a disadvantage of this approach is that it might not scale well. Keeping track of all the authorisations might be too much work for the scheme manager, especially in large ecosystems.

3.4.3 Separate authorisation party

Finally, it is also possible to have a separate party responsible for authorising requestors. This party could be a trusted, neutral party, like the scheme manager, but it could also be a commercial party. The same arguments as in section 3.4.2 apply in this case. For different credentials, however, different parties could be assigned, so there will not be a single party responsible for the whole burden of managing all authorisations.

3.4.4 Multiple designs

It could be desirable that for different kinds of attributes, different kinds of authorising parties would need to be defined. For more general, legally protected attributes, like the Dutch BSN that may only be processed by selected organisations for selected purposes, the scheme manager or a third party seems to be a good fit for the authoriser, while for certain more specific attributes, the issuer might be the best fit. Ideally, the SSI system allows for multiple models.

3.5 Finding a balance between data portability and privacy

Though implementing a system with the described features might not be challenging from a purely technological perspective, the consequences for the whole SSI ecosystem can be large. One of the fundamental principles of SSI wallets is portability [12]. While on the one hand, we want to implement a system preventing users from sharing attributes with parties they should not share them with; on the other hand, users should be able to share data with whoever they want, without anyone (e.g. the issuers) preventing it.

A completely open system (where anyone can request all attributes) is great for data portability but has privacy risks, as described above. On the other hand, a completely closed system, e.g. one where each credential can only be requested by a minimal number of requestors, cannot be considered genuinely self-sovereign. A balance must thus be found.

We will further illustrate this problem and possible considerations below.

3.5.1 Preventing a closed system

As described above, the issuer is the most practical choice for the authorising party. However, this gives issuers quite some power: they could simply close the system by protecting all credentials they issue and charge potential requestors money to be authorised. To safeguard the fundamental principles of SSI ecosystems, we argue that this should be prevented. This is an important consideration to make when designing a system. Protected attributes should add extra security to an SSI system but not violate its other principles. This can be a reason not to give issuers complete control over defining whether attributes should be protected and who could access them in those cases.

3.5.2 A new business model: verifier pays

Having the issuer fully decide on who is authorised to request an attribute could, on the other hand, also enable a new business model for issuers. In the current ecosystem, attributes are issued to users either for free (the vast majority) or paid by the user (for example, the *Kamer van Koophandel* credential, which costs a couple of euros, similar to requesting a traditional excerpt). Apart from issuer-pays and user-pays credentials, an ecosystem with authorised requestors would enable a third model: verifier-pays. Issuers freely issue credentials to users, but for a verifier to request them, they need to be authorised by the issuer, which requires them to close a contract.

A possible use case could be the financial or insurance sector with compliance checks. Parties like *Stichting CIS* or *Justis* already do compliance checks on individuals to determine whether they do not appear on international financial sanction lists or do not have criminal records. Such certificates already exist at present and can be pretty expensive (the costs of a Dutch VOG are 30 to 45 euros, and a compliance check can cost up to a few euros as well). Often, the party requiring such a claim pays for the research to be conducted.

One could imagine a situation where some relying party X (say, an insurance company) closes a contract with organisation Y, paying them an annual fee and, in exchange, be authorised to request the 'PassesComplianceCheck' attributes the issue. Other companies that do not pay the annual fee cannot request those claims. A system like this could be an important economic motivation for issuers to actually issue attributes to users (for free) and thus be good for the adoption of the platform.

This requires a highly modifiable system architecture where adding and removing authorised requestors is easy. Note that no architecture discussed before allows for pay-per-usage, where a verifier pays for every individual transaction, as this would break unlinkability.

3.5.3 Permissive or strict wallets

One consideration when designing this system is whether unauthorised requestors should result in a full abort by the wallet application or merely a warning displayed to the user. Chadwick et al. call this a strict or permissive wallet [4].

It might be desirable to implement a way to bypass access restrictions in the app, for example, by having a specific setting allowing for unsafe attribute disclosures and/or having the app show a warning: '*Party X has indicated only restricted parties are allowed to request this data about you. Party Y is not included on that list. Do you still want to proceed sharing this data with Party Y? (y/N)*'.

On the one hand, a warning would greatly help portability and user control, though, in a situation where there is a power imbalance between the user and requestor, only a full abort would work, as users could simply be instructed to 'skip the warning that you will see on your phone'. For both behaviours, a case is to be made. The exact user interface design is highly relevant for these cases. We will further discuss this in section 5.1.1.

3.5.4 Grounds for restricting an attribute

As described above, there is a significant challenge in maintaining an open system while implementing protected attributes. In order to keep a grip on this, a significant factor is determining which attributes should be restricted at all and which should be unrestricted.

While making this decision, it is crucial to keep in mind the ultimate goal of the restriction. Our main objective is to improve the user's privacy: preventing them from sharing attributes with parties they should not share them with (due to thoughtlessness or power imbalance). We, however, also identified that in very specific scenarios, economic arguments could be legitimately made. There is an important role for a scheme manager to make a clear policy on this.

Though we acknowledge that economic arguments can exist, we think those can be hard to judge. Therefore, we argue that these arguments should be made on legal grounds. That is, only attributes can be restricted if there is a national law explicitly restricting the processing or special types of 'personal data' under the definition of the GDPR. This is a clear and (reasonably) objective rule that the scheme manager can easily implement.

3.6 Implementation in the Yivi ecosystem

In the previous sections, we discussed several system architectures for SSI ecosystems to introduce protected attributes and the considerations to make when designing such a system. While being inspired by the Yivi ecosystem, we have tried to keep the discussion as general as possible to make it applicable to other ecosystems as well (though the concept of a scheme manager is very specific to Yivi). This section will discuss how the Yivi ecosystem could implement such functionality in further detail.

3.6.1 Authentication using TLS

In the previous, several system architectures were discussed. We have seen that a dedicated PKI has many advantages in expressiveness: it is easy to modify which requestors are authorised, define the authorisation context, and scales well. However, setting up and maintaining a dedicated PKI for this sole purpose is also a lot of work (for both requestors and the authorising party!). Keys need to be managed; authorisation certificates must be issued and revoked, all without leaking any information about the user. Implementation of a requestor PKI would take a lot of work.

An important observation to make is that the Yivi ecosystem already makes use of TLS. Sessions occur between a device (wallet application) and a Yivi server on the internet. The server's hostname (at least when developer mode is not enabled) is already authenticated over TLS. As long as we maintain this assumption (that sessions run over HTTPS), wallets can simply use the TLS hostname for authenticating requestors. This is a straightforward solution for authenticating requestors, as it does not require any additional infrastructure and is actually already implemented in the Yivi ecosystem.

Yivi already implements so-called '*pretty verifiers*', where verifiers can display a human-readable name and logo in the Yivi app instead of their hostname (to improve the user experience). This is implemented via a requestor scheme, where the hostname of the requestor is used for identification. We can simply use the same scheme for requestor authentication in the context of protected attributes.

3.6.2 Scheme-based authorisation

In line with how 'pretty verifiers' are implemented in Yivi, using a requestor scheme for authentication, we can also implement authorisation of requestors referring to this scheme.

In fact, we propose to use a two-level scheme. At the first level, the **requestor scheme** is used to authenticate a requestor, mapping their hostname to some requestor ID. This should be the responsibility of the scheme manager SIDN, which also already does this for Yivi pretty verifiers (though it could be a highly automated procedure, as we will further investigate in section 4.4.1). At the second level, the **issuer scheme** (in Yivi also simply referred to as '*the scheme*') is used to define the authorisation policy, mapping to requestor IDs from the requestor scheme.

This split has the advantage that whenever a requestor entity changes its hostname, only the requestor scheme must be updated without updating the issuer scheme. The scheme manager of the requestor scheme is responsible for verifying that the hostnames actually belong to the requestor.

```
<IssueSpecification version="...">
    ...
    <Attributes>
        <Attribute id="BSN">
            <Name>
                <en>Burgerservicenummer</en>
                <nl>Social security number</nl>
            </Name>
            ...
            <AuthorisedRequestors>
                <RequestorID>
                    pbdf-requestors.someauthorisedparty
                </RequestorID>
            </AuthorisedRequestors>
        </Attribute>
        ...
    </Attributes>
</IssueSpecification>
```

Listing 3.1: Example (partial) Yivi scheme for a BSN as protected attribute.

```
[
    ...
    {
        "id": "pbdf-requestors.someauthorisedparty",
        "name": {
            "en": "Example requestor",
            "nl": "Voorbeeld requestor"
        },
        "hostnames": [
            "authorised-requestor.example.com"
        ],
    },
    ...
]
```

Listing 3.2: Example (partial) Yivi requestor scheme referred to by Listing 3.1.

Required changes As mentioned before, the requestor scheme is already implemented in the Yivi ecosystem and can be used in exactly its current form. The issuer scheme, however, needs to be extended to support authorisation policies. We propose to add a new field to the issuer scheme, called `AuthorisedRequestors`. This field is a list of requestor IDs that refer to the

requestor scheme, defining which requestors are authorised to request that attribute. If the field is not present, the attribute is not protected and can be requested by any requestor. If the field is present but empty, the attribute should not be requestable by any requestor.

Features and limitations The benefit of this approach is that it is straightforward to implement. Almost no changes are required to the current Yivi ecosystem, except for adding an optional field to the issuer scheme. We do, however, acknowledge that this approach does not scale well. Modifying the authorisations of a single attribute requires updating the issuer scheme of that attribute, which can be expensive when there are many authorisations. Also, the requestor scheme does not scale well. In SSI ecosystems, the number of requestors is expected to be much higher than the number of issuers. The proposed implementation requires every requestor to be defined in the requestor scheme, which is not scalable.

Issuer-signed credential scheme An important observation is that in the current scheme, issuers do not sign the scheme of the credentials they issue themselves. The scheme is only signed by the scheme manager. As the scheme manager and issuer have separate agreements, this is not necessarily a big problem, though it would be elegant if the issuer were to sign the scheme of their own credentials as well. If issuers are to become the party responsible for authorising requestors' access to their own protected attributes and authorisations are to be defined in the scheme, this would even be more elegant. However, this would be relatively easy to implement in the existing infrastructure.

3.6.3 Issuer-announced authorisation

Above, we have proposed to define authorisations in the issuer scheme. As already mentioned, however, this approach does not scale well when there are a lot of credentials with authorisations that need to be frequently updated. As a better scalable alternative, issuer-announced authorisations can also be easily implemented in the Yivi ecosystem. Both systems can be implemented with relative ease and can coexist. We thus propose to implement both ways of defining authorisations, as they both might serve subtly different use cases.

For revocation of attributes, Yivi already knows so-called '*revocation servers*', which are servers that can be queried to check if an attribute is still valid. We propose to extend this concept to also include '*authorisation servers*'. An authorisation server is a server that can be queried to check if a requestor is authorised to request a particular attribute. The issuer can host this server, but it can technically also be hosted by a third party.

The advantage of this approach is that it scales much better while still being relatively easy to implement, as a lot of the infrastructure is already in place.

Concretely, protected attributes would get an `AuthorisationServer` defined in the scheme. Similar to the infrastructure for revocation, this server publishes accumulator values for the authorised requestor IDs. For transparency, this server would also need to make actual allow-lists available so people can verify which organisations are authorised (see section 3.3.5).

```
<IssueSpecification version="...">
    ...
    <Attributes>
        <Attribute id="BSN">
            <Name>
                <en>Burgerservicenummer</en>
                <nl>Social security number</nl>
            </Name>
            ...
            <AuthorisationServers>
                <AuthorisationServer>
                    https://issuer.example.com/yivi/AuthorisationServer
                </AuthorisationServer>
            </AuthorisationServer>
        </Attribute>
        ...
    </Attributes>
</IssueSpecification>
```

Listing 3.3: Example (partial) Yivi scheme for issuer-announced authorisations.

3.7 Certified wallets

So far, we have discussed solutions in which *the wallet application* is responsible for enforcing the authorisation policy. It is, however, important to note that in decentralised and open ecosystems (as fundamental to SSI), we have no guarantees on the wallet application that a user is using. In the Yivi ecosystem, users can use any wallet application supporting the Yivi protocol. Currently, such alternative applications do not exist, but nothing prevents them from being developed. If such an application were to be developed, it could ignore the authorisation policy and simply always send the attribute to the requestor.

At first, one needs to determine whether to consider this a problem at all. If people choose to use a different wallet application, they simply choose lower privacy guarantees. When implementing protected attributes purely for the user's own benefit, this should not be too much of a problem.

On the other hand, when we consider the issuer perspective, with protected attributes serving their legitimate interests, this behaviour would be undesirable. In fact, if issuers need the *guarantee* that users cannot violate the authorisation policies, the proposed solutions do **not** work.

To fix this, we must introduce the concept of **certified wallets**. The software of a wallet application would need to be certified by some central authority to ensure that it enforces the authorisation policy. The phone running the application could then prove that it is running the certified software. This implementation is rather straightforward in current software development. However, it does conflict with open software principles, which are also fundamental to SSI.

This proof would need to be sent either to the issuer (only at issuance) or (in

the Yivi ecosystem) to the keyshare server, which can then verify the proof and decide whether to send the attribute to the requestor or not (in which case the issuer needs to trust the keyshare server, too). This can be included in the existing keyshare protocol that Yivi uses. We emphasise that, while perhaps intuitive, it makes no sense to provide this proof to the requestor, as the requestor is the party that is not trusted in this scenario (they are the ones that are not allowed to receive the attribute).

As we can see, more complex technical changes to the infrastructure are required to fully enforce this policy and fully disable users from disclosing protected attributes in the scenario where users can create their own wallet application.

Chapter 4

Certified disclosure requests

In chapter 3, we discussed one method for implementing access control on the disclosure of what we called *protected attributes* in SSI wallets. This system is suitable for specific, highly sensitive attributes where it is feasible (either for issuers, the scheme manager or a third party) to establish a list of authorised requestors for that specific attribute.

Though this system is robust and a proper solution for specific cases, it cannot solve the problem of over-asking to its full extent. This is simply because it is not possible for all attributes to establish a list of their authorised requestors and the context in which a party should be allowed to request them. In fact, *most* attributes should be requestable by numerous parties. Yet, over-asking can still occur in these cases.

In this chapter, we will first (section 4.1) further analyse the situations that our proposal of protected attributes cannot solve, working towards a second solution that *is* able to prevent over-asking generally, yet requires a higher administrative burden (section 4.2 and 4.3). Finally, we again propose an implementation for the Yivi ecosystem in section 4.4.

4.1 Over-asking of non-sensitive attributes

Consider an online bookstore requesting your postal address and email address. Both attributes are not sensitive. One could argue, however, that only the postal address is actually required here to fulfil your order. Requesting the email address attribute could thus be considered over-asking.

To fully prevent this form of over-asking as well, we thus need to implement some form of access control on *every attribute*. As also discussed in section 3.3.4, we have seen that the context of a request is crucial for determining whether the request is justifiable. We see a similar thing here. While maybe the bookstore requesting your email address could be justifiable *in specific contexts* (perhaps, for marketing purposes or customer service), it could be *not justifiable* in the context of simply placing an order.

Instead of authorising a *party* to have access to specific attributes, in this scenario, it would be better to define for a party's complete *disclosure request*, including the context of that request, whether it is allowed. For example, the online bookstore that uses Yivi for placing orders might have authorisations for (1) requesting bank accounts and postal addresses for the fulfillment of orders, and (2) requesting email addresses for marketing purposes.

Our previous approach with protected attributes was credential-centric: for every credential, its authorised requestors are defined (together with the authorisation context). This works well for specific attributes, where the number of requestors is low and can easily be defined, or when the issuer has an important interest in the protection of the data as well.

For other credentials where this is not the case, a requestor-centric approach can be implemented more easily: defining for every requestor which requests this party is allowed to do. We will call such requests **certified disclosure requests**. This approach works better considering that SSI typically has many requestors and few issuers.

4.2 System designs

Similar to the credential-centric approach we discussed in the previous chapter on protected attributes, there are different system designs possible. In fact, the possible designs are very similar to the ones we discussed in the previous chapter, though there are fewer ad-hoc variations. In this section, we will discuss these different system designs and their advantages and disadvantages.

- First, again, a simple (X.509-like) **dedicated public key infrastructure (PKI)** can be implemented. In this case, however, there should just be a single central authority issuing certificates to requestors (instead of the theoretic possibility of having a different PKI per attribute). Every requestor receives a certificate on their identity that also includes *the request(s)* this party is allowed to perform (and in which context). These certificates are issued by a central authority that is responsible for the authorisations of all requestors in the system.
- Alternatively, we can again rely on a **lower-level protocol** (like TLS) for authentication of requestors. Authorisation is then done via a central scheme that includes the *certified disclosure requests* a requestor is allowed to make.

The first method is the most scalable since requestors carry their own certificates. Adding a new requestor to the system is easy since the central authority only needs to issue a certificate to this party. Certificate revocation is a bit more tricky (as timing may not leak information about disclosure sessions) but definitely feasible with existing solutions for PKIs, such as certificate revocation lists (CRLs).

In the second method, the central scheme needs to include the certified disclosure requests for every requestor, which does not scale well, especially considering that there are many requestors and few issuers in SSI ecosystems. The benefit,

however, is again that by relying on a lower-level protocol, we can use existing solutions for authentication that do not require key management.

Additionally, the second method could be considered to be more transparent since the authorisations are included in the scheme. In the first method, the authorisations are included in the certificates, which are not necessarily public. We would need to trust the central authority that issues the certificates. This is not necessarily a problem, but normative arguments can be made that this might be a desirable feature. As we will see in section 4.3.2, this transparency might even be an essential feature to lower administrative costs.

4.3 Authorisation procedure

As mentioned in section 4.2, for both system designs in this requestor-centric approach, a central trusted party (or parties) must be responsible for certifying all requests from requestors. In this section, we will go into more detail on how this process could work in practice.

4.3.1 Classic authorisation procedure

A classical, eIDAS-inspired authorisation procedure would consist of a verifier providing an authority with documents on the data they want to process. This could include a privacy statement, possible policy documents and agreements regarding data processing, et cetera. The authorisation authority could typically also require (regular) audits to assess whether the policies are being adhered to properly and security standards are met.

While such procedures tend to give high assurance on data security, which might be required for certain contexts, the costs are also extremely high and potentially unrealistic for an SSI system to be functional on a large scale in all contexts.

Of course, the procedure does not have to be extremely strict for all parties, which can reduce costs. Still, though, even when all auditing steps are removed and verifiers only need to submit just a short document describing the requests they want to do and for which reason, the administrative burden of running a governance organisation to decide on the justifiability of those requests, would already be very high. This is because it is a decision that can hardly be automated, and parties might appeal against the organisation denying a specific request.

4.3.2 Public self-registration

To reduce this administrative burden, we could consider a system where requestors can register *themselves* and use transparency to our advantage. This is an idea that has earlier been suggested as a possible solution to the problem of over-asking by the development team of the Dutch Digital Identity (EDI) Demo Wallet in the context of the EU Digital Identity initiative (section 2.2), though no exact details have been published yet at the time of writing.

The fundamental assumption behind this idea is that we do not necessarily aim to prevent over-asking completely. In fact, as discussed in section 1.1, over-

asking is not just an SSI-specific problem; it can occur in other forms of IdM as well. Only certain characteristics of SSI-based IdM make it a more significant problem, and the decentralized nature of SSI makes it hard to detect whether over-asking is happening at a large scale.

As a society (i.e. the way national data protection authorities currently work), we do not aim to protect everyone's privacy perfectly and prevent privacy violations completely. That would not be feasible. We focus instead on high-impact privacy violations (either due to the large scale on which they take place or the sensitivity of the violation). Therefore, a measure against over-asking in SSI also does not need to be perfect. It could be sufficient to implement a countermeasure against the negative consequences (intransparency) of the decentralized architecture that makes over-asking in SSI a more prominent problem.

Instead of having a central party that actually performs an audit upfront to determine whether a requestor's data requests are legitimate, we can have a central party that simply does not perform any substantive assessment. Instead, it only enables requestors to register themselves in a public registry. This registration should contain the attributes this party wants to request and for which reason. This only requires the organisation to *authenticate* a requestor, for which existing solutions exist and which can thus be automated.

Because the registry is public, this enables democratic bodies, interest groups, privacy organisations and authorities to monitor the contents of this registry. When a requestor is found to be requesting attributes that they should not request, this can be reported to the central party, who could then revoke the authorisation of this requestor.

The transparency of this system enables the public to monitor the authorisations of requestors while it removes the administrative burden upfront. And while over-asking can certainly still occur because of illegitimate entries in the registry, this system would make this visible again, which is a first step before sanctioning.

While monitoring the public registry, certain interest groups could prioritise specific highly sensitive attributes over others, while other groups could prioritise larger (multinational) organisations over smaller ones or focus on particular sectors, et cetera. Additionally, automated tools could scan the registry for parties that request a suspiciously large number of attributes or do not provide a seemingly reasonable purpose.

The rationale behind this approach is that by making the registry public, organisations have a natural interest in registering themselves properly and acting in accordance with regulations, because anyone can hold them accountable for this. Self-registration in a public registry would not be able to prevent illegitimate requests from being possible in the system upfront, but still, parties could be held accountable afterwards, which, in many cases, could be a sufficient countermeasure against over-asking.

4.3.3 Hybrid approaches

We emphasise that the approach of public self-registration does not *exclude* the possibility of requiring more extensive audits for requesting selected attributes either. It could be very reasonable to, for example, implement a system in

which, for general disclosure requests, public self-registration is sufficient, but for requests that contain selected, more sensitive attributes, further assessment and possible audits are required in order to be certified.

Finally, one could also make registration optional for permissive wallet applications. Wallets could be configured (possibly by users themselves) to just display a warning when a request has not been certified. While such decisions can be implemented easily, they can significantly influence the actual behaviour of the system. We will further discuss this in section 5.1.1.

Public self-registration could thus be considered as a practical intermediate form of authorisation, right between the current situation in which users are entirely on their own and a system where all requestors must undergo an extensive assessment of their requests, which has high costs.

4.4 Implementation in Yivi

Following the same reasoning as in section 3.6, we consider it essential for adoption *not* to introduce the burden of key management to verifiers, especially when we would require this for *all* requestors (and not just those few that might want to request specific protected attributes). Therefore, we again propose an implementation based on requestor authentication via TLS and authorisation based on a central scheme.

This can again be realised with relative ease via the requestor scheme that already exists in Yivi.

```
[
    ...
    {
        "id": "pbdf-requestors.someauthorisedparty",
        "name": {
            "en": "Example requestor",
            "nl": "Voorbeeld requestor"
        },
        "hostnames": [
            "authorised-requestor.example.com"
        ],
        "certified_requests": [
            {
                "disclose": [
                    [
                        "pbdf.pbdf.email.email"
                    ]
                ],
                "reason": {
                    "1": {
                        "en": "To send you a newsletter",
                        "nl": "Voor het versturen van een
                            nieuwsbrief"
                    }
```

```
                }
            },
            ...
        ]
    },
    ...
]
```

Listing 4.1: Example (partial) Yivi requestor scheme displaying a requestor being authorised to request an email address for sending a newsletter.

As seen in Listing 4.1, the requestor scheme needs to be updated to contain the certified disclosure requests a requestor is authorised to perform. For each request, the JSON serialized session request (in 'condiscon'[1] format) is included in the scheme, as well as a short description (for every individual attribute) of the reason for the request, which the wallet should display to each user.

The scheme could possibly contain additional information about the request, giving a more in-depth explanation of the context (e.g. why these attributes are required, how long the data will be stored, et cetera) or the requestor (e.g. contact details of the requestor's privacy officer). This data could be included to substantiate the request's legitimacy but could also be displayed to the user in the wallet (perhaps after the user clicks on a button for *more information*).

4.4.1 Online portal

Currently, the Yivi scheme (both the issuer and requestor scheme) is maintained by the Privacy By Design Foundation and signed by SIDN. From a technical perspective, it would also be practical to assign those parties the task of authorising requestors access to attributes. However, this is probably not a responsibility those parties would want to take on. Therefore, we propose the implementation of public self-registration.

To achieve this, an online portal should be created where Yivi verifiers can log in (which could, of course, be implemented via Yivi, too, based on the Dutch Chamber of Commerce credential) and manage their registration in the requestor scheme. Via this portal, organisations should be able to register the hostnames of the Yivi servers of their infrastructure, as well as the requests they want to perform (containing the attributes they wish to request and the reason for asking for them).

It is essential that some (automated) methods for proving ownership of these hostnames are implemented in this portal, too. This could be implemented in the form of a protocol with a Yivi server, as these should reside behind this hostname, or by setting specific DNS records for the domain.

Periodically (and automatically), based on the entries in this portal, a new version of the requestor scheme could be created, signed by SIDN and published to the wallets.

[1] `https://irma.app/docs/condiscon/`

We remark that this portal could also be used for issuers of Yivi credentials. Currently, issuers are expected to upload their public keys and change their scheme by opening a pull request on the GitHub repository containing the Yivi scheme. While this offers the same functionality (except for the automated verification of ownership of hostnames), implementing an online portal could potentially decrease the workload for SIDN.

4.4.2 Scalability

An obvious drawback of this approach is scalability. Currently, the Yivi ecosystem does not have such a large number of verifiers (yet) that this would immediately become a problem. However, including all verifiers in the scheme and pushing that scheme to all wallets on every update might not be a long-term sustainable solution if the ecosystem were to grow to a significantly larger number of verifiers. While accumulators could potentially alleviate the problem to some extent, the required update frequency still makes this approach poorly scalable.

Ultimately, a dedicated PKI is the only way to solve this scalability issue. This is feasible but will be a significant change to the existing system.

Chapter 5

Afterthoughts

So far, we have tried to solve the problem of over-asking by implementing technical measures that enforce some form of access control on SSI credentials. Especially for the broader definition of over-asking that requires a more general solution, we have seen that scalability is a fundamental problem, not just from a technical perspective (e.g. the size of the scheme) but also from a governance perspective. The costs of having a single central trusted authority certifying every data request (at least to some extent) can be very high and unrealistic for a globally interoperable system.

Meanwhile, in section 4.3.2 on public self-registration, we have mentioned that especially under the general broad definition of over-asking (not just considering specific highly-sensitive attributes), the goal does not necessarily have to be to completely prevent *any form* of over-asking *upfront*. Just adding more transparency to the system, together with the ability to take measures *after* over-asking is observed, can also be a reasonable solution. Sometimes, thus, more simple and elegant solutions that do not enforce technical preventive measures against over-asking, but *do* significantly *diminish* the problem at significantly lower costs could be considered preferable solutions.

In this chapter, we will further examine several other approaches to, perhaps not completely solving, but significantly preventing the problem of over-asking in SSI. By the nature of these approaches, we will not elaborate on them in as much detail as in the previous chapters, nor will we make concrete proposals for implementations. Instead, these ideas can be interesting considerations for solving the problem of over-asking in a non-default manner.

5.1 UX aspects of wallet applications

An important reason why complete prevention is a hard thing to achieve is that it requires universal judgements on whether some request is actually justifiable or not. Some even argue that leaving this decision to a third party is undesirable, as it takes away the user's autonomy and reduces privacy (which includes users having a *choice* on disclosure) to compliance with certain policies [2].

Instead of having a central party decide on whether something is over-asking, one could also consider the topic from the perspective of user empowerment and let users make that decision all by themselves, staying closer to the leading SSI paradigm.

Following this approach, the challenge for the system is to reduce all factors that currently contribute to over-asking being a more significant problem for SSI. The potential ignorance or unawareness of users (as discussed in section 1.1) should be reduced as much as possible by properly informing users of all factors that might be relevant to a deliberate decision on data disclosure.

To start with, the wallet should clearly display the context of the data request to the user, so they can verify if that context is actually correct and data disclosure in that context is acceptable for them. The application's user interface could also give the user additional tools to help users make a properly informed decision.

Considering permissive wallet behaviour, with users ultimately being able to make their own decisions based on the information provided by the wallet application, perhaps the greatest gains can be achieved in this area. And while the ultimate power imbalance between users and verifiers cannot be taken away, in many other cases where the power imbalance is not that important, these UX aspects should definitely not be ignored.

Terpstra et al. describe this approach as implementing *reflective (design) patterns*: "Individuals should thus be encouraged – through the design of the products [...] – to make individual choices [...] about privacy" [2]. To achieve this, deliberate *friction* could be implemented ([2], [3]) to break habits and give users time to think and become conscious of the interaction [2]. Examples include deliberately slowing down interactions, making them more complex or asking specific questions that might lead to different perspectives [2]. We will present some concrete examples in this section.

5.1.1 Permissive wallets and bypassing warnings

We briefly touched upon the topic of permissive and strict wallets in section 3.5.3. Chadwick et al. [4] introduced these terms to describe SSI wallets that either display a warning to the user (i.e. *permissive* behaviour) when they are about to disclose data to a (potentially) unauthorised party, or completely prevent disclosure in such cases (i.e. *strict* behaviour). While under the conditions of chapter 3, a case could be made for both behaviours (and perhaps a *strict* behaviour would be preferable considering the issuer's interests or a user subject to power imbalance), generally (considering the conditions of chapter 4), we consider a *permissive* behaviour to be preferred.

A permissive wallet more closely follows the principles of interoperability and data portability, which are fundamental to the SSI paradigm. The risk for a closed system, as discussed in section 3.5.1, where users are not ultimately in control anymore, is more significant in strict wallets. In fact, this problem is what drove user-centric IdM towards SSI (see section 2.1.1).

Moreover, permissive behaviour would make a system with certified disclosure requests more feasible, cheaper and easier for adoption because verifiers can still use the system without actually being registered as an authorised party (which

could have costs involved). Parties could thus decide to wait some time before actually going through this procedure in order to remove the warning.

So, while strict wallet behaviour could be desirable for protected attributes (with only limited use cases, as discussed in section 3.5.2 and 3.5.4), for general attributes, wallets should be permissive.

Considering a permissive wallet behaviour to be preferable, however, gives room for a wide range of implementations. The exact way in which a warning is presented to the user and how it can be bypassed has an immense influence on its actual effect.

Below, we list a number of proposals for how wallets can implement the warning, increasing in strictness.

- Wallets warn the user that they are about to disclose data to an unknown, potentially unauthorised party. The primary button (visually) is to *proceed*, nudging users in this direction.
- Wallets warn the user that they are about to disclose data to an unknown, potentially unauthorised party. The primary button (visually) is to *abort*, nudging users in this direction.
- The warning contains an elaborate explanation of the possible risks of disclosing the data that the user needs to read before choosing to proceed or abort.
- Bypassing the warning is only possible after entering a PIN or password or explicitly retyping the name of the party that is requesting the data.
- The wallet must be configured (via a settings screen) to allow warnings to be bypassed. By default, the wallet always aborts.
- The wallet must be configured (via a settings screen) to allow warnings to be bypassed. This configuration lasts for 15 minutes, meaning that after 15 minutes, users need to re-activate unauthorised disclosures in order to bypass the warning.

All behaviour could be considered permissive, though obviously, there is a big difference between all implementations. Depending on the actual SSI landscape and details of the certification process, different design decisions can be made.

Van Elteren [49] has researched in her master thesis how different forms of such friction can be implemented in Yivi, and to what extent it can be an effective countermeasure for protecting the Dutch BSN. While she could not find a significant effect in her research on *preventing* users disclosing the BSN (which could possibly be explained by the setup of the experiment), she did find users being *more alarmed* by the different approaches, so the measures are proven to have *some* effect.

An important remark that is also mentioned by Terpstra et al. [2] is that with these approaches, it is important that they do not become subject to habituation (it should not become a standard procedure for users to pass the warning).

5.1.2 Historic disclosure behaviour

There are more aspects in which the wallet could provide more tools to the user. For example, all times a user is about to disclose credentials to a *new* party could be specifically important. In the phishing scenario as described by Chadwick et al. [4], when users are disclosing data to a new organisation they have never interacted with before, we argue that it could be relevant for the wallet application to warn the user specifically about this fact. For the case of over-asking by legitimate parties, this could also be interesting, as disclosing an attribute to a new organisation for the first time is more sensitive than disclosing it a second or third time to an already known (and trusted) organisation.

Most wallet applications already keep logs of disclosure sessions that have been performed. Implementing this extra behaviour could thus be easy to implement.

As an example, every time a user is about to disclose their Dutch BSN to a new organisation, the wallet could warn the user '*You haven't disclosed your BSN to Organisation X before. Are you sure you want to disclose this data to Organisation X? (y/N)*'. Possibly, the warning could also include the attributes that this organisation *did* already receive from the user as well. For completely new organisations, the wallet could warn the user '*You haven't disclosed any attributes to Organisation X before. Do you want to continue? (y/N)*', possibly even warning that the identity of the requestor could not be reliably established. A similar implementation for Yivi has been proposed earlier by Schraffenberger and Jacobs [3].

Additionally, the wallet could, for each attribute, keep an overview of which organisations have received that attribute and make that overview easily available to the user. This could help the user make informed decisions about whether they want to disclose an attribute to an organisation (based on what they typically do).

Even more advanced, more modern wallet applications could implement some form of neural networks trying to establish a standard behaviour of data disclosure to parties and warning the user when, for example, they are about to disclose a large number of attributes that they usually would not disclose altogether to the same party.

While similar approaches are generally quite successful in different kinds of applications (e.g. web browsers, email clients, or bank accounts usually have similar features), we think it is important not to overestimate the strength of this measure for the field of SSI. Typically, we think the power of SSI systems lies in the ability to reliably share information with a *new* organisation *for the first time*, for example, during some form of enrolment. We think that it remains questionable how popular SSI will become as a regular means of authentication towards the same party many times (or to what extent other forms of authentication will remain popular). It could thus very well be that, in reality, a large part of the disclosure sessions for an SSI wallet will be to new organisations. In that case, the value of presenting such warnings will be low, as they will become subject to habituation [2]. The actual value of such measures is thus yet to be seen and will highly depend on the actual developments and adoption of SSI systems.

5.1.3 Displaying sensitive credentials

In chapter 3, we proposed the concept of protected attributes for specific, highly sensitive attributes that require extra protection. We have focused on protecting them with some mandatory form of access control for requesting them. However, in order to actually properly protect the credential, it should be protected in the user interface of the wallet application as well.

For example, we argue that it should not be possible to make screenshots of the wallet application interface and (accidentally or deliberately) disclose the credentials in that way. For most smartphones, measures can be implemented to complicate making screenshots, and we argue that those should be implemented when protected attributes are on screen.

Additionally, one could think of hiding the fields in the credential (like operating systems usually do with password fields) and only briefly displaying them after confirmation with a PIN or password. This offers some protection against so-called *shoulder surfers* (people looking over your shoulder at your screen) or just accidental disclosure when using the app for something different.[1] Implementing this behaviour might also, intuitively, make users aware of the sensitivity of such attributes.

5.1.4 Protected attributes without authorisation infrastructure

When fully taking the user interface approach, a much simpler solution for protected attributes could be considered that fully relies on the user interface without introducing any authorisation infrastructure for requestors.

A very subtle way of doing this[2] can be found in recent mock-ups[3] for the Dutch Digital Identity (EDI) Demo Wallet that is being developed by the Dutch government (section 2.2).

The Dutch BSN here is displayed with an asterisk (*) symbol, which indicates it is a sensitive attribute. Whenever a user is about to disclose this attribute to any party, a small additional warning is displayed on the screen, reminding the user that they are about to share a sensitive attribute that the other party may not be authorised to receive.

Additionally, Schraffenberger and Jacobs have suggested a similar implementation for Yivi by *color-coding* disclosure sessions [3], which was also researched by van Elteren as one form of friction [49].

While this is a very elementary measure that does not have any technical strength, it could also be a very elegant solution in its simplicity. When users are properly educated, and the wallet application sufficiently informs the user, one could argue that this pure UI approach to protected attributes actually

[1] Meanwhile, it *should* always be possible for the user to see the value of some attributes in some way for transparency purposes (as discussed in section 1.1.4 about encrypted attributes), so completely hiding the attributes would not be a proper solution.

[2] At least, the mock-ups of the user interface display this behaviour. It is unknown whether an additional trust infrastructure will exist in this wallet and in what form.

[3] `https://www.figma.com/file/dO5pKIIIyDgG0N2ZX4C2xd/Designs_Demo_NL-Voorbeeld-Wallet`

could be sufficient (though obviously, any economic arguments from the issuer's perspective as discussed in section 3.5.2 and 3.7 are incompatible with this approach, as disclosure protection here is not in the user's interest). Considering only a few attributes of a user need to be labelled as sensitive (and habituation is thus not a big risk), this simple approach might, in fact, be better than introducing a whole authorisation infrastructure as proposed in chapter 3 for just a few attributes.

We emphasise that in this case, while the *authorisation* infrastructure is left out, a proper *authentication* infrastructure becomes even more important. In order to let the user make an informed decision, the wallet application should display proper authenticated information about the requesting party.

5.2 Categorisation of credentials

As mentioned before, one of the problems with the technical solutions from chapter 3 and 4 is that they require a third-party judgement on the authorisation of requests, of which the costs can be high. And while the solution of self-registration can lower these administrative costs, the burden of maintaining the open registry still exists. An ideal solution does not require such a third party.

In section 1.1.5, we have discussed Walzer's *Spheres of Justice* [8], describing our society by the hand of a number of *spheres*. Together with Nissenbaum's view on privacy as contextual integrity, we have argued that privacy is maintained as long as personal data is kept within its context. While contexts consist of more than just the category or sphere, they largely do fit within a single sphere. A simple yet elegant and pragmatic approach to prevent over-asking could be to apply these spheres to credentials in an SSI ecosystem and wallet as well.

First, the SSI platform should decide on a concrete list of spheres. This could, in fact, be quite a challenge and a topic open for debate.

As a proposal, we define the following spheres: *public administration*, *education*, *healthcare*, *finance*, *work*, *leisure*, and *commerce*.

All issuers and their credentials must be categorised in one of these specific spheres. For example, someone's bank account will be categorised in the finance sphere, and their diplomas will be in the education sphere. Additionally, there should be a *general* category that contains basic information about one's base identity that is compatible with all spheres (such as name, date of birth, et cetera).

Additionally, verifiers should also register themselves in precisely one of these categories (for example, via a requestor scheme, but it can also be completely self-proclaimed). An online shop will, for example, register themselves in the *commerce* category.

Data shared within a single sphere could be considered acceptable in any case. However, when requesting data registered in a different sphere than the verifier is registered in, the wallet application should warn the user about this, mentioning that the user, for example, is about to share healthcare data with a commercial party, which they may not want to do.

As long as the verifier registration is immutable, this system requires little to no governance. A party registering themselves in a different sphere/category than they are supposed to be in will only result in that party not being able to request in its *actual* category. There is thus an incentive for requestors to register themselves in the correct category. The immutability can easily be achieved by, for example, making wallets keep a list of all requestors that it has interacted with and their category.

We typically want to prevent parties from receiving information from different spheres, and the proposed system does achieve this. As such, this could be an interesting approach to prevent over-asking that comes at little governance costs. Though it is far from perfect, it could be an interesting approach to further investigate when designing SSI systems.

It is important to notice that certain spheres will often be mixed. For example, your diplomas (from the education sphere) are relevant to your (future) employer (work sphere). For other spheres, however, this is a bigger violation, like healthcare data leaking to the commerce sphere. Wallet applications could potentially implement different behaviours for different spheres, e.g. displaying different kinds of warnings depending on the exact spheres of the issued credential and the requestor.

An important remark is that the Yivi system that we specifically discuss in this thesis, already applies some categorisation of credentials its wallet interface. However, these categories are not further used. Verifiers are not categorised and credential disclosure is not limited by these categories.

5.3 Federated schemes

As mentioned before, the most significant problem with the proposed solutions for the Yivi ecosystem, especially the one in chapter 4, is the scalability of the scheme approach, both from a technical perspective (the scheme becomes large) and a governance perspective (it is difficult for a party to manage the scheme). This makes the scheme approach in its current form already potentially problematic (where only issuers are included); even without any actual changes to the scheme, Yivi requires an annual key rollover, and all previous public keys for an issuer are stored in the scheme, resulting in quite a large scheme. When an authorisation infrastructure for requestors is introduced, this problem becomes significantly larger due to the relatively high number of requestors in an SSI system compared to a low number of issuers (even though we do not store public keys for requestors in our proposal).

It could be helpful to try to limit the technical and governance costs of the scheme by limiting the scheme size and updates and splitting up the governance responsibilities.[4]

[4]For now, we will only focus on the governance benefits, but in section 5.3.4, we will examine how this can also give us technical scalability benefits.

5.3.1 Governance benefits

Currently, the Yivi ecosystem only knows one production scheme, signed by one party based in the Netherlands. It would be logical for the system to support multiple schemes, at least one per country in which issuers and verifiers are based. This would split the governance responsibilities for the different countries.

One benefit would be that a national scheme manager will only be subject to their own national legislation, which could massively simplify their internal policies. In Europe, other legislation on data protection applies than in the United States. National scheme managers will have less trouble balancing themselves between multiple, possibly contradictory, legislation.

A nation-based approach, however, is not the only way to split up the scheme. In fact, a true hierarchical scheme could be implemented to get further governance benefits.

This approach could be taken for both issuers and verifiers (requestors).

Sector-specific authorities could be assigned to maintain a scheme for their sector. Such authorities have better connections with the actual parties they would need to include in the scheme, which can be a major benefit. For example, a National Research & Education Network (NREN) like the Dutch SURF cooperation (*Samenwerkende Universitaire RekenFaciliteiten*) could be a scheme manager for the education and research sector. SURF already has established contacts within the Dutch education and research sector with consequently a lot of domain knowledge, and it is also an internationally acknowledged organisation. This all contributes to the governance benefits of a split scheme approach.

For commercial parties, commercial scheme managers could exist. As such, a multitude of smaller Yivi schemes could exist that are easier to maintain than one single big scheme.

5.3.2 Hierarchical scheme signing

With multiple parties managing a (partial) scheme, there must be some way to discover and trust these schemes. For this, classical cryptography can be used in an X.509 Certificate Authority-inspired PKI. This approach is also very similar to the eIDAS implementation (see section 2.2.2).

National scheme managers could be used as a root. The first time using a wallet application, users should be able to configure the country or countries of which the scheme should be trusted (or this happens by default based on the country the user resides in).

In the national scheme, sector-specific organisations could be included. This is done by including the keys they use for signing and/or the endpoints at which they publish their (sub)scheme.

These lower-level (sector-specific) organisations can then define issuers and credentials in the scheme that they sign. Alternatively, these organisations could only define issuers and let the issuer manage the credentials themselves in yet another lower layer, as a subsubscheme.

5.3.3 Scheme federation

The possibility of multiple, hierarchical schemes has governance benefits but also comes with the significant drawback that it limits interoperability. If every country uses their own scheme, cross-country usage would not work: a Dutch verifier would not be able to request a German attribute as they reside in different schemes (or all parties would also have to configure the German scheme as root, which is unpractical). However, scheme federation can solve this problem.

National scheme managers should be able to include each other's schemes as federated schemes. For example, the Dutch scheme could include the German scheme as a trusted federated scheme and, as such, allow all users that use the Dutch scheme to also use the German one.

Ultimately, hierarchical schemes with federation basically allow us to implement the same functionalities as well-known PKIs with federated trust models. This has many benefits with regard to flexibility, yet the complexity of the infrastructure could be considered a disadvantage.

5.3.4 Just-In-Time scheme retrieval

While there are major governance benefits to a hierarchical federated scheme, this approach does not yet directly improve the *technical* scalability (size of the scheme) if we still consider schemes to be downloaded to a wallet device regularly. However, splitting up the scheme does enable improvements in this field.

An interesting observation in this regard would be that wallet applications do not necessarily need to keep a copy of the entire scheme. Instead, they only need to keep a copy of the parts of the scheme for which they contain credentials. We can use this to our advantage by implementing **Just-In-Time (JIT) scheme retrieval**.

When an issuance session is performed, first, the issuer sends the signed (partial) credential scheme of the credential that is about to be issued. Only then does the wallet learn about the scheme of the credential, and the wallet stores it before continuing the issuance session. A similar thing would happen upon disclosure. The verifier, upon requesting attributes, sends along with the request the signed (partial) schemes of all requested attributes and a signed part of a scheme that says something about itself as a verifier and the reason for requesting these attributes. Of course, all schemes need to be signed by a trusted key, either a root key or from an earlier installed (federated) scheme.

Changing the protocol in this way would severely limit the amount of data a wallet needs to store for the scheme, as only the scheme parts of actual credentials contained in the wallet need to be stored. Also, no updates have to be pushed to all wallets, but schemes can be updated right when a session is performed. This actually ensures that wallets will always have the latest, most up-to-date version of the scheme at the time they need to use it. Only issuers and verifiers will need to keep a signed copy of the credentials they are either issuing or requesting and can serve the scheme when they are interacting with actual users, right when it is needed.

Scheme updates only need to be pushed to issuers and verifiers and only for the attributes that they use. A challenge remains in ensuring freshness of the provided scheme (we must be certain that issuers and verifiers are providing us with the latest, most accurate version of the scheme), but default solutions exist for this, for example, considering technology behind certificate revocation lists (CRLs).

So, while from a governance perspective, a central and public scheme still exists (though possibly maintained by multiple parties in a hierarchical and/or federated manner), wallet applications do not actually have to keep track of the complete scheme. This massively improves scalability while keeping the benefits of the scheme approach.

Ultimately, this could be considered an intermediate version in the transition from a central scheme as a trust anchor for Yivi to a dedicated PKI.

Chapter 6

Conclusion

In this thesis, we have given an analysis of the topic of over-asking in SSI ecosystems. we have identified why this is a more prominent risk in SSI than in earlier forms of IdM and what factors contribute to this. While SSI is generally presented as a privacy-friendly technology and a privacy improvement over other forms of IdM, we argued that in certain aspects, it could also be considered a dangerous technology instead, as it burdens users with responsibilities they might not be able to live up to. Moreover, we have discussed several approaches to preventing over-asking, with specific proposals for implementations in the Yivi ecosystem.

While doing this, we have considered two definitions of over-asking.

First, in order to prevent unauthorised parties from requesting specific highly sensitive attributes (like the Dutch BSN or attributes from one's DNA), we have proposed the concept of *protected attributes*. These are attributes for which we specifically define which requestors should be authorised to request them, in contrast to general attributes in SSI that any party can request.

In Yivi, we have seen that this can be implemented with relative ease with verifier authentication using TLS and authorisation in either of two ways, depending on which is most applicable to the specific attribute: scheme-based (with the scheme manager carrying responsibility) or via an issuer-defined authorisation server with accumulators (which is less transparent but scales better when there are many verifiers). The benefit of this implementation (over a classical PKI) is that it allows for easy adoption by requestors as it does not require key management, and a lot of the infrastructure already exists in the Yivi ecosystem. We consider both to be essential for a pragmatic and successful solution.

For the broader definition of over-asking that does not just consider specific highly-sensitive attributes, there is an administrative/governance challenge, as this requires a trusted authority to decide on whether certain data disclosure requests are acceptable or not. The costs of running such an authority can be high, though they can be significantly lowered by implementing open public self-registration (at least partially, for most attributes), which enables democratic bodies and interest groups to audit the procedures.

For the Yivi ecosystem specifically, we propose both the implementation of protected attributes with strict wallet behaviour, but only for few selected attributes, and general requestor registration based on open public self-registration with permissive wallet behaviour, as both solutions have their own benefits and strengths for specific use cases.

Though both solutions can be implemented with TLS-based authentication to allow for easy adoption, the scalability of defining authorisation policies in a scheme remains problematic. To solve any scalability issues with the Yivi scheme, we have discussed a different setup for scheme management with a hierarchical and federated scheme.

Ultimately, however, we concluded that the solution for over-asking does not necessarily need to be purely technical. The goal does not have to be to fully prevent over-asking but only to implement countermeasures against the factors that make over-asking a more significant risk for SSI than other forms of IdM. We argued that SSI wallets should properly inform users of the context in which data disclosure is about to take place, provide users with the proper tools to do this and educate them about the responsibilities and expectations of the SSI system. This is not specific to the Yivi ecosystem but to any SSI system. As a starting point, we have listed a number of ideas that wallet applications could implement in order to help users better bear the responsibility of protecting their own data.

With the current developments around the EU Digital Identity initiative, it will be interesting to see how the SSI landscape will take shape in the upcoming years and how the problem of over-asking will be addressed by the architectural decisions that will be made, especially regarding the registration and certification of data disclosure requests.

Chapter 7

Future work

In this thesis, we have tried to cover the topic of over-asking in SSI ecosystems as completely as possible, with specific attention to the Yivi ecosystem. While we have provided several approaches to solve the problem to some extent, we ultimately did not find a perfect solution. In this chapter, we will identify several directions for future research.

7.1 Encrypted disclosure of protected attributes

In section 3.7, we have presented a problem with the proposed solution for *protected attributes*. In the proposal that was discussed, the wallet application itself (following the user's decision) is responsible for deciding whether to disclose an attribute to a requesting party or not. While this approach is feasible when the user's wallet is trusted, under certain scenarios, this might not be the case, especially when we consider protected attributes where the *issuer* (and not per se the user) has an interest in protecting the attribute, such as for economic reasons.

For this scenario, an alternative solution needs to be found. We already discussed the possibility of certified wallets (in section 3.7), but these require trusting the operating system. Instead, it would be interesting to solve this problem by altering the (cryptographic) protocols.

The wallet should only be able to disclose an encrypted version of the attribute, so only authorised verifiers with the proper decryption key will be able to decrypt the contents of the attribute. However, instead of doing this in the naive manner as described in section 1.1.4 on encrypted attributes (where simply the attribute's contents are symmetrically encrypted), the wallet itself should be able to see the contents of the attribute for transparency reasons. Only the signature on the credential must be encrypted in such a form that only the wallet itself and authorised requestors can verify it. This could potentially be achieved with polymorphic encryption of the credentials.

7.2 Empirical research to user perception

As discussed in chapter 1, a fundamental problem with over-asking in SSI ecosystems is that users can be unaware or ignorant about their own responsibility towards the protection of their own data in an SSI wallet. So far, we have based our research on a lot of assumptions about this topic. However, empirical research on the user perception of using SSI wallets is required to further substantiate these assumptions. What are the user's actual expectations about the SSI platform and wallet, and who do they consider responsible for data protection? This is essential for a proper design of an SSI system and wallet application.

Additionally, it is important to know how users experience the user interface and usability of the wallet, not per se for accessibility purposes, but more so for awareness of security and privacy. It is interesting to know how users experience strict or permissive wallet behaviour (and the exact design of the warning or consent screen), as well as the authentication of requestors and how they judge the legitimacy and trustworthiness of a requestor in their app. More research like that of Schraffenberger and Jacobs [3] as well as van Elteren [49] is required.

7.3 Legal data protection responsibilities for SSI

As mentioned in section 1.1.4, it is unclear to what extent issuers of credentials have a legal responsibility to protect the data contained in these credentials when they are in SSI wallets. On the one hand, users (data subjects) have a right to access the data an issuer (data processor or data controller) has about them. These data processors must even make the data available in a common format. While doing so, however, they must still properly protect the data. As an example, hospitals must provide patients with their medical files when they request them, but they should not provide those files by sending them over regular email. That would be considered an insecure way of sharing the data and thus a violation of their duty to protect them.

Similarly, one could argue that issuers might have to seriously assess whether certain data should be made available as an attribute in an SSI credential at all, as this makes the data vulnerable. Perhaps certain information is too sensitive to be issued to a user's SSI wallet. Both from a legal and ethical perspective, this topic will need more attention in the upcoming years, in which these questions will become more actual.

References

[1] R. H. Thaler and C. R. Sunstein, *Nudge: Improving Decisions About Health, Wealth, and Happiness*. Penguin Books, 2009, ISBN: 978-0-1410-4001-1.

[2] A. Terpstra, P. Graßl, and H. K. Schraffenberger, "Think before you click: How reflective patterns contribute to privacy," in *CHI Conference on Human Factors in Computing Systems. What Can CHI Do About Dark Patterns?* CHI Workshop, 2020. [Online]. Available: `https://hdl.handle.net/2066/246490`.

[3] H. K. Schraffenberger and B. P. Jacobs, "Friction for privacy - why privacy by design needs user experience design," *European Cyber Security Perspectives*, pp. 12–14, 2020.

[4] D. W. Chadwick, M. Kubach, I. Sette, and I. H. J. Jeyakumar, "Establishing trust in SSI verifiers," in *Open Identity Summit 2023*, Gesellschaft für Informatik, 2023, pp. 15–26, ISBN: 978-3-8857-9729-6. DOI: `10.18420/OID2023_01`.

[5] European Commission, *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) (Text with EEA relevance)*, 2016. [Online]. Available: `https://eur-lex.europa.eu/eli/reg/2016/679/oj`.

[6] M. Eisenlohr, "Constrain the verifier - preventing over-identification in self-sovereign identity," M.S. Thesis, Radboud University Nijmegen, 2023. [Online]. Available: `https://github.com/MaJoEi/scriptie-demo/blob/main/Master_Thesis.pdf`.

[7] T. Sharon, M. Stevens, S. Kraaijeveld, and L. Siffels, *Sphere transgression watch*. [Online]. Available: `https://www.sphere-transgression-watch.org`.

[8] M. Walzer, *Spheres of justice: a defense of pluralism and equality*. Basic Books, 1983, ISBN: 978-0-4650-8189-9.

[9] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, *Waarden, kansen en uitdagingen rond het Europese Digitale Identiteit raamwerk*, 2022. [Online]. Available: `https://www.rijksoverheid.nl/documenten/brieven/2022/07/26/waarden-kansen-en-uitdagingen-rond-het-europese-digitale-identiteit-raamwerk`.

[10] H. Nissenbaum, *Privacy in Context Technology, Policy, and the Integrity of Social Life*. Stanford University Press, 2009, ISBN: 978-0-8047-5237-4.

[11] K. Cameron, *The laws of identity*, Microsoft Corporation, 2005. [Online]. Available: https://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf.

[12] C. Allen. "The path to self-sovereign identity." (2016), [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[13] K. Jordan, J. Hauser, and S. Foster, "The Augmented Social Network: Building identity and trust into the next-generation internet," *First Monday*, vol. 8, 8 2003, ISSN: 1396-0466. DOI: 10.5210/FM.V8I8.1068.

[14] A. Jøsang and S. Pope, "User centric identity management," *AusCERT Conference*, vol. 22, pp. 77–89, 2005.

[15] K. Cameron, R. Posch, and K. Rannenberg, *A user-centric identity metasystem: Proposal for a common identity framework*, Microsoft Corporation, 2008. [Online]. Available: https://www.identityblog.com/wp-content/images/2009/06/UserCentricIdentityMetasystem.pdf.

[16] D. Loffreto. "What is "Sovereign Source Authority"?" (2012), [Online]. Available: https://www.moxytongue.com/2012/02/what-is-sovereign-source-authority.html.

[17] D. Loffreto. "Self-sovereign identity." (2016), [Online]. Available: https://www.moxytongue.com/2016/02/self-sovereign-identity.html.

[18] A. Giannopoulou and G. Nl, "Digital identity infrastructures: A critical approach of self-sovereign identity," *Digital Society*, vol. 2, pp. 1–19, 2 2023, ISSN: 2731-4669. DOI: 10.1007/S44206-023-00049-Z.

[19] M. Graglia, C. Mellon, and T. Robustelli, *The nail finds a hammer self-sovereign identity, design principles, and property rights in the developing world*, New America, 2018. [Online]. Available: https://www.newamerica.org/future-land-housing/reports/nail-finds-hammer/.

[20] A. Giannopoulou and F. Wang, "Self-sovereign identity," *Internet Policy Review*, vol. 10, pp. 1–10, 2 2021, ISSN: 2197-6775. DOI: 10.14763/2021.2.1550.

[21] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology — EUROCRYPT 2001*, vol. 2045, Springer Berlin Heidelberg, 2001, pp. 93–118, ISBN: 978-3-5404-4987-4. DOI: 10.1007/3-540-44987-6_7.

[22] J. Camenisch and E. V. Herreweghen, "Design and implementation of the idemix anonymous credential system," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ACM Press, 2002, pp. 21–30, ISBN: 1-58113-612-9. DOI: 10.1145/586110.586114.

[23] J. Camenisch, S. Krenn, A. Lehmann, G. L. Mikkelsen, G. Neven, and M. Ø. Pedersen, "Formal treatment of privacy-enhancing credential systems," in *Advances in Cryptology – ASIACRYPT 2016*, vol. 9566, Springer Verlag, 2016, pp. 3–24, ISBN: 978-3-319-31300-9. DOI: 10.1007/978-3-319-31301-6_1.

[24] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, pp. 1030–1044, 10 1985, ISSN: 1557-7317. DOI: 10.1145/4372.4373.

[25] D. Chaum and J. H. Evertse, "A secure and privacy-protecting protocol for transmitting personal information between organizations," in *Advances*

in Cryptology, vol. 263 LNCS, Springer Verlag, 1987, pp. 118–167, ISBN: 978-3-5401-8047-0. DOI: 10.1007/3-540-47721-7_10.

[26] K. Rannenberg, J. Camenisch, and A. Sabouri, *Attribute-based Credentials for Trust: Identity in the Information Society*. Springer Cham, 2015, pp. 1–391, ISBN: 978-3-3191-4438-2. DOI: 10.1007/978-3-319-14439-9.

[27] M. S. Ferdous, F. Chowdhury, and M. O. Alassafi, "In search of self-sovereign identity leveraging blockchain technology," *IEEE Access*, vol. 7, pp. 103 059–103 079, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2931173.

[28] M. A. López, *Self-sovereign identity — the future of identity: Self-sovereignity, digital wallets, and blockchain*, Inter-American Development Bank, 2020. DOI: 10.18235/0002635.

[29] N. Naik and P. Jenkins, "Self-sovereign identity specifications: Govern your identity through your digital wallet using blockchain technology," in *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, IEEE, 2020, pp. 90–95, ISBN: 978-1-7281-1035-6. DOI: 10.1109/MobileCloud48802.2020.00021.

[30] N. Naik and P. Jenkins, "Governing principles of self-sovereign identity applied to blockchain enabled privacy preserving identity management systems," in *2020 IEEE International Symposium on Systems Engineering (ISSE)*, IEEE, 2020, pp. 1–6, ISBN: 978-1-7281-8602-3. DOI: 10.1109/ISSE49799.2020.9272212.

[31] N. Naik and P. Jenkins, "Your identity is yours: Take back control of your identity using GDPR compatible self-sovereign identity," in *2020 7th International Conference on Behavioural and Social Computing (BESC)*, IEEE, 2020, pp. 1–6. DOI: 10.1109/BESC51023.2020.9348298.

[32] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena, *uPort: A platform for self-sovereign identity*, uPort, 2017. [Online]. Available: https://whitepaper.uport.me.

[33] N. Naik and P. Jenkins, "uPort open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain," in *2020 IEEE International Symposium on Systems Engineering (ISSE)*, IEEE, 2020, pp. 1–7. DOI: 10.1109/ISSE49799.2020.9272223.

[34] A. Tobin and D. Reed, *The inevitable rise of self-sovereign identity*, Sovrin Foundation, 2017. [Online]. Available: https://sovrin.org/library/inevitable-rise-of-self-sovereign-identity/.

[35] C. Pattiyanon and T. Aoki, "Compliance SSI system property set to laws, regulations, and technical standards," *IEEE Access*, vol. 10, pp. 99 370–99 393, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3204112.

[36] C. Pattiyanon, "Security weakness and privacy preservation analysis of SSI management systems using information retrieval and system modeling," Ph.D. dissertation, Japan Advanced Institute of Science and Technology, 2023. [Online]. Available: http://hdl.handle.net/10119/18425.

[37] I. Henderson, J. Jeyakumar, D. W. Chadwick, and M. Kubach, "A novel approach to establish trust in verifiable credential issuers in self-sovereign identity ecosystems using TRAIN," in *Open Identity Summit 2022*, Gesellschaft für Informatik e.V.., 2022, pp. 27–38, ISBN: 978-3-8857-9719-7. DOI: 10.18420/OID2022_02.

[38] *Electronic signatures and infrastructures; trusted lists*, ETSI 119 612 v2.2.1, European Telecommunications Standards Institute, 2016. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/119600_119699/119612/02.01.01_60/ts_119612v020101p.pdf.

[39] *The common union toolbox for a coordinated approach towards a European Digital Identity framework - the European Digital Identity Architecture and Reference Framework*, EUDI ARF v1.0.0, 2023. [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework.

[40] N. Anciaux, W. Bezza, B. Nguyen, and M. Vazirgiannis, "MinExp-card," in *Proceedings of the 16th International Conference on Extending Database Technology*, ACM, 2013, pp. 753–756, ISBN: 978-1-4503-1597-5. DOI: 10.1145/2452376.2452472.

[41] C. A. Ardagna, S. D. C. di Vimercati, S. Foresti, S. Paraboschi, and P. Samarati, "Minimising disclosure of client information in credential-based interactions," *International Journal of Information Privacy, Security and Integrity*, vol. 1, p. 205, 2/3 2012, ISSN: 1741-8496. DOI: 10.1504/IJIPSI.2012.046133.

[42] *Verifiable Credentials Data Model*, W3C VC-data-model v1.1, The World Wide Web Consortium, 2022. [Online]. Available: https://www.w3.org/TR/vc-data-model.

[43] H. K. Schraffenberger. "Pretty verifier names." (2021), [Online]. Available: https://creativecode.github.io/irma-made-easy/posts/pretty-verifier-names/.

[44] *Machine Readable Travel Documents - Part 11: Security Mechanisms for MRTDs*, ICAO Doc 9303-11, International Civil Aviation Organization, 2021. [Online]. Available: https://www.icao.int/publications/documents/9303_p11_cons_en.pdf.

[45] *Technical guideline advanced security mechanisms for machine readable travel documents and eIDAS token*, BSI TR-03110, Bundesamt für Sicherheit in der Informationstechnik, 2015. [Online]. Available: https://www.bsi.bund.de/dok/TR-03110-en.

[46] M. Harbach, S. Fahl, M. Rieger, and M. Smith, "On the acceptance of privacy-preserving authentication technology: The curious case of national identity cards," in 2013, pp. 245–264, ISBN: 978-3-642-39077-7. DOI: 10.1007/978-3-642-39077-7_13.

[47] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, *et al.*, "Accumulators with applications to anonymity-preserving revocation," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2017, pp. 301–315, ISBN: 978-1-5090-5761-0. DOI: 10.1109/EuroSP.2017.13.

[48] R. O. Mason, "Four ethical issues of the information age," *MIS Quarterly: Management Information Systems*, vol. 10, pp. 5–12, 1 1986, ISSN: 0276-7783. DOI: 10.2307/248873.

[49] L. van Elteren, "Boosting with friction - the effects of design friction on deliberation in the context of privacy decisions," M.S. Thesis, Radboud University Nijmegen, 2020. [Online]. Available: https://theses.ubn.ru.nl/handle/123456789/12735.