Master thesis
Cyber Security

Radboud University

Constrain the Verifier - Preventing Over-Identification in Self-Sovereign Identity

Author:
Matti Eisenlohr
S1000794
matti.eisenlohr@ru.nl
matthias.eisenlohr@web.de

First supervisor/assessor:
prof. dr. Jaap-Henk Hoepman
jhh@cs.ru.nl

Second supervisor/assessor:
Dr. ir. Oskar van Deventer
oskar.vandeventer@tno.nl

Third supervisor/assessor:
Alexander van den Wall Bake
alexander.vandenwallbake@tno.nl

September 20, 2022

Abstract

Nowadays, individuals often need to share their personal data with organizations for various purposes. Unfortunately, some organizations coerce users into sharing a disproportionate amount of data as a condition for their services. In SSI environments the consequences of this are aggravated for individuals due to the high level of assurance of the shared data.

This thesis proposes a countermeasure against over-asking organizations ("verifiers") by proposing that verifiers need to obtain context-specific authorization from a trusted third party (an authorizer). For this purpose, the thesis defines what makes a data request "over-asking" and lays down general requirements and considerations as to how verifier authorization would need to be organized.
On the technical side, it introduces a basic logic decision model to determine the specific set of data that may be requested in a specific context and introduces so-called authorization certificates in the form of verifiable credentials, which act as a proof of authorization. Most importantly, this thesis proposes an extension to existing presentation exchange protocols in which verifiers first present their authorization to users (or rather their wallets), who then evaluate the contextual decision model to determine which attributes may be requested and inform the verifier of the result, before the start of the regular presentation exchange, during which the wallet automatically denies "over-asking" requests.

This provides a more privacy-friendly approach to qualified data exchange with SSI and helps to significantly reduce the amount of data that needs to be shared to a fixed minimum. It also provides many pointers on which a practical implementation of the proposed scheme could be build and expanded upon.

Contents

Chapter 1

Introduction

Self-Sovereign Identity (SSI) offers a novel way to digitally exchange data and manage digital identity while aiming to give individuals more control over their personal data. SSI roughly describes the concept of individuals having direct control of their own digital identity and by extension their personal data.
Within SSI systems, individuals (in this context called *holders*) store their personal data in so-called digital wallets (usually an app on their phone). They obtain this data in the form of digital credentials from so-called *issuers*, who are the authoritative source of the data and "vouch" for the authenticity and correctness of this data. User can share this data directly with services (so-called *verifiers*). Trust in the correctness of the data is established by verifiers trusting the issuer of the data and recognizing them as a legitimate source for this data [38].
With SSI, qualified[1] personal data can be shared safer, quicker and more reliably than before, as individuals do not need to obtain physical documents from third parties anymore to prove the validity of their claims, which can often be a lengthy process [41]. However it also makes it easier for large organizations to collect large amounts of data and lead to "over-identification"/"over-asking".

The term "over-identification", a term used by (among others) Hoepman (2022) [18][2], refers to parties such as online services asking individuals for a disproportionate amount of personal data as a condition for offering their goods and/or services. This phenomenon is not exclusive to SSI or even digital transactions and has in fact existed for a long time, be it in the form of supermarkets asking for an excessive amount of data for discount pro-

[1]Qualified data = data that has been controlled and verified for authenticity, issuer, date and validity

[2]Not to be confused the psychological concept of over-identification, which refers to a person engaging "in excessive or inappropriate psychological identification" (Merriam-Webster dictionary [29])

grams (see section 2.2.1), having to share extensive documents containing large amounts of data just so that the other party can access one specific piece of information (see section 2.2.2) or states like China requesting a disproportionately large amount of personal and biometric data - among others Iris scans and finger prints from all 10 fingers - as a condition to enter their country.
The big difference to SSI is that currently, the data that users provide, especially online, is often unqualified[3], so they can just lie and provide incorrect data if they deem a specific data item irrelevant for a transaction or service [18]. With SSI, services could try to exploit the ease and comfort of the technology in requesting and sharing qualified personal data to mandate users to disclose certain information even where it is not strictly necessary. Services might take advantage of users simply clicking accept when prompted to provide certain information without paying much attention to what is actually being disclosed or might try to coerce users into disclosing any requested data, e.g. by refusing their services if their requests/demands are denied.
This problem has also been identified by the Dutch state secretary for digitalization. In the appendix of a letter sent by her to the Dutch parliament she expresses her concern that services might exploit SSI to ask for more data than before, knowing that consumers may often not be equipped to determine whether a data request is actually legitimate or desirable [43].
The fact that this data is now qualified removes this option to lie, so when a user feels uncomfortable to share specific data during a transaction due to the context of the transaction, they could now be coerced to share qualified data, which is highly-sensitive due to the level of assurance. Sharing qualified data can poses great risks for users in the case of data breaches and thus needs to be well justified.
It should be noted that over-identification is not always a purposeful action. Some companies have a habit of asking for certain data (like e.g. birth dates) without actually being aware of the purpose of this data collection, thereby unintentionally causing the over-identification of the user. Nevertheless, the coercion for users to have to disclose this data in order to use the service is still there, whether intentionally or not.

It is therefore very important to address over-identification within the design of SSI ecosystems and introduce countermeasures such that users cannot be coerced into disclosing an inappropriate amount of (qualified) data. Here it is important to take the context of a specific data request into consideration since what is an "appropriate amount of data" can vary greatly depending on the context, even when the party requesting the data is the same, since which data a party actually needs is highly dependent on the context/pur-

[3]Unqualified data = uncertified/unauthenticated data

pose of the corresponding data request. For example, it may be deemed appropriate (and often even required) that a company asks its employees to disclose their social security numbers when formalizing their employment but inappropriate to ask customers to provide social security numbers.

1.1 Research Question and Approach

The question that this thesis attempts to answer is the following:

> Can holders be protected against over-asking verifiers by requiring verifiers to first prove to the holder agent (wallet) that they have valid authorization from a recognized governing party to access the requested data in the specific context of their request?

The desired contribution of the research is a protocol between a holder agent (i.e. a user's SSI wallet) and a verifier agent for this scenario, in which the verifier proves to the holder agent that they are authorized to collect the requested data **in the specific context of their request**. This includes both access control and implementing disclosure rules that take the request context into consideration. In other words, the protocol should not only check who is requesting the data but also in what context their request is being made before determining the appropriateness of the request. This protocol should be in line with the principles of SSI. Specifically the principle that issuers of privacy-sensitive attributes should not know how this data is shared should not be violated.

1.1.1 Sub-Questions

In order to succeed with the objective of this research, it is imperative to address the following questions and issues:

1. What criterion is useful and acceptable for determining whether or not in a given context/situation, requested data qualifies as 'over-identification'?
2. Which countermeasures against over-identification already exist and what inspiration can be taken from them?
3. What are the requirements for a successful verifier authorization scheme?
4. What decision rules are needed in relevant contexts to determine the appropriateness of a data request in its context?
5. What should this authorization look like on a technical level (e.g. Verifiable Credential, Certificate, etc.)?

6. Which security properties should a protocol for the verification of a verifiers authorization have and how can they be achieved?

7. Does the proposed protocol provide adequate technical protection against over-asking without unreasonably violating the privacy and interests of the parties involved?

1.1.2 Chapter structure

This thesis is structured as follows:
Chapter 2 will introduce a model of over-identification and provide concrete use cases of over-identification both real and fictional(ized), thereby addressing sub-question 1.
In Chapter 3 we will examine relevant literature and measures against over-identification from non-SSI contexts, thereby addressing sub-question 2.
Chapter 4 will lay down some requirements and considerations regarding a possible verifier authorization scheme which can be used as a starting point and reference for further work regarding the administrative and legal aspects (thereby addressing sub-question 3).
Chapter 5 will introduce a model and syntax for possible decision/disclosure rules in consideration of existing literature (thereby addressing sub-question 4).
Afterwards, chapter 6 will cover the technical aspects, which will be the core contribution of this thesis. This includes defining security requirements based on a concrete threat model as well as a data model for verifiable authorization certificates and the actual authorization verification protocol, thereby addressing sub-questions 5 and 6.
Finally, chapter 7 will reflect on the preceding chapters and the contributions of this thesis while highlighting limitations and aspects for further research, thereby also addressing sub-question 7.

Chapter 2

Preliminaries

Before trying to solve the problem of over-identification, it is important to first define what over-identification even is, why it occurs and in which kind of situations it occurs. For this purpose it is helpful to examine concrete use cases (both real and fictional(ized) ones) and define a model to have a clear idea of what the problem even is. Everything explored in this chapter can then be used as a basis later to ensure that the proposed countermeasure actually addresses the stated problem adequately.

Switch section 2.1 and 2.2?

2.1 Mental Model Over-Identification

There are many situations where the sharing of data is being requested for various purposes. This poses the question of how to distinguish between a situation where a request is actually "over-asking" and where it is perfectly legitimate. Hence, it is helpful to first define a model and assumptions regarding over-identification on which the rest of this research is based.

Data is usually shared as part of an exchange between two parties, where the requesting party is offering some sort of service in return. Such an exchange usually consists of multiple transactions, in which one party acts as the initiator of a transaction (i.e. they request something from the other party) and the other party as the executor (i.e. they either fulfill or deny the initiator's request). A transaction, as defined by Dietz (2006, Chapter 10), consists of three phases, namely an order phase, where initiator and executor try to reach an agreement regarding the intended result of the transaction (including the task of the executor and the time frame in which the executor would need to perform that task), an execution phase, in which the executor actually produces the intended result, and a result phase, in which the parties try to reach an agreement about the result that was actually produced in the execution phase and whether it satisfies both parties [13]. Such a transaction might be a user requesting a specific service from a

relying party (in which case the user would be the initiator and the relying party would be the executor) or a relying party requesting data from a user (in which case the relying party would be the initiator and the user would be the executor).

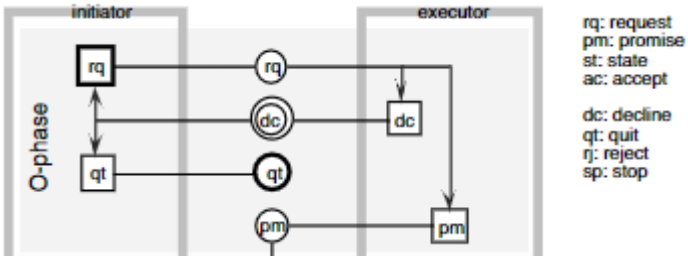

Figure 2.1: Order phase of the standard transaction pattern (Taken from: Dietz 2006, Figure 10.2 [13])

For the purpose of this research, data requests can be seen as part of the order phase (see figure 2.1) of a transaction with a relying party as the initiator and a user as the executor in which the relying party requests data from the user in exchange for a service. This transaction is the context of the data request and is usually part of larger set of transactions, where usually the user has first requested that service from the relying party. It is also within the ordering phase of this transaction that it needs to be evaluated if the request is legitimate or over-asking, before accepting or declining such the request.

The most straightforward reason as to why services might request specific data would be that they cannot perform their service without it. For example, a delivery service cannot actually deliver anything without knowing the customer's address (and possibly their name). Assuming they only request the data attributes that are strictly needed to perform the service, in these situations the processing of the requested data is quite objectively legitimate.
However, there is another factor that influences what data parties may request, namely their risk tolerance. Risk tolerance relates to "[...] the acceptance of the outcomes of a risk should they occur, and having the right resources and controls in place to absorb or 'tolerate' the given risk, expressed in qualitative and/or quantitative risk criteria" (Manoukian 2016) [27]. To reduce its perceived 'transaction risk' to a level that it can handle, i.e. that is within its 'risk tolerance', a party then requests data (among other things).
Parties will always inherently perceive risks that a party takes when offering a service or product to another party. An example of such a risk might be some sort of legal repercussions in case they fail to follow cer-

tain requirements, e.g. if they fail to check a customer's age when selling an age-restricted product like alcohol or cigarettes or if they fail to check the authorization of a person claiming to have been authorized to perform actions on behalf of someone else (see also section 2.2.2 for an example of this)[1].
But it might also be a financial risk (for example rental services like bike or car rentals have an inherent risk of customers just not returning the product they rented, so these services often require customers to share personal data such as name and address that they otherwise do not strictly need for the performance of their service since they are not willing to just accept this risk) or a non-payment risk dependant on the method and timing of a payment (credit card, PayPal and check payments for example can be withdrawn).

However, perceived risks can also be abused by services to collect a great amount of personal data. There is thus a balance to be struck between the risk tolerance of services (or rather the risks that do not fall within their risk tolerance) on the one hand and the privacy of users/customers on the other hand. However it is impossible to define an objective criterion for when a risk is great enough that it warrants the processing of additional data without it being deemed as "over-identification", as different parties would draw the line between tolerable and intolerable risks at different points.
Many situations often are also a moral hazard, i.e. a situation in which one party (in this case relying parties/verifiers) can take advantage of the fact that all negative consequences of their risky actions are incurred by the other party [20]. In this case, any negative consequences that arise from (over-asking) data requests such as data being abused, leaked or stolen hurt the data subjects significantly more than the party collecting the data.
Additionally, there is also a fundamental power imbalance that needs to be addressed here, as the over-asking party is often in a position where they can just deny access to their service entirely to parties that are not willing to comply with their data requests. Currently, citizens can counterbalance this over-asking by entering fake data such as a fake name, fake date of birth, anonymous email addresses and pseudonyms. However, when data is being requested through SSI, services can easily ask for qualified data, i.e. data that is issued by a (trusted) third party with an electronic signature, to prevent individuals from presenting fake data. Therefore other countermeasures against over-asking are urgently needed for SSI interactions.
Some more extreme privacy advocates might argue that requesting any data that is not strictly needed for the performance of a contract qualifies as over-asking while some services might be very paranoid when it comes to

[1]These types of risks could also be classified as a separate category of (legitimate) reasons for data collection, namely *legal obligations*, which would also contain rules and legistlation such as Anti Money Laundering legislation, e.g. in the form of "Know-Your-Customer" (KYC) rules [39]

perceived risks and might argue that mitigating all these risks by asking for more data is perfectly legitimate.
Take the fictional example of an organization, like for example a bank. This organization deals with very valuable assets and highly classified information. Due to this, they are very wary of the potential risk of a rogue employee, who abuses their position to leak information or steal assets. To mitigate this risk, they require extensive background checks for all potential employees. As part of this background check, employees need to disclose a criminal record, a list of all clubs and political organizations they are part of and a list of all their assets (e.g. real estate properties, stocks). While this is not strictly necessary for the completion of the underlying transaction (employing a new employee), they would argue that all this information is necessary to mitigate the risk of rogue employees.
Many might argue that, while mitigating the risk of rogue employees is indeed legitimate, the extent of their background check is not proportionate in relation to that underlying risk. They might argue that providing a criminal record is entirely sufficient for this purpose and that their other requirements are unnecessary and unduly violate the privacy of prospective employees (potentially even their legal rights when it comes to disclosing membership in political organizations). Other organizations on the other hand might want to go even further as they perceive the risk of rogue employees to be much greater. They might even require a full psychological profile of prospective employees in addition to all the requirements the bank has for its employees. They would argue, that they cannot trust employees of which they do not know their mental health status. While this is a quite extreme example, it shows that determining if a specific data request qualifies as over-identification or is justified by the risks of the requester is inherently subjective and is also highly dependant on the situation.

There is also a third possible motive for data collection, namely *personalization/marketing*. These motives may also be legitimate purposes if and only if they are opt-in (i.e. they are optional and users need to explicitly state that they are okay with sharing data for these purposes) and data requests tied to marketing should be completely separate from those that are mandatory for the usage of a service, such that users cannot be coerced into accepting these requests (which unfortunately is often not the case). Even in these cases it can be argued that data requests should still be proportional in which data is being asked of users, since organizations and services often tend to collect an excessive amount of data for marketing purposes often to a degree that even users that would generally be okay with sharing data for better personalization would be uncomfortable with. User preference is an important factor here, since individual users might be willing to share different sets of data items for marketing purposes than others.

Acknowledging this inherent subjectivity is important when talking about over-identification. However, in practice some party would need to decide whether a request is legitimate/proportional or over-asking while striking an appropriate balance between the privacy of civilians and the risks of services in line with societal norms and values. In the EU, there is some normative framework for this in the form of (among others) the GDPR and its various implementations in individual member states (which might be stricter than the GDPR itself), the ePrivacy directive ([32]) and rules for specific sectors (e.g. telecommunications, banks, healthcare, etc.). In the case of the GDPR, this normative framework is enforced through various data protection authorities in the individual member states, such as the Autoriteit Persoonsgegevens in the Netherlands, CNIL in France or "Der Bundesbeauftragte für den Datenschutz und die Informationsfreiheit" in Germany. These authorities often interpret the normative framework of the GDPR quite differently in similar situations. The French data authority for example is known to be very strict [23], whereas other data protection authorities may be much more lenient in their interpretation of these norms or lack the resources to go against anyone but the biggest offenders (the Irish DPA for example has been sued for their failure to properly investigate complaints against Google [26]).

However, it is not just the data protection authorities that have different ideas about the normative framework and about what is "perfectly legitimate" and what is "over-asking". On the one hand, there are privacy advocates, consumer organizations and privacy-conscious consumers. For these parties, the interests and the privacy of consumers are the highest priority, therefore for them a "perfectly legitimate" request needs to first and foremost respect the privacy as much as possible and should only ask for data that is either strictly needed to perform a service or strictly needed to mitigate a very great, unambiguous and well-defined risk (i.e. a risk that is not vague).

Less privacy-conscious consumers (which are arguably the majority of consumers) might still value their privacy, but to a much lesser degree and thus might have a broader interpretation of what kind of requests they consider legitimate. However, even these consumers might still be uncomfortable with particularly data requests such as the background check example form earlier and feel helpless against the perceived coercion.

On the other hand, there are private businesses and organizations who primarily care about their own profit and their own financial and legal risks. Some of them might be more benevolent in regards to the privacy of individuals and might genuinely try to minimize the amount of data they collect, although this amount they consider to be legitimate might still differ from what privacy advocates might consider legitimate. Other organizations might be less benevolent and might even try to financially profit from collecting and subsequently selling user data (e.g. Facebook [28]). These

organizations might try to use Article 6(f) of the GDPR, which states that organization might collect personal data if their processing is "[...] necessary for the purposes of the legitimate interests of the controller or by a third party" [33], to argue that their data requests serve their legitimate business interests and are thus not over-asking users, even though many other parties (including many data protection authorities) might disagree with this line of reasoning.
Furthermore, just like verifiers, citizens also have inherent risks in transactions where they need to disclose personal data that influence their views on the proportionality of data requests. For example the data they share with a service might be used for more purposes than originally stated by the service without their knowledge, purposes which they did not consent to and possibly would not have consented to if asked. Additionally, even when data is only used for purposes which the data subject explicitly consented to there is still the risk that their data gets stolen in case of a data leak, so security considerations also play a role. This again highlights the inherent subjectivity when it comes to determining when the collection of personal data is appropriate/proportional.

2.2 Use Cases and Contexts

To provide an even better idea of what over-identification looks like in practice, it is helpful to identify specific use cases in which over-identification might occur and what the cause of the problem is in these situations. The purpose of this is to identify different types of over-identification, which would need to be addressed and give a provide a better idea of the scale and complexity of the problems as well as motives behind it.

2.2.1 Albert Heijn

The Dutch supermarket chain Albert Heijn offers a bonus card program for customers, which can be used to get discounts on certain products and use the portable self-scanners. To activate one's bonus card, one currently needs to create an account on the Albert Heijn website and share one's name, birth date, address, e-mail and phone number. This seems rather disproportionate, as neither is all this data strictly needed for the activation of a bonus card nor is there an acute risk here that Albert Heijn would need to mitigate (the only possible "risk" would be people having multiple bonus cards, which is arguably tolerable for Albert Heijn).
It makes sense that an email address is being requested for example, since Albert Heijn informs bonus card customers about personal discounts every week via email. But there is no clear purpose for why a home address is being requested here, since there is neither something that Albert Heijn needs to physically deliver to the customer's house (at least not in this context)

nor is there a specific risk that Albert Heijn would mitigate by the collection of an address.

Section 4.4.3 of the Albert Heijn Privacy Policy ([10]) offers some insight into the true motives behind why this data is being collected. The core purpose for data collection in the context of the bonus card is, as elaborated in section 4.4.3.1, that Albert Heijn wants to give customers personalized discounts, thank-you notes, birthday surprises, promotions and send "relevant information" about AH products and services as well as personalize their website further on the basis of the personal data they collect [10].
In addition to the general personal data they collect as stated above, they also track all the products one buys (with time stamp and location (whether physical, in the AH app or the website)), how much money one spends, which products one inspects on the website and the entire behaviour of a user on their website. As stated in section 5 of the privacy policy, data regarding the user behaviour on the AH website is even shared with third parties such as Google DoubleClick, Facebook and Salesforce such that they can analyze the data[2]. The privacy policy does not really specify why all this data specifically is absolutely necessary. The only exception is the birth date for which they state that it is collected so that they can check if they may offer discounts on alcohol and for special promotions on the customer's birthday [10]. However, it remains unclear from the privacy policy why the home address of customers is needed for this and how they justify tracking customer behaviour to such an extreme degree[3].

While not explicitly stated by the privacy policy, the underlying motive is pretty clear. Albert Heijn wants to create rather extensive customer profiles such that their personalized promotions and discounts motivate customers to spend more money at Albert Heijn. The more data Albert Heijn collects, the more accurately they can predict how they can get customers to spend as much money as possible at their stores. They are thus incentivized to collect as much personal data as they possibly can in order to maximize their revenue and therefore also their profits.
Therefore, economic/monetary and marketing motives are the driving factor behind Albert Heijn's data collection policies. This is one of the most common causes of intentional over-identification, since personal data is a very valuable asset for organizations in a capitalist society as the more data they collect the more money they can make.
However adding SSI in this use case would introduce another issue here,

[2]While they do not directly share directly identifying information such as name or address, these third parties can still collect them from the AH website via cookies.

[3]It might for example be argued that to offer personalized discounts it would be sufficient to see what types of products customers buy (not even the specific product, but just a generic product type).

namely assurance. As the Albert Heijn does not strictly need most of the data collected in the context of the bonuskaart and only collects this data for more "personalization" (i.e. marketing), data requests relating to this do not warrant a high level of assurance. However with SSI, customers would inevitably disclose qualified data of high assurance to Albert Heijn, which is quite disproportional in relation to marketing goals. Therefore users still need to have the option to "lie" and provide unqualified or even completely fictional data.

2.2.2 Notary

In the Netherlands, the main task of a notary is to record specific agreements and declarations in so-called notarial acts ("notariële akten")[4]. Examples of such agreements would be wills, living wills (authorizing another person to take decisions on one's behalf, e.g. in cases where one ends up in a coma or suffers from dementia), real estate purchases, mortgages, stock transactions and specific authorizations and provisions. These acts contain an extensive list of identification data, such as all first names, last name, place of birth, birth date, current address and more, of all parties concerned by the specific agreement. For example, when someone buys a house together with their spouse, the notarial act documenting the purchase agreement would contain the data of the buyer themselves, of their spouse (as they are also buyers) and the person selling the house. The creation of such a notarial act is often required by law to validate an agreement/a declaration[5].
Article 40 of the Dutch "Wet op het notarisamt" (Notaries Act) from 1999 defines which data notarial acts must contain [35]. This article states that notarial acts need to at least contain the following data (in addition to the agreement/declaration documented by the act in question):

1. Last name, first name(s) and place of business of the civil-law notary before whom the act in question is drawn up[6] (The same data also for an assistant civil-law notary or an acting civil-law notary if they were involved in the creation of the act)

2. Last name, first name(s), date and place of birth, place of residence with address and civil status of all natural persons who appear in the act as parties[7]

3. Legal form, name and place of residence with address of all legal persons who appear in the act as parties [8]

[4]See https://www.knb.nl/english/the-notary/task-of-the-notary
[5]See https://www.knb.nl/english/the-notary/notarial-acts
[6]See Article 40(1) "Wet op het notarisamt" [35]
[7]See Article 40(2a) "Wet op het notarisamt" [35]
[8]See Article 40(2b) "Wet op het notarisamt" [35]

4. For all natural and legal persons who are defined by the deed to represent the aforementioned parties: The same data as defined in Art. 40(2a) and Art. 40(2b) (except for their legal status) and the grounds for their authority (address of an office may be given here instead of a residential address for natural persons)[9]

5. Last name, first name(s), date of birth and place of birth of each witness in whose presence the act has been drawn up[10]

6. Place, year, month and day on which the act was drawn up and registered[11]

7. If the act was drawn up in a language other than Dutch: last name, first name(s), date and place of birth and residence of the interpreter-linguist[12]

8. In the event that mention of the time of signing of the deed by the civil-law notary may be important in connection with the registration in the public registers or for any other reason, that time shall also be stated[13] [35]

These notarial acts, or rather some of the data contained within them, need to be shared in specific situations. One such situation, where over-identification occurs, would be the passing away of a person that has set up a will with a notary. This will lists all the possessions (such as bank account, stocks, real estate properties, etc.) of the deceased and defines how these possessions should be allocated as well as who will take care of all the remaining matters regarding the inheritance of the deceased. This authorized party is called an "executor" ("executeur/gemachtigde"). This executor then needs to prove their authorization to various concerned parties such as the deceased's bank or insurance. Currently, they need to present the whole notarial act to prove this, thereby sharing all the data contained within it while the other party actually only needs to know that a) the subject of the act/will has passed away and that b) the person standing in front of them has been authorized to take care of the deceased's inheritance by the deceased themselves.

A very similar situations are so-called "living wills" ("levenstestamenten"). A living will is a document that authorizes other parties (executors) to take decisions on someone's behalf when they are unable to, e.g. because they

[9] See Article 40(2c) "Wet op het notarisamt" [35]
[10] See Article 40(2d) "Wet op het notarisamt" [35]
[11] See Article 40(2e) "Wet op het notarisamt" [35]
[12] See Article 40(2f) "Wet op het notarisamt" [35]
[13] See Article 40(3) "Wet op het notarisamt" [35]

have fallen into a coma after an accident or are suffering from a brain disease such as dementia or Alzheimer. The subject of the living will can also define multiple different executors with different authorizations if they wish to. For example, they may authorize one person to consult their medical dossier/file, but whenever a decision regarding the treatment of the patient needs to be taken, different executors would need to agree to such a decision first. In this situation, the executors also currently need to share the entire notarial act regarding the living will to the doctor/hospital to prove that they have been authorized by the patient to take decisions on their behalf. Just like before, this means that a lot of unnecessary information gets shared with third-parties (in this situation the doctor/hospital).

As these two examples demonstrated, the core of the problem at hand is always the same. Some external party needs to process specific data that has been documented in a notarial act (which data exactly is needed depends on the context), but in order to provide this information one needs to share the entire notarial act, which in addition to the necessary data also contains a lot of data that the requesting party does not need (and also didn't even really ask for). However, this is currently the only way that the necessary information *can* be obtained at all. Thus, this is a clear example of unintentional over-identification.

2.2.3 Online Application Forms (by Anciaux et al. (2013) [4])

A different type of over-identification was identified by Anciaux et al. (2013) in the context of online application forms (see also section 3.2). They examined application forms for online services such as social care, tax services and bank loans and noticed that far more data needs to exposed on these application forms than is actually needed for the individual situations of applicants. This was caused by the fact that, depending on the individual situation of the applicant, some data fields become obsolete and others become mandatory and it was impossible to determine beforehand which data fields are actually needed and which are not [4].
In this example, there is a chain of dependency that influences which data is actually needed by the relying party and which is not. This can often cause over-identification, since the party requesting the data cannot predict how the dependency chain will apply to the situation of a specific individual. This type of over-identification is very common in the public sector, especially when it comes to social programs or welfare. Due to the chain of dependency, these cases are also harder to prevent than the previous examples since in the previous examples it should be obvious beforehand which data is actually needed whereas here this is not obvious.
Anciaux et al. have proposed their own solution to this sort of over-identification,

which introduces a minimal disclosure model based on specific collection rules that model such dependencies (for more details, see section 3.2). This approach is meant for non-SSI online application forms and involves removing unnecessary data items on the basis of the collection rules for the application form in question after applicants have filled out the form but before the data is disclosed to the service provider. In their demonstration, they introduced a trusted third party in the form of a smartcard to handle this process of eliminating unnecessary data items [4]. In SSI on the other hand data is almost always transferred more or less directly between holders and verifiers with only the wallet acting as a sort of intermediary.

2.2.4 Mortgages

Most people that want to buy a house do not have the cash ready to pay for a house out of their own pocket, so they will have to apply for a mortgage, the height of which is dependent on a whole bunch of factors such as one's age and income.

In the Netherlands people often have to get an official mortgage advice from a qualified mortgage advisor first, unless one can prove that they have very specific qualifications (e.g. if they are mortgage advisors themselves or prove their expertise). Getting an advisor is technically not required by law, but mortgage lenders do have a so-called duty of care to ensure that the mortgage a client wants to take out is actually a good fit for them, which they cannot guarantee if the client has not proven their expertise or have come through an advisor[14].
Clients go through a multi-step process before being granted a mortgage. This process goes as follows:

1. A client asks an advisor for an informal indication on the maximum mortgage amount to see what kind of house they are able to buy. During this step the advisor will ask their clients about specific personal data (often without the client having to provide proof at that point) in order to provide the client insight in the bandwidth they are able to spend on buying a house such that the client can comfortably make a first offer on a house. For this purpose the advisor often asks for data such as their total capital and assets, current income and income during the last five years and total debt accumulation (how much debt does the client have and where?).

[14]There are so-called Execution Only Mortgages, where clients accept that they are aware of the dangers of getting a mortgage without an advisor. However, even then they still have to pass a test to prove they are sufficiently financially literate [7]. This test is really difficult for most people, so these mortgages only make up a small percentage of the market.

2. When the client has won the bid on a specific house, they will engage with the mortgage advisor for a formal advice. At this stage, the client has to provide evidence regarding the data asked by the advisor in step 1 to the advisor so the advisor can verify that data. Advisors will also often ask clients about their pension accrual (with evidence) at this point, more because they can, not because they have to. For starters of a young age, they often do not ask about the pension accrual, since it is often not really needed (e.g. for a 25 year old who is planning to pay back the mortgage within the next 30 years the pension is completely irrelevant).
   It is also at this stage that the client has to fill in a standard form with all this information. This form is most often the same for every client and always asks for the same information independent of the client or the case. This is not ideal in terms of data minimization as, for example, a starter home requires different information than a mortgage for renovation or a second home where the surplus value of the first home is used as capital, but the same form is used in all these situations and thus also asks for information that may not be needed for the specific case at hand. This often leads to a similar form of over-identification as in the example provided by Anciaux et al. (see also sections 2.2.3 and 3.2) [4], since not all the information that needs to be provided on that form is actually relevant for every client (information about pension and retirement for example are not really relevant for young applicants) and the set of information that is actually needed depends on the individual client.
   The advisor then goes to a bunch of different mortgage lenders ("hypotheekverstrekkers") who then give back an offer with a certain interest rate to the advisor based on information about the client that the advisor shared with the lender. This data is based on what the customer has filled in on the aforementioned form, which the advisor transferred to their system and then sent to the mortgage lender via HDN (Hypotheken Data Netwerk) or services like Skydoo[15]. The data is then also stored in the lender's own system, which means it is stored in multiple different places.

   After having received enough offers from different mortgage lenders, the advisor meets again with their client such that they can select the most suitable offer for the client together. Most advisors will also look for the lowest net monthly expenses ("nettomaandlasten") for the mortgage of the client, since the client can get a subsidy from the Dutch tax office (Belastingdienst) on the mortgage based on their tax

[15]Skydoo is a platform that uses business rules to check whether the correct information and documentation has been received so that processing can begin.

bracket and on their net monthly expenses incurred due to the mortgage. Higher monthly expenses can be a good thing since they can give more flexibility[16], though that is not always the case. Good advisors will also often ask clients about their future plans, such as if they have any expensive hobbies, if they plan to get married and/or have kids and if they are planning any extensive travels, to better determine which offer suits the client.

3. After having selected and accepted a specific offer, the client then goes to the mortgage lender. They then have to provide extensive proof and breakdown of their capital, their income during the last five years and their debt to the mortgage lender such that the lender can see if the client can pay back the mortgage with interest in the coming years.

   It is however questionable whether they actually need that much data since the lender only has very limited risks here. If the client cannot fulfill their duties in paying back the mortgage, the lender can just take the collateral (in this case the house) and sell it in an open auction, so the loss on the part of the lender is very minimal at most and often they do not end up in a worse position than before.
   It is questionable whether this risk actually warrants asking for such an expansive breakdown of the client's data. Regarding the client's income for example, the only thing the lender really needs to know that the loan-to-income quota is not higher than the law allows[17] to cover their risk. Regarding the client's debt, they also only really need to know that the amount of debt is not above a certain limit. Therefore, they do not really need to know the credit card balance of the client or how much study debt they still have, so a breakdown of the client's debt is not really necessary. All the data that is disclosed to mortgage lenders here is essentially only used to calculate these values (loan-to-income quota, debt not above limit, etc.) anyway.

During the last step, we have a case of over-identification that is caused by disproportionate risk coverage, i.e. the amount and sensitivity of data that is collected to mitigate a risk is disproportional to the potential damages caused by the risk. However this is not entirely the fault of the mortgage lenders themselves. Regulators in the Netherlands require mortgage lenders to show the specific data on which a certain risk analysis is based due to their duty of care [3]. The regulators currently would not be satisfied

[16] It might for example be beneficial for a client to have higher monthly payments to provide for more flexibility in moving to another house in the not so distant future, since some mortgage lenders have so-called penalty clauses ("boete clausules") when one want to pay off their mortgage before the end of its runtime.

[17] The legal limit of this quota depends on the individual client and their income (See https://www.dutchhousingpolicy.nl/topics/financing/loan-to-income-lti)

with a "green tick" on the loan-to-income quota given by e.g. the Dutch Employee Insurance Agency UWV. Therefore, to start making this process more privacy-friendly and strip it of this over-identification, regulators will have to be included and authorize changes to this process.

Chapter 3

Related Work

Behind every data request, there is usually a specific goal the requesting party is trying to achieve for which the requested data is needed. For this purpose there is a legitimate data request and an adequate level of assurance and sensitivity. Over-identification occurs when not strictly necessary attributes or attributes of an inadequately high level of assurance (e.g. in situations where self-asserted data would be completely sufficient) or sensitivity is being requested. This is in no way a new phenomenon or unique to SSI environments.
This chapter examines how over-identification is handled in other (non-SSI related) contexts and which inspiration can be taken from these approaches for the context of Self-Sovereign Identity.

3.1 Access Control

One dimension to addressing over-identification is the implementation of access control mechanisms, i.e. restricting access to sensitive data to a limited set of parties.

A prominent example in which access control mechanisms are implemented are electronic passports and ID cards. In the European Union, two different access control mechanisms are being used by electronic travel documents: Basic Access Control (BAC) [31] and Extended Access Control (EAC) [16]. According to the ICAO specification, Basic Access Control "is a challenge-response protocol where a machine (RF) reader must create a symmetric key by hashing the data scanned from the MRZ [Machine-readable Zone] in order to read the contactless chip" (ICAO 2007) [31]. This mechanism is meant to prevent skimming as passports need to be physically opened and scanned in order to access data stored on a passport's chip. However it does not authenticate who is reading the passports data [19].
To offer better protection, Extended Access Control was introduced to pro-

tect highly sensitive biometric data such as fingerprints and iris scans. This mechanism consists of two phases: chip authentication and terminal authentication. Chip authentication serves to prove that the chip inside is genuine to protect against cloning. This involves the chip and the scanning terminal exchanging session keys using Diffie-Hellman key-exchange and the chip subsequently implicitly proving knowledge of the session key by successfully communicating using the session key [19]. This phase "also enforces strong encryption and integrity protection of the transmitted data" (BSI 2016) [16]. The actual access control part is performed during the terminal authentication stage, in which, during a challenge-response protocol, the passport chip verifies that the terminal has the proper authorization to access sensitive data stored on the chip [15]. The protocol starts with the terminal sending a certificate chain to the chip. The root certificate of this chain is issued by the country that has issued the password and is therefore verifiable with the CVCA public key that is stored on the chip [15]. The root certificate issues so-called Document Verifier (DV) certificates for each country that is intended to be granted access to the data on the chip. These DV certificates in turn are then utilized to generate so-called Inspection System (IS) certificates. These IS certificates are distributed to the devices that are used to read passport data and, if valid, grant access to the sensitive chip data. The authenticity of DV certificates and thus also IS certificates issued by a specific DV certificate can be verified by any passport of a particular country [19].
After verifying these certificates, the chip on the passwords then extracts the terminal's public key PK_{PCD} from them and generates a random challenge r_{PICC} which is then send to the terminal. The terminal then sends back a signature $s_{PCD} = \mathbf{Sign}(SK_{PCD}, ID_{PICC}||r_{PICC}||C)$ over the challenge, with ID_{PICC} being the identifier of the chip the terminal is communicating with and C being the terminal's compressed public key. The chip then finishes the Terminal Authentication protocol by validating the signature [15].

The Terminal Authentication phase of EAC for electronic passwords provides an example of a protocol in which the identity of a verifier (i.e. a party requesting access to specific data) is being verified first before granting them access to the requested data and a verifier authorization scheme (through the certificate chain). This approach provides partial protection against over-identification in the sense that it restricts access to parties that are actually intended to get access to sensitive biometric data.
Since over-identification is not just about who is requesting, but also for what purpose and in what context data is being requested, this by itself is not enough to offer full protection against over-identification. In this case, the authentication of the terminal is implicit since accessing data on passports is the only thing those terminals can actually do, so there are no diverging contexts that would need to be addressed. However it does gives an idea for

how the access control dimension of protecting against over-identification can be approached by introducing an authorization scheme.
In this context, the authorization is granted by the issuer of the passport.In a Self-Sovereign Identity context, this would not be ideal, since it would be somewhat impractical as verifiers would need to get permission from multiple different issuers for different purposes. Here, another party would be needed to grant authorization to verifiers. This will be explored in more detail in chapter 4. Nevertheless, EAC provides a technical implementation and protocol where an authorization is checked, from which inspiration can be taken even for the context of SSI.

3.2 Disclosure rules

Even though access control is an important aspect in addressing the issue of "over-identification", it is certainly not the only dimension to it. Defining and implementing specific rules for the disclosure of sensitive information that focus more on the context of an access request and less on the requesting party itself, are also a way to address the underlying problem.

One approach in this direction was taken by Anciaux et al. (2013) for the context of online application forms (e.g. for social care, tax services and bank loans). Here they identified a case of over-identification, as not all the data items that were collected were needed in every individual case, as some data was only needed in specific situations [4].
They provided an example where a dependant was eligible for financial support for a home aid if they either a) had a pension of less than €30.000 and an age of more than 80 years, b) a pension of less than €10.000 regardless of their age, or c) have at least two abilities that they cannot do anymore (e.g. getting dressed or bathing by themselves) ("lost abilities"). Depending on the individual situation of the applicant, the minimum set of data that needs to be disclosed might be different. For example, an applicant who is 60 years old, has a pension of €40.000 and has 2 "lost abilities" would only need to disclose that they have 2 lost abilities (as they fall under category c), whereas an 81 year old with a pension of €25.000 would need to disclose both their age and their pension (since they fall under category a) [4].

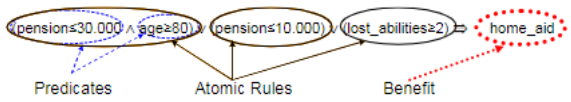

Figure 3.1: Collection rule example (figure 1 from [4])

Anciaux et al. translated these kind of situations into collection rules

in the form of Boolean DNF formulas (see figure 3.1). For the example described above they formulated the collection rule $((pension \leq 30.000 \wedge age \geq 80) \vee (pension \leq 10.000) \vee (lost\_abilities \geq 2)) \implies home\_aid$, which consists of three atomic rules modelling each category. These atomic rules give different minimal data sets that would need to be disclosed if an applicant falls under a specific category. These kinds of decision rules are then used in their system called *Minimum Exposure* which allows programs to automatically omit unnecessary data items via these rules [4].
In the original proposal by Anciaux et al. the actual data items that were determined to be necessary in an individuals situations were still disclosed (probably because the actual values of the items are still needed for further computations, e.g. to compute to how much financial support for home aid an applicant is entitled to) [4]. However, when executed locally (or by a trusted third party like a smartcard in their proposal or a wallet), the set of minimum disclosure in their example could be reduced to a simple 1 bit outcome, without disclosing the actual data items in situations in which a simple true/false answer is sufficient for the relying party (e.g. for a simple eligibility check).

A different approach to minimizing data disclosure that focuses on the client-side of credential based interactions by means of graph-based modelling has been proposed by Ardagna et al. (2012). This approach is centered around so-called sensitivity labels and context restrictions that users can assign to specific attributes, credentials or specific combinations thereof to limit how and when this information may be disclosed [5].
Sensitivity labels in this context are information privacy preferences users can associate with specific attributes, credentials or combinations thereof to indicate how sensitive they perceive their disclosure to be. For instance, users might prefer to disclose an email address over their physical address or a self-assigned username over their actual name. Sensitivity labels can be used to express these kind of preferences in a machine-readable manner [5].
Ardagna et al. keep the definition of these labels fairly generic such that these preferences could be expressed in whatever format one deems suitable, such as for example classic multilevel security classifications (e.g. top secret, secret, etc.) or integers, and that different methods of classifying compositions of sensitivity labels can be used (e.g. least upper bound, sum, maximum, minimum, etc.). Their scheme also allows for combinations of credentials and/or attributes to be assigned their own label in case they are deemed *more* sensitive than the composition of the individual labels (e.g. the combined disclosure of a medical dossier and a name might be more sensitive when disclosed together than the composition of their individual sensitivity labels) or even in case they are deemed *less* sensitive than the composition of their individual labels (e.g. the combined disclosure of a

phone number and a country might be less sensitive than the composition of their individual labels might be, since the phone number might already disclose the country anyway). Users can even outright prohibit the disclosure of certain combinations of attributes when they do not want them to be associated with each other [5].
Context restrictions on the other hand serve the purpose of limiting the disclosure of specific attributes or credentials within a set of interactions, based on the domain/sector of the server and the information held by the user. A user may for instance be willing to disclose a medical file to a hospital but not necessarily to a pharmaceutical company. Within the proposed scheme users can assign servers to specific categories (e.g. governmental, medical, bank, etc.) to then restrict the disclosure of specific attributes or credentials to specific categories of servers only. Categories can also be set up in a hierarchical manner (see figure 3.2), such that a broader category such as medical can have more specific sub-categories such as hospital, general practitioner or insurance, if a user wishes to have more specific context restrictions. Users can also further limit access for server of a specific category to specific types of transactions/interactions (e.g. payment, diagnosis consult, reservation, etc.). For instance users may restrict the disclosure of credit card information to financial service to payment transactions only. Just like categories, transactions can be defined in a hierarchical manner with more specific transactions as sub-types of broader ones [5].

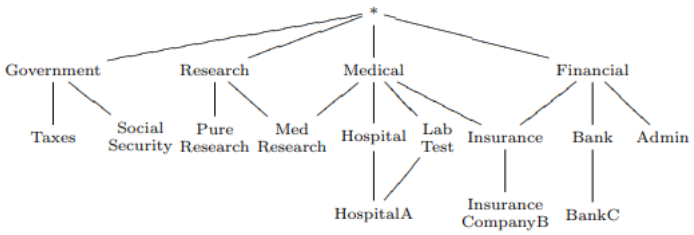

Figure 3.2: Example hierarchy of context categories (figure 5 from [5])

In addition to these to mechanisms Ardagna et al. also allow for users to keep track of a communication state within some time frame, i.e. which information they have already disclosed to a server. This is meant to serve two purposes, namely a) minimizing the information disclosed to a server overall and b) preventing linkability, i.e. preventing a server from linking two unrelated disclosures to the same client [5].
All these elements (sensitivity labels, context restrictions, communication state) are then used in their minimal disclosure algorithm that upon a given request computes the minimal data set that fulfills the request and does not violate any restrictions set by the client [5].

The first approach by Anciaux et al. (2013) [4] provides an example of how decision rules can be applied to limit the data disclosed in specific scenarios. It could be argued that in their example the atomic rules constitute contexts that are sub-contexts of a larger context (home aid in their example). These rules then give the necessary data set for this context. In that sense, some inspiration for the decision rules needed within SSI can be drawn from this approach.
The second approach by Ardagna et al. (2012) [5] also provides a practical example of how context can be defined in a hierarchical manner and disclosure restrictions based on the specific context of a data request. Their disclosure restriction work in a blacklisting manner, i.e. they prohibit the disclosure of specific data in a context. For SSI, a whitelisting approach would probably be better suited to make it explicit which kind of data is actually needed, but for specific scenarios there might also be merit to have some blacklisting rules as well. They also provide an example of a model where users can express their personal preferences regarding the disclosure of specific attributes and/or credentials.

Chapter 4

Verifier Authorization

As stated in the introduction of this thesis, the focus is on one specific measure to combat over-identification, namely the authorization of verifiers by a recognized governing party, i.e. a trusted third party. However, this concept is somewhat vague and can be set up in a lot of different ways, so it is important to define what such an authorization scheme would need to look like such that it can adequately protect individuals against over-asking verifiers while taking the specific contexts of requests into consideration. It needs to be defined what the authorization of verifiers should achieve, what conditions need to be fulfilled in order to achieve this goal and which parties might be suitable to act as authorizers.

4.1 Possible candidates "Authorizer" Role

In section 2.1, it was shown that there is an inherent complexity and subjectivity to identifying over-identification. Accordingly, any verifier authorization scheme would reflect the subjective perception of legitimate and over-asking data requests of the party that provides authorization. This places a great amount of responsibility on those authorizing parties as they would need to protect the privacy of individuals while taking the risks perceived by services/organizations into consideration.
As pointed out earlier, there is a fundamental power unbalance between individuals and verifiers. The objective of authorizing parties needs to be the protection of the rights and interests of the vulnerable party in this relationship (individuals). They therefore need to very carefully evaluate if a perceived risk really warrants collecting the attributes an organization wants to request to mitigate their risk and whether the risk warrants the collection of additional data, that is not strictly needed for the underlying transaction to succeed, at all and carefully design decision models for every possible context (see also chapter 5).

They are a couple of parties that could act as authorizers. Some of them might be more suited than others, so it is helpful to explore why certain parties may or may not be suited to bear the responsibilities. This is not an exhaustive analysis of all possible scenarios as that would be out of scope of this thesis. However, it is meant to be a starting point which can be used as a reference in further research into the governance and legal aspects of verifier authorization.

4.1.1 Data Protection Authorities (DPAs)

One candidate (at least in the European Union) to fill the role of "authorizers" are data protection authorities. DPAs are independent public authorities (one in each member state of the European Union) with the task of supervising and enforcing data protection legislation. They can also give advice to other parties about data protection matters and are also responsible for handling complaints about the violation of data protection laws [11]. Granting authorizations to verifiers based on their perception on whether the verifiers data request is legitimate with regards to the purpose/context of the request therefore would seem like a logical extension of their current responsibilities.

Having DPAs fill the role of authorizer has a number of advantages. For one, they act mostly independent and cannot be controlled directly by national governments, so their only responsibility is to fulfill their legal duties and enforce the rights of citizens in regards to data protection. So the potential for conflicts of interest is comparatively low (though lobbyism could be a problem). They can also be held accountable by courts if they do not fulfill their legal duties adequately (the Irish DPA for example got sued for failing to properly investigate Google after complaints about potential GDPR violations [26]). DPAs have even fined government agencies for GDPR violations in the past, like in April 2022 for example when the Dutch DPA fined the Dutch Tax and Customs Administration (Belastingdienst) €3.7 Million for illegally processing of personal data in and wrongly putting about 270,000 people on a blacklist which kept records of fraud [34].
Since the primary objective of DPAs is to enforce existing data protection legislation, their main priority - at least in theory - are the interests of citizens and consumers and not big corporations, which is an important quality for this authorizer role. Of course, they are still dependant on data protection legislation from the EU and its member states, so adequate legislation and an adequate legal framework regarding the legitimacy of data request within SSI specifically would be needed (although arguably this would be necessary anyway even if one does not introduce DPAs as authorizers).
DPAs would also have a very clear and unambiguous claim to authority, as their role and authorizations are clearly defined by EU legislation. Hence,

they have a solid trust relationship with all stakeholders involved, such that authorizations granted by them can be deemed trustworthy by wallets and individuals. The EU could even require wallets to accept legitimate authorizations from DPAs by law if needed.
This solution would also work very well in terms of interoperability between different SSI systems. The authorizations obtained from DPAs are not bound to one specific wallet or SSI implementation and there are little obstacles to their usage within all sorts of SSI ecosystems, at least within the European Union.
The main challenge in terms of interoperability would be that due to different jurisdictions and implementations of EU legislation within member states, companies would need to get authorizations from the corresponding DPA for every EU member state they wish to use SSI to request data in. However this is arguably still very feasible, as an organization would just need one authorization per request per country. Many smaller verifiers might also only operate in one country anyway and those that operate in multiple countries should have enough resources that this should not pose a big problem for them.

The biggest obstacle to making DPAs responsible for the authorization of verifiers would be that many DPAs are currently understaffed and are already struggling to keep up with their current work load [36]. Giving them the additional responsibility of having to authorize or deny specific data requests by many different organizations would require hiring and training a great number of additional specialized staff. This would in turn require significantly more investment by the EU and national governments into their DPAs due to the rising personnel costs.
Just like with any public authority there is a risk of corruption. This risk seems to be relatively low as there have not been many known instances of corruption within DPAs, however the Irish DPA for example has been facing facing a criminal complaint over alleged corruption and bribery for trying to coerce the European privacy campaign group nyob lead by Max Schrems to prevent them from publishing documents concerning GDPR complaints about Facebook. [25] While European countries are comparatively uncorrupt in relation to other countries worldwide according to the Corruption Perceptions Index, there are still some countries within the EU such as Bulgaria, Greece or Hungary that have significantly more problems regarding corruption than others such as the Netherlands, Denmark or Sweden [21].

In theory, Data Protection Authorities appear to be a suitable candidate to fill the role of authorizer due to their unique legal position and lack of own business interests that would conflict with the privacy interests of private citizens. However, there remain reasonable doubts about the practical feasibility of delegating this task to Data Protection Authorities. They

would need significantly funding from the EU and national governments to acquire more resources and hire more employees. And even then, they would need some sort of automated process as the sheer number of requests they would have to handle might be too much to be processed one-by-one by a human employee. Therefore, they might not be the most suitable candidate in practice.

4.1.2 Wallet providers

Another potential candidate to fill the role of authorizer would be wallet providers. The Dutch telecommunication provider KPN for example already acts as a sort of authorizer for their digital wallet "PiM ID". KPN determines for every verifier and for every action which data attributes are needed for an action and grants access rights to request specific attributes to verifiers based on their decision[1]. This works well for their individual system as these access rights are managed directly by KPN and can automatically be checked by their wallet without much overhead.
The benefit of conferring this responsibility to wallet providers would be that since it would ultimately be wallets that have to check whether a verifier has legitimate authorization for their data request, having said authorization be granted by wallet providers themselves would make checking the legitimacy of authorizations much more straightforward than it would otherwise be.

There are however many drawbacks to this solution. While it might work well for individual wallets on a smaller scale, this approach poses many challenges on a macro scale due to the (probable) lack of interoperability. In the worst case, verifiers might need to obtain authorization for every wallet they wish to support, which depending on future developments in the field of SSI might be a not insignificantly high number. Since different wallet providers probably have different ideas about when a data request is legitimate and when it is over-asking, this is not an unlikely scenario, if the responsibility of authorizing specific data request for specific verifiers were to be placed on wallet providers.
Even in the best case there might be individual agreements between wallet providers to accept authorizations from the other party, but this can quickly become a a complex web of agreements such that it might become hard to keep track of which authorizations are accepted by which wallet.
There is also the potential for conflicts of interest here, since many wallet providers are private businesses that may have business relationships with some verifiers and sometimes even act as verifiers themselves. Individuals would be completely dependant on the benevolence of the wallet provider in regards to their privacy, which has huge implications for the trust rela-

[1] See https://pim.app/

tionship between these parties. They would thus require supervision from public authorities such as data protection authorities to ensure that they do not violate the rights of private citizens.
Some less benevolent wallet providers might even abuse their position to sell access rights to organizations that are willing to pay a certain price, independent of whether they have a legitimate reason for requesting the data or not. The most malevolent of them might even try to turn this into a business model. This also means that their claim to authority might be questionable, as some wallet providers may be deemed far more trustworthy than others.
This solution can also give some rather powerful organizations like Microsoft, who often have their own wallet applications and might potentially obtain a big share of the SSI market, even more power and could therefore potentially intensify the power unbalance between big tech companies and individual consumers, which the authorization scheme is supposed to avoid.

Having wallet providers act as authorizers for verifiers can work when viewing different SSI systems and wallets in isolation, so when a wallet is not meant to be interoperable and only supposed to have a narrow scope of applications, this approach makes sense. However, it will introduce many problems when considering the larger SSI landscape with lots of different wallets due to the difficulty of making this approach interoperable. Therefore wallet providers, while not completely unsuitable, do not appear to be ideal candidates to take on this responsibility, at least not on a larger scale.

4.1.3 Decentralized Assurance Communities

Both of the previous approaches feature one central party that grants authorizations. However it is also possible to handle verifier authorizations decentrally through so-called assurance communities.

An assurance community, as defined by Joosten et al. (2021), is a community of parties with some common objectives that work together to form a consent on a specific set of rules for their stated scope of applicability. These communities can then provide products and services to help organizations adopt SSI in their business processes, e.g. specifying the structure and semantics of certain credential types, accrediting trusted parties (e.g. issuers/verifiers) or offering decision-tree support through providing arguments for specific decisions and validation policies for data [22].
An example of an assurance community given by Joosten et al. would be the Dutch "Outbreak Management Team" (OMT). The OMT is a standing organization consisting of representatives from (among others) the National Institute for Public Health and the Environment, research institutions and medical societies, which only convenes in cases of serious outbreaks of in-

fectious diseases such as COVID-19 to provide advice and guidance to the government on how to deal with the situation. They could be a SSI Assurance Community to specify necessary SSI components such as vaccination or testing credentials [22].
Another example of a potential assurance community can be found in the notary use case from section 2.2.2. As mentioned in that section, in the Netherlands a person can be authorized to access the bank account of someone else through a notarial act (e.g. if the account owner is in a coma and has authorized the other person to access their bank account through a living will). Banks and notaries are both stakeholders with a common goal in this business process (namely that authorized parties can access the bank account) and might form an assurance community together, which could consist of representatives of the large banks and Royal Dutch Association of Civil-law Notaries (KNB). Within this assurance community, they could form agreements on the ground rules for interactions in which banks have to verify authorizations granted through notarial acts within the Netherlands (e.g. which type of notarial act provides which authorization) and make relevant agreements to facilitate the implementation of SSI into this business process.
These types of assurance communities could also agree on which parties need which data in which contexts and act as authorizing parties based on these types of agreements. For this purpose, assurance communities could form for specific sectors (e.g. financial, online retail, travel, etc.) and/or specific business tasks. These assurance communities could consist of relevant stakeholders such as issuers and verifiers but also representatives of private citizens (e.g. through consumer associations such as the "Verbraucherzentralen" in Germany) to address the interests and concerns of all parties involved in the relevant business process(es). These assurance communities could then form agreements regarding relevant business processes and grant authorizations to verifiers based on these agreements (e.g. through a certificate signed with a private key associated with a specific assurance community). One party might even be part of multiple different assurance communities depending on their business processes.

Having assurance communities act as authorizing parties for specific sectors or business processes offers some benefits. First of all, since each assurance community would have a specified scope consisting of a defined set of business processes, the workload for each of these communities should be manageable, as they only need to grant authorization to a limited set of verifiers within their sector as opposed to more central parties like DPAs who would have to deal with all kinds of sectors that may want to use SSI. Hence, scalability would not a be a very large issue with this approach.
Furthermore, parties included in these assurance communities generally know the relevant business processes and contexts in their scope/sector very well,

so they should have a good overview of which data is strictly needed in which business process, which can lead to a quicker and well-informed design process for decision rules. It also democratizes the process of formulating relevant decision rules for specific contexts, since all relevant parties get to voice their interests and concerns (such as perceived risks) and participate in the formulation of and the agreement process regarding these decision rules. As long as all parties have an equal say within these communities and all parties have to agree to a proposed decision rule for it to be accepted, this approach has the potential to create a fair balance between the interests of all parties.

Just like all other approaches, this approach also has some drawbacks. First of all, having authorizations be granted decentrally through different assurance communities would require all of these assurance communities to use a common standard for as to how authorizations are granted on a technical level (e.g. a common standard format for authorization certificates). Otherwise, interoperability might become an issue as wallets need to be able to verify these authorizations which is much easier if there is one common standard. Getting all potential assurance communities to agree to use a common standard is not a trivial task[2].
A bigger concern however would be potential conflicts of interests and the formation of cartel-esque structures. Assurance communities are often made up of specific private organizations with specific business interests and might not always adequately involve representatives of private citizens. There is a possibility for these communities to make agreements that only serve to benefit specific business interests without addressing privacy concerns adequately and in the worst cases even just serve to legitimize over-asking requests, making the problem even worse. There is thus a significant risk of self-regulation, which is usually not in the interest of consumers. Assurance communities would thus require some sort of supervision by independent authorities (e.g. DPAs) to ensure they do not violate the privacy and data protection rights of citizens and do not just serve their own business interests. This concern is somewhat balanced by the fact that wallets need to actually recognize these assurance communities as "legitimate authorizers".

4.2 Requirements

Besides picking a suitable party that would be responsible for granting authorizations to verifiers, there are additional requirements that need to be met such that a verifier authorization scheme can actually help mitigate

[2]Somewhat ironically, this could be handled through a specific assurance community consisting of representatives of different assurance communities to ensure the adoption of common data standards.

the problem of over-asking verifiers in practice. While specific requirements depend very much on how a verifier authorization scheme is implemented in practice (which is a governance/legal issue and thus out of scope for this thesis), it is still possible to define some more generic requirements that apply to all possible implementations.
To determine what these requirements might be, we will use an approach based on Value Sensitive Design (see [14]). This involves identifying all relevant stakeholders and the values that are of central interest to them when it comes to a verifier authorization scheme. From these values we can derive relevant norms associated with these values from which in turn concrete requirements can be derived.
Friedman et al. (2013) [14] distinguish between two different types of stakeholders:

- **Direct Stakeholders:** These stakeholders are directly involved in the process of granting authorizations to verifiers. The direct stakeholders involved in this case are authorizing parties and the verifiers themselves.

- **Indirect Stakeholders:** These stakeholders are impacted by the process, but do not participate in it directly [14]. In this case, consumers/citizens are indirect stakeholders, since the authorization influences which data they may be asked to provide by verifiers. Issuers may also be indirect stakeholders since they might have certain preferences over how their issued credentials may be used (e.g. an employer might issue credentials to employees that are only intended to be used internally)[3]

Each of these stakeholders has specific values that they would want to be included in the design of a verifier authorization scheme based on how they will be affected by it. By examining these values for each relevant stakeholder, more specific requirements can be derived.

4.2.1 Authorizing Parties

Authorizing parties need to make decisions on when verifiers may ask for specific data and when they may not. There are a lot of different specific contexts for which data sets may be needed and a lot of different verifiers who may want "access rights" to specific data sets in some contexts. The biggest concern for authorizing parties is thus ***scalability*** such that the work load of the authorizer role stays manageable. This is also in the interest of verifiers, since otherwise it could take a very long time until they actually receive the authorization.

[3]How much influence issuers should have here if they are not part of a specific assurance community is somewhat questionable, since with SSI (at least in theory) users should decide themselves how they use their credentials, not the issuers.

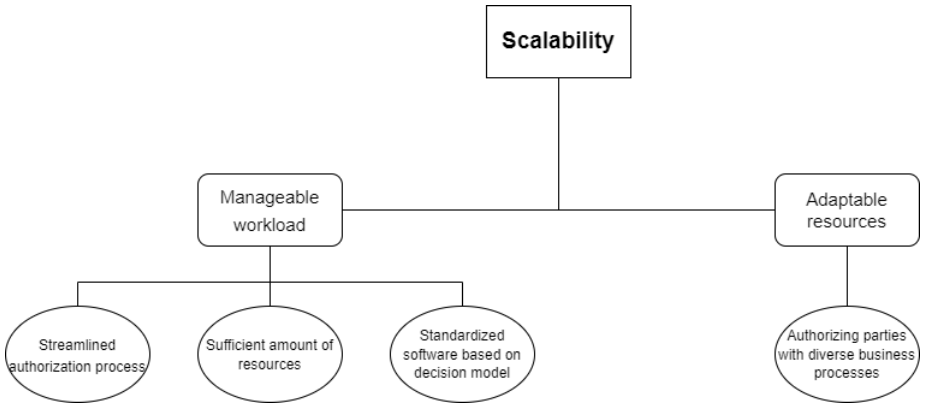

Figure 4.1: Value pyramid for ***scalability***

There are certain norms that define what scalability precisely means and from which more concrete requirements can be derived. These norms are:

- The workload generated from the verifier authorization scheme should be manageable. Even when there is a large amount of requests by verifiers to get a specific authorization it should still be possible to process these requests in a reasonable amount of time. This requires the following:
  - The process needs to be streamlined to reduce the time it takes for verifiers to get an authorization (or get denied an authorization). This includes having a standardized and well-defined (and, if possible, automated) process and a well-defined decision model.
  - Authorizing parties need to have enough resources and employees to be able to handle a large number of requests. The concrete number of people and resources needed depends on how the process of verifier authorization is organized in practice.
  - Standardized and easy-to-use software that authorizing parties can use to generate authorizations (e.g. in the form of certificates) with clearly defined decision models (see also section 5) needs to be readily available. Authorizing parties should be also able to add, modify or delete rules from decision models in the software.
- The amount of resources and personnel dedicated to authorizing verifiers by authorizing parties should be easily and seamlessly adaptable to the actual workload and amount of authorization requests. The following requirements can be derived from this:
  - Authorizing parties should ideally not be solely focused on verifier authorization such that when the amount of authorization requests is low or the process is mostly automated, resources and manpower can be devoted to other tasks.

4.2.2 Verifiers

Just like authorizing parties, verifiers are also involved as direct stakeholders as they need to request authorizations from authorizing parties and possibly go through some sort of bureaucratic process. Just like other stakeholders, they have their own values and interests in regards to verifier authorization due to how it would influence them.
One important value for verifiers is ***fairness***, in both the sense of freedom from bias (i.e. decision rules apply to all verifiers equally) and that underlying decision rules also address perceived risks in a fair manner.
Interoperability is also very important for verifiers to minimize the workload generated by having to obtain authorizations for SSI systems.
Finally, ***transparency*** is an important value for verifiers since they can adapt their business process according to what they are allowed to collect and process. Transparency is also important for consumers. While not all consumers might care, it is still important in democratic systems that consumers can get an insight into which data may be collected in which context and possibly also why that is the case.

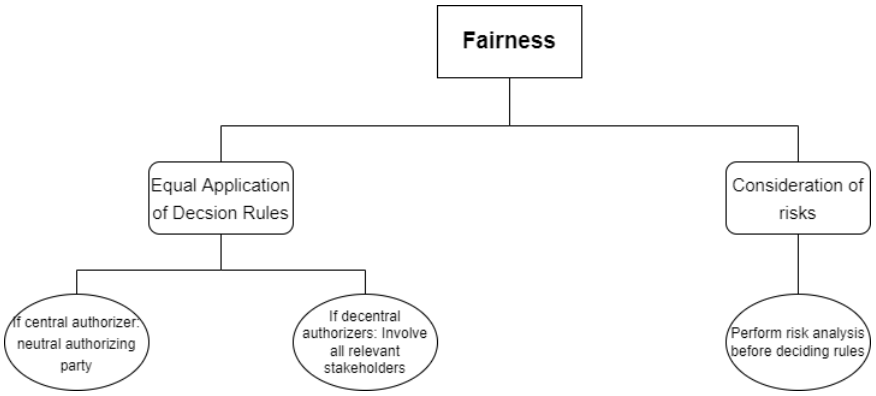

Figure 4.2: Value pyramid for ***fairness***

The following norms are associated with fairness (some of them have already been stated):

- Decision rules for a specific context apply equally to all verifiers.
  - If authorizing party is one central authority: Authorizing party should be completely neutral and not have any business relationships with or business interests in specific verifiers.
  - If authorization is handled in a decentral manner (e.g. through assurance communities): All relevant direct and indirect stakeholders in a specific business process, including relevant verifiers, should be involved in the process of defining rules and agreements to adequately address their interests and concerns.

- Perceived risks are adequately considered within defined decision rules.
  - A risk analysis has been performed before decision rules are defined to assess which risks warrant the processing of additional data (and which specific data is actually needed to mitigate a risk and proportional in relation to the sensitivity of the data and the severity of potential harms due to the risk).

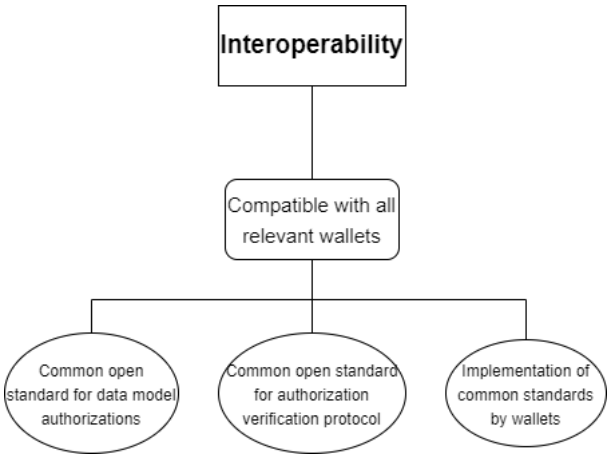

Figure 4.3: Value pyramid for ***interoperability***

The following norms are associated with interoperability:

- Authorizations should work with all relevant digital wallets.
  - A common open standard is needed for the data model of authorization (e.g. certificate).
  - A common open standard is also needed for the protocol in which wallets verify authorizations.
  - Wallets need to support and implement common standards.

The following norms are associated with transparency:

- Decision rules and their design process need to be well-documented.
  - Documentation needs to have a clear structure such that it is easy to find specific information.
  - Ideally, the documentation should clearly state the reasoning behind certain rules and feature definitions of different contexts.
  - Documentation should be written an easy-to-understand and plain language wherever possible (technical terminology probably cannot be completely avoided).

- Documentation should be publicly available on the internet.
  - It should be easy to find the documentation, i.e. it should not be hidden.
  - The documentation should not be redacted/censored in any way
- The process of obtaining authorization should be clear.
  - Clear instructions should be freely available online easy to find.
  - Any software used in this process should be user-friendly and provide clear instructions along the way.

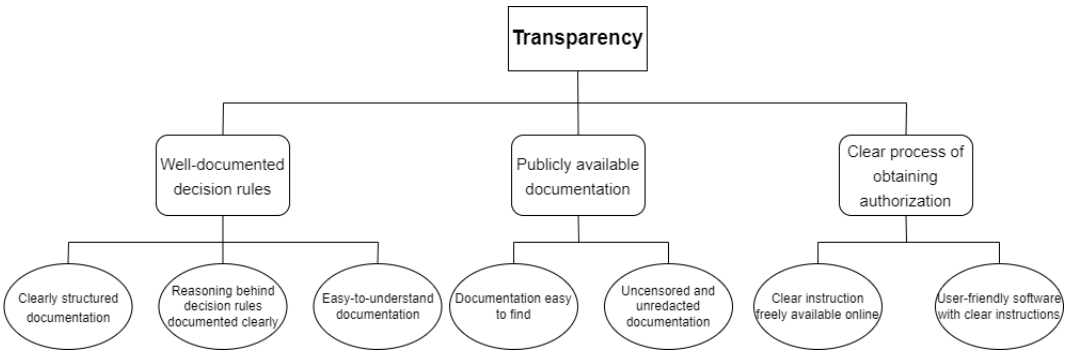

Figure 4.4: Value pyramid for ***transparency***

4.2.3 Consumers/Citizens

Whereas they are not directly involved in the verifier authorization process (mostly), consumers and private citizens in general are still crucial indirect stakeholders. An authorization scheme would determine which data they may be asked to provide in certain contexts and often have to provide. Therefore their values and interest play an important role.
The main value that consumers/citizens bring to the table is ***privacy***, since the whole purpose of authorizing verifiers is the protection of the privacy of individuals against coercion by over-asking verifiers. Of course they have also other values such as transparency (at least some individuals, probably not the average consumer), but these have already been accounted for (see above). The value of privacy brings with it the following norms:

- Whenever verifiers should be authorized to request data, that is not strictly needed for the performance of a transaction, the additional data collected needs to be proportional in relation to the sensitivity of the data and the severity of potential harms due to the risk.

- Risks need to be evaluated by authorizing parties on how big its likelihood is and how large potential damages would be to determine if mitigating this risk actually warrants the requesting of additional data. The more sensitive the additional data, the bigger the risk must be in terms of likelihood and potential damages.
- When in doubt, the privacy of individuals should always take priority over the interests of verifiers due to the power imbalance between them.

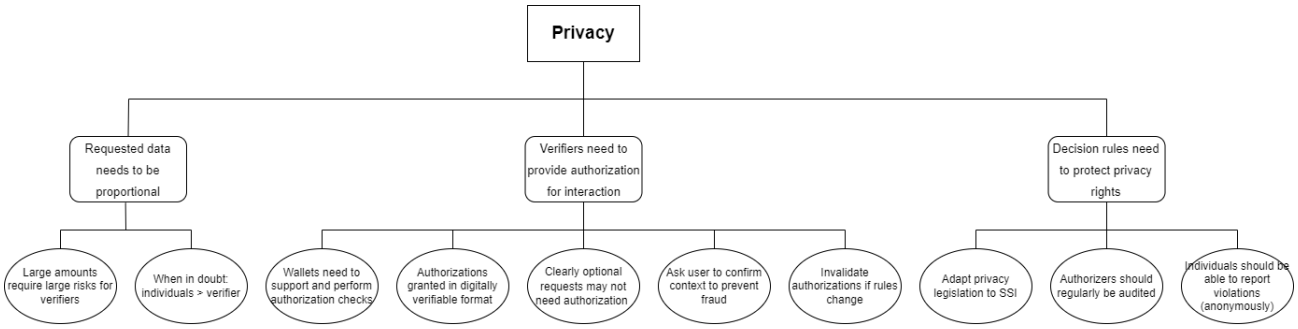

Figure 4.5: Value pyramid for ***privacy***

- Whenever verifiers require individuals to disclose data to perform a transaction/service, verifiers need to provide a corresponding authorization for this transaction.
  - Wallets need to support authorization checks. Ideally, they should legally be obliged to to actually enforce the authorization of verifiers.
  - Authorizations need to be granted in a digital format that is verifiable by wallets (e.g. as a digital certificate or credential).
  - If data requests are completely optional and thus not required, verifiers may not be required to provide authorization. However they should make it absolutely clear, that the data request is indeed optional and can thus be denied by users without the verifier in turn denying service to them.
  - As a fraud prevention measure, users could be asked by their wallet software to confirm that the stated context by the verifier is indeed correct or users could select a context from a drop down menu such that the wallet then asks the verifier to provide authorization for that context. In this case contexts need to have human-readable descriptions that are concise and easy to understand for regular citizens.

  - If necessary (e.g. as punishment for continued violation), it should be possible for authorizations to be invalidated (be it through expiration dates that require verifiers to (automatically) update their authorizations or other means).

- Underlying decision rules should adequately protect individuals' legal rights in regards to privacy.
  - Data protection legislation might need to be updated to accurately reflect the reality of SSI environments, especially when it comes to informed consent and protection against take-it-or-leave-it data requests based on consent (i.e. coercion under (implicit) threat of denial of service).
  - If authorization is handled by parties from the private sector, they should regularly be audited by independent authorities such as data protection authorities to ensure their policies are in line with the legal data protection framework.
  - Individuals need to be able to report possible violations to data protection authorities. Wallets should therefore keep track of past transactions and possibly facilitate reporting. Ideally, they should be able to report such violations anonymously, since the offender might otherwise try to abuse their power to "exact revenge" (e.g. if an employee reports their employer, the employer might try to punish the employee in some way).

Chapter 5

Decision rules

As mentioned in the previous chapter, formal, machine-readable decision rules are needed that ultimately determine which attributes/claims a verifier can access in a specific context[1]. For this purpose, a syntax and semantics for such decision rules need to be formally defined as well as a formal model for contexts/business processes. These decision rules should be defined in such a manner that they are machine-readable such that the decision process of granting authorization for specific business processes/context can be automated. Having consistent decision rules also ensures that the authorizations granted for a specific context remain consistent in which attributes they allow to be accessed. We can then use the use cases defined in section 2.2 to illustrate these decision rules. These use cases also give an idea of different types of over-identification that need to be modelled and addressed by these decision rules.

5.1 Application

First of all, it needs to be addressed who is supposed to apply these decision rules. It is clear that the ones that define these decision rules are the responsible authorizing parties. However, as for their application there are two approaches that can be taken here.

1. **Static access rights:** Authorizing parties apply these decision rules themselves to generate specific authorization certificates that precisely state which attributes the verifier holding the authorization is allowed to request in the context stated in the certificate and users just need know and verify the context as well as verify that the verifier did not include any additional attribute in their request that was not enumerated.

[1] While these rules are ultimately meant to be executed and verified by a machine, it might be handy to have official documentation with human-readable explanations by the authorizing parties.

This works fine for contexts where it can be reasonable determined beforehand which data is needed such as in the notary use case from section 2.2.2 or the Albert Heijn use case from section 2.2.1 and makes it also rather straightforward to verify for wallets if the verifier only requested attributes they are allowed to request. However, cases with a longer chains of dependency such as in the example stated by Anciaux et al. (2013) [4] (see also sections 2.2.3 and 3.2) or the form in step 2 of the mortgage use case where client data is transferred from the mortgage advisor to the lender (see section 2.2.4) are much more difficult to model this way, as the data needed by the verifier heavily depends on the individual holder's situation, hence a simple enumeration of attributes does not suffice for these use cases.
It is also not really possible for users to express personal preference this way. Instead, they have to rely on authorizing parties to choose the least sensitive attribute in cases where one of multiple attributes is needed (but not all of them and not through a dependency chain). Due to this lack of flexibility, this approach is probably not the most suitable to address all relevant kinds of over-identification, though it would probably work fine for simpler contexts where dependency chains and personal preference do not play a role.

2. **Dynamic access rights:** Authorizing parties grant generic authorization certificates for a specific context. These certificates should include a formal context ID and a human-readable description as a string, such that users can actually verify the context. Most importantly however, the certificate should either include the decision model for this particular context directly or include a link to an external location with the decision model that is then applied by wallets to determine which attributes may be requested by the verifier (see also section 6.3).
In this case the verifier would need to present their authorization first (without sending an actual data request) and the wallet would then need to inform the verifier about their access rights based on the decision model that was included. Verifiers could then generate a "formal" data request based on this information and wallets would then (just like in the first approach) verify that this request does not contain any additional attributes.
The advantage of this approach is that it is much easier to model chains of dependency this way as the wallet can just check which attributes are needed in the individual situation of their holder (just as in the minimum disclosure system by Anciaux et al. (2013) [4]). This approach could also allow for users to express personal preference in situations where the rules of the context allow for it (e.g. if the rules say the verifier may request either an email address or a phone num-

ber).
The only real drawback of this approach is that it requires a lot more initiative from wallets and wallet providers that need to actively determine the access permissions of the verifiers themselves, which of course requires more implementation effort on the wallet side. Depending on one's perspective, this can also be seen as an advantage as it gives more control to the user side of SSI interactions, which is not insignificant for SSI. Therefore, this seems to be be the more suitable approach in regards to the goals of this thesis.

5.2 Access control logic

5.2.1 Literature

In the scientific literature, many logic-based approaches to access control have been proposed over the years, most of them utilizing a form of predicate logic [42].

A very simple form of logic-based access control was proposed by Abadi (2003) [1], where he translated a classic access control matrix that grants a specific user read/write/execute privileges to a certain file or directory into predicate logic rules. The most simple predicate symbol he introduces is a simple tertiary predicate symbol `may-access(p,o,r)`, which grants the principal `p` (i.e. a user or role) the right `r` (i.e. read/write/execute) to object `o` (usually a file or directory). For example, the rule `may-access(eisenlohrmj,master_thesis.pdf,Wr)` would give the user `eisenlohrmj` the right to write to the file `master_thesis.pdf`. It is also possible to construct rules such as `may-access(p,o,Wr)` $\Longrightarrow$ `may-access(p,o,Rd)` to express that the write privilege is stronger than the read privilege [1]. Abadi also introduces the predicate `p says s` which states that principal `p` makes statement `s`. This could express a principal delegating some of their authority to another principal or express part of a security policy [1]. To express that a principal `p` delegates access right `r` to principal `q`, one can write `p says may-access(q,o,r)`, with the rule `p says may-access(q,o,r)` $\Longrightarrow$ (`may-access(p,o,r)` $\Longrightarrow$ `may-access(q,o,r)`) expressing `p` is allowed to delegate a right on an object to another principal if `p` holds that right themselves [1].

Another more complex logic-based access control approach was proposed by Bertino et al. (2000) [8]. Just like the approach by Abadi, their model relies on three basic components in the form of subjects (users, groups or roles), objects (the resources that need to be protected) and privileges (the specific access rights, e.g. in the form of read/write/execute, that a subject is granted on a specific object), all of which can be arranged in a hierarchical

manner.
Their model allows the expression of positive and negative authorizations denoted as (s, o, p, g) and $\neg(s, o, p, g)$ respectively, with s representing a subject that is granted privilege p on object o by a granter g, with g being either a specific user or a role. For example, the authorization (`TNO-SSI,SSI_Docs,read,Matti`) states that the user `Matti` granted all subjects that have the role `TNO-SSI` read permissions on the object `SSI_Docs` [8]. In their model, negative authorizations always overwrite positive ones if there is a conflict. The model also allows the derivation of additional rules from existing ones depending on the hierarchies of subjects, objects or privileges or simply from the presence or absence of a specific authorization (e.g. that a set of users must always have the same access rights on a specific object or that a specific user may not exercise a right on an object if another group has a specific privilege on that object) [8]. The model also allows for administrative authorizations via the privileges `own` and `administrate`, which give a subject the authority to grant privileges on a specific object to other subjects.

⟨authorization_program⟩	::=	⟨sequence_of_program_rules⟩
⟨sequence_of_program_rules⟩	::=	⟨ϵ⟩ | ⟨program_rule⟩ | ⟨sequence_of_program_rules⟩
⟨program_rule⟩	::=	(⟨object_identifier⟩.⟨simple_head_clause⟩)
⟨simple_head_clause⟩	::=	⟨head_simple_literal⟩ ← ⟨body⟩
⟨body⟩	::=	⟨ϵ⟩ | [*not*] ⟨literal⟩ {, [*not*] ⟨literal⟩ }
⟨literal⟩	::=	⟨simple_literal⟩ | ⟨referential_literal⟩
⟨referential_literal⟩	::=	⟨object_identifier⟩.⟨simple_literal⟩
⟨simple_literal⟩	::=	[¬]⟨atom⟩
⟨atom⟩	::=	⟨built_in_atom⟩ | ⟨user_defined_atom⟩
⟨built_in_atom⟩	::=	*auth*(⟨term⟩, ⟨term⟩, ⟨term⟩)
⟨user_defined_atom⟩	::=	⟨user_defined_predicate_symbol⟩ (⟨list_of_terms⟩)
⟨head_simple_literal⟩	::=	[¬]⟨user_defined_atom⟩ | [¬]⟨head_built_in_atom⟩
⟨head_built_in_atom⟩	::=	*auth*(⟨term⟩, ⟨term⟩, ⟨constant_term⟩)
⟨list_of_terms⟩	::=	⟨term⟩ {, ⟨term⟩}
⟨term⟩	::=	⟨constant⟩ | ⟨variable⟩
⟨constant_term⟩	::=	⟨constant⟩

Figure 5.1: Syntax of the logic language by Bertino et al. (2000) (originally Fig. 5)[8]

They incorporate these rules into a formal logic language (see figure 5.1). A program rule within this logic language is of the form $\langle o, r \rangle$, with o being an object and r representing a set of literals (a literal being a predicate symbol or a predicate symbol in reference to a specific object), with the head literal (i.e. the first literal in r) being a simple literal (i.e. non-referential) [8]. The simplest predicate symbol is *auth* which is of the form (Subject,Privilege,Granter), however the language also allows for other (user-defined) predicates. If the head literal is *auth*, then the rule is called an authorization rule. If in an authorization rule the body of r is empty (e.g. for a rule such as $\langle$`MasterThesis`, $auth$(`Oskar,read,Matti`) $\leftarrow\rangle$), it is an explicit authorization. In our example, the subject `Matti` grants the subject `Oskar` explicit permission to read the object `MasterThesis`. If the body of r is not empty, the authorization rule is called an derivation rule. An example of a derivation

rule would for example be something like $\langle$`MasterThesis`, $auth$(`Oskar,read,Matti`) $\leftarrow$ `SSI-Repository`.$auth$(`Oskar,write,X`)$\rangle$, whereby the subject `Matti` grants the subject `Oskar` permission to read the object `MasterThesis` if they also have write permission on the object `SSI-Repository` [8].

These are by far not the only logic-based approaches to access control, as there are many other papers on the topics such as by Abadi (2008) [2], Wang et al. (2004) [44], Barker et al. (2003) [6], Crampton et al. (2001) [12] and Bowers et al. (2007) [9]. What all of them have in common is that all of these approaches use a some form of logic language based on predicate logic to model classic access control, although the details of their logic languages and predicates differ.

5.2.2 Approach

The literature examined in section 5.2.1 mostly focuses on classic access control, which defines the access rights (such as read/write/execute) of a specific subject on a specific object (e.g. a file or a directory). This is somewhat different from the logic that is needed for verifier authorization in SSI. If we take the same notion of subject, object, privilege as in the literature, subjects would be verifiers, objects would be specific attributes and the privilege would be a simple `read` permission (i.e. they are allowed to request the disclosure of an object). In classic access control logic, the goal of authorizations rules is to determine the specific privileges a subject can enact on an object. For contextual verifier authorization within SSI however, the question is not which privileges verifiers can enact on an object (as that privilege is just simply `request-disclosure` (`read`)), but rather on which objects they are allowed to exercise this privilege.
Therefore, the privilege itself is irrelevant here, as there is only one possible privilege. What the decision rules need to model for verifier authorization is under which conditions a specific object may be requested. It also needs to be able to model situations where users may want to express personal preferences (e.g. in situations where multiple different attributes would be satisfactory for the verifier, but they only really need one of those).

With these requirements and the literature on access control logic in mind, we can define a suitable approach for decision rules to determine the set of attributes a verifier is allowed to request. A decision model for a specific context is essentially a sequence of rules connected via boolean operators. Rules should generally be of the form $IF\langle condition\rangle THEN\langle access\_permission\rangle$, which we formally denote as $\langle condition\rangle \rightarrow \langle access\_permission\rangle$. Both the condition(s) and the access permission(s) should be modeled as a predicate or a chain of predicate. In the case of condition predicates, these predicates need to return a boolean value, whereas predicates to determine access per-

missions should specify that a specific verifier can access a specific set of attributes. Hence, suitable predicates need to be determined for both of these purposes.

5.2.2.1 Condition predicates

Starting with the conditions, it is helpful to first define what kind of conditions need to be included in the model. The most simple type of condition that needs to be modeled is that the user who is asked to disclose data has a certain property. These types of conditions can easily be verified by wallets as they can just check the value of specific attributes that a user holds in their wallet. These types of conditions can be modeled through the following predicates:

- $equals(attr, val)$, with $attr$ being the value of an attribute and val being a value of the same type (which may also be the value of an attribute). This predicate yields `true` if $attr$ and val are equal (and of the same type) and `false` otherwise. For better readability we will denote this predicate as $attr = val$.
- $greaterThan(attr, val)$, with $attr$ being the value of a numeric attribute and val being a numeric value (which may also be the value of an attribute). This predicate yields `true` if $attr$ is greater than val (and both values are numeric) and `false` otherwise. For better readability we will denote this predicate as $attr > val$.
- $lessThan(attr, val)$, with $attr$ being the value of a numeric attribute and val being a numeric value (which may also be the value of an attribute). This predicate yields `true` if $attr$ is less than val (and both values are numeric) and `false` otherwise. For better readability we will denote this predicate as $attr < val$.
- $GreaterThanOrEquals(attr, val)$, representing the disjunction of the $equals$ and $greaterThan$ predicates, with both $attr$ and val having a numeric value. For better readability we will denote this predicate as $attr \geq val$.
- $LessThanOrEquals(attr, val)$, representing the disjunction of the $equals$ and $lessThan$ predicates, with both $attr$ and val having a numeric value. For better readability we will denote this predicate as $attr \leq val$.

However these are not the only types of conditions that would need to be modeled in practice. Another type of condition would be that the verifier, who presented their contextual authorization certificate to the user and wants to request data, has certain properties. For example, in the Netherlands only specific government organizations may process a citizen's BSN (a

unique identifier every Dutch citizen holds). So if a verifier would want to request a BSN within a specific context, this should be tied to the condition that they are one of these government organizations, e.g. through a rule like $IF\ isGovernmentOrganization(v)\ THEN\ mayRequest(v, IDCardCredential.BSN)$, with $isGovernmentOrganization(v)$ being the condition predicate (v representing the requesting verifier) and $mayRequest$ as the access permission predicate (more on those types of predicates later).
For these types of conditions we can imagine a large number of potential predicates. The question here is how these conditions can be verified by wallets, as they do not know the verifier's properties. I see two options as to how this could be approached:

- **External condition verification:** If decision models are stored on an external server hosted by the corresponding authorizing party, this server could also include a database with all relevant information about verifiers that have obtained authorization certificates from that authorizing party that is needed to verify conditions contained in the decision models of contexts the verifier has obtained authorization for. Wallets would then just need to query the server with the predicate and supply the public DID of the requesting verifier. The server would then search their data base and return either TRUE or FALSE to the wallet. Since the server and database are hosted by the authorizing party, the result could be deemed rather trustworthy.

- **Verification through verifier credentials:** Another possibility would be that verifiers (just like regular citizens) hold verifiable credentials holding information about their organization. These credentials must not be self-issued, i.e. the data should be qualified as this would otherwise open the door for deception by malicious verifiers. So if a wallet encounters a condition that depends on a certain property of the verifier, they could just send a data request to the verifier asking for the disclosure of attributes and the verifier returning a verifiable presentation. In that case, similar rule predicates such as the ones to verify properties about a user can be utilized, either through introducing similar predicates specifically for verifier properties or adding another element to the predicates to indicate whether this concerns a property of a user or a verifier.

A last type of conditions relates to past events. There might be contexts in which, for example, first, one verifier requests some information, and depending on the outcome some other verifier is allowed to request some information. Here, one can introduce predicates such as $hasReceived(v, attr)$, which states that verifier v has received attribute(s) $attr$. Actually verifying such condition predicates would require wallets to keep track of past transactions and events, which many wallets already do, so this should be at least

theoretically feasible.
Obviously, the amount of potential condition predicate is quite large. In practice however a standardized set of condition predicate would be needed such that their verification can be properly integrated into wallet applications. Which condition predicates are needed very much depends on the specific contexts that need to be modeled, so there will not be any exhaustive list included here. In a real-world context though, authorizing parties would very much need to define and standardize a specific set of condition predicate to allow for a proper implementation.

5.2.2.2 Access predicates

Finally, predicates that determine the actual access permissions are also needed. The most important predicate was already mentioned earlier, namely $mayRequest(v, o)$. This predicate models that a specific verifier may request a certain set of attributes. The parameter o denotes either an object (i.e. an attribute) or a set of objects and v denotes the subject (i.e. the verifier) that is allowed to access said objects. The subject can either be a specific pre-defined verifier (in which case v would be replaced by a specific identifier) or simply the verifier that the wallet is currently interacting with. In the latter case v would just be symbolically denoted as v in an actual rule to signify that the wallet should insert the "current" verifier (i.e. the party the wallet is interacting with).

5.2.2.3 Formal syntax

Figure 5.2 denotes the full formal syntax for contextual decision models, taking inspirations from other models such as the one by Bertino et al. [8] (see figure 5.1). A decision model is simply a sequence of rules of the form $\langle condition \rangle \rightarrow \langle access - permission \rangle$ (or simply $\langle accessPermission \rangle$ if no conditions apply for a particular rule). In the case that for a specific rule multiple atomic conditions are needed, they can be put in a sequence with the standard boolean operators such as $\wedge$ (logical AND), $\vee$ (logical *inclusive* OR) and $\oplus$ (logical *exclusive* OR).

Whenever rules are connected by a $\vee$ or $\oplus$ operator and the conditions of more than one of these rules apply, the user will have to indicate their preference as to which of these rules should apply. The question here is how users can express these preferences. This can be done in multiple ways:

1. Ask the holder directly after processing a decision rule. This could be a message from the wallet which lists all possible decisions to the holder. For rules connected via $\oplus$ operators such as $r \implies r_1 \oplus r_2 \oplus r_3$ and the conditions of at least two of these rules apply, then the user will have to choose exactly one set of attributes specified by one of those

rules which then gets added to the total set of permitted attributes for that interaction (see also example 5.2.2). For rule connected with $\vee$ operators such as $r \implies r_1 \vee r_2 \vee r_3$ and again the conditions for at least two of these rules apply, they will have to choose at least one (but may also select more) set of attributes specified by those rules. This can be perceived as annoying by some holders (although these holders might find the whole SSI data disclosure flow to be rather annoying anyway), but it does give direct control to holders, which is very much in alignment with the SSI principles of Sovereignty, Privacy and Data Access Control [30].

⟨rule_sequence⟩	::=	null | ⟨rule⟩ | ⟨rule⟩ $\wedge$ ⟨rule_sequence⟩ | ⟨rule⟩ $\vee$ ⟨rule_sequence⟩ | ⟨rule⟩ $\oplus$ ⟨rule_sequence⟩
⟨rule⟩	::=	⟨set_of_conditions⟩ $\rightarrow$ ⟨access_predicate⟩ | ⟨access_predicate⟩
⟨access_predicate⟩	::=	*mayRequest*(⟨verifier⟩, {⟨set_of_attributes⟩})
⟨set_of_attributes⟩	::=	null | ⟨attribute_identifier⟩ | ⟨attribute_identifier⟩, ⟨set_of_attributes⟩
⟨set_of_conditions⟩	::=	True | ⟨condition⟩ | (⟨condition⟩ $\wedge$ ⟨set_of_conditions⟩) | (⟨condition⟩ $\vee$ ⟨set_of_conditions⟩) | (⟨condition⟩ $\oplus$ ⟨set_of_conditions⟩)
⟨condition⟩	::=	simple_condition | condition_predicate
⟨simple_condition⟩	::=	⟨attribute_identifier⟩ = ⟨value⟩ | ⟨num-attribute_identifier⟩ > ⟨numeric_value⟩ | ⟨num-attribute_identifier⟩ < ⟨numeric_value⟩ | ⟨num-attribute_identifier⟩ $\geq$ ⟨numeric_value⟩ | ⟨num-attribute_identifier⟩ $\leq$ ⟨numeric_value⟩
⟨value⟩	::=	⟨attribute_identifier⟩ | ⟨numeral⟩ | ⟨String⟩ | ⟨Boolean⟩ | [...]
⟨numeric_value⟩	::=	⟨num-attribute_identifier⟩ | ⟨numeral⟩
⟨condition_predicate⟩	::=	*e.g. isGovernmentOrganization*(⟨verifier⟩) | [...]

Figure 5.2: Formal syntax for decision rules

2. Use the approach defined by Ardagna et al. (2012) [5] and have users themselves assign sensitivity labels and context restrictions to attributes and credentials, such that wallets can make these choices automatically based on the assigned labels and do not have to ask users every time. This might save time in the long run, but might also

be too complex for the average user.

3. Use a hybrid approach of the first two options and leave users the option to assign these labels and whenever they have not assigned a label, have the wallet ask users about their preference. This gives users more options and flexibility in how they wish to approach personal preference, as some users might very much appreciate having the option to have this process automated, whereas other users might not bother with assigning labels. A possible drawback of this would be that it requires wallet providers to implement a rather complex system such as the one proposed by Ardagna et al. (2012) [5] only for it to be only used by a fraction of their wallet's user base.

In practice, the exact approach taken may differ from wallet to wallet, as some wallet providers might prefer one way to allow user preferences to be expressed while others might prefer a different method.

5.2.2.4 Examples

The following examples will further illustrate how these decision rules are supposed to be evaluated and what they could look like in practice. These examples do not necessarily represent real decision models for the named use cases, but merely serve to illustrate what decision models could look like. Within these examples, attributes are denoted as ⟨`credential_type`⟩.⟨`attribute_name`⟩, which is a functional but not very elegant notation.

Example 5.2.1. First let us take a rather simple example of a kinsman of someone recently deceased that was authorized in the will of the deceased to access their bank account (see also section 2.2.2). In this case, the bank of the deceased may request five attributes:

1. The name of the deceased from their death certificate in order to confirm the death of the account holder.
2. The name of the executor named in the deceased's will
3. The subject of the will (to confirm it is indeed the will of the deceased)
4. The precise authorization of the executor (to confirm the executor indeed has the right to access the bank account).
5. The name of the person standing before them to confirm that that person is indeed authorized to access the bank account.

This can be expressed via the following rule:

$$r \implies r_1$$
$$r_1 = isBank(v) \rightarrow$$
$$mayRequest(v, \{\texttt{DeathCertificateCredential.nameDeceased},$$
$$\texttt{WillCredential.executor.name}, \texttt{WillCredential.executor.authorization},$$
$$\texttt{WillCredential.nameSubject}, \texttt{IDCardCredential.name}\})$$

Here we have one condition predicate, namely $isBank(v)$, which denotes that the rule only applies if the requesting verifier is a bank, and a simple *mayRequest* predicate stating that that same verifier may request the specified attributes.
For many contexts, decision rules will probably be a simple enumeration of the permitted attributes. However not all use cases are this simple so it might be helpful to also examine a more complicated use case.

Example 5.2.2. Anciaux et al. (2013) [4] provide a more complicated example where data is being collected to determine whether a person is eligible to receive financial support for home aid. In this case there is a chain of dependency to actually determine the minimal set of data needed by the relying party, which are modeled by Anciaux et al. through collection rules (recall figure 3.1). If we translate their rule to the syntax we defined, the result looks roughly like this:

$$r \implies r_1 \oplus r_2 \oplus r_3$$
$$r_1 =$$
$$(\texttt{PensionCredential.amount} \leq 10000) \rightarrow$$
$$mayRequest(v, \{\texttt{PensionCredential.amount}\})$$
$$r_2 =$$
$$(\texttt{MedicalFileCredential.amountLostAbilities} \geq 2) \rightarrow$$
$$mayRequest(v, \{\texttt{MedicalFileCredential.amountLostAbilities}\})$$
$$r_3 =$$
$$(\texttt{PensionCredential.amount} < 30000) \wedge (\texttt{IDCardCredential.age} > 80) \rightarrow$$
$$mayRequest(v, \{\texttt{PensionCredential.amount}, \texttt{IDCardCredential.age}\})$$

This set of decision rules is obviously more complicated than in our first example, but it does model the three different minimum disclosure scenarios described by Anciaux et al. (2013) accurately. These three scenarios were:

1. If the applicant had a pension of less than €10.000, it suffices to disclose the amount of their pension to show they are eligible for financial support for home aid.

2. If the applicant had more than two lost abilities, they only need to disclose the amount of their lost abilities.

3. If the applicant had a pension of less than €30.000 and is over eighty years old, they need to disclose both the amount of their pension and their age.

For each of these we need a separate rule with a *mayRequest* predicate, as different conditions determine whether these cases are even relevant for the individual applicant. In this context personal preference also plays a role, e.g. if an applicant has a pension of less than €10.000 and has at least two lost abilities. In that situation, both the first and second scenario would apply, which would result in a set with two elements (namely {`PensionCredential.amount`, `MedicalFileCredential.amountLostAbilities`}), while the applicant only needs to share one of these attributes. Therefore, these three rules have to be connected via $\oplus$ operators and the applicant will have to communicate which one they would prefer to disclose.
This example is a rather simple chain of dependency. In reality, these chains are much more complex, thus decision rules for these contexts would be even longer and more complex in practice.

Chapter 6

Technical Implementation

While the governance aspects of verifier authorization are of utmost importance, authorizations will need to be verified by wallets on a technical level. This requires a data model for verifiable authorizations as well as a secure communication protocol between verifiers (or some software acting on behalf of the verifier) and users (or rather their digital wallet) during which the wallet verifies the validity of the authorization and whether the verifier only asks for attributes they are authorized to for the context of the request.

6.1 Threat model

Before defining the authorization data model as well as the protocol, it is important to first define a concrete threat model that needs to be addressed. This threat model can then be used as a basis to define security requirements for both the authorization data model as well as the verification protocol.

The biggest threat here is obviously that malicious verifiers are somehow still able to ask holders for data that they have not been authorized to process by an officially recognized authority. Verifiers might try to deceive users or trick the system through various means such as:

- Forging a fake authorization that seems legitimate to wallets. (Authorization from illegitimate source)
- Modifying a legitimate authorization in such a way that they may request more data.
- Presenting legitimate authorization that was issued to someone else and not to them or imitating another verifier entirely.
- Using legitimate authorization obtained for a different context to ask for more data than they are supposed to. (Misrepresenting the context)

There is also the threat of an attacker eavesdropping on the communication between the verifier and the wallet and later replaying the messages from the verifier (including the presentation of the authorization) to possibly obtain illegitimate access to a holder's personal data through the imitation of a verifier that is authorized to request that data.

Another threat is an attacker spying on the interactions of a certain user. This attacker might deduct what kind of transactions users are involved in from the context stated by the authorizations presented by verifiers and be able to profile them. This would be a huge violation of a user's privacy.

Note that these are not the only possible threats, but just those that are within the scope of this thesis. One could for example imagine a scenario where a malicious issuer that ask user's to physically hand over their phone and let them browse their wallet, however this scenario is clearly outside of scope as there is no real technical countermeasure to it.

6.2 Security requirements

From the threat model defined in section 6.1, the following security requirements can be derived:

SR1 **Integrity of messages:** It should be easy to detect whether a message has been modified.

SR2 **Integrity of authorizations:** It should be easy to detect whether an authorization has been modified by someone other than the corresponding authorizing party.

SR3 **Non-repudiation of communication:** Verifiers should not be able to repudiate messages they sent to users/wallets. This is important such that users are able to report (and prove) fraudulent behaviour.

SR4 **Authentication of the verifier:** Verifiers need to be authenticated such that the user knows that they are truly the legitimate holder of an authorization, i.e. the wallet needs to know the identifier (DID) of the verifier they are interacting with. Authorizations also need to state the legitimate holder of a specific authorization, such that the wallet can verify whether this matches the identifier of the party they are interacting with or not.

SR5 **Replay protection:** It should be easy to detect whether a message has simply been replayed.

SR6 **Authenticity of authorizations:** It should be easily verifiable for wallets whether an authorization presented to them is legitimate (i.e.

the authorization has been issued by a recognized authorizing party) or not.

SR7 **Verification of context:** Users or their wallets need to verify that the context for which a given authorization is valid is actually the context of the data request. For example, if the certificate states that it is valid for the context of applying for a mortgage, then the user should verify that that is actually the transaction they are engaging in.

SR8 **Confidentiality of authorization certificate:** The information contained within authorization certificates should remain confidential between the verifier that holds the certificate and the users/wallets they have chosen to present a certificate to (and the corresponding authorizing party(/parties)). This is because the certificate reveals the context of the data request which is somewhat privacy-sensitive, so only the verifier holding that certificate and the user they sent it to should be able to know the contents of the certificate.

Some of these security requirements are already covered by existing SSI standards, however it is still important to explicitly name them for the purpose of verifier authorization.

6.3 Data model authorizations

With the security requirements defined in the previous section, we can define a data model for verifiable authorization certificates. These certificates serve as a proof from a verifier to a wallet/holder that they are authorized by a trusted authority to request specific data in the context of the current transaction between these two parties. These certificates need to contain the following data:

(1) In which context the verifier is allowed to request these attributes.

(2) Which attributes the verifier is allowed to request or at least the underlying decision model (or a link to it) of the underlying context.

(3) The holder of the certificate (i.e. which party has been authorized to request this data).

(4) The authority that has issued the certificate.

(5) A human readable description of the context to allow users to verify the context.

(6) A cryptographic signature over all this data by the issuing authority of the certificate via their private key to prove the authenticity and integrity of the certificate.

As for points (3) and (4): In SSI ecosystems, the identities of parties such as wallets (or rather the holder of a wallet), issuers and verifiers are often denoted by so-called Decentralized Identifiers (DIDs). A DID is an identifier that a party generates themselves and share with other parties. This identifier is usually of the form did:⟨DID method⟩:⟨DID method-specific identifier⟩(e.g. did:example:123456789abcdefghi), which via the specified DID method can be resolved to a DID document which contains information about the party identified by the DID. Most importantly, the DID method contains a public key and the key algorithm of that key, which other parties can use to communicate with the subject of the DID. DIDs can also be either public (i.e. they are registered on public infrastructure such as a server or a ledger) or private (i.e. they are only know to two parties) [37]. Therefore, to denote the identity of the holder of an authorization certificate (a verifier) and of the issuer of the certificate (an authorizing party), a public DID of each party could be used.

One open standard that can be used for authorization certificates are Verifiable Credentials (VCs) [17]. Verifiable Credentials are already a widespread standard within SSI systems as a data model for digital credentials issued to individual holders with the purpose of proving some properties about their subject which a trusted "authority" (the issuer) vouches for. An authorization certificate is essentially the same, only that the subject/holder is an organization rather than an individual.
Verifiable Credentials already contain fields for the issuer, the subject (who the claims within the credential apply to) and a cryptographic proof in the form of a signature by the issuer over the data [17], thus data items (3), (4) and (5) are already included by design. An authorization credential could have a credential type such as `AuthorizationCredential`, to distinguish it from other forms of verifiable credentials or verifiable presentation, since they need to be processed/verified in a unique way.
The context of the request (not to be confused with the `@context` property of VCs) could either be a claim or could be included in a credential type unique for that context. The latter would require users to actively request a specific type of authorization (thus users would need to specify the context themselves beforehand (e.g. by selecting it from a drop-down menu)) whereas the former would require users to verify that the value of the `context` claim corresponds to the actual context of the request (e.g. by the wallet asking if the context is indeed correct).
The remaining issue is the incorporation of the underlying decision model. This can be incorporated in one of two (and a half) ways:

1. **Internal Decision Model:** Add it as a claim (e.g. `contextDecisionRule` or `contextDecisionModel`) to the authorization certificate, therefore incorporating it internally. This introduces no additional security re-

quirements to protect the integrity of this decision model and would enable offline verification, however it makes updating a decision model more difficult as it can only really be updated by either revoking all authorization certificates for this context or wait for them to expire to update them one-by-one, both of which are certainly not very ideal. This is not insignificant, as context decision models are not always static and may require updates from time to time (e.g. when there are changes in relevant legislation).

2. **External Decision Model:** Have the decision model be stored at an external location and have the `context` itself either include a URI to this location or have an extra claim for this with a URI to the location as its value. Having this decision model at an external location makes it much easier for authorizing parties to update a decision model, but it would also mean that authorization certificates could not be verified offline. This introduces additional security requirements. these additional security requirements would be:

   SR9 **Integrity of the file:** Independently of where the decision model (or a file containing it respectively) is stored, whether it be a central or decentral server, a cloud, a blockchain or some form of distributed ledger, the integrity of the decision model (file) needs to be guaranteed or at least manipulation should be easily detectable.

   SR10 **Availability of the file:** The availability of the file needs to be guaranteed, since authorization certificates cannot be verified. Therefore choosing a secure location is of utmost importance.

   SR11 **Authenticity of the file:** It needs to be clear that this file was created by the responsible authorizing party. This could be achieved by the data being digitally signed by the authorizing party, which also helps with securing its integrity.

   SR12 **Confidentiality of the file contents:** Since the decision model reveals the transaction/context, the file contents need to be confidential to prevent the profiling and tracking of holders by an external attacker such as an eavesdropper. If communication is done via an existing protocol such as TLS, this requires no additional actions as communication is already encrypted.[1]

   SR13 **Secure communication channel with server/cloud/ledger:** The communication channel between a holder's wallet and the data structure where the decision model is stored needs to be

[1] While slightly out of scope, side channel attacks might still be a concern (e.g. the length of a file revealing its identity)

secure in terms of **integrity** (such that messages are not manipulated), **availability** (the location needs to be available at all times to be reached by wallets), **authenticity** (the wallet needs to know that the decision came from the right server as included in the authorization certificate) and **confidentiality** (if a server/-cloud/etc. has multiple decision models stored (which will most likely be the case), an external party should not be able to see which file(s) are accessed by a specific wallet).

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.overidentification-protection-authority.gov/2022/authorizations/v1"
  ],

  "id": "http://overidentification-protection-authority.gov/certificates/1337",

  "type": ["VerifiableCredential", "VerifierAuthorizationCredential"],

  "issuer": "https://overidentification-protection-authority.gov/issuers/42042",

  "issuanceDate": "2022-07-06T14:23:24Z",

  "credentialSubject": {

    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",

    "context": {
      "contextID": "910c109b-2e2c-46e4-96f6-c98dc7aeabb4",
      "decisionModel": "https://overidentification-protection-authority.gov/context-models/910c109b-2e2c-46e4-96f6-c98dc7aeabb4",
      "description": "Check if whether the bank account holder has passed away and the holder has
                      been authorized to access the account in the will"
    }
  },


  "proof": {

    "type": "RsaSignature2018",

    "created": "2017-06-18T21:19:10Z",

    "proofPurpose": "assertionMethod",

    "verificationMethod": "https://example.edu/issuers/565049#key-1",

    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5X
      sITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-pQy7UJiN5mgRxD-WUc
      X16dUEMGlv50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-kronKb78cPN25DGlcTwLtj
      PAYuNzVBAh4vGHSrQyHUdBBPM"
  }
}
```

Figure 6.1: Simplified example of an authorization certificate credential (modified example 1 from VC specification [17])

Figure 6.1 provides an example of an authorization certificate in the form of a verifiable credential, where the `context` claim consists of three sub-claims, namely a formal ID of the context, an URI linking to the (external) decision model associated with that context and a human-readable description of the context. This human-readable description is meant to be presented

to the holder such that they can verify that the context fits the intended transaction. For this they need a human-readable description, since most holders probably would not know what the formal context names mean exactly.

6.4 Protocol

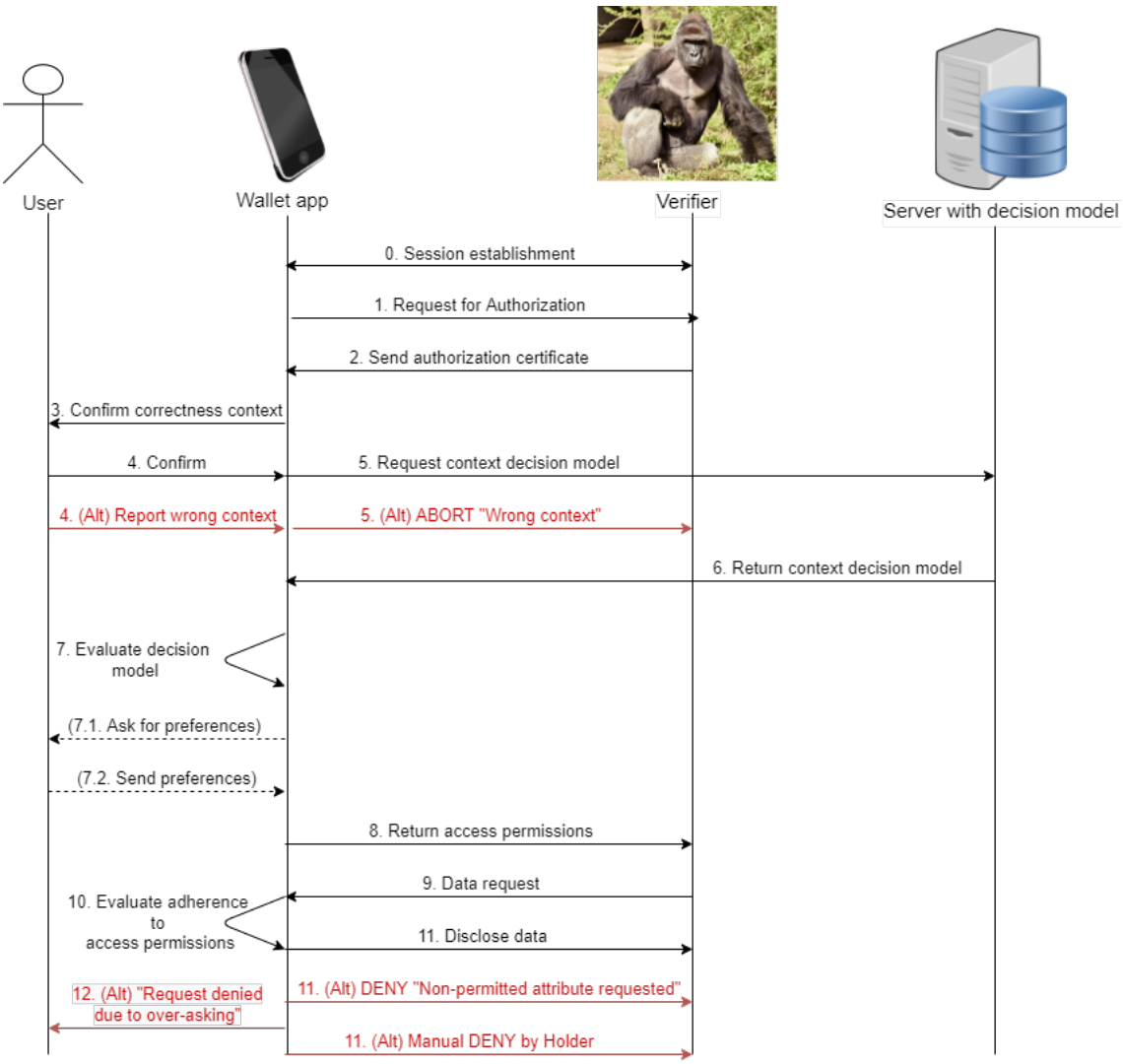

Figure 6.2: Basic flow of the authorization verification protocol

After having defined the underlying data model for contextual authorization certificates, a protocol for the verification of a contextual authorization can

be sketched[2]. Figure 6.2 gives a more detailed overview of the general flow of this communication. This flow works under two assumptions, namely that a) contextual decision models are stored on an external server and are thus not internally included in the certificate itself and b) personal preferences of holders are expressed by their wallet asking them directly whenever they have a choice.

This protocol is meant to be an extension of existing presentation exchange protocols such as the DIDComm Present Proof protocol [24] or OpenID for Verifiable Presentations [40]. As such it is intentionally kept somewhat vague to allow for it to be integrated into these existing protocols. Steps 9 and 11 respectively model messages that are already part of these protocol standards, with step 9 being roughly equivalent to the "Request Presentation" message of the DIDComm protocol and the "Self-Issued OpenID Provider Authentication Request" of the OIDC protocol and step 11 being roughly equivalent to the "Presentation" message of the DIDComm protocol and the "Self-Issued OpenID Provider Authentication Response" of the OIDC protocol.
The proposed protocol extends these protocols by introducing additional messages and checks before the start of the "actual" credential sharing protocol to determine the contextual access permissions of the verifier as well as adding a check in between the data request and the response to verify whether the data requested only contains attributes within said set of access permission. To guarantee the integrity, confidentiality and authenticity of messages, all messages are encrypted with the other party's public key and contain a signed hash of the message signed with the sender's private key. The full extended protocol consists of the following steps:

0. **Session Establishment**
   As a pre-requisite it is assumed that the wallet and the verifier have already established a session (which involves mutual authentication) (e.g. through DIDComm Relationship Establishment [45] or an OIDC SIOP Metadata Exchange as described in [46]). This means that they are aware of each other's identifiers (for this session in the form of private DIDs) and have set up a secure communication channel through an exchange of public keys. All subsequent messages are thus encrypted with the other party's public key to ensure the confidentiality of messages.

1. **Request for Authorization Certificate** The user's wallet initiates the protocol by sending a request for authorization together with a challenge (e.g. a nonce). The purpose of this message is to prevent other parties from replaying messages. This also aligns with existing

[2]The basic flow of this communication was already mentioned in chapter 5.

SSI protocol standards such as DIDComm or OIDC as messages in these protocols are always exchanged in a request-response manner.

2. **Presenting Authorization Certificate**
   The verifier replies by presenting an authorization certificate credential that fits the context of their transaction to the wallet. The wallet then should automatically verify, that the issuer of this certificate is indeed a trusted authorizing party and should verify that the subject of the certificate is indeed the party that presented the certificate to them (by verifying the public DID of the subject of the certificate) and that the signature of the certificate checks out. It should also verify that the context ID matches the ID stated in the link to the decision model. This message should at least contain the authorization certificate as well as the challenge the wallet send in the previous message.

3. **Request for Context Verification**
   The wallet displays a message to the holder asking them to confirm that the context stated by the authorization certificate sent by the verifier is indeed the context of the transaction at hand. This message includes a human-readable description of the context (in the form of the String value of the `"context.description"` claim from the certificate) such that they can more accurately determine if the context is correct as formal context names will probably not be helpful for holders.
   This message could roughly follow this template:

   > <Verifier> claims that the purpose of this transaction is the following:
   > <`"context.description"` claim>
   > Is this accurate?

   The wallet then displays two buttons, one to confirm the accuracy of the context and one to refute this claim.

4. **Result Context Verification**
   The user either confirms the correctness of the stated context or refutes it by pressing one of the buttons. If the user refutes the correctness of the context here, the wallet will send an abort message to the verifier with an error code informing the verifier that the process has been aborted since the context stated by the authorization certificate they presented is not actually the context of the transaction. In case the verifier did not state the correct context, this event should be logged to enable users to report fraudulent verifiers.

5. **Request Context Decision Model from Server** If the holder has confirmed the accuracy of the stated context, the wallet sends a re-

quest to the address stated by the `"context.decisionModel"` claim included in the authorization certificate to obtain the underlying decision model for that context. As mentioned in section 6.3 the communication between the server and the wallet needs to be secure in terms of integrity, availability, authenticity and confidentiality. Web communication protocols such as TLS provide sufficient security here, so there is (probably) no need for any extra measures.

6. **Server Reply**
   The server sends back the requested decision model to the wallet.

7. **Evaluation Decision Model**
   The wallet evaluates the decision model to determine the (maximum) set of data the verifier is allowed to request in the context of the transaction as described in chapter 5. If after the evaluation of multiple rules connected via $\vee$ or $\oplus$ operators there are multiple options, the wallet asks the holder to pick their personal preferences for each choice as described in chapter 5.

8. **Inform Verifier about Access Rights**
   The wallet sends a message to the verifier which contains the (maximum) set of credentials they are authorized to request during this transaction. This message should also include a challenge in the form of a nonce for replay protection.

9. **Data Request**
   The verifier sends their data presentation request to the wallet. They should also send back the nonce to respond to the wallet's challenge and send a new challenge in the form of a nonce to the wallet.

10. **Evaluation Legitimacy Data Request**
    The wallet evaluates if the data request only includes claims the verifier is authorized to request. If the request contains an unauthorized attribute, the wallet automatically denies the request and sends back an error code to the verifier and also informs the holder why the request was denied.

11. **Reply to Data Request**
    If the wallet confirmed that only permitted attributes were requested, the holder can choose to disclose the requested attributes or deny the request[3]. If the holder wishes to disclose this data, the wallet sends back a verifiable presentation, otherwise they send back an error message stating that the holder manually denied the request. The response message also contains the challenge sent by the verifier.

[3] In that case, the verifier was acting in a legitimate manner, so they cannot be punished for their request and may abort the transaction.

6.5 Demo

To give an idea of what the application of this modified presentation exchange protocol could look like in practice, I build a demo using python that simulates the communication between a wallet and a verifier and shows the user experience. This demo can be found at https://github.com/MaJoEi/scriptie-demo. It should be noted that this is **NOT** a full production-ready implementation of the protocol, as many aspects (such as among others the verification of credential signatures, the exchange of keys and identifiers as well as the processing of certain message items) are mocked.

This demo models the protocol as a simple client-server interaction over a socket, with the wallet acting as a client and the verifier acting as a server. All messages sent between the wallet and the verifier are encrypted using RSA and include an RSA-signed SHA256 hash of the message contents. The messages themselves are modelled after the messages from the OpenID for Verifiable Presentations standard in JSON format [40].

```
{
    'response_type': 'vp_token',
    'client_id': 'https://client.example.org/',
    'redirect_uri': 'https://client.example.org/',
    'presentation_definition': {
        'id': 'Request for contextual authorization',
        'input_descriptors': [
            {
                'id': 'Verifier Authorization Credential',
                'format': {
                    'ldp_vc': {
                        'proof_type': [
                            'RsaSignature2018'
                        ]
                    }
                },
                'constraints': {
                    'fields': [
                        {
                            'path': [
                                '$.type'
                            ],
                            'filter': {
                                'type': 'string',
                                'pattern': 'VerifierAuthorizationCredential'
                            }
                        }
                    ]
                }
            }
        ]
    },
    'nonce': 'd30af247f61044d88e4a0f6e703ffba2'
}
```

Figure 6.3: Example of the request for an authorization certificate (message 1)

The first message in which the wallet requests the verifier to provide

an authorization certificate is modelled after the OpenID Authorization Request, with the role of sender and recipient being reversed and the `presentation_definition` parameter specifying that the wallet is requesting a credential of the type `VerifierAuthorizationCredential` (see also figure 6.3 for an example).


```
{
    'client_id': 'https://client.example.org/post',
    'redirect_uris': ['https://client.example.org/post'],
    'response_type': 'vp_token',
    'response_mode': 'post',
    'presentation_submission': {
        'id': 'Verifier Authorization Credential example presentation', '
        definition_id': 'Verifier Authorization Credential example',
        'descriptor_map': [
            {
                'id': 'Verifier Authorization Credential',
                'format': 'ldp_vc',
                'path': '$'
            }
        ]
    },
    'vp_token': {
        '@context': [
            'https://www.w3.org/2018/credentials/v1',
            'https://www.overidentification-protection-authority.gov/2022/authorizations/v1'
        ],
        'id': 'http://overidentification-protection-authority.gov/certificates/02ee6ba183fc485a848202b5d208b790',
        'type': ['VerifiableCredential', 'AuthorizationCredential'],
        'issuer': 'did:key:zTL5kCGeFa3dV3BF3wu2TgrN8fBAM6RmcQDfmbJy89RppdpW2uKo3LQV5LC3mFKuQS2Z6f9q37xNJhnFkSGvVn7wdFAyMUEWnmYNkgQ94fXVGACQJzfMrYhuE3CrVe
        'issuanceDate': '2022-07-06T14:23:24Z',
        'credentialSubject': {
            'id': 'did:key:zTL5kCGeFa3dV3BGe1wE8iTvLuPA2G3QitEyV7WatH7BAj5rgRft6tfRnUUYLJQyhjWh9hd1m286HEuBk1p3fi7fzLbTDW6Q7oP6KiqxT3efDRxbpmoDSUUHTdJU8c
            'context': {
                'contextID': '081863fe03ad49e8b6e58e34704e0c67',
                'decisionModel': 'https://overidentification-protection-authority.gov/context-models/081863fe03ad49e8b6e58e34704e0c67',
                'description': 'Check whether the bank account holder has passed away and the holder has been authorized to access the account in their w
            }
        },
        'proof': {
            'type': 'RsaSignature2018',
            'created': '2017-06-18T21:19:10Z',
            'proofPurpose': 'assertionMethod',
            'verificationMethod': 'https://example.edu/issuers/565049#key-1',
            'jws': 'eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5XsITJX1CxPCT8yAV - TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq - pQy7UJiN5mgRx
        }
    },
    'nonce': 'd30af247f61044d88e4a0f6e703ffba2'
}
```


Figure 6.4: Example of the disclosure of an authorization certificate (message 2)

Accordingly, the second message in which the verifier sends an authorization certificate to the wallet is modelled after the OpenID Authorization Response, with the authorization certificate being included as a `vp_token` (see also figure 6.4). The third message on the other hand, in which the wallet informs the verifier of their access rights, is not modelled after a specific message from the OpenID protocol, however it does follow a similar template, as can be seen in figure 6.5. This messages includes a parameter called `permitted_attributes` which lists the attributes the verifier is allowed to request as part of the current transaction between user and verifier.

```
{
    'response_type': 'access_permissions',
    'client_id': 'https://client.example.org/',
    'redirect_uri': 'https://client.example.org/',
    'permitted_attributes': {
        'DeathCertificateCredential.nameDeceased',
        'WillCredential.nameExecutor',
        'IDCardCredential.name'
    },
    'nonce': '40f21de6054346c6b6ce023c37fc9305'
}
```

Figure 6.5: Example of the message informing a verifier of their access rights (message 3)

The fourth and fifth message are the same as the OpenID authorization request and response from the OIDC4VC protocol respectively, as those two messages represent a "classic" SSI presentation exchange. The only change is that the fourth message contains an additional parameter called `nonce_challenge`, in which the verifier returns the nonce the wallet send to them as a challenge in the third message.

The demo application is terminal-based and requires user input for three different purposes, namely context verification, expression of personal preferences during the evaluation of a contextual decision model (if needed) and confirming they want to disclose the requested attributes (if the data request was legitimate). The decision models themselves are modeled as JSON files in order to make them more machine-readable. These files generally follow this template to represent the formal logic:

```
1 {
2         "contextID": "The formal contextID of the
              context modeled through this decision
              model",
3         "amount_rules": "The number of atomic rules
              of the form <conditions> -> <
              access_permissions>",
4         "operators": "The set of logical operators
              connecting the atomic rules from left to
              right",
5         "r_1": {
6           "condition": {
7                 "amount": "The number of conditions
                      that apply for this rule",
8                 "c_1": {
```

```
                    "predicate": "The condition
                       predicate of this condition",
                    "parameters": "The set of
                       parameters needed for the
                       predicate specified above"
                }
            },
            "access_permissions": {
                  "predicate": "The predicate used to
                     grant access permission. Usually
                     mayRequest",
                  "verifier": "The verifier whom
                     access is granted to. If this is
                     simply the current verifier and
                     not some specific verifier, this
                     value should simply be v",
                  "attributes": "The set of attributes
                      to which verifier specified in
                     the previous field should be
                     allowed to request"
            }
        }
}
```

Examples of context decision files used as part of the demo can be found in appendix A as well as in the Github repository under /src/decision_models. These are based on the example decision models defined in section 5.2.2.4.

Chapter 7

Discussion and Conclusions

The goal of this thesis was to answer the question if users can be protected against over-asking verifiers by requiring verifiers to obtain authorization for the specific context of a transaction from a recognized governing party (essentially a trusted third party) and prove to users that they are indeed allowed to collect the requested data as part of their transaction
To answer this question, we defined what over-identification or over-asking means for the context of this thesis, defined requirements for the bureaucratic aspects of verifier authorization, defined a decision model for determining which data a verifier is authorized to request in a specific context and defined a modified presentation exchange protocol.

However, there is one question that still remains, namely: Does this modified presentation exchanged as proposed during this thesis actually provide adequate protection against over-asking without unduly violating the privacy and interests of all parties involved?
This final chapter serves to address this question and reflect on the previous chapters. It will also address the limitations of this research and provide an overview of further work necessary to further address the research question and translate the contributions of this thesis into practice.

7.1 Evaluation

From the start, the desired contribution of this thesis was an extended presentation exchange protocol which involves the verifier proving in some way that they have authorization to request the attributes they are requesting in the specific context of their request. The technical concepts for this were introduced in chapters 5 and 6, laying down a logic to determine which attributes a verifier is allowed to request in a specific context, a verifiable credential with which verifiers can prove contextual authorization and the flow of the aforementioned extended presentation exchange protocol. Now

it is time to address in which ways these concepts provide protection against over-asking and in which ways they do not, i.e. where their limitations lie.

7.1.1 Protection against over-asking

The proposed verifier authorization scheme protects users against over-asking mainly by restricting the set of attributes that may be requested in a particular context. For every context there is a specific decision model that after execution returns a specific set of attributes that may be requested in a specific transaction. This decision model is executed by the user's wallet and is returned to the verifier the user is interacting with. If the verifier then tries to request attributes that are not part of the set of authorized attributes, the wallet would automatically deny the request as well as create a log of the event. This means that if verifier authorization is enforced by the wallet, verifiers are tied to what the wallet tells them they are allowed to do based on the authorizing party's decision model.

Additionally, since specific rules of the decision model may be tied to conditions, the set of attributes a verifier may be allowed to request can be different for different users. This allows the model to depict use cases such as the application form and mortgage application use cases described in sections 2.2.3 and 2.2.4, where the minimum set of data needed is dynamic since it depends on specific properties of the party that needs to disclose their personal data. This further minimizes the amount of personal data that can be requested in certain contexts.
The model also allows users to express user preferences, albeit in a limited fashion. Whenever there are multiple possibilities on which sets of data may constitute "legitimate" requests, they can choose to only allow the option they deem to contain the least sensitive set of data. It may also allows users to optionally disclose specific attributes where they are not strictly needed, e.g. when they may want to opt-in for more personalization or more personalized marketing. But it does not really leave room for further expression however. Users may express preferences when there are multiple options, however they are a) limited to these options and b) not all rules return multiple options, so even if they would rather not disclose some of the attribute returned as a result of these rules, the verifier will be allowed to request these attributes independently of the user's preferences.

Since it is the wallet that executes the decision model, there is less room for coercion by verifiers, as they are bound to whatever the decision model that the authorizing party (or parties) designed for a specific context outputs on the wallet side. So unless they somehow managed feed the wallet a fake decision model (which they should not be able to as that would invalidate the signature of the authorization certificate credential), they have no influence

on the outcome of the decision model. Verifiers may have some influence on the design of a decision model (especially if authorization is granted by a recognized assurance community which a verifier is part of), but once a decision model is set, verifiers will be forced to stick to it.

Another positive effect of the proposed verifier authorization scheme would be that it forces organizations and stakeholders involved to really assess which data they actually need and what risks actually warrant the collection of specific additional data, as they would need to have a convincing case for each transaction. In some capacity, this has already happened in the EU, as the GDPR already forces organizations to disclose which data they collect and why. However, the GDPR still offers the possibility to collect (and coerce users into providing) data that might not really be needed for specific transactions and does not tie data collection to specific transactions, which would be problematic when qualified data is exchanged (which would be the case with SSI). With the proposed scheme, organizations would be more limited in which data they may collect and in what context they may collect specific data. This would mostly be limited to data they strictly need and data that is necessary to cover a risk that warrants the collection of said data, which would be subject to the authorizing party's subjective assessment of its proportionality.

7.1.2 Limitations

The biggest limitation of the proposed protocol in particular is that there is currently no mechanism to have the wallet automatically verify the context, i.e. verify that the context stated by the verifier is actually the context of the transaction at hand. In the current proposal, the context is verified by authorization certificate including a user-friendly description of the context, which the wallet then shows to the user and asks them to confirm whether the description accurately describes their current interaction with the verifier. Verifiers should not be able to modify this description without wallets noticing, as that would invalidate the signature of the authorization certificate, which as a verifiable credential is tamper-evident.
This obviously requires active input from users and assumes they are at least paying enough attention to read the description, which may not always be the case. If the descriptions are not concise and/or clear enough, this may not always lead to accurate context verification, which would be problematic as the whole scheme relies on the verification of the correct decision model. Furthermore, the expression of personal preferences, which may be mandatory in certain contexts, would also require active user input and puts additional responsibility on users. Some users may find this additional interaction annoying and some might even find it overwhelming. If not implemented in a user-friendly manner by wallets, this might at worst

significantly reduce the usability of their software and may put users off. Here it is important that the interface does not automatically nudge users to hit confirm as is often the case with cookie banners, which make it very easy to hit accept but very hard and annoying to actually deny cookies. The difference is that cookie banners are created by those collecting data, while the context verification interface is designed by wallet providers who at least in theory should only have the interests of their users in mind. The visual design of this screen is therefore very important.
Authorizers and legislators can also disincentivise lying about the context for verifiers by introducing high fines and other significant punishments for violations. Here it is important that wallets log and report incidents to relevant authorities (e.g. their DPA) in which users indicated that the context stated by a certificate was not correct. The problem with this is of course that users can abuse this to report verifiers even when they did nothing wrong. Verifiers have a legitimate interest here to not be wrongly punished. To prevent this kind of abuse, the logs could include not just the message containing the authorization certificate and the description provided to the user, but also from which URL the interaction started, such that the authorities can reasonably determine if the verifier truly tried to deceive the user or not.

The fact that in the current proposal decision models should be publicly available to provide transparency may also have an unintended side effect of implicitly revealing certain properties of users. For example, if the model for mortgage applications includes a rule that states that if the applicant has a partner then they need to disclose both their own salary and the salary of their partner and otherwise they only need to disclose their own salary, the output of this rule will reveal if the applicant has a partner or not. Unfortunately, these types of implicit revelations cannot really be prevented unless the decision model is completely secret, such as in the approach by Anciaux et al. (2013) [4]. However, making the decision model completely secret would also require a much higher level of trust in authorizing parties from private citizens specifically and would be certainly not be ideal in regards to the value of transparency, which is pretty important for democratic societies.

Due to the high amount of different contexts, the scheme would also introduce a lot of bureaucratic overhead for both verifiers and authorizing parties, at least in the short term, if it were to be implemented in practice. Therefore its implementation would certainly take some time and would most certainly be a long-term project, possibly with multiple stages in between where it is implemented for SSI transactions in some sectors but not for others. In the long term, this overhead may become significantly less as decision models can be updated continually (e.g. in case some relevant laws change) and the number of new contexts should be manageable.

7.2 Further work

This research introduces a lot of requirements and concepts to prevent over-asking in SSI interactions. These are meant as a starting point from which an actual verifier authorization scheme could be designed and implemented in practice. As such, there are a lot of aspects that require further research, work or specification before such a scheme could actually be implemented in practice.

- **Legal aspects:** One aspect that was not covered during this thesis is the legal side of verifier authorization, as the main focus were the technical aspects. The proposed scheme would require a strong legal basis to be put into practice. This would need to go far beyond current data protection legislation such as the GDPR, as the current legal framework still leaves room for many types of over-asking and implicit coercion. Legislation is also needed to actually enforce verifier authorization since otherwise verifiers have a decent incentive to just not bother with the process. Here, further work is needed to work out which type of legislation is needed, where verifier authorization needs to be mandatory and where it might not be needed and what such legislation addressing over-asking in SSI needs to look like and cover.

- **Translating high-level requirements into more specific requirements:** In chapter 4 we explored some high-level requirements for a verifier authorization scheme and highlighted a few candidates as to who might act as an authorizing party, with a few arguments for and against each candidate. These requirements are intentionally kept generic as the specific governance and legal aspects were kept out of scope for this thesis. Obviously, for an actual implementation, more specific requirements need to be derived from these generic requirements depending on the given circumstances in which such a verifier authorization scheme should be implemented. This also includes a more detailed analysis of the stakeholders involved regarding their role, capabilities and resources. A follow-up research could therefore focus on defining a specific process for verifier authorization with focus on the governance and organizational aspects.

- **Further specification access control logic:** In chapter 5 of this thesis, we defined a simple syntax for an access control logic language which should determine the precise set of attributes which a verifier should be allowed to request during a specific transaction. This logic language is rather currently simplistic since my own knowledge and understanding of formal logic is rather limited. Further research could be done to create a more "sophisticated" logic language with more complex semantics for this purpose with a detailed analysis of the

syntax and semantics of such a logic language and an implementation with a logic-based programming language such as Prolog.
The current specification also defines very few predicates. As already mentioned in section 5.2.2.1, in an actual implementation more predicates would need to be defined to fully model specific contexts. So in the future it needs to be worked out what specific predicates would be needed to fully model all relevant conditions, ideally through predicates that represent broader condition instead of hyper-specific ones in order to minimize the amount of predicates needed.
Another aspect that is missing in the current logic language are negations. This is due to the fact that negations are rather semantically complex and could lead to conflicting rules. However, for some contexts negations for conditions and access predicates might be very useful or even needed. Therefore, the current logic language could be extended with negations. This would also require a deeper analysis of the formal semantics.

- **Automatic context verification:** As already mentioned in section 7.1.2, the manual verification of the context by users is the biggest weak link in the current proposal, as humans are fallible and might just hit accept without actually reading the description. Ideally, context verification should be automatic, which would also make the experience more user-friendly.
  One possible approach here might be extending contextual decision models. As a context is essentially a set of external conditions, in theory contexts could be modeled as sets of conditions within the decision model, for example by prepending a rule like $\neg\langle context\rangle \rightarrow blacklistAll(v)$, that states that if the conditions that model the context do not apply, then the verifier may not request anything (thus other rules do not need to be evaluated). The wallet should then log this and possibly report it to an authority.
  The difficulty here is to define suitable predicates to accurately model context and finding a method to actually verify these conditions. This is not a trivial task and might require a more complex logic language (therefore, this also relates to the previous point). Thus, a follow-up research could be conducted to update the existing protocol by implementing a method for automatic context verification.

- **Standardization:** This thesis introduces a lot of technical concepts such as the aforementioned logic language, verifier authorization certificates in the form of verifiable credentials and a new presentation exchange protocol which enforces contextual rules for the exchange of qualified data within SSI. All these concepts would need to be standardized to allow for interoperability and the integration of a given

verifier authorization scheme across different wallets and ecosystems. More specifically, a standardized logic language with a standardized set of predicates (especially condition predicates) and standardized verification methods, a standardized template for authorization certificate (e.g. by creating a specific standardized schema for them) and a proper integration of the proposed protocol into existing protocol standards such as DIDComm or OIDC are needed.

- **Full implementation of the protocol:** As part of this thesis, I created a demo application which simulates the protocol proposed in section 6.4. However that application mocks a lot of aspects, so it is not a "proper" implementation. For a future project, a full implementation integrated into an actual wallet application and existing protocol standard could be created as a pilot project to test the protocol in practice.

- **Perception of over-asking verifiers:** In section 2.1, we highlighted the inherent subjectivity when it comes to perceiving and identifying over-asking and over-identification, as different parties have different perception of what data requests are legitimate and which are asking for too much data. Especially for the legal and governance aspects of verifier authorization it might therefore be helpful to conduct legal/sociological research into why some verifiers are perceived as over-asking/over-identifying in the first place.

7.3 Acknowledgements

First of all I would like to express my gratitude to all three of my supervisors, Jaap-Henk Hoepman at Radboud University and Oskar van Deventer and Alexander van den Wall Bake at TNO, who provided me with useful input and feedback throughout the entire process of writing this master thesis.

I would also like to thank my TNO colleagues Rieks Joosten, for helping me define a mental model for over-identification (section 2.1)), Erieke Weitenberg and Marie-Beth Egmond, for their support for not just me but all interns in the ACQuA and CST departments at TNO, Daniëlle Keus and Hennie Jelsma-de Valk for helping me with administrative questions at TNO as well as Peter Langenkamp, Sebastiaan Tesink and Stefan van den Berg for their assistance with technical questions I had during the implementation of the demo.
Additional acknowledgement goes out to Leon Roseleur at KNB for providing me with the notary use case (section 2.2.2) and giving me a comprehensive overview of possible over-identification related to notarial acts. Finally, I would also like to thank all ACQuA and CST colleagues at TNO

Eindhoven for making my time at TNO very enjoyable and a pleasant work environment and experience.

Bibliography

[1] Martín Abadi. Logic in access control. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 228–233. IEEE, 2003. https://ieeexplore.ieee.org/abstract/document/1210062/.

[2] Martín Abadi. Variations in access control logic. In *International Conference on Deontic Logic in Computer Science*, pages 96–109. Springer, 2008. https://link.springer.com/chapter/10.1007/978-3-540-70525-3_9.

[3] Financieel Recht Advocaten. Hypotheek. https://www.financieelrechtadvocaten.com/ik-ben-een-particulier/hypotheek.

[4] Nicolas Anciaux, Walid Bezza, Benjamin Nguyen, and Michalis Vazirgiannis. Minexp-card: limiting data collection using a smart card. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 753–756, Mar 2013. https://dl.acm.org/doi/pdf/10.1145/2452376.2452472.

[5] Claudio A Ardagna, Sabrina De Capitani di Vimercati, Sara Foresti, Stefano Paraboschi, and Pierangela Samarati. Minimising disclosure of client information in credential-based interactions. *International Journal of Information Privacy, Security and Integrity 2*, 1(2-3):205–233, 2012. https://www.inderscienceonline.com/doi/abs/10.1504/IJIPSI.2012.046133.

[6] Steve Barker and Peter J Stuckey. Flexible access control policy specification with constraint logic programming. *ACM Transactions on Information and System Security (TISSEC)*, 6(4):501–546, 2003. https://dl.acm.org/doi/abs/10.1145/950191.950194?casa_token=JFbEIDR5YEcAAAAA:1Rs5So2RF5E77Gx9C_X8Q45qUwVE4kh7OQg1ndEMGRC864z6D-oSTKCPQZ0wBTD1VL1-yQnPn1xp.

[7] Centraal Beheer. Hypotheek execution only. https://www.centraalbeheer.nl/hypotheek/kennis/hypotheek-execution-only.

[8] Elisa Bertino, Francesco Buccafurri, Elena Ferrari, and Pasquale Rullo. A logic-based approach for enforcing access control. *Journal of Computer Security*, 8(2-3):109–139, 2000. https://content.iospress.com/articles/journal-of-computer-security/jcs131.

[9] Kevin D Bowers, Lujo Bauer, Deepak Garg, Frank Pfenning, and Michael K Reiter. Consumable credentials in logic-based access-control systems. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS'07), San Diego, California*, 2007. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/fox/mosaic/people/fp/papers/ndss07.pdf.

[10] Albert Heijn B.V. Privacybeleid van Albert Heijn B.V. https://static.ah.nl/binaries/ah/content/assets/ah-nl/core/legal/privacy/20211019-privacybeleid-albert-heijn.pdf.

[11] European Commission. What are Data Protection Authorities (DPAS)?, Nov 2019. https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-are-data-protection-authorities-dpas_en.

[12] Jason Crampton, George Loizou, and Greg O'Shea. A logic of access control. *The Computer Journal*, 44(2):137–149, 2001. https://ieeexplore.ieee.org/abstract/document/8140361/.

[13] Jan LG Dietz. *What is Enterprise ontology?* Springer, 2006. https://link.springer.com/content/pdf/10.1007/3-540-33149-2.pdf.

[14] Batya Friedman, Peter H Kahn, Alan Borning, and Alina Huldtgren. Value sensitive design and information systems. In *Early engagement and new technologies: Opening up the laboratory*, pages 55–95. Springer, 2013. https://link.springer.com/chapter/10.1007/978-94-007-7844-3_4.

[15] Bundesamt für Sicherheit in der Informationstechnik. Technical Guideline TR-03110-1: Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 1 – eMRTDs with BAC/PACEv2 and EACv1, Feb 2015. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_TR-03110_Part-1_V2-2.pdf?__blob=publicationFile&v=1.

[16] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-03110 Technical Guideline Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token, Dec 2016. https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/TR03110/BSITR03110.html.

[17] W3C Verifiable Credentials Working Group. Verifiable Credentials Data Model V1.1, Mar 2022. https://www.w3.org/TR/vc-data-model/.

[18] Jaap-Henk Hoepman. Civil Liberties Aspects of the European Digital Identity Framework, Jan 2022. https://blog.xot.nl/2022/01/31/civil-liberties-aspects-of-the-european-digital-identity-framework/index.html.

[19] Jaap-Henk Hoepman, Engelbert Hubbers, Bart Jacobs, Martijn Oostdijk, and Ronny Wichers Schreur. Crossing borders: Security and privacy issues of the european e-passport. In *International Workshop on Security*, pages 152–167. Springer, 2006. https://link.springer.com/chapter/10.1007/11908739_11.

[20] Corporate Finance Institute. What is Moral Hazard?, May 2020. https://corporatefinanceinstitute.com/resources/knowledge/other/moral-hazard/.

[21] Transparency International. Corruption Perceptions Index 2021, 2022. https://www.transparency.org/en/cpi/2021.

[22] Rieks Joosten, Stelle R den Breeijen, and Drummond Reed. Decentralized ssi governance, the missing link in automating business decisions. *Researchgate*, 2021. https://repository.tno.nl//islandora/object/uuid:a4357734-b80a-414e-be61-f62c017b74cd.

[23] Theano Karanikioti. CNIL, Europe's Strictest Data Protection Authority, is not invincible after all, Jun 2020. https://theplatformlaw.blog/2020/06/23/cnil-europes-strictest-data-protection-authority-is-not-invincible-after-all/.

[24] Nikita Khateev and Stephen Curran. Aries RFC 0036: Issue Credential Protocol 1.0, Jun 2021. https://github.com/hyperledger/aries-rfcs/blob/main/features/0454-present-proof-v2/README.md.

[25] Natasha Lomas. Facebook's lead EU Privacy Watchdog accused of Corruption, Nov 2021. https://techcrunch.com/2021/11/22/facebooks-lead-eu-privacy-supervisor-hit-with-corruption-complaint/.

[26] Natasha Lomas. Ireland's privacy watchdog sued over Google Adtech inaction, Mar 2022. https://techcrunch.com/2022/03/14/dpc-sued-google-rtb-complaint/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAKqlSyuyIFvCIIh6obf-sDg-QxeVR0AMVtnPP3RYUplv1zXeE5gcLBnKJokB9ZB8FcWDiZJYf3CLOJR97s-sdegaDsZq1sRItnvw482TXIcBcdSq-i3bnq8BNHcnXH4cY8VUBdrkWcxFq

[27] Jean-Grégoire Manoukian. Risk appetite and risk tolerance: what's the difference?, Sep 2016. https://www.wolterskluwer.com/en/expert-insights/risk-appetite-and-risk-tolerance-whats-the-difference.

[28] Jim Martin. How much Facebook earns from your data each year, Jan 2022. https://www.techadvisor.com/news/security/how-much-facebook-earns-from-your-data-3812849.

[29] Merriam-Webster. Overidentify definition amp; meaning. https://www.merriam-webster.com/dictionary/overidentify.

[30] Nitin Naik and Paul 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)*, pages 1–6. IEEE, 2020. https://ieeexplore.ieee.org/abstract/document/9272212/.

[31] International Civil Aviation Organization. Machine Readable Travel Documents (MRTDs): History, Interoperability and Implementation, Mar 2007. https://www.icao.int/security/mrtd/downloads/technical%20reports/icao_mrtd_history_of_interoperability.pdf.

[32] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), Jul 2002. https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32002L0058.

[33] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), May 2016. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

[34] Autoriteit Persoonsgegevens. Boete Belastingdienst voor Zwarte Lijst FSV, Apr 2022. https://autoriteitpersoonsgegevens.nl/nl/nieuws/boete-belastingdienst-voor-zwarte-lijst-fsv.

[35] Rijksoverheid. Wet op het notarisambt, Apr 1999. https://wetten.overheid.nl/BWBR0010388/2021-07-01.

[36] Geert Somers and Bernd Fiten. 2 Years GDPR: An Overview of Enforcement, Warnings and Fines, Jun 2020. https://www.timelex.eu/en/blog/2-years-gdpr-overview-enforcement-warnings-and-fines.

[37] Manu Sporny, Dave Longly, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. Decentralized Identifiers (DIDs) v1.0, Jul 2022. https://www.w3.org/TR/did-core/.

[38] Jens Strüker, Nils Urbach, Tobias Guggenberger, Jonathan Lautenschlager, Nicolas Ruhland, Vincent Schlatt, Johannes Sedlmeir, Jens-Christian Stoetzer, and Fabiane Völter. Self-Sovereign Identity: Grundlagen, Anwendungen und Potenziale portabler digitaler Identitäten. 2021. https://www.fit.fraunhofer.de/content/dam/fit/de/documents/Fraunhofer%20FIT_SSI_Whitepaper.pdf.

[39] SWIFT. What is KYC? https://www.swift.com/de/node/235031.

[40] Oliver Terbu, Torsten Lodderstedt, Kristina Yasuda, Adam Lemmon, and Tobias Looker. OpenID for Verifiable Presentations, Jun 2022. https://openid.net/specs/openid-4-verifiable-presentations-1_0.html.

[41] TNO. Eenvoudiger en veiliger digitaal leven met self-sovereign identity. https://www.tno.nl/nl/aandachtsgebieden/informatie-communicatie-technologie/roadmaps/data-sharing/ssi/.

[42] Boston University. The Syntax of Predicate Logic, Oct 2008. https://www.bu.edu/linguistics/UG/course/lx502/_docs/lx502-predicate%20logic%201.pdf.

[43] Alexandra Van Huffelen. Waarden, kansen en uitdagingen rond het Europese Digitale Identiteit Raamwerk, Jul 2022. https://www.rijksoverheid.nl/documenten/brieven/2022/07/26/waarden-kansen-en-uitdagingen-rond-het-europese-digitale-identiteit-raamwerk.

[44] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A logic-based framework for attribute based access control. In *Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, pages 45–55, 2004. https://dl.acm.org/doi/abs/10.1145/1029133.1029140?casa_token=NINW3Yw-JAkAAAAA:IQtVjLZ2su27xP_t2giNWi1kxa8CXKz_j6ehr4evofXMpcevyLDt8xen1REzE935gXQRhTW0ahr6.

[45] Ryan West, Daniel Bluhm, Matthew Hailstone, Stephen Curran, Sam Curren, and George Aristy. Aries RFC 0023: DID Exchange Proto-

col 1.0, Apr 2022. https://github.com/hyperledger/aries-rfcs/blob/main/features/0023-did-exchange/README.md.

[46] Kristina Yasuda, Michael B. Jones, and Torsten Lodderstedt. Self-Issued OpenID Provider v2, Jun 2022. https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

Appendix A

Examples of decision model files

A.1 Notary use case (see also section 2.2.2)

```
{
    "contextID": "081863fe03ad49e8b6e58e34704e0c67",
    "amount_rules": 1,
        "r_1": {
      "condition": {
        "amount": 1,
        "c_1": {
          "predicate": "isBank",
          "parameters": ["v"]
        }
      },
      "access_permissions": {
        "predicate": "mayRequest",
        "verifier": "v",
        "attributes": [
          "DeathCertificateCredential.nameDeceased",
          "WillCredential.executor.name",
          "WillCredential.executor.authorization",
          "WillCredential.nameSubject",
          "IDCardCredential.name"
        ]
      }
    }
}
```

A.2 Application form use case (see also section 2.2.3)

```
{
        "contextID": "bda90835c9fa4617ad86d0ae8c0d3e
           3f",
        "amount_rules": 1,
        "r_1": {
      "condition": {
        "amount": 0
      },
      "access_permissions": {
        "predicate": "mayRequestOne",
        "verifier": "v",
        "attributes": ["r_1_1", "r_1_2", "r_1_3"]
      },
      "r_1_1": {
        "condition": {
          "amount": 1,
          "c_1": {
            "predicate": "lessThanOrEquals",
            "parameters": ["PensionCredential.amount
               ", 10000]
          }
        },
        "access_permissions": {
          "predicate": "mayRequest",
          "verifier": "v",
          "attributes": ["PensionCredential.amount"]
        }
      },
      "r_1_2": {
        "condition": {
          "amount": 1,
          "c_1": {
            "predicate": "greaterThanOrEquals",
            "parameters": ["MedicalFileCredential.
               amountLostAbilities", 2]
          }
        },
        "access_permissions": {
          "predicate": "mayRequest",
          "verifier": "v",
```

```
            "attributes": [
              "MedicalFileCredential.
                 amountLostAbilities"
            ]
          }
        },
        "r_1_3": {
          "condition": {
            "amount": 2,
            "c_1": {
              "predicate": "lessThan",
              "parameters": ["PensionCredential.amount
                 ",30000]
            },
            "c_2": {
              "predicate": "greaterThan",
              "parameters": ["IDCardCredential.age",80
                 ]
            },
            "operators": ["AND"]
          },
          "access_permissions": {
            "predicate": "mayRequest",
            "verifier": "v",
            "attributes": ["PensionCredential.amount",
               "IDCardCredential.age"]
          }
        }
      }
}
```