Title:

Decentralized Identity Management System for Self-Soverign Identity

Theme:

Master Thesis

Project period:

February - June 2018

Project group:

Group ICTE4SER 4.2

Members:

Angel Angelov
Mihail Milkov
Markus Sørensen

Supervisor:

Henning Olesen

No. of Pages: 129

No. of Appendix Pages: 9

Total no. of pages: 138

Finished: June 6, 2018

Aalborg University Copenhagen
A. C Meyers Vænge 15
2450 København SV
Secretary: Maiken Keller
Telephone (+45) 9940 2471
mks@staff.aau.dk

Abstract:

In this project we investigate the problem of creating an identity management system that interacts with networks utilizing decentralized identifiers (DIDs) and decentralized public key infrastructure (DPKI). The identity system is called *DIdMS* (Decentralized Identity Management System) and is put into the context of user-service interaction. The aim was to develop a DIdMS concept that would server as a proof-of-concept and provide Self-sovereign identity to the individual. The DIdMS has been analyzed based on three major scenarios and as a result requirements were derived. From these requirements a small prototype was designed, which showcased how a DIdMS can be used by users to provide identities to service providers in accordance with the Self-sovereign principles and relevant regulations.

Aalborg University

Master Thesis

Decentralized Identity Management System for Self-Soverign Identity

---

Author:
Markus Sørensen, Mihail Milkov & Angel Angelov

Supervisor:
Henning Olesen

A thesis submitted in fulfillment of the requirements for the degree of M.Sc. In Engineering of Inovative Communication Technologies and Entrepreneurship

in the

Center for Communication, Media and Information Technologies
Dept. of Electronic Systems

June 6, 2018

AALBORG UNIVERSITY

Abstract

Technical Faculty of IT and Design
Dept. of Electronic Systems

M.Sc. In Engineering of Inovative Communication Technologies and Entrepreneurship

Decentralized Identity Management System for Self-Soverign Identity

by Markus Sørensen, Mihail Milkov & Angel Angelov

In this project we investigate the problem of creating an identity management system that interacts with networks utilizing decentralized identifiers (DIDs) and decentralized public key infrastructure (DPKI). The identity system is called *DIdMS* (Decentralized Identity Management System) and is put into the context of user-service interaction. The aim was to develop a DIdMS concept that would server as a proof-of-concept and provide Self-sovereign identity to the individual. The DIdMS has been analyzed based on three major scenarios and as a result requirements were derived. From these requirements a small prototype was designed, which showcased how a DIdMS can be used by users to provide identities to service providers in accordance with the Self-sovereign principles and relevant regulations.

Contents

List of Figures

List of Tables

List of Abbreviations

API	**A**pplication **P**rogramming **I**nterface
CR	**C**onsent **R**eciept
CRUD	**C**reat **R**ead **U**pdate **D**elete
DAPP	**D**ecentralized **App**lication
DDO	**D**ID **D**ocument **O**bject
DID	**D**ecentralized **ID**entifier
DIF	**D**ecentralized **I**dentity **F**oundation
DIdMS	**D**ecentralzied **Id**entity **M**anagement **S**ystem
DLT	**D**istributed **L**edger **T**echnology
DPKI	**D**ecentralized **P**ublic **K**ey **I**nfrastucture
GDPR	**G**eneral **D**ata **P**rotection **R**egelation
IdD	**Id**entity **D**irectory
IdMS	**Id**entity **M**anagement **S**ystem
IdP	**Id**entity **P**rovider
IoT	**I**nternet **o**f **T**hings
JSON-LD	**JSON** **L**inked **D**ata
MNID	**M**ulti **N**etwork **Id**entifier
PII	**P**ersonally **I**dentifiable **I**nformation
REST	**Re**presentail **S**tate **T**ransfer
RP	**R**elying **P**arty
RWOT	**R**ebooting **W**eb **O**f **T**rust
SP	**S**ervice **P**rovider
SSI	**S**elf-**S**overeign **I**dentity
TLS	**T**ransport **L**ayer **S**ecurity
VC	**V**erifiable **C**redential
W3C	**W**orld **W**ide **W**eb **C**ornsortium

Chapter 1

Introduction

Technology has thrived and has continued to develop and expand at an extremely accelerated pace. This process has facilitated the proliferation of new services which find their way in numerous aspects in our daily lives. The growth of the Internet has made the world very interconnected which has granted the service providers with the opportunity to reach a vast customer base, constituted by individuals from all around the globe and to be able to do that in a fast and efficient manner. Different service industries, such as healthcare, banking, entertainment and many others have been made accessible without the need to request the users to leave their homes.

In order to use the services they would like, the users need to register as a customer to the respective providers. During this process of registration, the user has to provide some type of information which is, in turn, used by the service provider in order to deliver a better individually customized experience. In the past, some of the types of information would not even be stored anywhere or it would have been written on a piece of paper placed away in a folder in some dusty cabinet. However, with the technological advancement these types of data are stored in rich databases with different fields containing a wide spectrum of personal information. A larger problem is arising from the combination of the aforementioned processes. The more services made available, the more opportunities the users have, but they have to go through the same sign up process of each of them individually which in turn results in losing transparency of the data they have provided. In other words, consumers are trading their data in order to obtain access to different benefits and services.

The provision of the data all over the Internet means that the consumers cannot keep track of what data they have provided to the different services and what it used for. Recently there have been a lot of cybersecurity and privacy issues covered by the medias which has increased the awareness of the general population of online users and has resulted in the emergence of trust issues between customers and service providers. This brings about a new challenge which needs to be tackled. The trust gap between the consumers and the providers needs to be bridged and the handling of personal information made more manageable.

1.1 Background & Motivation

With the new technologies emerging and all of the numerous possibilities that are being opened to humanity new problems arise.

1.1.1 Identity & Trust

Before we dive deeper into digital identity, it is important to first define what does the term *identity* actually exemplifies. The french philosopher Paul Ricoeur defines identity as a notion which is associated with two different aspects. The first one being *Idem*, which is the latin word for *sameness*[1]. This facet of identity deals with characteristics that never change. In other words, identity are the set of features that persist through time and will keep an entity the same[1]. The second aspect defined by the philosopher is *Ipse*, which is translated from latin to *selfhood*[1]. This aspect of identity is associated with attributes and characteristics which are resilient to change throughout time. I.e. it is constituted by features that make someone unique among a set of others[1].

The philosophical aspects of identity are rather complex and sometimes hard to grasp. There are also other definitions related to the term identity. Chadwick defines digital identity as *A set of characteristics or attributes that can uniquely identify and distinguish one entity from another in a given context.*[2]. From both definitions we can derive that an identity is a concept which represents an entity through features that either make that entity unique or that persist through time. From this we can conclude that identity allows us to better know other entities we are interacting with. By gaining this knowledge we can easily separate the good from the bad, which in turn allows us to establish a foundation on which we can build trust[3].

The term **trust** is defined as *a firm belief in the truth or ability of someone or something to perform a task in a reliable matter, in a specific context*[4]. Trust is an essential aspect and it is the "glue" that holds our society together. If there was no trust, there would be no cooperation between parties, trade and commerce would not be possible and governments would not rule[3].

In figure 1.1 an illustration of the different contextual situations an individual might find themselves in are shown. Each of the different situations are concerned with different attributes which are disclosed by that individual. Each of the smaller ellipses are the different **partial identities** which the individual uses to represents themselves to other parties[5]. All of these partial identities make up the complete identity of that same individual[5].

In our daily lives identity is established through presenting different attributes. There are two distinct cases between which we need to differentiate. An example of the first case is an individual stating that their favorite ice cream flavor is chocolate. This can be seen as a **claim**[3]. The individual is claiming that chocolate is their favorite flavor. These types of claims bear no consequences and can be self-asserted[3]. However, lets take the situation that the same individual goes to a car dealership and would like to lease a new car, for which they would have

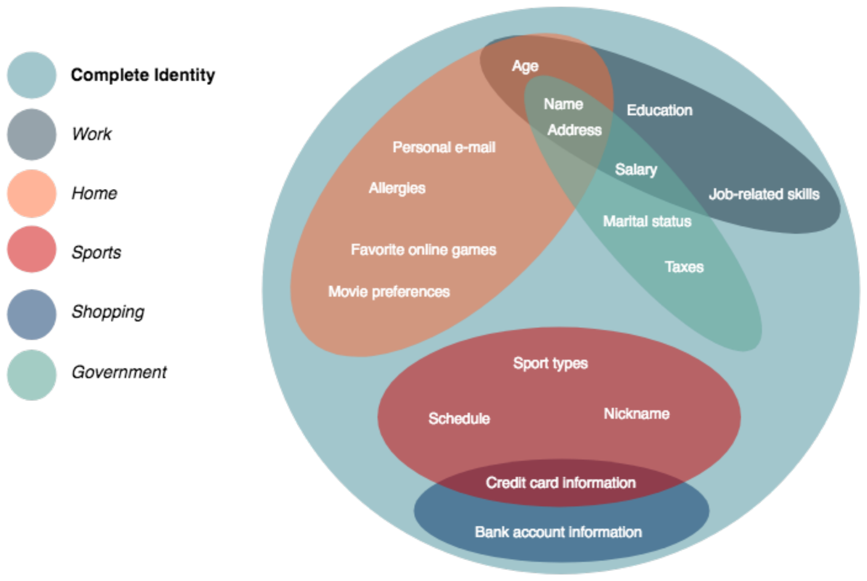


Figure 1.1: Contextual-specific partial identities

to make monthly payments. In this situation they cannot simply claim that they would be able to make the payments. They need an **assertion**. An assertion is a confident and forceful statement of fact or belief [6]. This assertion needs to be provided by an authority which is trusted by both the potential buyer and the salesman. In this case the most reputable entity would be a bank. The individual can go to a bank and request that they issue a statement, based on some attributes that he has presented to them, which proves that he is able to pay the necessary monthly amount for the car he would like to buy. This assertion then needs to be put on a transportable medium. Requirements which are related to the medium include that it has to be resilient to falsification and it has to explicitly state who the issuer is. Furthermore, it should also distinctly present the object of the assertion. In our case this could be a statement printed on a paper which is signed and stamped by the bank to prove its authenticity.
Based on above real-life example a *complete identity* consists of following elements:

Attributes Descriptive elements of the related identity.

Claims An attribute which have been claimed to be true towards another entity.

Asserted Claims A claim which has been asserted true by a trusted issuer.

Partial Identities A subset of attributes and claims, not containing the complete identity.

Digital Identity

With technological advancements, the Internet has evolved beyond being a digital environment filled with static content. Additionally, it provides many different services to its consumers[7]. However, the internet was built without an identity layer[6]. This flaw is depicted by a very popular cartoon created by Peter Steiner, and published in *The New Yorker* in 1993[8]. The single comic strip features two dogs, one of which is sitting at a desk in front of a computer and is speaking to the second dog, saying - *"On the Internet, nobody knows you're a dog."*[8] The cartoon illustrates the problem that on the Internet an individual can be whoever they would like to be. In contrast, when we take the same situation, in the context of the physical world, individuals usually have knowledge of who are they interacting with. Another definition of the problem is presented by Kim Cameron in [6] -*"The internet was built without a way to know who and what you are connecting to"*.

Kim Cameron suggests that an identity should be based on claims. The identity architect defines the term *digital identity* as *"a set of claims made by one digital subject about itself or another digital subject"*[6]. He further defines claim is *"an assertion of the truth of something, typically one which is disputed or in doubt"*[6]. The same parts which are relevant for physical identity are relevant for the digital identity as well. However, some challenges which are not present in the physical world are present in the digital.

Identity in the digital realm can be established by providing a claim, however, the aspect of being always in doubt implies that there are some cases in which further verification of a specific claim might be necessary[3]. In other words, an assertion by a trusted party needs to be provided, analogically to the physical world example with the car purchase. I.e. there needs to be a party which is as reputable as the bank, that is able to assert a digital claim. These assertions need to be packaged in a secure format, equivalent to an envelope in the physical world. This format has to allow the receiving party of the claim to validate the issuer of the assertion and also its subject.[3].

Furthermore, it is important to specify the types of data which are conveyed with these claims. Kim Cameron states that the claims *might convey personally identifying information – name, address, date of birth*[6]. However, it is important to define the term personally identifying information and what are its implications. Additionally, what are the rights does the owner of such information have. E.g. the owner might not always want to be identified,

According to a recommendation by NIST the general term which constitutes the different types of information disclosed by the user to the service providers they use, including the user's name, electronic and physical address, is *PII*[9]. The definition of PII given by NIST is *any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or bio-metric records; and (2) any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information"*. The phrase *to distinguish an individual* is associated with the process of identifying a specific entity through the use of their personal information. The term *trace* refers to the process of performing a series of actions which can then allow the process to be able to make definitive conclusions on the specific individual's status or activities[9].

GDPR (General Data Protection Regulation) describes and addresses *PII*, however they have coined the term by using a different naming convention - *personal data*[10]. The definition of *personal data* is - *any data that directly or indirectly relates to an identified or identifiable natural person.*[10], further presented in section 3.1.1. Furthermore, the term *identifiable natural person* represents an individual who can be identified directly or indirectly in reference to an identifier. Such identifiers include name, location, and other factors that are closely related to the physical, psychological, cultural, social and economic identities[10]. Furthermore, the regulation defines two primary categories which both fall under the term *personal data* - *common* and *sensitive*[10]. The terms which fall into the latter category are the different factors such as racial or ethnic origin, political opinions, religious or philosophical beliefs and others[10].

1.1.2 Management of Digital Identity

Before introducing the different flows which are associated with the process of managing a digital identity, a definition of the different actors which participate in the flows described in the latter parts of this subsection is presented[11]:

The User A person who uses an information system and its resources for any purpose.

The Identity Provider A service entity which creates, maintains and manages identity information for users and provides user authentication on behalf of other service providers.

The Service Provider A system entity that decides whether or not to take an action based on information provided by another system entity - e.g. an Identity Provider.

So from the above identified digital identity and what it contains, how does individuals manage it in to days world? There is two main ways individuals distribute their identity to services providers online - also dictated by the needs of the service provider. In the first case the individual communicate solely with the service provider and the conveying of claims is happening explicitly between the two entities[11].

The second case is when the SP needs some claims to be asserted by a trusted issuer. Introducing the need of providing assertions by a trusted party means that there is a third entity that is involved in the flow when a user is interacting with a service provider - An Identity Provider. In figure 1.2 the flow of information between the three primary actors are illustrated. First there need to be established a relationship between Identity Provider(IdP) and Relying Party(RP). When the User initially interacts with the RP **(1)**, the RP will specify IdPs which it trusts to assert claims. The User will then go to one of the IdPs - one the User trusts - and ask for asserted claims for the RP, **(2)**. After receiving the asserted claims, the User will take these to the SP, **(3)**.

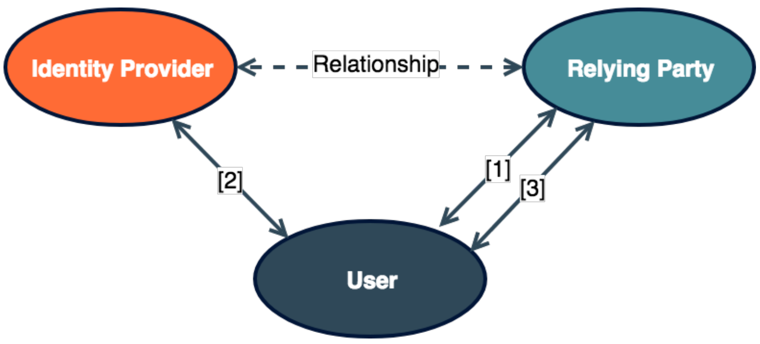


Figure 1.2: Current Identity Management Flow.

Furthermore, Kim Cameron, Microsoft's Identity Architect, defined the *Laws of Identity*[6]. These laws need to be complied with in order for an identity management system to be deemed good:

- People should be in control of how the disclosing of personal information is done, similarly to the physical world.
- The amount of information which is being disclosed should be sufficient for the purpose it is released for and the entities which are receiving it should be only the ones who need it. Furthermore, additional details and information should not be kept for a longer period than necessary.
- The user should be given a choice of terms and conditions over who provides their identity information
- It should not be possible to link up all of the aspects which describe how is an individual behaving on the Internet.
- The different devices and technologies which are utilized should offer the same kind of identity controls to the user.
- The system should incorporate mechanisms which ease the process of human-machine communication.
- Users are able to experience a consistent experience throughout different contexts of use.

The absence of the identity layer on the Internet has made it necessary for service providers to find their own solutions to how will their users be identified. These solutions included the users filling in forms with personal information and other attributes such as name, date of birth, address and so forth. The combination of these attributes can then be used to identify each of the individual consumers of the service providers. However, this process resulted in every individual creating a multitude of identities which are **scattered** around the Internet for various services, all pointing to the same entity. For the users, the management of these scattered identities is extremely challenging.

1.1.3 Privacy Issues

The following section presents an introduction to the term identity and its different concepts and definitions. Furthermore, we will present the various aspects when it comes to dealing with identity in the digital realm.

There were also certain drawbacks to the aforementioned model. Most of them are related to conflicts with the laws of identity, defined by Kim Cameron[6]. The most crucial aspect that they breach is **privacy**. We have defined the term identity and have outlined that it is essential to building trust. However, there is another concept which is closely related to identity - privacy. The definition of the term privacy that we are going to refer to in the context of this report is *Privacy of a physical entity is the result of negotiating and enforcing when, how, to what extent, and in what context which of its data is disclosed to whom*[12].

In the last decade different events revolving around privacy have occurred, which dramatically changed the situation regarding the awareness of the general public. One of the most notable events is the revelations of the former government contractor - Edward Snowden[13], which made people all around the globe conscious that consumers of services need to concern themselves with how their data is being handled and is it used for only the intended purposes for which they have provided it initially.

From the revelations made by Edward Snowden it was obvious to that there was no user consent and control. The services did not comply with the requirements of minimal disclosure for a constrained use. The spectrum of information that was collected by services was so wide that some people might ask why did the parties involved need all of this data. Furthermore, some of the actors that were in the middle of these occurrences were unknown until the very last minute which is, again, going against one of the laws suggested by Kim Cameron - *Justifiable Parties*. Furthermore, the systems did not consider the use of *Directed Identities* law which would reduce linkability. In other words the information and identifiers should be used in such a way that there would be no possibility to distinguish a specific entity based on a set of attributes or a partial identity. However, the way they did it was actually the opposite - the design choices made favored linkability. Furthermore, there was overexposure of data. This meant that everytime the user went to an IdP that entity could register the different Relying Parties that the individual is interacting with. Moreover, the RP usually gets too much information which can pose additional privacy risks associated with the identity of the user[14].

Other problems which did not subside include phishing - in the case when users are being unable to differentiate or are being lied to about the genuine of a website this resulted in them authorizing a malicious service access to their attributes. Additionally, a type of credential harvesting attack which is executed on a larger scale is pharming - this includes redirecting the traffic of a website to another fake cite. This allowed criminals to perform identity theft[15].

Furthermore, user tracking, which is done through cookies, logs and primarily the smart devices they use, has proved to be a great breach of privacy. An investigation done on the different applications used on both Apple Smartphones and Android Phones shows that out of a total of 101 popular mobile applications tested 56 of them transmitted data to unknown parties without the user being notified or agreeing to this data transaction[16]. These findings unveil how much other parties actually intrude our privacy. Another recent event which occurred was the Facebook scandal[17]. This showed us that the personal data of the individuals has become a currency for the bigger corporations. However, these events have not gone unnoticed by the public and people are becoming more aware of what is happening around them. Different statistical studies have been conducted which support this statement. One of them showed that people are concerned about their privacy, however, they are not knowledgeable enough in order to do anything about it[18]. Another study carried out in the US shows that 96 percent of internet users in the US fear an attack on their privacy and personal attributes by hackers[19].

1.1.4 Challenges and Issues

From above the following is considered problems and challenges regarding the current models of individuals managing their digital identity:

Disclosure of Information The current models have limited support for the user to choose the exact claims given to service providers. This relates to both how many claims is need as well as the information in the claim. E.g. if the SP need to know if the user is over 18, they do not need to know the exact birthday.

Validity of Claims As the use of Social IdPs are very popular, the validity of their asserted claims can be questioned. E.g. creating an account on Facebook, one can enter whatever name, mail, etc. they want.

Linkability With a centralized IdP the user are able to be linked between different service providers using the same IdP. Multiple SPs can correlate on the IdP unique identifier and construct a more complete identity of their users.

Scattered Identities Instead of the individual having a complete identity, the user have multiple scattered identities. This presents a big problem in managing all these identities and the attributes in them. Resulting in SPs having having old data and the individuals not having a single place for updating their attributes.

IdP knowing too much Due to the IdP asserting to claims for multiple SPs they will know which services the user is consuming.

Missing Support for Partial Identities Following the example of the physical world where the user can represent themselves in a different manner depending on the context, this aspect has not been reflected in the digital world.

No Consent Management Current models do not provide the user with consent management, meaning they have no chance of knowing who has their. This also means that the possibility of revoking consents is limited.

These current challenges and issues are regarded as contradicting Kim Cameron's Laws of Identity, which means the current IdM systems are failing.

The year 2018 marks an important objective for data privacy and security in EU countries. In May, the General Data Protection Regulation becomes active and this will change the way people share their personal data across the web[10]. In addition, the different participating organizations, such as IIW[20], have been working on changing the way identity management is done and there appears to be a light at the end of the tunnel - that light is called *Self-sovereign identity*.

1.1.5 Self-Sovereign Identity

As the digital identity has evolved through time and complication with managing them, a new concept called *Self-Sovereign Identity*(SSI) has emerged. The term is not necessarily new, but the definition and importance of it is current. As one of the first mentions of the concept and proposed definition, was done by Christopher Allen, Co-author of TLS Security Standard, in blog post. In this post he describe the evolution of digital identity and propose this term[21].

> *"Self-sovereign identity is the next step beyond user-centric identity, and that means it begins at the same place - The user must be central to the administration of identity."* - C. Allen[21]

When looking at the evolution of identity, C. Allen stated that the time for self-sovereign identity is now. He defines the concept by stating ten principles:

Existence *Users must have an independent existence*

Control *Users must control their identities*

Access *Users must have access to their own data*

Transparency *Systems and algorithms must be transparent.*

Persistence *Identities must be long-lived*

Portability *Information and services about identity must be transportable*

Interoperability *Identities should be as widely usable as possible*

Consent *Users must agree to the use of their identity*

Minimization *Disclosure of claims must be minimized*

Protection *The rights of users must be protected.*

The post concludes by stating that these principles and definition should be looked at as starting point for discussion. Together with Shannon Appelcline, C. Allan constructed below definition for a Rebooting the Web Of Trust[1] design workshop back in the fall 2017:

[1]See section 3.2.1

Self-sovereign identity is centered on a person, free from dependence on any corporation, organization, or nation-state.[22]

The SSI concept targets identity providers and mangers. It introduces a set of goals or ideals that should be achieved by the new-coming identity management systems.

The question that comes to mind, following this discussion, is how would an Identity management provider (system) follow and implement these goals. With the complexion of digital identity, the challenges of trust and the defined new objectives, how do we create the next generation of identity that can benefit the general public? These were the major question that this report investigates. In the following section we introduce concrete problem field that defines the scope, research and direction of this paper.

The regulation also defines an identifiable natural person as one who can beidentified directly or indirectly in particular in reference to an identifier. Examples of identifiers would be name, location, identification number. Also, other factors related to health, bio-metrics, cultural and social identity can identify a person

1.2 Problem Description

In this section we will present the problem field and problem definition of the paper. This is based on the challenges defined in the preceeding section, as well as principles of self-sovereign identity(SSI). As a general problem formulation that defines the research objective, we propose:

> **How can individuals manage their digital identity via a modern identity management solution that utilizes the core principles of SSI and ensures that user rights are protected as specified in the GDPR?**

This research question is divided into smaller sub-questions, which are presented below. Each of these sub-problems/questions identifies key area that builds to the general problem formulation. Answering them contributes to the solution of the main research question and enables us to gain more elaborate insight into the challenges and problems that need to be addressed:

1. *What architecture/ecosystem can enable the independent existence and persistence of a digital identity?*
   An identity management solution or provider is rarely a single standalone unit. Usually, it is part of a bigger architecture or ecosystem that enables specific flow or functionality. We have presented, thus far, how the current model works and by asking this question we aim at investigating what relevant research and ideas are developed that could enable an ecosystem where an identity solution can be build to achieve the SSI principles.

2. *"How can we enable the user to have control over the management of their identities and minimize the disclosure of claims?"*
   A core part of any identity management solution is the actual structure, storage and overall life-cycle of the data that is processed. Answering this question will help us understand what requirements have to be met in order to structure, store and provide the user's identity data.

3. *How can the user provide asserted claims in a well-defined way to service providers?*
   Another important aspect of data provision is its data-structure. In order for computers to exchange data and enable services, a machine-readable format and structure has to be established so that user agents can seamlessly transmit data to requesting parties. The answer of this research question should give us more insight into how to transmit, form and validate identity transactions between the different parties. For example, how technically would the user data be provided to interested service providers?

4. *What are the privileges granted to the individual by GDPR and how can these be facilitated by the identity management system?*
   As GDPR is being introduced, the dynamic between a consumer(user) and a service provider is being changed. Service providers now have to support newly given rights to the user and more explicit consent flow. The answer of this question aims at establishing necessary rights and requirements that the service provider needs to support in regards to the user. The relevance

to the identity management system, comes from the idea that users would use the identity management solution to interact with the service provider. For example, user uses the identity management system to create and provide identity data to a specific service.

After introducing the problem field, we now have to scope the project. In the following section we present the scope and delimitation, chosen for the research.

1.3 Scope & Delimitation

In order to efficiently be able to answer the aforementioned questions and complete the objectives different delimitations need to be defined.

1.3.1 Scope

1. Only partial requirements and implementation will be made, so that proof-of-concept can be established
2. The identity management solution will be analyzed in accordance with specified scenarios
3. The identity management solution will consider the transaction of identity data between a Service provider and a User. Any additional cases of interaction will be omitted.
4. No user involvement will be present during the development cycle.

1.3.2 Delimitation

1. Due to time restrictions a full-scale implementation of the system will not be possible.
2. Some of the standards discussed will only be reviewed, but not implemented.
3. It is assumed that internet connectivity is always available so no reasearch and discussion on how will the system.
4. Due to the scope of prototype, we employ platform specific frameworks and programming languages that we are familiar with.

1.4 Reader's Guidelines

The report is constituted by a total of 8 chapters. The first chapter sets the scene with an introduction to how identity and trust are established and what are the associated problems with the employed model. Furthermore, a proposed emerging concept which solves all of the problems is presented which is used as a build up to the problem formulation addressed in the report.

The second chapter presents the methodological structure and approach which was taken when dealing with the problem at hand. It specifies how has the requirement elicitation is done and the development approach followed and the different tools. It is important to state that the structure presented in this section does not follow the structure presented with the methodological process.

The third chapter reviews the associated literature which needs to be taken into consideration. This includes information about the General Data Protection Regulation and all of the current drafts, and their respective authors, of different specifications which are necessary to enable us to achieve the SSI concept. Furthermore it presents the current market products which have either been developed or are a work-in-progress which are yet to establish themselves.

The forth chapter encapsulates the analytical process executed with all of the different aspects which need to be considered in order to be able to complete the different principles defined by the SSI concept and also other implications, e.g. the implications associated with the GDPR.

The fifth chapter presents the design process of the proposed solution to the stated problem formulation.

Furhtremore, the sixth chapter describes the development process and the implmentation steps that we have gone through in order to create a proof-of-concept of the proposed system.

Chapter seven discusses the different challenges and aspects which have not been considered in-depth throughout the project.

Finally, chapter eight concludes the report. It presents a discussion which encapsulates how well have we answered the defined problem formulation and its associated supplementary questions.

Chapter 2

Methodology

As a limited time project, the result is not to have a complete system implemented. The main focus is on the requirements for such an identity management system and how it can solve some of the current shortcomings of IdMs presented earlier in the background, section 1.1.

Note that this project is being developed as a generic product, as the software specification is solely controlled by the authors of this report[23]. This means the requirements for the system design are not identified in cooperation with a customer.

For these needs, the engineering spiral by I. Sommerville [23], presented in figure 2.1, could be a good fit. This spiral illustrates and describes how the process of collecting and identifying requirements can be facilitated. The use of such spiral model accommodates requirements at different levels of detail[23]. This fits together with an agile development process which can be used during the prototyping phase. For this project the spiral is divided into two initial main phases, which should not be seen as linear. These phases are illustrated in the figure with two colors dividing the spiral - Red and Blue. The model allows for iteration, which means that we will use iterative process, moving around the spiral. Note, the last part of the spiral - "Reviews" would include a customer reviewing defined requirements, we will omit this part since it falls outside of our scope.

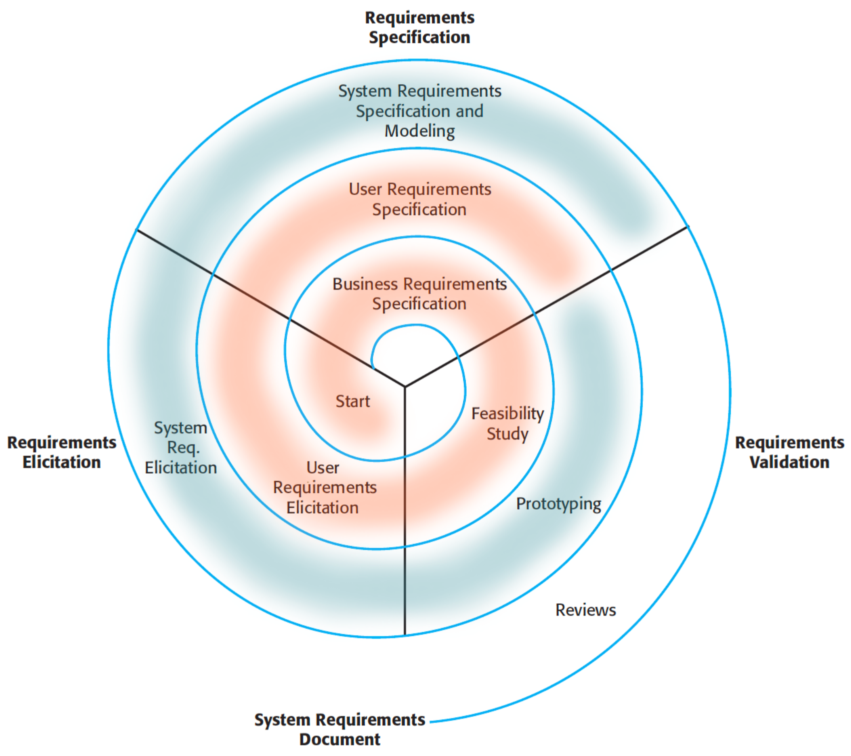

Requirements Specification
System Requirements Specification and Modeling
User Requirements Specification
Business Requirements Specification
Start
Feasibility Study
Requirements Elicitation
System Req. Elicitation
User Requirements Elicitation
Prototyping
Requirements Validation
Reviews
System Requirements Document

Figure 2.1: Requirement engineering process[23] with two main phases.

The red colored "first" phase is referenced as the research and requirement elicitation process. This is to establish general business knowledge of the area this project concerns and specify the user requirements for such a solution. Having some basic user requirements defined, this project will move into the "second" phase. Note that the wording ’first’ and ’second’ referencing the phases is in quotations, since it can loop, e.g. "first" phase the becomes third and so on. The second phase is where developing is happening, and this project will try to follow a more elaborate agile developing process to further identify requirements whilst building a product.

2.1 Research & Requirement Elicitation

The *’Business Requirement Specification’* and *’Feasibility Study’* are the beginning of this initial phase of the project. Here the focus is on gather knowledge and understanding of the high level "business" - domain knowledge - and look at the state-of-the-art. This is doing research on the basic idea of what the imagined system should be able to do. By looking into each of the research questions, presented in section 1.2, current solutions, technologies, standards and other, relevant work will be evaluated. For this part of the phase desktop/secondary research will be done. Mainly collecting and evaluating relevant material which can help answer the research questions.

Once establishing what is feasible, the *'User Requirement Elicitation'* can begin, which should conclude in an initial *'User requirement specification'*. The basis of the elicitation will be based on user scenarios describing the use of the imagined system, combined with the material researched earlier. The involvement of potential users is deliberately kept out, since the benefits in this initial phase are low. These scenarios would also assist in scoping the functionality of the system. Additional rounds in the spiral could result in additional scenarios.

2.2 Development

As the second phase of the spiral includes *prototyping* and specifying the *System Requirement*, this project will use an agile development process to assist. The initially identified user scenarios and requirements will be used as a starting point, but as the spiral progresses they could be subjected to changes. To facilitate the agile process will use a mixture of concepts from various methods [23]:

- Initial architectural design of the system, containing the main components.
- Design, implementation and testing is intertwine.
- Simple design, meaning design is done when needed and to a minimum. I.e. no complete or extensive design, since are very much subject to change, based on implementation.
- Small releases and incremental planning. Meaning useful functionality is delivered more often and therefore could help identify further requirements.

2.2.1 Use of UML

As a toolbox to assist in the development of the software UML is used. Unified Modeling Language consists of a standard way of constructing diagrams to help in the process of developing and documenting[24]. In this report especially sequence and class diagrams will be used at different levels of the process. Use case diagrams will also be used to help map the functionality provided by the actors.

Chapter 3

Research

The following sections will provide insight and description of the different concepts and already existing solutions which need to be reviewed. This is done so that the different objectives, previously stated in section 1.2, can be achieved. By reviewing the different solutions and exploring the concepts applied within the area of our project we will be able to identify the different problems and solutions, relevant to the problem statement. The chapter is divided into three parts. The first part introduces GDPR and organization that work in the field of SSI and identity management. These organizations also produce a variety of projects, most relevant of which are presented as a follow up in the second section of this chapter. Finally, we review state-of-the-art solutions that employ SSI principles and provide identity solutions similar to the problem statement.

3.1 General Data Protection Regulation

The processing of personal attributes and data concerning users are going to be regulated by the EU *General Data Protection Regulation* (GDPR)[10] as of 25th of May 2018. This means the service providers will need to meet certain restrictions and requirements presented in the GDPR, when handling such personal data. In regards to the problem statement, we have to examine what exactly GDPR is, what rights does it introduce for the user and what the service providers need to consider.

In this section relevant material within the regulation is presented in regards to help answer research sub-question 4 presented in section 1.2.

3.1.1 GDPR Definitions

The following list defines the entities that GDPR identifies when personal data transactions occur.

Personal data Any data that directly or indirectly relates to an identified or identifiable natural person (**data subject**).

Controller The entity which alone or in unity with others establish the purpose and means of the processing of a data subject's personal data.

Processor An entity which processes personal data on behalf of the controller - could be the same entity as the controller.

Consent A consent of the data subject, is a freely given, specific and unambiguous statement of how his/hers data should be processed.

3.1.2 Obligations for Entities

The above listed *controller* and *processor* have certain general obligations, which are described in chapter 4, section 1 of the GDPR[10]. Below some of these are presented.

Controller

The controller has the primary responsibility of insuring that the processing is happening in compliance to the GDPR. The controller has to make sure that all of the requirements for storing, protecting, exporting and transmitting personal data are met. Furthermore, the controller is to enforce *data-protection by design and by default*. The controller, in practice, is usually any service provider that requires personal data. In some cases, a controller could send data for processing to another controller. When such instances are present, the two controllers need to establish clear responsibilities and rights in regards to the processing of data. Overall, if an entity is assuming the role of a controller, it has to make sure that all of the processing is in compliance with GDPR [10].

In addition to that, the controller can provide data to processors. In the case of a controller wants to appoint a processor, an agreement has to be made that ensures the privacy and lawfulness of the processing. This means all processing must implement appropriate technical or organizational measures to ensure the protection of the rights of the data subject. The appointed processor must not, without the knowledge and authorization of the controller with another processor. Finally, the appointment of a processor shall be done in form of a binding written agreement, which states that the processor have certain responsibilities. These must as a minimum include stipulations specified in Article 28(3).

Processor

As already stated, a processor is appointed by a controller, and he is to follow the instructions established mutual agreement agreement. If the processor should fail to comply, i.e. act on their own, the GDPR specifies that "the processor shall be considered to be a controller in respect of that processing" [10].

Consent

As the definition of a consent outlined above implies, a consent is a description or agreement between a controller and a data subject of how the controller may process data relating to the data subject. Article 7 of GDPR declares four *Conditions for Consent*:

- The controller must be able to demonstrate that the data subject has given consent
- The request for consent shall be presented in a clear and unambiguous manner
- Withdrawal of a consent by the data subject must be possible

- The performance of the contract is not conditional on the data subject given consent.

When developing an identity management system, it is important to consider the GDPR since the identity management system could be viewed as a controller (depending on the use case), which would result in additional requirements. Furthermore, the identity management is an enabler for the user when he provides data to services, hence the identity management must be able to support the new user rights e.g. the right to be forgotten. This is one of the most relevant parts that GDPR outlines for our problem definition.

3.1.3 GDPR Data Subject Rights

As part of the problem statement, we set as an objective to find what privileges the GDPR defines for users and try to outline and provide them as part of a solution to the problem. As explained in the GDPR regulation [10] there are several rights defined:

- The right to be informed.
- The right of access.
- The right to rectification.
- The right to erasure.
- The right to restrict processing.
- The right to data portability.
- The right to object.
- Rights in relation to automated decision making and profiling.

We will not consider supporting all of the listed rights, but will instead try and analyze those who are relevant to the data and identity - The right to rectification; The right to erasure and The right to be informed. These are the core rights that we view as most important. They reflect the right of the user to have privacy and also remove unwanted trails.

In the analysis we will elaborate on them, as we will explore how can they be supported within the context of SSI and identity provisioning.

3.2 Relevant Work

After presenting GDPR, in this section we introduce the relevant entities - foundations, government bodies and working groups working in the area of identity and data privacy. The activities of these entities are relevant to our aim of developing SSI identity management solution.

3.2.1 Rebooting Web Of Trust

This is a workshop that takes is upon itself to reboot the web-of-trust concept in the context of self-sovereign identity networks[25]. The workshop deals with various technical and conceptual problems of how to enable web-of-trust identity systems. The relevant projects to the problem at hand are[26]:

- Decentralized Identifiers (DIDs)
- Decentralized Public Key Infrastructure (DPKI)
- JSON-LD

These projects are enablers of trust in decentralized networks. We will later dive deeper into what exactly they do, but it is important to note that they were developed with the idea and principles of SSI, hence they are deemed relevant.

3.2.2 Kantara Initiative

Kantara is a global non-profit organization that strives to give control of the data back to the users[27]. They are the people behind the User Managed Access[28] and the GDPR-ready Consent Receipt specification. The working group behind the GDPR-ready Consent Receipt specification is the Consent & Information Sharing Work Group and they aim at changing the way users consent to data disclosure. With the GDPR in place, the consent receipt standard can be used to standardize a consent format that is GDPR compliant.

3.2.3 Decentralized Identity Foundation (DIF)

The Decentralized Identity Foundation is constituted by a variety of working groups that aim to resolve issues in regards to decentralized identity[29]. One of the most notable is called "Identifiers, Names, and Discovery" which has as an objective developing papers, protocols and standards that would allow the exchange, look up and publishing of unique identifiers for peers who participate in the network. Another important working group is "Storage & Compute". They focus on the storage and management of personal data so that the identity owner retains maximum control. You can view their objective as building a decentralized identity ecosystem. One of DIF's most notable projects which are relevant to the project is the Universal Resolver[29]. Another project on which the organization is working on that is relevant for the report is DIF Identity Hub[29]. Both concepts are reviewed later in sections 3.3.2 and 3.3.3 respectively.

3.2.4 World Wide Web Consortium

W3C is a an organization made of different actors which has the task of synchronizing and developing web standards[**About W3C**, 30]. One of the most notable project that is being developed by them is called Web Authentication API[31]. Web Auth API enables a new user-friendlier way of authenticating consumers in the web ecosystem. For more details please refer to 3.3.7

3.3 Relevant Tech & Standards

This section will present technologies, standards and specification produced by entities presented in previous section. The technologies and standards presented here will elaborate more how SSI principles can be facilitated. We will also introduce the concept of decentralized networks which are later described in section 3.4.2.

3.3.1 Decentralized Identifier

Decentralized identifiers are a key concept in the SSI solutions and emerging frameworks like Sovrin, Veres One and Uport. These DID represents, as their name suggests, decentralized unique identifiers and their publishing, reading, updating and revoking is done via a Decentralized Public Key Infrastructure or DPKI. In PKI, certificates are issued by trusted third-parties and are used as form of identity proof. In DPKI there are no trusted third-parties (hence decentralized) and instead of certificates, it relies on unique global identifier strings called DIDs for short. DID is a draft specification defined by W3C [32] and it is a key-value entry of an identifier (DID) and a document (DDO) containing meta-data about the DID.

Listing 3.1: Example of DID Document [32]

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

In listing 3.1 the example of a basic schema is presented of how a DID record could look like [32]. Usually these key-value pairs would be written on a decentralized database (ledger) and can be used by peers. The DDO contains a mandatory element "@context" which has to be the same for every DID in the environment. It structures the data in a recognizable way and allows for better development of machine-to-machine communication. The next important key-value pair to notice is the "publicKey". This key specifies the public key of the owner of the DID [32]. This is very similar to the way certificates contain public keys of their respective owners. The "authentication" field is optional but it contains public keys that can be used as proof of ownership if additional authentication is required. Finally, the "serviceEndpoint" is a web address where a requesting peer can communicate with the DID owner.

On a more general note, every DID has a public-private cryptographic key pair [32]. The ownership of a DID is proven by cryptographic algorithms that relies on a secret private keys, which only the owner of the DID has. Thus, in order to ensure trust and reliability, the DID owners need to protect their private keys. Furthermore, the DID specification also supports revocation and recovery of keys which is very useful in case of security and trust compromise.

DID method specification

DIDs are designed to be used in decentralized networks as identifiers and as such they have to be published, changed, queried or deleted. Since different networks implement the specification, there will be, in theory, different implementations of the CRUD operations regarding the DID. These implementations are called method specifications [32]. At minimum the method specification must be able to define the following attributes:

- The DID method name.
- The ABNF structure of the method-specific identifier.
- How the method-specific identifier is generated or derived.
- How the CRUD operations are performed on a DID and DID document:

This format is essential for every network. The DID method name defines the namespace of the network while the ABNF specifies a structured language for bidirectional communication protocol. In addition, the specification need to define how can unique DIDs can be created and what format is supported in the CRUD operation requests.

3.3.2 Universal Resolver

This relates to the solutions/networks presented in section 3.4.2. The identity foundation have defined the standard for the universal resolver. The aim was to develop a software unit which takes a valid DID as input and resolve it to a DDO. Universal resolver is specifically developed to work for decentralized identifiers and it can work with multiple different networks[33]. It solves the problem, established in the previous section 3.3.1, of different networks having different method specifications.

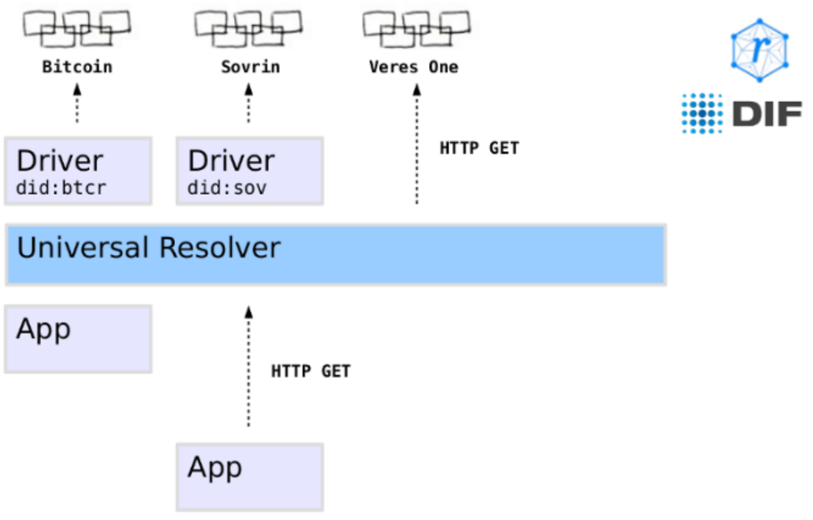


Figure 3.1: DIF's Universal resolver for DID.[33]

As seen in 3.1, the universal resolver can have different drivers for each network and use them to apply different CRUD operations[33]. The most important note is that this resolver can be used as a service by other clients to enable the use of decentralized networks without the need for the clients to directly consider network specific requirements. The Universal Resolver directly enables data formats like Verifiable credentials and protocols like DIF's Hub protocol [33] which are later discussed in this chapter.

Overall, with the development of more and more Self-Sovereign Identity networks and services using decentralized identifiers, the universal resolve becomes more and more potent.

3.3.3 DIF Identity Hub

Much of the data of the users nowadays resides in the cloud. However, there is a need to make it more secured, highly available and put under the absolute control of its respective owner. This has brought the need for an interoperable protocol, which allows easy management of digital identity credentials, recovery and management of cryptographic keys. The *Identity Hub* is a potential solution to all of these problems [34].

An Identity Hub is a concept developed by the Decentralized Identity Foundation[29]. It is important to state that the *Identity Hub* is a work in progress which is subject to change, in other words some of the requirements which will be stated in the latter parts of this section may be changed at the time of reading.

The requirements stated in the introductory paragraph of this section comprise a small fraction of the total requirements elicited so far by DIF. The idea behind the hubs is to be central tools which allow the users to securely store their data on them and have control over with whom it is shared [34]. DIF defines each hub as a data store, owned and signed by an identity, which can be made accessible by using a globally recognized API format. The following requirements have been listed so far[34]:

- **One DID to Many Hub Instances** - This requirement revolves around the idea that a single identity owner may have one or multiple instances of an Identity Hub and all of their instances must be addressable through a URI mechanism which is linked to the individual's identifier. The different instances have to sync states.
- **Syncing data between Hubs** - Different hub instances, belonging to the same identity owner, must be able to seamlessly sync data. At the time of this project a protocol which would suffice the needs for this functionality to be made possible is still in the phases of definition and selection, however, DIF states that it should be relatively easy to have this range of capabilities implemented on top of any NoSQL datastore.
- **Hub data serialization and export** - The data contained in the hubs should be easily serializable. The goal of this requirement is to allow the consumer to maintain full control over their data by giving them the freedom to migrate whenever they want.

Another requirement is that each hub needs to have a well-known endpoint, which will allow different web servers to interact with these Identity Hubs. Following the URI specification[35], DIF states that a path to the different hubs should abide the following format `:root/.identity` [34]. In the *root* will represent the path to the specific Hub. Furthermore, it is important that the different links which are representing a path to an identity owner's specific attributes do not depend on a specific hub instance. In order to achieve that DIF presents the following URI scheme `hub://did:foo:123abc/`. By utilizing this schema it will allow services to look up the different instances of an entity's hubs via that entity's DID and then access the hubs via the specific *Service Endpoints* which are contained in the identifier itself, as reviewed in section 3.3.1 (DDOs).

Furthermore, the draft of the specifications state that the process of authenticating different requests will follow the DIF/W3C DID Authentication scheme [34]. Additionally, DIF defines some specifications which apply for each respective Identity Hub's API, allowing a higher level of security and privacy to be maintained, which are also requirements set by the self-sovereign identity concept, as discussed in section 1.1.5. Moreover, the specifications outline a set of interfaces, the idea behind which is to make the common interactions concerning an individual's digital identity easier. Another important aspect which needs to be stated is that the format of the request URLs which are made towards a target's Identity Hub will not follow a REST-based schema, due to having different types of data presented under the form of strings the request URL[34]. The schema for requests is one which allows encapsulation of the requested objects which provides the necessary protection against different third party observers.

The schema used for structuring the data,as presented in the examples below, within the identity hub is specified by *Schema.org*. Furthermore, each Identity Hub has a *profile* object which describes the owner of the specific hub.

An example is given of a request format message, the type of which is about the profile of a specific entity owner.

Listing 3.2: Example of a request to the Identity Hub[34]

```
{
  iss: 'did:foo:123abc',
  aud: 'did:bar:456def',
  '@type': 'Profile/Request'
}
```

And the respective response to the aforementioned request which is defined using the *Person* schema, defined by *Schema.org*.

Listing 3.3: Example of an Identity Hub's response of arequest[34]

```
{
  '@type': 'Profile/Response',
  response: {
    requestHash: HASH_OF_REQUEST
  },
  payload: [{
    "@context": "http://schema.org",
    "@type": "Person",
    "name": "The Dude",
    "description": "That's just, like, your opinion, man.",
    "website": [
      {
        "@type": "WebSite",
        "url": "http://www.thedudelovesbowling.com/"
      }
    ],
    "address": {
      "@type": "PostalAddress",
      "streetAddress": "5227 Santa Monica Boulevard",
      "addressLocality": "Los Angeles",
      "addressRegion": "CA"
    }
  }]
}
```

However, it is important to note that DIF gives developers a degree of freedom in terms of a data schema selection by stating that the profile object should use any schema and object that will best depict the entity owning the hub.

The Identity Hub is a concept which extends the idea of a *fungible* cloud that will empower the user to navigate and steer their digital identity in a way closer to true Self-sovereign identity.

3.3.4 Linked Data, JSON-LD & Schemas

The term Linked Data, was coined by Sir Tim Berners-Lee in the year 2006[36], and the idea behind it is to create standards-based interpretable data access multiple documents and domains. It enables an application to begin at a Linked Data object and follow links within to other Linked Data.

JSON-LD

JSON-LD, which stands for JSON for Linked Data, is a lightweight syntax to produce Linked Data in JSON[37]. Compatible with JSON, it introduces minimal changes and can be used as a way to *"disambiguate keys shared among different JSON documents by mapping them to IRIs via a context"*[37]. This mean a with JSON-LD the properties in a JSON object can have a contextual meaning. In the example below a simple JSON-LD with inline context definition is listed.

Listing 3.4: Example of JSON-LD with inline context definition[37].

```
 {
  "@context": {
    "name": "http://schema.org/name",
    "image": {
      "@id": "http://schema.org/image",
      "@type": "@id"
    },
    "homepage": {
      "@id": "http://schema.org/url",
      "@type": "@id"
    }
  },
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

In above example the ’name’, ’homepage’ and ’image’ is put into context of schema.org’s definition. Meaning, the reader (human or machine) of this JSON-LD will know what these property means. This could be useful when communication between two entities happens for the first time. This could be between service provider(SP) and user, where the SP defines some required information from user - i.e. specific claims.

Schemas

As above example, using JSON-LD can be used for conveying the context of a JSON object. But when two entities are communicating and one of them dictate a context, it would be nice if the other had knowledge about that context. E.g. the sender can dictate the context of a message, but if the receiver does not know that context, the message will not make sense. So it would be beneficial if the context specified in a JSON-LD would be something must readers would understand. This is where a common defined schema is needed to explain certain contexts.

Example of a popular schema is *Schema.org*, with founding members such as Google, Microsoft, etc.[38] Schema.org is a "..collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet"[38]. As simple example of the use of the Schema.org Person schema[1] in a JSON-LD is shown below.

Listing 3.5: Example of the use of Schema.org in a JSON-LD document[38].

```
{
  "@context": "http://schema.org",
  "@type": "Person",
  "name": "George Bush",
  "disambiguatingDescription": "41st President of the United States"
      ,
  "children": {
    "@type": "Person",
    "name": "George W. Bush",
    "disambiguatingDescription": "43rd President of the United
        States"
  }
}
```

3.3.5 Decentralized PKI

The Rebooting Web of Trust organization has proposed a concept named Decentralized Public Key Infrastructure. However, in order to understand why there is a need to introduce the *decentralized* aspect, a definition of PKI needs to be done.

Communications and interactions happening in the digital domain are secured through the successful and safe exchange of *public keys*. The sender uses a recipient's public key to encrypt the necessary information, so that the recipient can decrypt the sent information by using the corresponding private key[39]. However, there is a missing element in this schematic. There is an absence of a mechanism to establish trust and proof of identity. This is the type of problems that the Public Key Infrastructure solves[40]. PKI involves the use of certificates and trusted third parties. PKI consists of different software and hardware components which allow such trusted third parties to validate the integrity and ownership of these *public keys*[41].

In a white paper provided by *Rebooting the Web Of Trust* organization they introduce the concept of *DPKI*. Furthermore, they go on to state that this decentralized approach aims to resolve the issues which have "plagued" the traditional PKI[42].

Identities belong to the entities they represent. [42]

[1]See `http://schema.org/Person`.

One of the problems which is outlined in [42] is that there is a single point of failure in the traditional PKI. An example they provide is a web hosting company which does the key management for all of their clients and save it in their own repository. This poses a potential security risk, because that repository if compromised can result in breaking the security of the websites whose keys are in that storage. Another problem associated with the traditional approach is that fraudulent certificates. Because there is a limited numbers of Certificate Authorities, CA for short, if a case occurs that such an authority is compromised it would allow for seamless Man-in-the-Middle attack[42].

However, the group of authors behind the white paper also suggest that when thinking about how to solve these problems PKI should not be abandoned, but rather it should be *decentralized*. Hence, the term Decentralized Public Key Infrastructure. By decentralizing the traditional model trust is diffused around all of the participated entities, which results in no third party can compromise the integrity of the system[42]. The decentralization of DPKI is based on key-value data stores, such as *blockchains* and other technologies. Another party involved is the - *validators*, which have the responsibility of ensuring the integrity and responsability of the decentralized ledger[42]. We will discuss in greater detail what blockchain and distribute ledgers are in section 3.3.9. For now, you can imagine them as publicly available databases that are "owned" and managed by more than one entity, hence the term decentralized.

3.3.6 Verifiable Credential

Verifiable credential is an extension of the idea of claims, presented in section 1.1. When presented with any claims, the receiver might need to have some third-party validating them in order to "trust" their origin. This references back to the idea of claim-based approach where anyone can make claims regarding his identity or others but also can provide trusted attestations,assertions, which serve as higher level of proof. In an effort to standardize this concept of claims on the Web, the W3C Verifiable Claims Working Group(VCWG) have published a 'First Public Working Draft' on this topic - with the intention of it becoming a W3C Recommendation [43].

In short a *verifiable claim* is a claim which can not be tampered with and which authorship can be cryptographically verified[43]. The ecosystem of such a verifiable claim can be seen on figure 3.2, which shows the roles within.

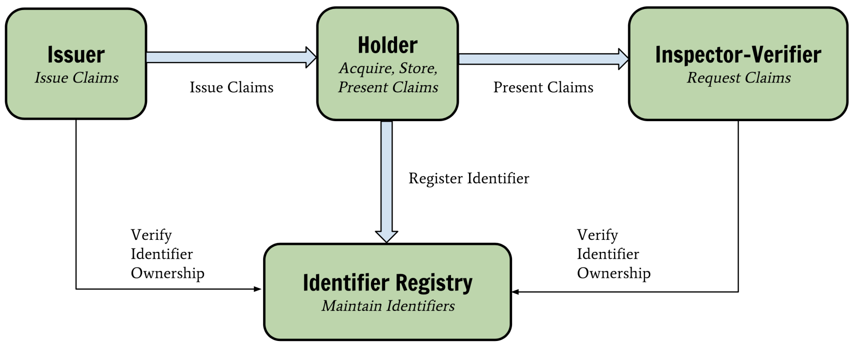


Figure 3.2: The roles and information flow of the Verifiable Credential entities [43].

On this figure the *holder* is going to be the one initiating the process of getting a claim verified by an *issuer*. The verifiable claim needs to be associated with an identifier at an *identifier registry* for it to be valid (an example of identifier is the DID). Once the ownership of the identifier is verified, the issuer can associate a claim to that identifier and return it to the holder [43]. The issuer will put a signature within the claim, making it a verified claim. The specification does not define a specific signing method to use, so this is left as a design choice. Once in possession of the verified claim, the holder can present this to an *inspector-verifier* who can validate the signature and thereby trust the claim.

Verifiable credential model

To model this concept, the issuer constructs a verifiable presentation that represents the verifiable claims for the holder. To do so, the first step is to construct an array of claims.

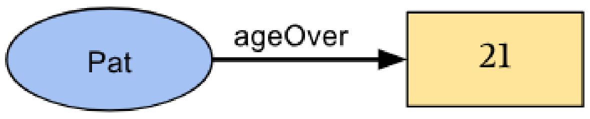


Figure 3.3: Claim structure [43]

In figure 3.3 the claim is shown [43]. It can be viewed as a key-value pair that states something about a subject (holder).

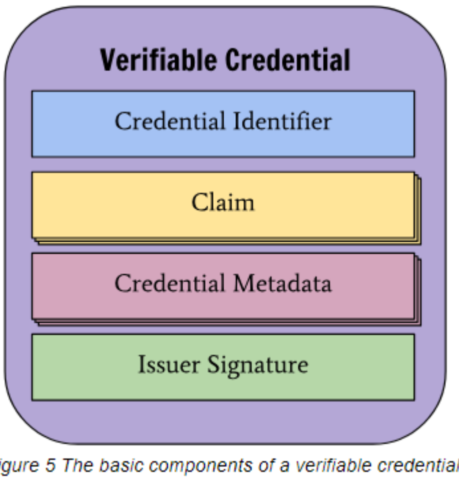


Figure 3.4: Verifiable credential structure [43]

After the claims have been identified, they are "packaged" with an identifier, signed and represented as a verifiable credential [43]. This verifiable credential can now be given to the holder and he can present it to any requesting party. The requesting party can present the credential to a verifier, who only needs to trust the issuer. The verifier then just checks the signatures. If the verifier does not know the public key to check the signature, he can use the identifier to "look it up" at the identity registry. This structure is shown in figure 3.4.

Verifiable Presentation model

If a holder needs to provide verifiable credentials from different issuers, he can combine them into a structure called Verifiable Profile [43]. This structure can contain an array of Verifiable Credentials and could be used to give flexibility to the user when he decides how to structure his identity data for a given requesting party.

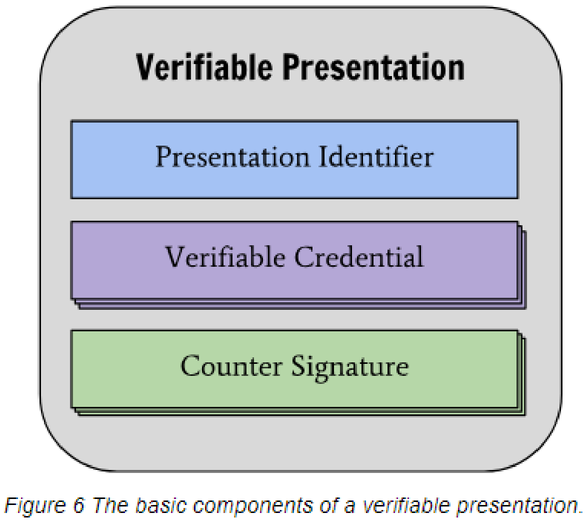


Figure 3.5: Verifiable presentation structure [43]

In 3.5 the structure is shown where similar to 3.4 the presentation structure contains an identifier and a signature. Note that the specification is work in progress, which means that changes may be present at the time of reading this project. Since writing it, the *Verifiable Presentation* was introduced to replace *Verifiable Profile*. If you encounter the term "Verifiable Profile" you can view it as the same thing as Verifiable Presentation.

Example

You can see an example of how a verifiable claim can be represented in machine-readable format:

Listing 3.6: Verifiable Credential Example[43]

```json
{
  "id": "http://example.gov/credentials/3732",
  "type": ["Credential", "ProofOfAgeCredential"],
  "issuer": "https://dmv.example.gov",
  "issued": "2010-01-01",
  "claim": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "ageOver": 21
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "creator": "https://example.com/jdoe/keys/1",
    "nonce": "c0ae1c8e-c7e7-469f-b252-86e6a0e7387e",
    "signatureValue": "BavEll0/I1zpYw8XNi1bgVg/sCneO4Jugez8RwDg/+
      MCRVpjOboDoe4SxxKjkCOvKiCHGDvc4krqi6Z1n0UfqzxGfmatCuFibcC1wps
      PRdW+gGsutPTLzvueMWmFhwYmfIFpbBu95t501+rSLHIEuujM/+PXr9Cky6Ed
      +W3JT24="
  }
```

This example is in JSON, but it stated that the specification is meant to be language independent, which also means the signature could differ. This specification essentially describes the data model and have sections discussing privacy considerations - such as signature-based correlation, storage providers and data mining, etc. Notice that proof contains all the information that is required to use this structure as verifiable credential.

Furthermore, more thorough description of roles and use-cases is provided in a ’W3C Working Group Note’ also published by VCWG. In this note details of flow, user tasks, roles, etc. are described [44].

While the specification was developed to be used by any identifiers, the Verifiable Credential ecosystem is especially useful in networks and system which rely on DIDs for identifiers. DIDs contain public-private key pairs which are ideal for cryptographically signing credentials and maintaining trust. With the development of resolvers and SSI solutions 3.4.2, Verifiable Credentials are a nice way of structuring and distributing identity data that needs attestation.

3.3.7 Web Authentication API

Web Authentication API is a new standard that defines a concept of password-less authentication on the web. Based on the idea of public-key credentials, it provides authentication of different identities to Relaying parties [45].

Web Auth. API consists of several important key terms. The identity owner/user is the one who interacts with a Relaying party. A Relaying party is the web service that provides functionality. An authenticator is a client application that acts as a middle man, which authenticates users without revealing the actual credentials to the RP. The authenticator registers and authenticates credentials by establishing a specific protocol for each action.

There are two general use-cases that are included in the flow of Web Auth. API- registration and authentication.

Registration The user visits a web site and opts to register. The site finds available authenticator and connects to it. The authenticator then prompts the user to create registration by putting his PIN or biometrics on his phone. When this is done successfully, the authenticator generates attestation and public-key credential and sends it to the web site.

Authentication The user now has a credential for the web site and decides to log in. The web site finds available authenticator, connects to it and waits. The authenticator prompts the user to choose an identity and enter PIN or biometrics to confirm authentication. Once done successfully, the authenticator sends a message to the web site and the user is now authenticated.

These basic use cases show the benefits of using such approach. There is no necessity for the user to remember any passwords or username-password combinations. He just has to operate via an authenticator UI and provide PIN or biometrics and he can easily authenticate to any service on the web. Furthermore, it decouples the credential provisioning and management from services, which enables the user to have more control.

Web Auth API can be used as an authentication protocol for an identity management systems that follows SSI principles. The password-less approach and the authenticator generated credentials, decouple the authentication process from the service provider and empowers the user.

3.3.8 Consent Receipt

Consent receipt is an emerging concept that tries to implement the principles of consent information of the GDPR. The Kantara Initiative have developed this standard to be used between consumers and services when personal data is being provided[46]. In this specification the CISWG suggests a structured way of constructing a receipt for such a consent. The recommendation mainly uses terminology and definitions as presented in *ISO/IEC29100:2011 "InformationTechnology-Securitytechniques-PrivacyFramework"*[47]. Nevertheless, it stays consistent with terms used in the GDPR. E.g. the term *PII Principal* is used, which corresponds to *data subject* in GDPR, as defined in section 3.1.1. The overall data structure of this receipt, which is proposed to use JSON formatting, can be seen in figure 3.6 Furthremore, an example of how does a CR look like in JSON format is presented in Appendix C.

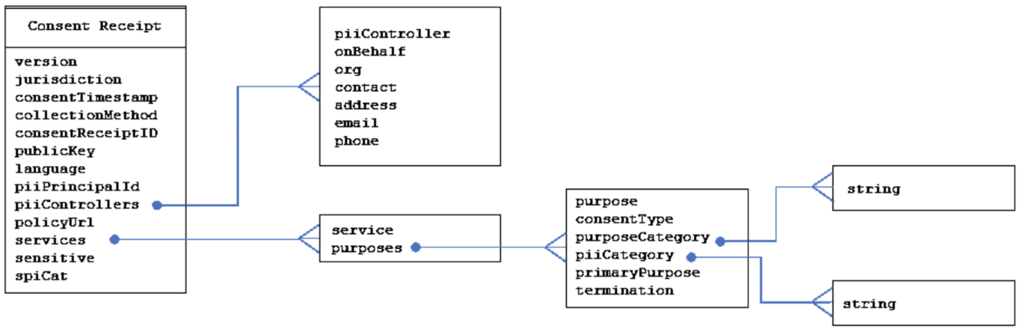


Figure 3.6: Consent Receipt data structure[46].

In the recommendation the consent receipt consists of three elements, each with multiple JSON fields as in figure 3.6 (with names and types). These elements are as follows:

Consent Receipt Transaction Fields This element consists of administrative fields, which are used as meta-data, describing the consent receipt. These fields include *language* in which the consent was obtained, *timestamp* of when consent when given, *public key* of the controller and others.

Consent Transaction Parties Fields Here the information regarding the parties involved in the consent described. Such as, *PII Principal* identifier, *controller* data and a link to *privacy policy* of the controller.

Data, Collection, and Use Fields In the third element the fields regarding the services used, represented with the *services* field. Furthermore, the different personal information categories,*piiCategory*. Additionally, the different attributes which are required, the PII and its sensitivity are also included.

The other fields which are not mentioned are just strings that are filled in by the service providers. In addition, the workings of this standard could be seen in Sovrin's concept. If consumer and service both have signed copies of a consent receipt they can use it as a proof of consent and retain legal rights in the event of dispute.

On a general note, GDPR does not specify a specific standard implementation of consent. However, using a standard as the one proposed by Kantara Initiative can help increase coherence and interoperability between many consumers and service providers.

3.3.9 Distributed Ledger Technology & Blockchain

Note, the following section does not present in specific details how exactly do distributed ledgers and blockchain technically operate and work. This is beyond the scope of this project. People use the terms interchangeably, however, that is not always the case and this section will present why[48]. Furthermore it is important to understand why they are related to SSI and what their use entails for identity management software that wish to interact with a system using them.

In general, a distributed ledger is a general term used to describe a public database where each party, or a selected bunch, have the same copy of it and each party can make updates to the ledger, thus maintaining a global database where there is not a single node owning the actual database since it is shared[49]. Blockchain is a particular version of DLT where the security, integrity and validation of the transactions added is done in a specific way - chaining blocks of data[50].

As showcased so far in this report distributed ledgers can assist in facilitating a self-sovereign identity, as presented in section 3.4.2. Blockchain has almost become synonymous with SSI, however it is important to distinguish them as separate concepts that do not necessarily intertwine. From the section DLT-enabled solutions 3.4.2 an example can be drawn of how trust and public database of identifiers can be implemented with blockchain. In that scenario, the blockchain concept is used to establish a database for identities and trust. The data once inserted to the blockchain can not be altered or changed. This could be very useful when two parties are trying to communicate and they need to resolve or provide unique identifiers that both understand. For example, in Sovrin, DIDs are saved on the ledger and whenever two peers want to know where to communicate, they exchange their DIDs, which are then looked up on the ledger and from there they are resolved to specific endpoints for exchange. This is similar to how certificates work in PKI where a certificate identifies the identity of a party, but in a more decentralized way since the ledger is not owned by a single entity, whereas certificates are issued by root authorities.

This could be particularly useful, since we can use this ledger to save, revoke and maintain identifiers. The analogy in PKI terms would be that each peer (user or service) has certificates. Furthermore, according to Sovrin, DIDs should be inexpensive and they set it as a design goal that for each transaction of identity data, new DID is generated on both sides and exchanged - pairwise identifiers. The use of pairwise identifiers provides unlinkability in the ledger domain since a service provider would have unique DID for each user and would not be able to use it in other domains to try and correlate the owner of that DID or any other data he may have provided to other domain. Since the ledger is decentralized, the identifiers registered there would not be owned by a single organization, but will be owned instead, by the data owners that created/published them. This resembles closely the idea of SSI and it is also why many of the emerging concepts rely on distributed ledgers.

3.4 State-of-the-art

In this section we introduce commercial solutions which follow partially the SSI principles. Part of these solutions are not an implementation of a complete SSI ecosystem, but instead they are implementing only some of the principles. We denote these products with the term *"Non-DLT Enabled solutions"*. This is done to distinguish them from concepts and solutions which are more closely aligned to the SSI. Solutions which fall into the latter category will be denoted as *"DLT-Enabled solutions"*. The *"DLT"* abbreviation stands for "Distributed Ledger Technology" and relates to the new idea of using DLT and blockchain to achieve SSI. Essentially, instead of having an organization storing and maintaining the identities, we can use distributed ledgers which are more decentralized. The projects and specification presented in the previous section 3.2 are all referenced here and essential, in order to understand the presented concepts.

3.4.1 Non-DLT Enabled Solutions

In this section the current solutions regarding identity management in a "traditional" sense is presented. By traditional is meant an IdP which assist in providing users identity to other service providers, not using a distributed ledger. These solutions showcase a way of implementing the core of SSI into the currently existing model of trust.

DigiMe

One of the more popular solutions is *digi.me*. The product supports both mobile and desktop versions. The initial idea embedded into the solution was to be a social media back up. However ,it has evolved throughout the years and now it allows users to gather all of their personal information from different sources scattered around the web in a single repository [51]. The way that their system is structured is such that it does not handle any of the information of its users. From there on the users can share their information with different service providers.

In other words, the product is a platform which grants the user the ability to take control of managing their own personal data and who has access to it, and under what conditions. Furthermore, this way of placing the data on its respective owners premises is empowering the users and at the same time can increase the quality of the services they receive due to the common base on which the data is structured.

The way the system works is when a user wants to register on the platform they are prompted to select in what repository would they like to have their personal data to be stored. At the moment of this review the only two possible options are Dropbox and Google Drive, however, the creators of the solution are promising more to come. Once the choice of repository has been made the user is prompted to select from where do they want their data to be gathered. At this specific point in time, the user is presented with a list of the different possible data sources, which include the popular social media platforms such as Facebook, Twitter and Instagram [52]. Other options on the list consist of sources from which other types of data, besides social, can be extracted - financial and health. Which comes from institutions that have a mutual agreement with digi.me. Some of the more popular companies are Fitbit, Spotify, VISA, Master Card [52]. Once the user have gone through all of the aforementioned steps, the gathered data is encrypted on the selected user's device or storage and can be accessed only through a set of credentials explicitly set by the user - a username and password of their own choice [53].

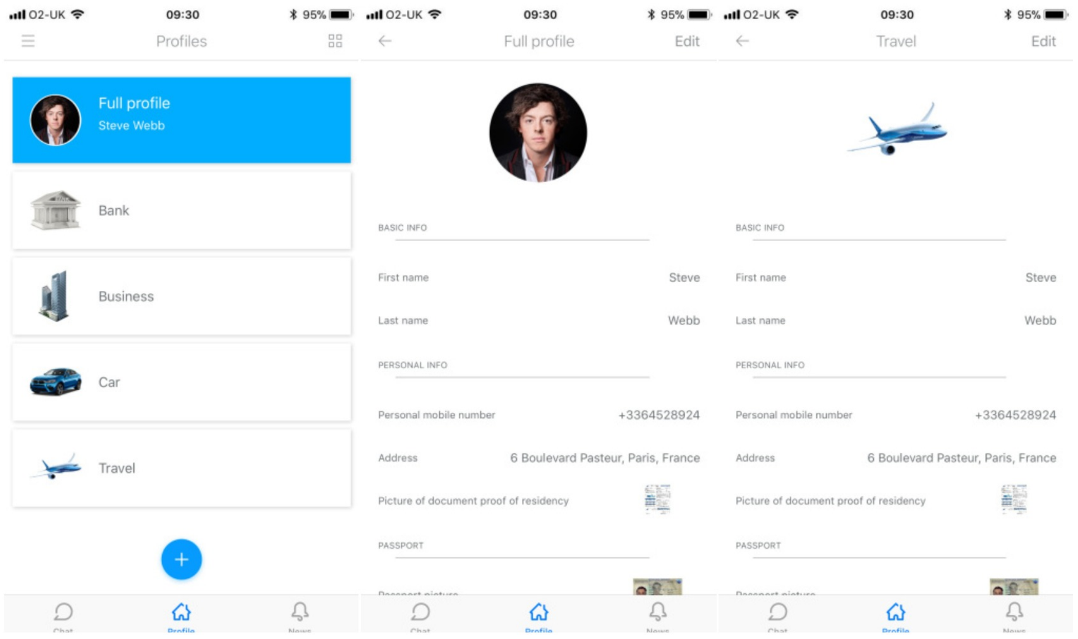
Another aspect which the developers of the system pride themselves with is that digi.me does not touch any of the users data. It lays down an architecture which connects their users' private information spread across domains and allow different search and processing functions to be performed on it. Additionally, by synthesizing their data in a single repository users can then selectively deliver it to other service providers and also get information about what value do they get out of it by doing so. Furthermore, the users are able to see what the company to which they have provided information will get out of it, what type of processing they will do with it and what will the users receive in return for this exchange. Accordingly with the GDPR, discussed in section 3.1.1 the service provider also has to give an explanation of what type of data they will retain, if any, and why. Additionally, they have to specifically specify if the service provider supports the *"Right to be forgotten"* [51].

Furthermore, digi.me is one of the two companies, alongside iWelcome, that have launched the Kantara Initiative Consent Management Work Group [54]. The relevance of this group will later be explained as consent management is important part of achieving SSI.

Chekk

Chekk is the second solution, which falls into the non-DLT enabled category, that we have reviewed. Developed by Chekk Limited, the platform allows better management of personal data both for users and businesses. The Chekk product platform comes in two versions - a web platform, for the businesses, and a mobile app for the individual users. The Chekk mobile application acts as a mobile data wallet which allows users to control what data they share. The web portal is used by businesses to make requests and for better interaction with their customers.

A user can download the mobile app, which at the time of reviewing it is only available on the app store for iOS devices, there is no Android version. Afterwards they need to register and input their personal information, which is denoted as a *Full profile* within the context of the application. Afterwards the user can create other profiles containing context-specific information, meaning they can select data from their main profile to be put into this specific context-dependent profile and additionally add more information. In the context of the system these situational profiles are known as *cards* - examples are a card containing relevant information about travelling, about banking and so forth.[55] The user is able to interact with businesses which support the Chekk platform. Additionally, they can chat with them if any exchange of information is needed. Furthermore the user can receive alerts, news, statements, different offers and even rewards by businesses when sharing data with them [55]. Once the users have input all of their data into their wallet, it is encrypted at a user and data level, meaning the data is encrypted with a key only the users knows and the data itself is also hashed.



BASIC INFO

First name Steve

Last name Webb

PERSONAL INFO

Personal mobile number +3364528924

Address 6 Boulevard Pasteur, Paris, France

Picture of document proof of residency

PASSPORT

Passport picture

Figure 3.7: Chekk User Application [55]

When a business would like to exchange data with its customers via Chekk, the company offers a web portal that eases the communication between its users and themselves - allowing the business to share different news, send messages and access the latest up-to-date data of its customers[55]. Another functionality of the web portal is that the specific business can "grant" a role to their employees to their specific Chekk app which instead of acting as a personal data wallet to each staff member, the app can be used as a medium to allow communication between those staff members - e.g. relationship managers or sales department, to interact with the customers. Additionally, Chekk provides a set of APIs which allows businesses to perform data requests [**Businesses**, 55]. An example of the graphical interface of the business version of the application is shown in 3.8.

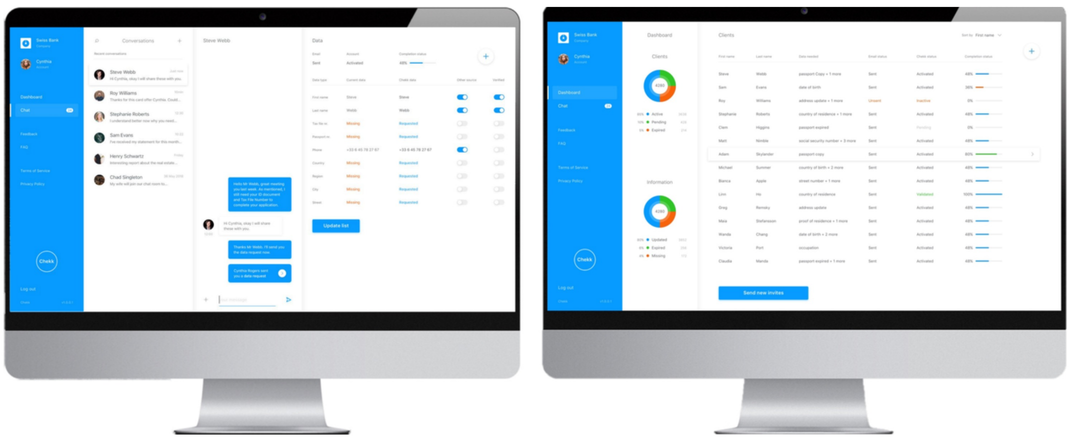


Figure 3.8: Chekk.me Dashboard Manager for Businesses [**Businesses**, 55]

While these solutions empower their users, they still have their limits. For example, Chekk allows users to distribute data to selected parties, i.e. services, but the only way for this to happen is if the services also use the Chekk platform. These prerequisites create a closed environment. Digi.Me has a similar problem. Both solutions empower the user, but instead of acting as an agent for the user identities they are brokers. This means that in order for anyone to make use of their functionalities, they have to become part of their ecosystem. This is also the major issue that is solved by the DLT enabled solutions.

3.4.2 DLT-Enabled Solutions

Solutions presented in the following sections make use of distributed ledger and blockchain technologies to enable the concept of Self-Sovereign Identity to the users. Note that unlike the non-DLT enabled solutions, these are products that are still under development and are yet to be completed or tested in the real world. The presented description of them is based on different documentations and recommendations which provide a proof-of-concept explanation. While the Non-DLT solutions are commercial products, the introduced DLT enabled solutions are more close kin to an ecosystem or a network which supports the concept of Self-Sovereing Identity. Futhrermore, these solutions can showcase how the shortcomings of Non-DLT solutions can be solved.

Sovrin

Sovrin is an open source project governed by the Sovrin foundation group. Sovrin defines a framework for building decentralized identity network with trust using blockchain and public ledgers technology [56]. In general, the concept enables users to achieve a Self-Sovereign Identity. In other words, the platform allow them to directly control the creation, provisioning and revocation of their identities.

How does Sovrin work?

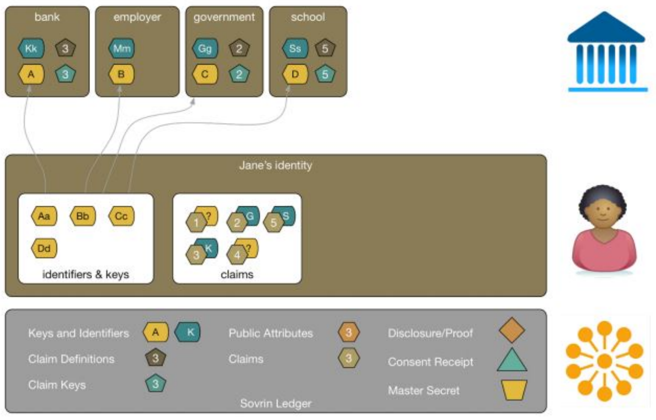


Figure 3.9: Sovrin virtual representation [57]

In figure 3.9 a virtual representation of how conceptually the Sovrin framework works is illustrated. At the bottom of the figure next to the Sovrin logo, the public Sovrin ledger is represented. Ledgers can be seen as databases and public ledgers as publicly readable databases. The ledger itself can be accessed by anyone and is used to keep global, immutable evidence of "identity records". In the figure, different types of records have been depicted in the form of a legend. It is important to clarify that no personal information is stored there [57].

In the middle of the figure, the actor Jane and her identity storage are illustrated. There she keeps her own and other services' identifiers. In addition she can store there her own identity claims, proofs, consent receipts and others [57]. This can be viewed as a private ledger to which only Jane has access to. Note, that for each entity different identifiers and keys are generated. This provides protection for Jane's identity across platforms since the identifier that the bank has can not be linked to her identity by the government or any other external entity. The identifiers are based on the DID standard.

Next to the identifiers, different claims are shown. The claims colored in yellow are asserted by Jane and the ones colored in blue are asserted by other entities. When Jane wants to connect to new service and provide claims she can use the already existing ones. The important point here is that if the service requires that she is over 18 and it would only trust the a government entity, Jane can provide this claim signed by the government and meet the trust requirements [57]. Of course, sometimes information is too sensitive to share. For cases like this, Sovrin facilitates zero-knowledge proofs [56]. For example, if the aforementioned service wants to just confirm that Jane is over 18 and does not care about her specific age, Jane can create a proof request saying "I am over 18". This request is then sent to the government, which validates and signs it, and then present the proof to the service [57]. The service can then validate the signature.

In addition to this, for each claim exchange between Jane and a service, a consent receipt is issued and written to the public ledger. This allows for proof of consent if such is desired.

The framework provides a powerful way for any individual user to control identity provisioning and removes the problem of "scattered identities" across the web, discussed earlier in section 1.1

Sovrin in-depth

One of the key issues that Sovrin has to solve is trust. In nowadays web, trust is established in centralized form where trusted third-parties (certificate authorities) issue certificates which are trusted by vendors and clients across the network. Certificates act as identifiers and provide a form of identity authentication.

In Sovrin, the trust is build upon several core principles[58]:

- *Independence* and *Self-governance* - the identity owner has complete and irrefutable ownership of their Sovrin identity.
- *Guardianship* - allows for identity owners to outsource control to a trusted party, which can be referred to as a guardian. For example, elderly people can delegate rights to their children to have control over their identity.
- *Diffuse trust* - trust is shared among participants and is not given to a single entity.
- *Web of trust* - where nodes with specific duties shall not be selected hierarchically, but with peer-to-peer connections (this will become more clear later)
- *Security, Privacy , Diversity by design* and others

Using the aforementioned principles as an underpinning, the Sovrin foundation has created a framework where a public ledger will contain unique identifiers for every identity. These identifiers are in the form of DID and are designed in a specific structure:

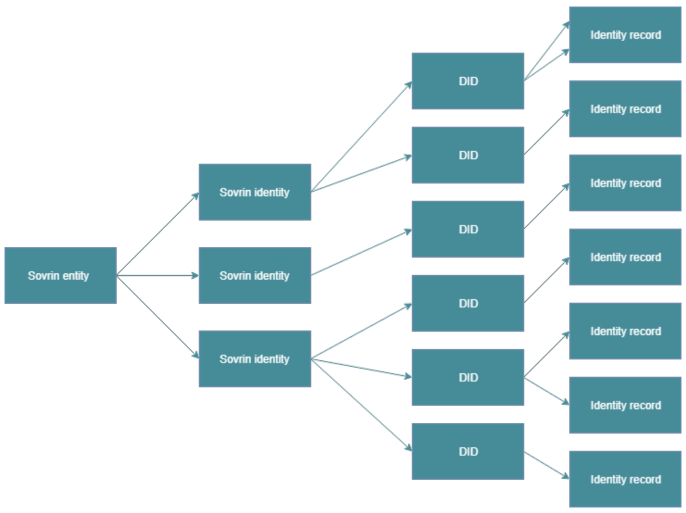


Figure 3.10: Sovrin identity structure [58]

Figure 3.10 illustrates the structure of identities. The Sovrin entity can be either a *"thing"* as in IoT or it can be an identity owner, either private or corporate [58]. Each entity can create multiple Sovrin identities which are an analogy to partial identities. For example, a person is an entity but can have a Sovrin identity when he works as an employee, another one as a husband and so on. Each of these identities can have multiple identifiers and each identifier corresponds to a record on the ledger [58]. At first glance this may seem counter-intuitive but there is a very important concept here. Whenever an entity establishes connection to other entity, both of them create new DIDs for the specific connection [56]. This means that a single identity owner can use the same Sovrin identity to communicate to one or more services or other peers. Each connection to a peer requires a new identifier, hence a Sovrin identity can have multiple DIDs. Furthermore, when creating a new DID it is not possible to trace their ownership to a single entity. This prevents cross-context identification. For example, if a user has provided two DIDs - one to service "A" and one to service "B", if service "A" takes the DID that has been given to it and takes it to a bank (which is service "B" in that context), service "A" will not be able to correlate the DID it has received to the DID the bank has, even though both of them are owned by the same user.

Note here, identity records and DIDs are stored on the public ledger, while entity information is stored on the private ledger, which corresponds to Jane's identity wallet from the example in figure 3.9. This raises the question of who is responsible for registering DIDs to the ledger? In Sovrin there are special nodes on the network called *Stewards* [58]. These entities have the responsibility to assure the validity of different identity transactions and what is written to the ledger. In addition there are Trust anchors which are organizations or individuals who have show sufficient public evidence about their trustworthiness and accountability that fulfills a level of trust. Whenever a new individual wants to register to the network, his first-time provisioning has to be done through a Trust anchor [56]. The Trust anchor sends a request to the Steward to register a new DID. The DID can be either be blinded or a verinym [58]. A blinded DID can either be an anonym or a pseudonym. This means that the actual DID cannot be directly correlated to a legal or real identity. This ensures protection of the privacy of peers. However, in cases in which an assertion to a real identity, either a legal or an individual, is required, a verinym is created. A verinym is signed by a *Trust anchor* and can directly or indirectly identify the owner of the DID. It is important to note that when a new DID is registered it is defined as a blinded type. This means that the verinyms are not the default option. However, they are important for the Trust anchors in case of legal requirements.

So far we have identified how new identities are registered and used, but how is the user going to create and manage them? According to Sovrin each new DID has to relate to its own endpoint where peer-to-peer communication can be established. If a Sovrin identity has many DIDs, all of which have the same endpoint, then the concept of privacy and unlinkability can be compromised, since multiple DIDs can be linked together. This type of scenario should be used by public organizations which want to expose their public identities.

To allow users to interface with the network, Sovrin has defined a *Sovrin client component* and *Sovrin Agents* [58]. If the user wants to directly interact with the ledger, he can use a software that runs a Sovrin client component and connect to the network. However, if he wants to allocate more power to his client, he can use an agency which runs an agent on their behalf, interacting with the network [58]. Some of the pros of agents are higher availability, easier maintenance of data, arguably higher security and less user involvement. However, the drawbacks are one more entity that has to be trusted and possible loss of privacy control. For example, a cloud agent can be configured with specific endpoints and it can communicate with requesting services without involving the user, since these processes could be delegated to a server. But if the user prefers own client running locally on his mobile phone, then whenever a request comes from a service, he has to be able to ensure availability of his device to the network and also be involved in the process. Furthermore, this opens the possibility of tracking and mapping DIDs to internet addresses (if the user resides exclusively on single network). In the cloud agent, the client identity is protected by the agent, since requesting parties communicate with agents and not clients directly.

You can view the agent as a Sovrin client with more power which can act in some use cases on behalf of the user and provide certain abstraction for the user on the network.

In general, Sovrin can be viewed as an ecosystem that enables the facilitation of SSI. It provide the conceptual framework necessary to create identity management systems that want to comply with SSI principles.

Evernym

Evernym is another company which is a founding member and developer of the self-sovereign identity with verifiable claims concept[59]. Additionally the creators of the platform are the same people who are the founders the Sovrin network, described in section 3.4.2. They provide different solutions which allow people to not only *rent* their digital identity, but actually *own* it. They provide different solutions for each type of customers, either is it an individual or a whole enterprise.

The version of their solution which is targeted at individual users will review is *Connect.me*. At the time of the review the product is still in Beta phase[60]. The product is a mobile application which the individual user downloads to their phone and allows users to get different credentials specific to the context of their use.

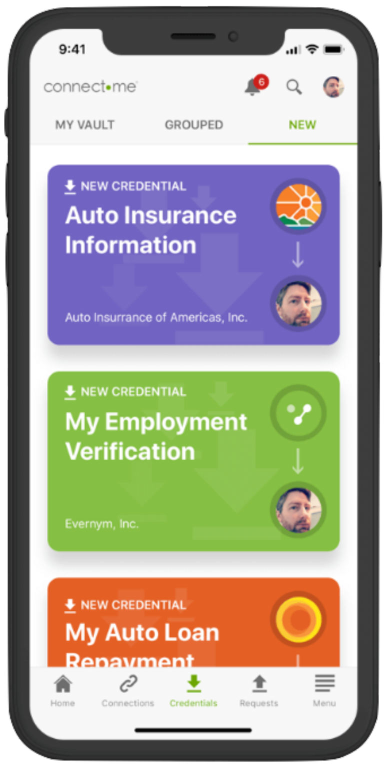


Figure 3.11: ConnectMe credential views[60].

An example of the functionality the application provides is the user can have his boarding pass and password verified by the airport authorities automatically without even them taking out the phone out of their pocket[60]. Another use case is the user can apply for a mortgage and have all of their information accurately and securely disclosed. Additionally, the product provides the option of alerting the users when suspicious transactions are about to be executed and can block them entirely[60]. The creators of the solution do not own any of the individual's information and they state that the data is strongly secured with elliptic curve cryptography[60].

The Veres One Project

Another project under development which is categorized as DLT-enabled solution is The Veres One Project ("Veres One")[61]. Their vision is very similar to that of Sovrin - to enable individuals of establishing full control over their data and whom they share it with. Veres One has a three layer architecture:

Decentralized Identifiers DIDs is the fundamental standard that will enable self-administered identifiers - for device, person or organization.

Verifiable Credentials This layer is about storing and controlling one's digital credentials. Being passport, driver license and other third-party issued credentials.

Decentralized Identity Management The actual software application, to enable managing to who and which credentials should be shared under which circumstances - including a digital wallet.

At the time of writing, it is stated that the goal of Veres One is to provide the *decentralized identifier* layer. For this they have built a fit-for-purpose blockchain, which they claim is more cost-efficient and privacy enhancing compared to other blockchains used for identity management. It is a public and permission-less blockchain (accessible by the public), which means everybody can run a *node* and provide computational/storage resources to the network. I.e. validate blocks appended on the blockchain. There is going to be an initial fee for creating a DID, which at this point is set to around 1$. Beside nodes the Veres One Network will consist of *maintainers* who maintain and update the Veres One software, *board of governors* which is responsible of the governance[61] and finally *accelerators* which enable users to create DIDs fast and efficient. This ecosystem can be seen on figure 3.12.

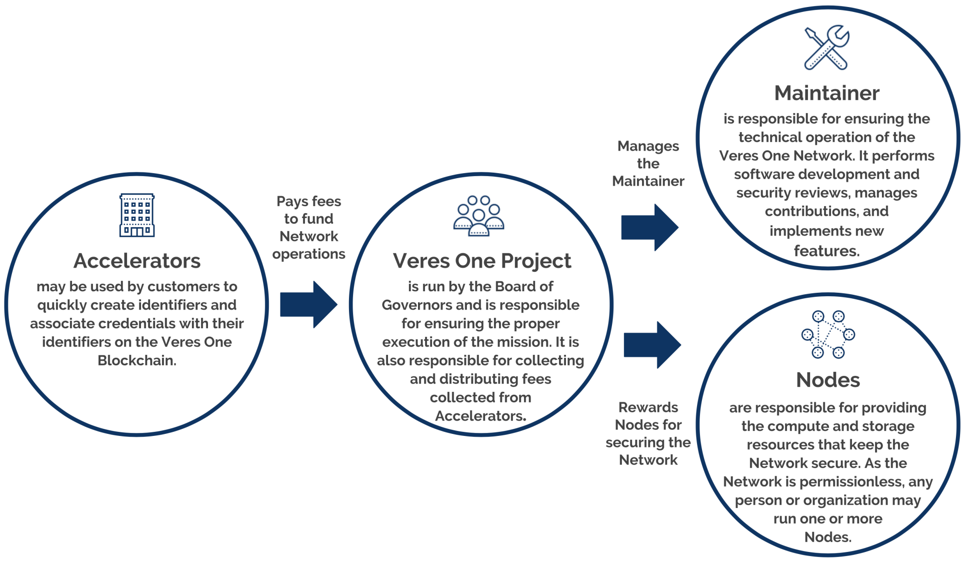

Accelerators
may be used by customers to quickly create identifiers and associate credentials with their identifiers on the Veres One Blockchain.
Pays fees to fund Network operations
Veres One Project
is run by the Board of Governors and is responsible for ensuring the proper execution of the mission. It is also responsible for collecting and distributing fees collected from Accelerators.
Manages the Maintainer
Maintainer
is responsible for ensuring the technical operation of the Veres One Network. It performs software development and security reviews, manages contributions, and implements new features.
Rewards Nodes for securing the Network
Nodes
are responsible for providing the compute and storage resources that keep the Network secure. As the Network is permissionless, any person or organization may run one or more Nodes.

Figure 3.12: Veres One ecosystem [61]

The specification also defines how the nodes on the network must provide endpoints for accessing DID Documents. It also presents how creating and updating of DIDs should happen.

It should be noted the the Veres One Project is very much under development and is said to be production ready of June 2018[62].

uPort

uPort is a standard and a platofrm that has been developed to provide identity layer for decentralized networks. Furthermore, uPort defines a standard of protocols to build the identity layer and has been implemented to work with Etherium network. uPort allows developers which create dApps, which stands for decentralized Applications, to offer their consumers an improved privacy. I.e. uPort is a privacy preserving identity system for Ethereum dApps[63]. The solution follows several principles[63]:

Store User Data *Off-Chain* to Combat Permanence uPort defines that user data should be stored in an off-chain manner. This is due to the fact that the blockchain is both public and permanent[63]. uPort suggest that user data should be stored on a user-managed vault, or a private Identity Hub, or both[63]. Furthermore, each user is offered a backup service, named Caleuche, which in essence is allowing the users to upload their *symmetrically-encrypted identity data* to a server. This is used as a fallback mechanisms in case the user loses their smartphone and would like to restore their identity data, however, if the want they can opt out of the option and not use it[63].

Create new user accounts for each dApp uPort promotes the idea that the users should use different dApp-specific accounts on its identity system[63]. This would allow the users to eliminate the correlation factor which originates from the use of the blockchain. By having this safeguard it, called multi-identity architecture, it would make it much more difficult for malicious actors to track an individual user by just analyzing the blockchain[63].

Additionally, uPort contains its own implementation of DIDs and provides a way to use uPort identifiers across different Etherium networks. Their implementation of the DID is called *Multi-Network Identifier*[64]. The adjective *Multi-Network* is derived from that the uPort identifier works with multiple ethereum blockchains[64]. Furthermore, in the specifications it is stated that this type of identifier can have its use be extended to other networks which are even non-blockchain[65].

3.4.3 Summary

To conclude this section, we are going to point out several important take-away points. The first is that in a truly self-sovereign system the user must have full control over his data, as shown in "Digi.me". The idea of outsourcing the data storage to the user's preference is very interesting and very important. This is a desirable functionality for any client software that deals with user identities and data. Second, both "Connect.me" and "Chekk" suggest of using a base identity as a generator for "cards" or other sub-identities. This a useful way of disclosing only what is necessary from one's identity to requesting parties. Third, "Chekk", introduces the idea of providing consumer data to services. While their implementation is designed with the idea of "Chekk" being a broker, this idea could be taken into the dimension of decentralized systems and create brokers who only acts as a bridge between the client and service and be more privacy-friendly towards the user. Finally, Sovrin, Veres One and Uport provide the fundamental backbone of a truly SSI system. Understanding how this backbone works and what standards and mechanisms are involved allows us to identify possible use cases and try and build a better identity management solution that could utilize the benefits of it.

3.5 Conclusion

To conclude this chapter we will summarize important points that will be used further in the analysis and design to answer the problem statement.

We looked into state-of-the-art products and how and why they were created. With the emerging concept of SSI and its integration into Distributed Ledger technology, different organizations have begun collaborating and developing standards that can provide the user with full control over his identity provisioning. In regards to the problem formulation at hand, we can see the Identity management tool as a software that interacts with such distributed networks like Sovrin, Veres one and Uport is a possible concept that should be explored further. We have examined important protocols and standards that provide the ability for anyone to register, operate, distribute and control identifiers and identities.

To develop a truly SSI identity management solution, we take the presented concepts, ideas and protocols as a stepping stone that will be used in the analysis to design and create an agent that can be used as an answer to the problem formulation. It is important to understand that this chapter servers as an inspiration and a starting point that allows us to understand what are the current trends and how other groups of people see the future of Self-Sovereign Identity.

In the following chapter we will analyze in greater detail the information stated here and will reflect it back to the problem definition in order to come to some specific requirements that should outline a potential solution to our problem.

Chapter 4

Analysis

In this chapter we will reflect on the problem definition and the sub-problems identified in section 1.2. The process stars with introducing the definition of what we consider a decentralized identity management solution. Afterwards we define three scenarios that were used as a root starting point for the analysis. To improve the process, we opt for UML diagrams and investigate in detail each of the scenarios. The outcome of the analysis is a core requirements and specification list that identifies requirements that need to be met, in order to create a DIdMS that supports the scenarios.

4.1 Decentralized Identity Management System

As form of introduction to the analysis, we will present the concept of a decentralized identity management solution. The term is based on the research chapter of this project. There are core assumptions and constraints that are taken into consideration and they will be expanded upon.

From section 3.4.2 we have seen the development of new SSI ecosystems that are dependent on DLT and Blockchain. The workings and operation of these systems is still work in progress, but as specified in Sovrin, section 3.4.2, there will be a need for user agents and client software that would interact with the ledger and serve as a bridge between the user and the network. From the perspective of an identity management solution, such networks are enablers. What the Sovrin people call an Agent is a kin, in terms of functionality, to what an identity management solution system should be able to do. With that said, we view the development of an identity management solution as part of these new SSI ecosystems and will consider it in such context in the coming sections of the analysis. The main reason for this choice is that these networks strongly support the SSI principles, defined in section 1.1.5, which are part of our objectives when finding a solution to the problem statement. To reference that we use it in that context we will refer to the identity management solution as *DIdMS - **D**ecentralized **Id**entity **M**anagement **S**ystem.* Note, that the DIdMS, should not be constricted to using only a specific network, instead it is considered to interact with different DLT-enabled SSI networks and provide seamless interaction with them from the user perspective.

Based on the above introduction of a DIdMS and material presented in the research chapter 3, the context of the imagined DIdMS can be seen in figure 4.1.

The system is put into context in terms of the terminators and interactions with them.

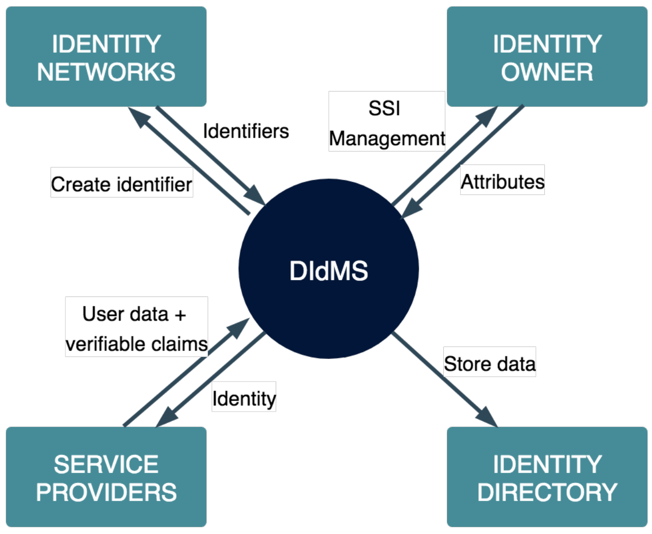


Figure 4.1: Context diagram of the DIdMS.

Identity Owner This is a natural person who wants to manage his SSI(s) and its attributes in a single place. The platform will enable this, by being the broker between the entities needed for that.

Service Providers (SP) These will be services who provide services to the identity owners and may need their identity (or some part of it) to provide a service. This can be both private and public/governmental organizations, as long as they provide some sort of service to the user. E.g. Netflix providing movie streaming, universities providing diplomas, government providing driver licenses, etc. The DIdMS will need to be able to provide the necessary identity to the SP, defined by the user. This is similar to the idea of Chekk, section 3.4.1, but in a more decentralized network. This also references back to our scope of facilitating user-service identity exchange.

Identity Directory A directory in which the user's actual identity (attributes, claims, general data, etc.) is stored. This storage should happen on the user's specified option, e.g. storing on Google Drive, Dropbox or local device. The platform will need to be able to read and write to the directory. This is similar to Digi.me 3.4.1 and their way of empowering the user by letting him decide where to store his data.

Identity Networks These are the networks and ecosystems described by the DLT-enabled SSI section in the research 3.4.2. As these networks are based on DID and the use of cryptography keys, the DIdMS must facilitate these interactions on behalf of the user.

Having an identity management solution working on top of DLT-enabled networks provides a solution to our sub-problem 1. The introduction of public ledgers and DIDs as part of the DLT-enabled ecosystem, means that many of the SSI principles are met. The user has control over his identifiers, he is not dependent on other entities and he is in full control over the distribution, provision and creation of them.

Due to the workings of DLT-enabled SSI networks, a core part of the DIdMS is the management of DIDs. In addition to that, the interaction between a service and a consumer constitutes additional requirements that need to be addressed. In the following sections we will outline some of these core business requirements for the DIdMS.

4.1.1 Support of Multiple DID Networks

The solutions presented in section 3.4.2 are networks and ecosystems trying to facilitate more complete SSI management. However, they differentiate in how they establish trust in their networks. For example, Sovrin is permissioned (only certain participants maintain the ledger), while the other two are permissionsless (everybody contributed to the maintenance of the ledger). Other differences could be found in the way they define the interaction with the network. Overall, we can see that on top of supporting DIDs and other core feature, each network has its own interface requirements.

In order to follow two of the SSI principles *interoperability* and *existence*, the DIdMS should be able to support multiple networks. In other words, the user should not be constrained to using only Sovrin or only Veres One. He should be able to use any supported system he likes. Depending on the network, the DIdMS would have differently defined role. In terms of Sovrin framework the DIdMS would be an Agent and in Veres One it is a facilitator on the decentralized identity management layer. Nonetheless, it is necessary for the DIdMS to support the requirements for any role that the DIdMS would take within these frameworks. In figure 4.2 the DIdMS is depicted acting as a unified interface towards the identity owner, being able to operate on multiple networks.

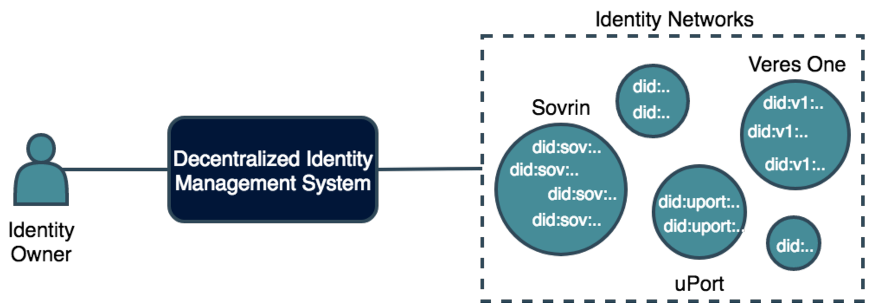


Figure 4.2: A DIdMS supporting multiple identity networks.

Here you will see the identity owner using the DIdMS as a single point of interaction, but still being able to operate on multiple networks. This will enable the the identity owner to generate identities on any required identity network. No matter where the identity owner is, the generation of a DID, it will happen the same way for him. This idea of supporting multiple networks is already being investigated and work have proposed tools to assist, *identity hub*(section 3.3.3) and *universal resolver*(section 3.3.2).

4.1.2 Providing Identity to Service Providers

A main functionality of an any IdP is to provide the identity of a user to a service provider(SP). The content of such an identity and the role of IdP are presented in section 1.1. As a modern DIdMS, this functionality must also be supported. In that regard there are three main areas considered essential for a DIdMS, described below.

Assertions & Verifiable Claims

One of the major changes introduced by the use of DLT-enabled SSI networks is that the DIdMS does not (necessarily) provide assertions, as it would be in the usual IdP framework. As an underlying goal of the identity networks the assertions or verifiable claims can be issued by any identity on the network. This means that the verifier will decide weather it trusts the issuer(s). Therefore is seems essential that the DIdMS must assist the identity owner to get issued correct verifiable claims needed by the verifier/SP. This translates to the concept of claim aggregation that Sovrin proposes 3.4.2.

Collection and publishing user data

As part of identity data transaction, the user needs to collect his data, "package it" and send it to the service provider. Instead of this happening in the domain of the service provider, the DIdMS could facilitate this interaction. This could improve the user experience since he will be able to reference his existing data and use it when creating new data transactions. In addition, this allows the user to manage more easily each relationship he has with the service providers. Once the data is gathered and ready for the service provider, the DIdMS must also be able to send this data or notify the service provider where it can be queried.

Up-to-date attribute values

Another functionality of providing identity owners data to SPs, should be to ensure that data is always up-to-date. And furthermore provide a single place for the identity owner to update such data. Since the DIdMS enables the collection and publishing of the data, it must be also responsible to updated it. The updated data can then be requested from the DIdMS when service providers need it.

4.1.3 Consent Management

Due to the scope of consumer-service interaction, the DIdMS must also enable support for consent management. As part of the identity transactions, consent management has to be supported in the form of presenting consent from the service provider to the user. The user then has to accept the consent and only then can the transaction take place. The DIdMS must facilitate these actions within its domain and also store different consents that the user has given to the service providers. This enables accountability on both parties and servers as evidence that necessary consent is given where it is legally required.

So far we have introduced a basic conceptual model for a DIdMS. The discussion, presented in this section, outlines general considerations that are part of the development of a DIdMS. These considerations are also translated into scenarios presented in the following section. As a solution to our problem statement, we consider the approach of identifying requirements for DIdMS. As DLT-enabled networks already support SSI, the development of DIdMS that interacts with such networks is seen as a missing step towards adoption of complete Self-sovereign identity.

4.2 User scenarios & Use Cases

In this section the DIdMS is described in relation to user scenarios and use cases. By reading this section, you should have better understanding of what functionality a DIdMS will provide to the identity owner.

4.2.1 Scenarios

It is impossible to capture all of the required scenarios that a DIdMS should support. In this section we will present only scenarios that are seen as relevant and most important in regards to the problem field. In order to reference the DIdMS in the scenarios, we use the term *didmssolution.com*.

Scenario: 1 (Registration)

Jane wants her identity in a single place. Jane decides to try out didmssolution.com, a DIdMS to manage her identity, which she has heard about from a close friend. Jane picks up her phone and downloads the app. She presses create account and the creation process begins. Jane first have to agree to the terms and service agreement, before proceeding. She is then prompted where she would like to store her data - my personal device/ Google Drive/ Dropbox or some other cloud storage service. She decides to use her Google Drive, which means she is prompted to give didmssolution.com access to her Google Drive storage. After selecting storage she opts how to she wants to authenticate towards didmssolution.com and enters a username and password. After this she has the option to enter some basic information about herself. She sees it is possible to import some attributes from Facebook and decides to do so. From Facebook she can get her name and birthday, but she enters her address manually. After that she has successfully registered at didmssolution.com and can now log in.

Scenario: 2 (Request service)

Jane has created her account in didmssolution.com and now decides to try it out. She loves watching movies so she goes to a popular streaming platform - Netflix.com. She applies for registration and gets informed of Netflix's DID to use for providing her identity. Taking that DID, Jane goes to didmssolution.com and selects to register a new service provider. Prompted to input the SP's DID, Jane have copied Netflix's DID and now pastes that to the input box. Proceeding, she is now shown a consent specifying what information Netflix needs, as well as how they will process and handle her data. Accepting this, Jane has to select which of her defined attributes she wants to provide to Netflix. She then is prompted to choose one of her existing identities. After selecting her base identity, Jane is then redirected back to Netflix where her registration is finalized. Jane can now fully use the movie streaming service with the identity data she just provided.

Scenario: 3 (Manage identity)

After awhile of using didmssolution.com, Jane now has multiple attributes defined and has registered multiple service providers using didmssolution.com. Jane is moving to new address which means she needs to update her address for all of her service providers which use it for shipping. She goes to didmssolution.com and goes to see all of her defined identities. She selects to change her attribute regarding her address in her base identity and inputs the new address. She is prompted to confirm and the system then updates all other identities (given to SP) that are generated or used by that base identity. After the update is completed Jane navigates to the section regarding her current SPs and the relationships with them. She decides that in addition to the update she would no longer like the use Netflix.com as a service. She presses a button to revoke the consent for the SP, breaking the relationship with Netflix. In the process she has the option of requesting to be forgotten, which she opts for. The DIdMS informs Netflix.com and updates the state. Jane has now successfully updated her identity address and revoked consent for identity transaction to Netflix.com

4.2.2 Use Cases

Use cases are an UML tool that allows us to present how core functionalities could be facilitated in the domain of the system. To get a better understanding of the functionality of the platform, use case diagrams have been constructed. These are primarily based on the scenarios presented in section 4.2.1.

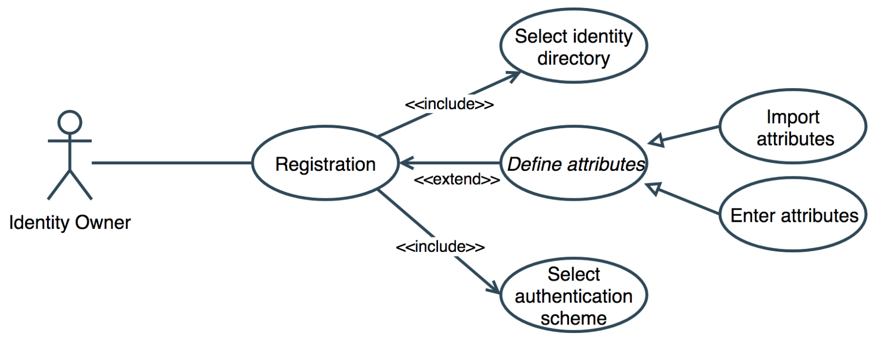


Figure 4.3: Use case diagram of *Registration*

The first use case diagram seen on figure 4.3 depicts identified use cases based on scenario one presented in section 4.2.1. The actor *Identity owner* performs use case *Registration* which is dependent (includes) two others. The actor must *Select identity directory* to specify where his data should be stored. Here the possibility of different directories is presented and one is chosen. Possibilities of storing on an Internet connected device or in various cloud providers are assumed. In addition to this the actor selects authentication scheme, since preference towards authentication and security could vary from user to user.

Before finishing the *Registration*, the identity owner has to complete *Define attributes*. Here attributes are able to be defined and their value can either be imported from a third party (as was the case with Jane importing data from Facebook) or entered manually. In other words, the actor could import his identity data from an existing IdP or manually enter it. The option of importing data create a better user experience and reduces the time of registration. Also, the definition of attributes or claims is part of the registration, because the DIdMS can not be used if the user has not provided any identity data.

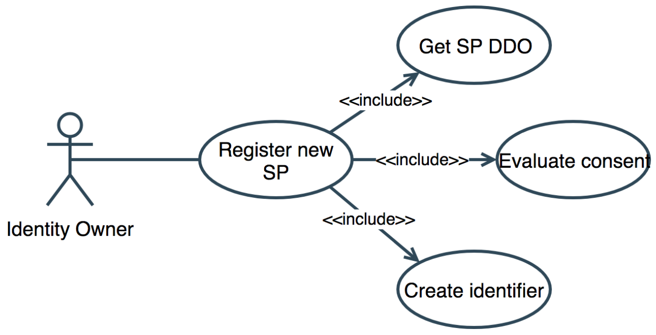


Figure 4.4: Use case diagram of *Registering a new service provider*.

After initial registration and authentication, the use cases presented in figure 4.4 are now accessible to the identity owner. The figure shows the use cases relating to scenario two in section 4.2.1. Here *Register new service provider(SP)* is performed, which is initiated by the identity owner providing a DID for which SP he wants to register. With the DID of the SP the DDO is assumed to be resolved and available, containing data necessary for presenting a consent and constructing SP specific identity. The reasoning for this comes from the idea of having the collection and management of the relationship done through the DIdMS. This use case includes the identity owner *Evaluating consent* before accepting and finally *Creating identifier* for that specific SP.

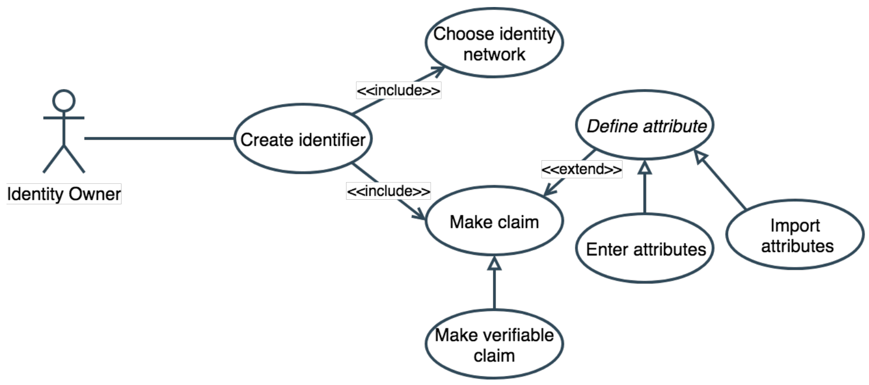


Figure 4.5: Use case diagram of *Creating new identifier*.

In figure 4.5 the use case *Create identifier* is presented. The identity owner could *Choose identity network* for which network he wants to create the identifier or DID. A default could be set, but an option of selecting one or more desired network(s) must be possible. For this identifier (in the form of DID) there will be associated certain claims which could come from the identity data provided from the *Registration* use case. The structure of this claims could be in the form of verifiable claim. This format enables verification and data integrity to be applied. In addition, depending on the claim request, the user may already posses the claim value or he may have to enter it for the first time. For this purpose, the extension *Define attribute* is added to show that when a request for an identity comes from a service, the user can either import existing values or enter them manually. On a more general note, the extension enables the user to construct more flexible and varied identities by using signed claims from third-parties. This is a process that is also discussed in Sovrin, section 3.4.2, and is important for providing higher levels of assertions.

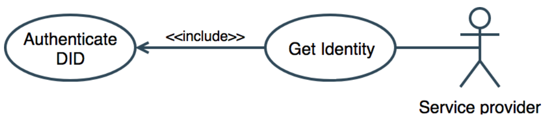


Figure 4.6: Use case diagram of a service provider requesting an identity.

When the identity is created with its associated claims, it should be then accessible to the service provider. In figure 4.6 we have depicted the use case for such functionality. In order for the actor to get the identity data, he must first authenticate as a valid requesting party. Due to the nature of DID and DDO, this can easily be achieved since when registering the identity in the previous use case, the user also supplies the service provider's DID. From this follows that authentication can be proven using the service provider's public key.

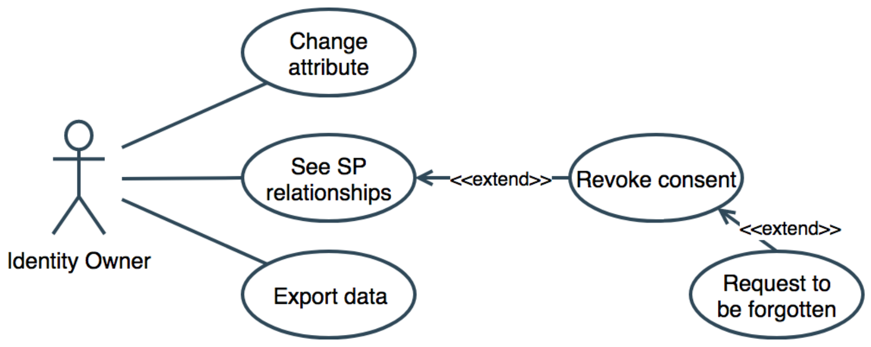


Figure 4.7: Use case diagram of managing identity.

In figure 4.7 optional use cases are presented which are in relation to management of identities. Once the provisioning and registration are done, the actor should also be able to change existing identity data. This is reflected by scenario 3 from section 4.2.1. Furthermore, with the GDPR being implemented, service providers must be able to recognize when a user is revoking consent. Since the actor is interacting with he SP through the DIdMS, the system must enable this type of communication. By extension of this, comes the use case of *Request to* *be forgotten* which is also defined by the GDPR. Finally, the user may want to migrate to another DIdMS and in order to support this, the *Export data* use case is shown.

With the establishment of the major functionalities in regards to the DIdMS, we will now try and elaborate how the identity structuring should be done.

4.3 Attribute & Data Management

In the following section an analysis of research question 2, stated in section 1.2, will be presented. With the discussion in this section we aim at outlining a good structure for designing and storing identities in the DIdMS.

4.3.1 Attribute Collection

The first aspect which needs to be considered is the process of gathering the user data. More specifically, how will this data be gathered and from what sources should the user be able to import it.

As presented in section 1.1 in the current model the user has different digital identities, depending on the context and application of their use. These are referred to as **scattered identities**. Each of these identities contains different information which is dependent on the context in which the specific application or service is working in. Hence we need to find a way through which we can aggregate all of the user data into one place.

Reviewing the approach the creators of *Digi.me* took, presented in section 3.4.1, where they allow the user to select the different applications from which they would like their information to be gathered from, we derive that it would be beneficial for the DIdMS to be able to support similar way of aggregating data. This will allow the consumers of the *DIdMS* to have flexibility of choice and would allow them to gather their scattered identities in one single repository. Furthermore, one of the primary options which need to be provided, following Digi.me's example is social media platforms. A study conducted on privacy also shows that people disclose a lot of information on social networking sites which makes them one of the richest repositories of personal data [66]. Additionally, it is important to extend the scope of applications beyond that of just allowing the user to select different social medias - such as Facebook, Instagram and Twitter. They should be able to add other types of information. An instance is healthcare data which could include knowledge of current and past injuries, illnesses and other related information regarding their well being. Another possibility is to allow them to add financial information - such as active bank accounts, current balance, spendings, etc. Furthermore, another alternative sources of information could be different IoT and Smart Home appliances the user has already installed and connected to an IoT hub[67]. An example of that could be the user having a NEST Thermostat[68], from which preferences about the temperature he resides in can be gathered. Moreover, the user can also connect different devices and information related to their physical activity - such as FitBit device[69] or Apple Watch[70].

Another approach which can be taken, if the user would not like to specify the different applications from which they would like their information to be gathered, is to have them manually input the data. This idea was inspired by the way Chekk does the gathering of their data, described in section 3.4.1. They prompt the user to manually input some of the information. This could prove to be useful since some users might not have completely up-to-date data in some of their fragmented identity profiles, or purely out of personal preference they would feel more comfortable to go through the process of entering their own personal information.

Summarizing the aforementioned processes, the system should be able to provide the user with the freedom to select the *data sources* from which he would like to have his data gathered from or what part of the information *fields* included in their user profile they would like to fill in manually, and further authorize our solution to be able to extract and accumulate all of the information.

4.3.2 Data Structuring

Once the user has explicitly selected the different data sources or has manually input their personal information and it has been gathered in one place the next logical step is analyzing determining how it is going to be structured and what a user profile, in the context of our system would look like.

In order to be able to define this common architectural pattern of the profile we first need to define the goals and objectives that we would to achieve. From sections 1.1 and 3.3 the following objectives have been identified.

Accommodate user data From the possibility of importing and providing data form different sources we can observe that a lot of information is going to be gathered in one place and the different fields in the profile need to be diverse enough so that they can accommodate all of the incoming information. This also brings about the need to define a scheme with a wide-spectrum of properties which can be mapped against the personal information.

Incorporate context Another aspect which was identified is that the different fragmented identities scattered across the web depend on the context of their use, as stated in section 1.1. This has shown us that depending on the context of the application the user can have different preferences. An analogy would be making two different playlists on Spotify depending on the different situations throughout the day - in most cases they would have different preferences in terms of music genre when commuting or driving to work and when going out for a run in the park. This generates the need to grant the user the ability to define different preferences depending on the context. I.e. the system should allow the user to generate **partial identities**, which they can later use to present themselves in a manner of their liking. A simpler option which could be employed is to have the service provider which the user is interacting with can convey the context in some form which can then be recognized by our system and if the user has some specific attributes corresponding to that context they can be invoked, otherwise a generic set of preferences will be sent back.

Dynamic bundle creation Another goal which needs to be made possible is to have the profile structured in a way that allows claims to be dynamically generated depending on the data requested by the service that the user is interacting with.

Increase data quality The final objective of the profile structuring is to increase the data quality delivered. Derived from the research in section 3.3, structuring the data in a specific way which not only fulfills the aforementioned objective, but also is kept updated will allow the services to receive a higher quality data which should return an improvement of the quality of the services they deliver back to the consumer.

Based on the previously analyzed objectives it is important to specify how are they going to be achieved, more specifically what type of format and schema would best fit them. There are two primary aspects which need to be reviewed - one is the hierarchical structure of the user profile information and the second one is the schema that is going to be used when defining the format of the data inside that profile.

In figure 4.8 the hierarchy making up the user profile is depicted in the form of a tree. The different nodes are presented in different colors, which differentiates between three primary types (tiers) of nodes. It is important to state that this structure is elicited in order to see what will the point-of-view for the user be when interacting with the system. This tree-like structure can be adopted when explaining how will the user overview his complete identity and how will the different identities be provided to services. The main root node, colored in dark, is the part of the profile which will contain all of the information regarding the user. The second tier of nodes consists of different contextual profiles, which in essence are derivatives of the main root, meaning they are presented as a small subset of the different parameters included in that main node with the possibility of each parameter having a different value which depends on the context of their use. Finally the third type of nodes, colored in orange, represent the different service profiles, which are derived from their parent node.

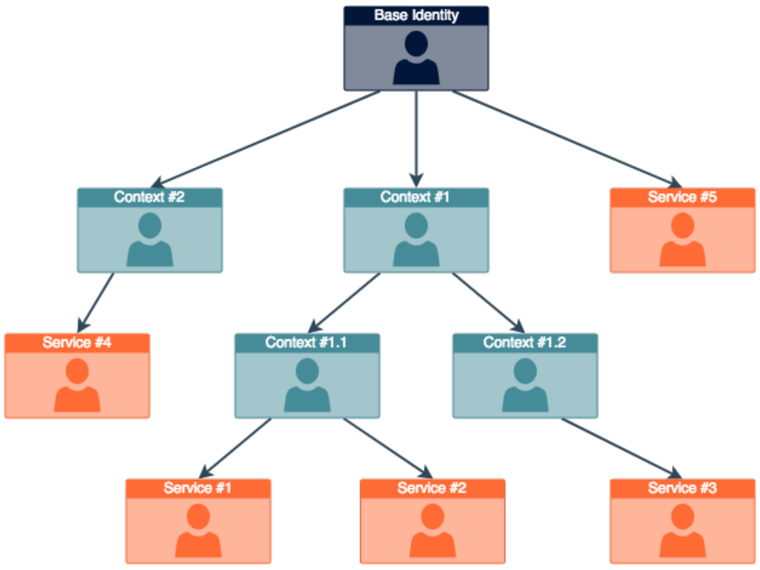


Figure 4.8: User Profile Hierarchy

The primary root of the tree has to contain information which relatively or almost never changes plus additional *default* fields, which will be further defined by the schema that will be adopted in the system. From this prime profile different preferences can be derived and a service-specific identity can be created. An illustration of this case is the main root of the tree being connected to node with name *Service #5*, containing preferences bundled directly from that main profile. Another possibility is having the different contextual-dependent profiles which are noted with names *Context #1* and *Context #2* respectively. These elements contain information fields which may also be included in the primary profile, however, they will have different values. It is evident that the former node is split further into two separate children - *Context #1.1* and *Context #1.2*. An analogy to this case could be having John Doe's details represented in his main node and the two children of that node - *Context #1* and *Context #2* represent two different profile extensions which depict his preferences in different situations of his lifestyle. The former node could be represented as *John's General profile* and the latter one could be represented as *John's Work Profile*. In this context, the first child would contain John's private e-mail and home phone number, whereas the latter child will contain John's work email address and his work phone number. The further split of the *Every Day John* profile is regarding additional contextual specifications. The child with label *Context #1.1* could be a profile containing information and preferences when John is at home, whereas the second child, *Context #1.2*, could represent again some type of individual-specific information, but in the context of John being on *Vacation*. Again each of these profiles contain some parameters that depend on context, which are derived from their parents. And each of them can then be parents themselves to different services.

In the context of our system it would make very little sense to try and define our own data schema. This is due to the fact that if we opt for that choice it would bring in additional requirements to the service providers which would like to interact with our system. The wiser choice would be to select one of the available schemes which all already in use in the digital world because that would allow our solution to be more compliant and interoperable with a wider spectrum of services.

After identifying the potential data sources from which the user can gather their personal information, and discussing how should these resources be structured the next step is to review what are the different possibilities in terms of where should the user's data be stored.

4.3.3 Identity Directory

In our problem formulation we have stated that one of the most important aspects of our system is following SSI principles in regards to control and management of user data. In the context of our report we will coin the term *Identity Directory*, IdD for short. The IdD can be viewed as an enabler to the Self-Sovereign Identity core principles since it gives the user choice of storage, which in turn gives them more control over their identity, while removing such requirement from the DIdMS. This also creates a decentralization on a small scale and increases the protection of user identity data, because it is not gathered at a centralized storage.

The IdD is an abstract term which is referred to the storage element in which the user's identity data is placed. There are fundamental aspects which need to be reviewed before being able to elicit requirements and specify where should the user's identity data reside in the context of the DIdMS. It is important to note that if the data should reside on a server which is explicitly managed by the DIdMS it would result in taking away some of the control of the user. Furthermore, this would result in specific GDPR implications which the system should comply with. If we select to provide a type of *Storage-as-a-Service* this would put us at the category of *Controller*, specified by the GDPR, and would introduce further liabilities which need to be taken into consideration. Having in mind that our objective is to empower the user as much as possible and place the complete control in their hands the option which could provide these aspects is to have them select a storage which is managed by them. This is also confirmed by observing some of the solutions which we have reviewed. For example, Digi.me and Chekk both put the data at the control of their user. Additionally, Digi.me allows the user flexibility in terms where this data would be stored at. Following their example, it would be a more sensible choice for us to provide the user with the same functionality. However, it is important to elicit requirements for this storage as well in order to make sure that the user's identity data will not be put in jeopardy.

Some requirements for the Identity Directory can directly be derived from the Identity Hub concept requirements, provided by DIF in section 3.3.3. It would be beneficial to take DIF's specifications into consideration because the define the Identity Hub as a protocol which facilitates easy management of digital identity and all of its related aspects. Furthermore, this remote repository must be able to facilitate secure storage with secure transmission of the identity data.

Another critical aspect which needs to be considered if the system would need to access the user's data under any conditions. If such cases need to be fascilitated the DIdMS needs to strictly follow the General Data Protection Reglulation's implications, such as that before any access to data is granted the user must have given explicit consent and authorized the system. Furthermore, our system is seen as running on the web, which means that the storage has to also support transmission via web protocols, specifically HTTPS. A specific purpose for accessing the user's identity data could be to provide it to other Service Providers.

If the DIdMS adopts for this tactic, it would become an aggregate of data and would enable the user to interact with other peers on the network, without taking the additional responsibilities of storage. In addition to that, Digi.me also have established a mechanism for protecting the user data from the storage service by encrypting all of the data on the client side before it is send to the storage. This could be a good design choice that increases protection and security and if chosen as part of the DIdMS. However, if that is the case, the respective requirements for key-generation, management and encryption have to be considered.

Another requirement which needs to be regard is the need for facilitating seamless synchronization and data portability. This means that the DIdMS should fascilitate the synchronization of data if the user has it in multiple places. Furthermore, it should allow for easy export of the identity data in the case that the user would like to migrate to another identity provider or would like to take out his data of the DIdMS.

4.4 Establish Relationship with Service Provider

After identifying the data structure and storage, we now focus on how to facilitate the relationship between a user and a service within the DIdMS. As already stated, in sections 4.1.2 and 4.1.3 the DIdMS must be able to create service data bundles, publish them and also support consent management. In addition, there are also requirements reflected by the sub-problem 4 that have to be addressed. As seen from section 3.1.1 the DIdMS must support two major user rights - to revoke consent and to be forgotten. In this section we will analyze how can the DIdMS facilitate all of this.

Once the identity owner is registered to the DIdMS, he can start using it to provide his identity to SPs. In the scenarios an example with the popular streaming platform Netflix is given. As briefly mentioned in section 4.1.1, the DIdMS must support multiple identity networks and must be able to work with DID method specifications for each one. This means that the DIdMS will be able to resolve a DID to its corresponding DDO, regardless of the network. As described in section 3.3.1 the DDO of a DID can contain endpoints and other information relevant for communication between peers and we can utiliza this to our advantage.

The overall flow of establishing this relationship between identity owner and SP is shown on figure 4.9. Some of the specific steps within are further discussed in following subsections. As depicted the identity owner will initiate by visiting a service provider's website. The SP will provide their DID to the identity owner, **(1)**, and the identity owner will provide this to the DIdMS, **(2)**. This tells the DIdMS that a relationship is wanted with this DID. To get the DDO of that DID - the information needed for communication with the SP - the DIdMS will go the related identity network and fetch it, **(3)**. Having the appropriate information regarding the SP provided in the DDO, the DIdMS will now query the SP, **(4)**, for what data it needs from the identity owner and what consent should be signed by the identity owner. Knowing the SP's required data and consent, the DIdMS will present this to the identity owner, **(5)**. If the user accepts the SP's consent receipt and provide the required attributes, the DIdMS will generate a new DID, **(6)**, for the identity owner. Upon finishing the publication of DID and associated DDO, the DIdMS will provide the new DID to the identity owner, which should be used for that new SP, **(7)**. The identity owner will now take that new DID to the SP, **(8)**, essentially saying "this is my identity for you". Getting the DID, the SP will need to make sure that the one providing it is in fact controlling that DID. Fetching the DDO of the DID, **(9)**, the SP will have the information needed for authenticating the DID at the endpoint specified by the DDO. Via the endpoint in the DDO, the SP will go to the DIdMS (the endpoint is protected and managed by the DIdMS), **(10)**, to get a challenge signed by the identity owner. After this process, the relationship between SP and identity owner is considered established.

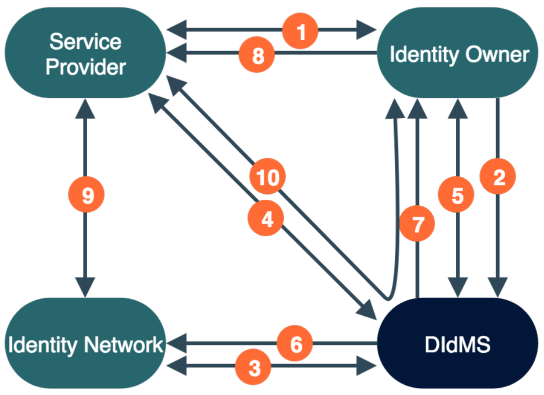


Figure 4.9: High level flow of identity owner registering a service provider.

Service Provider Exclusive DID

The DID specification, section 3.3.1, discusses a privacy issue regarding the risk of correlation with the identifiers. The problem is that if the user creates unique DID of a service provider and then gives the same one to another service provider, the two providers could collude to link this DID to the Identity owner. This could compromise the privacy of the user, which is undesirable. To tackle this issue, the DIdMS should by default protect the identity owner's privacy by creating unique DIDs for each SP relationship. This creates the process of linking the DIDs to the same Identity owner a lot harder and strengthens the privacy. As part of the DID there is also a partial (service) identity created. Hence, by creating a new DID for a SP, the identity owner essential creates a new partial identity.

4.4.1 Get Service Provider DID

As described above the relationship between identity owner and SP is also a relationship between two DIDs. The DIdMS will need to know the SP's DID and associated DDO to establish the relationship - this is step **(1)** and **(2)** in figure 4.9. In the scenarios it is expressed that Jane goes to Netflix and is redirected back to the DIdMS. In practice, this might not be so practical. The DIdMS could be one of the many available for identity owners, thus Netflix can not be expected to reference every single DIdMS and hold an available address for redirection.

In order to initiate the sequence of registration, the identity owner needs an available, valid DID of Netflix. Since the DIdMS is network agnostic, the DID could be specified on any network - Sovrin, uPort, Veres One, etc. The obstacle that rises form this is how does Jane "bring" the DID of Netflix to her DIdMS, so that the process can begin? The "bringing" can happen, for example, in one of two ways:

Manual copy/paste Jane will simple copy a DID provided on Netflix's website and go to her DIdMS and paste it in when required.

Browser plugin detects DID Jane will have a browser extension installed, developed by her DIdMS, to automatically detect a DID on a visited website. This extension can then provide the functionality, upon Jane's initiative, of starting the registration flow and "bringing" the SP's DID to the DIdMS.

The later is preferred since it is considered to be done in a more user-friendly and seem-less way where Jane does not have to manually be involved. Once this is established, Jane should be able to immediately continue with the other steps as specified. The DIdMS now knows to which DID Jane wants to create a relationship. It should be noted that the DIdMS should not know or care to whom the DID belongs, i.e. that Jane is using Netflix.

4.4.2 Service provider's DDO

Having the DID of the SP, the DIdMS will need further information regarding the SP to complete step **(3)**, **(4)** and **(5)** in figure 4.9. Associated to the DID is DDO which is stored at the *Identity network*. According to the DID specification there are some minimum requirements for what should be included into the DDO schema, section 3.3.1. One of those required properties is 'service' which refer to *service endpoints*. In this property, endpoints can be specified for further interaction with the owning entity. For the DIdMS to facilitate the relationship between user and a service provider, this DDO of the SP needs to have not only the endpoint for communication but additional ones that provide access to other necessary information.

Endpoint specification

As minimal disclosure of information is wanted, relating to both GDPR and SSI principles, the SP must provide a list of what is required, in terms of user claims. Since the DIdMS is external to the domain of the SP, the service provider could include additional endpoint that could provide specification for such data. Furthermore, the GDPR specifies rights that have to be supported by every service that utilizes personal data. One way providing this functionality is to have the user enter the domain of the SP and have the process done there. However, if the DIdMS is to support this type of requests on behalf of the user, the SP has to provide a way for external parties to make these requests. A convenient solution is to create two additional endpoints that allow third-parties to request these rights on behalf of the user. The last thing we have to consider from the SP is the consent description. As part of the flow in figure 4.9, the DIdMS has to present in a readable way the consent that the service provider has defined. Again, by using the DDO specification, the service could define another endpoint where this could be facilitated.

To summarize this you can see listing below. This serves as an example of how using DDO specification we can facilitate the requirements for user-service relationship within the DIdMS.

```
{
  "service": [
    {
        "type": "ExampleService",
        "serviceEndpoint": "https://example.com/endpoint/8377464"
    },
    {
        "type":  "consentFormSpec",
        "serviceEndpoint":  "https://example.com/endpoint/consent"
    },
    {
        "type":  "revokeConsent",
        "serviceEndpoint":
           "https://example.com/endpoint/revokeconsent"
    },
    {
        "type":  "requestToBeForgotten",
        "serviceEndpoint": "https://example.com/endpoint/forgetme"
    },
    {
        "type":  "credentialsSpec",
        "serviceEndpoint":
           "https://example.com/endpoint/credentialsSpec"
    }
  }]
}
```

Listing 4.1: Example of a service endpoints in the DDO of a service provider.

In the listing, the "type" defines the endpoint's specific type and the "serviceEndpoint" specifies a link where relevant data can be gathered. Of course, this is only one way of doing it, but since DID's specification allows such flexibility, we consider it to be a viable solution that is both easier to implement on the service provider's and on the DIdMS' sides.

4.4.3 Prove DID Ownership and Control

As final steps in establishing relationship is for the identity owner to prove that he owns/controls the provided DID - steps **(9)** and **(10)** in figure 4.9. The DID specification, see section 3.3.1, do specify how authentication information can be included in the DDO. Furthermore, the term *DID Auth* has been introduced in *rebooting the web of trust* workshops regarding "..."proving control over a DID" in one way or another."[71]. There are various methods to do this proving. One of the more flexible one is using a *challenge/response* method. An example here is to have a 'DID Auth service' endpoint specified in the DDO of the user's DID, where the SP can send a challenge. The identity owner should then create a response to this challenge, by making a cryptographic signature with the DID's associated private key. This means the DIdMS will need to be able to redirect such a challenge/response flow to the identity owner. While this type of authentication is a design choice, making use of the already existing key-pairs from the DID's is a solution that reduces overhead and makes the system easier to maintain.

4.5 Consent management

As the scope of the DIdMS is user-service communication, one of the most important aspects is consent. Consent is important because it informs the user and gives legal accountability to both parties.

In the following section we will extend the discussion, which begun in section 4.1.3, regarding user consent. We will review and explains the different elements which need to be considered in the context of consent management and DIdMS. The discussion also reflects research question 4 which is aims at identifying GDPR privileges and requirements.

4.5.1 Consent Structure of Service Provider

We have discussed in both sections 4.4 and 4.2.2, that the DIdMS should be able to retrieve and deliver the consent of the respective service provider. Consent is a conceptual term. A consent can be implemented in a myriad of ways and usually service providers are tasked with that. Before the GDPR, the service providers could just include a long list of description of their service and user rights and then prompt users to just accept or leave. However, from May 25, 2018 changes have to be made. The service providers have to consider how they process user data and distinguish between data necessary for core functionality and data processed for additional gains. For example, if Netflix wants to collect my address, but that unit of information is not relevant to their core service, then Netflix has to explicitly create a consent which can not be "take it or leave it". This empowers the user, because the service provider can no longer say "you either accept what I am doing with your data or leave". Furthermore, there are specific fields and information that has to be included in the new consents. All of this bring the question "how would the new consent look like in practice?".

Kantara Initiavie have created a data structure that tries to solve this question - Consent receipt, section 3.3.8. Consent receipt is specifically created to be a compliant format with GDPR that can be used by service providers. In addition to that, it provides in a human-readable way the core processes and the information that the user needs to approve.

In regards to consent receipt and their usage in DLT-enabled networks, the consent receipt can be viewed as an evidence that a transaction of personal information has occurred. Whenever a such a receipt is issued, the involved parties, both provider of the information and its receiver, must digitally sign it. The signature is usually generated by hashing the consent receipt data. Once both entities have signed it, each party must store securely their copy in order to prove that the aforementioned transaction has occurred, if required. Consent receipts are legally binding and they are an important aspect of insuring one's rights in these DLT-enabled networks. As referenced in section 3.3.8, a consent is an abstract term and has been developed into a specific standard by The Kantara Initiative. The standard specifies a structured way of documenting consent transaction and it takes into account the implications of GDPR on transactions which involve personal information. However, it does not present a way in which the *right to be forgotten* can be facilitated, or any additional functionalities that might be necessary if special conditions occur. For example, if the user changes his mobile phone and other contact data, the service might require him to inform them in a reasonable span of time of the change if the relevance of that information is critical to adequately delivering the services the user has signed up for. All of these special cases have not been reflected in the current form of the consent receipt standard.

Another possible data structure, which can be used as a substitute and could solve some of the aforementioned problems is *link contract*. It is a specification which is developed by XDI Technical Committee, part of the OASIS organization[72]. Linked contract is a concept that builds on the idea of smart-contracts. It gives the ability of parties to define specific conditions and respective protocols that need to happen in the event of these conditions. Linked contract also gives more flexibility of defining rights and responsibilities between to parties, while consent receipt is more constrained. Since, linked contract is not defined yet to the extend consent receipt is, we can not really compare the viability of it. If the concept is proven to work, it could be a more suitable choice as means of expressing consent between service and user. However, at the time of writing this paper, we consider consent receipt as a more viable option that can be used to represent the consent.

We have already mentioned that we have to consider how are we going to provide the right of the user to be forgotten in section 4.4.2. While the consent receipt reflects the why and how the user's data is going to be used, services also have to consider how can they provide a functionality that enables the user to request his data to be deleted and removed. In the DLT-enabled networks, a user can deny access to his data by revoking his DID, which has been given to a service. However, he can not ensure that the service provider has not stored the data, which was retrieved by them once the relationship was established. This is an issue which needs to be taken into consideration when analyzing how will the user executes a request to be forgotten.

A solution to this problem is to rely that each service specifies independently how this process can happen. This can be done by asking each service provider to include an endpoint as part of the service DDO schema, illustrated in listing 4.1 in section 4.4. This would allow agents acting on behalf of the user to make such requests and would enable better user-experience since the user would only have to access his DIdP and manage this process from there, instead of visiting each service and following their custom processes. Additionally, by saving a legitimate copy of each individual consent receipt of a specific identity owner, it would allow an them to have an easy overview and the ability to inspect the receipts corresponding to all service providers to which they have given consent.

Consent Receipt Format and Flow

Back to the "consentFormSpec", we have to evaluate what type of information is going to be specified there. First of all it is important to establish what are the necessary steps and preconditions for a service provider to issue a consent receipt to a user. A process of generating a consent receipt which will be delivered to a user will be done when the specific user is not registered to the service. If that is the case the following steps will be executed:

1. User connects to a service
2. The service needs to generate a consent to use the user's data.
3. The service calls its respective consent management server
4. The consent management server create a new consent form for the user and returns it back
5. The service presents the consent form and prompts the user to accept or deny it.
6. The user accepts the consent form and a consent receipt is issued and stored in the consent management server

This consent flow was created in the context of currently operating web services which rely on PKI and do not yet support DLT-enabled infrastructure.

In order to facilitate this process for services in more decentralized networks, they should provide the structure of their consent receipt at the endpoint which is denoted **"type": "consentFormSpec"** as presented in the listing 4.1 in section 4.4. The structure could be JSON format that includes key-value pairs of data. The "key" side should be the same as specified in the consent receipts standard by The Kantara Initiative[46].

4.5.2 Revocation of Consent

While consent receipt can be used to prove a consent transaction and establish boundaries and terms of data usage, the DIdMS needs to also support the case where an identity owner is no longer in need of a service and wants to revoke their specific consent. This means that a process which allows consent revocation needs to exist and be supported by both DIdMS and the Service Providers.

The current standard of consent receipt[46] supports "Revoke consent URL" and "PolicyURL". The "PolicyURL" field is a link that points to a document that specifies how the process is going to be conducted and what appropriate actions need to be taken. Each service will implement their own functionality of consent revocation, depending on how they store, manage and maintain consent receipts. However, following one of the problems elicited in the former parts of this section - the service provider needs to be notified in a timely fashion that an identity owner no longer would like to share their data and would therefore like to revoke their consent. From the point of view of the DIdMS, the service should provide an endpoint which can be queried to initiate the process of revocation. The result of this call should also be provided back to the DIdMS so that the system can successfully convey the result of the consent revocation of the request to the identity owner. This is an essence the use of the "revokeConsent" endpoint, showcased in section 4.4.2

It is important to note, that a revoked consent receipt does not mean that it will be completely deleted. It could simply be flagged as invalid or inactive. The reason for that could be that a user would like to stop to use the service for an uncertain amount of time, but later on decide to go back to it and when they make that decision The process has to be done seamlessly from the identity owner’s point of view. They should easily be able to read the revocation policy and then request revocation through the DIdMS.

4.5.3 Request to be Forgotten

Another problem which was outlined earlier was that the "Right to be forgotten" should be supported. The service provider must also support a way which enables the user to request all of their data to be deleted, as stated in the GDPR. The solution to the issue can be seen as an extension to the consent revocation flow, because the user would have to ask their consent to be revoked, meaning they would like to stop sharing their data and afterwards they would tell the service provder to delete all of the records of his data.

We have already examined that this is an important aspect of the client-server relationship and as a way of supporting it we have suggested that the service should expose an endpoint for that functionality as showcased in section 4.4.2.

Now that we have examined what the DIdMS must facilitate as requirements between the service provider and the user, the next step is to analyze how the actual identity transaction will be structured.

4.6 Claim Management and Structure

As part of the research question 3, we need to identify how can we transfer and create claims. The claims need to support attestation in the form of signatures that can be validated. These claims are the core identity data that constitutes an identity transaction between a service provider and a user.

4.6.1 Dynamic Credential Creation

As the SP will have to define which attributes it needs from the identity owner, the DIdMS must be able to handle this process. We already explained that an endpoint can be used for this purpose, section 4.4.2. In figure 4.10 the imagined process of dynamically creating credentials is illustrated. Entities are named according to the convention used in section 3.3.6 verifiable credentials. The *DIdMS/holder* will fetch the credential specification from the *SP/verifer* given the endpoint specified in the DDO. When looking at the credential specification, the DIdMS should look at the identity owner's defined attributes to try and match. If this matching can not happen automatically, the DIdMS must prompt the identity owner to match or provide a value. If the SP has specified that some of the credentials must be signed by certain *issuer(s)*, the DIdMS will need to go to that issuer to get a valid verifiable credential. The DIdMS should redirect the identity owner to the correct issuer, while specifying which credential(s) must be signed. Upon possible authenticating the identity owner, the issuer will return a verifiable credential to the DIdMS.

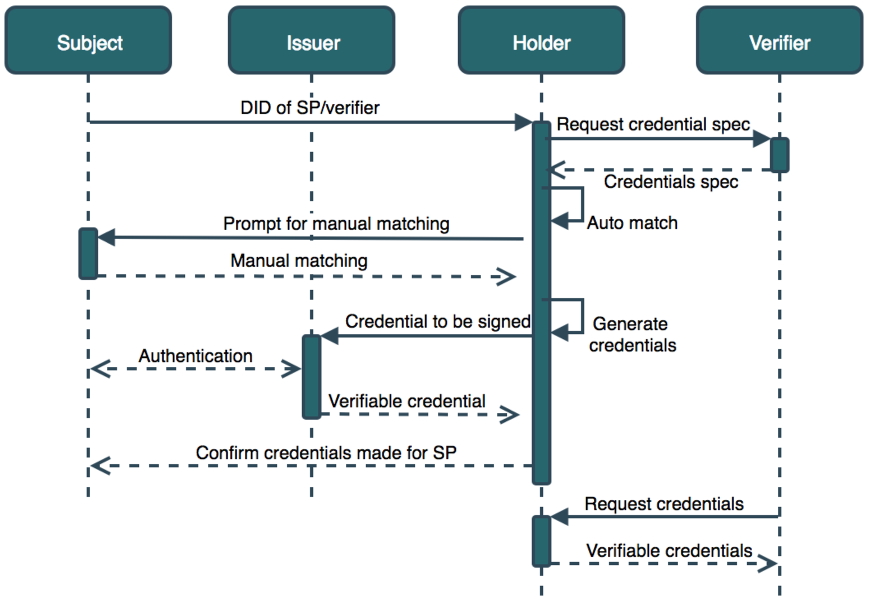


Figure 4.10: Sequence diagram of dynamically creating credentials.

While the dynamic generation of signed claims is showcased as part of providing identity data to a service provider (verifier), it is possible to request verifiable credentials as an independent flow.

4.6.2 Verifiable Credential Process

From our research, we can see that a good data structure that enables the transmission of cryptographycally signed claims is the Verifiable credential 3.3.6.

To support the process of getting verifiable credentials from an issuer, as presented in section 3.3.6, in a DLT-enabled ecosystem some issues have to be addressed. Since a new DID is created for each relationship with SP, this means that a new verifiable credential has to be created as well. Verifiable credentials rely on using public-private key cryptography and each new DID will have unique pair of keys. Hence, there will be one-to-one relationship between DID - Verifiable credential - Service provider. However, Verifiable credentials are made out of claims. Claims are usually more persistent and don't change over time that much. Whenever a new service registration process is started, the DIdMS uses the claims to dynamically generate verifiable credential for that relationship.

Another aspect of the asserted claims is trust. The specification regarding verifiable credential described in section 3.3.6 presents a one-way trust model where the *verifier* has to trust the *issuer*. This means that the verifier will have to be able to communicate to the *holder* which issuer it trusts. If the holder does not hold a verifiable credential from a trusted issuer, it must go get one. An example of this is that Netflix would like credit card information, validated by the user's bank. The DIdMS must recognize when such assertions are required and also be able to know where to go in order to provide a way for the user to obtain them. Some issues rises for this specific process of getting the right verifiable credentials from certain issuers.

- One issue is that the issuer must then have a defined way of providing such a verifiable credential - more specifically regarding the communication between DIdMS/holder and issuer. This could be defined endpoints, which could be present in the issuer's DID/DDO. Since it is machine to machine communication the verifier will also need to make sure that the issuer will support the attribute which it requires. E.g. the verifier, specifying a required attribute called 'ageOver18', will need to know that the issuer(s) know what that attribute means and are able create a verifiable credential for that.
- Another issue is, it can not be assumed that the *subject* (user) has a relationship with any of the verifier's trusted issuer(s). Depending on the nature of the issuer, authentication by the subject could be necessary for issuer to issue a verifiable credential. Furthermore, the subject needs to trust the issuer, that it will not provide false credentials.

There is a differentiation in how the identity networks can establish trust between issuer and verifier. This means the exact process is not quite clear, but it must be assumed that cross-identity network supports the discovery of issuers via DID.

4.6.3 Match Between Requested Claims and Existing Identity

When a SP dictate which credentials are needed in the credential specification, values for these have to be specified in one way or the other. The identity owner will need to construct claims matching the SP dictated credentials, which could happen automatically or manually.

Automatically matching

To enable automatically matching of the required attributes to the identity owner's defined attributes, a common context or schema must be used. Some obstacles of automating this matching could be:

- Attribute names could be spelled different. E.g. SP dictates an attribute called 'name', but identity owner has one called 'Name'.
- The same attribute could reference two different objects. E.g. identity owner has defined an attribute called 'size' which is his shoe size, but a SP needs a 'size' attribute, referring to pants size.

To overcome these obstacles and provide automatically matching of attributes, the use of JSON-LD and Schema.org could be implemented 3.3.4. With the SP specifying a context for its required attributes, the DIdMS will be able to check if the identity owner has any attributes defined within same context.

Manually Matching

If the above process of automatically linking a required attribute to an identity owner is not possible, the user will have to manually match the missing claims.

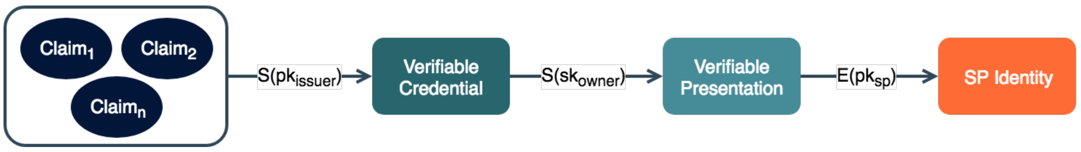


Figure 4.11: Expected process of creating the identity for a SP.

Since the user will probably be able to understand the context of which the required attribute is in, a drag-and-drop feature could support him linking his own defined attribute. If a fitting attribute has not been defined by the user and it also needs attestation from trusted third-parties he can use dynamic claim generation.

4.7 Providing Identity To Service Provider

The following section will analyze the process of how the identity provider retrieves the user's identity data once the user has authorized it. The preceding sections analyzed the process of establishing the relationship between identity owner and SP. Once established the SP will need to retrieve the identity owner's partial identity - created specifically for that SP. Described in section 3.3.3 the proposed Identity Hub is a concept of how this can be done. The concept of the Identity Hub defines a single place to manage and store the DID related identity.

Following the requirements elicited in the DIF Identity Hub document[34], we can derive that the newly created identity will be in the user-specific Identity hub. Furthermore, in section 4.3.2, we state that each user will have their own Identity hub. This implies the aspect of high availability. This should also reflect the DDO of the new DID, which must contain an endpoint to that Identity Hub for discovery purposes. The Verifiable Presentation(VP) containing the claims for the specific SP is located in the Identity Hub. This VP can be cryptographically verified, since it is signed with the private key of the identity owner's DID. The DIdMS or anybody else can not break the entegrity of the VP. But as the VP was created for a specific audience, a SP, nobody else than the SP should be able to read the VP - not even the DIdMS. As a resonse to that, the VP can further encrypted using the public key of the SP. The idea of what exactly is provided to the SP is depicted on figure 4.11. On this figure the process of signing is denoted as $S(< keyusedforsigning >)$ and the process of encrypting as $E(< keyusedforencryption >)$. The keys are denoted as pk for bulic key and sk for secret/private key.

After creating the partial identity - the SP identity - it has to be put in a Identity Hub. The DIdMS must publish the required endpoints for that Identity Hub. Due to the requirement of high availability it would make very little sense to have this at a user-managed device, because it is highly probable that the device might not be online or available when a SP would like to initiate an identity information transaction. Therefore the DIdMS must take this responsibility and be able to generate the endpoints where the data will be accessed by the SP. In addition, when the endpoint is live, it has to be protected, which means the Identity

Hub must be configured accordingly. This ensures that only the appropriate service provider can gain access to the identity data (partial identity) stored at the endpoint. Furthermore, the logging of the SPs requests can be logged for audit purposes and give the identity owner a overview of the usage.

4.8 Managing the Key-chain

The term *key-chain* refers to the storage, in the form of bundle, of the private keys which are related to DIDs. In other words, the key-chain is the structure where private key of various DIDs are stored and protected. This relationship is depicted on figure 4.12 where the private keys are shown as the controlling both the associated DID containing related public key and the identity profiles in the identity directory. This relationship is shown since we already established one-to-one relationship between DID(user), Verifiable credential (service identity) and Service provider (DID).

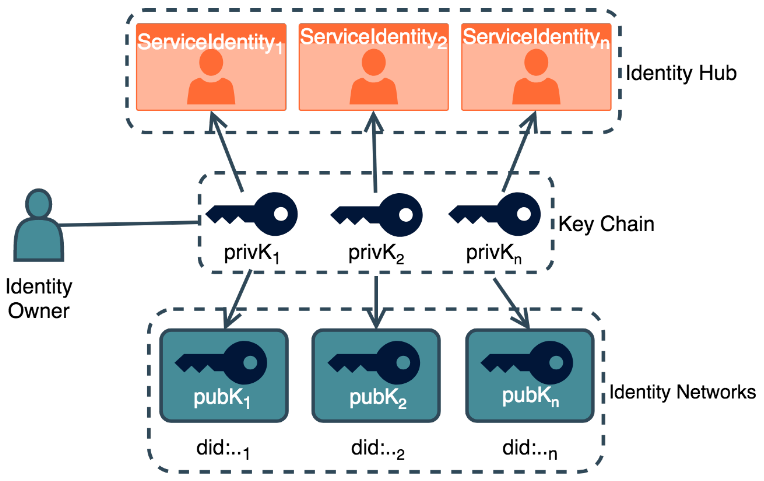


Figure 4.12: Use of identity owner's private keys in a wallet.

This key-chain will essentially control the complete identity of the identity owner and is regarded as very personal. The key-chain is separate and owned entirely by the owner. The owner of the key-chain, owns the private keys which in turn translates to owning the DIDs. You can also see this reflected in figure 4.12. In regards to the DIdMS, the user will have to be able to use the key-chain functionality and storage while operating within the domain of the DIdMS.

4.8.1 Purpose of the Keys in the Chain

To better analyze where and how the key-chain should be stored and managed, the exact use of the keys within the DIdMS have to be established. Figure 4.12 shows an overall relationship with the identities and DIDs, but it does not present a complete picture. The keys are also used to provide functions such as proving control of DID, enable managing the DID, authentication between peers (DIDs), etc. Beside controlling the DID, a private key will also be used in the signing of service identities and verifiable credentials. This means when creating a specific identity for a SP, the DID associated private key will be used as proof that the owner of the DID has constructed it and not somebody else.

Due to the mentioned use cases and the importance of managing correctly the private keys, the access to it by other than the identity owner should be minimized.

4.8.2 Storing of the Key-Chain

In general the DIdMS might need to have access to the key-chain to provide the functionality required. The main question regarding the key-chain is to which extend the DIdMS should have access to it and where is should be stored. In table 4.1 this is discussed in relation to server and device - device being a user controlled device and server could be DIdMS server or an Identity Directory.

		Server	
		Yes	No
Device	Yes	✓	✗
	No	✗	-

Table 4.1: Storage of the Key-chain within the DIdMS.

Within the context of this discussion, the *Device* is a user control and possesed device - phone, personal computer etc. The *Server* is an abstract entity that resides outside of the user control. It can be a cloud storage, identity directory or backend server of the DIdMS.

Looking at the table, the important take-away is that if we store the key-chain only on the device, the user will be the only one who has access, but he will also be the one bearing the risk of loss. If the device is broken or lost, all of the private keys stored in the key-chain will be gone. This means that one accident could lead to complete loss of identities and DIDs which is unacceptable. On the other hand if it is stored entirely on the server side somewhere, then user has almost no control. In addition, the server has to be protected and if something goes wrong, many DIDs could be compromised due to their centralization at the server. As a remedy to this problem, we propose storing the key-chain on both the device and the serve. However, the server copy should be stored in an encrypted form and should be decryptable by a secret only known the owner of the key-chain. By doing this we create a back-up of the user's key-chain, without risking compromization if the server copy is stolen or hacked.

4.9 Authentication Protocol

As part of the use case 4.3, the user has to select and authentication mechanism.

Selecting an appropriate authentication protocol is relevant in couple of regards. First, user experience is crucial and conventional email/password solutions have proved to have their weaknesses. Second, the authentication schema entails requirements regarding the storage and manipulation of data. For example, if password is used as token of proof for an account, then the system must be able to store registered passwords and link them to an account. For these purposes, we investigate how can we achieve user authentication to the DIdMS that is more user-friendly and also requires less capabilities for protection.

As described in 3.3.7, Web Authentication API is a new standard that builds on the concept of password-less experience, by providing authentication protocol to web services and users. In this standard, the user relies on an authenticator to store and mange his credentials. The authenticator in turn validates and attests credentials to the requesting services by involving the user to provide biometric, PIN or other challenge.

4.9.1 Accessibility in DIdMS

An important benefit of using a DIdMS is that it could be completely device agnostic. This means that the user can use the DIdMS service from any device or computer as long as he can present valid credentials for his account. The employment of a standard like Web Auth API promotes such behavior and reduces the authentication capabilities for the service providers.

In order to employ it to the DIdMS, the user needs to posses an authenticator. An authenticator can be either working on the device as a software or it could be on a separate hardware device like a phone. Usually the authenticator is supported by a client platform that enables the connection between authenticator and service. The approach of Web Auth API is more user-centric where the authentication decision is no longer done by the service, but it is outsourced to a trusted device. The service only stores a public-key credential. While authentication to the DIdMS is not part of the core functionality that it provides, the discussion in this section identifies Web Auth API protocol as a good candidate that could support better user experience and less protection capabilities considerations for the DIdMS.

4.10 GDPR Considerations

As already mentioned in section 3.1.1, GDPR is a regulation that aims at improving the handling of personal data in the European Union. This means that every company within that jurisdiction has to be compliant and needs to address how, why and where it stores personal data. This means greater overhead, costs and overall resources have to be devoted to this problem.

In regards to the DIdMS the goal is to empower the user and his identity control, which inevitably includes personal data, thus we have to consider how exactly the GDPR is reflected.

We have seen form DigiMe 3.4.1 that it is possible to construct a platform (system) where you are GDPR compliant and don't necessary have access to personal data. By access we mean that the platform itself can not get the personal data (independently), store it and use it. They have done this by using encryption, minimal disclosure and "bulky" client-side.

Extending this concept, one of the interesting approaches is the idea of storing user's encrypted identity data on a chosen repository such as Google drive, Dropbox or other. This was also the major reason for us to introduce the term *Identiy directory*. Digi.me has created a concept that avoids GDPR regulations and we view this as a desirable approach as well.

In regards to how they actually do this, they employ encryption when storing the data. The encryption key is generated dynamically from user input, which means that Digi.me does not store keys. Hence, even if Digi.me had access to the data in the repository, they could not read it since they don't have the key. Well, then the question that may come from this is: if Digi.me provides the function for encryption, how is that they do not know the key? The answer of this question is that they just do all of the logic on the user's device. Every encryption, decryption or any processing is done entirely on the client side, in this case the user's phone. Once the processing is completed the key is dropped. By doing all of this, they avoid having to deal with GDPR, because technically speaking all of the processing of personal data is done on the user side and Digi.me does not use it in any other way. They have achieved the goal to be able to say that even if their system is compromized, user's data is not.

Reflecting this concept, we can extrapolate an important requirement. The DIdMS solution must not store any personal data in its database or backend. It should do all the relevant processing on the client side, in order to avoid GDPR implications. The question of how to achieve this goal is a design solution, therefore we will again revisit this question in the design section and discuss it there.

If we manage to achieve what Digi.me have done, the DIdMS may not even have to consider any capabilities required by the GDPR. This would be beneficial for both the user, since he will retain full control over the storage and viewing of his data, and the DIdMS, since it will not need to implement the added cost and overhead of GDPR.

In regards to the privileges and rights provided by the GDPR, we have already discussed in detail how they can be enabled. The implementation of the endpoints, as specified in section 4.4.2, are seen as viable solution that could be implemented.

4.11 Requirements

In this section the requirements identified up until the end of this project is presented. It should not be considered as a finite list of requirements for a DIdMS. As mentioned in section 2, if further development would have happened further requirements would have been identified. For the time being, these requirements have been identified based on the analysis chapter. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD

NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this section are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels"[73].

Since it is primarily the core functionalities of the DIdMS which have analyzed so far, most of the requirements are viewed as a **must**. In addition, the requirements are broken into *functional* and *non-functional*. Functional requirements are denoted by "FR" and non-functional are denoted by "NFR".

4.11.1 Functional Requirements

Table 4.2: Table of the functional requirements

No.	**Requirement**	**Section**
FR1	The identity owner **should** be able to define attributes by retrieving data from his Facebook, Twitter or Google account.	4.3.1
FR2	The identity owner **may** be able to define attributes by retrieving data from a user-specified sources - bank, hospital, IoT devices.	4.3.1
FR3	The identity owner **must** be able to save all data generated with the DIdMS in his Google Drive protected by a self defined secret.	4.3.3
FR4	The identity owner **must** be able to construct identities which hold attributes defined by the identity owner.	4.3.2
FR5	The identity owner **must** be able to share the same attribute between multiple identities.	4.3.2
FR6	The identity owner **must** be able to have an overview of all his identity transactions with service providers.	4.7
FR7	The identity owner **must** be able to inspect all consent receipts given for service providers through the DIdMS.	4.5
FR8	The identity owner **must** be able to revoke a consent receipt given for a service provider.	4.5.2
FR9	A service provider **must** be able retrieve an identity owner's identity at an unique endpoint.	4.7
FR10	The identity owner **must** be able to prove control and ownership of a DID towards a service provider.	4.4.3
FR11	Upon an identity owner registers a service provider, the system **must** be able to present the consent in an human readable format.	4.5.1
FR12	The user **must** be able to authenticate to the system by providing a username and password.	4.9
FR13	The user **may** be able to authenticate to the system by using an authenticate as described WebAuth.	4.9
FR14	The user **should** be able to request assertions from any issuer which have published their DID on an identity network.	4.6.2

4.11.2 Non-functional Requirements

After defining the functional requirements, the non-functional are elicited in table 4.3. The different requirements which fall into this category are related to operations that would allow the system to facilitate the specific behavior explained in the table containing the functional requirements.

Table 4.3: Table of the non-functional requirements

No.	**Requirement**	**Section**
NFR1	The system **must** be able to publish new DIDs on an identity owner specified network on behalf of the identity owner.	4.4
NFR2	The system **should** be able to support the necessary authorization protocol for Facebook, Twitter and Google to access protected resources of the identity owner.	4.3.2
NFR3	The system **must** be able to perform symmetric encryption of the identity owners data.	4.3.3
NFR4	The system **must** be able to generate a key for symmetric encryption based on a secret entered by the identity owner.	4.3.3
NFR5	The system **must** be able to support a hierarchical structure of identities, enabling one identity to inherit one or more attribute values from another identity.	4.3.2
NFR6	The system **must** log all queries coming from the a service provider requesting identities.	4.7
NFR7	The system **must** by default generate a new identity for an identity owner for each service provider relationship.	4.3.2, 4.4
NFR8	The system **must** be able to resolve and validate a consent receipt delivered from service providers.	4.5
NFR9	The system **must** be able to notify a service provider upon an identity owners revocation of their consent.	4.5.2
NFR10	The system **must** be able to dynamically generate unique endpoints for identity owners DIDs.	4.7
NFR11	The system **must** be able to authenticate an authorized service provider requesting an identity at an endpoint.	4.7
NFR12	The system **must** be able to notify an identity owner of the necessity to prove ownership of a DID.	4.4.3
NFR13	The system **must** be able to cryptographically sign and validate data using private/public key pairs.	4.8
NFR14	The system **may** be able to support Web Auth API specification as authentication method for the identity owner.	4.9
NFR15	The system **must** be able to resolve a DID to a DDO.	4.4

4.12 Conclusion of Analysis

In this chapter we have examined different solutions to our research problems. We started by identifying that in order to answer our problem formulation we have to create a Decentralized Identity Management System. The system is based on the emerging DLT-enable network which utilize the concept of ledgers and blockchain to create public register that can contain unique identifiers for identities. By basing the DIdMS on these networks we enable the user to have more complete SSI and also create the necessity for the DIdMS to facilitate user-to-service provider interactions that would usually take place at the SP's domain. With the employment of scenarios and use cases we identified the core of these interactions. We also identified how to facilitate GDPR requirements and make sure that the DIdMS follows the GDPR while at the same time preserving user's privacy. Building on that, we found a suitable way to structure user's identity and specified a flow that facilitates the provisioning of it to service providers. As product of this analysis we outlined key requirements for the development of the DIdMS and in the next chapter we will continue our discussion. We will take some of the requirements and try to develop a DIdMS system that can facilitate the scenarios and user cases we have established.

Chapter 5

Design

This chapter will present the design done for this project, according to the development of a prototype and meeting the requirements established in the analysis. First the overall system architecture will be illustrated assisted with sequence diagrams depicting the interactions between the entities in architecture. The sequence diagrams showcase different core functionalities which are described in the analysis. As an iterative/spiral approach is taken, see section 2.2, the design presented here should not be seen as final. The goal is not to have a complete design in place, before moving on to implementation, but design the core use cases and scenarios that have been established so far. This chapter represents our contribution as to how we imagine the development of a DIdMS.

5.1 System Architecture

On figure 5.1, the overall system architecture of a DIdMS is shown, which is a client-server architecture. The server will handle all interactions with the *Identity networks* and *Service Provider*, whilst the *Client* will interact with the *Identity Owner* and *Identity directory*. This architecture could imply the main part of the business logic is done server side and keeping the client rather small - handling the presentation logic. Mainly due to requirement section 4.11, a client with more capability is needed and more of the business logic will happen on the client. Both the clients and SPs will communicate with the server by well defined web APIs.

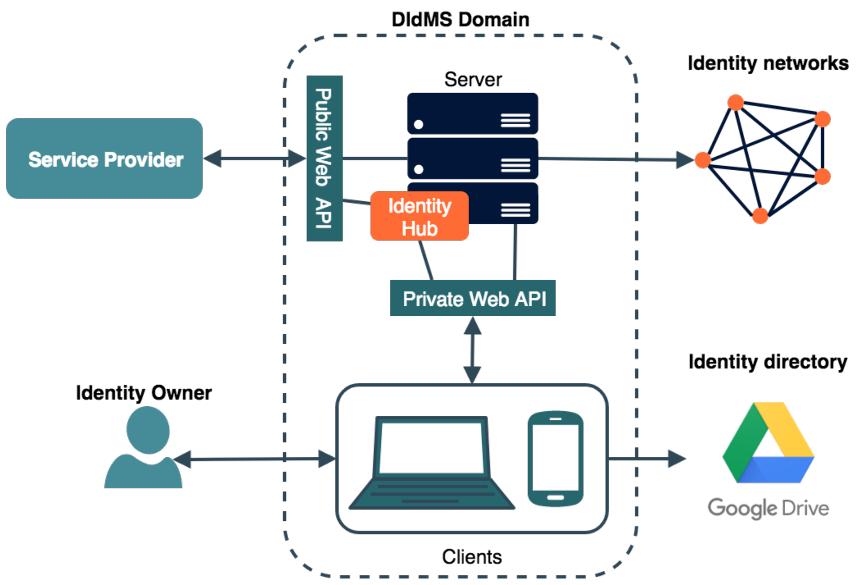


Figure 5.1: Overall System Architecture

This system architecture is based on the context diagram we presented in figure 4.1. The server is depicted as an abstract entity, in practice it could be consisting of one or multiple machines or entities. Since for this project we are focusing initially on a prototype and proof-of-concept, the complexity of the server/backend is intentionally kept low.

In regards to actual functionalities, we presented in the GDPR section 4.10 in Analysis that we would like to avoid storing personal data on the server. We would like to have a heavy client that processes and deals with personal data, thus we opt for segregating the usage only at the user's side. This is the stepping stone for our design of the interaction between entities, depicted at figure 5.1. Any personal data that might be processed by the DIdMS should only be kept at the client side. Thus, the *Identity directory* is connected only to the *Clients*. The overal domain of the DIdMS can be divided in two parts - Frontend and Backend. Frontend includes the Clients, while Backend consists of an application serve that exposes both private and public APIs.

However, if we want to process personal data only on the *Clients* but also want to have the endpoints (containing identity data) at the Backend Server, how can we do that? The answer is encryption. We will go in further details in section 5.2 explaining how we achieve the distribution of identity data, without having to expose such at the Backend side.

5.1.1 Domain Model

Domain model is a term that describes how the different entities within the context of the DIdMS are interconnected with each other. The Domain model is broad and it is not divided into Backend and Frontend. Its goal is to explain the hierarchy of dependency, in order to outline a skeleton structure for the system. The model can be seen in figure 5.2. This domain model diagram is done using UML class diagram notation, including aggregation/composition associations and generalization. The classes included should not be considered to be implementation classes since implementation class diagrams may differ. This diagrams is kept abstract. For detailed class diagrams please refer to 5.9.

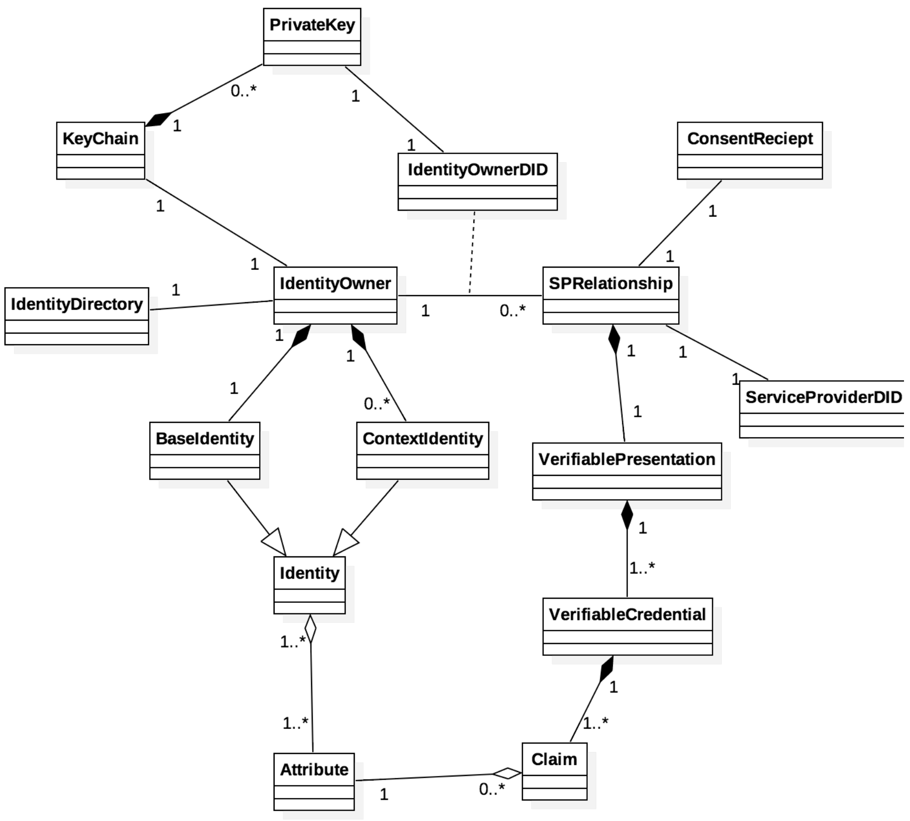


Figure 5.2: Abstract Domain Model of the DIdMS.

Identities

Following the analysis section 4.3.2 an *Identity Owner* will have exactly one *Base Identity*, but up to many *Context Identities*. The *Base Identity* is also a *Context Identity* but is considered as a root. In the figure 5.2 the *Identity* class is an abstract class that contains only the skeleton of what constitutes an Identity. It contains the schema or claim structure that is used by any other Identity. Every contextual identity and *Base Identity* extends it.

Identity Owner

The *Identity Owner* is a class that depicts the user. The user can have only one identity directory selected for storage and this is show by the *Identity Directory* class. In addition, the identity owner also has one *Key Chain* class. This class holds his cryptographic keys. As part of the DID specification, crytpo keys are required, which means that they have to be reflected as a data structure. This is also seen in the figure, where a *Identity Owner DID* class corresponds to one *Private Key*.

SP Relationship

Within the DIdMS domain an identity owner can have multiple *Service Provider Relationships*. When creating a new relationship a service provider's unique DID must be provided for the identity owner. A service provider relationship is then defined by an *Identity Owner DID*, *Service Provider DID*, *Consent Receipt* and a *Verifiable Presentation*. The verifiable presentation as described in 3.3.6 is unique for each SP relationship. And the content of these verifiable profiles is co-defined by the SP.

Verifiable Presentation

The *Verifiable Presentation* can be viewed as the actual data structure that contains the partial identity, as explained in section 4.3.2. Looking at figure 5.2 we can see the hierarchy that follows the creation of a *Verifiable Presentation*. First, notice that an *Attribute* is part of the *Identity* and also the *Claim* class. This means that *Identity* will contain attributes but when we want to create partial identities from its children, these attributes are converted to *Claim* structure. Having defined the claims for the partial identity, they are then transformed into a *Verifiable Credential* as specified in section 3.3.6. Having established verifiable credentials, a *Verifiable Presentation* is created that aggregates them. This data structure represents the identity data provided from the identity owner to the service provider.

5.2 Privacy & Security Choices

One of the objectives of the DIdMS is to protect the privacy of the identity owner. This section presents how the DIdMS will make sure the privacy of the identity owner and the security of his data is protected. We will also elaborate how we achieve the objective of not storing personal data while at the same time exposing such at protected endpoints.

Decentralized storing of user data. An user specified identity directory (initially supported is Google Drive) will store all his identity related data. This will make sure that if the DIdMS server gets compromised, non of the user's identity and data will not be compromised.

Pairwise unique DID. To avoid correlation issues the DIdMS will generate unique DIDs for each service provider the identity owner establish a relationship with, discussed in section 4.4.

Minimize unencrypted data on server. The server should have limited, if possible avoid, access to unencrypted identity data or even handling of it.

5.2.1 Identity Data Encryption

The data stored in the identity directory is considered confidential and private for the identity owner. This is why the DIdMS should assist the user in encrypting all data put there - using a symmetric-key algorithm, like AES. Since the user must have access to the data from various clients, a client/device unique key can not be used. This means the user will need to be able to generate a key from each client used, based on some secret only the user knows. This is depicted on sequence diagram 5.4 regarding log in, where the user will enter a master secret and the client will decrypt - step 7 and 8 the data from the Identity directory. To generate the key for symmetric-key algorithm a *key derivation function* can be used - keyed hash function or maybe better Argon2[74].

The encryption of the data stored at the identity directory protects the identity of the user if his identity directory is breached. Since the decryption is only available at the client software of the DIdMS, the data is completely protected from leaks and other threats at the storage domain.

5.2.2 Verifiable Presentation Encryption

Providing the verifiable presentation(VP) to the SP is happening by the SP querying the DIdMS public web API. Since the VP is considered confidential to the identity owner and SP, the DIdMS server should not have a plain text version stored. For this the DPKI nature of the identity networks can be used, utilizing the public key associated to the DID of the SP. The VP specification already specify how signing and keeping integrity of the VP can happen, but a further encryption of the VP can be provided to ensure the confidentiality. This can be done by using a freshly generated symmetric-key algorithm to encrypt the VP, and then use the public key of the SP to encrypt that symmetric key with an asymmetric-key algorithm. Upon receiving the encrypted VP and encrypted symmetric key, the SP can decrypt the key and then decrypt the VP. Note, the symmetric key generated here is different than the one in the identity directory.

By using this strategy we ensure that only the service provider can decrypt the data stored at the endpoint. Furthermore, the DIdMS only stores cypher text which means that even if attacked occur, no losses of privacy can take place. In addition, the storage of cipher text also means that GDPR is not enforced and the backend server does not have to consider additional capabilities.

By employing the encrpytion strategy at the identity directory and endpoint, we can see that at no point the backend of the DIdMS processes or stores personal data. We also ensure that only the user and the service provider can see the respective identity data in plain text.

Backup

Since the DIdMS will not know the value of the master secret (it is known only by the user) for deriving the key used for encrypting user's data, the DIdMS will not be able to help if the user forgets this master secret. The problem is that once the data is encrypted at the drive, the DIdMS relies on the user to provide the correct master secret in order to generate the key. If the master sercret is lost or forgotten, the data at the endpoint will never be decrypted.

In addition, the identity directory could also be corrupted and the data on it destroyed. To avoid these issuers a backup is required and this backup must be encrypted with something else than master secret defined by the user. A solution to this can be to have the user register a trusted device on which to store a backup. This backup can then be encrypted with a device unique key, which the user essentially knows nothing about. This also reflect the analysis presented in section 4.8 regarding the storing and access to the key-chain.

In section 4.8 we discussed the key-chain storage and argued that it is a good idea to have a back-up stored somewhere of it. In this section we build on top of this and suggest that in addition to the key-chain we can add the user's identity data. The key-chain plus the identity data constitutes the complete profile that user has within the DIdMS. By having the backup being encrypted by a device secret, rather than a user selected one, allows the user to recover his identity if he forgets or losses the master secret. This also means that DIdMS must be able to recognize registered devices that are provided to specific user profile, in order to begin the process of restoration.

5.3 Sequence Diagrams

In this section we will present specific use cases with their appropriate Sequence diagrams. The use cases were chosen in accordance with the design and the objectives of the report.

We reference the following requirements in the sequence diagrams: FR1, FR3, FR4, FR7, FR8, FR9, FR11, FR12, NFR1, NFR2, NFR3, NFR4, NFR5, NFR7, NFR8, NFR9, NFR13, NFR15.

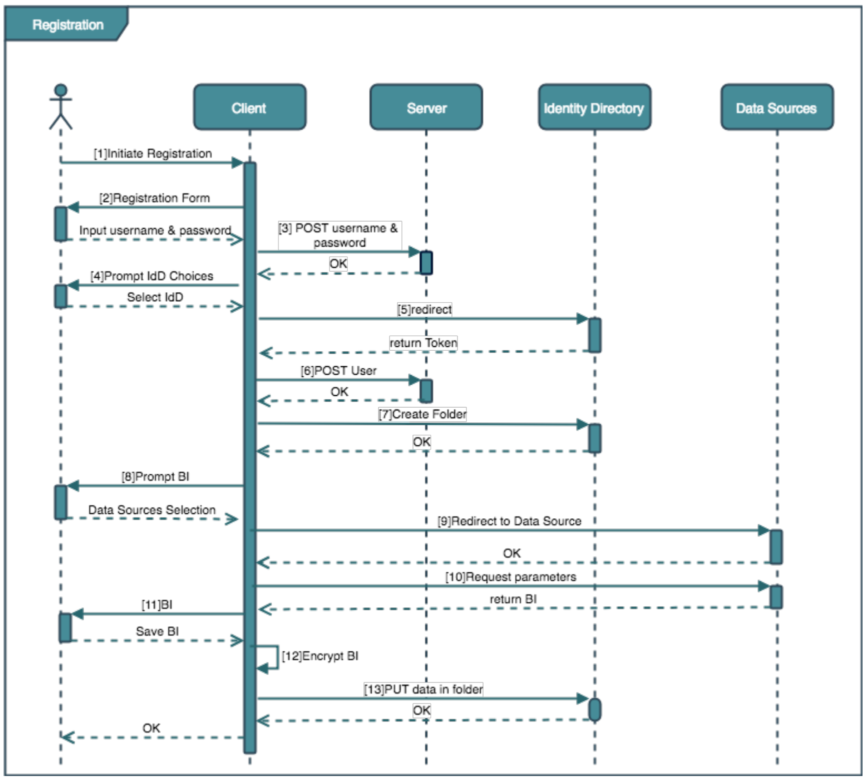


Figure 5.3: Client Registration Flow

The process illustrated in figure 5.3 begins by the user initiating a registration flow, denoted with **(1)**. The client presents a **(2)** Registration form, which is comprised of several steps. The first step requires the user to input a simple username and password combination. Then the system executes action **(3)** which involves sending a POST request to the DIdMS. Afterwards the user is presented with a list of potential Identity Directory options from which they have to select one. Once they have selected a specific identity directory they are redirected to the specific platform where they have to authorize the application client to be able to save their identity there. The result from the action is that the client obtains an access token. Then the client makes another POST request, indicated as **(6)**, to the server where the IdD field of the specific user is updated. Once the authorization is complete the user creates a root folder in the previously selected IdD. In this folder their identity will be stored, as discussed in section 4.3.3. The next step in the registration process is denoted with number **(8)** Prompt BP. This indicates that the user is prompted to select the different sources for importing his identity, analyzed in section 4.3.2. The user is then redirected to the specified sources where they have to authorize the client to be able to retrieve their data. This data is then aggregated and their complete identity is constructed. Once action **(10)**, is complete the data is returned to the client and is presented to the user for a final overview - action **(11)**. If the user is satisfied with the gathered information they can approve it which will trigger action **(12)** that encrypts the identity that was aggregated and once the process is completed the generated cyphertext is saved to the previously selected folder in the IdD. The user is then informed that the process has been successful.

The sequence diagram in figure 5.4 reflects the process of the user authenticating to the client. The following diagram is associated with requirement FR12.

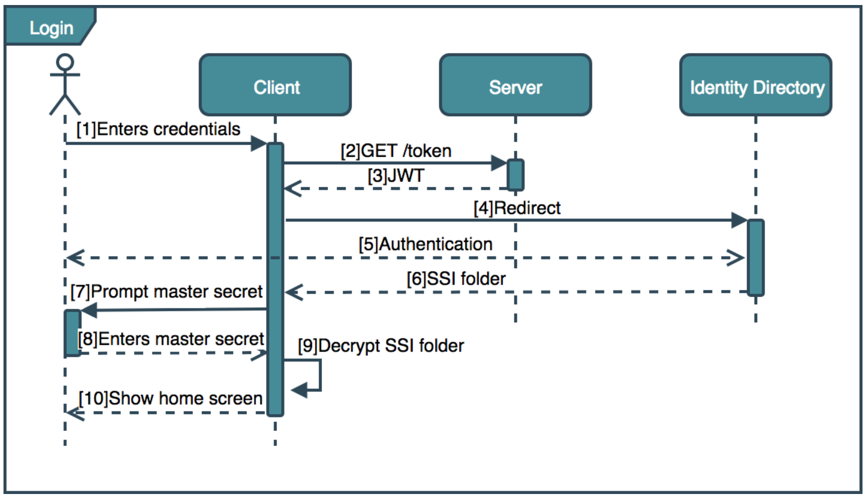


Figure 5.4: Login sequence

The flow is initiated when the user opens the client applicaton and is prompted to enter his credentials, denoted as **(1)**. Once the credentials have been entered a GET request is sent to the backend of the DIdMS, which checks if the credentials are valid and if they are it sends back a response. Afterwards the user is redirected to the identity directory from where the client application retrieves the folder which contains the identity stored in ciphertext. The user is then prompted to enter the master secret, action **(7)**, which then results in decrypting the identity. Finally, the home screen is showed to the user and the client is ready for use.

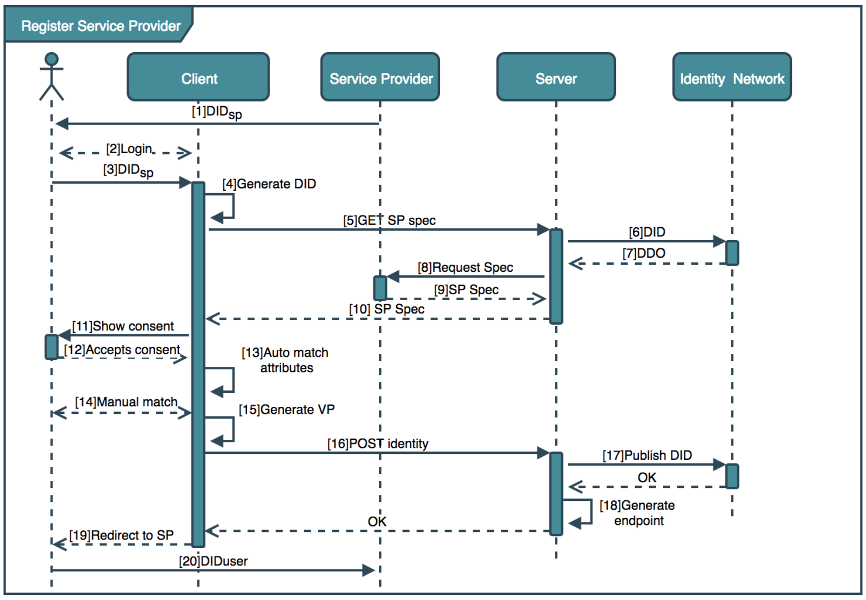


Figure 5.5: Sequence diagram of registering a new service provider.

In figure 5.5 we have displayed the sequence of registering a new partial identity for a service provider in the DIdMS. To start this sequence, the user must have already registered to the DIdMS and also have logged in. The sequence begins with the DID of the service provider (denoted as DIDsp) being provided to the client application. The application then generates a new DID pair for the user and requests the Server to resolve the DIDsp **(4,5)**. The serve resolves the DIDsp to a DDO **(6,7)** and also queries to see the attribute specification that the SP requires [8,9]. All of this is then returned to the client, where the consent is presented for approval **(11,12)**. The Client checks if all of the requested attributes can be matched by an existing schema **(13)** and if there is a miss-match, the form prompts the user to enter manually the missing information. Note, that the auto-matching is started after a user accepts the consent and also selects an Identity context. The matching is then done in relation to the chosen identity context. Once the form is filled in and valid, a verifiable credential is generated and presented as a summary of what will be included as data to the Service provider. The verifiable credential is then put into a verifiable presentation, which in term is included in a bundle. Check 5.4.1 for more clarification. The client then generates a private symmetric key, encrypts the bundle with it and then encrypts the new key with the public key of the Service Provider. The encrypted key and data are then sent to the Server **(16)**. The Server takes the newly generated DID of the user (from **(4)**) and publishes it to the Identity Network. When this is done, the endpoint of that DID is generated and the encrypted data is included there. The user can now take his new DID and give it to the Service Provider who can then resolve it to the fresh endpoint and request the data. Note, only the Service provider can decrypt the data since the key for decryption of the Bundle is encrypted with the Service Provider's public key. By doing this, we ensure that even though the Serve stores the bundle data at the endpoint, it can not access it in plain text.

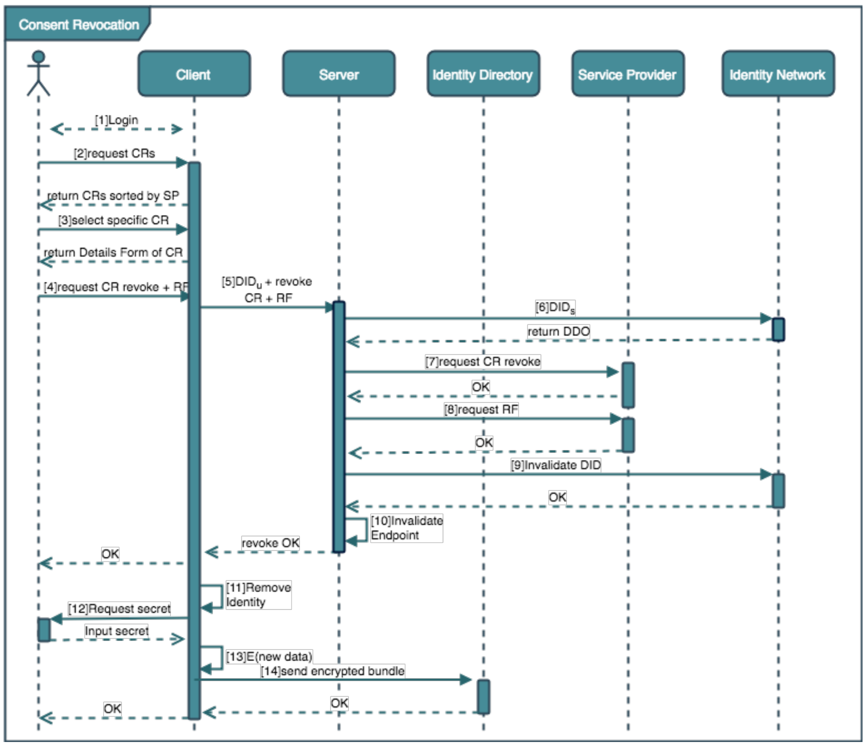


Figure 5.6: Consent Revocation Flow

In figure 5.6 the process of a user revoking their consent is illustrated. The flow depicted in the figure is described earlier in section 4.2.1, and relates to the use cases elicited in figure 4.7.

The initial step **(1)**, and a requirement in order to be able to initiate the process is the user has to be logged in. Afterwards the user can request to view all of their consent receipts which is possible due to the fact that when establishing a new relationship the user saves a copy of the corresponding consent receipt, as stated in section 4.5. That way the system can return all of the consent receipts sorted by service provider. This will allow better overview when inspecting those receipts. The user can then select one of those specific CRs, denoted with action **(3)**. This action returns detailed information of the selected CR along with potential options. The user can then select the option to revoke the specific consent, designated with **(4)**. Additionally, as discussed in section 4.5, the user must be granted the option to select whether they would like to only revoke their consent or also execute the right to be forgotten. In the context of this flow we assume that the user would like to both revoke consent and request all the data associated with them to be deleted. The *Right to be Forgotten* option is abbreviated as **RF** in the figure. Once all of these actions have been executed, the client sends **(5)** the DID_u, which is the identifier that corresponds to the specific SP relationship, discussed in section 4.4. Afterwards the DIdMS feeds the DID_s, corresponding to the specific service, in the DID Resolver, as discussed in sections 4.1.1 and 4.4. The DID resolver is then used to retrieve the DDO that belongs to the DID_s, so that the DIdMS can obtain the endpoints, discussed in section 4.4. Once the process of resolving the DDO of the service provider is complete the DIdMS can initiate action **(7)** which sends a request to the specified endpoint that the user would like to have their consent revoked. Upon receiving an acknowledgement that the request is successful, the system queries another request, this time to the endpoint specified for *Request to be Forgotten* and that way it informs the SP that the user would also like to have all associated data to be permanently deleted. Again the system will await for a success response. Afterwards, once the service has acknowledged that all of the requests have been completed successfully the DIdMS invalidates the DID_u. Once that operation is successful the endpoint corresponding to that DID is also invalidated. The reasoning behind the sequence of the aforementioned operations is that we would like to avoid any unnecessary inconveniences that a potential failure could cause. An potential problem could be to first invalidate the endpoint and then try to invalidate the DID on the Identity Network. If there is a case where the operation to render the DID invalid fails and a service provider decides to reach for the user data at the endpoint and that endpoint has been invalidated earlier that would cause conflicts. After the endpoint has been invalidated the user is given a confirmation that the operations associated with revoking the consent are successful, the partial identity associated with the specific DID_u is also deleted, action **(11)**. Afterwards when the user decides to conclude his session with the client he is asked to input their secret which is then used to encrypt**(13)** the new data. By new data we mean his complete identity in which the previously deleted partial service-specific identity is not present. Then the encrypted bundle is sent to the user-specific identity directory and the process is complete.

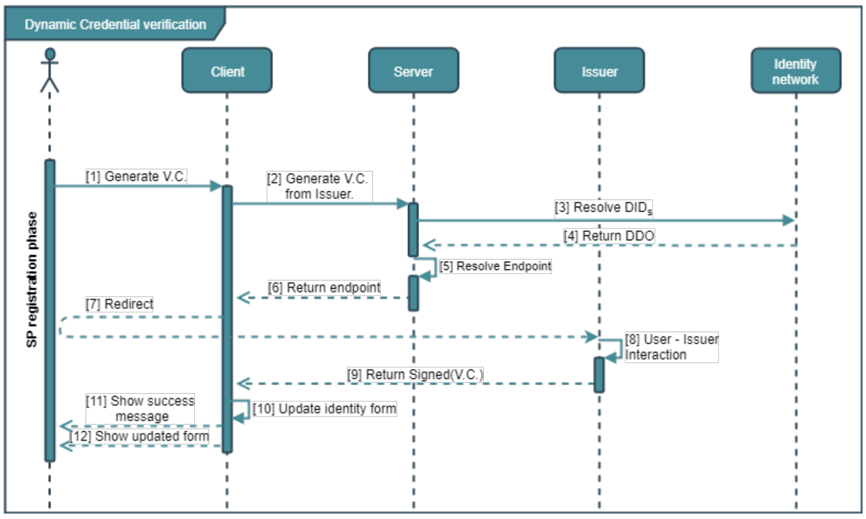


Figure 5.7: Sequence diagram of creating verifiable credential.

In figure 5.7 we have depicted how a user can generate verifiable credential from a specific Issuer whenever such is required by a Service provider. This flow is part of the provisioning of identity to Service provider, showcased in 5.5. After the auto-matching is done in 5.5 the user is presented a form to confirm and fill in missing claims. At this point he can initiate the flow depicted at 5.7. It starts by selecting an issuer and a claim that the user wants to create a verifiable credential for **(1,2)**. The client then makes a request to the server to resolve and tell it where to find that Issuer **(3,4,5,6)**. Once resolved, the client redirects the user to the endpoint of the Issuer where the user has to undergo authentication or other relevant actions specified by the Issuer **(8)**. After the interaction, the Issuer returns a verifiable credential to the client **(9)**, the form is updated **(10)** and the sequence is done.

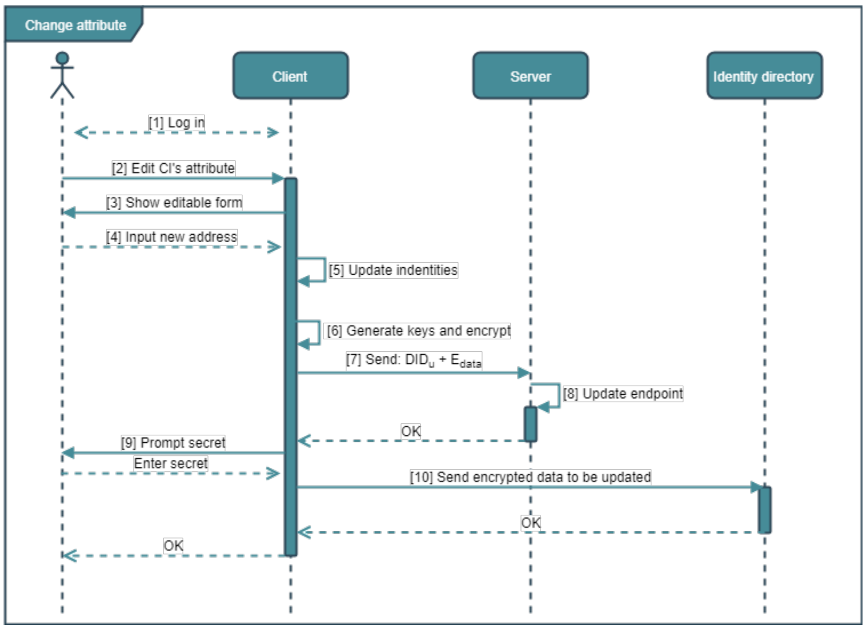


Figure 5.8: Sequence diagram of changing the value of an attribute.

Another important sequence from the perspective of the user is updating or changing his contextual identities. In figure 5.8 we have depicted the process. As a prerequisite the user has to be logged in **(1)**. He then selects to edit a contextual Identity **(2)**. A form is presented and the user can select the specific value from the contextual identity and change it to a desired new **(3,4)**. Since the contextual Identity can be used in partial (service) identities , the partial identities have to be updated as well **(5)**. Updating these partial identities means that the corresponding endpoints at which they reside have to receive an update with the new cyphertext value. The system then generates a new symmetric key. Encrypts the data, encrypts the key with the public key of the service for which the partial identity was created **(6)** and sends the new Bundle to the Server **(7,8)**. In addition, the data has to be also updated at the Identity directory. For that purpose the user is prompted to enter his secret **(9)**. From the secret, the symmetric key for the drive is created and the new data is encrypted and saved **(10)**.

5.4 Frontend

In the Frontend section we will dive deeper into two major sequences. On the mobile device we will display the registration process to our system and in the web section we will investigate the registration of a service within out system.

5.4.1 Class Diagram

As part of the Frontend design, we will present in this section a Class diagram that showcases the domain model of the application. Figure 5.9 reflects the provisioning of identity to a service provider and is only developed in that scope. The class diagram extends the the domain model presented earlier in figure 5.2. This class diagram showcases the major model classes that are needed, in order for the client to provide the processes described in the sequence diagrams.

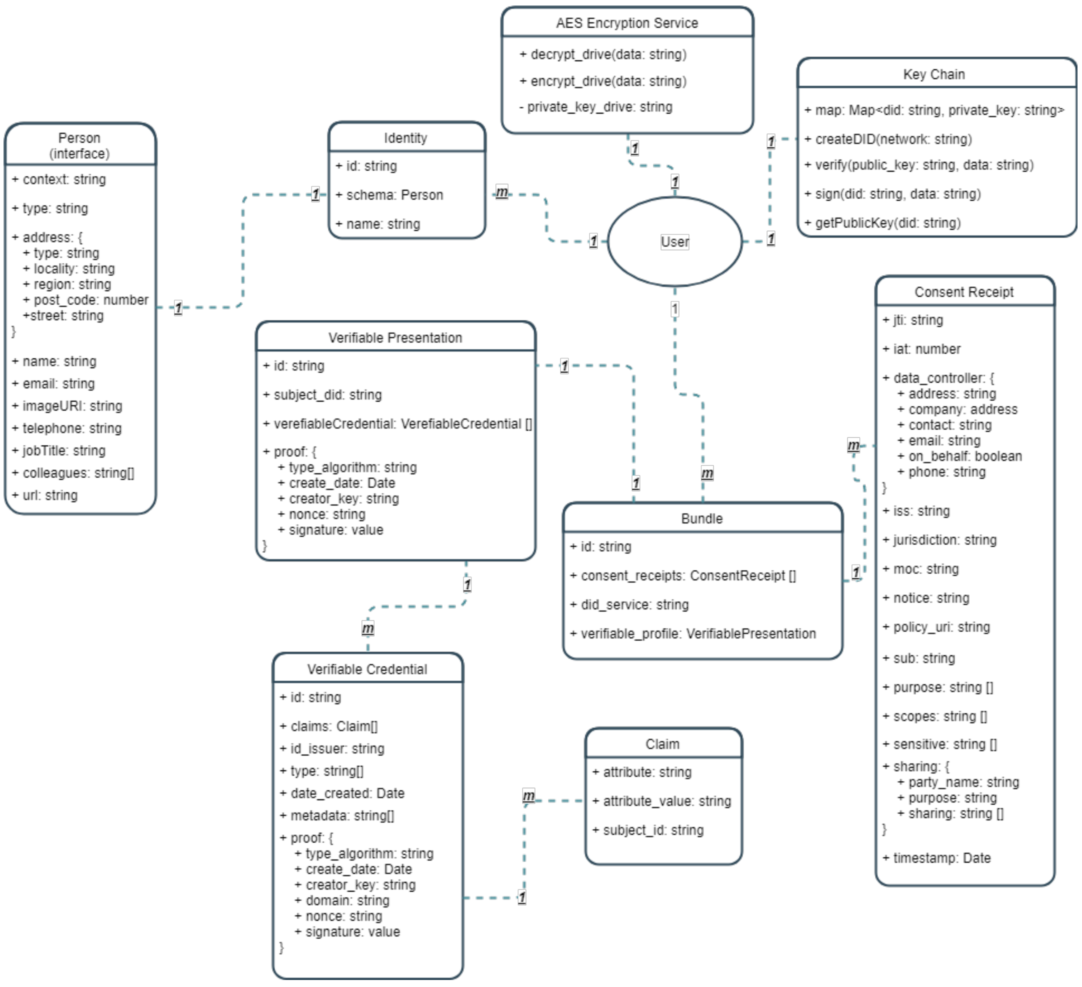

AES Encryption Service
+ decrypt_drive(data: string)
+ encrypt_drive(data: string)
- private_key_drive: string
Key Chain
+ map: Map<did: string, private_key: string>
+ createDID(network: string)
+ verify(public_key: string, data: string)
+ sign(did: string, data: string)
+ getPublicKey(did: string)
Person (interface)
+ context: string
+ type: string
+ address: { + type: string + locality: string + region: string + post_code: number +street: string }
+ name: string
+ email: string
+ imageURI: string
+ telephone: string
+ jobTitle: string
+ colleagues: string[]
+ url: string
Identity
+ id: string
+ schema: Person
+ name: string
User
Consent Receipt
+ jti: string
+ iat: number
+ data_controller: { + address: string + company: address + contact: string + email: string + on_behalf: boolean + phone: string }
+ iss: string
+ jurisdiction: string
+ moc: string
+ notice: string
+ policy_uri: string
+ sub: string
+ purpose: string []
+ scopes: string []
+ sensitive: string []
+ sharing: { + party_name: string + purpose: string + sharing: string [] }
+ timestamp: Date
Verifiable Presentation
+ id: string
+ subject_did: string
+ verefiableCredential: VerefiableCredential []
+ proof: { + type_algorithm: string + create_date: Date + creator_key: string + nonce: string + signature: value }
Bundle
+ id: string
+ consent_receipts: ConsentReceipt []
+ did_service: string
+ verifiable_profile: VerifiablePresentation
Verifiable Credential
+ id: string
+ claims: Claim[]
+ id_issuer: string
+ type: string[]
+ date_created: Date
+ metadata: string[]
+ proof: { + type_algorithm: string + create_date: Date + creator_key: string + domain: string + nonce: string + signature: value }
Claim
+ attribute: string
+ attribute_value: string
+ subject_id: string
1
m

Figure 5.9: Class model

In order to structure the use case of provisioning identity, we start of with the central entity - the User. The user has one Key Chain which is a data structure that stores the ctpyogrpahic keys of his DIDs. The Key Chain also exposes some public API that supports signing and verifying signatures. This promotes encapsulation and centralizes the cryptography operations into a single structure. In addition, we also have an AES Encryption Service that facilitates the encryption of the data that is going to be stored at the Identity directory. The reason for this functionality being outsourced is that Key Chain is part of the data bundle that is going to be encrypted and saved at the drive. AES Encrpytion Service also supports generating private key from password.

Having discussed the cryptography part, we now turn to how the identity and contextual sub-identities will be constructed. As seen in figure 5.9 a user can have one or many Identities. Each Identity has its own name and a schema that describes the structure of attributes that are associated with that identity. In our case we have chosen the Person schema. This is the Contextual Identity structure as described in the analysis, section 4.3.2. Note that the root Identity structure is abstract, hence we do not display it here. Is implicitly part of each contextual identity.

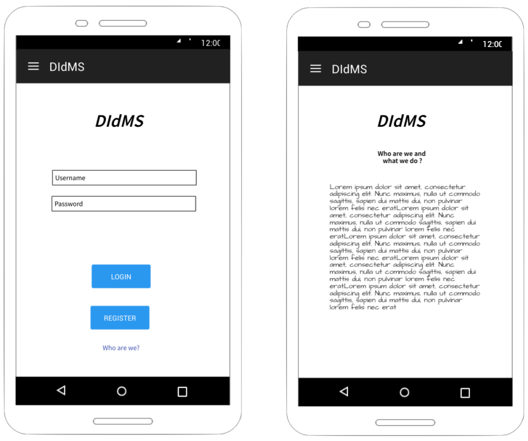
Once the Identities are established, we have to now facilitate the different partial-identities that the user creates for service providers. We start from the Claim structure. Using the Identity schema, claims are created. Once the claims are created, the structure of verifiable credential is generated to encapsulate them and add proof. The proof support integrity and is part of the verifiable credential standard that support validation of origin. Once the Verifiable Credential is done, it is put into a verifiable presentation that could contain multiple verifiable credentials. This the service identity structure from section 4.3.2. In order however, to put the data at the endpoint and enable its use by service providers, we introduce the structure of Bundle. Bundles are stored at endpoints, encrypted and they are the glue between the user and the service provider. A user can have multiple bundles. In addition, the bundle includes the consent receipt for the service provider, since a bundle is generated uniquely for him.

5.4.2 User Interface Design

The following section will present the design of the user interface. We have selected to present the frontend design of the DIdMS in the form of web and phone client. The phone client will represent the registration sequence while the web client will showcase the provisioning of service identity.

Phone

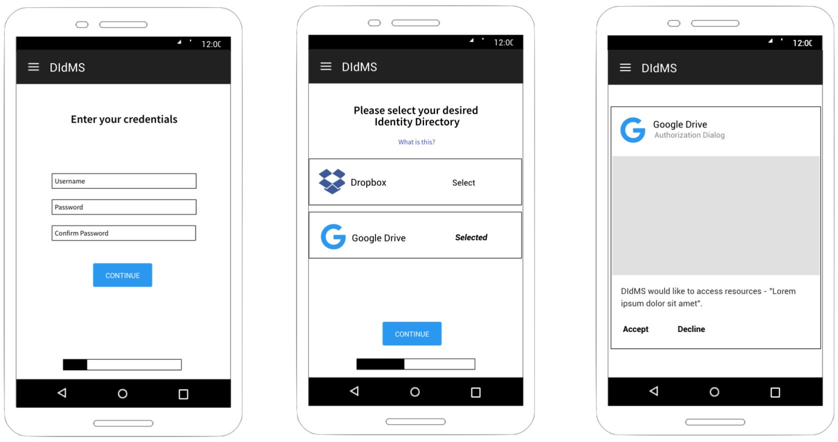
The first platform which is presented is the smartphone application running on the Android OS. The following mock ups of a mobile screen will illustrate what will the user see when they are registering to the DIdMS. All of the actions are in consideration with the flow depicted in figure 5.3. A complete illustration of the created mock ups and further documentation of the individual iterations over the development process of the application are in appendix B.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec eratLorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec eratLorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec eratLorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec eratLorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat

Figure 5.10: Login Screen and System Information

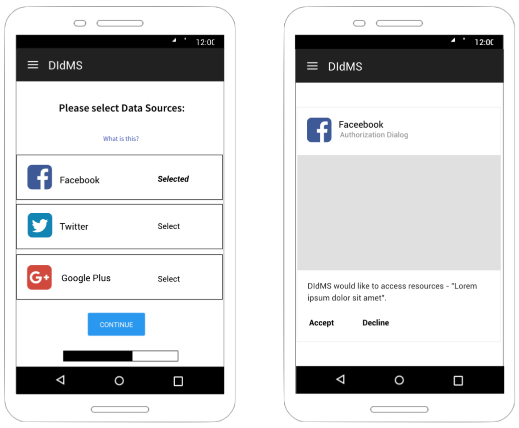
The initial screen that the user will see is going to be is the Login Screen, displayed in figure 5.10. From this point since the individual is not a registered user yet, they have two possible options from this screen. The first option is to press the *REGISTER* button which will redirect them to the registration form, illustrated in figure 5.11. However, they are also able to click on the *Who are we?* button which results in them being transferred to another screen which gives a description of what does the DIdMS give them.



DIdMS would like to access resources - "Lorem ipsum dolor sit amet".

Figure 5.11: Registration Form and IdD Authorization

Once the user are redirected the registration form screen they are prompted to enter a simple username password credentials. After they complete this task they are redirected to the next step which they have to fulfill - select their *Identity Directory*. The user is presented with a list of the available options which can be used as their IdD. Furthermore, there is a small hint button under the instructions label which says *What is this?*. If pressed, the user will be presented a dialog box which will give an explanation of what the Identity directory is and all of the necessary information. An example of how this action looks is again presented in the appendix. In the mock-up, two options are displayed - Google Drive and Dropbox. In this case the user selects to use Google Drive. This redirects him to a screen which displays him information that the DIdMS client would like to be authorized. This is necessary so that the client can automatically create the folder which is needed to be used as a container for the users digital identity.



DIdMS would like to access resources - "Lorem ipsum dolor sit amet".

Figure 5.12: Data Source Selection

Once the user completes the authorization procedure they are redirected to yet another screen with options. However, the options this time are about selecting the Data Source from which they would like to gather their data. The user is presented with a list of options, in the figure 5.12. The available options are the social platforms Facebook, Twitter and Google Plus. The sequence of actions they should do is similar to the previously explained order. Again, for clarification purposes, the user can click on the *What is this?* hint button which will present them with the necessary information about the data sources which are listed. After they have been introduced to the Data Sources they have to select one from which the data that will construct their digital is going to be gathered.

In this case the user clicks on the Facebook platform which then redirects him to the Facebook site where he is prompted with a dialog window. Through this window the user is notified that the DIdMS client would like to access his resources on the social platform. Once they have authorized the client all of their data is gathered, it is presented to the user in a structured manner for them to review and potentially edit if necessary. This is illustrated in figure 5.13. If the users are satisfied with the constructed digital identity they can press the *CONFIRM* button which triggers the encryption of the information, by deriving the key used for the algorithm from their password, as specified in section 5.2. After this action has finished the ciphertext is put in the previously selected folder on the Identity Directory of the user.

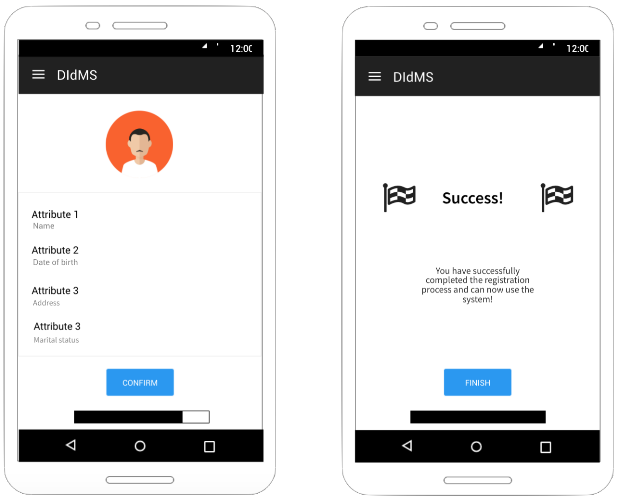


Figure 5.13: Identity presentation

The user is presented with the final screen,illustrated in figure 5.13 , which signifies that the registration flow has concluded and now they can use the mobile application.

Web APP

The following section presents the interface mocking of an application that facilitates the frontend of the DIdMS in the context of a browser. The design is specifically tailored to scenario 2 4.2.1 and showcases the process of registering new identity for a service provider using the DIdMS.

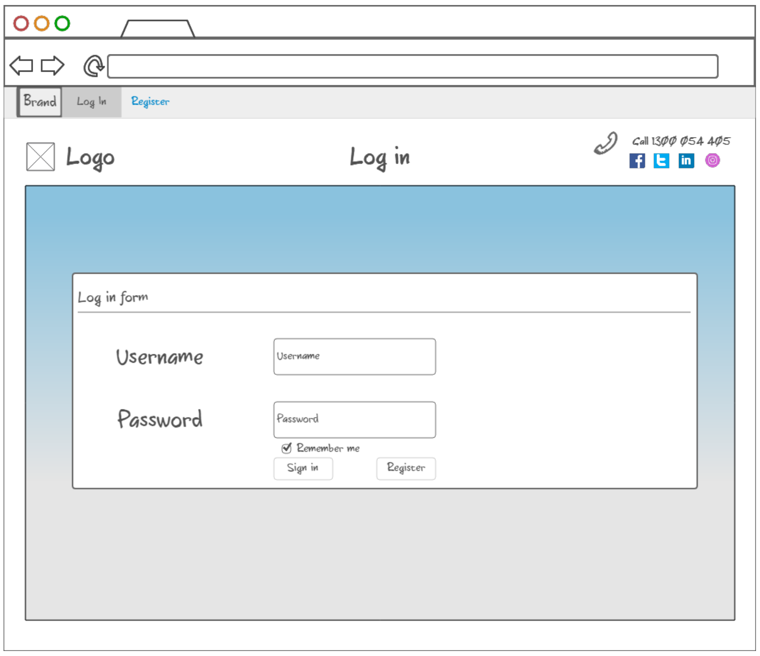
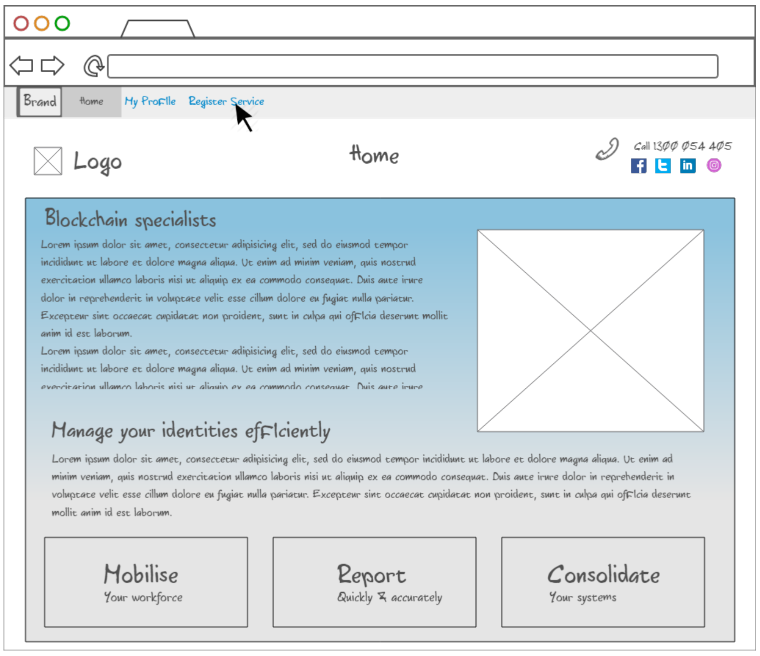


Figure 5.14: Log in screen

In accordance with the sequence of registering a service, we first have to provide a log in interface. The user enters his username and password in the form, presented in figure 5.14. Once logged in the user is redirected to the home screen. But, before redirection, the web app queries the encrypted identity data stored at the identity repository and decrypts it. Once the decryption is successful, the user can continue using the application. The reason for doing this is that the actual identity data is stored at the identity directory. Note, since the user already entered his password for authentication, we use it to generate the private key to decrypt the data.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Figure 5.15: Home screen

In figure 5.15 we present the home. The home screen is just a generic representation of the initial state that the user receives after he is logged in. Once there, the user navigates to the "Register" navigation he begins the process of creating new verifiable profile.

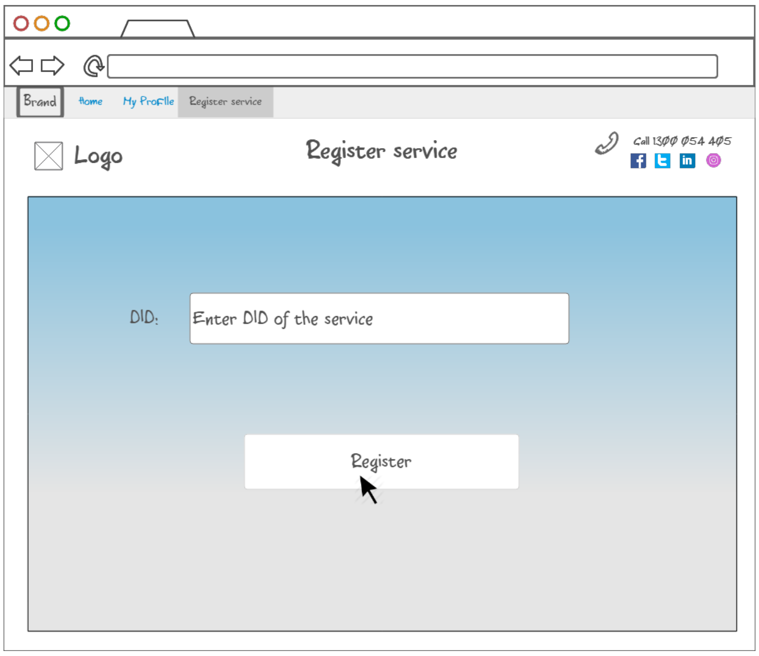
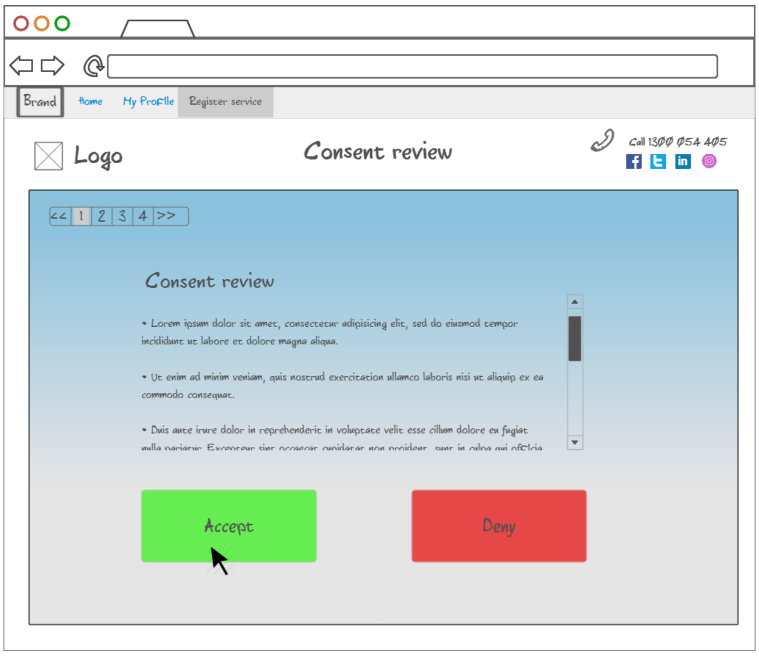


Figure 5.16: Register service home screen

As seen in figure 5.16, to begin the registration, the DID of the service has to be provided. Note, that for showcase purposes we have designed it with a form where the user enters manually the DID. In practice, this part should be automated and done in a more user-friendly way. The user should not be prompted to enter the value of the service DID, instead, there should be a script that does that for him. However, we want to clearly show the beginning and the input required to register a service, thus we have made it more explicit.



• Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

• Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

• Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia

Figure 5.17: Register service Step 1

When register is hit, the web application generated a request to the backend server and sends the value entered by the user. The backend resolves the DID to a DDO and from the DDO it takes the consent receipt specification and returns it to the web client. Note, that the backend, also sends the claims with the corresponding assertion levels that are required to create verifiable profile for the service. This data is also extracted from he service's DDO.

The web client then takes the consent, process it to be displayed clearly to the user and prompts him to accept or deny, presented in figure 5.17. Note, if the DID entered is for some reason invalid, the sequence will be aborted and the user would be returned back to the home screen.

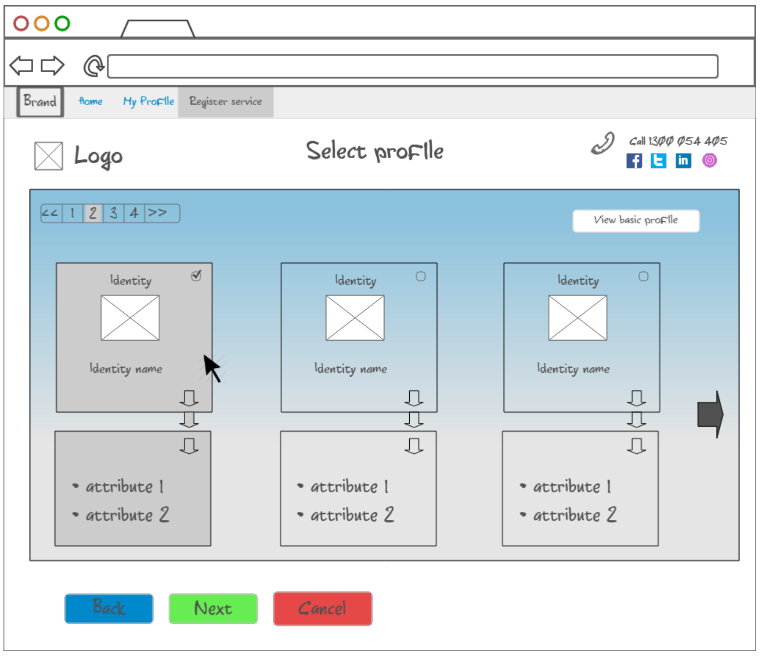


Figure 5.18: Register service Step 2

During step 2, shown in figure 5.18, the web client represents all of the users available identities. The identities will share common schema for defining attributes, but each of them could have unique or same values. For example, the three identities could be "BASIC", "WORK", "HOBBY" where each of them will have an attribute email, but in "BASIC" it is "mypersonal@mail.com", in "WORK" it is "mywork@mail.com" and in "HOBBY" it is again "mypersonal@mail.com". This type of distinction gives the user an option when deciding what identity should be provided to the specific service. Note, that this choice serves more as a template and could be customized even more in step 3.

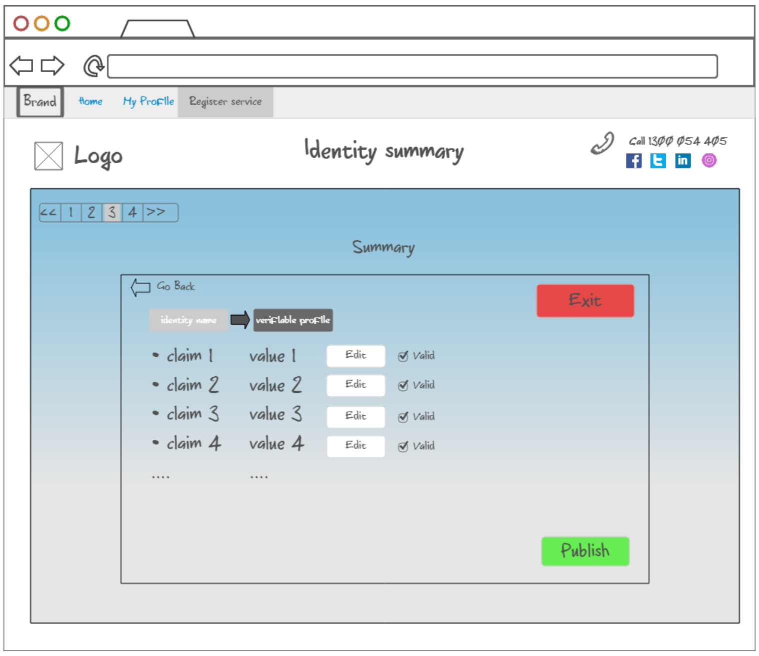


Figure 5.19: Register service Step 3

In figure 5.19 a summary of the new verifiable profile is displayed. Before showing this UI element, the web app automatically extracts the attributes from the chosen identity, matches is to what the service provider requires and if some attributes are missing form the identity, they are created as empty values and prompted for the user to "EDIT" them. The web app automatically checks if all of the values are valid. In addition to this, even automatched claims could be altered. For example, "calim 2" could have been matched to "value 2" which is self-signed by the user. But the user decides that he would like to enter a different value or select one from different identity. He can enter or chose any other value as long as it passes the validation check. Note, the validation may require claims with third-party signatures which is also supported by the web application.

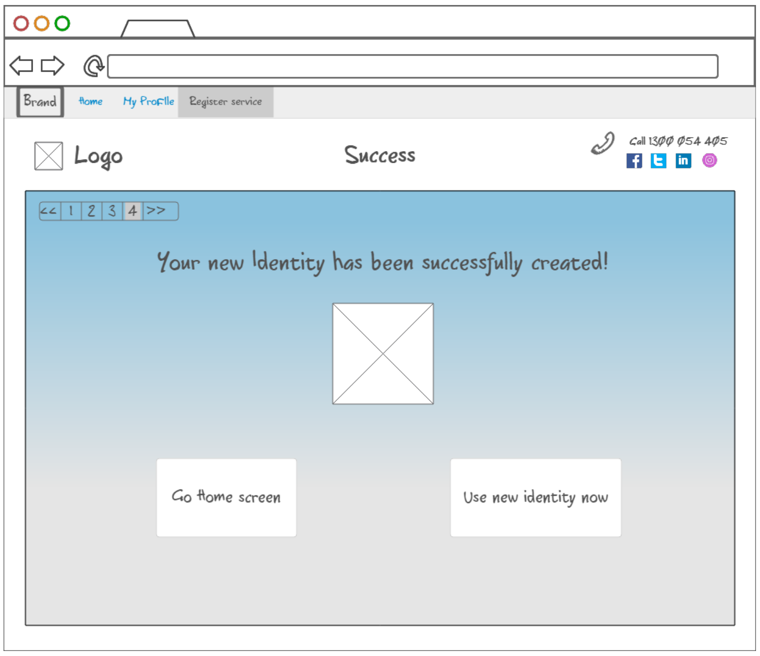


Figure 5.20: Register service Step 4

Once the user is satisfied with his new verifiable profile (data given to the serivce), the web app process it to create a new verifiable profile, adds it to a bundle and sends the bundle to the Backend server. When the Backend server confirms the request, the web app finally updates the identity data of the user by adding the new verifiable profile and prompt his password to encrypt it. Once encrypted the web app connects to the user's identity directory and stores it. This finalizes the process and the user is shown the final screen, presented in figure 5.20.

5.5 Backend

As mentioned in section 5.1, the complexity of the backend is kept low for the developing of the prototype. This means a monolithic approach is taken at this time, but a production version of this could include micro-services and/or other structure required. The design presented in this section will focus on the main functionality of the backend - user management, required APIs and identity network interaction.

The more extensive documentation of the APIs can be found in appendix A, which is the one used for implementation.

5.5.1 Private Web API

As way of communication between the DIdMS' server and client, an internal and private API is used. This API will provide endpoints for DIdMS specific functionality - user authentication and DID/DDO management. Managing of the DID related identity should be happening through the *Identity HUB*, which is explained in section 3.3.3. Some of the endpoints related to the identity hub can be protected by the DIdMS' private API, so only requests by registered users will be handled. The private API is essentially defined by endpoints which only registered identity owner's can use.

The endpoints of the private API should only be accessed by authorized and authenticated users. Since users should not enter user-name and password for each request to the API, a token based authentication approach using JSON Web Tokens(JWT)[75] is chosen. To get a valid access token, valid credentials have to be provided at the access token endpoint. The JWT of type JWS will be signed using a symmetric-key algorithm, meaning the DIdMS server will be able to sign and validate the signature. In the payload of the JWT, the DIdMS user can be defined - e.i. system specific id. By providing the JWT on following requests to necessary endpoints, the API will be able authenticate and identify the user.

Beside the HTTP status codes, some error codes can be defined so that the clients are able to do the right action upon a non 200 HTTP response from the API. This can include showing appropriate message to the user or other appropriate action. For the endpoints documented and implemented for the prototype and time of writing can be seen in appendix A. The more interesting endpoints are presented below:

Creating an Identity (and Hub)

At the endpoint *POST /identity* the clients will be able to inform the server, that the identity owner wants to create a new identity for a specific SP. Since the server will know nothing about the identity owners DIDs and keys, the generation of the new DID is happening on the client. This also means that the server will need all information required for publishing the DID - this can vary depending on the network, but as a minimum the generated DID and associated public key is needed. After publishing the DID, the server will essentially create an *identity hub* for that new identity. Following the identity hub recommendation and the URL path convention, this will mean generation of a new endpoint called */.identity/:new_did* .

5.5.2 Identity Hub

With the identity hub recommendation, the Decentralized Identity Foundation (DIF) try to standardize the sharing if identity related to a DID - see section 3.3.3. As seen above when an identity owner created a partial identity for a SP, a new hub instance is created. This new identity hub is located at */.identity/:new_did* and should support the functionality proposed by the recommendation. The DIF recommendation is at the moment not very mature and do contradict itself on some areas. But some interfaces for the Hub is introduced, and interesting for the current state of this project is:

Profile Is the primary descriptor object of the owning identity and what is referenced in this report as an partial identity. It is also proposed that this object could use the Schema.org Person schema.

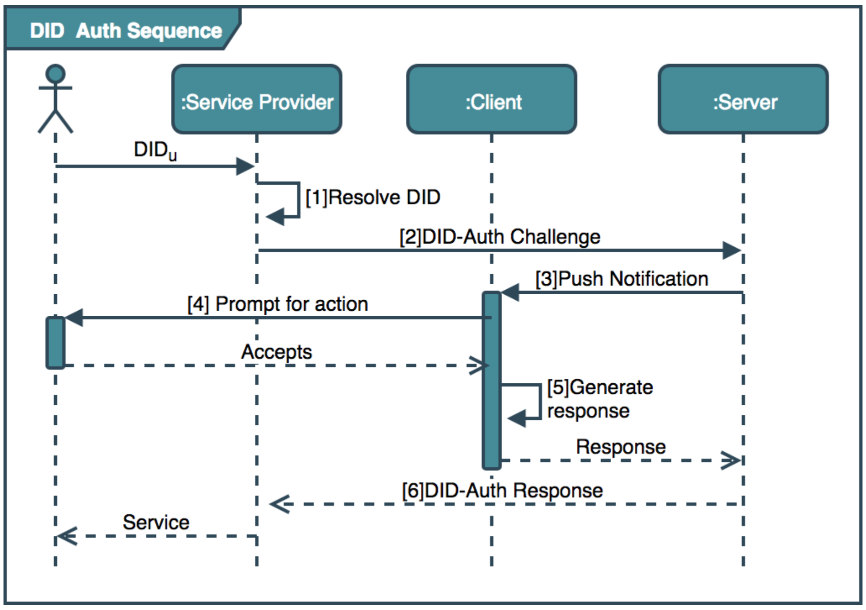


Figure 5.21: Sequence diagram of the DID-Auth challange/response.

Permission Containing the access control JSON document containing *CapabilitySpecifications*. Each capability describing permissions for a certain DID, and what this DID should be able to on identity hub.

The above interfaces should enable the identity owner to have a partial identity and control the SP access to it. All authentication for the requests to the identity hub should be using the DIF/W3C DID Auth schemes, which have been presented.

5.5.3 Public Web API

Beside the API endpoints regarding the identity hub, the DIdMS is required to present another one towards the SPs - at the moment. This specific endpoint is for enabling the identity owner of to prove ownership of a DID. The current endpoint *POST /didauthchallange/did:sov:1234abcd* is intended for a SP to post a challenge which then has to be signed by the user. Since the signing can happen with a private key, the server will need to contact the trusted device of the user and prompt for action. This sequence can be seen in figure 5.21, which happens after the sequence of register service provider shown in figure 5.5.

When the DIdMS have created a new DID for the user, this DID will be provided to the service provider. Upon getting the DID the SP will resolve it and get the associated DDO **(1)**. In the DDO there will be a service endpoint looking like:

Listing 5.1: Example of DID Challenge endpoint

```
{
  ...
  "service":[
    ...,
    {
        "id": "did:sov:1234abcd",
      "type": "DidAuthService",
      "serviceEndpoint": "https://didms.com/didauthchallenge/did:
         sov:1234abcd"
   },
   ...
  ]
}
```

Having the right service endpoint the SP can send a challenge by making a POST request to the endpoint, **(2)**. Two formats of the challenge is proposed using *JWT* or *Verifiable Credential*[71]. After receiving the challenge, the server will notify the client, **(3)**, which could prompt the user for action, **(4)**. The private key for corresponding DID is needed for signing and creating the response, **(5)**. Sending the response back to the server, the server will pass it on to the service provider **(6)** - a callback URL can be present in the initial challenge. Upon a valid response the service provider can provide the requried service to the user.

This sequence is a way for the SP to confirm that the DID that the user has just provided is indeed the one that represents his identity at the hub. This challenge/response eliminates the possibility of a user providing a DID which is not generated or owned by himself.

Chapter 6

Implementation

This chapter will present how the prototype of the *DIdMS* is developed based on the design in chapter 5. We have implemented only two of the sequence diagrams, presented in section 5.3. As a phone app, we have developed the *Registration* sequence. As a web app, we have developed the *Identity provisioning to SP* sequence. Both of these implementations rely on the same backend. The backend implements part of the requirements from the private API but it also mocks-up requests and responses from other external entities.

6.1 Mobile Application

The following section will present the implementation of the DIdMS mobile application client. The structure of the section will follow sequence diagram 5.3. It is important to note that the implementation is done exclusively on Android, in Java programming language.

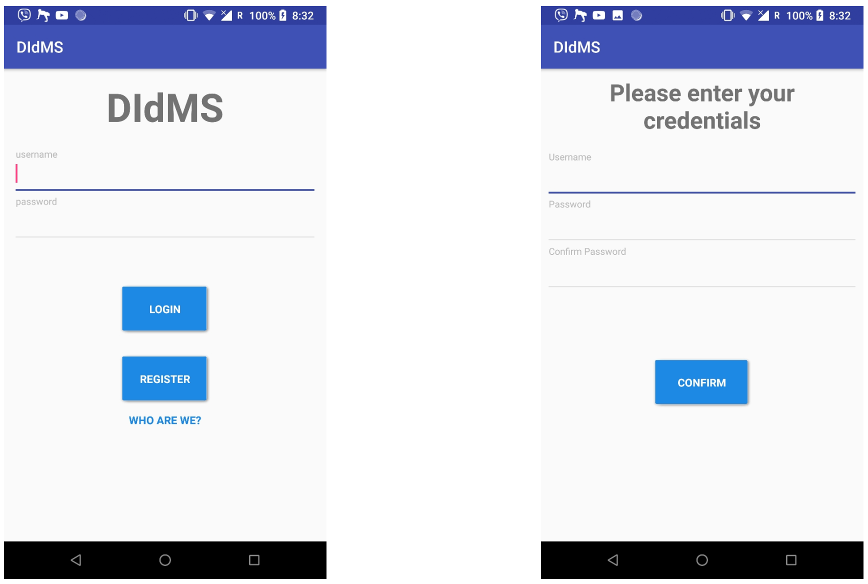


Figure 6.1: DIdMS Home Screen and Registration Form

Once the user fills in the form and the input information is checked. This includes seeing if the username has not been taken by anybody else, checking if both passwords match and if all the conditions are complete a call to the DIdMS's endpoint is done which contains the user information so that the individual can be registered in the database. Following the documentation specified in Appendix A we can see that if the request returns an error code 10 it means that the username already exists.

6.1.1 Registering in the DIdMS

The user is presented with a registration form which has the following fields, shown in figure 6.1.The application provides the necessary feedback in the form of an error message. If the POST request to the server is successful the user proceeds to the Identity Directory selection screen.

6.1.2 Identity Directory Selection

Once the user is at the Identity directory screen he is presented with a list of possible selections, as shown in screenshot 6.2. The only option made available for the prototype is *Google Drive*.

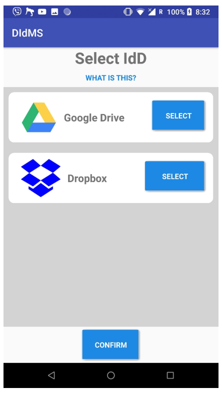


Figure 6.2: Select Identity Directory screen

Google Drive Authorization

In the prototype in order to achieve the functionality of interacting with Google Drive the client will utilize the Google Drive Android API. Afterwards the client needs to be registered in the Google Developers Console in order to create the necessary credentials to facilitate the authorization flow.

Once the client registration is complete the credentials are put in the application's manifest. Once the user taps on the Google Drive Field the authorization flow is initiated where the user is presented with a dialog window in which it is stated that they need to authorize the client to have read/write access to their Google Drive.

The user then has to press to press accept to conclude the authorization process. Now the prototype can create the folder in which the user's identity will be stored.

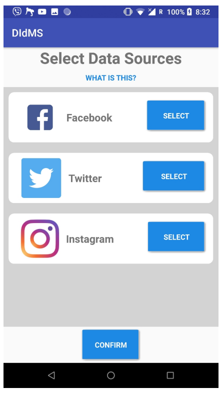


Figure 6.3: Select Data Source screen

Creating root folder

The creation of the identity directory is relatively straightforward, the application uses the builder by the Google Drive API, we specify a root folder which will be later used to store the user's identity. The name of the folder is specified as *Identity Directory*.

6.1.3 Data Source Selection

Once the user have authorized the client and their identity folder has been created, they are presented with the list of potential data sources. However, the only implemented option for this prototype is using *Facebook* as a data source.

Facebook Authorization

When the user clicks on the Facebook option in the list they need to again, similarly to Google Drive, authorize the client. If they have not logged in Facebook a window will pop up which will require them to input their credentials, once the authentication is successful, another window is shown which asks the user to authorize *DIdMS* application.

Data retrieval

After the authorization phase is successful, the necessary user data is pulled from Facebook by using the Facebook Graph API. The data is retrieved in JSON format and is then deserialized before presented to the user. Once that is finished the user's data is going to be presented so that the user can confirm that everything is correct.

6.1.4 Saving DIdMS Data

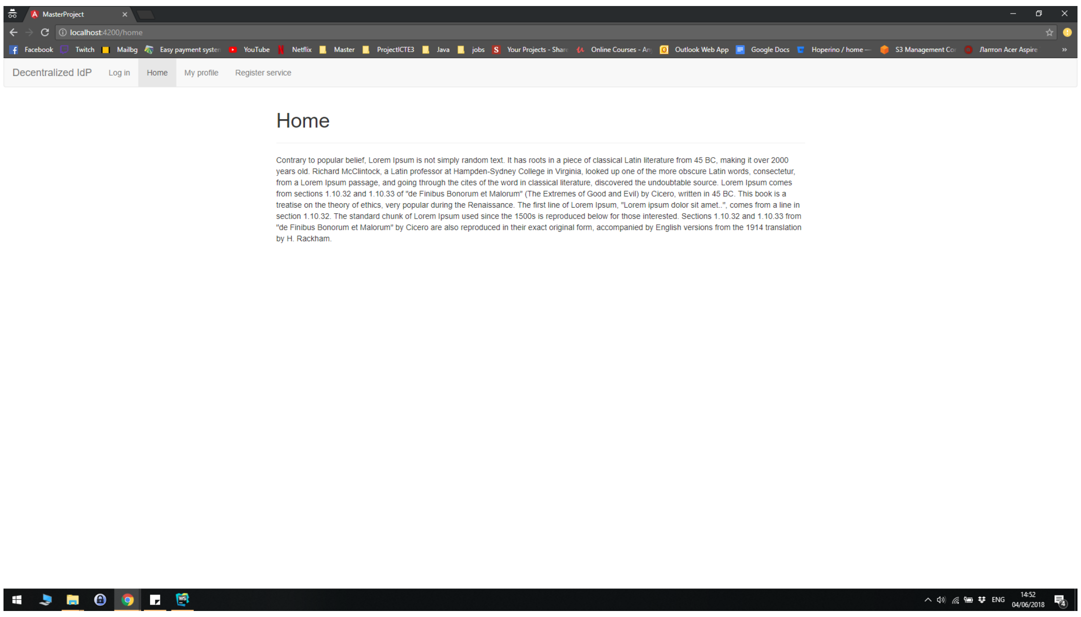
Finally, upon pressing the *Confirm* button, the data is serialized in JSON format again and is encrypted using AES-256, the key of which is going to be derived from the already input secret by the user upon the first screen of registration. The encrypted data is saved as a file in the already created *Identity directory* folder in the user's Google Drive and is also saved to the mobile phone, again as cipher text in order to have it as backup.

6.2 Web Application

To implement scenario 2 (provision of identity to SP) we have opted for a web application. To build the application we have chosen a popular JavaScript framework - Angular 5. In order to showcase the design choices and flow we will omit showing actual code snippets. We have provided screenshots of the application with the console open so that the reader can follow what is happening "under the hood".

6.2.1 Home Ccreen and Login

When the application is opened, the first screen that the user sees is the home view, presented in figure 6.4. In order to use the application, he then has to *log in*. The user navigates to the log in section by clicking the "Log in" button in the navigation bar.



Home

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

Figure 6.4: Home screen

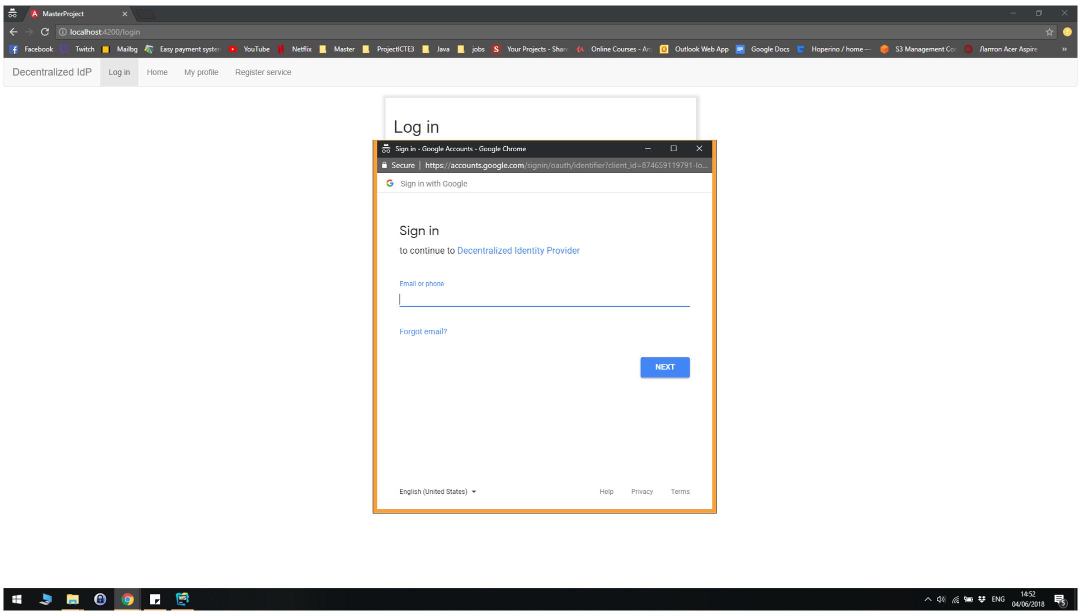
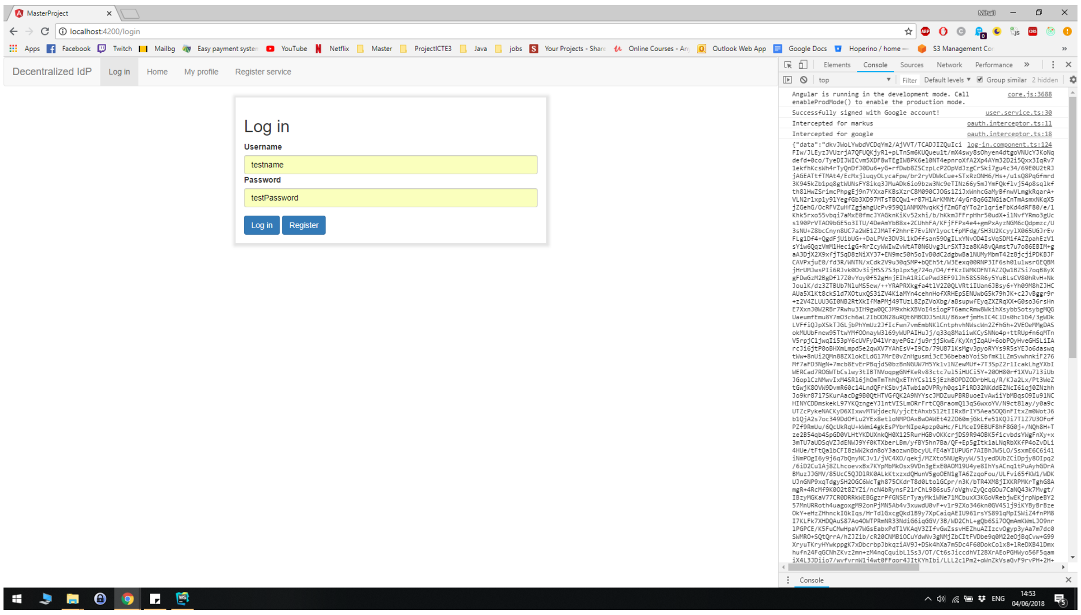


Figure 6.5: Log in screem

In figure 6.5 you can see a simple form of log in. As specified by the sequence of data provisioning 5.5, user's profile data is stored encrypted in his identity directory. After successfully providing his log in credentials, the application prompts the user to provide access to his Google Drive (identity directory). The provisioning requires the user to authenticate and authorize the DIdMS. Once this is done, the application uses the access token to Google Drive and requests his identity data. As seen in the console in figure 6.6, the data is received as a JSON object and it contains the key "data": —encrypted identity—. The app uses the password that user entered for log in in order to generate the key for decryption.

In addition, you can see in figure 6.7 the message that was returned from the backend of the DIdMS containing the access token for the DIdMS.



Log in

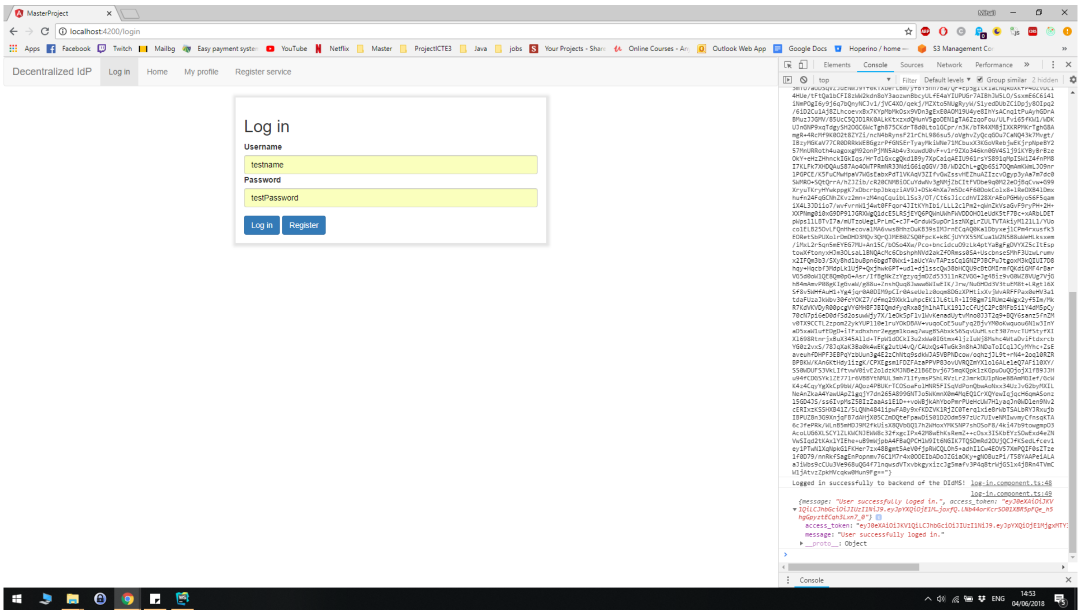
Username: testname

Password: testPassword

Log in | Register

```
Angular is running in the development mode. Call      core.js:3688
enableProdMode() to enable the production mode.
Successfully signed with Google account!              user.service.ts:30
Intercepted for markus                                oauth.interceptor.ts:11
Intercepted for google                                oauth.interceptor.ts:18
{"data":"dkvJWoLYwbdVCDqYm2/AjVVT/TCADJIZQuIci       log-in.component.ts:124
FIw/JLEyzJVUzrjA7QFUQKjyRl+pLTnSm6KUQueu1t/mX4swy8sOhyen4dtgoVNUcYJKoNq
defd+0co/TyeDIJWICvm5XDF8wTEgIW8PK6el0NT4epnroXfA2Xp4AYm32D2i5Qxx3IqRv7
1ekfhKcsWh4rTyQnDfJ0Du6+yG+rfDwb8ZSCzpLcP2OpVdJzgCrSki7gu4c34/69E0U2tRJ
jAGEATtfTMAt4/EcMxjluqyOLycaFpw/br2ryVDWkCue+STxRzONH6/Hs+/u1sQ8PqGfmrd
3K945kZb1pq8gtWUNsFY8ikq3JMuADk6io9bzw3Nc9eTINz66y5mJYmFQkflvj54p8sqlkf
th8lHwZSrimcPhpgEj9n7YXxaFKBsXzrC8M090CJOGs1ZiJxWnhcGaMyBfnwVLmgkRqarA+
VLN2rlxp1y9lYegfGb3XD97MTsTBCQw1+r87HlArKMNt/4yGr8q6GZNGiaCnTmAsmxNKqX5
jZGehG/OcRFVZuHfZgjahgUcPv959Q1ANMXMvqkKjfZmGFqYTo2r1qrieFbKd4dRF80/e/1
Khk5rxo55vbqi7aMxE0fmcJYAGknKiKv52xhi/b/hKkmJFFrpHhr50udX+ilNvfYRmo3gUc
s190PrVTAO9bGE5o3ITU/4DeAmYbB8x+2CUhhFA/KFjFFPx4e4+gmPxAyzNGM6cQdpmzc/U
3sNU+Z8bcCnyn8UC7a2WE1ZJMATf2hhrE7EviNYlyoctfpMFdg/SH3U2Kcyy1X065UGJrEv
FLg1Df4+QgdFjUibUG++DaLPVe3DV3L1kDffsan59OgILxYNvOD4IsVqSDMifAZZpahEzV1
sYiw6QqzVmM1HecigG+RrZcyWWIwZvWtAT0N6Uvg3LrSXT3za8KA8vQAmst7u7o86EBIM+g
aA3DjX2X9xfjTSqD8zNiXY37+EN9mc50h5oIvB0dC2dgbwBa1NUMyMbmT42z8jcjiPDKBJF
[illegible]
```

Figure 6.6: Log in screen 2



Log in

Username: testname

Password: testPassword

Log in | Register

```
[illegible]
BPBKW/KAn6KtHdy1izgK/CPXEgsmIFDZFAzaPPVP836VUVRQZmYX1oI6ALeIeQ/AFI10XY/
SS0WDUFS3VkLIftvwV0ivE2oldzKMJNBe21B6Ebvj675mqKQpk1zKGpuOuQOjojXlfB9JJH
u94fCDGSYklZE77lr6VBBYtNMUL3mh71IfymsPShLRVzLr2JmrkOU1pNoe8BAmMGIef/GcW
K4z4CqyYgXkCp9bW/AQoz4PBUKrTCOSoaFolHNR5FISqVdPonQbwAoNxx34UzJvG2byMXIL
NeAnZkaA4YawUApZ1gqjY7dn265A899GNTJo5WKmnX0m4MqEQ1CrXQYewIqjqcH6qmASonz
15GD4JS/ss6IvpMsZ5BIzZaaAs1E1D++voWBjkAhYboPmrPUeHcUW7HlyaqJn0WDlen9Nv2
cERIxzKSSHXB41Z/5LQNh4841ipwFABy9xfKDZVK1RjZC0Terqlxie8rWbTSALbRYJRxujb
IBPUZ8n3G9XnjqFB7dAHjX05CZmDQteFpawDiS01D2Odm597zUc7UIveNMIwvmyCfnsqKTA
6cJfePRk/WLnB5mHDJ9M2fkUisX8QVbGQ17h2WHoxYMKSNP7shOSoF8/4ki47b9towgmpO3
AcoLUG6XLSCYlZLKWCNJEWW8c32fxgcIPx42M8wEhKsRemZ++cOsx3ISKbEYzSOwExd4eZN
VwSIqd2tKAxlYIEhe+uB9mWjpbA4FBaQPCHlW9It6NGIK7TQSDmRd2OUjQCJfKSedLfcev1
ey1PTwNlXqNpkG1FKHer7zx48Bgmt5AeV0fjpRWCQLOh5+adhIlCw4EOV57XmPQIF0sZTze
1f0D79/nnRkfSagEnPopnmv76C1M7r4x0OOEIbADoJZGiaOKy+gNOBuzPi/T58YAAPeiALA
aJiWbs9cCUu3Ve968uQG4f7lnqwsdVTxvbkgyxizcJg5mafv3P4q8trWjGSlx4jBRn4TVmC
W1jAtvzZpkHVcqkw0Hun9Fg=="}
Logged in successfully to backend of the DIdMS!   log-in.component.ts:48
                                                  log-in.component.ts:49
 {message: "User successfully loged in.", access_token: "eyJ0eXAiOiJKV
▼ 1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1M…joxfQ.LNb44orKcrSO01XBR5pFQe_h5
 hgGpyztECqh3Lxn7_0"}
   access_token: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MjgxMTY
   message: "User successfully loged in."
 ▶ __proto__: Object
>
```

Figure 6.7: Log in screen 3

At this point, the user is logged in the DIdMS and his Google account. The application has two access token respectively and can now begin the sequence of identity provisioning.

6.2.2 Register Identity for Service Provider

In figure 6.8 the provisioning of service DID is depicted. This is the first step of the sequence to register new service identity. The DID entered is not real, it is just a string mock-up.

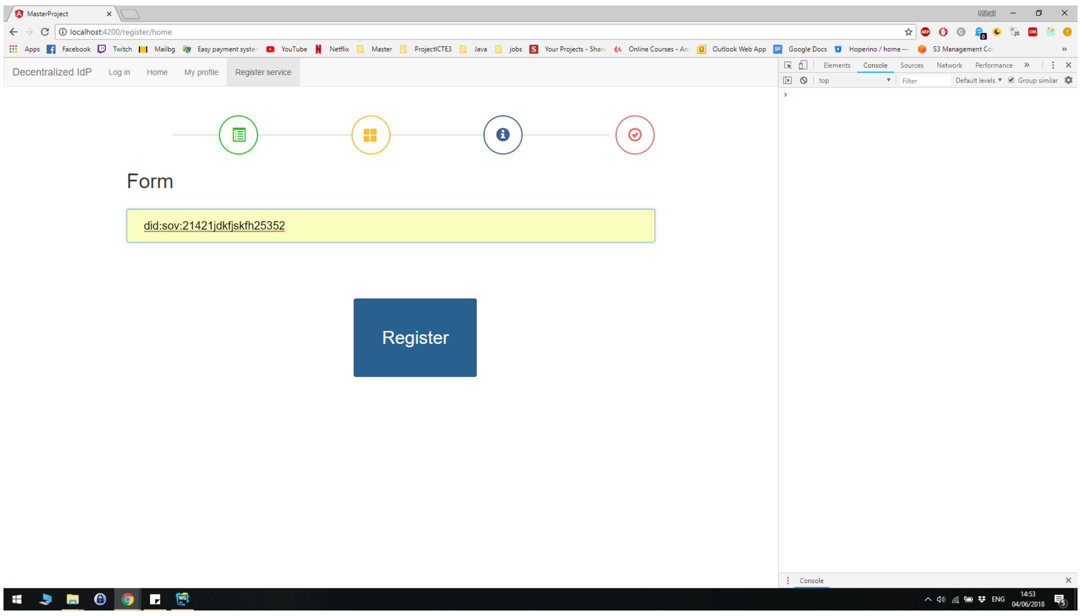
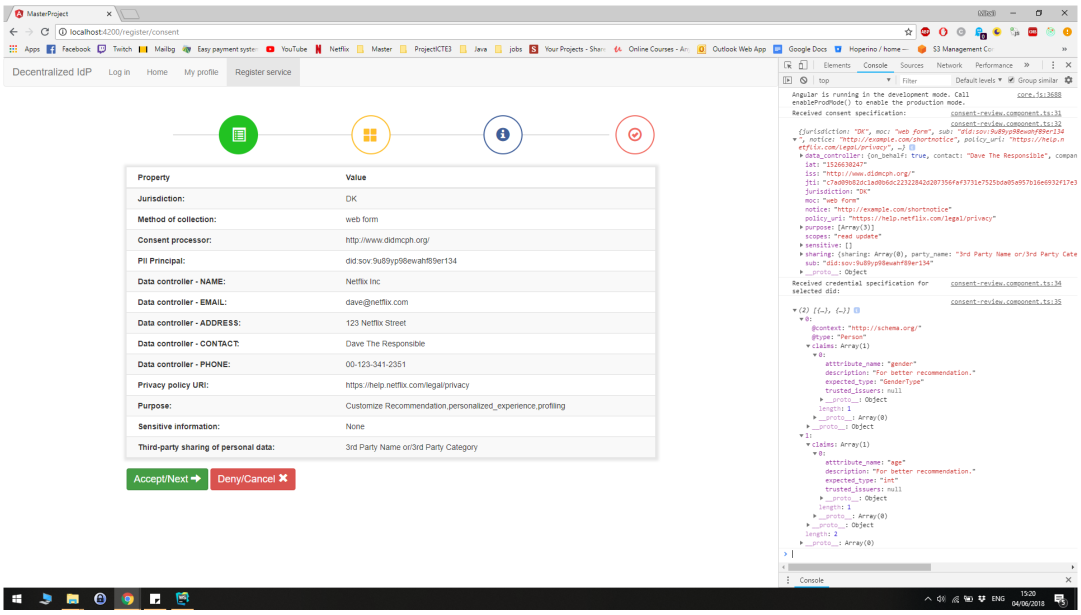


Figure 6.8: Service DID provision screen



Property	Value
Jurisdiction:	DK
Method of collection:	web form
Consent processor:	http://www.didmcph.org/
PII Principal:	did:sov:9u89yp98ewahf89er134
Data controller - NAME:	Netflix Inc
Data controller - EMAIL:	dave@netflix.com
Data controller - ADDRESS:	123 Netflix Street
Data controller - CONTACT:	Dave The Responsible
Data controller - PHONE:	00-123-341-2351
Privacy policy URI:	https://help.netflix.com/legal/privacy
Purpose:	Customize Recommendation,personalized_experience,profiling
Sensitive information:	None
Third-party sharing of personal data:	3rd Party Name or/3rd Party Category

```
Angular is running in the development mode. Call          core.js:3688
enableProdMode() to enable the production mode.
Received consent specification:          consent-review.component.ts:31
                                         consent-review.component.ts:32
  {jurisdiction: "DK", moc: "web form", sub: "did:sov:9u89yp98ewahf89er134
 ▼", notice: "http://example.com/shortnotice", policy_uri: "https://help.n
  etflix.com/legal/privacy", …}
   ▶data_controller: {on_behalf: true, contact: "Dave The Responsible", compan
    iat: "1526630247"
    iss: "http://www.didmcph.org/"
    jti: "c7ad09b82dc1ad0b6dc22322842d207356faf3731e7525bda05a957b16e6932f17e3
    jurisdiction: "DK"
    moc: "web form"
    notice: "http://example.com/shortnotice"
    policy_uri: "https://help.netflix.com/legal/privacy"
   ▶purpose: [Array(3)]
    scopes: "read update"
   ▶sensitive: []
   ▶sharing: {sharing: Array(0), party_name: "3rd Party Name or/3rd Party Cate
    sub: "did:sov:9u89yp98ewahf89er134"
   ▶__proto__: Object
Received credential specification for    consent-review.component.ts:34
selected did:
                                         consent-review.component.ts:35
 ▼(2) [{…}, {…}]
   ▼0:
      @context: "http://schema.org/"
      @type: "Person"
     ▼claims: Array(1)
       ▼0:
          atttribute_name: "gender"
          description: "For better recommendation."
          expected_type: "GenderType"
          trusted_issuers: null
         ▶__proto__: Object
        length: 1
       ▶__proto__: Array(0)
     ▶__proto__: Object
   ▼1:
     ▼claims: Array(1)
       ▼0:
          atttribute_name: "age"
          description: "For better recommendation."
          expected_type: "int"
          trusted_issuers: null
         ▶__proto__: Object
        length: 1
       ▶__proto__: Array(0)
     ▶__proto__: Object
    length: 2
   ▶__proto__: Array(0)
```

Figure 6.9: Consent review screen

Once the users presses "Register", the app simulates a call to resolve the DID to a DDO and then queries the endpoint "/spspec", specified by section A.1.3. The result is that the app receives a specification in the form of Consent receipt that is displayed to the user, presented in figure 6.9, and also receives the claims that the service wants. You can see both in the console, displayed in JSON format. Notice the usage of "@Person" as a schema. This is used to resolve the requested claims to existing ones that the user may already have.

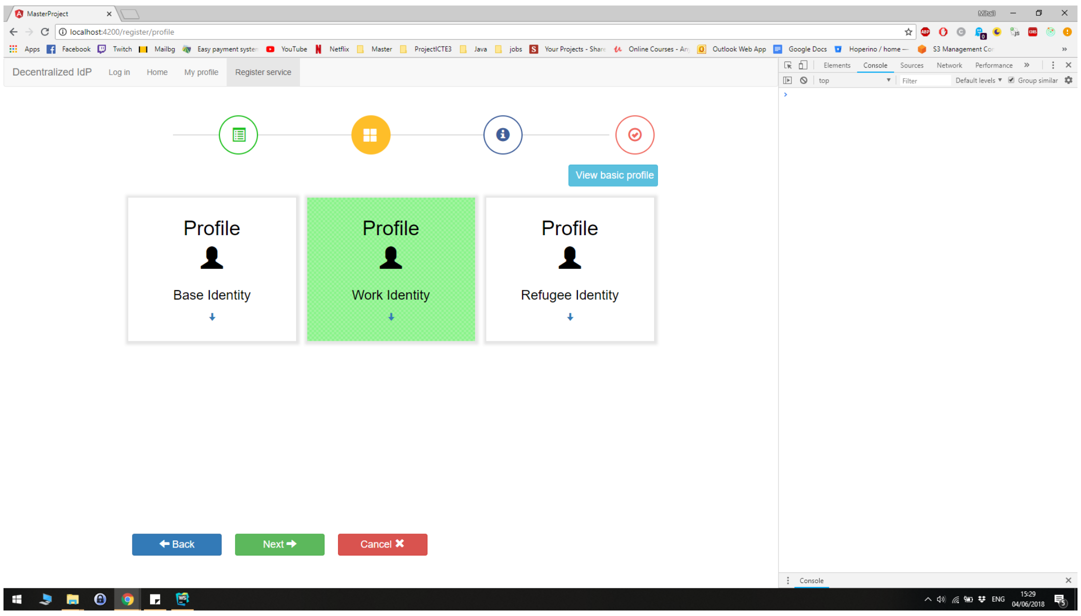
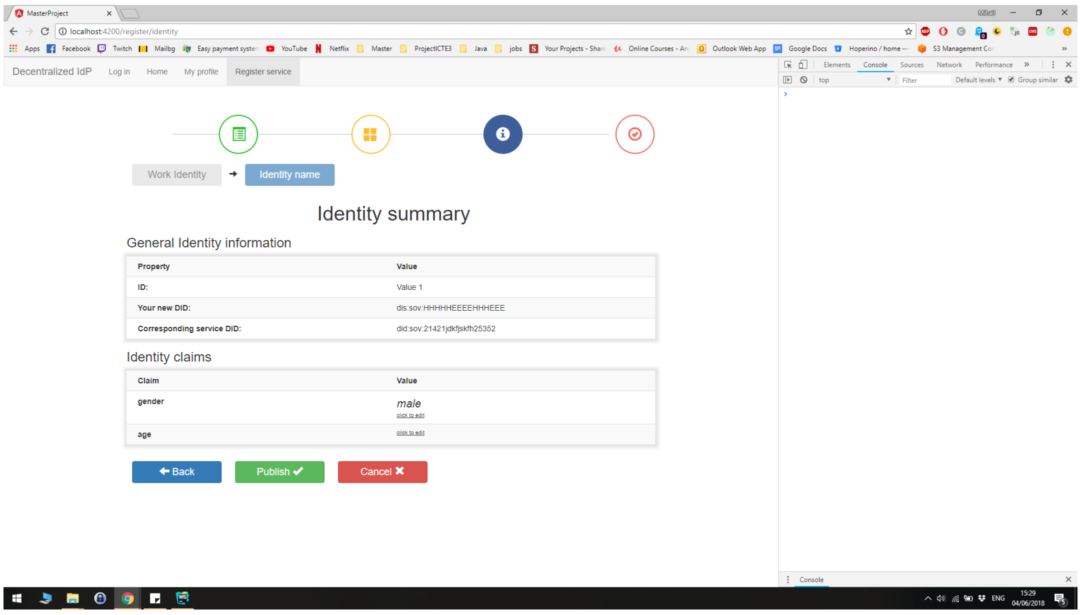


Figure 6.10: Context identity selection screen

Upon accepting the consent, the user is then prompted to select a contextual identity that will be provided to the service, see figure 6.10. From the requested claims we create a new service identity. We look from the chosen contextual identity if there is a match between what is already present (in the form of claims) and what is requested. In our example the user is requested to provide "gender" and "age". Since he selects to use his "Work Identity", we scan the "Work Identity" to see if he has field which correspond to "gender" or "age". The result: "Work Identity" does not contain a field with name "age" but it does have "gender" which value is "male".



Identity summary

General Identity information

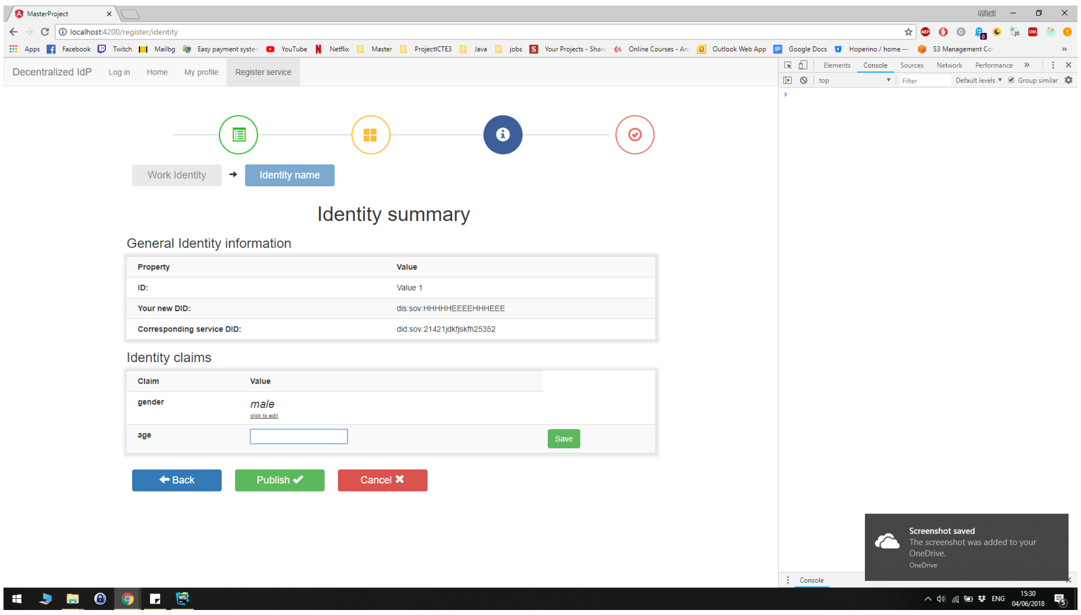
Property	Value
ID:	Value 1
Your new DID:	dis:sov:HHHHHEEEEHHHEEE
Corresponding service DID:	did:sov:21421jdkfjskfh25352

Identity claims

Claim	Value
gender	*male* click to edit
age	click to edit

Figure 6.11: Identity form screen

The application constructs this matching and presents a form to the user where the missing value of "age" has to be provided, see figure 6.11. By clicking on the table, the value of age can be manually inserted, depicted in figure 6.12. In our example, we enter "25" and then press publish to finalize the service identity creation, shown in figure 6.13.



Identity summary

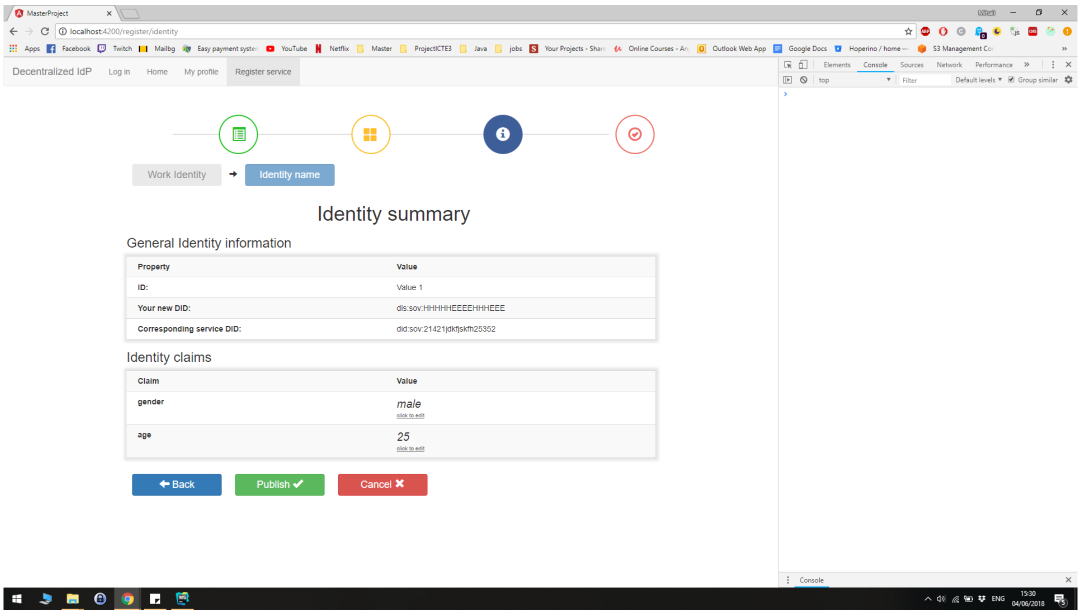
General Identity information

Property	Value
ID:	Value 1
Your new DID:	dis:sov:HHHHHEEEEHHHEEE
Corresponding service DID:	did:sov:21421jdkfjskfh25352

Identity claims

Claim	Value
gender	*male* click to edit
age	

Figure 6.12: Identity form manual input screen



Identity summary

General Identity information

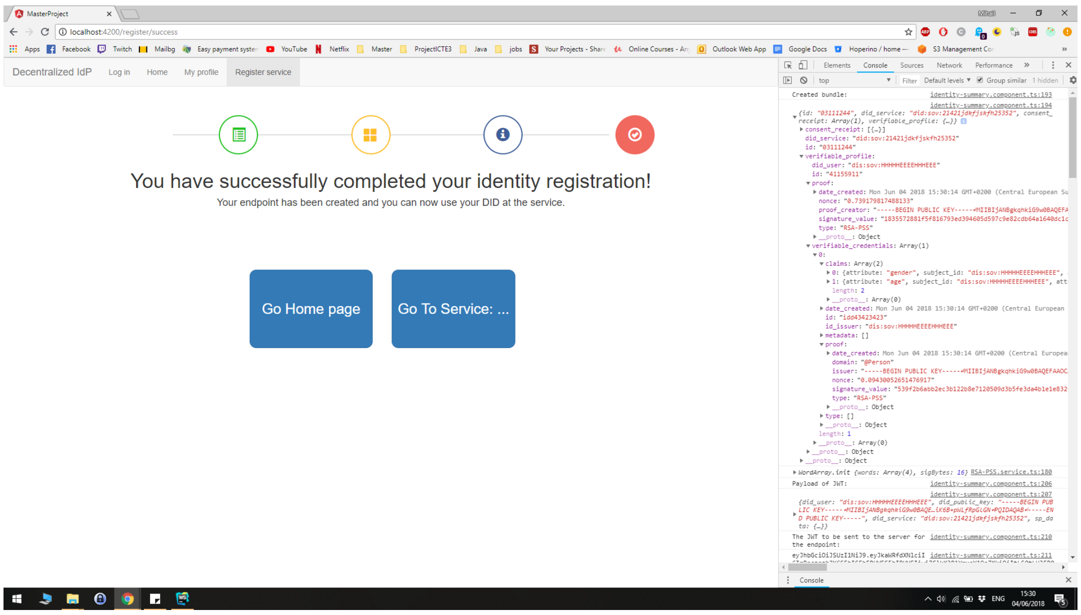
Property	Value
ID:	Value 1
Your new DID:	dis:sov:HHHHHEEEEHHHEEE
Corresponding service DID:	did:sov:21421jdkfjskfh25352

Identity claims

Claim	Value
gender	*male* click to edit
age	*25* click to edit

Figure 6.13: Identity form completed screen

When "Publish" is pressed, the application automatically creates new claims for "age" and "gender" and self-signes them. These claims are then put into a verifiable credential which is put in a verifiable presentation. Afterwards the "Bundle" is created, as specified by the design figure 6.14. The bundle is then encrypted with a freshly made AES 256-bit key, inserted (the bundle) into a JWT and sent to the backend of the DIdMS where it will be put at the endpoint. In addition to the bundle, the AES 256-bit key is also included, but it is encrypted with the public key of the Service provider, depicted in the console in figure 6.15. This concludes, the process and now the user can tell the service provider that his identity is available for access. Note that when we resolved the SPs DID, we simulate the derivation of the public key there.

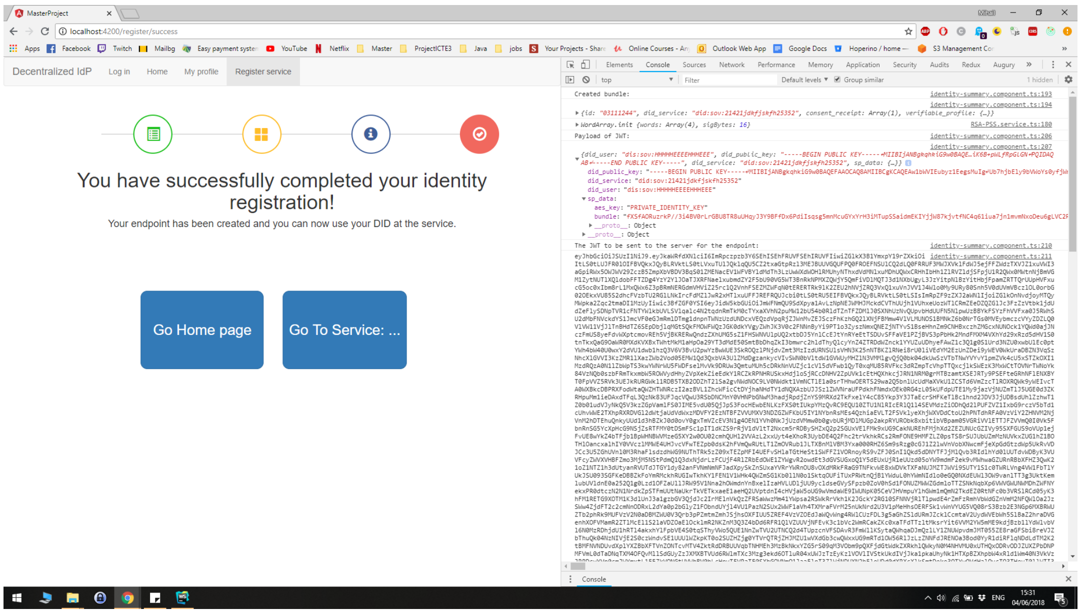


You have successfully completed your identity registration!

Your endpoint has been created and you can now use your DID at the service.

```
Created bundle:                                  identity-summary.component.ts:193
                                                 identity-summary.component.ts:194
▼{id: "03111244", did_service: "did:sov:21421jdkfjskfh25352", consent_
 receipt: Array(1), verifiable_profile: {…}}
  ▶consent_receipt: [{…}]
   did_service: "did:sov:21421jdkfjskfh25352"
   id: "03111244"
  ▼verifiable_profile:
     did_user: "dis:sov:HHHHHEEEEHHHEEE"
     id: "41155911"
    ▼proof:
      ▶date_created: Mon Jun 04 2018 15:30:14 GMT+0200 (Central European Su
       nonce: "0.739179817488133"
       proof_creator: "-----BEGIN PUBLIC KEY-----↵MIIBIjANBgkqhkiG9w0BAQEFA
       signature_value: "1835572881f5f816793ed394605d597c9e82cdb64a1640dc1c
       type: "RSA-PSS"
      ▶__proto__: Object
    ▼verifiable_credentials: Array(1)
      ▼0:
        ▼claims: Array(2)
          ▶0: {attribute: "gender", subject_id: "dis:sov:HHHHHEEEEHHHEEE",
          ▶1: {attribute: "age", subject_id: "dis:sov:HHHHHEEEEHHHEEE", att
           length: 2
          ▶__proto__: Array(0)
        ▶date_created: Mon Jun 04 2018 15:30:14 GMT+0200 (Central European
         id: "idd43423423"
         id_issuer: "dis:sov:HHHHHEEEEHHHEEE"
        ▶metadata: []
        ▼proof:
          ▶date_created: Mon Jun 04 2018 15:30:14 GMT+0200 (Central Europea
           domain: "@Person"
           issuer: "-----BEGIN PUBLIC KEY-----↵MIIBIjANBgkqhkiG9w0BAQEFAAOC
           nonce: "0.09430052651476917"
           signature_value: "539f2b6abb2ec3b122b8e7120509d3b5fe3da4b1e1e832
           type: "RSA-PSS"
          ▶__proto__: Object
        ▶type: []
        ▶__proto__: Object
       length: 1
      ▶__proto__: Array(0)
    ▶__proto__: Object
  ▶__proto__: Object
▶WordArray.init {words: Array(4), sigBytes: 16} RSA-PSS.service.ts:180
Payload of JWT:                                  identity-summary.component.ts:206
                                                 identity-summary.component.ts:207
 {did_user: "dis:sov:HHHHHEEEEHHHEEE", did_public_key: "-----BEGIN PUB
▶LIC KEY-----↵MIIBIjANBgkqhkiG9w0BAQE…iK6B+pWLfRpGLGN↵PQIDAQAB↵-----EN
 D PUBLIC KEY-----", did_service: "did:sov:21421jdkfjskfh25352", sp_da
 ta: {…}}
The JWT to be sent to the server for             identity-summary.component.ts:210
the endpoint:
eyJhbGciOiJSUzI1NiJ9.eyJkaWRfdXNlciI             identity-summary.component.ts:211
```

Figure 6.14: Final summary



You have successfully completed your identity registration!

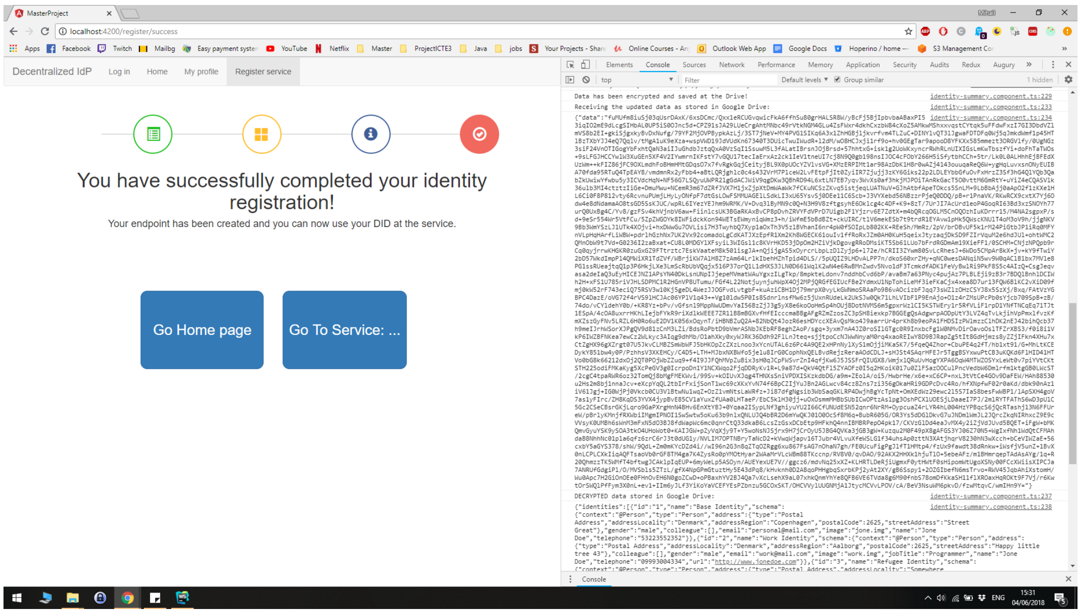
Your endpoint has been created and you can now use your DID at the service.

```
Created bundle:                                                              identity-summary.component.ts:193
                                                                             identity-summary.component.ts:194
▶{id: "03111244", did_service: "did:sov:21421jdkfjskfh25352", consent_receipt: Array(1), verifiable_profile: {…}}
▶WordArray.init {words: Array(4), sigBytes: 16}                                          RSA-PSS.service.ts:180
Payload of JWT:                                                              identity-summary.component.ts:206
                                                                             identity-summary.component.ts:207
▼{did_user: "dis:sov:HHHHHEEEEHHHEEE", did_public_key: "-----BEGIN PUBLIC KEY-----↵MIIBIjANBgkqhkiG9w0BAQE…iK6B+pWLfRpGLGN↵PQIDAQ
 AB↵-----END PUBLIC KEY-----", did_service: "did:sov:21421jdkfjskfh25352", sp_data: {…}}
   did_public_key: "-----BEGIN PUBLIC KEY-----↵MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAw1bWVIEubyz1EegsMuIg↵Ub7hjbEly9bVWoYs0yfjW
   did_service: "did:sov:21421jdkfjskfh25352"
   did_user: "dis:sov:HHHHHEEEEHHHEEE"
  ▼sp_data:
     aes_key: "PRIVATE_IDENTITY_KEY"
     bundle: "fXSfAORuzrkP//3i4BV0rLrGBU8TR8uUHqyJ3Y9BFfDx6PdiIsqsg5mnMcuGYxYrH3iMTupSSaidmEKIYjjW87kjvtfNC4q61iua7jn1mvmNxoDeu6gLVC2F
    ▶__proto__: Object
  ▶__proto__: Object
The JWT to be sent to the server for the endpoint:                           identity-summary.component.ts:210
eyJhbGciOiJSUzI1NiJ9.eyJkaWRfdXNlciI6ImRpczpzb3Y6SEhISEhFRUVFSEhIRUVFIiwiZGlkX3B1YmxpY19rZXkiOi identity-summary.component.ts:211
ItLS0tLUJFR0lOIFBVQkxJQyBLRVktLS0tLVxuTU1JQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSU1CQ2dLQ0FRRUF3MWJXVk1FdWJ5ejFFZWdzTXVJZ1xuVWI3
aGpiRWx5OWJWV29ZczB5ZmpXbVBDV3BqS01ZMENacEV1WFVBYldMdTh3LzUwWXdWOHlRMUhyNThxdVdMNlxuMDhUQWxCRHhIbHh1ZlRVZldjSFpjU1R2QWx0MWtnNjBmVG
M1ZytNUT1XQldobFFTZDg4YzY2Y1JOaTJXRFNaelxubmdZY2F5bU90VG5WT3BnRkNPMXZQWjY5QmFiVDlMQTJ3d1NXbUgyL3JzYitpN1BzYitHbjFpamZRTTQrUUpHVFxu
cG5oc0xIbm8rL1MxQWx6Z3pBRmNERGdmVHViZ25rc1Q2VnhFSEZMZWFqN0tERERTRk91K2ZEU2hNVjZRQ3VxQ1xuVnJVV1J4Wlo0My9URy80Snh5V0dUVmVBczlOL0orbG
02OEkxVU85S2dhcFVzbTU2RGlLNkIrcFdMZlJwR2xHTlxuUFFJREFRQUJcbi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLSIsImRpZF9zZXJ2aWNlIjoiZGlkOnNvdjoyMTQy
MWpka2Zqc2tmaDI1MzUyIiwic3BfZGF0YSI6eyJidW5kbGUiOiJmWFNmQU9SdXpya1AvLzNpNEJWMHJMckdCVThUUjh1VUhxeUozWTlCRmZEeDZQZGlJc3FzZzVtbk1jdU
dZeFlySDNpTVR1cFNTYWlkbUVLSVlqalc4N2tqdnRmTkM0cTYxaXVhN2puMW12bU54b0RldTZnTFZDMlJ0SXNhUzNvQUpvbHdUUFN5NlpwUzBBYkFSYzFhVVFxa0J5RWhS
U2dMbFNVckdYS1JmcVF0eGJmRmlDTmg1dnpnTWNzUzdUNDcxVEQzdVpqRjZJWnMvZEJSczFhKzhGQ21XNjFBMmw4V1VLMUNOS1BMNkZ6b0NrTGs0MVEybmczcVYyZDZLQ0
V1VW11VjJlTnBHdTZ6SEpDbjlqMGtSQkFMOWFWQzJGK0dkYVgyZWhJK3V0c2FNNnByYi9PT1o3ZyszNmxQNEZjNTYvS1BseHhnZm9CNHBxczhZMGcxNUNOck1YQWd0ajJN
czFmUS8yeFdvWXptcmovREh5VjBKRERwQndzZXhUMG5sZ1FHSWNVU1pUQ2xtbDJ5YnlCcEJtYnRYeEtTSDUvSFFaVE1PZjBVS3pPbHk2MndFMXM4VXhYd29xRzd5dHV1S0
tnTkxQaG9OaWR0MXdKVXBxTWhtMkM1aHpOa29YT3dMdE50SmtBbDhqZkI3bmwrc2hldThyQlcyYnZ4ZTRDdWZnck1YYUZuUDhyeFAwZ1c3Q1g0S1Urd3NZU0xwbUlEc0pt
YWh4bW40U0wxY2dVU1dwb1hzQ3V6V3BvU2pwYzBwWUE3SkROQzlPNjdvZmt3MzIzdURNSUlsVHN3K25nNTBKZlRNei8rU0liVEdYM2EzUnZDei9yWEV0WkUraDBZN3VqSz
NhcX1GVVI3KzZMR11XazZWb2Vod05EMW1Qd3QxbVA3U1ZMdDgzankycVIvSWN0bV1tdW1GVWUyMHZlN3VMMlgvQjQ0bk04dkUwSzVTbTNwYVYvY1pmZVk4cU5xSTZkOXI1
MzdROzA0N11ZbWpTS3kwYWNrWU5FWDFse1MvVk9DRUw3QmtuMUh5cDRkNnVUZjc1cV15dVFwb1QyT0xqMU85RVFkc3dRZmpTcVhpTTQxcjlkSWFzK3MxWCtTQVNrTWNoYk
[illegible]
```

Figure 6.15: Final summary screen with JWT containing data for the endpoint

In addition to this, we have also provided a view of how the data in the drive looks. If it is encrypted it has the structure as seen in the console of figure 6.16.

If decrypted it has the structure seen in the console of figure 6.17.

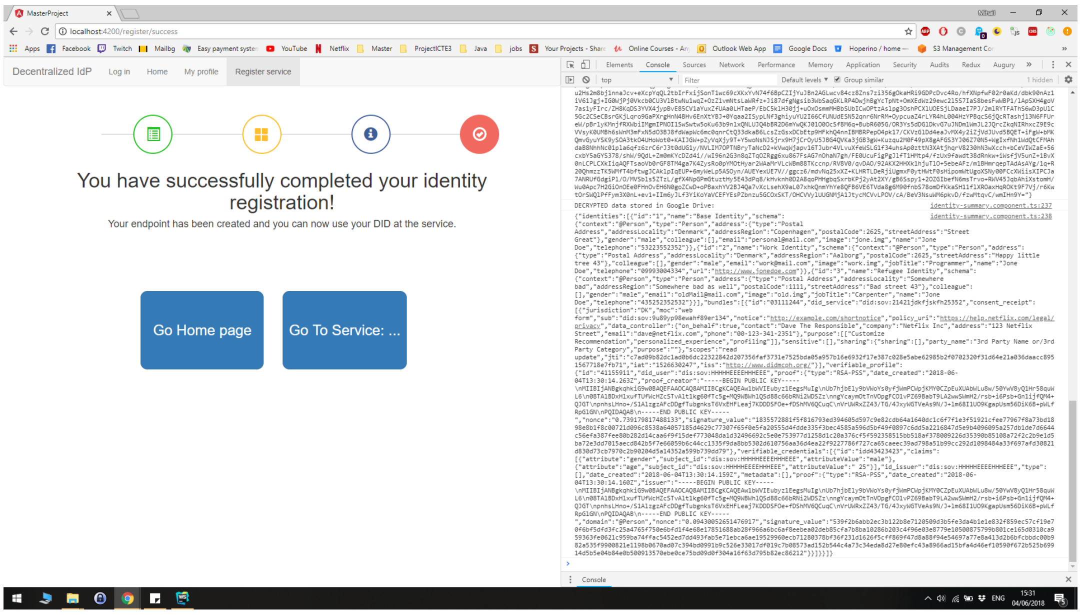


You have successfully completed your identity registration!

Your endpoint has been created and you can now use your DID at the service.

```
Data has been encrypted and saved at the Drive!                  identity-summary.component.ts:229
Receiving the updated data as stored in Google Drive:            identity-summary.component.ts:233
{"data":"fuMUfm8iuSj03qUsrDAxK/6xsDCmc/Qxx1eRCUGvqwicFkA6ffhSu80grHALSR8W/yBcFj5BjIpbvbaABaxPI5   identity-summary.component.ts:234
3iqIO2mE9dLcgSIHbAL0UP5iS0OJnc5d+CPZ91sJA29LUeCrgAhtMNbc49rVtkNGM4GLu4ZsFWxr4dkhCxzbW84cXoZ5AMkwMShxxvqstCYtqk5uFFdwFxzI7GI3DbdVZl
mVS8b2EI+gkiSjgxkyBvDxNufg/79YF2MjOVPBypkAzLj/3ST7jNeV+MY4PVG1SIKq6A3x1ZhHGBjljkvrfvm4TLZuC+DINY1vQT3lJgwaFDTDFq0Wj5qJmkdWmf1p45HT
1BzTXbYJJ4eQ7Qqlv/tMgA1uK9eXza+wspVWD19JdVUdKn67340T3DUicTwuIWudR+12dM/wO8HCJxji1rf9o+hv0GEgTar9apooD8YFKXx585mmezt3ORGVlfy/0UgNGz
3siF24VnOTIGogYbFxhtQaN3aiIJuGhdbJztqQxA0VzSqI1SsuwM5L3fALatIBrsnJOjBrsd+57hhtxG+isk1g2UoWKxyncrRWhRLnUIXIGsLmKwTbszfYi+doFhTaTWOs
+9sLFGJHCCYwlW3XuGEn5XF4V2IYwmrnIKFstY7vGQU17tecIaErxAz2ck1IeV1tneUI7cj8N9Q0gb198nsIJOC4cFObY266H5iSfytbhCCh+5tr/Lk0L0ALHhhEjBFEdX
UzWm++kFIZ86jFC9OXLmdhFoBHmHMtGDqsO7x7fvRgkGqjCeityjBL9X0pUOcYZVlvsVG+XMzERPIMt1ar98AzDbK1H8r0wAZj4143ouuqaReQ6W+ygHqLuvxsnONyEUIB
A70fda95RTuQ4TpEAY8/vmdmnRx2yFbb4+aBtLQRjgh1c0c4s432VrM7PlceW2LvFEtpFjIt0ZyiIR7Zjujj3zKY6Giks22p2LDLEYbbGfuOvFxHrzZ3Sf3hG4Q1YQb3Qa
bZkUwiwYfwbu5y3ICVdcHqN+NF56G7LSQyuUWPR2lgGdACJWiV9qgDKw3QBhRD94L6xtLN7EB7yqv3WvXs0af3hkjMJPOiTAnRkGacT5O0vttM6GmRtY+uYiZ4eCQASVlk
36ulb3MI4ctztzIiGe+DmuMwu+NCemR3m67dZRfJVX7H1jxZjpXtDmWAaWk7fCKuNCSzZKvq5istjeqLUATNuV+GJhAtbfApeTOkcs5SnLM+9LbBbAjj0aApO2f1zKXe1H
L6Ci0F8P812vty6RcvnuPUmjLHyLyONfpF7dtGsLOwFSMMUAGE1LSdkLI3xU65YsvSj0DEe11C6Scb+J3VYXebd56NBzzrPjeQ0DDQ/pB+r1PnaVK/RKwRCX9cxtX7Yj6O
dw4e8dNdamaAO8tsGD5SsKJUC/wpRL6IYezYEJhm9WRMK/V+Dvq31ByMN9c0Q+N3H9V8zftgsyhE6Oklcg4c4DF+K9+8zT/7UrJI7AcUrd1eoP4GoqRI63Bd3xzSNOYh77
urQ0UxBg4C/Yv8/gzFSv4khVjnbV6aw+FiinlcsUK3BGaRKAxBvCP8pDvhZRVYFdVPrD7Uigb2F1Yjzrv6E7ZdtX+m4bQRcqOGLM5CnOQOzhIuKDrrr15/M4NA2sgpxP/s
d+9eSr554Wr5VtFCu/5ZpZWGOYk8IWFidckKon94WETsEWmyniqWmz3+h/iWfmE5bBdBZt+ckUIRC/tlV6mekESb7t9trdR1EYAvw1pHk5QWscKNU1T4oM3oV9h/jjgNKV
[illegible]
u2Hs2m8bj1nnaJcv+eXcpYqQL2tbIrFxijSonT1wc69cXKxYvN74f6BpCZIjYuJBn2AGLwcv84cz8Zns7zi356gOkaHRi9GDPcDvc4Ro/hfXNpfwF02r0aKd/dbk90nAz1
iV61Jgj+IG0WjPj0Vkcb0CU3VlBtwNu1wqZ+OzZ1vmNtsLaWRfz+Ji87dfgNgsib3WbSaqGKLRP4DwjhBgYcTpNt+OmXEdWz29ewc215S7IaS8besFwWBP1/lApSXH4goV
7as1yFIrc/ZH8KqDS3YVX4jypBvE85CV1aYuxZfUAa0LHTaeP/EbC5k1H30jj+uOxOsmmMHBbSUbICwOPtzAslpg3OshPCX1UOESjLDaaeI7PJ/2m1RYTFAThS6wD3pU1C
5Gc2CSeCBsrGKjLqro9GaPXrgHnN4BHv6EnXtYBJ+0Yqaa2ISypLNf3ghiyuYU2I66CfUNUdESN52qnr6NrRM+OypcuaZ4rLYR4hL004HzYPBqcS6jQcRTashj13N6FFUr
eW/pBrlyKMnjfRXWbiIMgmIPNOI1SwSwtw5oKu63b9nlxQNLUJQ4bBR2D6mYwQKJ01O0OcSf8M6q+BubR605G/OR3Ys5dDGlDkvG7uJNDmlWmJL2JQrcZkqNIRhxcZ9E9c
VVsyK0UMBh6sWnM3mFxN5dO3BJ8fdWapWc6mc0qnrCtQ33dkaB6LcsZzGsxDCbEtp9HFkhQ4nnIBMBRPepO4pk17/CKVzGlDd4eaJvMX4y2iZjVdJUvd5BQET+iFgW+bMK
QmvGyuYSK9ySOA3tkO4UHoWot0+KAIJGW+pZyVqXjy9T+Y5woNsNJSjrx9H7jCrOyU5JBG4QVKa3jGB3gW+Kuzqu2M0F49pX8gAFGS3YJ06Z70N5+WgIxfNh1WdQtCFMAh
daB8NhhNc01pla6qfz6zrC6rJ3t0dUG1y/NVLIM7OPTNBryTaNcD2+kVwqWjapv16TJubr4VLvuXfeWSLG1f34uhsAp0zttN3XAtjhqrV8230hN3wXcch+bCeVIWZaE+56
cxbY5aGYS378/shW/9QdL+Zm0mKYcDZd4i//wI96n2G3n8qZTqOZRgg6xu867FsAG7nOhaN7gh/FE0UcuFigPgJlfT1HMtp4/fzUx9fawdt38dRnkw+iWsfjV5unZ+1BvX
0nLCPLCXkIiqAQFTsaoVb0rGF8TM4ga7K4ZysRo0pYMOtHyar2WAaMrVLcWBm88TKccnp/RV8V0/qvDAO/92AKX2HHXk1hjuTlO+5ebeAFz/m1BHmrqepTAdAsAYg/1q+R
20QhmzzTK5WMfT4bftwgJCAklpIqEUP+6myWeLp5ASOyn/AUEYexUE7V//ggcz6/mdvNq25xXZ+KLHRTLDeRjiUgmxF0ytHWtF0sHipomWtUgoXSNy00FCcXWiisXIPCJa
7ANRUfGdgiPl/O/MVSbls5ZTzL/gfX4NpGPmGtuztHy5E43dPq8/kHvknh0D2A8qoPHHgbqSxrbKPj2yAt2XY/gB6Sspy1+2OZGIbefN6msTrvo+RWV45JqbAhiXstomH/
Wu0Apc7H2GiOnOEe0FHnOvEH6N0goZCwD+oPBaxhYV2BJ4Qa7vXcLsehX9aL07xhkQnmYhYe8QFB6VE6TVda8g6M90fnbS78omDfKkaSH11flXROaxHqROKt9F7Vj/r6Kw
tOrSWQlPfFym3X0nL+ev1+IIm6yJLf3YiKoYaVCEFYEsPZbnzu5GCOxSKT/OHCVVylUUGNMjA1JtycMCVvLPOV/cA/BeV3NsuWM6pkvD/fzwMtqvC/wmIHn9Y="}
DECRYPTED data stored in Google Drive:                           identity-summary.component.ts:237
{"identities":[{"id":"1","name":"Base Identity","schema":        identity-summary.component.ts:238
{"context":"@Person","type":"Person","address":{"type":"Postal
Address","addressLocality":"Denmark","addressRegion":"Copenhagen","postalCode":2625,"streetAddress":"Street
Great"},"gender":"male","colleague":[],"email":"personal@mail.com","image":"jone.img","name":"Jone
Doe","telephone":"53223552352"}},{"id":"2","name":"Work Identity","schema":{"context":"@Person","type":"Person","address":
{"type":"Postal Address","addressLocality":"Denmark","addressRegion":"Aalborg","postalCode":2625,"streetAddress":"Happy little
tree 43"},"colleague":[],"gender":"male","email":"work@mail.com","image":"work.img","jobTitle":"Programmer","name":"Jone
Doe","telephone":"09993004334","url":"http://www.jonedoe.com"}},{"id":"3","name":"Refugee Identity","schema":
{"context":"@Person","type":"Person","address":{"type":"Postal Address","addressLocality":"Somewhere
```

Figure 6.16: Final summary screen where drive is encrypted



You have successfully completed your identity registration!

Your endpoint has been created and you can now use your DID at the service.

```
u2Hs2m8bj1nnaJcv+eXcpYqQL2tbIrFxijSonT1wc69cXKxYvN74f6BpCZIjYuJBn2AGLwcv84cz8Zns7zi356gOkaHRi9GDPcDvc4Ro/hfXNpfwF02r0aKd/dbk90nAz1
iV61Jgj+IG0WjPj0Vkcb0CU3VlBtwNu1wqZ+OzZ1vmNtsLaWRfz+Ji87dfgNgsib3WbSaqGKLRP4DwjhBgYcTpNt+OmXEdWz29ewc215S7IaS8besFwWBP1/lApSXH4goV
7as1yFIrc/ZH8KqDS3YVX4jypBvE85CV1aYuxZfUAa0LHTaeP/EbC5k1H30jj+uOxOsmmMHBbSUbICwOPtzAslpg3OshPCX1UOESjLDaaeI7PJ/2m1RYTFAThS6wD3pU1C
5Gc2CSeCBsrGKjLqro9GaPXrgHnN4BHv6EnXtYBJ+0Yqaa2ISypLNf3ghiyuYU2I66CfUNUdESN52qnr6NrRM+OypcuaZ4rLYR4hL004HzYPBqcS6jQcRTashj13N6FFUr
eW/pBrlyKMnjfRXWbiIMgmIPNOI1SwSwtw5oKu63b9nlxQNLUJQ4bBR2D6mYwQKJ01O0OcSf8M6q+BubR605G/OR3Ys5dDGlDkvG7uJNDmlWmJL2JQrcZkqNIRhxcZ9E9c
VVsyK0UMBh6sWnM3mFxN5dO3BJ8fdWapWc6mc0qnrCtQ33dkaB6LcsZzGsxDCbEtp9HFkhQ4nnIBMBRPepO4pk17/CKVzGlDd4eaJvMX4y2iZjVdJUvd5BQET+iFgW+bMK
QmvGyuYSK9ySOA3tkO4UHoWot0+KAIJGW+pZyVqXjy9T+Y5woNsNJSjrx9H7jCrOyU5JBG4QVKa3jGB3gW+Kuzqu2M0F49pX8gAFGS3YJ06Z70N5+WgIxfNh1WdQtCFMAh
daB8NhhNc01pla6qfz6zrC6rJ3t0dUG1y/NVLIM7OPTNBryTaNcD2+kVwqWjapv16TJubr4VLvuXfeWSLG1f34uhsAp0zttN3XAtjhqrV8230hN3wXcch+bCeVIWZaE+56
cxbY5aGYS378/shW/9QdL+Zm0mKYcDZd4i//wI96n2G3n8qZTqOZRgg6xu867FsAG7nOhaN7gh/FE0UcuFigPgJlfT1HMtp4/fzUx9fawdt38dRnkw+iWsfjV5unZ+1BvX
0nLCPLCXkIiqAQFTsaoVb0rGF8TM4ga7K4ZysRo0pYMOtHyar2WAaMrVLcWBm88TKccnp/RV8V0/qvDAO/92AKX2HHXk1hjuTlO+5ebeAFz/m1BHmrqepTAdAsAYg/1q+R
20QhmzzTK5WMfT4bftwgJCAklpIqEUP+6myWeLp5ASOyn/AUEYexUE7V//ggcz6/mdvNq25xXZ+KLHRTLDeRjiUgmxF0ytHWtF0sHipomWtUgoXSNy00FCcXWiisXIPCJa
7ANRUfGdgiPl/O/MVSbls5ZTzL/gfX4NpGPmGtuztHy5E43dPq8/kHvknh0D2A8qoPHHgbqSxrbKPj2yAt2XY/gB6Sspy1+2OZGIbefN6msTrvo+RWV45JqbAhiXstomH/
Wu0Apc7H2GiOnOEe0FHnOvEH6N0goZCwD+oPBaxhYV2BJ4Qa7vXcLsehX9aL07xhkQnmYhYe8QFB6VE6TVda8g6M90fnbS78omDfKkaSH11flXROaxHqROKt9F7Vj/r6Kw
tOrSWQlPfFym3X0nL+ev1+IIm6yJLf3YiKoYaVCEFYEsPZbnzu5GCOxSKT/OHCVVylUUGNMjA1JtycMCVvLPOV/cA/BeV3NsuWM6pkvD/fzwMtqvC/wmIHn9Y="}
DECRYPTED data stored in Google Drive:                           identity-summary.component.ts:237
{"identities":[{"id":"1","name":"Base Identity","schema":        identity-summary.component.ts:238
{"context":"@Person","type":"Person","address":{"type":"Postal
Address","addressLocality":"Denmark","addressRegion":"Copenhagen","postalCode":2625,"streetAddress":"Street
Great"},"gender":"male","colleague":[],"email":"personal@mail.com","image":"jone.img","name":"Jone
Doe","telephone":"53223552352"}},{"id":"2","name":"Work Identity","schema":{"context":"@Person","type":"Person","address":
{"type":"Postal Address","addressLocality":"Denmark","addressRegion":"Aalborg","postalCode":2625,"streetAddress":"Happy little
tree 43"},"colleague":[],"gender":"male","email":"work@mail.com","image":"work.img","jobTitle":"Programmer","name":"Jone
Doe","telephone":"09993004334","url":"http://www.jonedoe.com"}},{"id":"3","name":"Refugee Identity","schema":
{"context":"@Person","type":"Person","address":{"type":"Postal Address","addressLocality":"Somewhere
bad","addressRegion":"Somewhere bad as well","postalCode":1111,"streetAddress":"Bad street 43"},"colleague":
[],"gender":"male","email":"oldMail@mail.com","image":"old.img","jobTitle":"Carpenter","name":"Jone
Doe","telephone":"435252352532"}}],"bundles":[{"id":"03111244","did_service":"did:sov:21421jdkfjskfh25352","consent_receipt":
[{"jurisdiction":"DK","moc":"web
form","sub":"did:sov:9u89yp98ewahf89er134","notice":"http://example.com/shortnotice","policy_uri":"https://help.netflix.com/legal/
privacy","data_controller":{"on_behalf":true,"contact":"Dave The Responsible","company":"Netflix Inc","address":"123 Netflix
Street","email":"dave@netflix.com","phone":"00-123-341-2351"},"purpose":[["Customize
Recommendation","personalized_experience","profiling"]],"sensitive":[],"sharing":{"sharing":[],"party_name":"3rd Party Name or/3rd
Party Category","purpose":""},"scopes":"read
update","jti":"c7ad09b82dc1ad0b6dc22322842d207356faf3731e7525bda05a957b16e6932f17e387c028e5abe62985b2f0702320f31d64e21a036daacc895
1567718e7fb71","iat":"1526630247","iss":"http://www.didmcph.org/"}],"verifiable_profile":
{"id":"41155911","did_user":"dis:sov:HHHHHEEEEHHHEEE","proof":{"type":"RSA-PSS","date_created":"2018-06-
04T13:30:14.263Z","proof_creator":"-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAw1bWVIEubyz1EegsMuIg\nUb7hjbEly9bVWoYs0yfjWmPCWpjKMY0CZpEuXUAbWLu8w/50YwV8yQ1Hr58quW
L6\n08TAlBDxHlxufTUfWcHZcSTvAlt1kg60fTc5g+MQ9WBWhlQSd88c66bRNi2WDSZz\nngYcaymOtTnVOpgFCO1vPZ69BabT9LA2wwSWmH2/rsb+i6Psb+Gn1ijfQM4+
QJGT\npnhsLHno+/S1AlzgzAFcDDgfTubgnksT6VxEHFLeaj7KDDDSFOe+fDShMV6QCuqC\nVrUWRxZZ43/TG/4JxyWGTVeAs9N/J+lm68I1UO9KgapUsm56DiK6B+pWLf
RpGlGN\nPQIDAQAB\n-----END PUBLIC KEY-----
","nonce":"0.739179817488133","signature_value":"1835572881f5f816793ed394605d597c9e82cdb64a1640dc1c6f7f1e3f51921cfee77967f8a73bd18
98e8b1f8c00721d096c8538a64057185d4629c77307f65f0e5fa20555d4fdde335f3bec4585a596d5bf49f0897c6dd5a2216847d5e9b4096095a257db1de7d6644
c56efa387fee80b282d14caa6f9f15def773048da1d32496692c5e0e753977d1258d1c20a376cf5f592358515bb518af378009226d35390b85108a72f2c2b9e1d5
ba72e3dd7015aecd842b5f7e66059b6c44cc1335f9da8bb5302d610756aa36d4ea22f9227786f727ca65caeec39ad798a51b99cc292d1098484a33f697afd30821
d830d73cb7970c2b90204d5a14352a599b739dd79"},"verifiable_credentials":[{"id":"idd43423423","claims":
[{"attribute":"gender","subject_id":"dis:sov:HHHHHEEEEHHHEEE","attributeValue":"male"},
{"attribute":"age","subject_id":"dis:sov:HHHHHEEEEHHHEEE","attributeValue":" 25"}],"id_issuer":"dis:sov:HHHHHEEEEHHHEEE","type":
[],"date_created":"2018-06-04T13:30:14.159Z","metadata":[],"proof":{"type":"RSA-PSS","date_created":"2018-06-
04T13:30:14.160Z","issuer":"-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAw1bWVIEubyz1EegsMuIg\nUb7hjbEly9bVWoYs0yfjWmPCWpjKMY0CZpEuXUAbWLu8w/50YwV8yQ1Hr58quW
L6\n08TAlBDxHlxufTUfWcHZcSTvAlt1kg60fTc5g+MQ9WBWhlQSd88c66bRNi2WDSZz\nngYcaymOtTnVOpgFCO1vPZ69BabT9LA2wwSWmH2/rsb+i6Psb+Gn1ijfQM4+
QJGT\npnhsLHno+/S1AlzgzAFcDDgfTubgnksT6VxEHFLeaj7KDDDSFOe+fDShMV6QCuqC\nVrUWRxZZ43/TG/4JxyWGTVeAs9N/J+lm68I1UO9KgapUsm56DiK6B+pWLf
RpGlGN\nPQIDAQAB\n-----END PUBLIC KEY-----
","domain":"@Person","nonce":"0.09430052651476917","signature_value":"539f2b6abb2ec3b122b8e7120509d3b5fe3da4b1e1e832f859ec57cf19e7
0f6bf5dfd3fc25a4765f750e6bfd1f4e68e17851688ab28f966a6bc6af8eebea02deb85cfa7b8ba10286b203c4f96e03e8779e10500875799b801ce165d0310ca9
59363fe0621c959ba74ffac5452ed7dd493fab5e71ebca6ae19529960ecb71280378bf36f231d1626f5cff869f47d8a88f94e54697a77e8a413d2b6bfcbbdc00b9
82a535f9900821e1198b0670ad07c394bd0991b9c526e33017df019c7b08573ad152b544c4a73c34eda8d27e80efc43a8966ad15bfa4d46ef10590f672b525b699
14d5b5e04b84e0b500913570ebe0ce75bd09d0f304a16f63d795b82ec86212"}}]}}]}
```

Figure 6.17: Final summary screen where drive is decrypted

It is important to note that the implementation of the app is only partial. We have successfully manged to connect to Google Drive, encrypt and store the data there and later fetch and decrypt it. In terms of generating a verifiable credential, we have used RSA signing and verification and have successfully implemented it as well. The important part that is missing is the interaction with the decentralized networks. This and also the service provider has been completely mocked-up by us, in order to demonstrate proof-of-concept. We also have hard-coded a schema that defines the identities. To store the data at the drive, we have just create a big JSON structure that facilitates all of the user data. In terms of DID key management, we have not implemented a Key chain structure, but instead have used hard-coded secrets that are used to generate the keys.

6.3 Backend & Test Network

The prototype backend is developed in parallel with the frontend applications described above, to provide the functionality needed. Implementing the RESTful API was done in Python Flask[1] application with the help of an assisting library called 'flask-restful'[2]. For storing necessary application data a MySQL database is used. The implemented endpoints and functionality follow the documentation in appendix A.

To simulate the actual interaction with an identity network the Hyperleder Indy Project[3], which is an open source project regarding a DLT, purpose-built for decentralized identity, is used. Sovrin is built on top of the Hyperledger Indy codebase and it is possible to create a pool of nodes which use the Sovrin's governance and trust framework. For a proof-of-concept and prototype, a test pool of nodes will be spawned with three entities publishing DIDs - the DIdMS, identity owner and service provider.

[1]`http://flask.pocoo.org`
[2]`https://flask-restful.readthedocs.io/en/latest/`
[3]`https://github.com/hyperledger/indy-node#about-indy-node`

Chapter 7

Discussion & Reflections

This chapter will present some topics of discussion regarding areas which have not been thoroughly investigated during the project. These can be areas, which were mentioned during the project, but due to time and scope constraints could not be included.

Throughout this chapter references to a talk and discussion between the authors of the paper and Henrik Biering, CEO of Peercraft Aps and OpenID DK Organizer, regarding SSI and identity in general will occur[1]. Furthermore, a presentation given by a Dr. Torsten Lodderstedt at the European Identity & Cloud Conference 2018 titled "Is Blockchain the Silver Bullet for Identity"[2] will also be referenced.

7.1 Challenges with Blockchains for SSI

7.1.1 Governance of the Ledger

As a backbone of the proposed DIdMS, we use DLT-enabled decentralized networks that facilitate SSI and its respective principles. Three of such solutions were presented, all storing DIDs on their respectively ledgers. The one which have been researched the most for this project, mainly due to available documentation, is Sovrin. It is also the only one of the three which is permissioned, i.e. not everybody can become a node who can write to the ledger. This means a selected few organizations have control over the ledger and essentially the identifiers on it. This is a reason some people may worry, since it could still create a form of centralization a kin to oligopoly. But having a permissionless blockchain is probably not the right way to go, since everybody can then contribute in validating transactions on ledger. This means that if one person controls a large amount of nodes, he could essentially dictate the current state of the blockchain. Not to mention that trust problems may also be more difficult to solve.

> *"There is always a hinge somewhere.."* - Henrik Biering talking about blockchain identity .

This was also a concern Henrik raised when talking about Sovrin being backed by Evernym, stating *"..thats an additional thing why i am skeptical of Sovrin."*. He tells about how there seems to be hinge behind these blockchains for identity,

[1]Audio file can be found at "`https://159.89.110.18/appendix_files/talk.m4a`" - username: 'reportReader', password: 'iamreadingthereport2018'.

[2]Presentation can be seen at `https://youtu.be/mt6MMUIyM1s`.

and gives an example of IBM now becoming a Sovrin Steward.

7.1.2 Trust & Accountability

Sovrin has established a *Sovrin Trust Framework* which states the business, legal, and technical terms that members of the Sovrin Network have to agree on. It is the only one of the three presented DLT-enabled solutions, which had such a Trust Framework. And exactly the issue of trust and how the network builds it seems essential for the adoption of such a network. One of the basic ideas behind DLT and blockchains is that you do not need to trust a sole entity, because the trust is in the network as a whole. This could be a fair assessment in relation to trusting the actual ledger and the truth about what is on it. I.e. transactions has been validated correctly and there is non-repudiation. But since we are talking about network used for identities, which means issuing of assertions as we have presented, there is need for trust in a sole entity/identity on the network. On the network with thousands of issuers, how can one figure out which issuer they should trust to assert certain claims? Dr. T. Lodderstedt also touches upon this when talking about the challenges for the relying parties(RP) in such identity networks. This challenge is from both the identity owner and RP perspective and he asks the question: *"In such a network how do i know which of the claim issuers is really eligible to attest that i am Torsten ..., in a way that i can use that for really regulated transactions"*. He gives examples for buying a prepaid sim card and opening a bank account. For transactions which depend on jurisdiction and legal frameworks, he does not see a working concept in the current state of the SSI landscape.

Another related topic to trust is *accountability*, which Henrik Biering actually states could be one of the main challenges to establish. He says *"you do not (necessarily) want trust, you want accountability"*. Accountability meaning if entity A hurts entity B, then B can somehow hurt A. By this A is accountable for what he is doing and others can see that entity A is "bad". In the context of accountability, how would a SP hold an pseudo anonymous identity accountable for misuse of its service? Or how does SP hold the issuer accountable if an asserted claim is false?

7.2 Adoption of DID Networks

As presented the DIdMS is a system which through existing identity networks manage identity owner’s data and identities. The focus of the project and report is to utilize the identity networks for the relationship between two specific types of identity owners - service providers and their consumers. This means for the DIdMS to have an existence, at least these two types of identities have to be present on the network. Furthermore, to facilitate assertions of claims an issuer also has to be present. This is the old "chicken and egg" problem - without any consumers why would the service providers join and without any SPs why would consumers join? And if only consumer and SP are present and no issuers, limited use-cases are available.

7.2.1 Incentives for Service Providers

As seen throughout the report, it is proposed and assumed service providers will present endpoints for various reasons. These endpoints are discover-able in the DDO of the SP's DID, and they have to be present for the proposed DIdMS to function. So the main question that raises is why the SP would have these endpoints implemented? Given they are even present with a DID on a network. The case is the same for the issuers, which is required to have certain endpoints for communication, when claims have to be dynamically asserted. There need to be some incentive for the SP to do this. Mainly two reasons have been discussed:

Minimize Cost of GDPR Compliance By enabling the SP to query identities and personal data by reference (ask each time it is needed), the SP do not need to store it. Also, they will rely on the network and maybe a broker such as a DIdMS, to handle the consent flow. This mean they will easily get a valid consent and disputes between data subject and controller can be handled by looking at the ledger.

Better Data Quality Another area which the SP could benefit from, is the quality and correctness of their users' data. By always having the updated and correct partial identity if their consumers, they should be able to provide better service.

By either expanding the DIdMS functionality or by other means, the SPs and issuers could also have a third party assisting their management of DIDs and identity on the networks.

7.2.2 Governments & Identity Blockchains

Previously in this chapter, the discussion regarding the challenges of utilizing a Blockchain for SSI was presented. Here the discussion was about the trust framework and governance of the network and how there might be concerns in how example Sovrin does it. This concern could prevent individuals and service providers joining the network. For mitigating some of these concerns a more regulated and governed ledger could be established. The nature of all introduced ledgers for identity management is that they do not require a central authority. But as seen there is still a need for somebody to maintain and govern the Blockchain.

That somebody could be multiple governments and/or other public organization. In the Sovrin Trust Framework, the requirements for becoming a Steward on the General Availability Network[3] are specified. The first category stated for an entity to be eligible to become a Steward is:

> "A governmental body or agency, or an entity predominantly owned and controlled by the state, in a jurisdiction that is a member of the United Nations, has observer status in the United Nations, or participates in a specialized agency of the United Nations."[58]

[3]The next generation network rolled out by Sovrin, which is proposed should be general for everybody, and transactions on the ledger are permanent.

This means that in the case of Sovrin the Stewards could be a mix of private and public agencies, which in the end would probably be the best solution.

When discussing the possibility of an European or even global identity Blockchain, governed and utilized by states, Henrik Beiring was skeptical. The context was creating an alternative or substitute to national identification systems, e.g. NemID. To this Henrik response: *"that would require development of standards. And it would take very very long time.."*. For EU or even single states to develop such an identity system, it would require extensive standardization.

Official Issuers

Another thing is the presence of official entity and/or governmental agencies on the network, being able to act as issuers. As relying parties can not trust anybody to assert claims, these agencies have to be present - banks, governments, hospitals, universities, etc. Each entity could have their own identity on a network, each implementing the necessary functionality for issuing Verifiable Credentials. Dr. Torsten Lodderstedt touched upon a scenario like this, since there are 5500 banks in Europe alone. The issue becomes how to find the right bank, etc. Another solution could be the banks via a consortium would have an agent acting on behalf of the banks on the network. This means one identity will be representing the banks and being able to assert to claims. The relying parties will only need to know about one and trust one issuer.

Chapter 8

Conclusion

In this project we have presented some of the challenges and remedies of digital identity. With the development and adoption of online service providers, more and more users have scattered identities across the Internet. The emerging social networks have created threats to user's privacy as they act as a huge repository that often exposes too much personal data. Many organizations have tried to find a solution that could improve the privacy of the users and give them more control over their own data.

In this project we put forward a problem formulation that reflects these issues:

> **How can individuals manage their digital identity via a modern identity management solution that utilizes the core principles of Self-Sovereign Identity and ensures that user rights are protected as specified in the GDPR?**

This problem statement has several dimensions of research that were investigated. As a starting point we took the principles of Self-sovereign identity and investigated organizations and projects that try and bring it to life. The popularity of DLT and blockchain has translated to new frameworks and ecosystems that utilize DIDs and DPKI to create a network where identities can be freely exchanged. These solutions empower the user by giving him full control over his identity provision without having him to rely on a single identity provider. As part of these networks, the users would need an agent that would allow them seamlessly to utilize the benefits of SSI. We coined the term *Decentralized Identity Management System - DIdMS* to differentiate this agent from the traditional Identity Management Systems. We then took the idea of DIdMS and investigated its use cases and functionality in the context of service-user interaction. Within that context we looked into how claims and assertions can be managed and allow the creation of partial identities, how service providers can support defined rights by regulations and how can the DIdMS contain, store and protect user's identities. On top of this the management of consents and protecting of the data subjects rights according to GDPR was included as functionality of the DIdMS. By investigating these research questions we aimed at answering the proposed problem formulation.

The first supplementary question we defined was *"What architecture/ecosystem can enable the independent existence and persistence of a digital identity?"*. The answer to this question were solutions like Sovrin, Veres One and uPort - all of which utilize distributed ledgers and propose ecosystems built around identity purposed blockchains. These solutions and their networks all include the use of Decentralzied Identifiers (DIDs). With this specification it is possible to create persistent identifiers which are globally recognizable and discoverable. Their creation and maintenance does not require a single central authority, but instead can be shared among any number. Using DLT with DID as a backbone for identity management means one can have an independent and long-lived existence.

Having solutions and ecosystems enabling the Self-sovereign identity and allowing independent existence of the digital identities. However, if the identity owner has no control over their identity and how it is managed, none of these aspects would matter. This is why the second question which is asked is *"How can we enable the user to have control over the management of their identities and minimize the disclosure of claims?"* The DID specification on the DLTs forms the basis necessary for facilitating Decentralized PKI. DPKI enables the user to have full control of his DIDs by being in possession of a related private keys. Another aspect which allowed the identity owner to move further away from the conventional centralized model was migrating all of their identity data to a vault which is managed explicitly by them. Furthermore there is a necessity to allow the user to be able to create as many partial identities as they want, mimicking the real life examples. A tree-like architecture is proposed is able to accommodate all of the potential partial identities the user would like to make by allowing them a more fine-grained management of attributes. By facilitating this fine-grained management the identity owner can explicitly define the different partial identities he would like to use to represent themselves in their digital interactions. This will allow them to have better management and control over what types of data is disclosed.

Establishing how should the structure of the identity and presenting how the owner of the information can handle the creation of different partial identities, the next step was defined by the third research question - *"How can the user provide asserted claims in a well-defined way to service providers?"*. It is essential to define how will the identity transactions will be done on the distributed ledger networks. The use of these networks and DPKI means that any identity on the ledger can become an *issuer*. This means that when requiring a set of information from an identity owner, the requesting *SP* needs to provide a list of issuers it trusts. A common ground for providing such claims with the necessary level of trust required by the entities involved in a communication process can be established through the use of W3C's working draft for Verifiable Credentials. The draft specifies a way in which the different claims with their respective assertions can be presented in a structured way. Furthermore, having the data migrated into a user-managed vault brought about the need of defining a way it will be made available to the service providers the specific user is interacting with. This meant that there is a need to establish a way for the Service Provider to be able to retrieve the identity data which is provided by the identity owner. This is where the Identity Hub working draft is used. Taking the requirements elicited in the draft we create the Identity Hub on a conceptual level in the context of the DIdMS. Taking the requirement of high availability it was evident that a user-managed vault would not be the best option for providing these endpoints. Therefore it is decided to allow the DIdMS to take on these responsibilities and provide an encrypted bundle of identity information, explicitly overseen by its owner, at an endpoint. This endpoint is then disclosed through a pair-wise DID's DDO to the SP it corresponds to. Thus we have defined how an identity owner can provide their information to a service provider in a manner which they still retain control over the information they are disclosing.

However, stating that the DIdMS will provide some of the user data which means that it should follow any of the guidelines and implications associated with the General Data Protection Regulation. Hence, we have formulated our fourth research question - *"What are the privileges granted to the individual by GDPR and how can these be facilitated by the identity management system?"*. In regards to GDPR the data subject/user has rights when it comes to service providers using their data. We found that the DIdMS should provide consent management functionality, including conveying, accepting and revoking consents. For this the Kantara Initiave specification Consent Receipt(CR), describing what should be included in a consent, is selected. This CR can not be created by the DIdMS, as it knows nothing about the SP, so the SP must provide a valid CR. For this, another endpoint is proposed to be included in the DDO of the SP, which will specify such a consent receipt. The DIdMS will obtain, resolve and present this CR to the user. If the user decides to accept it and give consent to the SP, the DIdMS will assist in the user signing it. Then this signed CR will be provided to the SP. As a result both parties will have a copy. While these choices enabled the interaction between a service provider and a user, the DIdMS itself had to also be considered in regards to GDPR since it could be viewed as a service from the perspective of the user. From the previous question it is stated that the user's identity data is stored in an identity directory. This choice meant that the DIdMS would not store any personal data. However, to enable the provisioning of partial identities, which correspond to tier 3 nodes in the tree-like structure, we chose to include the DIdMS as a part of the concept of *Identity Hub*. This allowed us to define endpoints that contained the data that the service provider needs. Since this data could be personal, a design choice was made of storing it encrypted at the endpoint. By utilizing the specification of decentralized identifiers we created a simple flow that stored the data at the endpoint encrypted and only allowed for the service provider, which has been given consent to use it, to decrypt it.

To avoid dealing and managing personal data, we divided the DIdMS into *frontend* and *backend* applications. The frontend is called a *Client* and is the only place within the system where personal data is processed. This means that any processing of personal data will happen exclusively on the user's side. The backend, on the other hand, facilitated the endpoints and Identity Hub concept. By doing this separation, we protect the privacy of the user, while maintaining the use cases specified from he scenarios. The design of the DIdMS was created in order to facilitate the requirements from the scenarios. We did not develop a complete design specification.

In regards to our research questions we found that the decentralized networks are a good enabler of the self-sovereign identity. We identified key GDPR rights that have to be supported and also found a reliable way of structuring and transmitting identity data.

In regards to the general problem statement we identified a way for the user to create and manage his identity in accordance with the SSI principles. However, the proposed solution is a proof-of-concept. It contains only partial functionality and in practice a more extensive requirements and specification have to be developed. In addition, all of the discussion and analysis is based on the assumption that decentralized networks are a work-in-progress, and they have not yet established themselves. In order to make the claim that we have successfully answered the defined problem statement, we need to conduct practical testing with real users and services. For the purposes and scope of this paper, we conclude that we have only provided a theoretical answer that can be used as the foundation for building a decentralized identity management system which facilitates the communication with multiple distributed ledger networks in order to enable a truly SSI identity.

Bibliography

[1] David Pellauer and Bernard Dauenhauer. “Paul Ricoeur”. In: *The Stanford Encyclopedia of Philosopy*. Ed. by Edward N. Zalta. Winter 2016. Metaphysics Research Lab, Stanford University, 2016.

[2] David W. Chadwick. “Identity Management”. In: *Foundations of Security Analysis and Design V*. Ed. by Roberto Gorrieri Alessandro Aldini Gilles Barthe. Cambridge University Press, 1984, pp. 96–120. doi: `10.1017/CBO9780511625138.007`.

[3] 3G Americas. *Identity Management - Overview of Standards & Technologies for Mobile and Fixed Internet*. `https://observatorio.iti.upv.es/media/managed_files/2009/01/12/3GAmericas_Unified_Identity_Management_Jan2009.pdf`. Online; Accessed 20-05-2018. 2009.

[4] Morris Sloman Tyrone Grandison. *A Survey of Trust in Internet Applications*. `https://spiral.imperial.ac.uk/bitstream/10044/1/13945/2/Trust_Survey.pdf`. Online; Accessed 21-05-2018. 2001.

[5] Katrin Borcea-Ptzmann et al. “Scenario, Analysis, and Design of Privacy Throughout Life Demonstrator”. In: *Privacy and Identity Management in Europe for Life*. Ed. by Katrin Borcea-Pfitzmann. 1.0. Feb. 2011.

[6] Kim Cameron. *The Laws of Identity*. Online; Accessed 10-02-2018. 2005. url: `http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf`.

[7] Sebastian Ryszard Kruk et al. “D-FOAF: Distributed Identity Management with Access Right Delegations”. In: *The Semantic Web*. Ed. by F. Giunchiglia R. Migozuchi Z. Shi. Springer, 2006.

[8] Glenn Fleishman. *Cartoon Captures Spirit of the Internet*. `https://web.archive.org/web/20171229172420/http://www.nytimes.com/2000/12/14/technology/cartoon-captures-spirit-of-the-internet.html`. Online; Accessed 14-05-2018. Dec. 2000.

[9] Erika McCallister, Timothy Grance, and Karen A. Scarfone. *SP 800-122. Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)*. Tech. rep. Gaithersburg, MD, United States, 2010.

[10] European Commission. “REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)”. In: *Official Journal of the European Union* 2016/L119 (May 2016).

[11] Ping Identity. *Internet-Scale Identity Systems: An Overview and Comparison*. http://www.eb2bcom.com/information/whitepapers/pingidentity/internet_scale_identity_systems.pdf. Online; Accessed 21-05-2018. 2010.

[12] Katrin Borcea-Pfitzmann Manuela Berg. "Implementability of the Identity Management Part in Pfitzmann/Hansen's Terminology for a Complex Digital World". In: *Privacy and Identity Management for Life*. Ed. by Simone Fischer-Hubner et al. 2010, pp. 15–26.

[13] Lee Raine. *The state of privacy in post-Snowden America*. Online; Accessed 24-4-2018. Sept. 2016. url: http://www.pewresearch.org/fact-tank/2016/09/21/the-state-of-privacy-in-america/.

[14] Sabouri A. Rannenberg K. Tesfay W. "Introduction". In: *Attribute-based Credentials for Trust*. Ed. by Sabouri A. Rannenberg K. Camenisch J. Springer, 2015.

[15] Valerie Kimball Richard Brody Elizabeth Mulig. "Phishing, pharming and identity theft". In: *Academy of Accounting and Financial Studies Journal* 11 (2007), pp. 43–56.

[16] Scott Thurm and Yukari Iwanti Kane. *Your Apps are Watching You*. https://www.wsj.com/articles/S3703574602. Online; Accessed 20-05-2018. 2010.

[17] Richard Nieva. *Facebook says Cambridge Analytica had data on 87 million people*. https://www.cnet.com/news/facebook-says-cambridge-analytica-had-data-on-87m-people/. Online; Accessed 21-05-2018. 2018.

[18] 2016 TRUSTe/NCSA Consumer Privacy Infographic – US Edition. *National Cyber Security Alliance*. Online; Accessed 24-4-2018. 2016. url: https://www.trustarc.com/resources/privacy-research/ncsa-consumer-privacy-index-us/.

[19] Statista. *Online privacy*. Apr. 2017. url: https://www.statista.com/topics/2476/online-privacy/.

[20] Internet Identity Workshop. *IIW Homepage*. https://www.internetidentityworkshop.com/. Online; Accessed 03-06-2018. 2017.

[21] Christopher Allen. "The Path To Self-sovereign Identity". In: *Life With Alacrity* (Apr. 2016). url: http://www.lifewithalacrity.com.

[22] Christopher Allen and Shannon Appelcline. "A Primer on Self-Sovereign Identity". In: *RWOT5*. Rebooting the Web of Trust. Sept. 2017. url: https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-fall2017/blob/master/topics-and-advance-readings/self-sovereign-identity-primer.md.

[23] Ian Sommerville. *Software Engineering*. Ed. by Marcia Horton. 9th ed. Addison-Wesley Publishing Company, 2010.

[24] UML.org. *Introduction To OMG'S Unified Modeling Language*. http://www.uml.org/what-is-uml.htm. Online; Accessed 24-05-2018. July 2005.

[25] Radoplhe Marques. *Rebooting the Web Of Trust Workshop: A Recap*. `https://blog.bigchaindb.com/rebooting-the-web-of-trust-workshop-a-recap-c3ecbd755790`. Online; Accessed 02-06-2018. 2017.

[26] Web Of Trust. *Specifications and Reports*. `https://www.weboftrust.info/specs.html`. Online; Accessed 02-06-2018. 2017.

[27] Kantara Initiative. *About*. `https://kantarainitiative.org/about/`. Online; Accessed 02-06-2018. 2018.

[28] Dewni Weeraman. *User Managed Access—UMA 2.0*. `https://medium.com/@dewni.matheesha/user-managed-access-uma-2-0-bcecb1d535b3`. Online; Accessed 02-06-2018. 2017.

[29] Decentralized Identity Foundation. *Decentralized Identity Foundation Grows To 56 Members In Our First Year*. `https://medium.com/decentralized-identity/decentralized-identity-foundation-grows-to-56-members-in-our-first-year-3ec117e811d8`. Online; Accessed 02-06-2018. 2018.

[30] W3C. *World Wide Web Consortium*. `https://www.w3.org/`. Online; Accessed 02-06-2018. 2018.

[31] Nick Steele. *Web Authentication: What It Is and What It Means for Passwords*. `https://duo.com/blog/web-authentication-what-it-is-and-what-it-means-for-passwords`. Online; Accessed 02-06-2018. 2017.

[32] Drummond Reed et al. *Decentralized Identifiers (DIDs) v0.9*. Draft Community Group Report. W3C, Apr. 2018.

[33] Markus Sabadello. "A Universal Resolver for self-sovereign identifiers". In: *Life With Alacrity* (Nov. 2017). url: `https://medium.com/decentralized-identity/a-universal-resolver-for-self-sovereign-identifiers-48e6b4a5cc3c`.

[34] Daniel Buchner. *DIF Identity Hubs*. `https://github.com/decentralized-identity/hubs/blob/master/explainer.md`. Online; Accessed 24-05-2018.

[35] E. Hammer-Lahav M. Nottingham. *Defining Well-Known Uniform Resource Identifiers (URIs)*. RFC 5785. RFC Editor, Apr. 2010. url: `https://tools.ietf.org/html/rfc5785`.

[36] Sir Tim Berners-Lee. *Linked Data*. World Wide Web Consortium, July 2006. url: `https://www.w3.org/DesignIssues/LinkedData.html`.

[37] Manu Sporny et al. *JSON-LD: A JSON-based Serialization for Linked Data*. Draft Community Group Report. Version 1.1. W3C, May 2018.

[38] Schema.org. *About Schema.org*. `http://schema.org/docs/about.html`. Online; Accessed 1-06-2018.

[39] Charles P. Pfleeger, Shari Lawrence Pfleeger, and Jonathan Margulies. *Security in Computing (5th Edition)*. 5th. Upper Saddle River, NJ, USA: Prentice Hall Press, 2015. isbn: 0134085043.

[40] John R. Vacca. *Public Key Infrastructure: Building Trusted Applications and Web Services*. 1st. Boston, MA, USA: Auerbach Publications, 2004. isbn: 0849308224.

[41] Mike Ricks, Sergey Simakov, and Shawn Rabourn. *Securing Public Key Infrastructure (PKI)*. Tech. rep. Microsoft, May 2014.

[42] *Decentralized Public Key Infrastructure*. `https://danubetech.com/download/dpki.pdf`. Online; Accessed 29-05-2018. 2018.

[43] Manu Sporny and Dave Longley. *Verifiable Claims Data Model and Representations*. First Public Working Draft. W3C, Aug. 2017.

[44] Joe Andrieu, Sunny Lee, and Nate Otto. *Verifiable Claims Use Cases*. Working Group Note. W3C, June 2017.

[45] Dirk Balfanz et al. *Web Authentication: An API for accessing Public Key Credentials Level 1*. W3C Candidate Recommendation. W3C, Mar. 2018.

[46] Consent & Information Sharing Work Group. *Consent Receipt Specification 1.1.0 DRAFT 8*. Technical Specification Recommendation. Kantara Initiative, Feb. 2018.

[47] *ISO/IEC 29100:2011 Information technology – Security techniques – Privacy framework*. Standard. International Organization for Standardization, Dec. 2011.

[48] Shaan Ray. *The Difference Between Blockchains & Distributed Ledger Technology*. `https://towardsdatascience.com/the-difference-between-blockchains-distributed-ledger-technology-42715a0fa92`. Online; Accessed 25-05-2018. Feb. 2018.

[49] Mark Walport. *Distributed Ledger Technology: beyond block chain*. Tech. rep. UK Government, Office for Science, 2015. url: `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf`.

[50] Dylan Yaga et al. *Draft NISTIR 8202. Blockchain Technology Overview*. Tech. rep. Gaithersburg, MD, United States, 2018.

[51] Natasha Lomas. *Digi.me bags $6.1M to put users in the driving seat for sharing personal data*. Online; Accessed 25-03-2018. 2016. url: `https://techcrunch.com/2016/06/30/digi-me-bags-6-1m-to-put-users-in-the-driving-seat-for-sharing-personal-data/`.

[52] digi.me Ltd. *Data sources – check out our massive and growing catalogue!* Online; Accessed 25-03-2018. 2018. url: `https://digi.me/sources`.

[53] digi.me Ltd. *Get started with Digi.me*. Online; Accessed 25-03-2018. 2018. url: `https://digi.me/get-started`.

[54] Kantara Initiative. *iWelcome and digi.me Launch Kantara Initiative Consent Management Solutions Work Group*. 2017. url: `https://kantarainitiative.org/iwelcome-and-digi-me-launch-kantara-initiative-consent-management-solutions-work-group/`.

[55] Chekk.me. *Businesses*. Online; Accessed 10-03-2018. url: `http://chekk.me/`.

[56] Jason Law Drummond Reed and Daniel Hardman. *The Technical Foundations of Sovrin*. Sept. 2016. url: `https://www.evernym.com/wp-content/uploads/2017/07/The-Technical-Foundations-of-Sovrin.pdf`.

[57] Phillip J. WindleyChair. *How Sovrin Works*. Sept. 2016. url: https://sovrin.org/wp-content/uploads/2017/04/How-Sovrin-Works.pdf.

[58] Sovrin Board of Trustees. *Sovrin Provisional Trust Framework*. June 2017. url: https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Provisional-Trust-Framework-2017-06-28.pdf.

[59] Evernym Inc. *The Solution*. https://www.evernym.com/solution/. Online; Accessed 25-05-2018.

[60] Evernym Inc. *Connect.Me*. http://connect.me/. Online; Accessed 14-03-2018.

[61] Veres One. *The Veres One Project*. url: https://veres.one/.

[62] Manu Sporny. "Deep Dive on the Veres One Decentralized Identifier Blockchain". In: *Internet Identity Workshop 26*.

[63] Michael Sena. *Privacy Preserving Identity System for Ethereum dApps*. https://medium.com/uport/privacy-preserving-identity-system-for-ethereum-dapps-a3352d1a93e8. Online; Accessed 1-06-2018. 2018.

[64] Aldi Gjoka. *uPort on any Ethereum blockchain*. https://medium.com/uport/uport-on-any-ethereum-blockchain-c54368a12e8c. Online; Accesssed 1-06-2018. 2018.

[65] Pelle Braendgaard and Michael Sena. *Multi Network Identifier-spec and reference implementation*. https://github.com/uport-project/mnid/blob/master/README.md. Online; Accessed 1-06-2018.

[66] M. Johnstone, ed. *A Survey Of Social Media Users Privacy Settings and Information Disclosure*. (Leuven, Belgium). Research Online, Dec. 2016.

[67] Harriet Green. *The Internet of Things in the Cognitive Era: Realizing the future and full potential of connected devices*. Tech. rep. Online; Accessed 22-04-2018. IBM, Dec. 2015.

[68] Samuel Gibbs. *Nest Learning Thermostat third-gen: the simple, effective heating gadget*. Online; Accessed 22-04-2018. Mar. 2016. url: https://www.theguardian.com/technology/2016/jun/03/google-nest-learning-thermostat-third-generation-home-gadget-smart-heating.

[69] James Peckham. *Best Fitbit 2018: which is right for you?* https://www.techradar.com/news/wearables/best-fitbit-which-is-right-for-you-1322700. Online; Accessed 1-06-2018. 2018.

[70] Apple Inc. *Apple Watch*. https://www.apple.com/watch/. Online; Accessed 1-06-2018. 2018.

[71] Markus Sabadello and Danube Tech. *DID Auth: Scope, Formats, and Protocols*. Online; Accessed 28-05-2018. Mar. 2018. url: https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-spring2018/blob/master/topics-and-advance-readings/DID%20Auth:%20Scope%2C%20Formats%2C%20and%20Protocols.md#did-auth-scope-formats-and-protocols.

[72] Massimo Villari Antonio Celesti Francesco Tusa and Antonio Puliafito. *An XDI-Based Approach to Represent and Exchange Data Between Federated Clouds*. June 2012. url: https://pdfs.semanticscholar.org/a50b/ff33fd3ddab833d9deeea83cf60f28e7d4bb.pdf.

[73] Scott Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. BCP 14. `http://www.rfc-editor.org/rfc/rfc2119.txt`. RFC Editor, Mar. 1997. url: `http://www.rfc-editor.org/rfc/rfc2119.txt`.

[74] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. *Argon2: the memory-hard function for password hashing and other applications*. Tech. rep. University of Luxembourg, 2017.

[75] M. Jones et al. *JSON Web Token (JWT)*. RFC 7519. RFC Editor, May 2015. url: `https://tools.ietf.org/html/rfc7519`.

Appendix A

API Documentation

Base url: https://159.89.110.18/didmapi

A.1 Internal API

All communication must

A.1.1 General Response

Any non 200 HTTP response will contain the following in the body:

```
{
    "error_code": number,
    "error_message": string
}
```

The error code being a number for machine reading and the error message a possible string which can be shown to the user.

General error codes

Error code	Description
0	Token invalid.
1	Request body did not include required data.
2	Request arguments must be present.
3	Access token must be provided in the Authorization header.

A.1.2 Authentication

All endpoints below needs an access token, included in the 'Authorization' header. For all endpoints a 401 is returned if the access token is not provided or not valid. This access token is provided at following endpoint, giving a user is registered.

GET /accessToken

Request arguments

As the salting needs to happen server side (same salt needs to be used),

Param name	**Value**
username	The username of the user wanting to log in.
password	Associated password.

Response
In a successful request the response status code will be 200 with a body containing the access token which should be used for following interactions with the API for that session.

```
{
    "access_token": string
}
```

For all other responses following error codes can happen:

Error code	**Description**
11	Wrong username or password.

A.1.3 Endpoints

POST /user

This endpoint is for registering a user to the service. This endpoint should be called from the client when a new user wants to be created within the system. For obvious reasons this endpoint do not require an access token.

Request body

```
{
    "username": string,
    "password": string
}
```

Response
In a successful request the response status code will be 200 with a body containing the access token which should be used for following interactions with the API for that session.

```
{
    "access_token": string
}
```

Upon an unsuccessful request a 400 will be returned with following possible error codes:

Error code	**Description**
10	Username already exists within the system.
30	Non valid value for identity directory type.

POST /me

Endpoint for updating users information. Currently supported update values are available are shown below. *Request body*

```
{
    "identity_directory_id": int
}
```

GET /me

Does not take any arguments.Returns the users information in the system.

Response body

```
{
    "identity_directory_id": int
}
```

GET /spspec

Endpoint for getting a service providers specification of what credentials are needed as well as consents reciepts needed to be signed.

Request

Param name	Value
didsp	The DID of the service provider.

Response

```
{
    "service_provider_spec": {
        "sp_did_public_key": string,
        "consent_reciepts": [
            {},
            ...
        ],
        "credential_spec": [
            {
                "credential_name": string,
                "description": string,
                "trusted_issuers": [] or none
            },
            ....
        ]
    }
}
```

POST /identity

This endpoint is for a user wanting to publish a new identity which should be made in to an endpoint for service providers.

Request body

```
{
    "new_identity_jwt": base64jwt
}
```

The jwt is expected to include the following payload:

```
{
    "user_did": string,
    "did_public_key": string,
    "did_service": string,
    "sp_data": {
        "aes_key": string,
        "bundle": string
    }
}
```

And must be signed using RSASHA256 with the key associated with the DID, hence 'did_public_key'.

Response

In a successful request the response status code will be 200 with a body containing the access token which should be used for following interactions with the API for that session.

```
{
    "access_token": string
}
```

Upon an unsuccessful request a 400 will be returned with following possible error codes:

Error code	**Description**
40	DID already exists, and is therefor not valid.

Appendix B

Mobile App Documentation

B.1 General Goals

The creation of the mobile application we shall allow the user to register in our system. They will have to create an account, select the different sources of information that will be aggregated in their complete digital identity which is then saved to a user-selected Identity Directory.

B.2 Platform

Android

B.3 UI/UX

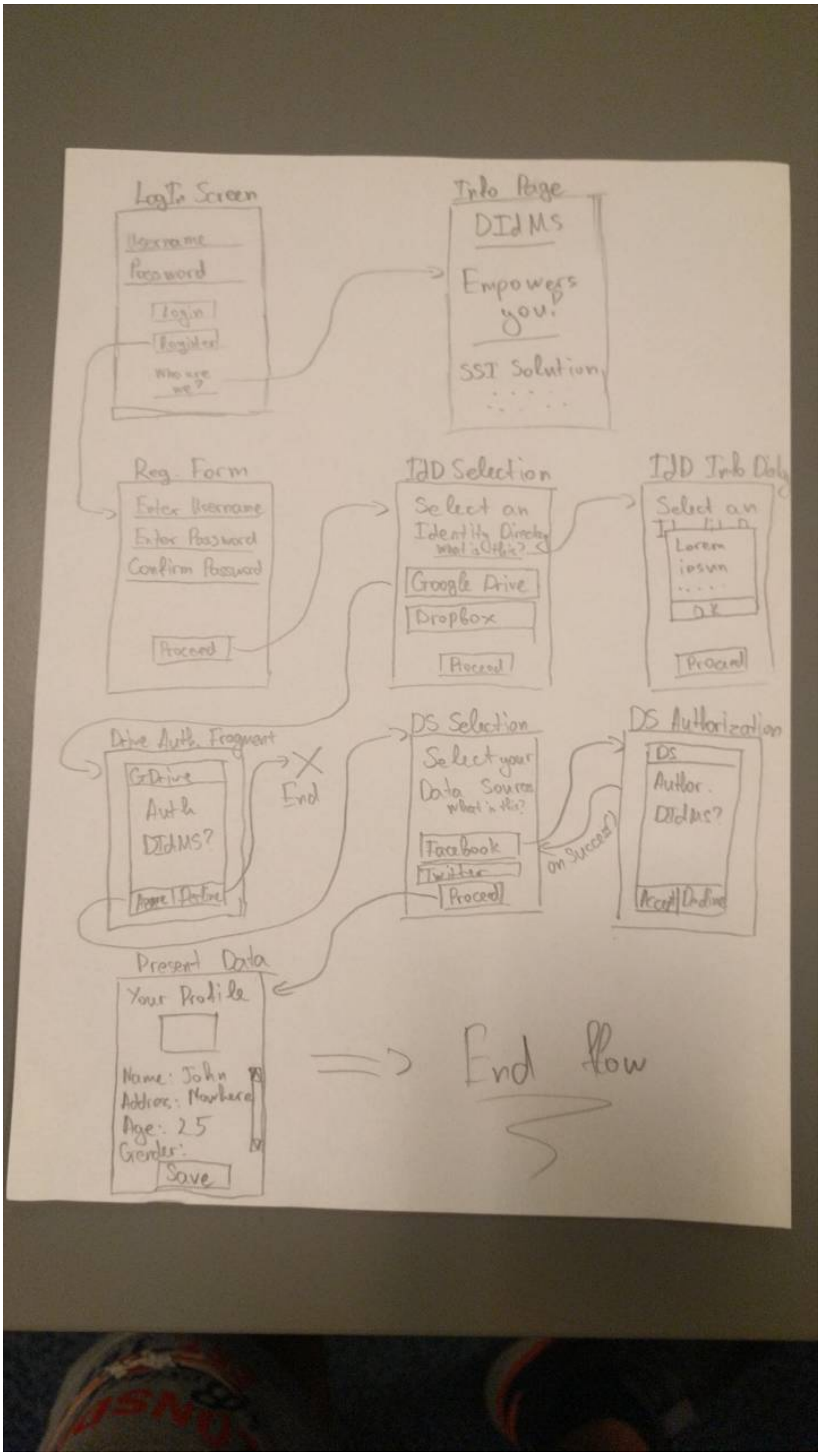
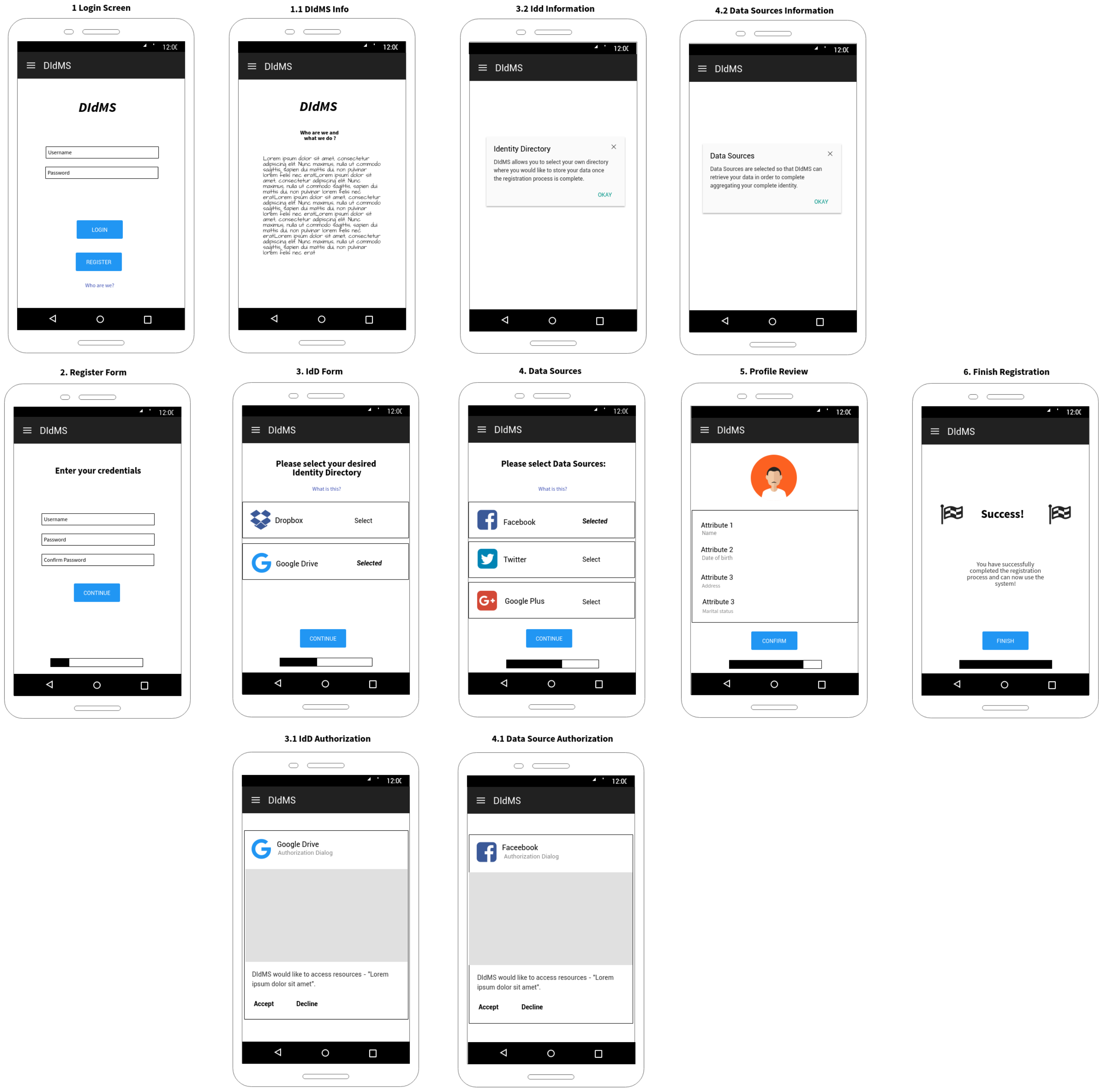


Figure B.1: Hand-Drawn Mobile Application Design



DIdMS allows you to select your own directory where you would like to store your data once the registration process is complete.

Data Sources are selected so that DIdMS can retrieve your data in order to complete aggregating your complete identity.

You have successfully completed the registration process and can now use the system!

DIdMS would like to access resources - "Lorem ipsum dolor sit amet".

DIdMS would like to access resources - "Lorem ipsum dolor sit amet".

Figure B.2: Complete Mocks

Appendix C

Consent Receipts

C.1 Kantara Initiative Consent Receipt Example

Listing C.1: Example of Consent Receipt; Generated in: `http://api.consentreceipt.org/`

```json

{
  "jurisdiction": "US",
  "moc": "Web Form",
  "sub": "example@example.com",
  "notice": "http://example.com/shortnotice",
  "policy_uri": "http://example-service.com/privacy",
  "data_controller": {
    "contact": "Dave Controller",
    "company": "Data Controller Inc.",
    "address": "123 St., Place",
    "email": "dave@datacontroller.com",
    "phone": "00-123-341-2351",
    "on_behalf": false
  },
  "purpose": [
    [
      "Service1",
      "personalized_experience",
      "biographical",
      "social_contact"
    ]
  ],
  "sensitive": [
    "official_identifiers"
  ],
  "sharing": {
    "sharing": [
      "biographical"
    ],
    "party_name": "3rd Party Name or/3rd Party Category",
    "purpose": "personalized_experience"
  },
  "scopes": "update data",
  "jti": "e8e2563a4303ebc1cdeb3481592ace2c6d3ab751109a440075a095f1486f871177b101bee4a6bcf8c20a53896ef037fea99414bf91133f4744c9f122ae6764e7",
  "iat": "1528278571",
  "iss": "http://www.consentreceipt.org/"
}
```