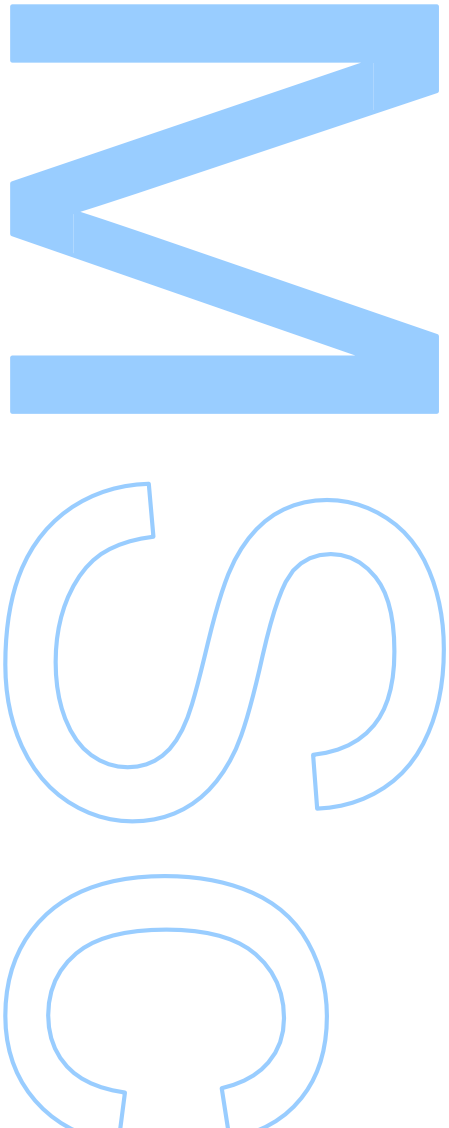
SSI Technology in the context of eIDAS 2.0

João Manuel Alexandrino de Castro Ribeiro
M:SI- Mestrado em Segurança Informática
Departamento de Ciência de
Computadores da Faculdade de
Ciências da Universidade do Porto
2023

Orientador
Prof. Dr. Manuel Eduardo C. D. Correia, Professor Associado do Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto

Coorientador
Prof. Dr. Rolando da Silva Martins, Professor Auxiliar do Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto



Acknowledgements

I would like to acknowledge and thank my supervisor Prof. Dr. Manuel Eduardo Carvalho Duarte Correia and my co-supervisor Prof. Dr. Rolando da Silva Martins, for the guidance they gave me and for helping out even with such a late start. I would also like to thank my parents for supporting me during the writing of this article, as well as all the time before that. And finally I would like to thank my friends for being around and helping make this time more enjoyable and memorable.

UNIVERSIDADE DO PORTO

Abstract

Departamento de Ciência dos Computadores
Faculdade de Ciências da Universidade do Porto

MSc. Information Security

SSI Technology in the context of eIDAS 2.0

by João RIBEIRO

Self Sovereign Identity (SSI) is an identity paradigm where a user is in charge of their own identity, as opposed to the system currently in place where every service has its own version of a user's identity. It is a very relevant topic nowadays, both with the problems presented by the current silo-based digital identity architecture that is currently the base of Internet identity, and with the burdensome nature of physical documents and the processes associated with them. eIDAS is an European regulation aimed at implementing a unique, user-owned European digital identification framework which aims at having a seamless digital wallet in place by 2030.

This dissertation covers SSI technologies and the systems underlying them, such as Distributed Ledger Technology and Decentralized Public Key Infrastructures, as well as the European electronic Identification, Authentication and Trust Services regulation and the frameworks and technologies adjacent to it, such as the Architecture Reference Framework and the European Digital Identity Wallet.

The goal of the dissertation is an extensive and comprehensive aggregation and analysis of the technologies and regulations listed previously, as well as the assembly of a simulated SSI ecosystem that functions under the rules studied in the theoretical sections, while also doing an in depth study of the technologies applied in this ecosystem, those being the and Wallet Kits developed by the WaltId group and the EBSI network.

The technologies adjacent to SSI were extensively explored as well as the eIDAS regulation and its dependant technologies which were also comprehensively studied and listed, and although some practical elements of the SSI system are lacking, a solid foundation for a more robust system was set in place, while also covering in depth the theory behind it.

UNIVERSIDADE DO PORTO

Sumário

Departamento de Ciência dos Computadores
Faculdade de Ciências da Universidade do Porto

Mestrado em Segurança Informática

SSI Technology in the context of eIDAS 2.0

by João RIBEIRO

Self Sovereign Identity, ou Identidade Auto-Soberana, é um paradigma de identidade que estipula que cada utilizador tem poder sobre a sua identidade, em contraste com o sistema atualmente em uso em que cada serviço possui a sua versão da identidade do utilizador. Isto é um tópico muito relevante hoje em dia, por causa de ambos os problemas causados pela arquitetura atual da identidade digital, na qual assenta uma grande parte das identidades da Internet, e da natureza dos documentos de identidade físicos e dos processos associados a estes. A regulação eIDAS aponta para implementar uma estrutura de identidade digital Europeia única e controlada pelo utilizador, e que aponta para ter uma carteira digital de uso acessível completa até 2030.

Esta dissertação cobre os temas de tecnologias SSI e os sistemas que lhes servem de base, por exemplo DLTs (tecnologia de ledgers distribuídos) e infraestruturas de chaves públicas descentralizadas, e cobre também a regulação eIDAS (Identificação, Autenticação e Serviços de Confiança da Identidade Eletrónica Europeia), tal como as estruturas que lhe são adjacentes, como é exemplo a ARF (Estrutura de Referência e Arquitetura) e a Carteira de Identidade Digital Europeia.

Os objetivos principais desta dissertação são fazer um levantamento extensivo e análise das tecnologias e regulações listadas anteriormente, e construir um ecossistema simulado de SSI que funcione de acordo com as tecnologias e regulamentos estudados na porção teórica do documento. De igual importância é fazer um estudo aprofundado das tecnologias concretas usadas na porção prática, o kit waltid e a rede EBSI.

As tecnologias adjacentes a SSI foram extensivamente exploradas tal como a regulação eIDAS e as tecnologias de que esta depende, que também foram completamente estudadas e listadas, e embora alguns elementos do sistema SSI estejam em falta, foi construída

um fundação sólida para um sistema mais robusto, enquanto a teoria por traz disto foi explorada em profundidade.

Contents

List of Figures

Acronyms

AdES Advanced Electronic Signature. 42

ARF Architecture and Reference Framework. xi, 7, 32, 34–38, 67

DID Decentralized Identifier. 21–23, 25, 39–45, 47, 48, 50–55, 58–61, 63, 65

DLT Distributed Ledger Technology. 25, 67

DPKI Decentralized Public Key Infrastructure. 26–28, 43, 67

DSS Digital Signature Services. 43

EAA Electronic Attestation of Attributes. 33

EBSI European Blockchain Services Infrastructure. iii, 7, 8, 29, 40–46, 48, 50, 52, 57, 58, 60, 61, 67

ECDSA Elyptic Curve Digital Signature Algorithm. 60

EDDSA Edwards-curve Digital Signature Algorithm. 60

eIDAS electronic Identification, Authentication and Trust Services. 6–8, 30–32, 36, 42, 43, 50, 51, 67, 68

ESM European Single Market. 42

EUDI European Digital Identity. xi, 10, 26–28, 31–35, 37, 49, 50, 67

FIM Federated Identity Management. 12

IdM Identity Management. 11

IdP Identity Provider. 11, 12, 15–17, 39, 41

JAdES JSON Advanced Electronic Signature. 42, 43

JSON JavaScript Object Notation. 16, 42, 50, 52, 53, 60, 61

JSON-LD JSON for Linking Data. 42, 60

JWK JSON Web Key. 42, 44

JWS JSON Web Signature. 42, 43, 53

JWT JSON Web Token. 17, 42, 44, 60

OAuth Open Authorization. xi, 13–18, 46, 47

OID4VC OpenID for Verifiable Credentials. 37, 46

OIDC OpenID Connect. xi, 16–18, 39, 41

OpenID4VP OpenID for Verifiable Presentations. 61

PID Personal Identification Information. 33–37

PKI Public Key Infrastructure. 26, 27, 42, 57

QEAA Qualified Electronic Attestation of Attributes. 33, 34, 36, 37

RP Relying Party. 16–18, 24

RSA Rivest-Shamir-Adleman. 60

SAML Security Assertion Markup Language. xi, 15–17

SP Service Provider. 11, 12

SSI Self-Sovereign Identity. xi, 6–8, 10, 18–22, 24, 29, 31, 32, 39, 43, 44, 49, 55, 58–60, 67, 68

SSO Single Sign-on. 12, 15, 16, 67

TAI Trusted Accreditation Issuer. 45

TAO Trusted Accreditation Organization. 45, 52, 57, 58

TI Trusted Issuer. 44, 52

TIR Trusted Issuer Registry. 52

TSP Trust Service Provider. 33, 34

TSR Trusted Schema Registry. 42

VC Verifiable Credential. viii, 20, 39–44, 47, 52, 53, 59–61, 63

VCI Verifiable Credential Issuance. 46

VP Verifiable Presentation. viii, 20, 41–43, 47, 48, 53, 54, 56, 59–61, 65

WCT Wallet Conformance Testing. 48

XML Extensible Markup Language. 15

Chapter 1

Introduction

The current landscape of Internet identity is not only unwieldy, but it also takes power from users while revealing too much personal information to other parties that have no business knowing it. On the other side of the identity landscape, we have physical identity documents, such as identification cards or a driver's license, which are practical to carry around, but can become burdensome when they have to be presented physically. The physical presentation nature of these processes often leads to delays due to the need of an appointment, for example for the creation of a bank account.

Everyone is familiar with how burdensome physical identity processes can be, but only some of the downsides of the current landscape of digital identity are intuitive to the end user. They include the onerous task of having to memorise a username and password for each different service that is used, at least if proper security guidelines are being followed and the same password is not repeated for multiple services. This problem has at its core the fact that a user does not currently own their identity: their identity is bound to each individual service and in no way is it under the user's control.

This not only puts the onus of remembering all the multiple credentials on the user, but also removes power from them as they cannot manage their own digital identity. Also, services often request more information than is necessary for their functionalities, allowing personal data to be collected and profiling to be done on the user. This can then be linked to other profiles of the same user in other services, effectively being a breach of privacy with consequences ranging from targeted ads to bad faith practices by services that limit user options based on their profiles.

The most explored means of solving these problems is decentralized identity, an approach to identity management where the user is in control of their identity, as opposed

to the current implementations where the service owns the identity. This approach solves the issues presented earlier: there is no longer a need to memorize a username password pair for every service, since there is no longer an identity for each of them. It can also be implemented in a manner that limits or completely inhibits profiling and linking of profiles across services. Decentralized identity also serves as the foundation from which SSI systems are developed, which solve the physical problems listed earlier.

The solution, already in place in many services, is a system called self sovereign identity. The principle at its core is that the users own their identity, and they can choose which of their personal information they are willing to share with an interested third party or service, as opposed to the current system. In an SSI system there are three core elements: the holder of the identity, which is the user who collects credentials, for example a digital identification card or driver's licence, in their digital wallet; the issuer, which is the entity responsible for the issuance of the credentials that users hold, an example being an university issuing diplomas to alumni; and the relying parties, which represent the services that require the user's information to grant them access to some part of their services.

This kind of system puts identity in the hands of the user, and allows them to choose what information they want to reveal to a third party. This paradigm shift would need widespread adoption, specific infrastructure to support it, as well as a set of regulations that would define how exchange protocols are established, how trust is set for the issuers, as well as everything else relating to the ecosystem. To this end, the eIDAS regulation was created. It aims to create an environment where citizens have access to a seamless electronic identification ecosystem across Europe, while also granting businesses guarantees of the legal status and trustworthiness of the system. eIDAS also sets specific timelines for the goals it sets out to achieve to be completed by each member state.

1.1 Thesis Goals

This dissertation has two main goals, both of which split into a set of more concrete objectives. The first objective is to do a comprehensive literature review and aggregation of the relevant technologies to the future eIDAS aims to achieve, and the second, more practical one, is assembling some of the covered technologies to perform a simulation of an SSI ecosystem.

The first goal comprises most of the theoretical portion of the dissertation and can be split into researching SSI and its underlying technologies, those being Distributed

Ledgers, Decentralized Public Key Infrastructures, Single Sign-On technologies, as well as Decentralized Identifiers and Verifiable Credentials. It also includes doing a review of the eIDAS regulation and the Architecture and Reference Framework, as well as of the European Digital Identity wallet and relating these to the SSI technology studied before.

The second goal has a theoretical and a practical component. The practical component is the goal of establishing a simulated SSI ecosystem, which can be split into having a functional Verifiable Credential and Presentation Schema framework, as well as the Holder, Issuer and Verifier core that is capable of issuing and verifying said credentials, as well as having an identity and trusted issuer registry to make the previous process trustworthy. The theoretical portion includes studying and understanding the technologies that will support the practical component, mainly waltid and the EBSI network.

1.2 Thesis Contributions

This dissertation's gathering of academic studies and technologies relevant to the eIDAS regulation works as an in-depth introductory study to anyone who wants to study and expand the work eIDAS aims to do. The comprehensive explanation of the technologies adjacent to eIDAS, as well as that of the systems underlying SSI technology and all its requirements, complete with an explanation of the relationship between both and the inherent need of SSI that eIDAS has, makes it a solid article for those who want to pick up either SSI technology or work to further eIDAS's goals. The dissertation's SSI prototype serves as a solid foundation from which a more robust and complete SSI ecosystem could be built and developed, and the technologies explored adjacent to it are also of great worth to anyone wanting to advance eIDAS.

1.3 Thesis Layout

Chapter 1 gives an overview of the thesis's field of study as well as the motivation for the work, while also setting the goals for it. It also provides the layout of the document and a short explanation of each chapter.

Chapter 2 is an introduction to the concept of digital identity, a comprehensive exploration of the current SSI landscape and the underlying technologies, a complete introduction to Single Sign-On systems, as well as also covering the European eIDAS framework and the ARF.

Chapter 3 provides a more technical view of some of the systems underlying SSI and eIDAS, specifically those engaged with during the practical portion of the dissertation, which are the waltid application and the EBSI network, as well as illustrating some core Verifiable Credential Schemas.

Chapter 4 goes over the implementation and how the practical portion of the dissertation was created, with waltid at its core.

Chapter 5 is a conclusion chapter and reflects on the results achieved as well as on future work.

Chapter 2

State of the Art

2.1 Digital Identity

2.1.1 Defining Identity

Identity can be defined by multiple different attributes and in different ways depending on context. An intuitive definition would be the set of attributes that define and identify an individual, ranging from birth date to eye colour and to political choice. This definition is in line with Merriam-Webster's Dictionary's definition [1] but even though it is a common word whose meaning most people would understand, the same people would have some trouble providing an apt and all-encompassing definition for it.

Paul Ricoeur spoke of identity in a way that concludes with "identity can be defined as the representation of an entity through features that make the entity unique and/or persist through time, in a given context."[2] This definition is close to the modern definition but is not enough, since the current definition describes identity as the set of attributes that uniquely identify an entity over a lifetime [3].

2.1.2 Digital Identity

In a digital context, the requirements of identity are easy to list: it must be the set of attributes able to represent an entity. Phillip Windley defines digital identity as collections of data about a subject that represent their attributes, preferences and traits [4].

Kim Cameron suggested the Seven Laws of Identity, while defining identity as a set of claims made by a digital entity about itself or another. The laws are as follows [5]:

1. Information about a user must only be revealed or shared with the consent of the user;
2. When information is to be shared, the amount shared should be minimal, its use as limited as possible and its lifetime as short as needed;
3. Disclosure of information should be limited to parties that are needed and justified in participating in the process, and the disclosing user must be made aware of all participating parties;
4. Identity systems must allow both public and private identities, such that the system does not allow correlation and identification of identities by unauthorized entities;
5. The identity system must support interoperability between established identity technologies;
6. The system must integrate user experience in its design, with usability and security as the focus;
7. The system must create a consistent user experience across identity contexts;

Cameron's definition and laws fit the design of an SSI system, as well as that of the EUDI (European Digital Identity) perfectly, as it not only gives a concise and programmable definition of identity, it also provides a set of rules to follow when designing the system it will depend on.

2.1.3 The missing Identity Layer

The Internet was built without an identity layer [5]: the way the Internet was built does not allow a user to know who they are interacting with, nor does it allow for the other party to identify the user, at least not by design. The current paradigm depends on a jumble of singular identities, that is to say a user has to have a unique identity for each Internet service they use and as such they have no identity of their own.

This comes hand in hand with the fact that a user does not own any of their identities, they are owned by the services, which results in a silo-based mapping of identity, each service being a silo of identities useful only to itself. The Internet's lack of ability to identify people means each individual service must do that itself, which also results in

an environment where a username and password system thrive, leading to horrible user experience.

Opposed to this is the idea of a singular, portable, user-bound identity, that each person should own and be the one capable of managing and keeping it up to date. This identity should be portable across services on the Internet, but a lazy implementation of this kind of system would lead to more problems than solutions, for example it would lead to trivial linkability of the user's services which would be a breach of privacy. Therefore, a security focused development of this portable user-bound identity is of great importance to the current Internet identity environment.

2.2 Identity Management

Identity and Access management are terms used to define the administration of individual identities within a given system, as well as the permissions associated with them. The main functionality of Identity Management systems is to increase security and productivity, while decreasing the cost, downtime, and repetitive tasks. These include user creation, user deletion, user locking and unlocking and granting and revoking access [6]. Some of the functionalities that are expected of an IdM are: authorization, authentication, directory services, provisioning, workflow automation, delegated administration, password synchronization, self-service password reset, policy based access control, enterprise and web single sign-on, identity repository/directory services, metadata replication/synchronization engine and workflow application development. [7]

The core participants in a traditional IdM system are the user, the Service Provider (SP) and the Identity Provider (IdP), which is responsible for both user registration and identity data storage, as well as being responsible for processing requests from the SP for user authentication. IdM models can be divided into centralized, federated and decentralized, what follows is an overview of each.

A centralized identity management model is one where there is a central IdP responsible for both the storage and the authentication of all identities, and all SPs use this global IdP. Within a centralized identity management, the company is in control of law-enforcement and there is no need for further interoperability. [8] It is still used, even being the first step in the evolution of IdM models, and some of its main problems are a lack of interoperability and trustworthiness.

A federated identity management (FIM) model is one that integrates different domains and makes them virtually a global unique domain[9]. This allows for a system where secure and seamless cross-domain authentication can happen. A user can have accounts with multiple IdP as well, and the SP communicates with the relevant IdP for the required set of attributes [10]. One of the most common examples of FIM is logging into a service using a Google account, as it is very prevalent in the current identity landscape.

A decentralized identity management model, also known as a user-centric identity model, was developed in response to the problems raised in the previous chapter, such as the lack of control users have over their own digital identity and the lack of interoperability of older systems. Despite the solutions given by current Identity Management technologies to improve the management of user authentication and resources access, they still suffer from several limitations, and they are not optimal to ensure data protection against abuse, fraud and criminality [11]. The core principle of an user-centric identity management is the idea of the user being in control of their own identity, and it is one of the main themes explored in this dissertation.

2.3 Single-Sign On

Authentication, in computer science, is the process of proving a user's identity is genuine. Historically, it has been based on a username and password system and over time it has evolved to two and then to multi-factor authentication. This means more than the password is required from the user to prove their identity, such as access to their email or phone, or biometric information.

Single sign-on (SSO) is a mechanism that uses a single action of authentication to permit an authorized user to access all related resources without being prompted to log in again at each of them during a particular session [12].

SSO systems rest on a central authentication service that other applications rely on to log in users. The skeleton for an SSO protocol is as follows: when a user wants access to an application that requires authentication, the application sends the user to the central service, where the user logs in resulting in them being sent back to the original application with their identity data. This allows the user access to the application, but if the user also wanted access to another application or resource within the same central service sphere and of the same access level, they would be able to access it without needing to authenticate a second time.

2.3.1 OAuth 2.0

OAuth is an authorization protocol, designed with the goal of allowing access to a set of resources. It does this by means of Access Tokens, which represent the right to access the resource. An entity with a specific Access Token would be allowed to use a specific resource.

By its defining specification: "The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf." [13]

Although it is officially recognized as an authorization protocol, OAuth is in fact a delegation protocol. " OAuth itself does not carry or convey the authorizations. Instead, it provides a means by which a client can request that a user delegate some of their authority to it." [14]

For an example use case: you are using a cloud storage service to store photos and you want to print them using a cloud photo printing service. Assuming the photo printing service and the storage service can communicate with each other through an API, the pictures can be exchanged smoothly, but since both services are under different corporations, the storage service will deny the printing service access as your accounts on each site have nothing linking them by default. OAuth fixes this scenario because it lets you delegate access to your photos to the printing service without actually granting them access to your storage service account.

The key parties in the OAuth 2.0 protocol are: the Resource Owner, who owns the resource that needs to be accessed (for example, an end user); the Client, the party who needs access to the resource; the Authorization Server (described later on), a party that mediates the exchange, receiving the access request from the Client and sending the authorization request to the Owner; and the Resource Server, where the resource is located.

Looking back on the photo printing scenario, the resource is the storage site API, you, the user, are the Resource Owner, the Client is the photo printing service who needs to access the photos. The final goal is for you to be able to delegate a portion of your rights over the photos to the printing service: you do not want to allow access to all your photos, nor do you want the printing service to have deletion or writing rights in the storage server.

This is reflected in the scope of the desired resources, that is, when a Client requests access to resources from the Owner, they will not be granted access to the whole set of resources. Instead, they specify a scope of the set they will be granted access to and the authorization protocol only allows access to that.

One of the methods to achieve this in the past was simple credential copying: the client would ask the user for their credentials so it could access the resource using them. Aside from the obvious problems that arise from granting your username and password to a third party that has no right knowing these, this also grants them access to the whole resource space, since the specified scope does not exist in this method.

In the photo printing use case, these problems become clear: since the photo printing service is using your credentials, the storage service has no way to distinguish them from you, so they cannot deny the requests in case the photo printing service asks for photos to be deleted, or to access photos beyond the scope you have defined.

Another method would be for the user to generate special one time passwords to allow access to specific subsets of the resource space. But this is not only very heavy on the user side, being as far from user friendly as it gets, but it also leaves the onus of revocation on the user, since the management of the one-time passwords does not fall on the resource service. Not only this is too much of a burden for the user, but it is also rather unfeasible.

The OAuth solution adds the Authorization Server to the set of involved parties. The Authorization Server is trusted by the service with the resource that is being accessed and acts as an intermediary that issues specific access tokens.

The protocol is as follows, assuming the Client has acquired an identifier and secret to authenticate itself before the Authorization Server:

1. The Client sends an authorization request to the Authorization Server, along with its identifier and secret for self-authentication, as well as the scope of the desired resources.

2. The Authorization Server authenticates the Client and verifies if the scopes are permitted.

3. The Resource Owner authenticates themself before the Authorization Server, verifies the Authorization Server's request and chooses whether or not to grant access (the following steps presume access is granted).

4. The Authorization Server redirects an Access Token or Authorization Code to the Client.

5. The Access Token allows the Client to access the desired Resource OR

6. The Client exchanges their Authorization Code for an Access Token

Alternatively, the user can do the verification of the scope manually in step 3 as opposed to automated in step 2.

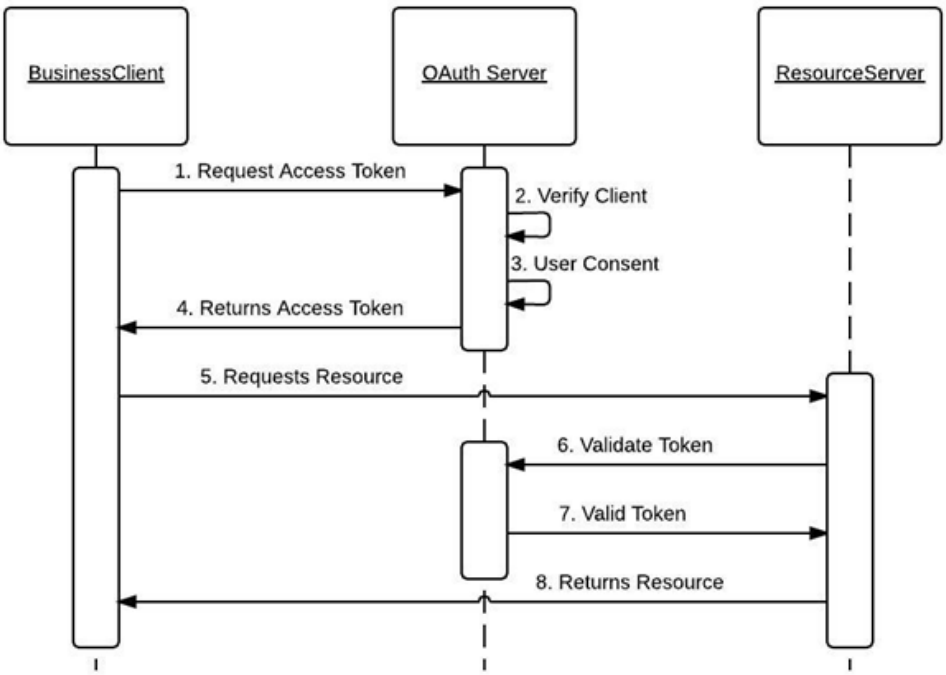


FIGURE 2.1: A basic OAuth protocol. [15]

2.3.2 SAML 2.0

Security Assertion Markup Language (SAML) is an XML based authentication protocol designed for cross-domain SSO. It is widely adopted and mainly used in government and business. [16]

The key parties in the SAML protocol are the Identity Provider (IdP) who is responsible for managing user credentials, the end user and the Service Provider (SP), who holds the service the end user wants access to and who interacts with the IdP to authenticate the user. The protocol depends on SAML Assertions, digitally signed XML documents with user identity data stored in the form of attributes.

One of the main advantages of SAML 2.0 is how well established it is: it has been in use for nearly two decades and is used by a great number of organizations. On top of that, it has many options and functional capabilities available which allows it to cover many identity requirements.

It also has a single point of authentication, the secure IdP, which can be a benefit, but also means there is a single point of failure. It also features increased compliance due to the ability to receive and forget attributes at will, enhancing privacy.

Before the protocol begins, the IdP and the RP need to exchange metadata which includes endpoint addresses, cryptography certificates and supported connection methods, among others.

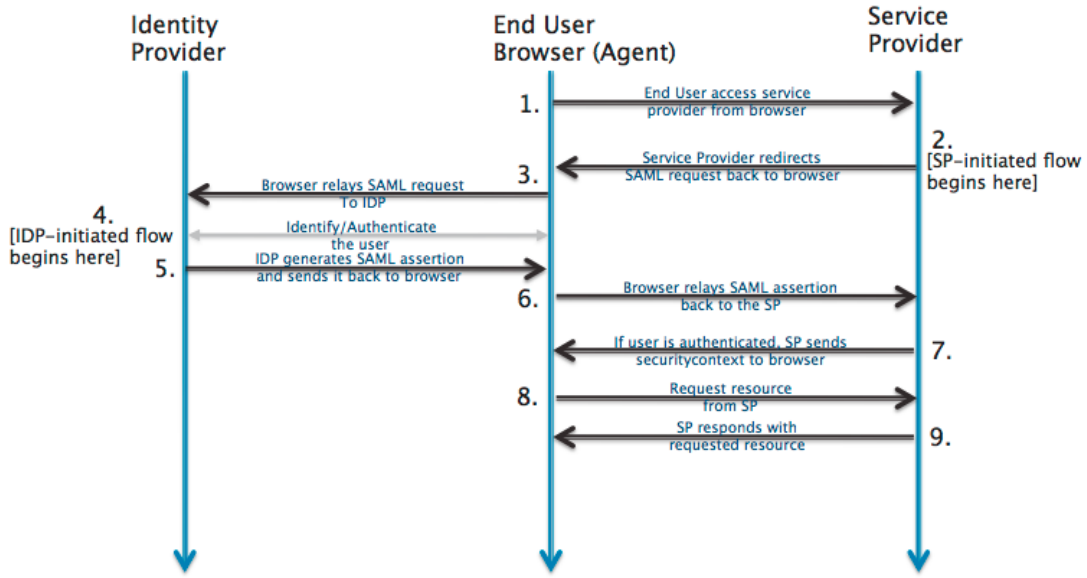


FIGURE 2.2: The SAML protocol. [17]

2.3.3 OpenID Connect

OpenID Connect is an identity layer that allows for both normal authentication and SSO. It makes use of OAuth 2.0 for its authorization mechanisms and builds an identity system on top of it, alongside JSON Web Tokens.[18]

A common sight on the Internet nowadays is the option of logging into services online with Google or Facebook accounts, or other options depending on the website. This is feasible not only but mainly due to OIDC, also being achievable with OAuth 2.0 but in a less secure and efficient way. OAuth access tokens can be of two types: reference tokens, which is an opaque token with meaning only to the authorization server, since the client

has to present the token to it; and a self-contained identification token which carries some basic information about the user, client, and scope.

The problem with the use of just OAuth 2.0 for identity is that the access token it uses is a tool to be presented in order to gain access to a resource, not a token for usage by the client itself. The Facebook flow has the user authenticate before Facebook which grants the client application an access token, but this does not immediately allow it to authenticate the user; instead it is used to access Facebook's own authentication API, and then the user has to be authenticated.

To avoid this, OpenID Connect defines both an authentication token, the ID token and its specification, and an authentication flow, which specifies how the token is to be safely transported to the end user.

Similarly to SAML, the key parties are the IdP, who deals with user authentication and consent, as well as token issuance, the RP, who requests the user's identity, and the user itself. On top of this, OIDC uses three different JWT tokens:

1. ID Token: provides information on the authentication's outcome; it may have the user's identity data as well as a profile, called a claim.

2. Access Token: token from OAuth 2; temporary and optional, it permits resource delegation

3. Refresh Token: also a token from OAuth 2; a long-lived token that is used to get new Access Tokens

An ID Token carries with it claims and assertions which are cryptographically backed. The types of assertion it is designed to issue are attribute assertions, authentication assertions and authorization assertions.

1. **Attribute Assertions** are statements about a quality or attribute of someone or something, for example a sports club membership card might have its owner's name, age and its own expiration date, all of these being attributes;

2. **Authentication Assertions** are statements describing how the ID Provider authenticates a user;

3. **Authorization Assertions** are statements that grant someone or something authorization to do or access something. For example, an attribute assertion claiming you

are above 18 years old can be used to allow you to purchase alcoholic beverages, but an authorization assertion claiming you are allowed to purchase alcoholic beverages leads to the same result without sharing your age.

OAuth 2.0, on top of which OIDC was build, has three different flows of operation: in the Implicit Flow, the access token is returned directly to the RP; in the Authorization Code Flow, the tokens are not returned directly; and the Hybrid Flow mixes both, not returning the Access Token directly and instead returning an Authorization Code.

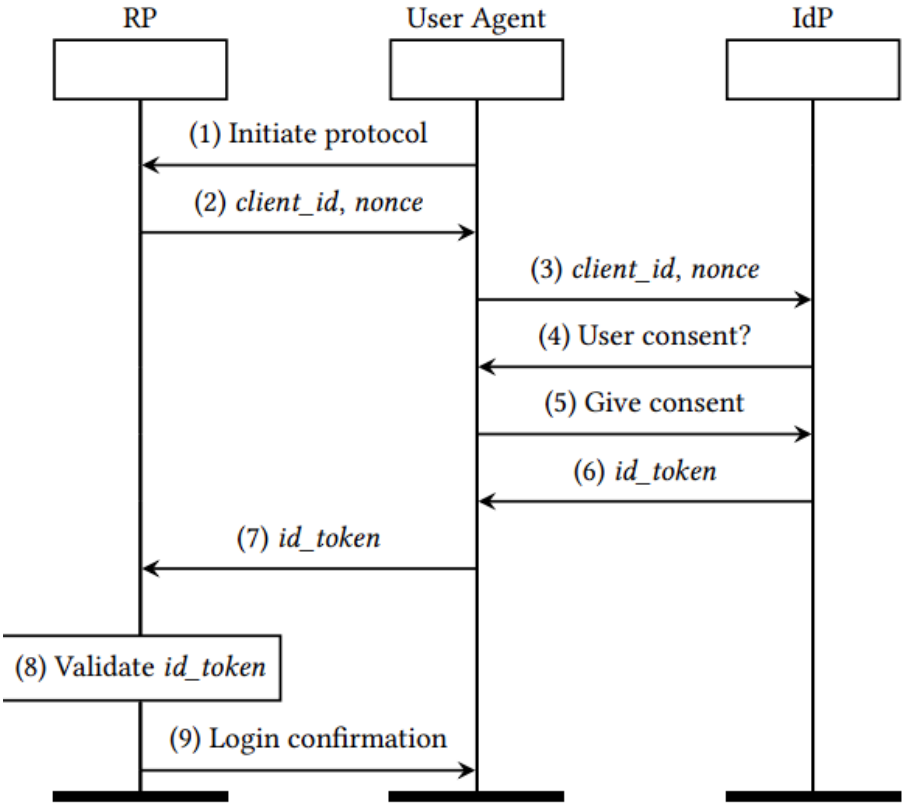


FIGURE 2.3: Simple OIDC protocol. [19]

2.4 SSI: Self-Sovereign Identity

SSI is a solution to the digital identity landscape problem presented earlier: it aims to be the well thought out and security and privacy focused system to implement a user-centric portable identity. This identity is by definition decentralized, the goal of it being to move away from the centralized digital identity model pervasive through the Internet.

There is much discussion about the exact definition of Self-Sovereign Identity: Christopher Allen [20] first spoke of SSI in a forum post where he addressed the current problems with digital identity and listed what he believed to be the core principles of SSI.

1. **Existence:** Users must independently exist;
2. **Control:** Users must have full control over their identities;
3. **Access:** Users must have access to their own data;
4. **Transparency:** The systems and algorithms that govern the ecosystem must be transparent;
5. **Persistence:** Identities must be long lived;
6. **Portability:** Information and services about identity must be portable;
7. **Interoperability:** Identities should be usable in as many services as possible;
8. **Consent:** Users must agree to the use of their identities;
9. **Minimization:** When disclosing information, as little as necessary must be disclosed;
10. **Protection:** The rights of the user must be protected.

The W3C broadly defines SSI as a claim-based system where users have control over their own identity, having an outlined structure where there are issuers responsible for supplying users with verifiable claims [21]. Users will then be able to present these claims, fully or partially and in an unlinkable manner to a relying party and be granted access to some service or resource. This system either follows or allows implementation of the core principles put forth by Christopher Allen. Once a claim has been issued and is in possession of a user, they have full access to their own data, and since their identity is in the hands of the user, they have full control over their identity. The existence of the user is guaranteed if the issuer has checks in place to avoid, for example, bots having access to an identity, which should be a realistic part of its system. Transparency and Portability will always depend on factors outside this definition of the system, but it does allow for both. Persistence is covered by the identity the user owns being permanent bar special cases, such as a loss of keys, with only claims having a possibility for a validity date. Consent is inherent to the system as users are the ones willingly presenting information about them,

and minimization is covered by an implementation of selective disclosure. This way, the rights of the user should mostly be covered, which means Protection is also covered.

This system is what is at the core of modern SSI systems, being a skeleton covering its basic requirements while also providing good feature scalability.

A Verifiable Credential is an open standard for digital credentials that are at the core of SSI systems - for example, a driver's license or a university diploma would each be represented by a VC, held by a user and presentable to a verifying third party [22]. A Verifiable Presentation (VP) is a structure that assembles multiple VCs into a tamper-evident document, which is used during the presentation part of the SSI system.

Preukschat and Reed go over the building blocks of an SSI system in a comprehensive manner, listing its main components as: Verifiable Credentials, the trust triangle of Issuers, Holders and Verifiers, digital wallets and agents, decentralized identifiers, blockchains and governance frameworks [23].

A credential can be a birth certificate, or a driver's license, since it is a set of claims about its subject, made by some authority. Claims are able to cover attributes, relationships or entitlements of the subject. A physical credential can be easily verified as most of them have an embedded proof of authenticity, but for digital credentials this becomes more troublesome. There is the need to prove who issued the credential, its integrity and expiration date. Cryptography solves these issues, as a digital signature coupled with a hash can solve both the "who issued" problem, as well as verify integrity, and if the hash has a timestamp and expiration date covered, those are also ensured along with integrity.

Issuers are responsible for the generation and supplying of credentials and they can be organizations such as universities, hospitals, financial institutions, among others. Holders are the entities requesting claims from the issuers and presenting them before relying parties. Important to Holders are digital wallets, where they will keep the claims, which have some SSI specific requirements: the need of the implementation of open standards for VCs and other personal data, and the need to work with digital agents. These wallets should be easy to use and install on any device, as well as being compatible for back-up creation. Later in this article, the European Digital Identity Wallet will be explored in depth.

Digital Agents serve as an in-between for wallets to interact with each other or with relying parties: it is the software required to replace the human part of the process of finding who or what to present your claims to. There are two types of agents, edge and

cloud agents. Edge agents are responsible for a holder's wallets and local devices while cloud agents are hosted by cloud computing platforms and can also serve for file storage.

Decentralized Identifiers, or DID, are the SSI equivalent of IP addresses for the SSI ecossystem. The result was the creation of DIDs, a new type of identifier that had to meet the requirements of being permanent, resolvable, cryptographically verifiable, and decentralized. By being a resolvable identifier that is tied to a public key, as well as other data, if necessary, it can be cryptographically verified, and by being stored in a blockchain it becomes permanent, as well as decentralized.

Governance frameworks aim to fill the void left by cryptographic trust, that is, human trust, by having governance authorities design and publish frameworks that function as legal rules for the SSI ecosystem to enable an interoperable digital trust.

The basis for this system is a blockchain, which will take the place of the central identity registry in a traditional system, allowing the so desired decentralization. While user claims will still be stored elsewhere, an asymmetric cryptography system in combination with decentralized ledger technology is sufficient to replace the traditional architecture. As stated by Mühle et al., an Identity registry, that is, a public blockchain where users' public keys are linked to their DIDs is enough to achieve identification and authentication [21].

The claims themselves should be stored off-chain and not be publicly available, and can be secured in two ways: by being linked to the identifiers in the Identity Registry, which allows for their association when being presented, or by being linked to an extension of the Identity Registry, named Claim Registry by using their cryptographic fingerprints.

After having the core elements of an SSI system, what is left to do is defining the protocols and how each interaction in the system should work, such as authentication, authorization, issuing, and all the rest.

It is important to go over the process of revocation early, as it is relevant to every other process in the context of verifying if the credential or claim being checked is no longer valid. One of the systems for revocation is also relatively simple because it is based on the existence of a Revocation Registry - similar to the Identity Registry but instead used to store pointers to claims and credentials that have been revoked. If an issuer wishes to revoke a claim or credential all they need to do is register its DID in the Revocation Registry. Another scheme for revocation was designed making use of chameleon hashing

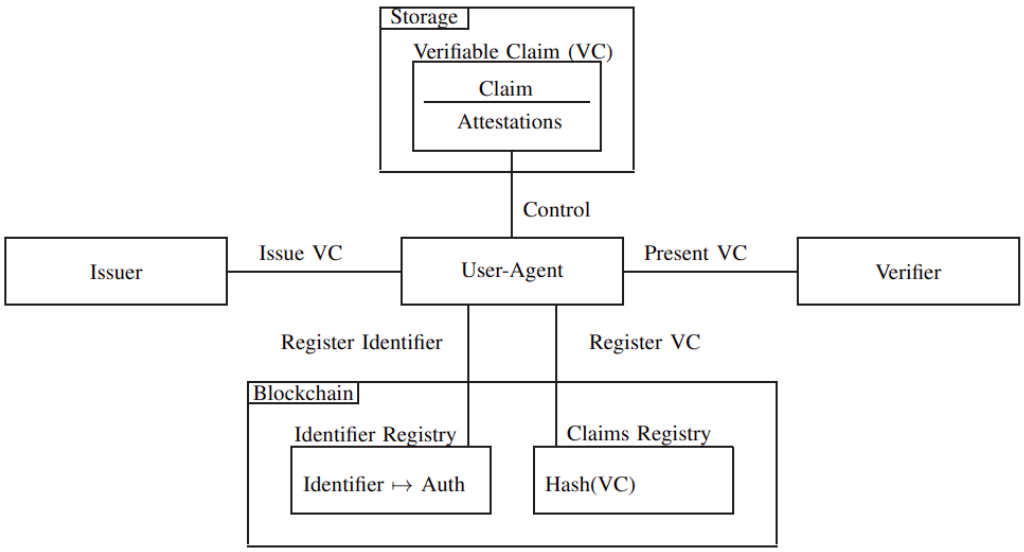


FIGURE 2.4: The core SSI system.[21]

[24], which is a type of hashing that incorporates a trapdoor, allowing the hasher to find a collision as long as they have their trapdoor key, which was then used to redact blocks from a blockchain to delete and as such revoke certain information within it [25].

Authentication is the process where a party proves to the other that they are who they claim to be, and its functioning has already been hinted at earlier. If a user wants to authenticate themselves before an issuer or relying party, or wants to authenticate either of those, all they need to do is provide proof of ownership over their private key, such as by a signature. The authenticating party can then check it against the alleged identity's public key registered in the Identity Register, which should be publicly available. This is achieved by doing a DID lookup on the blockchain and finding the user's public key. Once the checks are done, both parties can recognize each other as who they claim to be. [26]

Authorization is the process where a party who provides a service or owns a resource verifies if another party requesting access has the permission or credentials to do so. The party who controls the resource or service will request the necessary proof of the party who wants to access them - an example would be a user wanting to create a bank account and the bank requesting proof of the user's age. The user would then present the smallest set of claims that meets the requested criteria - in the example case the verifiable credential

that holds the information could be a driver's license. Then they should selectively disclose their age, possibly through a zero knowledge proof, resulting in the bank knowing they are of age without knowing anything more about them, not even their age.

The authorizing entity would then need to verify if the claim itself is valid, which includes checking: if it is not yet expired, if it is in fact about the user that is attempting to get authorization, if it has not been revoked, and if the issuer of the claim is trustworthy. The exact technical process here might vary depending on how the claims are structured, but an example would be the authorizing entity looking up the issuer DID on the Identity Registry, checking if it is an issuer they trust, verifying their signature against their public key and if it matches. Finally, it would check if the claim was made about the user requesting authorization, its expiration date, as well as looking up its DID in the Revocation Registry to check if it has been revoked. If everything is in order, access is granted to the user, and the bank would allow the individual to create an account in this example.

After a user has been authorized, the process of delegation follows, and it can vary depending on what exactly the authorization is for, and on how important the privacy or exclusivity of the resource or service in question is. For example, if the user got authorization to access a low clearance company file, it could simply be sent to the user through an encrypted channel and the process is over, with the user now having access to the resource they needed.

If, on the other hand, the user wants access to a high clearance company development environment, a more complex process is in order, to guarantee the user only has the correct permissions to act within it, as well as assuring accountability, although these specifics change from case to case. For this process, the best solution might be Single-Sign On technology, which is explored in the following chapters.

A system for delegation of credentials was proposed by Lim et al. where a delegatee would only have access to an encrypted version of the credentials, which would limit their actions, while also needing the original owner of the credential to accept or deny the use of the delegated credential by the delegatee. [27]

Another important topic that comes up when granting more control to users and making them responsible for holding their own information is back-up and key recovery mechanisms. A compromise of security and usability is reached by Singh et al [28], where the user has to answer a set of questions, and the input is encrypted and used to produce a recoverable key. Another method that has been proposed is the declaration of

trusted peers by the wallet owner, which are then used to recover a lost key [29]. Linklater et al. devised a system where a Certificate Authority rotates certificates to users periodically, claiming that key recovery is guaranteed as long as the CA's private key is safe. [30]

Data minimization is an important aspect of SSI, and there have been multiple studies in the area. Zero Knowledge proofs are a mechanism that allows a statement to be proved true or false without disclosing the information that proves it, for example, proving you are over 18 years old without providing your exact age. A ZKP system using Water's signature along with BLS was designed by [31] and enables the selective disclosure that is ideal to SSI systems.

Another important problem to solve in SSI is the reuse of information presented in credentials, for example, a Relying Party could reuse the verifiable presentation given by a user. Kang and Lemieux try to solve this issue by making use of Fully Homomorphic Encryption, a method that allows the usage of data without decrypting it, skipping selective disclosure entirely and using a trusted third party to process the user data. this makes it so the RP cannot reuse the user data, but introduces the possibility of user personal data being revealed during the processing of the information. [32]

Trust models are of great importance to an SSI ecosystem as they can make the process of deciding which issuer to trust much more fluid. A study developed a system that creates an endorsement graph by aggregating endorsements by the members of the system. The graph is then navigated to calculate trust scores for each participant [33]; this model was later further improved by adding a time element to it. [34] Zhong et al. developed a system that employed cross-chain smart contracts to compute a credibility score from the performance assessment of each member [35]. A probabilistic model was also designed which calculated the probability of a set of claims to be trustworthy based on past performance. [36]

The adoption of a SSI ecosystem and how well it will spread to mainstream use heavily depends on how invested the decision-makers are and how dedicated they are to making it work, and also on how many sacrifices they are willing to make for it to work [37].

2.4.1 Trust Systems

Issuers are a core element of a SSI Ecosystem, but a problem inherent to this kind of system is knowing who to trust. For example, a Verifier who receives a presentation

showing a Holder's university diploma signed by an Issuer needs to know if the entity that signed the diploma is trustworthy or not. To this end, a trust chain is established and entities responsible for maintaining the trust chain are put in place. The core of this system is the Trust Registry, a registry where Credentials relevant to trust are stored.

There are three types of credentials: Verifiable Accreditations, Verifiable Attestations and Verifiable Authorizations, these two being variations of the first which are studied in Chapter 3. A Verifiable Accreditation is a document issued by a trusted entity (with all trust originating from some root source of trust in a trust chain) that accredits another entity with the trust to issuer credentials up to a certain level, referencing it by its DID. An entity can only issue the type of Accreditations for which it is accredited to issuer Credentials [38].

2.5 Distributed Ledger Technology

Distributed Ledger Technology (DLT) provides a distributed peer-to-peer system for storage of data without any mediation from a central authority.[39] The most prominent type of DLT is the blockchain, a data structure that ensures immutability by "mining" for hashes of blocks of transactions through a grindy trial and error process.

A blockchain uses proof-of-work to achieve immutability and solve the double-spending problem. Proof-of-work consists of hashing a timestamped block of transactions into a continuous record, making every block from the start of the chain up to the currently hashed one impossible to change without making the hash change as well. The downside of this kind of hack resistant system is that it requires an extremely high level of energy consumption and the proof-of-work trial and error process is very costly, making it expensive both financially and environmentally.

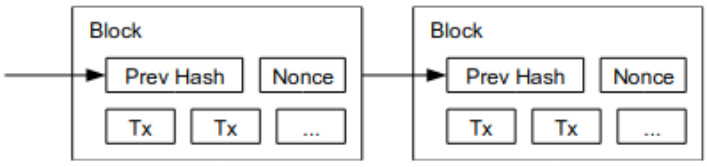


FIGURE 2.5: A simple overview of the PoW process. [40]

2.6 Decentralized Public Key Infrastructure

A Public Key Infrastructure (PKI) is a widely adopted type of structure whose main goal is the generation and supplying of public key certificates, as well as the storage of these and any other relevant information. It provides an interface for users to certify other users and their public keys, allowing them to vouch for their reputation, and it also allows users to retrieve keys from other users along with proof of the key's legitimacy. [41]

A PKI has to provide several services: registering an identity and public key pair, updating it, looking up keys by identity, verifying if a key matches an identity, and revoking a key pair.

A Decentralized Public Key Infrastructure (DPKI) is a set of changes made to traditional PKI in order to make it more usable and secure mainly in IoT scenarios. These are relevant for the EUDI Wallet since many of its interactions will involve IoT mechanisms, such as card readers, and all interactions that use them must be secure.

Won et al. lists all the obstacles that general PKI face when it comes to dealing with IoT mechanisms: [42]

1. IoT devices are an environment where it is difficult to manage key certificates since there are no standard protocols for installing, retrieving and updating the certificates designed for these devices;

2. IoT manufacturers generate and install the keys used by the IoT device as a result of the barrier to installing certificates. This results in the manufacturer knowing the keys used by the device which constitutes a massive security risk as they can reveal the keys to any third party if they so desire;

3. PKI is a structure with a single point of failure by nature, i.e. if any Certificate Authority is compromised, the system has to stop authenticating and generating certificates until all members have removed that CA from their trusted list. Anything within the network that was signed by the compromised CA is no longer trustworthy since it may have been signed by the bad actor. This problem is not inherent to IoT, but is worth mentioning nonetheless;

4. Due to the lack of a standardized way for IoT devices to update their certificate related data, it is very likely that they will be missing updates to root certificate list. This means they are open to being deceived by compromised or outdated CAs,

which could lead them to authenticate devices used by bad actors using compromised CA certificates;

5. Online Certificate Status Protocol responders are vulnerable to DDoS attacks, another issue that is not inherent to IoT PKI but should also be mentioned.

Bearing this in mind, it is recommended to have a DPKI for the adequate functioning of the EUDI ecosystem, to avoid the problems inherent to a PKI and to solve the issues specific to the IoT elements of the EUDI system. Therefore, a blockchain solution is the most intuitive solution, but IoT devices have another issue with a blockchain system, namely their limited resources and processing ability. This leads to the need of a more customised solution to the problem.

Won et al., listing the problems IoT has with PKI presents a solution to the problem by designing a distributed system to replace the PKI ecosystem [42]. The proposed system has distributed blockchain nodes taking the role of Certificate Authorities, while replacing digital signatures as a means of binding an identity to a key with the proof-of-work process. This implementation's first boon is the removal of the central point of failure present in traditional PKI, as a result of the decentralization of the system. The users now can also generate and register their private/public key pair in the blockchain by using their smartphone or computer as an aid, which solves the problem of the manufacturer of the device generating and knowing its keys, since the user now generates them. This also gives the user the power to update or revoke their public key if needed. As a final advantage, this system proved to have a certificate status look-up time 25 times faster than the traditional system.

Isirova and Potii also devised a blockchain based solution using Hyperledger Fabric, but instead of delegating the resource's heavy work related to the blockchain to an aid device like in the previous system, they compiled the Advanced Reduced Instruction Set for Computer Machine 64 bit architecture with the goal of simplifying the process as much as possible while choosing a blockchain that fits their requirements. They also defended that full decentralization is not necessarily desirable as "it would take away the level of trust one may have in an identity that is stored on the distributed network." [43]

They conclude by asserting that, with the current direction of growth of the computational capacity of IoT machines, they should be able to support and maintain blockchain processes even in their restricted environment.

Fromknecht et al. designed a DPKI system with the goal of increasing identity retention [44], which is not directly related to IoT but is of interest to the EUDI Wallet ecosystem. They did this with Namecoin, a cryptocoin designed to function as a decentralized DNS, as its building block, by at first simply registering the identity/key pairs to the blockchain, calling this Version 0. Lookup operations were done in standard blockchain procedure, just searching for the latest instance of that identity. For Version 1, they introduced a cryptographic accumulator to increase key verification efficiency while also increasing space efficiency. In Version 2, the goal was the increase of public key lookup efficiency by implementing a distributed hash table, which reduces the lookup time from linear to constant. This combined with the cryptographic accumulator creates an efficient DPKI ecosystem. The compatibility of this approach with IoT could be a relevant topic of research, but is not within the scope of this paper.

2.7 Digital Wallets

Digital Wallets are a concept that appeared in the 1990s and were popularized by PayPal, a company founded in 1998 that provided a safe, secure and user friendly cyber-wallet, facilitating online financial transactions. Years later, with the advent of smartphones, digital wallets such as Google Pay and later the Apple Wallet became even more convenient, when coupled with NFC technology. In Portugal we have also seen the widespread adhesion to MBWay, a nationally available digital wallet provided by SIBS. A digital wallet is a virtual storage system that can capture one's identity and digital credentials.[45]

Digital wallets would let users manage multiple monetary and ID instruments and quickly search them by name, type, or other keywords. In addition, a digital wallet would enhance security as all data would be encrypted and backup options would make recovering from loss or theft easier. [46]

Nowadays, digital wallets are also used to store personal data and documents, just like a physical wallet would. Another example in Portugal is the ID.gov.pt wallet, which already allows the storage of the Citizenship Card and the Driver's Licence, together with eleven types of identification documents.

The general consensus of the definition of a digital wallet is that it is a storage for any and all information, credentials or documents that relate to an individual's identity and that are controlled by this same individual. The data stored in a digital wallet should be cryptographically protected by the user's key and the principle of data minimization

should be followed so that only the minimum required amount of information is revealed when presenting it, making use of selective disclosure.

In the context of SSI, it is important to distinguish between Organization Wallets and Individual Wallets. The former are used by companies, or other entities aiming to provide a service, be it issuance or verification, and the latter are wallets for personal use, made for natural persons. The European Blockchain Services Infrastructure, EBSI, has a Wallet Conformance testing portal, as well as a list of all the wallets that comply with it. Below are overviews of some of these types.

2.7.1 Organization Wallets

Organization wallets must be able to do at least one of the following: accredit/authorize, issue or verify. There are currently nine wallets for accreditation, twelve for issuance and nine for verification that are EBSI compliant [38]. Five of them fulfil all the requirements and seven are able to issue and verify. Most of these wallets support same-device and cross-device operations, and are supported by both web and mobile apps.

The most remarkable are the five capable of the three operations: PrimusMoney, developed by a company that specializes in development in the Ethereum platform; PwC-ID, the wallet developed by pwc; eKibisis Wallet, a wallet developed by Goldman; Enterprise Wallet, developed by iGrant.io, a company aimed at creating infrastructure for secure data exchange; and the CERTH SSI Wallet.

2.7.2 Individual Wallets

Individual wallets must be able to complete two tasks: request credentials and present verifiable presentations. In addition to that, all of the complying individual wallets support same-device and cross-device operations, as well as web and mobile applications.

There are ten EBSI complying digital wallets and some examples are: Teknei's EBSI Wallet, developed by a Spanish tech company; VeloxWallet, a wallet developed by Velox-soft; the DS Wallet, developed by the Portuguese company digitalsign.

2.8 eIDAS: electronic Identification, Authentication and Trust Services

The eIDAS regulation is responsible for developing a framework that has as its goal creating an environment where business electronic transactions are safer, faster and more efficient across countries in the European Union. eIDAS appeared as a result of each country starting to develop their own electronic identity systems with no cross-country interoperability in mind.

eIDAS addresses two major problems in the European digital landscape: the aforementioned ability for any EU member to authenticate before another member state using their electronic identification, and the enforceability of laws regarding trust services, since only electronic signatures were covered previously.[38]

2.8.1 Trust Services

The regulation provides a legal framework for electronic signatures, electronic seals, electronic time stamps, electronic documents, electronic registered delivery services and qualified certificates services for website authentication.[47]

Each state defines the legal value and the laws that govern these trust services, except when it is specified by eIDAS, and each member state is free to add other trust services to the list.

This allows trust service providers to have their product on the market while being safeguarded by law, therefore creating a safer ecosystem for their dissemination. On the other hand, this also introduces liability and responsibility for the providers, as they are now bound by law to ensure the quality of their services.

2.8.2 LoA: Level of Assurance

Since eIDAS does not intend to create an EU-level identity for every user, instead intending to create interoperability between member states's identity systems, it is important to keep in mind that member states will have different levels of privacy and security built-in to their identity systems. Therefore, a Level of Assurance scoring system was devised, including minimum levels that must be ensured by each state.

The LoAs are low, substantial and high, and they must ensure that a presented identity is in fact who it claims to be with low, substantial and high confidence, while being assessed with a set of defined standards and controls, according to Art.8 of eIDAS.[]

2.8.3 Cooperation and Interoperability

Article 12 of eIDAS covers the requirements for an interoperability framework and for the cooperation between member states.

The interoperability framework must guarantee that it follows European and international standards, facilitates privacy by design and lawfully treats user data. This article also states member states must exchange information regarding identification schemes and assurance levels, and ensure peer reviewing and examining developments in the identification scheme sector.

2.8.4 eIDAS 2.0

In 2021, the European Commission submitted a proposal regarding eIDAS, not with the purpose of replacing it, but of updating it regarding electronic identification. The main goal was to integrate decentralization and SSI in the regulation, while also improving certain areas of the previous iteration of the articles. eIDAS 2.0 should be functional by 2025 at the latest, and the goal is to have an electronic identification system in place that works across countries in Europe by 2030, making use of the EUDI Wallet, the European Digital Identity Wallet where personal data would be stored with the format of Verifiable Credentials.

One of the new regulations enforced by eIDAS 2.0 is that every member state must notify the relevant parties of at least one identity scheme and a unique EUDI identifier it supports. It must also provide at least one EUDI wallet solution to its natural entities, either provided by the member state itself, published under the authority of the member state or at least recognized by it. This wallet should hold the information relevant to the identity of its respective holder, such as a citizenship card or a driver's licence, and every citizen should have access to one of their own, which means that eIDAS 2.0 supports the SSI core.

Another point enforced by eIDAS 2.0 is the fact that only Qualified Attestation Services can interact with EUDI wallets. With that in mind, it establishes mandatory requirements for the creation of European Standards for both of these. In addition to this, eIDAS 2.0 links digital identity and trust services, that is, they determine each other.

eIDAS 2.0 also introduces trusted services for electronic ledgers which implies a minimum level of trust being given to the electronic ledgers in place.

With eIDAS having at its core the holder and their EUDI wallet, the Qualified Attestation Services and services wanting to verify the information within the EUDI wallets, the way it fits in an SSI ecosystem is clear. It already outlines the Holder, Issuer and Verifier core as its building blocks, especially since it has trusted specifications for electronic ledgers, elements such as Identity Registries and Trusted Issuer Registries can operate without having to keep in mind complex trust managing protocols.

The following section covers the Architecture and Reference Framework designed to guide the development of the EUDI Toolbox which supports the EUDI Wallet. It describes all the actors that should take place in the ecosystem, as well as use cases, the lifecycle of the wallet, potential requirements of the actors defined earlier, architectures and flows, and wallet configurations - all for the adequate operation of the EUDI ecosystem.

2.9 Architecture and Reference Framework

The Architecture and Reference Framework (ARF) is a set of standards and technical specifications defined by the European Digital Identity Commission. Its main goal is to allow the experts that will build the EUDI Toolbox to understand the objectives of the EUDI wallet, the purpose of each actor in the Wallet's ecosystem, as well as the wallet's requirements and building blocks. The document is not a legally binding set of requirements, instead it functions as a list of "shoulds" for the developers of the Toolbox.

To begin with, the ARF defines the primary use cases of the EUDI which are as follows:

1. Secure and Trusted Identification to Access Online Services: in addition to secure authentication, relying parties, both public and private should allow the usage of EUDI Wallets as a means of identification before them;

2. Health: the EUDI Wallet should allow access to patient data and prescriptions both nationally and cross-border;

3. Mobility and Digital Driving License: a fully digital driver's license should be allowed by the EUDI Wallet, as it enables many further attestations and legal requirements checks, both online and offline;

4. Education: education certificates are a costly and time-consuming venture for both the end user and the provider. The EUDI Wallet should allow the fluid usage of digital diplomas in a national and cross-border context. The EUDI Wallet should function as both a wallet for these diplomas and as a way of exchanging and sharing them;

5. Digital Finance: the high trust secure authentication created by the EUDI Wallet should allow financial transactions to use it as a main platform.

2.9.1 Actors in the EUDI

Understanding the actors within the EUDI Wallet ecosystem is crucial to have a proper understanding of it and what follows is an overview of the more important ones.

End users are the legal persons who will own a Wallet and the attestations and attributes to be shared, while also allowing them to create digital signatures.

Wallet Issuers are the entities tasked with making the EUDI Wallet widely available, being either Member States or entities recognized by Member states.

Providers of Personal Identification Data (PID) are responsible for making PID available to each respective Wallet, maintaining a means for providing the PIDs to the Wallets, as well as making sure Relying Parties have access to the necessary information to verify the PID.

Providers of Registries of Trusted Sources may be necessary as a source of trust for most of the other services listed in this section.

Qualified Electronic Attestation of Attributes (EAA) Providers have a self-explanatory name, being responsible for providing an interface for Wallets to get QEAAs, as well as a mutual authentication interface for EUDI Wallets. Non-Qualified EAA can be provided by any trust service provider (TSP), while Qualified EAA must be provided by qualified TSPs.

Authentic Sources are public or private repositories that are legally bound to be recognized by relying parties and contain the attributes about a legal person. The sources in the scope of the article range from address, age and gender to training qualifications and financial data.

Relying parties are the entities that request attributes or attestations from a EUDI Wallet, whether for legal requirements, contractual agreements or other potential reasons. Relying Parties need to inform the respective Member State of why they were established and of their goals, as well as of the reason why they need to rely on EUDI Wallets. They are responsible for authenticating the attributes and attestations they receive and must provide and maintain an interface for the Wallets so that it can request the required information.

Conformity Assessment Bodies perform audits on Qualified TSPs frequently and are recognized by the Member States.

2.9.2 Life Cycle of the EUDI Wallet

Following the actors in the EUDI, the ARF describes the life cycle of the EUDI Wallet, as well as that of its core components, PID and QEAA and creates a working model for each. ARF begins by presenting a simplified scheme of how the EUDI Wallet should operate, as follows:

The distinction between Wallet Solution and Wallet Instance is important: a Wallet Solution represents the service as a whole, while the Wallet Instance refers solely to a user's instance of the EUDI Wallet.

The life cycle of PID and QEAA is equal and interchangeable, and they can take the following states: issued, it exists but has not been activated yet; it then becomes valid after it is activated or its validity period starts; after this, it can either expire or be revoked, naturally overturning its validity. A PID cannot be renewed, that is, turned back to valid once revoked or expired, and the renewal process will always require a reissuing of the PID.

A EUDI Wallet Solution is considered to be in the Candidate state when it has been fully implemented, but it has not yet been verified and validated by Conformity Assessment Bodies. After it has been certified, it is considered to be in a valid state and the Member State can start using it to supply users with Wallet Instances based on it and the Wallet Solution is considered officially launched.

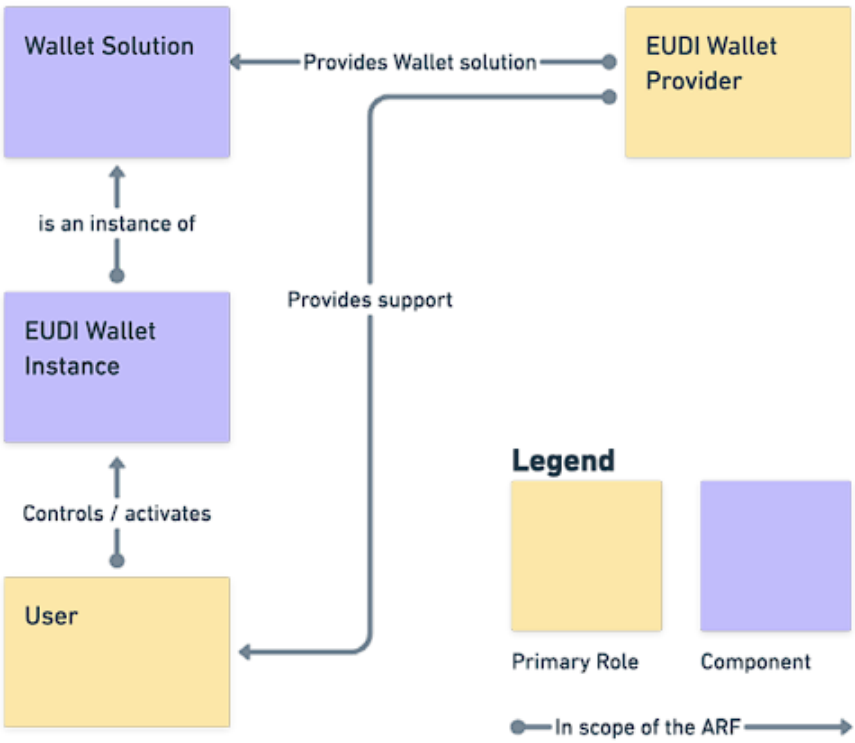


FIGURE 2.6: The ARF representation of a simple EUDI Wallet cycle. [48]

According to the legal document, a Member State can then choose to suspend the Wallet Solution, for example, for security reasons. From the suspended state, the Member State can choose to lift the suspension on the Wallet Solution, bringing it back to a valid state. The Member State can also choose to shut it down, bringing it to a withdrawn state.

The life cycle of the EUDI Wallet Solution is such that the state it is in will affect the state of the Wallet Instances below it within their own life cycles.

A EUDI Wallet Provider will provide an instance of a Wallet Solution to an end user which is in its operational state. A wallet in this state can be used for functions unrelated to the EUDI, but it will need to be assigned valid PID to move on to its valid state, becoming able to engage in EUDI operations. When this PID expires, the Wallet Instance is moved back to operational, maintaining operability, but losing its EUDI function until PID is reissued. The Instance can also be deactivated by its user.

2.9.3 Requirements Regarding PID and EAA

A PID Provider is responsible for issuing PID to a EUDI Wallet, and the legal document states that the means to create and supply PID are to be decided by each member state and are only restricted by legal requirements like GDPR and LoA High.

This starts by establishing core principles for PID. To begin with, PID should be such that no two distinct persons have a matching PID set of mandatory attributes. The PID set should contain at least the attributes considered mandatory by the eIDAS regulation and the set of mandatory attributes should be limited by the intersection of what all members states can provide for their users as attributes.

The attributes required by the eIDAS regulation are current family and first names, date of birth and a unique identifier, and the additional attributes suggested by the ARF are nationality and other attributes used at national level, such as the social security number. Besides this metadata relating to the PID, it might include date of issuance and expiration, issuing entity, information and location necessary for services regarding the PID.

The chapter is followed by the set of rules which must be followed by PID and QEAA Attestations. These include basic security primitives such as means of verifying authenticity and integrity and cryptography mechanisms, but most notably they also feature the need for both PID and QEAA attestations to support the selective disclosure of attributes.

2.9.4 Reference Architecture and Flows

This section covers the needs of a system that should support multiple situations where either the user or the relying party or both are offline, while also accommodating the Member States' need to implement the Wallet with different configurations and elements.

To this end, a reference architecture was designed with the following elements at its core: a Cryptographic Key Management System; an Attestation Exchange Protocol, which defines how to request and present QEAA and PID; an Issuance Protocol, for PID and QEAA issuance; a Data Model, PID and QEAA Schemas, which defines the structure of both. It must also contain extra information such as verification mechanisms, Formats for PID and QEAA, Signature Formats for integrity, Cryptographic Suites and Mechanisms, Entity Identifiers, a Validity Status Check and a Trust Model that guarantees the legitimacy of all the moving pieces of this system.

Four flows are defined for this system, a close range supervised flow, as well as an unsupervised one, and then long distance flows, one for same device and the other for cross-device.

The first two flows happen in instances where the User is physically near the Relying Party and the request and attestation are done in close range using protocols such as

Bluetooth, NFC or QR-Codes. The flows differ in that this exchange can happen under the supervision of a human user or not: in the first flow, the user presents its attributes to a person, while in the second one they present their attributes to a machine.

The other two flows cover the cases where the exchange of attributes happens on the Internet: one where the User executes both authentication and the exchange within the EUDI Wallet device, while in the other flow case the EUDI Wallet device is used only for authentication to another machine, where the actual exchange happens.

Of note to these flows is that user authorization always takes precedence in these processes, and regarding the proximity flows, any combination of the user and the relying party being offline is possible.

2.9.5 Wallet Configurations

Wallet Configurations come up in the context of the EUDI Wallet's main goal being the synchronous development of a European wallet with as little limitations to interoperability as possible. With that in mind, the ARF has two main purposes in designing Wallet Configurations: linking EUDI Wallet capabilities to their respective use cases and creating a tool for potential extensions of the Wallet use cases.

To this end, they designed two configurations. Type 1 configuration is targeted at cases where the Relying Party requires LoA High and therefore is aimed for PID use cases. Type 2, on the other hand, is designed with flexibility in mind, attempting to cover use cases that the Type 1 configurations cannot solve. The ARF presents a table with every specific requirement for the configurations and distinguishes the configurations by the Type 1 obligation to follow the requirements (must) and the fact that most Type 2 requirements are guidelines (should). The requirements indicated are the ones listed in the second paragraph of the Reference Architecture and Flows subsections.

It is worth noticing that both Type 1 and Type 2 Configurations must support OID4VC Issuance as an Issuance Protocol, although this only applies to QEAA, and that EUDI Wallets must support Type 1 Configuration as they are mandatory for the support of PID.

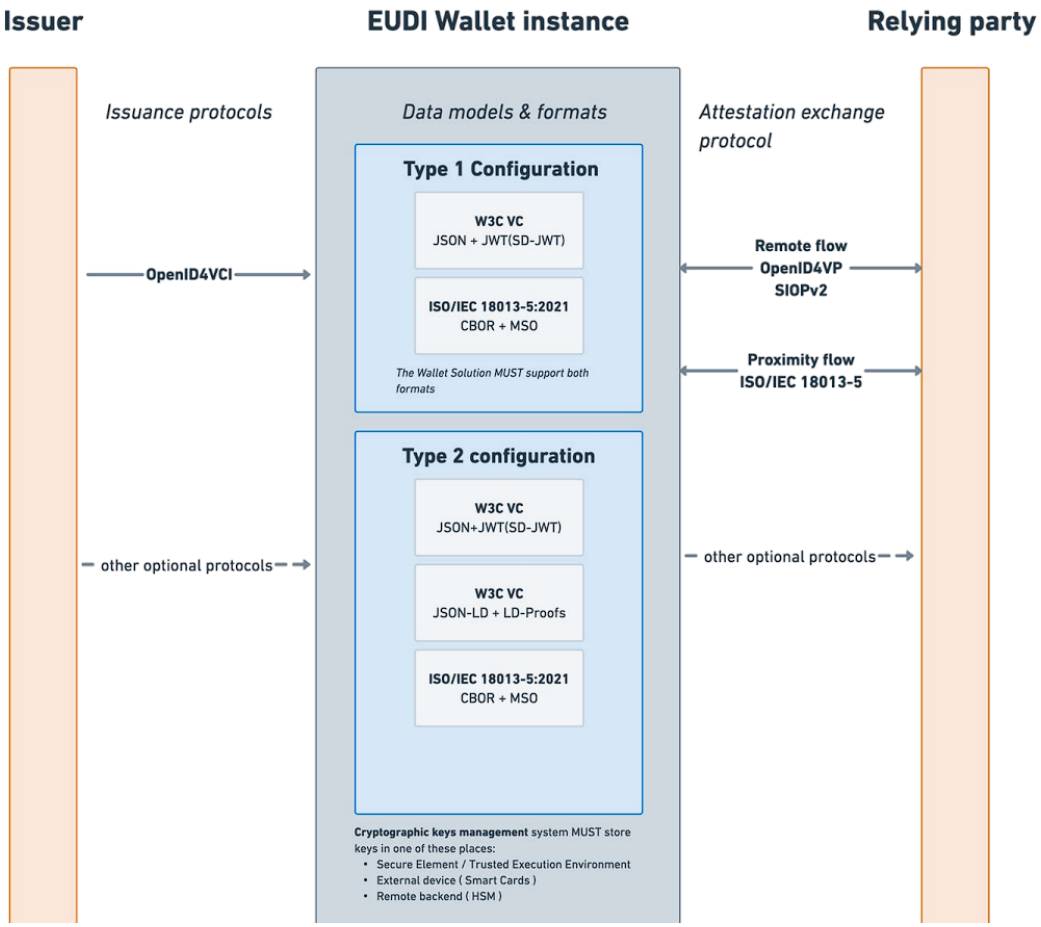


FIGURE 2.7: Visual representation of both configuration types, from the ARF article.[48]

Chapter 3

Development and Technologies Used

In the practical section of this dissertation's implementation phase, the primary objective was to design and develop a robust Self-Sovereign Identity (SSI) ecosystem encompassing the key components of Holder, Issuer and Verifier, to facilite the issuance and verification of credentials. This endeavour was executed using SSIKit, a library devised by the Walt-ID group. SSIKit was specifically engineered to manage cryptographic keys, Decentralized Identifiers (DIDs), and Verifiable Credentials, while providing an essential infrastructure encompassing Issuers, Holders, and Verifiers. This framework laid a solid foundation for the construction of a comprehensive SSI system, merely requiring the incorporation of an interactive Wallet interface tailored to the Holder's needs in order to achieve full functionality.

3.1 The walt.id Infrastructure

The walt.id team developed a mostly complete infrastructure for supporting an SSI ecosystem. To this end they developed an assortment of libraries ranging from basic SSI building blocks to secure storage and identity providers. The SSI Kit provides the basic building blocks for SSI, the management of DIDs and VCs, as well as the Holder, Issuer and Verifier who rest at the core of SSI. They have also developed other resources: a framework for building wallets, the Wallet Kit; a kit supporting NFTs, the NFT Kit; a kit for developing zero-trust storage and privacy preserving data sharing, the Storage Kit; and the IdP Kit which allows the launching of an OIDC compliant identity provider.

What follows is an explanation of the mechanisms behind each of those kits, some more in-depth than others depending on how relevant they were for the project's development.

3.1.1 SSI Kit

The SSI Kit represents a Kotlin-based library with the purpose of facilitating the management of cryptographic keys, Decentralized Identifiers (DIDs), and Verifiable Credentials (VCs). Its seamless integration is made available via Gradle/Maven dependencies or, alternatively, through a REST API. In the context of this project, the predominant development efforts were concentrated within the confines of the Wallet Kit, relegating the SSI Kit's utilization primarily to the realms of testing and demonstrative exhibitions, as well as functioning as a lower level dependency of the Wallet Kit. In this capacity, the REST API emerged as a fitting and adequate means for fulfilling the project's requirements.

The functionalities offered by the SSIKIT are the following, as listed by their developers:

1. **Key Management**: generation, as well as importing and exporting;
2. **DIDs**: creation, registration, updating and deactivation;
3. **VCs**: issuance, presentation and verification;
4. **European Blockchain Services Infrastructure (EBSI)**: the related use cases, such as onboarding and VC exchange.

For EBSI in particular, SSIKIT supports the onboarding of natural persons, including the generation and registration of DIDs, and the enabling of Trusted Issuers, which is the process of declaring an entity as a Trusted Issuer within the ecosystem. It also support the issuance of credentials by Trusted Issuers according to W3C standards, as well as the verification of these credentials.

3.1.2 Wallet Kit

The Wallet Kit provides a library and the infrastructure for a custom wallet solution's back-end, while also providing presets for a front-end, allowing a possible full solution on its own. The three main components of the back-end API are the wallet kit (the holder), the verifier portal (the relying party) and the issuer portal (the issuer).

The wallet kit is responsible for user management, although authorization is still not fully functional; user context switching is functional, as well as being responsible for data management by the user such as the listing of DIDs and credentials. It also supports the credential and presentation exchange protocol necessary for the system to function as a whole.

The verifier portal back-end is responsible for supporting its portion of the credential and presentation exchange protocol, that is, the verification of the presented data, but it also allows for the configuration of a list of supported wallets.

Finally, the issuer portal back-end is similar to the verifier because its main functionalities are supporting its part of the exchange protocol - in this case the issuing of credentials - while also supporting the creation of a list of supported wallets.

3.1.3 Storage Kit

The Storage Kit is written in Kotlin and follows the DIF specification for confidential storage, and its goal is to allow secure confidential data storage. It also allows easy interfacing with Encrypted Data Vaults, structures that provide interoperability without more information about the user being revealed than what is necessary. The system is separated into Storage Kit Server (responsible for providing the service), Storage Kit Client (the client) and Service Wrapper (the service).

3.1.4 IDP Kit

The IdP Kit is a library that allows the launching of an identity provider compliant with OIDC standards, fetching data from wallets as OIDC user info or mappable into OIDC claims.

3.2 EBSI Verifiable Credentials

The current landscape of technologies and components being developed in Europe in the Verifiable Credentials field are well documented on the EBSI website, and it is important to explain and list them for this work.

The EBSI VC/ VP models are built upon the W3C Verifiable Credentials, responsible for defining the standard way of presenting VCs on the Internet nowadays. EBSI has defined many different VC data models due to the vast amount of use cases that must be

considered, and naturally many more are bound to come up. Depending on the use case and the required signature type, VCs normally are JWT or JSON-LD.

The W3C VC lifecycle was as simple as its issuing, storing the credential in a repository, such as a wallet, and then composing credentials into a presentation, which is then provided. In contrast, the EBSI VC lifecycle starts with the onboarding of the participants in the system (legal entities as Issuers and legal persons as Holders), after which credentials are issued and stored for both parties, followed by the actual composing and sharing of presentations. In addition, the EBSI ecosystem also features TSRs, Trusted Schema Registries, which manage the accepted schema data models.

3.2.1 E-Signatures of VCs and VPs

The standards for electronic signatures defined by eIDAS are the basis of trust in the EBSI ecosystem since they are the cryptographic primitives that guarantee the secure communication between the participating parties.

EBSI currently supports two signature types for their VPs: the JWS (JSON Web Signature) models that must support the ES256 algorithm, and the JAdES model which must support eSeals. The supported digital signature schemes are ECDSA and RSASSA.

A JWS VC consists of a header, a payload and a signature, and must support ES256 signatures. The header must contain the issuer's DID, the signing algorithm and a type indicating it is a JWS, and conditionally the JWK public key to generate the DID document. For the payload, the timestamp fields are mandatory, as well as the issuer's, the recipient's and the claim's identifiers, followed by the claim itself. The expiration date field is optional.

For a JWS VP, the header's requirements are the same as a VC's, and the payload's requirements are similar, but also have an audience field which must specify the DIDs or URIs of the intended audiences of the presentation, as well as a nonce to prevent replay attacks. Instead of a VC, the payload has a VP field.

The eIDAS regulation specified standards required for electronic signatures in order to secure online business in the European Single Market (ESM). For this reason, the advanced electronic signature (AdES) type was developed, relying on PKI, and legal entities must use it to sign documents meant for the ESM. JAdES is a specialization of AdES for JSON data.

JAdES is built on top of JWS and simply adds additional fields to the header and allows for four different signatures, each incrementally increasing the security of the message. The new fields introduced represent the certificate thumbprint, its chain, a timestamp of the signing process and the names of all the signed header parameters.

The process of signing a VC or VP begins with the preparation of the VC/ VP itself, i.e. the actual content that goes into the VC/ VP field of the JWS/ JAdES payload. After that, for a JWS signature any JWS signing library can be used, while for a JAdES signature the DSS should be used, since it was designed with the purpose of supporting eIDAS and European digital signatures.

On the other hand, the verification of a signature starts by checking which type of signature is being used on the header of the presentation. For JWS, the verifier must first check if the presentation is about a legal entity or a natural person. If it is about a legal entity, the verifier will resolve the DID and the public key; if it is a natural person, they will create a relationship DID from their identifier and the given public key. After this, verifying the signature only requires a JWS signature library. For a JAdES signature, DSS should be used again.

3.2.2 DID Method

DIDs occupy a pivotal role within the operational framework of an SSI ecosystem and, commensurately, the DID documents housing pertinent information about them are equally significant. Their primary relevance pertains to Legal Entities, which require secure retention within a specialized DID registry — a critical component of the DPKI within the broader SSI system. This registry serves as a pivotal interface enabling DID owners to perform essential functions such as registration, revocation, and updates of their DIDs.

Within the context of the European Blockchain Services Infrastructure (EBSI), DID Methods are carefully crafted to cater to both Legal Entities and Natural Persons. For Legal Entities, these methods yield globally unique identifiers, rigorously validated by the DID Registry. In contrast, for Natural Persons, the methods engender pseudonymous identifiers, ensuring privacy and discretion in identity management. EBSI defines a DID as:

did:ebsi[**network**]:**method-specific-identifier**

where network is omitted for the EBSI blockchain and method-specific-identifier is a unique identifier for legal entities and for natural persons. There must be a distinction between both because DIDs for natural persons are governed by the GDPR.

Only legal entities can register DID Documents on the EBSI Ledger, since the ones for natural persons are derived from signed JWTs. EBSI's DID Documents are compliant with the W3C definition but also define the fields for the verification method, its key, and the assertion method as required instead of optional.

For natural persons, their public key information is hashed into the DID Method as a JWK thumbprint. Therefore, to allow the validation of a natural person's DID they must use the JWK field of the JWT header to carry public key information. The assertion of a received DID is done by transforming the received key materials into a natural person DID and comparing it to what was received.

3.2.3 Trust Models

In an SSI ecosystem, trust is of utmost importance: the Holder must trust both the Verifier and the Issuer, Issuers must trust Wallet Providers and Verifiers, and Verifiers must trust Issuers and Wallet Providers. The EBSI framework allows the flexible and scalable mapping of the environment's trust relationships, thus making the process of exchanging credentials safer and smoother.

Holders and Verifiers must be able to verify an Issuer's identity, to check if their credentials grant them the right to issue credentials and to see if these credentials have not expired or been revoked. Holders should also be able to verify the Verifier's identity and check under which policies they operate.

Legal entities must be accredited before they can act as Trusted Issuers (TI) and start issuing VCs. This is done by having a Trust Accreditation Issuer issue them an accreditation, or having the EBSI Support allow their self-declaration of trust, and multiple accreditations can be issued to a legal entity. The entities that are relevant to the accreditation process, and their details, are the following:

1. **EBSI Use Case Authorization Issuer**: authorizes legal entities to self-declare as Trusted Accreditation Issuers; this authorization must be approved by EBSI;

2. **Root Trusted Accreditation Organization**: (Root TAO) allowed to self-accredit, and can issue Verifiable Accreditations of their own accredited types for legal entities. Since it can self-accredit, it is not limited regarding what type of Verifiable Accreditations it can issue, which allows the creation of a hierarchy;

3. **Trusted Accreditation Organization**: similar to the Root TAO but cannot self-accredit, so it is limited in the types of Verifiable Accreditations it can issue; their primary goal is to issue accreditations to legal entities and extend the trust chain by accrediting other legal entities to make TAOs for them;

4. **Trusted Issuer**: can issue Verifiable Attestations they are accredited to attest for natural persons and legal entities; they are the leaf level of the trust chain tree.

Accreditations certify an entity's ability to accredit, attest and authorize, and are individually limited to specific use cases. When issued, they can limit the ability of the recipient to accredit, attest or authorize. Authorizations are allowances for self-declare and are also governed by use-case limitations. A legal entity that self-declares is promoted to a Trusted Accreditation Issuer (TAI) without a trust anchor on top, which leads to the creation of a new trust chain.

All of the above parties, except for the EBSI Use Case Authorization Issuer, must register their DIDs as well as their Verifiable Accreditations in the Trusted Issuer Registry, their DID Document in the DID Registry and the Accreditation types in the Trusted Schema Registry.

The process of onboarding a legal entity into the ecosystem is as follows:

1. The legal entity creates its own DID and DID Document;

2. It then requests a Verifiable Authorization from the TAI, which should issue it after validating and identifying the legal entity;

3. The legal entity exchanges the Verifiable Authorization for an access token at the EBSI Authorization;

4. With the access token, the legal entity registers its DID and DID Document at the relevant registries.

When it comes to accrediting a top-level legal entity, the process is different.

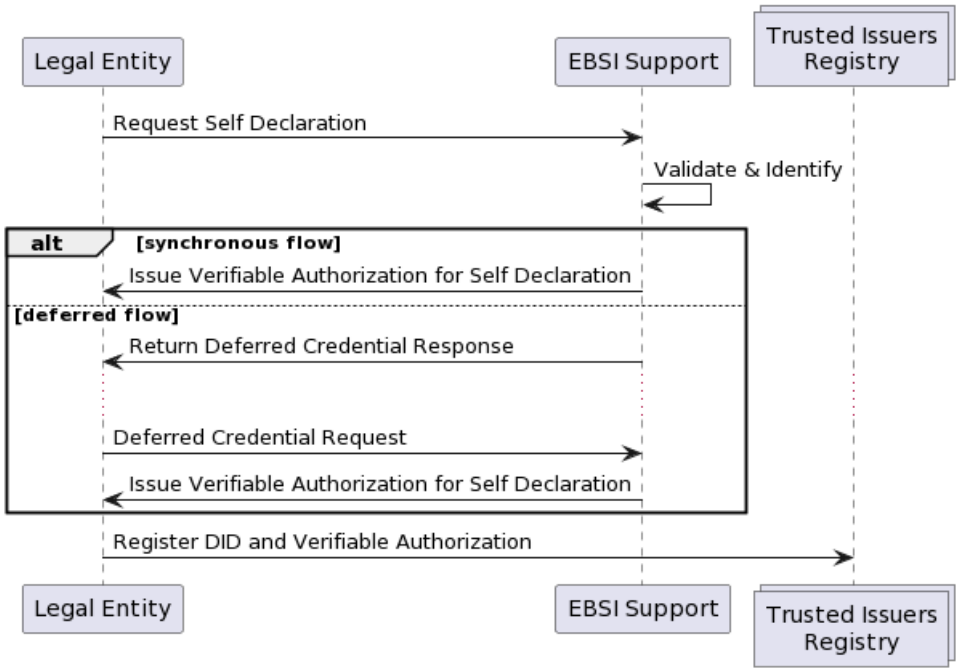


FIGURE 3.1: The flow of accrediting a top level entity. [38]

Lower level legal entities that wish to be accredited follow an almost identical process to the one described in the above flow, but instead of requesting a Self-Declaration they request a Verifiable Accreditation, and instead of making the request to the EBSI Support, they present it to a Trusted Accreditation Issuer.

Since all necessary information to trace back accreditations and attestations can be found within the respective registries, it is easy to check if everything is in order and if entities are accredited to do what they claim to be able to do.

The system for verifiers is near identical, with the possibility of having self-hosted identity and policies.

3.2.4 OpenID4VC

OID4VC defines protocols and standards for the issuing of Verifiable Credentials. There are two key participants in the protocol, the Authorization Server and the Credential Issuer. The first does authentication and authorization for the second, while the Credential Issuer acts as an OAuth 2.0 protected endpoint that issues credentials. The Verifiable Credential Issuance (VCI) process begins with redirects or QR codes, it cannot be called directly through the Internet, and it can be started by the Issuer or the End User themselves.

The flow starts when the End-User opens their wallet with the intention of acquiring a new VC, for example a Proof of Residence. They find the relevant Issuer - in this case it would be a government-controlled Issuer - and apply for a Proof of Residence. The Issuer requests that the user log in to their government account, and also asks for the user's permission to create a Proof of Residence about them. After both requirements are met, the credential is created and sent to the user's wallet. The scenario where the Issuer begins is almost identical apart from the start: instead of having the user applying for a Proof of Residence at the Issuer, they find a link at the Issuer's page prompting the creation of a Proof of Residence.

User authentication can be achieved by any of these three methods, or any combination of them: logging in to the Issuer's authentication service, proving control over a DID that already has a connection established with the Issuer, or presenting a Credential that serves as authentication before the Issuer and is recognized by it.

Credential Offering is a service presented by some Issuers where an OpenID Credential Offer endpoint is setup by Issuers. The Wallet endpoint starts the communication flow with the Credential Offer endpoint and, with the user's consent, receives the offered credential, depending on how trustworthy the issuer is. The Credential Offer can be passed on as a value, or as a uri to be resolved.

As mentioned before, the Authorization Server is responsible for doing authentication and authorization on behalf of the Issuer. It is able to request user ID tokens, exchange data over VPs and issue access tokens. The call for an authorization request is issued by the user, and must carry with it the detail of what VCs the user is requesting. It is built upon the OAuth 2.0 Rich Authorization Request, a message type designed to carry fine-grained authorization data. The user receives an authorization code after a successful exchange.

In the authentication process, the Server sends an ID Token request to the user, who signs it, proving ownership of a DID in doing so, and sends it back, completing the authentication process. After this, a Token Request exchange can be done, where a user exchanges an authorization token for an access token and an id token, which allow them to proceed with Credential Issuance operations.

The Credential Issuer has two endpoints, the Credentials Endpoint and the Deferred Credentials Endpoint, the latter being used for asynchronous operations. A credential request that is sent to the Credentials Endpoint must request the same credentials that

were specified in the original Authorization Request and must carry with it proof that the request is bound to a DID. The response can be real-time or asynchronous (deferred): the real-time response carries the credential data while the deferred one carries an acceptance token, usable later to claim the respective credentials.

3.2.5 Wallets

Since Wallets are the main way users will interact with the EBSI ecosystem, and given the fact that they will not be developed and provided by EBSI itself, it is necessary to make sure wallet providers comply with the EBSI framework and standards, using the EBSI Wallet Conformance Testing (WCT). The WCT provides a set of tests for wallets that allow providers to enhance their wallets to fit the needs of the EBSI ecosystem. Overall, it can: grant a stamp of approval, increasing the wallet's credibility; prove it is capable of seamless interoperability within the EBSI systems; and boost the wallet's visibility in the competitive wallet marketplace. It provides tests for Holder wallets, for evaluating the compliance with the Issuer Trust Model, as well as the accuracy of the information being stored in the Trust Registries. It also checks if a wallet is functional and compliant with the credential issuance model, and finally it checks if it is capable of fitting into a VP validation and verification scenario.

3.3 Use Cases

The uses of this technology are varied and allow for much innovation and flexibility, but in the context of this work, the more relevant ones are those that fit into the context of the European Union, whether across countries or within each individual nation.

3.3.1 E-Prescriptions

A use case that is both realistic and problematic arises when a person under a strict medical regimen needs or wants to travel within the European Union. The system in place as of the time of writing this paper is already somewhat aligned with the goal discussed here, and allows for any prescription written within a member country of this framework to be used across borders, as long as it is within the EU (it includes 25 member states of the EU). The countries who are not yet involved in this framework may or may not have systems in place to allow for cross-country e-prescription interoperability, as there

is no single standard in place for it. If you are planning on travelling to a country that does not support your country's e-prescriptions, you need to ask your doctor to make sure the physical prescription has a specific set of necessary information for cross-border EU verification. The burden of having to carry the physical article becomes a part of the equation, the recipient of the prescription must carry it with them wherever they travel, which can be a bother. This can become especially burdensome when there are multiple prescriptions to take care of, especially with each having their own expiration date and method of renewing.

The sequence of events an individual would go through in the traditional system is: going to the doctor and receiving the prescription, checking if it contains all necessary information for cross-country validation, and carrying it with them on the trip, and presenting it on a pharmacy or hospital - something which may still involve some complications.

The flow of events using the EUDI digital wallet would have the prescription (in this case e-prescription) being sent to their wallet by their doctor. It would come in a standardized format that they could present to the pharmacist in another member country who would be able to verify its authenticity. A system similar to this one is already in place among 25 members of the EU.

3.3.2 Bank Account

Creating a bank account can be a very daunting and time-consuming process. Someone wanting to open a bank account needs to spend a good amount of time at the bank for the process. It would also be necessary to present many documents, for example, in Portugal you need an identification document, such as a citizenship card, as well as a proof of address and a proof of employment, which all together create a slow and tiresome process.

The EUDI Wallet would allow this process to be much faster and streamlined since all the necessary documents would be present within the user's personal wallet and all they would have to do was pass them on to the bank with a verifiable presentation. After verification, the bank would allow the account's creation. It would also make the process of accessing the bank account easier as it would be directly connected to the user's wallet.

3.3.3 Education

Obtaining an educational diploma certificate, for example a Master's Degree Diploma, is a process that is greatly improved by the implementation of an SSI system. The use of

the certificate itself also becomes more portable and seamless with the EUDI and EBSI efforts. Obtaining a certificate becomes as easy as going to the university's Issuer and requesting a certificate Credential, authenticating by logging into the student's university account. Presenting the certificate while applying for a job, for example, also becomes a simple process: generate a Verifiable Presentation that includes the certificate along with the other data requested by the potential employer and present it to them. They can then verify it by checking the Trusted Issuer Registry, as well as checking if the Certificate is indeed aimed at the user's DID.

3.3.4 Miscellaneous Cases

Many documents that used to be carried in a person's wallet or car would become digital, such as a driver's licence or car related documents. An European Digital identity could also be used to allow a person to purchase a SIM card from a local provider when travelling abroad without any problems.

3.4 Schemas

EBSI defines multiple JSON Schemas to fit the multiple possible use cases for the ecosystem, while also having at its core many others defined for the systems it was built on. Use case Schemas are a constantly developing field, together with their respective use cases, but the core Schemas that allow the baseline functioning of the system are also of great importance.

Of the W3C standards, EBSI inherits the Verifiable Accreditation and the Verifiable Presentation Schemas, while also having Accredited Verifiable Attestation, Verifiable Authorization and Verifiable Authentication Schemas as its own trust anchoring Schemas. Other Schemas are considered use-case but are crucial to the functioning of the system, such as Natural Person and Legal Entity Verifiable ID Schemas. Verifiable IDs are a special type of Verifiable Credential within the EBSI system that can be presented as evidence that owner is who they claim to be, whether they are a Legal Entity or a Natural Person.

3.4.1 Natural Person Verifiable ID

A verifiable ID of a Natural Person carries the eIDAS minimum information data set and includes the following fields:

Mandatory:

1. id: the DID of the subject;
2. first and familyName: names of the subject;
3. dateOfBirth: date of birth of the subject;
4. personalIdentifier: the personal identifier used within the Member State the subject belongs to;

Optional:

1. nameAndFamilyNameAtBirth: subject's name and family name at birth;
2. placeOfBirth: subject's place of birth;
3. currentAddress: subject's current address;
4. gender: subject's gender.

3.4.2 Legal Entity Verifiable ID

A verifiable ID of a Legal Entity carries the eIDAS minimum information data set and includes the following fields:

Mandatory:

1. id: the DID of the subject;
2. legalName: subject's legal name;
3. domainName: subject's domain name;

Optional:

1. legalPersonalIdentifier: subject's national identifier;
2. legalAddress: subject's official legal address;
3. VATRegistration: subject's VAT number;
4. taxReference: subject's tax reference number;
5. LEI: subject's official Legal Entity Identifier;

6. SEED: subject's System for Exchange of Excise Data;
7. EORI: subject's Economic Operator Registration and Identification;
8. SIC: subject's Standard Industrial Classification;

3.4.3 Verifiable Accreditation

Verifiable Accreditations, which are issued for TAOs, TIs and EBSI Onboarding Services, include the following fields:

1. credentialSubject: an object that defines additional information about the subject of the Accreditation; the only field in it is the id of the subject, that is, the DID, in this case representing the organization whose activities are being accredited;
2. id: the id of the Accreditation;
3. authorisationClaims: the list of claims that define the permissions the subject has for issuing different VC types;
4. authorisedSchemaId: uri to a Verified Accreditation Schema on the Trusted Schema Registry; it lists all schemas whose respective VCs the subject of the Accreditation is allowed to issue (while also being allowed to issue those extending the listed ones);
5. limitJurisdiction: uri to the jurisdiction within which the Accreditation is valid;

The registration of a Verifiable Accreditation in the Trusted Schema Registry involves a previous verification of its validity, since its stay is permanent in the blockchain. The first step of the verification of an Accreditation is validating its JSON Schema formatting, i.e. ensuring it is properly formatted and in accordance with the standards. After this has been proven to be true, it is necessary to validate the entities within the accreditation: checking that the issuer is registered in the TIR and that the credentialSubject is a valid EBSI DID; that it is in the TIR, and that it has a Legal Entity associated with it in the TIR (i.e. that it has a valid Legal Entity Verifiable ID as an attribute). After this, the Verifiable Accreditation data is checked, which involves checking everything related with the issuer and their capability to issue the level of accreditation being issued in the Verifiable Accreditation. Finally, the status of the Accreditation is checked together with the domain and business specific data.

3.4.4 Verifiable Presentation

The fields within a VP are:

1. context: the W3C Specification;
2. id: optional, it is present to give the VP a unique identifier;
3. type: defines the type of what is in the JSON Schema, in this case a Verifiable Presentation;
4. holder: uri to the person who is generating the presentation;
5. credential: an array of Verifiable Credentials;
6. proof: optional, its own object, with a type, a purpose field, a timestamp, an indication of the verification method, an optional challenge (similar to a nonce), an optional domain defining the operational domain of the proof, and the proof value in a JWS format.

3.4.5 Verifiable Attestation

Verifiable Attestations are a branch of Verifiable Accreditations and include the following fields:

1. context: defines the semantic context of the Attestation;
2. id: unique identifier of the Attestation;
3. type: as in VPs, defines the VC type, in this case it must have "VerifiableCredential", "AccreditedVerifiableAttestation" and "VerifiableID";
4. issuer: DID of the issuer of the Attestation;
5. issued, validFrom and validUntil: timestamp and validity data, respectively, validFrom and validUntil being the time when the Attestation becomes valid and invalid;
6. credentialSubject: the same object field as in Verifiable Accreditations, but with the subject of the Attestation in this case;
7. credentialStatus: an object containing information allowing the discovery of the current status of the Attestation, i.e. a uri to a record of the validity of the Attestation and a status field type declaring its status type;

8. credentialSchema: an object containing information about the Schema the Attestation was based on, i.e. an id field which is a uri to the Schema template and a type field declaring the Schema type;

9. evidence: array of information on the events that led to the issuance of the Attestation, i.e. the type of evidence and an id field which is a uri to the evidence;

10. proof: object carrying proof information (mostly identical to VP proof, without domain or challenge);

11. termsOfUse: the terms of use under which the Attestation was issued; as with the previous cases, it features an id field which is a uri to the terms, and a type field for the type of terms.

Every field, except for the validUntil, credentialStatus, evidence, proof and termsOfUse fields, is mandatory in normal Verifiable Attestations, while for Accredited Verifiable Attestations termsOfUse is also mandatory.

The process of verifying an Accredited Verified Attestation is a bit different from a normal Verifiable Accreditation verification due to the termsOfUse field being active. In addition to the standard Accreditation verification procedure, the uri of the Issuer is fetched from the termsOfUse and then a top-down verification of its accreditations is done.

3.4.6 Verifiable Authorization

A Verifiable Authorization is a Verifiable Attestation that has the DID id of the subject of the authorization as a mandatory field in the credentialSubject object of the Attestation.

Chapter 4

Code and Practical Explanation

This dissertation's practical objective is to establish an SSI ecosystem which features the core of Holder, Issuer and Verifier, ideally with the structures that support them, such as a the necessary registries, and Trusted Accreditation Organizations, as well as having multiple Wallet Provider solutions available. More specifically, the planned structure is to have at least two entities representing teaching institutions, another representing a government institution and another representing a company and two user use cases.

The first use case represents a student who has just finished a course, for example, a bachelor's degree, and wants to get a Verifiable Diploma so they can apply for a job in the area. In this use case, the student is the Holder, while the university they graduated from acts as the Issuer, and the company they are applying for acts as the Verifier, with the Verifiable Diploma being the credential which is "passed around".

The flow will have the student request the Verifiable Diploma from the University they graduated from, University A in the diagram, which will generate the Verifiable Diploma in the student's name and store it in the Claims Registry. Then it will pass a reference towards it in the Claims Registry to the student, who can now share it with others. The student will then apply for a job at the company, which will request the Verifiable Diploma from them. When the Diploma is presented, the Company will resolve it in the Claims Registry, and after checking if its information is what is necessary, it will resolve the Issuer's DID in the Trusted DID registry, to know if the Verifiable Diploma was issued by a trustworthy entity. When all this is validated, the job application is accepted.

The second use case also features a university alumni, a doctorate for example, who, after going through the process of claiming a Verifiable Diploma from the university from which he graduated, wants to apply for a teaching job in another university. This job

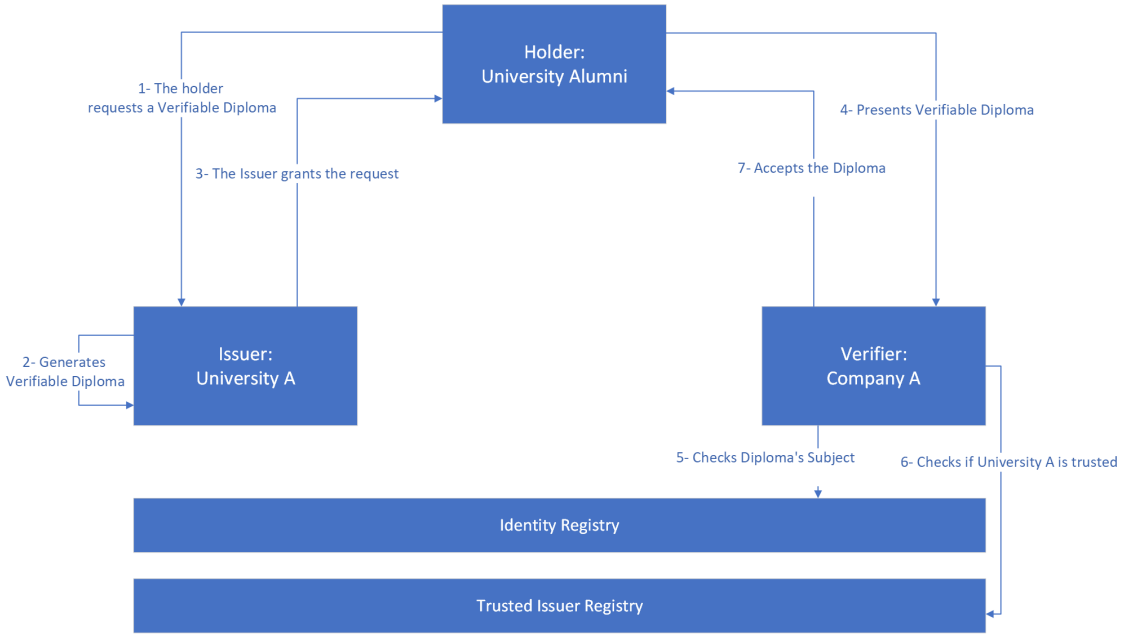


FIGURE 4.1: Flow of use case 1.

requires not only a Verifiable Diploma, but also a Proof of Residence. In this case, we will have two entities acting as Issuers: University A where the alumni graduated, and a government organization responsible for issuing Proofs of Residence. In addition to this, the alumni will be the Holder, while University B, where the alumni is applying for a job, is the Verifier.

The first block of the second use case mimics the first's first half, with the alumni requesting and being issued a Verifiable Diploma, and is followed by the same process. But instead of requesting a Verifiable Diploma from University A, the alumni requests a Proof of Residence from the Government Organization. After this, the alumni applies for the teaching job at University B, which will prompt them to present the two Credentials they have obtained. Following this, the alumni composes a Verifiable Presentation with the Verifiable Diploma and the Proof of Residence and presents it to University B. This university will "unpack" the VP and will follow the same protocol as in the first use case, resolving both credentials, and then resolving both Issuers, after which it will accept the application if they are both valid.

Some of these goals were achieved and some could only be explored in theory. The second included the existence of a Trusted Accreditation Organization and the Trusted Issuer Registry, as well as the possibility of using multiple wallets. This latter case was implemented in waltid but was not explored due to time constraints. The system that was

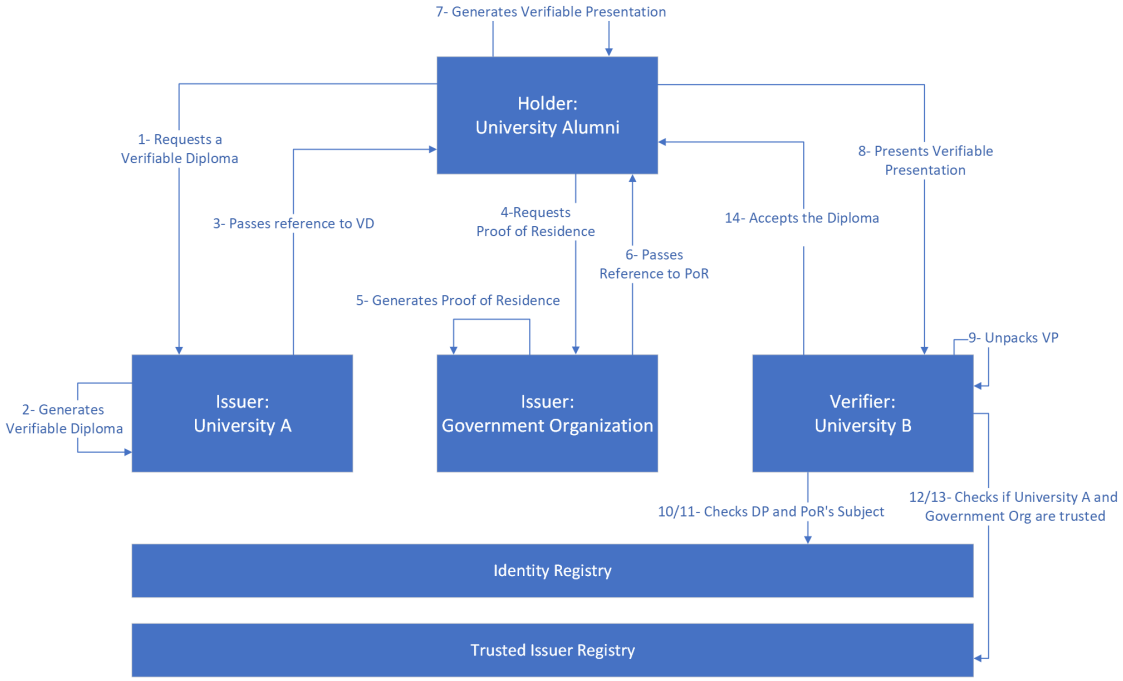


FIGURE 4.2: Flow of use case 2.

achieved in the end has a singular wallet type available, which works under presumed trust between Verifiers and Issuers. In the first use case, the Company implicitly trusts University A, and in the second case, University B implicitly trusts both University A and the Government Organization. Similarly, the Identity Registry could not be implemented on time, but it is not as important in this case since the PKI nature of the ecosystem allows the Holder to prove they are who they claim to be with a signature using a private key.

4.1 Trusted Issuer Registry and Trusted Accreditation Organizations

In the theoretical scenario there would be a third party to the ecosystem, a Trusted Accreditation Organization or above which would have issued a Verifiable Accreditation to all relevant Issuers, defining each respective Credential type they were allowed to issue, and that would have stored them in a Trusted Issuer Registry. The TAO itself would have had either a Verifiable Accreditation in its name already in the Registry, or the system would need the other parties relevant to the Accreditation process - a Root TAO and the EBSI Use Case Authorization Issuer - to create a chain of trust. The original plan was to merely have a Verifiable Accreditation referring to the first TAO within the Trusted Issuer

Registry. It would be issued by an EBSI Use Case Authorization Issuer that does not actually exist (as there is no need to issue further Verifiable Accreditation in this case), but is implicitly trusted by every participant in the ecosystem.

This means that the Trusted Issuer Registry does not exist in none of the use cases, and step 7 in the first use case and steps 14 and 15 in the second do not actually happen in practice. In theory, there should exist an authorized ledger holding the Verifiable Accreditations, which would have the EBSI Use Case Authorization Issuer as an entity with the ability to add to the ledger at will. It would also have further TAOs and Root TAOs be able to add Verifiable Accreditations to the ledger according to the trust chain established so far.

The ledger would be transparent to anyone since resolving Issuer DIDs is a basic requirement of the SSI ecosystem. DID resolution would probably have been done using the EBSI DID Resolver, and the lookup portion of steps 7 of the first use case and 14 and 15 of the second would be done through this mechanism. The DID validation would be done by recursive lookups to each Verifiable Accreditation's issuer until it reaches a TAO it inherently trusts. This would be hard-coded but, to keep realistic functionality, it would either stop at any Root TAO or at the EBSI Use Case Authorization Issuer, at which point the Issuer's trusted status would be validated.

4.2 Trusted Wallet Solutions

As for the other portion that could not be implemented, which is the Wallet variety, the necessary elements for its operation would be multiple wallet solutions, as well as the implementation of a list of trusted wallet solutions. These could either be a decentralized authorised registry or, in a realistic setting, a public listing verified and signed by EBSI itself. Wallets are evaluated regarding compliance with EBSI Compliance Tests on the EBSI website and the listing of allowed wallets would aim to emulate the existing list of wallets that have passed such a test. Updating of this list would have to be done manually, as the real system relies on an online test. Looking up the wallet specifications would be similar to the lookup of Verifiable Accreditations, but without the recursion because the wallet specifications would be issued directly by the wallet solution and verified by EBSI.

Having access to the specifications of the wallet solution, the interaction with a holder's wallet would follow the usual exchange protocol described throughout this document.

4.3 SSI Kit Implementation

The SSI Kit is the waltid underlying implementation of the Holder, Issuer and Verifier core, as well as of the VC/ VP exchange protocols. It provides a Signatory API for Issuers, a Custodian API for Holders and an Auditor API for Verifiers.

It is implemented in Kotlin and comprises a RESTful web service and a CLI tool, while being modular and composable, therefore allowing customization and functionality extension with other systems. It also allows the seamless addition and implementation of extra use-cases to the system.

The services provided by the SSI Kit are all pointed at SSI functionalities. It provides tech-agnostic registry operations, such as reading and writing, as well as key management operations, i.e. generating, signing, importing, exporting and lifecycle related operations. It also provides operations for DID management, i.e. creation, resolution and lifecycle operations, and for VC/ VP operations, i.e. creating, issuing, presenting and verifying. Finally, it also supports some ecosystem specific operations, such as onboarding.

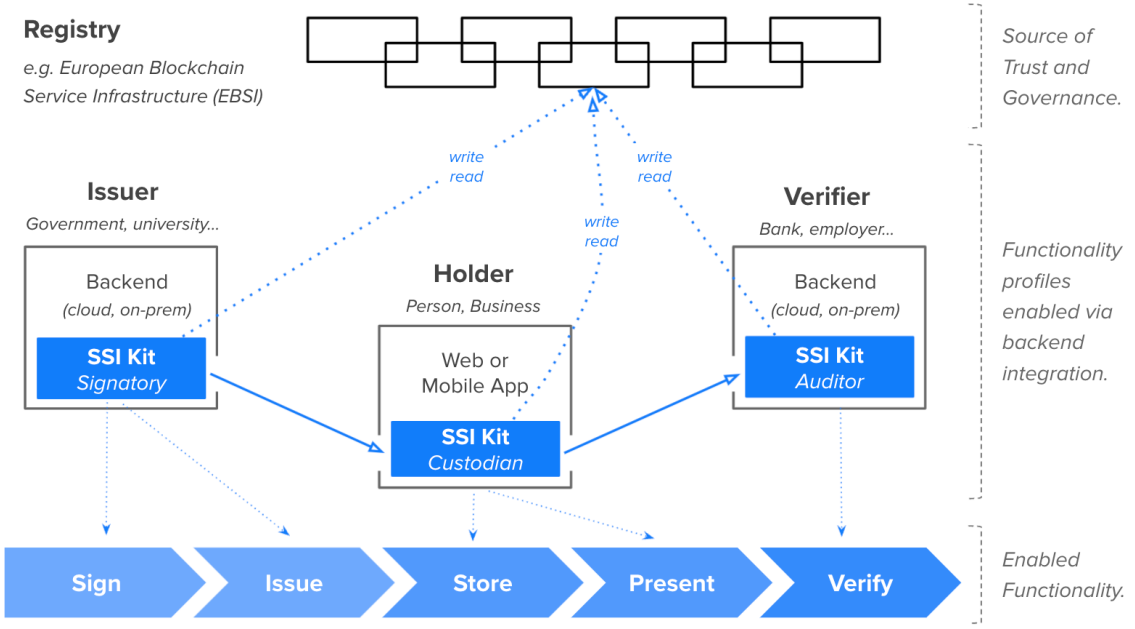


FIGURE 4.3: SSI Kit Architecture, by waltid. [49]

Below are indicated the underlying technologies of each key element of the SSI Kit. The Trust Registries accepted by the SSI Kit are:

1. Permissioned and permissionless Blockchains;

2. Domain Name Services (DNS);

3. purely Peer to Peer approaches.

The cryptographic keys supported by the SSI Kit are:

1. EDDSA / ed25519
2. ECDSA / secp256k1
3. ECDSA / secp256r1
4. RSA

The supported DID methods are ebsi, web, key, jwk, iota and checkd, while the supported Verifiable Credential formats are JSON / JWT and JSON-LD. The supported data exchange protocols are OpenID for Credential Issuance and OpenID for Verifiable Presentations.

The SSI Kit architecture has three layers: the Low-Level Services Abstraction layer, the Ecosystem Abstraction layer and the High Level Interfaces layer.

The Low-Level Services Abstraction layer holds core elements to the functioning of the SSI system, such as key interfaces with abstract key and signature operations. It supports extensions to HSMs and WebKMS for storage, DID interfaces - responsible for abstracting DID and DID document operations - and Credential interfaces, which take care of abstracting VC and VP operations. The Ecosystem Abstraction layer allows for the use of several different ecosystems, with EBSI being the most important in this context. The last layer, High Level Interfaces, is there to make the complexity introduced by the low level services and the multiple ecosystems more accessible to developers. It grants access to operations to issue, hold and verify credentials.

4.4 WaltId Implementation

The implementation of the waltid ecosystem has four main parts: the Issuer Portal, the Verifier Portal, a Wallet solution and a web wallet kit. This wallet implementation is a custodial wallet, which means keys and data are ultimately stored by the wallet provider. It allows for the addition of other wallets, which was the goal of this work but this was not achieved due to time constraints. The wallets that would have been tested in this environment would be a selection from the compliant wallet list from EBSI - Veloxsoft or DS Wallet, for example.

The Wallet kit provides the following functionalities: a user-friendly web interface for the wallet, user context separation for key, DID and credential storing, user data management for DIDs and credentials, EBSI DID ecosystem integration, as well as VC and VP exchange protocols using OpenID4VP.

The architecture of the Wallet kit ecosystem has the Wallet, Verifier and Issuer backends at its core. These are the components responsible for the source code of those entities and each is managed by a Context Manager, which also operates the SSI Kit. Each backend is connected to high-end APIs which in turn connect to the frontends, the Web Wallet and the Issuer and Verifier Portals.

Due to difficulties with the practical section, the Claims Registry could not to be implemented in the correct decentralized manner that was described in the theory section.

The deployment used for tests was done entirely locally, using ports 8080 and up, scaling up as necessary.

4.4.1 Issuer Code

The following code shows the class for a Credential on the Issuer's side. A Credential is a simple pair of a String dictating its type along with a Map of JSON schema elements and their respective values. Its initiating function fromTemplateId simply picks up an identifier and fetches the corresponding Verifiable Credential Schema. The Issuables class represents a list of credentials and its initiating function receives a list of details about the credentials to be issued (both the type and the field values). It also maps the field values according to the respective fields, after calling the individual credential initiating function.

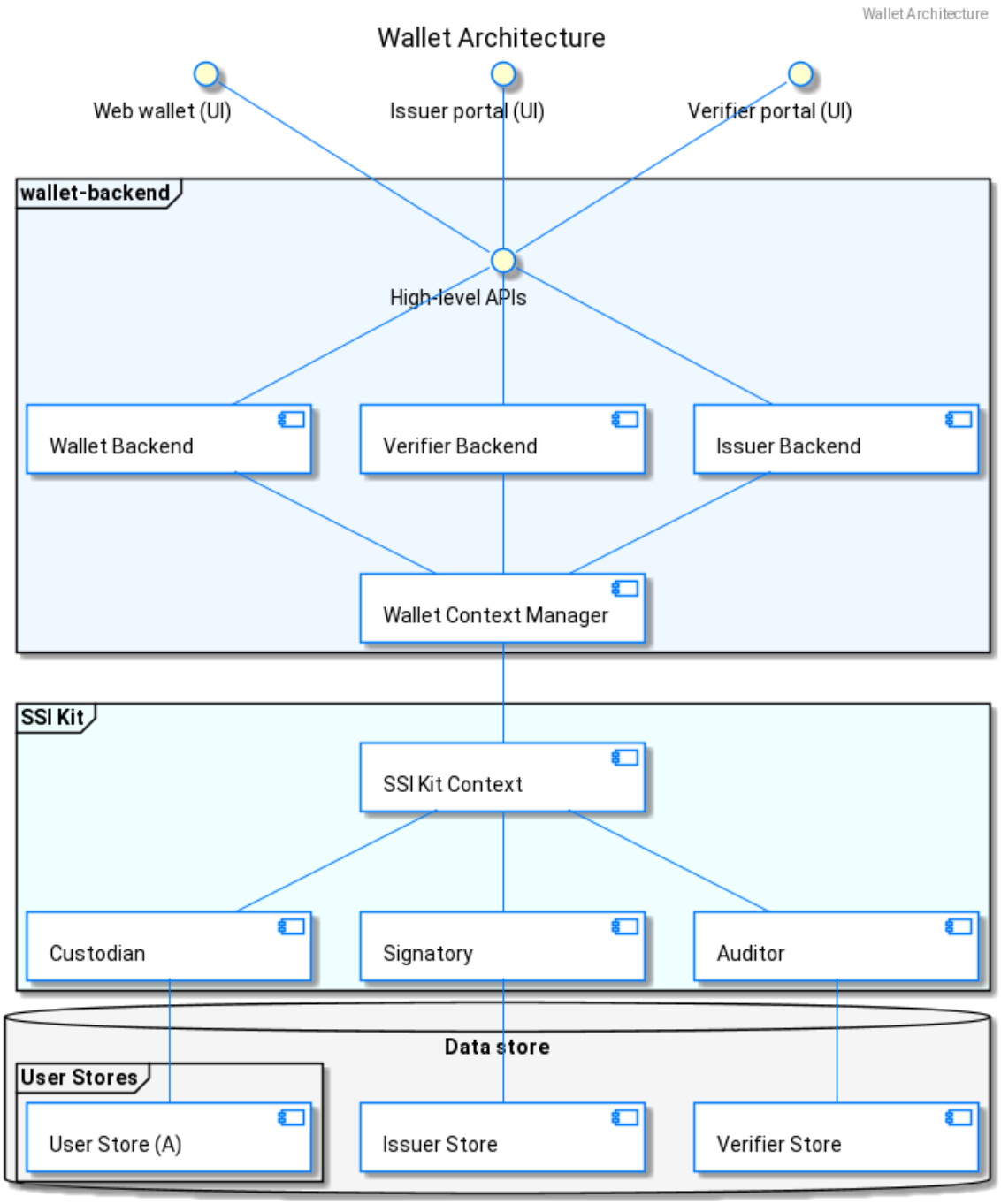


FIGURE 4.4: Wallet Kit Architecture, by waltid. [50]

```
data class IssuableCredential(
val type: String,
val credentialData: Map<String, Any>? = null)
{
companion object {
    fun fromTemplateId(templateId: String): IssuableCredential {
        val tmpl = VcTemplateManager.getTemplate(templateId, true).template!!
        return IssuableCredential(
            templateId,
            mapOf(
                Pair(
                    "credentialSubject",
                    JsonConverter.fromJsonElement(tmpl.credentialSubject!!
                    .toJsonObject()) as Map<*, *>
                )
            )
        )
    }
}
data class Issuables(
    val credentials: List<IssuableCredential>) {
    companion object {
        fun fromCredentialAuthorizationDetails(credentialDetails:
        List<CredentialAuthorizationDetails>): Issuables {
            return Issuables(
                credentials = credentialDetails.map { IssuableCredential
                .fromTemplateId(it.credential_type) }
            )
        }
    }
}
```

The rest of the Issuer-side process is as it would be expected for this type of exchange: the Issuer Portal webpage has a form for the user to choose the desired Verifiable Claim(s) type and, depending on the scenario, the field values will be filled in by the Issuer or the requesting user. This form is then passed in the format of a List of CredentialAuthorizationDetails which is used to create the Issuables data type, which is then cryptographically signed and sent to the user. If the Claims Registry and the Trusted Issuer Registry were in place, the Claim would instead be stored in the Claims Registry. The Issuer's authority to do it would be verified in the Trusted Issuer Registry and a DID reference pointing towards it would be sent to the requesting user instead of the VC itself.

For the described test use cases, two different Issuers had to be set up, one for University A and another for the Government Organization. Both should fill in the credential

information on their own with the information associated with the user's login in each environment. For example, the University A Issuer should present all Diplomas they are able to claim to the user when they ask for a credential. These depend on what courses the user has completed and what corresponding Verifiable Diplomas the user wants, with all relevant information filled in by the University. In this case, it is the user's personal information that goes in the Diploma in addition to the course's name and level, as well as the place of studies.

4.4.2 Verifier Code

The verification protocol begins with the user attempting to access something on the Verifier's end, i.e. being able to submit an application for a teaching position in the second use case and being able to apply for a job at a company in the first. In both cases, there is a request created on the Verifier's side and sent to the Holder's wallet containing the specification of what credential types are necessary for this exchange. What happens on the Wallet's side is described in the next chapter, but after the end of the Wallet's operations, the Verifier receives a SIOPv2Response data structure, which is received by the VerifierManager's verifyResponse function. Here the Verifiable Presentation is extracted from the SIOP response and matched against the credential types that were requested in the original request. The matching is done by calling the SSI Kit's Auditor element, who does the verification operation.

```
VPVerificationResult(
    vp = vp,
    vcs = vp.verifiableCredential ?: listOf(),
    verification_result = Auditor.getService().verify(
    vp, getVerificationPoliciesFor(req)
    )
)

//the Auditor.getService().verify call gets redirected until finally it reaches
the following end function

fun <T> VerifiableCredential.verifyByFormatType(
jwt: (String) -> T, ld: (String) -> T): T = when (this.sdJwt) {
    null -> ld(this.encode())
    else -> jwt(this.encode())
}
```

This is where the Trusted Issuer Registry would come in if it had been implemented. The Verifier would check the DIDs of the Issuers that were responsible for signing each Credential present in the VP. It would resolve them in the Trusted Issuer Registry to check if they were accredited to perform each respective issuance and also to retrieve their public key and verify the signature on the Credentials. If any of these failed, the verification process would output a fail and the presentation would not be passed. If it is a success, the Verifier would send a session token along with a success message to the wallet; otherwise, it would just send a failure message.

For the use cases described, two Verifiers had to be set up: one for the Company of the first use case and one for University B of the second use case. The Company works as a simple verifier that only requests the Verifiable Diploma and verifies it, whereas University B also functions as an Issuer. This is not used in the test case or presentation but creates a realistic functioning of what a teaching institution's portal should be able to do.

4.4.3 Wallet Code

The wallet side of the code has to support a user friendly presentation of Credentials and of the wallet as a whole, and also has to handle issuance and presentation requests. The Wallet's side of the Credential Issuance process covers the handling of the request to issue a Credential, as well as the management of the newly received Credential. The former is just a binary inquiry passed to the user of whether or not they want to accept the Credential; its result is then returned to the Issuer.

4.4.3.1 Verification

The Wallet side of the verification process consists of receiving a presentation request, putting together a passable Verifiable Presentation and sending it back to the Verifier. In more detail, after receiving the details of the types of credentials being requested, an SSI Kit Custodian function is called to list all owned Credentials that match the requested types. From these, the user can pick the specific ones they want to present (for example, the user may have more than one Verifiable Diploma and only want to present one to the Company). After the credentials to be presented are selected, they are packaged into a Verifiable Presentation along with the DID and a challenge nonce using the Custodian API again. This is packaged in a SIOPResponse object which is sent back to the Verifier;

after the Verifier finishes its side of the protocol, it either sends a failure message or a success message along with a session token.

```
fun fulfillPresentation(sessionId: String, selectedCredentials:
List<PresentableCredential>): PresentationResponse {
    val session =
        sessionCache.getIfPresent(sessionId) ?:
        throw IllegalArgumentException("No session found for id $sessionId")
    val did = session.sessionInfo.did ?:
    throw IllegalArgumentException("Did not set for this session")

    val myCredentials = Custodian.getService().listCredentials()
    val selectedCredentialIds = selectedCredentials.map { cred -> cred.credentialId }
    .toSet()
    val selectedCredentials =
        myCredentials.filter { cred -> selectedCredentialIds.contains(cred.id) }
        .map {
                cred -> id.walt.credentials.w3c.PresentableCredential(cred)
        }.toList()
    val vp = Custodian.getService().createPresentation(
        selectedCredentials,
        did,
        null,
        challenge = session.req.getCustomParameter("nonce")?.firstOrNull(),
        expirationDate = null
    ).toVerifiablePresentation()

    val siopResponse = OIDC4VPService.getSIOPResponseFor(session.req, did, listOf(vp))
    val rp_response = if (ResponseMode("post") == session.req.responseMode) {
        OIDC4VPService.postSIOPResponse(session.req, siopResponse)
    } else null

    return PresentationResponse.
    fromSiopResponse(siopResponse, rp_response != null, rp_response)
}
```

Chapter 5

Conclusion

In the introduction two main goals for the dissertation were established: a literature review and aggregation of the technologies relevant to eIDAS, and the assembling of a functioning SSI ecosystem. Of these, the former was achieved in a complete manner, and a comprehensive elaboration on the technologies underlying and adjacent to eIDAS has been put together, eIDAS itself is explained, as well as the ARF and the EUDI wallet, while technologies such as DLTs and DPKIs are expanded on. The SSO authentication systems are also explained in a detailed manner.

The theoretical portion of the second goal was also concluded, including a complete explanation of the working of waltid and EBSI, while also establishing use cases and presenting some of the Verifiable Credential Schemas available. On the other hand, the practical implementation fell short of what was desired. The final product was missing registries such as the Trusted Issuer Registry and the Identity Registry, as well as the ability to choose from more than one Wallet provider, which resulted in a less complete ecosystem than what was desired. It was only possible to implement the Verifiable Credentials and Holder/Issuer/Verifier core.

The eIDAS regulation aims for a future where the burdensome and time-consuming bureaucracies involving physical identification documents and all the processes associated with them become seamless digital processes that are both less time consuming and user friendly. Given the current landscape of digital identity and all the problems and inconveniences that come with it, SSI is an excellent solution that comes hand in hand with the future eIDAS aims to achieve. It does this by separating digital identity from all the problems that plague it in the current landscape, while providing a sturdy paradigm

from which eIDAS can build its infrastructure, which allows for the interoperability and digital mobility that such a system needs to work across the European Union.

5.1 Future Work

The next step in this area would be the completion of the SSI ecosystem that was planned in the goals section of the introduction, that is, adding Trusted Issuer Registries to the system, as well as the Trusted Accreditation Organizations that accompany them. Of equal importance would be adding the Identity Registry to the system, together with the option to choose between multiple wallet providers when presenting credentials.

In a larger scale, and within the context of eIDAS, it is very important to develop Verifiable Credentials, together with the study and development of the infrastructure surrounding use cases not yet covered, in order to have the widest possible coverage of use cases on completion.

Bibliography

[1] Merriam-Webster, "Identity." [Online]. Available: https://www.merriam-webster.com/dictionary/identity [Cited on page 9.]

[2] R. Soltani, U. T. Nguyen, and A. An, "A survey of self-sovereign identity ecosystem," *Security and Communication Networks*, vol. 2021, pp. 1–26, 2021. [Online]. Available: https://doi.org/10.1155/2021/8873429 [Cited on page 9.]

[3] F. Wang and P. De Filippi, "Self-sovereign identity in a globalized world: Credentials-based identity systems as a driver for economic inclusion," *Frontiers in Blockchain*, vol. 2, p. 28, 2020. [Online]. Available: https://doi.org/10.3389/fbloc.2019.00028 [Cited on page 9.]

[4] P. J. Windley, *Digital Identity: Unmasking identity management architecture (IMA).* " O'Reilly Media, Inc.", 2005. [Cited on page 9.]

[5] K. Cameron, "The laws of identity," *Microsoft Corp*, vol. 12, pp. 8–11, 2005. [Cited on pages 9 and 10.]

[6] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation computer systems*, vol. 28, no. 3, pp. 583–592, 2012. [Online]. Available: https://doi.org/10.1016/j.future.2010.12.006 [Cited on page 11.]

[7] V. Kumar and A. Bhardwaj, "Identity management systems: a comparative analysis," *International Journal of Strategic Decision Sciences (IJSDS)*, vol. 9, no. 1, pp. 63–78, 2018. [Online]. Available: https://doi.org/10.4018/IJSDS.2018010105 [Cited on page 11.]

[8] D. Pöhn and W. Hommel, "An overview of limitations and approaches in identity management," in *Proceedings of the 15th International Conference*

on Availability, Reliability and Security, 2020, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3407023.3407026 [Cited on page 11.]

[9] Y. Cao and L. Yang, "A survey of identity management technology," in *2010 IEEE International Conference on Information Theory and Information Security*. IEEE, 2010, pp. 287–293. [Online]. Available: https://doi.org/10.1109/ICITIS.2010.5689468 [Cited on page 12.]

[10] M. V. Bhonsle, N. Poolsappasit, and S. K. Madria, "Etis–efficient trust and identity management system for federated service providers," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2013, pp. 219–226. [Online]. Available: https://doi.org/10.1109/AINA.2013.13 [Cited on page 12.]

[11] S. El Haddouti and M. D. E.-C. El Kettani, "Analysis of identity management systems using blockchain technology," in *2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*. IEEE, 2019, pp. 1–7. [Online]. Available: https://doi.org/10.1109/COMMNET.2019.8742375 [Cited on page 12.]

[12] V. Radha and D. H. Reddy, "A survey on single sign-on techniques," *Procedia Technology*, vol. 4, pp. 134–139, 2012. [Online]. Available: https://doi.org/10.1016/j.protcy.2012.05.019 [Cited on page 12.]

[13] M. Jones and D. Hardt, "The oauth 2.0 authorization framework: Bearer token usage," Tech. Rep., 2012. [Cited on page 13.]

[14] J. Richer and A. Sanso, *OAuth 2 in action*. Simon and Schuster, 2017. [Cited on page 13.]

[15] "Oauth 2.0 use case flow," oracle's OAuth 2.0 page. [Cited on pages xi and 15.]

[16] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra, "Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps," in *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, 2008, pp. 1–10. [Online]. Available: https://doi.org/10.1145/1456396.1456397 [Cited on page 15.]

[17] "Saml," https://developer.okta.com/docs/concepts/saml/, accessed: 2023-09-6. [Cited on pages xi and 16.]

[18] N. Sakimura, J. Bradley, M. Jones, B. De Medeiros, and C. Mortimore, "Openid connect core 1.0," *The OpenID Foundation*, p. S3, 2014. [Cited on page 16.]

[19] S. Hammann, R. Sasse, and D. Basin, "Privacy-preserving openid connect," in *Proceedings of the 15th ACM Asia conference on computer and communications security*, 2020, pp. 277–289. [Online]. Available: https://doi.org/10.1145/3320269.3384724 [Cited on pages xi and 18.]

[20] "The path to self-sovereign identity," http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html, accessed: 2023-09-6. [Cited on page 19.]

[21] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, pp. 80–86, 2018. [Online]. Available: https://doi.org/10.1016/j.cosrev.2018.10.002 [Cited on pages xi, 19, 21, and 22.]

[22] J. Sedlmeir, R. Smethurst, A. Rieger, and G. Fridgen, "Digital identities and verifiable credentials," *Business & Information Systems Engineering*, vol. 63, no. 5, pp. 603–613, 2021. [Online]. Available: https://doi.org/10.1007/s12599-021-00722-y [Cited on page 20.]

[23] A. Preukschat and D. Reed, *Self-sovereign identity*. Manning Publications, 2021. [Cited on page 20.]

[24] H. Krawczyk and T. Rabin, "Chameleon hashing and signatures," *Internet-http://wwwresearch. ibm. com/security/projects. html*, 1997. [Online]. Available: https://ia.cr/1998/010 [Cited on page 22.]

[25] J. Xu, K. Xue, H. Tian, J. Hong, D. S. Wei, and P. Hong, "An identity management and authentication scheme based on redactable blockchain for mobile networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6688–6698, 2020. [Online]. Available: https://doi.org/10.1109/TVT.2020.2986041 [Cited on page 22.]

[26] M. S. Ferdous, F. Chowdhury, and M. O. Alassafi, "In search of self-sovereign identity leveraging blockchain technology," *IEEE access*, vol. 7, pp. 103 059–103 079,

2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2931173 [Cited on page 22.]

[27] S. Lim, M.-H. Rhie, D. Hwang, and K.-H. Kim, "A subject-centric credential management method based on the verifiable credentials," in *2021 International Conference on Information Networking (ICOIN)*. IEEE, 2021, pp. 508–510. [Online]. Available: https://doi.org/10.1109/ICOIN50884.2021.9333857 [Cited on page 23.]

[28] H. P. Singh, K. Stefanidis, and F. Kirstein, "A private key recovery scheme using partial knowledge," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2021, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NTMS49979.2021.9432642 [Cited on page 23.]

[29] R. Soltani, U. T. Nguyen, and A. An, "Practical key recovery model for self-sovereign identity based digital wallets," in *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2019, pp. 320–325. [Online]. Available: https://doi.org/10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00066 [Cited on page 24.]

[30] G. Linklater, C. Smith, A. Herbert, and B. Irwin, "Toward distributed key management for offline authentication," in *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 2018, pp. 10–19. [Online]. Available: https://doi.org/10.1145/3278681.3278683 [Cited on page 24.]

[31] A. Abraham, F. Hörandner, O. Omolola, and S. Ramacher, "Privacy-preserving eid derivation for self-sovereign identity systems," in *Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers 21*. Springer, 2020, pp. 307–323. [Cited on page 24.]

[32] M. Kang and V. Lemieux, "A decentralized identity-based blockchain solution for privacy-preserving licensing of individual-controlled data to prevent unauthorized secondary data usage," *Ledger*, vol. 6, 2021. [Online]. Available: https://doi.org/10.5195/ledger.2021.239 [Cited on page 24.]

[33] A. Grüner, A. Mühle, T. Gayvoronskaya, and C. Meinel, "A quantifiable trust model for blockchain-based identity management," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1475–1482. [Online]. Available: https://doi.org/10.1109/Cybermatics_2018.2018.00250 [Cited on page 24.]

[34] M. P. Bhattacharya, P. Zavarsky, and S. Butakov, "Enhancing the security and privacy of self-sovereign identities on hyperledger indy blockchain," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2020, pp. 1–7. [Online]. Available: https://doi.org/10.1109/ISNCC49221.2020.9297357 [Cited on page 24.]

[35] T. Zhong, P. Shi, and J. Chang, "Jointcloud cross-chain verification model of decentralized identifiers," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2021, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IPCCC51483.2021.9679363 [Cited on page 24.]

[36] A. Grüner, A. Mühle, and C. Meinel, "Using probabilistic attribute aggregation for increasing trust in attribute assurance," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 633–640. [Online]. Available: https://doi.org/10.1109/SSCI44817.2019.9003094 [Cited on page 24.]

[37] G. Laatikainen, T. Kolehmainen, and P. Abrahamsson, "Self-sovereign identity ecosystems: benefits and challenges," in *Scandinavian Conference on Information Systems*. Association for Information Systems, 2021. [Cited on page 24.]

[38] "Ebsi guidelines," https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/, accessed: 2023-09-14. [Cited on pages xi, 25, 29, 30, and 46.]

[39] J. J. Hunhevicz and D. M. Hall, "Do you need a blockchain in construction? use case categories and decision framework for dlt design options," *Advanced Engineering Informatics*, vol. 45, p. 101094, 2020. [Online]. Available: https://doi.org/10.1016/j.aei.2020.101094 [Cited on page 25.]

[40] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008. [Cited on pages xi and 25.]

[41] U. Maurer, "Modelling a public-key infrastructure," in *Computer Security—ESORICS 96: 4th European Symposium on Research in Computer Security Rome, Italy, September 25–27, 1996 Proceedings 4*. Springer, 1996, pp. 325–350. [Online]. Available: https://doi.org/10.1007/3-540-61770-1_45 [Cited on page 26.]

[42] J. Won, A. Singla, E. Bertino, and G. Bollella, "Decentralized public key infrastructure for internet-of-things," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 907–913. [Online]. Available: https://doi.org/10.1109/MILCOM.2018.8599710 [Cited on pages 26 and 27.]

[43] K. Isirova and O. Potii, "Decentralized public key infrastructure development principles," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2018, pp. 305–310. [Online]. Available: https://doi.org/10.1109/DESSERT.2018.8409149 [Cited on page 27.]

[44] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," *Cryptology ePrint Archive*, 2014. [Cited on page 28.]

[45] M. A. Hassan and Z. Shukur, "Review of digital wallet requirements," in *2019 International Conference on Cybersecurity (ICoCSec)*. IEEE, 2019, pp. 43–48. [Online]. Available: https://doi.org/10.1109/ICoCSec47621.2019.8970996 [Cited on page 28.]

[46] R. K. Balan and N. Ramasubbu, "The digital wallet: Opportunities and prototypes," *IEEE Computer*, vol. 42, no. 4, p. 100, 2009. [Online]. Available: https://doi.org/10.1109/MC.2009.134 [Cited on page 28.]

[47] C. Cuijpers and J. Schroers, "eidas as guideline for the development of a pan european eid framework in futureid," 2014. [Cited on page 30.]

[48] E. Commission, "The european digital identity wallet architecture and reference framework," 2023, https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework. [Cited on pages xi, 35, and 38.]

[49] "walt.id ssi technologies and concepts," https://docs.walt.id/v/ssikit/ssi-kit/what-is-ssi/technologies-and-concepts, accessed: 2023-09-14. [Cited on pages xi and 59.]

[50] "walt.id issuer and verifier portals," https://docs.walt.id/v/web-wallet/wallet-kit/issuer-and-verifier-portals, accessed: 2023-09-14. [Cited on pages xi and 62.]