FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Use Cases for Decentralized Identity
Student:	Miroslav Škrabal
Supervisor:	Ing. Marek Skotnica
Study program:	Informatics
Branch / specialization:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	until the end of summer semester 2022/2023

Instructions

Living in the digital age allows us to access services worldwide and interact with thousands of other people. However, all applications and services collect and store personal data that cause security and privacy concerns. Decentralized identity (DiD) introduces an ability to own and control our digital identity and securely protect our personal information. This thesis's primary goal is to explore how the DiD can be used in state-of-the-art applications and demonstrate it on a proof-of-concept case study.

- Review the essential components of the DiD and provide an overview of the most notable use cases.
- Review the privacy and the security aspects of the DiD, including the zero-knowledge-proofs.
- Analyse one use case and create a proof-of-concept case study.
- Evaluate the case study's security with DiD compared to state-of-the-art approaches.

Electronically approved by prof. Ing. Pavel Tvrdík, CSc. on 1 February 2022 in Prague.

Bachelor's thesis

USE CASES FOR DECENTRALIZED IDENTITY

Miroslav Škrabal

Faculty of Information Technology
Department of Information Security
Supervisor: Ing. Marek Skotnica
May 4, 2022

Contents

List of Figures

List of Tables

List of code listings

I would like to thank my supervisor Ing. Marek Skotnica for all his advice and guidance.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 4, 2022 .....

Abstrakt

Decentralizovaná identita je nově vznikající technologie, jejímž cílem je zavést nový standard pro reprezentaci online identity. Je v opozici k současným systémům identity, kde uživatelé obvykle nemají kontrolu nad svou identitou a osobními údaji. Cílem decentralizované identity je zvýšit bezpečnost a soukromí koncových uživatelů. Je založena na peer-to-peer protokolech, kryptografii a kryptograficky ověřitelných tvrzeních o identitě.

Tato práce zkoumá možné případy použití decentralizované identity jako je autentizace, autorizace nebo soukromá komunikace. Zároveň se zaměřuje na aspekty bezpečnosti a soukromí. V tomto kontextu představuje zero-knowledge proofs nebo jak decentralizovaná identita může pomoci k omezení úniků databází hesel. Nakonec je jeden z případů použití demonstrován na případové studii s využitím open-source softwarových nástrojů.

Klíčová slova Decentralizovaná Identita, Bezpečnost, Soukromí, Případová Studie, Případy Použití, Bezpečnostní Analýza

Abstract

Decentralized identity is a newly emerging technology that aims to establish a new standard for online identity representation. It opposes the current identity systems, where users are usually not in control of their identity and data. The decentralized identity aims to enhance the security and privacy of end-users. It is based on peer-to-peer protocols, cryptography, and verifiable identity claims.

This thesis explores the possible use cases of decentralized identity like authentication, authorization, or private communication. It also discusses the security and privacy aspects. In that context, it introduces the zero-knowledge proofs, or how decentralized identity can help to mitigate password database leaks. Finally, one of the use cases is demonstrated in a case study using open-source software tools.

Keywords Decentralized Identity, Self-Sovereign Identity, Security, Privacy, Case Study, Use Cases, Security Analysis

List of abbreviations

DID	Decentralized identifier
DiD	Decentralized identity
SSI	Self-sovereign identity
SIOP	Self-Issued OpenID connect
dApp	Decentralized application
ZKP	Zero-knowledge proof
VC	Verifiable credential
VP	Verifiable presentation
PoW	Proof of work
EHR	Electronic health record

Introduction

Identification on the internet has been posing problems ever since the internet has gained a wider audience. The founders of the protocol did not include an identity layer in it, and the identification problem has since remained unsolved. Users are victims of various phishing attacks and identity thefts, their privacy is limited, and overall, they have minimal control over how their personal information is dealt with. Identity representation on the internet has been changing since its inception. However, a universal standard that would mitigate those problems is yet to emerge.

The evolution of the identity representation on the internet can be categorized into three main categories—centralized, federated, and the newly emerging decentralized. The centralized is rather ad-hoc—individual services built custom identity solutions that suit their needs. They are usually in a position of identity providers, i.e., users are not the sole owners of their data. The federated identity model emerged as an attempt to solve the problem of the fragmentation of the identity systems. Companies like Google or Facebook provide an option to be identity providers for other systems. As such, they decrease the fragmentation but increase the centralization of power.

The decentralized identity attempts to solve the problem of identification on the internet. Its main principle is to put control into the hands of the users. The idea behind its protocols is that the users should be able to decide what information they want to share, when they want to share it, and with whom. It also attempts to provide standards for identity representation that could be utilized across all systems, thus providing greater interoperability and improving user experience.

This thesis firstly introduces the maian concepts and components upon which the decentralized identity is built. It follows with an explanation of how those components interoperate. After the essentials are discussed, the thesis explores the most notable use cases like authentication, the European Self-Sovereign Identity Framework, or possible usage in healthcare. The decentralized identity is still somewhat in the conceptual stage with few emerging protocols and standards, and therefore only a high-level overview of the use cases is provided. This thesis also analyzes the privacy and security aspects of the decentralized identity. Advanced cryptographic techniques like zero-knowledge proofs are introduced, and their usage in the ecosystem is described.

The thesis introduces a proof-of-concept case study to showcase how the discussed topics can be put together. The case study shows how decentralized identity can be used for creating privacy-preserving COVID-19 vaccination credentials.

Lastly, a security and privacy comparison of the proposed credentials to the vaccination credentials used in the European Union is provided.

Chapter 1

Introduction to Decentralized Identity

This chapter briefly introduces the history of identity representation on the internet. Then it proceeds to provide a high-level description of the decentralized identity[1] (DiD) ecosystem, its principles, and the problems it tries to solve.

1.1 Evolution of Digital Identity

How identity is handled on the internet is an ever-changing process, but certain vital phases can be laid down. In [1] Christopher Allen argues that the identity on the internet evolved in four main phases: centralized, federated, user-centric, and self-sovereign identity.

Centralized Identity In the 80s and 90s, after the internet was created, organizations that would oversee the identification on the internet started to emerge. Namely, it was IANA to oversee the allocation of IP addresses, ICANN to oversee the domain names, and certificate authorities (CAs) to act as the trusted third parties and allow users to prove their identity via means of cryptography [1]. But as stated in [1], such a system can suffer from the centralization (though at least in the case of the CAs hierarchies emerged) because the authorities can deny the user's identity or they can create illegitimate ones. Companies like Google or Facebook can decide to deny a user's identity and prevent him from accessing his personal information stored within their systems.

Apart from the problems with centralization, another problem with fragmentation occurred [1]. As the internet evolved and new online services were created, they all had to face the problem of identifying their clients. As a result, many custom and noninteroperable identity solutions were created, which led to the fragmentation of the identity system.

[1] states that an emerging decentralized alternative in that era was PGP, which introduced the Web of Trust, which allowed peers to act as certificate authorities by themselves. The PGP model allows users to introduce new peers to the Web of Trust by signing their public keys and thus vouching for their identity. However, PGP never became widely adopted.

[1]The decentralized identity is sometimes called self-sovereign identity (SSI). The distinction is not clearly defined, and thus the terms are often used interchangeably. Nevertheless, suppose a distinction was to be made after all. In that case, SSI can be perceived as more focused on individual users, their sovereignty, and the DiD as the broader identification concept, which applies to all entities, e.g., persons, trade items, or IoT devices.

Federated Identity Federated identity systems allow users to link their identities across multiple identity management systems. For example, federations can be used for a single sign-on inside one organization. They can also be used more globally, like in the case of Google or Microsoft, who act as identity providers and provide independent organizations the service of validating the identities of their customers.

This is an improvement over the fully centralized environment, mainly in terms of interoperability. On the other hand, the big companies gain access to more personal information, minimally the information about what services the users use. It also causes the users to be more dependent on the identity providers and creates a single point of failure.

User-centric Identity User-centric identity systems prelude the decentralized identity vision, and they focus on interoperability and users being in the control of their data [2].

Users can store their identity data within an identity provider independent of the actual service provider. Services then can connect to this identity provider and incorporate relevant identity claims, e.g., into the authentication process. Consequently, the services do not act as identity providers for that particular user. Users can connect their identity provider to multiple services and thus create a more interoperable system.

In a user-centric system, the flow of personal data from an identity provider should be only initiated by the user [2]. Multiple famous services that adopted this vision are: OpenID, OpenId Connect, or OAuth [1].

Users can create their OpenID identity providers and thus share identity claims in an autonomous way [1]. However, this autonomous vision has not gotten widely adopted amongst individuals and instead converged to the federated identity model, as Google and Microsoft often act as the identity providers.

Self-sovereign Identity The self-sovereign identity (SSI) mandates that users are in complete control of their identities. Users should not be dependent on any identity provider, and instead, they should be able to act as their identity providers [2]. They should be able to decide with whom and when to share their personal information. SSI is based on three main principles—security, controllability, and portability. Those principles are expanded on in the following section. How users can be their own identity providers and how the principles can be maintained is discussed throughout this thesis.

1.2 Principles of Decentralized Identity

DiD systems can be characterized as systems where users exist independently from services and fully own and manage their identities [3]. However, as is common with decentralized technologies, there exists no exact definition agreed upon by everybody, and providing a precise definition isn't possible [3]. Christopher Allen's post [1] outlined basic properties of self-sovereign identity, which were adopted by Sovrin foundation in [2] and grouped in 3 overarching categories [3].

Table 1.1 Principles of Decentralized Identity [2]

Security the identity information must be kept secure	**Controllability** the user must be in control of who can see and access their data	**Portability** the user must be able to use their identity data whereever they want and not be tied to a single provider
Protection	Existance	Interoperability
Persistence	Consent	Transparency
Minimisation	Control	Access

1. **Security** DiD attempts to increase the security of the user's data and his overall privacy. User's data should be stored securely and controlled by the user himself. The identity data should be persistent, not tight to one single storage. Storing the personal data encrypted with the user's private key on cloud storage might be necessary. The principle of minimal disclosure should be followed, i.e., only the data necessary for the given use case should be disclosed. The user's digital fingerprint should be minimized, and data should be discarded after no longer needed. Techniques like zero-knowledge proofs for sharing personal information are endorsed.

2. **Controllability** The user's identity should exist independently of any service provider. All personal data manipulation should be consent-driven, i.e., the user should be able to decide when with whom and to what extent to share his personal data. The user's personal data should be persistent. However, he can decide to delete any of it.

3. **Portability** The identity is independent of services, and it is not tied to a specific domain. The user should be able to port his personal information across different domains whenever he wishes. Personal data should always be accessible to the user, and services should be transparent about using users' personal data.

One of the essential parts of the DiD is the data that the user can accumulate about his identity. The user can receive the data in the form of claims. The DiD ecosystem utilizes verifiable claims, i.e., the claims are digitally signed.

[4] talks about a concept of an identity container. In DiD system, an individual can create an identity container that allows him to receive and issue claims from/to other third parties. Claims are cryptographically verifiable and can be created by anyone. The claims can have a form like: "Person A is X years old." or "Person A studied at college Y.". Each user is in full possession of this container. He can provide claims from the container to others, and they can either accept or deny that claim based on whether they trust the issuer. A digital identity wallet would usually represent this container.

Another important dimension is how users in DiD connect. In DiD, decentralization is emphasized, which is also the case for information exchange and data transmission. Individuals often have the roles of peers and communicate with each other on a peer-to-peer basis. An example of a decentralized communication protocol is DIDComm, discussed later in 3.5.

In conclusion, DiD tries to build an identity system that has properties of security, controllability, and portability. Users can issue, hold and verify claims that are digitally signed. Each user is the possession of such claims and decides when and with whom to share them. The exchange of claims, and overall, the entire communication, is envisioned to be peer-to-peer and built on private and secure protocols.

Chapter 2
Essential Components

This chapter describes the essential components of the decentralized identity ecosystem. It introduces domain-specific components like decentralized identifiers (DIDs), DID documents, and verifiable credentials (VCs). Later, it introduces the components that are not intrinsic to the identity domain but essential for the ecosystem to work—such components are wallets and blockchains. Finally, it is described how these components interoperate.

2.1 Domain-Specific Components

Each identification system has to assign identifiers to the individual users. In DiD system, this is done using *decentralized identifiers*. The DIDs are usually based on cryptographic keys and thus are not human-readable. The DIDs usually resolve to a *DID document*. However, that is not always the case, like in the case of peer DIDs, which are discussed in detail in 4.4.2. The DID document is a data structure usually stored on a DLT, mainly blockchains. This data structure stores the user's communication endpoints and public keys (and other optional data). Users also can issue and receive claims. In the DiD systems, those are called *verifiable credentials* or *verifiable claims*. Those three components are the essential building blocks from which the others are spawned. The following sections will discuss them in detail.

2.1.1 Decentralized Identifiers

DIDs are identifiers that can refer to any resource and provide functionality for verifiable and decentralized identity [5]. DIDs are specified by the W3C DID Working group. They were designed so that they can be created without centralized identity providers or certificate authorities [5]. From the structural standpoint, DIDs are a specific type of URI and evolved to have multiple different representations, specifically called *DID methods*, which further specify the operations on the DID. The entity that the DID identifies is called a *DID subject*. The DIDs allow creating associations between the DID subject and exactly one DID document [5].

[7] mentions 4 essential characteristics of DIDs:

1. *Decentralization*: creating a new identity is permissionless. No central authority oversees the process. The user can create as many identities as he wants by himself.
2. *Resolvability*: a guarantee of the ability to query the corresponding identity information.
3. *Cryptographic verification*: possibility to provide cryptographic proof of ownership of the DID.

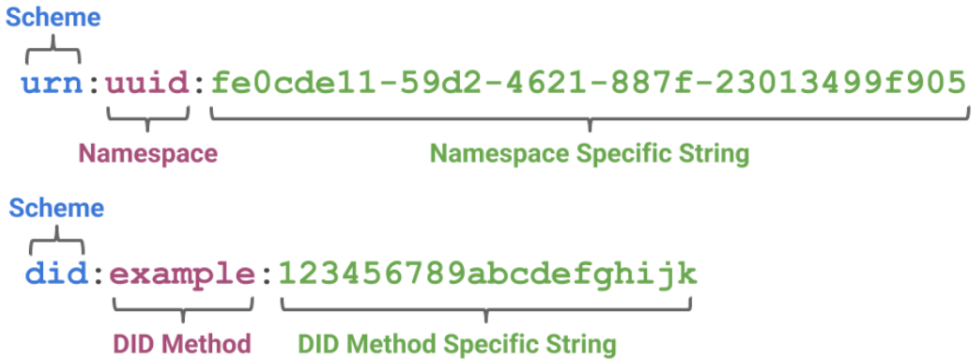

Figure 2.1 Structure of DIDs [6]

4. *Persistence*: only the user can revoke the identifier. He can do so independently of external systems.

2.1.1.1 DIDs As Public Key-Based Identifiers

As stated, DIDs can be created without centralized identity providers. This is because they are derived from the user's public key. The same approach for creating identifiers is used in the cryptocurrency world, where it is often the case that the address is derived from a public key, usually by applying a hash function [8].

Public key-based identifiers have two main disadvantages. Firstly, the public key-based identifiers are not human-readable. Secondly, once it is needed to rotate the public keys, the identifiers itself has to change. Such identifiers, therefore, lack in terms of persistence [8].

However, public key-based identifiers also have several advantages. The main advantage is that a user can easily prove that he *is the owner* of the given identifier, e.g., he can provide signed proof using his private key. In the current systems, binding an identifier (usually a public key) to its controller is commonly done using a X.509 certificate issued by a certificate authority. The certificate binds an identifier (for example a URL) to a public key that the controller controls. In the DiD system, the binding is gotten for free. However, the binding does not guarantee anything about the DID subject itself. The certificate authorities often also verify the owner of the identifier and thus provide another binding—binding of a real entity to its virtual identifier. In DiD the second type of binding is envisioned to be done using VCs issued by some trusted party. Thus, in this case, the model would be similar to the model of certificate authorities. However, in DiD, anyone can easily have the role of a certificate authority, and therefore, it is expected that such binding services would be more decentralized and diverse. The second advantage is that anyone can create the identifier in a decentralized fashion, and as such, the users cannot be victims of censorship or denial of service. Another advantage is that by using public key-based identifiers a naming collision cannot occur with any significant probability.

How DIDs Solve the Disadvantages The DiD system has solutions for both of the mentioned disadvantages. The readability and meaning can be brought to the identifiers via VCs. Naming services are expected to emerge. They would issue VCs that would assign the DID subject a name and attest to the uniqueness of the name. The VC layer can be, from a specific standpoint, seen as the semantic layer for the DIDs. Another possibility is to map a DID to a name via some DApp. For example, on Ethereum there exists Ethereum naming service (ENS), where users can buy names (domains) in a decentralized fashion in a bidding process. By using a decentralized service, it can be ensured that any authority cannot revoke the name. On the other hand, the decentralized protocols for naming also bring some disadvantages. One of them

is name squatting, where anonymous users can buy, for example, names of famous brands. Because of their anonymity, there can be no legal steps taken to defend the brand's name against misuse.

The persistence problem can be solved using blockchains and DID documents. The keys are updated in the DID document, but the DID remains the same. The exact process for doing this is described in later chapters.

Comparison to Centrally-Owned Identifiers Centrally-owned identifiers are only useful in specific contexts and are recognizable only by some portion of authorities [5]. The central authorities can revoke the user's identifier, or the identifier can cease with the failure of the issuing organization [5]. They often can be replicated and asserted by malicious parties, which can even lead to identity theft [5].

Decentralized identifiers should be in opposition to the negatives of current identifiers. [5] lists the benefits of the DIDs:

- Users can easily generate, using systems they trust, as many new identifiers as they wish to maintain the separation of identities.
- Users can prove control of the identifier using cryptographic proofs.
- DIDs are a permanent identifier and do not have to change.

2.1.1.2 DID Document

A DID document is a set of data that describes a DID subject and is relevant for communicating with the DID subject [5]. A DID corresponds to one DID document and is used for resolving the DID document. To avoid tampering with the data inside a DID document, DID documents are usually stored in a distributed ledger. In the case of peer DIDs, explained in 4.4.2, the DID document is stored by the second peer in the connection.

A DID document should include just the information needed to enable secure communication with the associated DID subject [8]. A DID controller can change values inside a DID document. A DID document usually contains [6]:

- the corresponding DID itself,
- public keys of the DID subject,
- communication endpoints of the DID subject,
- timestamps of changes.

2.1.1.3 DID Method

The DID method defines how the formal syntax of the decentralized identifier is implemented [5]. The DIDs are the same from the functional standpoint, but they differ on the formal side [8].

No central authority oversees creating new decentralized identifiers, and the platforms that utilize the DIDs differ from the architectural standpoint, resulting in multiple representations of the identifiers [8]. Currently, over 110 DID methods are registered [9].

Each DID method has its name (like ethr or btcr). It also has a method-specific identifier which works like a namespace for the given platform. Such string is unique for a given method and should be unique globally [8]. For the DIDs to be useful, users must be able to interact with them, and thus the DID method defines basic CRUD operation for interaction with the DID. [8] describes the operations as:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

Code listing 1 A Simple DID Document [5]

- *Create* creates a DID and its associated DID document.
- *Read* resolves the DID and obtains the associated DID document.
- *Update* updates the data in the DID document.
- *Delete* revokes the DID.

2.1.1.4 Resolving DIDs

A DID method defines a READ operation on the DID. This operation is used to resolve the corresponding DID and retrieve the underlying DID document. The resolution process is handled by specialized servers—resolvers. Clients for resolvers can be included as a library in an application or operating system—similarly to current DNS resolvers. Running own node is resource-intensive, so the client-server paradigm is expected to be the most heavily used. Therefore, it is essential not to centralize the system around a few resolvers, which would handle the majority of the DID resolution.

The READ operation is userul mainly for the following operations: [8]

- lookup a public key of an issuer to verify subjects VC,
- authenticate the DID subject or controller,
- request DID-DID connection with the DID subject or controller.

2.1.2 Verifiable Credentials

A claim is some statement about a property of a specific subject. A verifiable claim is a claim that has an associated proof and can be converted to a verifiable presentation and sent to a verifier.

”Verifiable claims are verifiable through a signature of an attestation issuer that has either issued the claim himself or can atest the correctness of it. Attestation can be seen as a proof in the form of a signature attesting to a certain claim and metadata needed for verification such as a name, validity period and signature scheme.” [3]

A subject can have many claims associated with its digital identity. For example, a university can claim that a student successfully finished the curriculum. Unlike credentials in the physical

world, digital credentials are more tamper-evident and can be transmitted rapidly, providing users the option to establish trust quickly at a distance [10].

2.1.2.1 Flow of Verifiable Credentials

In [10] abstract roles to describe the DiD ecosystem were proposed:

- *Holder*: an entity that holds VCs and can use them to create a verifiable presentation out of them.
- *Issuer*: an entity that asserts claims about other entities.
- *Subject*: an entity about which the claims are made. The subject will often be the same as the holder, but they can differ. For example, a parent can be the holder of claims of their children.
- *Verifier*: an entity that verifies the claims that it received from holders.
- *Verifiable data registry*: a role that can be mediated by a system, which can provide the creation and verification of identifiers, keys, revocation registries, and other relevant data [10]. A distributed ledger would usually mediate this role.

Lifecycle of VCs In [10] a typical lifecycle for VCs and the mentioned entities is shown:

1. Issuer issues a verifiable credential.
2. Holder stores the credential, e.g., in an identity wallet.
3. Holder creates a verifiable presentation using the verifiable credentials he holds.
4. Verifier uses the proof contained in the verifiable presentation to verify the credentials he is presented with.

2.1.2.2 W3C Verifiable Credentials

Verifiable credentials (VCs) are standardized by W3C and have their own W3C Recommendation. A VC is a set of claims about an entity, accompanied by metadata and proof. The metadata, amongst others, includes a public key for verifying the claims, validity period, and identifiers of the participants. VCs are assembled in a way that is tamper-evident and verifiable [10].

Data Fields of VCs A VC can contain any fields needed to express the semantics of the given use case. However, some fields are declared as mandatory in the W3C Recommendation [10]:

- *id*: identifier of the VC,
- *type*: type of the VC, e.g., a university diploma,
- *issuer*: issuer of the VC,
- *credentialSubject*: the subject about which claims are made,
- *issuanceDate*: date of issuance of the VC,
- *proof*: signature of the VC provided by the issuer.

The data fields can be serialized in any format capable of retaining their meaning. However W3C standard opted to realize the data model via JSON and JSON-LD [10].

```
{
    "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/1872",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuer": "https://example.edu/issuers/565049",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": [{
        "value": "Example University",
        "lang": "en"
      }, {
        "value": "Exemple d'Université",
        "lang": "fr"
      }]
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://example.edu/issuers/565049#key-1",
    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYj.......TCYt5X"
  }
}
```

Code listing 2 A Simple Verifiable Credential [10]

Presentation of VCs Verifiable presentations (VPs) are used for exchanging VCs. VP is similar to a VC, but it does not contain claims, but VCs themselves [8]. The holder may sign VP and thus make the presentation verifiable. A VP has only one mandatory field, which is the *type* field [10]. Creating and sharing a VP would often be mediated by the user's digital wallet. VCs can also be presented directly and become verifiable presentations themselves. [10]

Presentations and ZKPs: The VCs can also have a particular derived representation. For example, such a representation can have a form of zero-knowledge proof (ZKP). ZKPs are discussed later in detail in 4.1. Those presentations are incorporated to enhance the privacy of the users. [10] mentions use cases of ZKP presentations:

1. Combine multiple VCs into the presentation without revealing the VC or subject to the verifier.
2. Selectively disclose parts of a VC without the need to create a new VC that would contain just that one part.
3. Present a VC in a new format different from the original one issued by the issuer while retaining verifiability without contacting the issuer.

Meaning of Fields of VCs VCs can have various "raw" fields. Schemas are used to interpret the meaning of those fields and find out what types of values these fields can have. With VCs, two types of schemas are used— so-called credential schemas and JSON-LD schemas. Both types serve the same purpose but are used for different types of VCs.

A credential schema is a data template for a credential. It provides a structure and meaning to credentials derived from the schema [11]. It specifies the format and type of data in the credential and the metadata about the schema like id, type, name, or author [11]. Issuers publish their schemas, usually to a blockchain, and verifiers can later query the blockchains for the schemas to extract meaning from the credentials they verify [11]. The goal of schemas is to enhance the semantic interoperability between different issuers and verifiers. Apart from interoperability, schemas also provide an option for reuse, i.e., it is expected that one standard schema would emerge for each type[1] of VC. Credential schemas are mainly used with one particular type of VCs called AnonCreds. AnonCreds are used in the Indy ecosystem, but their popularity is decreasing.

Currently, VCs based on JSON-LD are more popular. JSON-LD is used with structured templates called schemas [8]. JSON-LD processors can parse the schemas and know what values the schema properties expect [8]. The location for finding a given schema is specified with a *@context* property. To specify which schema to use from the given location, *type* property is used. The *@context* property can be thought of as a namespace. Different VCs can have the same fields interpreted differently based on their context [8].

2.2 DiD Related Infrastructure

This section introduces the main infrastructural components used in the DiD ecosystem. It focuses on agents, wallets, and blockchains. It also discusses their interoperability and the relationship to the domain-specific components introduced in the previous sections.

2.2.1 Wallets

An identity wallet acts as a secure database for the user's private data, like VCs, DIDs, or keys, and as an agent that uses this data to perform DiD-related actions, like routing messages, key

[1] The meaning of type here is different from that of type as it was introduced in the first paragraph of the section. Here it is used to distinguish between the purpose of the VCs, i.e., distinguish between a diploma VC and a vaccination VC. The second meaning refers to the syntactical structure of VCs.

management, or sharing VCs. This section discusses the concept of agents, wallets, and their design principles.

Agents are software that operates atop the resources in the wallet. They are essentially wrappers around the wallets [8]. The agents are envisioned to be split into two layers: an edge and a cloud layer [12]. The **cloud layer** is responsible for mediating interactions between peers [12]. Their role in DiD infrastructure is similar to that of web and SMTP servers for mail and web applications, respectively [12].

The **edge agents** operate on the users' local devices and are interacted with directly. Unlike cloud agents, they should be mainly responsible for key management, as they are directly under the control of the users. They are the least accessible for network intrusion. A compromise of an edge device should yield private data of only one user [12]. The edge agents can also mediate p2p interactions, e.g., via NFC, Bluetooth, or by scanning a QR code [12]. The edge and the cloud agents are directly paired. Through the cloud agents, the data flow between peers is mediated [12].

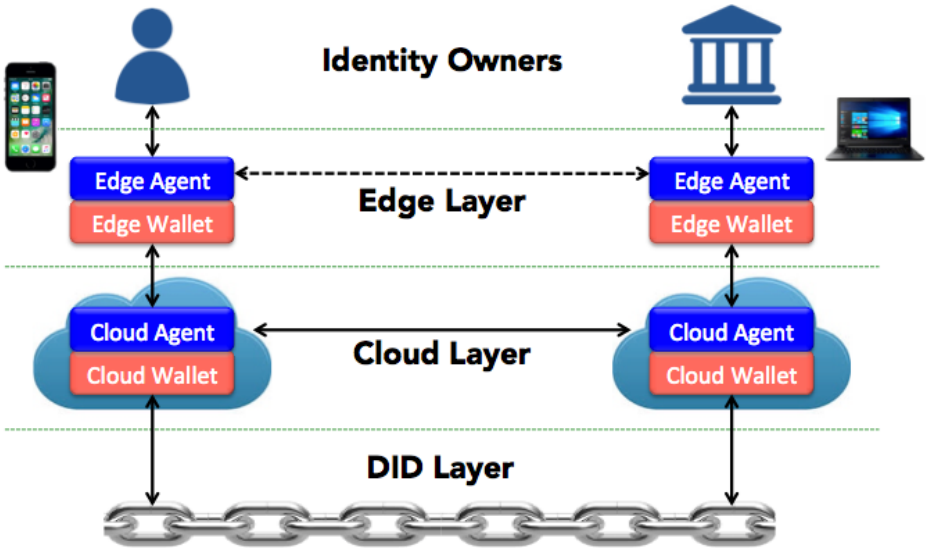

Figure 2.2 High-level architecture of agents and their interactions [12]

Principles for Creating Wallets Main principles for developing identity wallets were mentioned in [13]:

1. *Consent*: wallet should function transparently, and all of its actions should be initiated by the user.

2. *Privacy*: wallet should reveal only the information the user wishes to and only the minimal subset of information to perform the task.

3. *Security*: wallet should adhere to the state-of-the-art security principles, i.e., use the standard cipher sets, security protocols, and libraries. It should utilize the hardware for secure computation, mainly for key management. It should be auditable.

4. *Portability*: wallet should implement open standards and accept any standardized VCs to enable portability amongst different wallet vendors.

Functionalities of Identity Wallets Identity wallets are expected to provide those main functionalities eventually:

1. *Claims*: the wallet should support storing, signing, and verification of claims. It should also act as the mediator of exchange of the verifiable claims and presentations.
2. *Storage*: the wallet should handle storage of all identity material. That includes VCs, DIDs, cryptographic keys, etc. It should also handle the production of backup files.
3. *Cryptography*: the wallet should be responsible for verifying signatures on presented claims, signing verifiable presentations, key-management and DID generation.
4. *Authentication*: the wallet should provide the user an option to securely connect to services by mediating the role of an identity provider.
5. *Recovery*: the wallet should handle recovery[2] and backup of stored data. The backups should be distributed to avoid a single point of failure. Backups should be encrypted.

2.2.2 Blockchains

This section discusses the relationship between blockchains and DiD. In the context of DiD, blockchains are mainly helpful in validating public keys that correspond to proofs of claims. Additionally, they are helpful for VC revocation. Blockchains also provide discoverability for DIDs. The DiD-related data stored on a blockchain is called a DID document, a data structure containing a public key and communication endpoints. Anyone can query the blockchain, read the DID documents, and establish communication channels with the relevant DID controllers.

Important Properties of Blockchains for DiD "Blockchains are tamper-evident and tamper-resistant digital ledgers implemented in a distributed fashion and usually without a central authority." [3] [14] Blockchains are also immutable[4]. Thus, users can rely on the DID documents not to change. Blockchains thus can be considered as distributed sources of truth for all identity data that users decide to publish in a DID document. On the other hand, sometimes immutability can be problematic. For example, in the European Union, there exist a privacy regulation called GDPR which states, besides other, that everyone has the right "to be forgotten" and get their digital footprint deleted.

The decentralized[5] nature of blockchain is another vital building block for the identity system because it can ensure that no one will be censored while creating their identity. Decentralization provides another benefit—no single point of failure.

2.2.2.1 Usecases of Blockchain in DiD

Key Compromise and Rotation Sometimes a private key can get compromised. In such case, its controller might wish to rotate and revoke the corresponding public key. Without a possibility for a rotation, the DID might be forever compromised and become worthless. Revocation

[2] Wallets are expected to run mainly on smartphones, which can easily be lost or stolen. It is, therefore, crucial to support robust recovery methods. Otherwise, users could potentially lose all their identity data.

[3] Describing blockchain technology in detail is out of the scope of this work. Only the essential properties relevant to DiD are mentioned. An extensive overview can be found in [14].

[4] This does not have to be strictly true. For example, in PoW systems, a 51% attack can be made, and a new longest chain with alternative history can be created [14]. That changes the consensus of what is the valid state of the blockchain. However, such attacks would usually be costly.

[5] Not all blockchains are decentralized. Some are permisioned, some permissionless. The overall openness and decentralization of the platform depend on many parameters like consensus algorithm, number of validator nodes, etc. Some blockchains might be a wrong choice for creating an identity system.

is often handled using centralized servers. However, such an approach is against the principles of DiD. Instead, DiD aims to use blockchains for this purpose.

Blockchains help distribute the information about rotation and revocation of the keys to other users who might want to use the keys, e.g., for verification purposes. Blockchains are also convenient for reestablishing the existing connections after the rotation. After a key of one peer is rotated, other peers can query the blockchain with the known DID and retrieve the new public key.

Blockchain properties are overall convenient for the revocation use case—it is important that each user can check that his keys were really updated by himself—this is sometimes called inclusive accountability. Additionally, it is essential to act quickly in case of a security breach. In this context, the property of blockchains being always-on and non-discriminatory is very important.

What Data Goes on Blockchain The primary data structure stored on a blockchain is a DID document, which was already discussed. Another data that might go on the blockchain is VC metadata. The main reason for this is VC revocation [15]. For example, EBSI, discussed in 3.3.1, wants to support the revocation of individual VCs. For that reason, it is planned that EBSI will store some VC metadata on the blockchain [15]. For example, a university might wish to have an option to revoke a credential of its alumni in case of a plagiarised thesis [15]. One of the main benefits of storing such information on a blockchain is accessibility. The verifiers do not have to retain a connection to each issuer if they need to check whether some credential is revoked or not. The only connection needed in such a case is the one to the blockchain. Another benefit is the availability of information on the blockchain. A verifier can query it anytime he needs it.

Another common type of data stored on a blockchain is VC and DID schemas. Those are used to provide interoperability across the ecosystem. For example, EBSI plans to use so-called Trusted Schemas Registry for exactly this purpose [16]. Overall, there exists no rule that would decide what should be stored on-chain, but in the classical notion of DiD, only the DID document, schemas, and optionally the VC metadata are stored on-chain.

Blockchain Interoperability With many different did:methods (did:sov, did:btcr, did:ethr, did:celo, etc) that implement the functionality of DIDs and DID documents atop different ledgers, the question of interoperability arises. Resolvers can be used to solve this problem. A resolver takes a DID as input and produces a DID document as an output [5]. Once the DID document is resolved, it can be operated with platform-agnostic [6]. Interoperability can be achieved by developing a network of resolvers, which would be able to query the most common types of blockchains. However, it is essential not to centralize the system around a few universal resolvers that would handle the majority of DID resolution.

Conclusion Blockchains are usually tamper-resistant, distributed, and persistent. They can act as a source of truth for various DiD-related data. They are vital when it comes to key management. It enables users to easily rotate and revoke keys and update the information in a globally discoverable and accessible way. Users can rely that their data will update if they wish so, and they can check that it was updated.

On the other hand, many functionalities of DiD can be used without blockchain as they are based on pure peer-to-peer interactions. Such an example is DIDcomm explained in 3.5 and did:peer explained in 4.4.2.

[6] In this sense, interoperability in DiD is much simpler than in the case of cryptocurrencies, where it is handled by interoperability networks like Axelar or, more generally, by bridges. In DiD, the identity data is handled primarily off-chain.

Chapter 3
Use Cases

This chapter explores use cases of the decentralized identity. It covers concrete implementations and also high-level concepts. The DiD itself has not gotten traction yet and lacks robust protocols and tooling thus it is not always possible to provide a detailed description of the use case from a technical standpoint.

3.1 Authentication

Motivation Signing in to the web services can be unpleasant from the UX standpoint and is often a security vulnerability. Users have to keep track of dozens of credentials, and the service providers have to store them securely.

The username and password pair usage is just one of many authentication options. Generally, authentication falls into three main categories—an identity secret (a password), a trust object (a token), and biometry (a fingerprint) [17]. However, using trust objects or biometry usually requires additional hardware equipment on the client-side and additional protocol support on the server-side. They are not as widely used on the web, although tokens like YubiKey are becoming popular for 2FA.

Some of the mentioned problems can be mitigated using authentication based on the DiD. Authentication with DiD can be more convenient from the UX standpoint and provide better security. How DiD authentication can help from the security standpoint is described later in 4.2.

Principle of DiD Authentication Each DID subject can prove ownership of his DID. He also should be the only one with access to the corresponding private key. On those facts is the authentication mechanism based.

Authentication via decentralized identity is being worked on via the development of multiple protocols. However, all of them use the same central idea—provide proof of ownership of a DID to the relying party. Currently, the community (mainly the DIF foundation and OpenID Foundation) is focusing on developing the Self-Issued OpenID Provider protocol (SIOP), which is based on OpenID Connect. However, other authentication protocols such as Webauth or Message Layer Security are also to be worked on in the future [18]. OIDC, as such, is a well-known, widely used, and mature protocol. Therefore, it was chosen as the first for decentralized identity authentication [18].

Self-Issued OpenID Provider v2 OpenID Connect (OIDC) is a protocol that enables clients to verify the end-user's identity via authentication provided by an Authorization server. OIDC is widely used, and the Self-Issued OpenID Provider (SIOP) protocol aims to be backward

compatible with it. SIOP is a protocol that allows end-users to authenticate themselves with self-issued ID tokens and present claims to the relying party (RP) without interacting with any third party [19]. In SIOP, the ID tokens used for authentication are signed by the end-user's private key, and the end-user mediates the transmission of all information. This is in contrast to the OIDC, where RP trusts an ID token based on the relationship with its provider. The provider usually is a third party with some reputational or legal stake [19]. SIOP does not have to run on the device of the end-user. It can be hosted on a cloud [19].

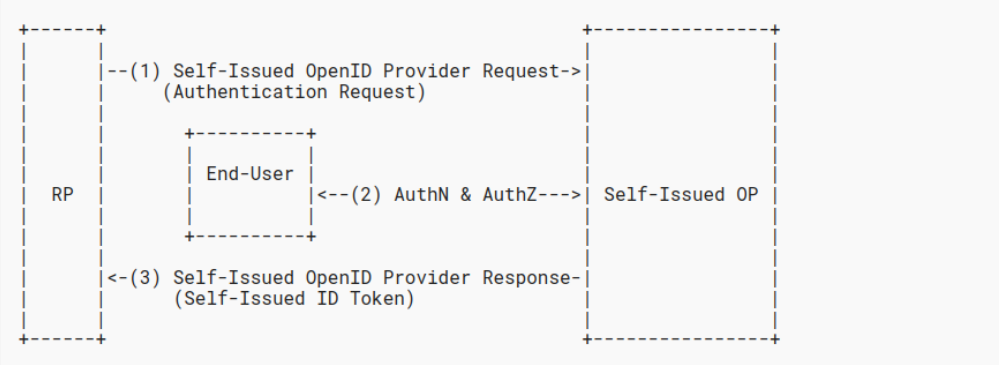

Figure 3.1 SIOP Protocol Flow [19]

Additional Benefits of SIOP SIOP provides some additional benefits:

1. *Availability*: user and the client are not dependent on the authorization server, which can have downtimes or might be unavailable due to other reasons.
2. *Portability*: DIDs are portable. A user is not dependent on the third party that would provide him an identifier [19]. He can also easily create new identifiers for new connections, which is beneficial from a privacy standpoint.
3. *Edge-authentication*: the authentication can be done directly on the edge, even in a low connectivity environment [19]. It is possible because no third-party authorization server has to be involved.

3.2 Authorization

Authorization is concerned with access control. Based on access policies, users are granted or denied access to resources. The policies decide what permissions users get and what actions they can take based on those permissions. [20] provides a basic general authorization schema that consists of:

1. *resource*: physical resource/service/software,
2. *resource owner*: an entity that controls the resource and decides the access policies,
3. *authorized user*: a user granted some permissions and has the privileges to operate with the resource.

There exist many authorization strategies how for granting permissions to users. The most well-known are attribute-based access control (ABAC) and role-based access control (RBAC) [20].

ABAC In the ABAC systems, the authorization is based on attributes associated with the users' identity. A user is granted or denied access to resources based on the attributes that constitute his identity.

RBAC RBAC systems authorize the users based on roles they have. A role is a set of permissions. Administrators of resources can assign roles to the users, and the access to the resource is then based on checking the roles. Roles are a more scalable solution because administrators can handle the permissions collectively for large groups of users [20].

JSON Web Tokens "JWT is a compact, URL-safe means of representing claims to be transferred between two parties." [21]. The data inside a JWT are encoded as JSON and can be both signed for integrity (using JWS) verification and encrypted (using JWE) [21].

JSON Web Token (JWT) is commonly used for web authorization. Users usually obtain the JWT after successful authentication. After the authentication, the users include the JWT in their requests, and by providing the JWT, they prove that they are authorized to access specific resources.

DiD and Authorization The paradigm of issuing and verifying claims on which DiD is based is well suited for authorization. VCs can provide integrity and cryptographic verifiability, which are essential properties for building secure authorization schemes. Both ABAC and RBAC systems can be modeled using DiD. VC is essentially a set of attributes that are cryptographically signed. Disclosing the attributes, e.g., using selective disclosure and ZKPs, immediately provides a convenient way to bootstrap an ABAC system. On the other hand, new VCs that represent the user's role and are tied to his identity can be issued to create an RBAC system.

One of the proof formats that the W3C VC standard [10] describes is JWT. It describes all the necessary encoding rules for the VC data model to be represented as JWT. This provides a convenient way to use VCs for authorization in the current web infrastructure.

DiD based authorization can be used in various domains like:

- *public services/governance*: healthcare, elections, various licences (driving licence, building licence),
- *online services*: grating roles in IT systems, age-restricted content, location-restricted content,
- *traveling*: traffic stamps, flight tickets.

DiD authorization in its current form is not a silver bullet. Due to its considerable overhead, it would be an inconvenient choice for performance-oriented environments like operating systems.

3.3 Life-Long Credentials

The DIDs are persistent and should be interoperable and portable. They are a great foundation upon which a user can build his digital identity that would be everlasting.

The number of credentials a person will receive will accrue throughout his lifetime. The credentials include education diplomas, citizenship certificates, employment certificates, vaccination certificates, etc. In the DiD system, those credentials will be in the form of VCs and, as such, will be digitally signed by the issuing party and thus easily verifiable. Because it is expected that governments and organizations will create their decentralized identities and engage in the identity ecosystem, those VCs will be very valuable. They could be used throughout the society to prove various statements about their holder.

This model is very similar to the current analog credential reality. People hold credentials in various forms: physical cards or paper certificates. However, the analog credential counterparts are easier to forge than digital signatures and are usually hard to maintain throughout life.

Not all VCs must be everlasting. It is entirely possible to issue a VC with a specific validity period. Similarly, services like the current revocation registries of the CAs are expected to emerge for VCs.

Sociates are expected to adopt this new DiD standard. The European Union is building the European Blockchain Services Infrastructure (EBSI) [22] with the European Self-sovereign Identity Framework (ESSIF) as one of its central concepts. Endeavors in a similar direction can be seen in the Pan-Canadian Trust Framework.

3.3.1 The European Blockchain Services Infrastructure and DiD

In 2018 EU Member states signed a declaration creating the European Blockchain Partnership (EBP). The EBP then assisted the European Commission in establishing European Blockchain Services Infrastructure (EBSI) [22].

The European Commission operates nodes on a European level. The Member State's authorities mandated by the EBP Policy group operate the EBSI nodes at a national level—the system is permissioned. All the nodes can broadcast transactions to update the ledger [23].

EBSI sets multiple goals [22], the ones where DiD is involved are:

1. transition from paper to digital,
2. acceleration of creation of cross-border services for public administration,
3. simplified verification processes:
   - *citizenship*: easy and secure control and access to personal data across Europe,
   - *businesses*: simplified interaction with governments, reduced administrative costs,
   - *governments*: prevent fraud, increase transparency and make verification and data authenticity easier.

3.3.1.1 ESSIF

The framework will allow EU citizens to create, control, and use their digital identity while being GDPR compliant and aligned with eIDAS, so it can benefit from existing legal frameworks and support legal enforceability [24]. Apart from providing an identity solution, it also aspires to create definitions and guidelines for other technological components of a decentralized identity ecosystem like wallets, ledgers, trusted issuer registries, or APIs to access various services [24].

ESSIF will provide trusted digital identities that are secure and interoperable across the whole EU and give control to the citizens [22]. It aims to make interactions between individuals and institutions easier and faster. It also aims to facilitate and digitalize many processes among public administrations and private parties in the EU [22].

eIDAS, in its current state, covers mainly electronic identification, but it does not support cross-border exchanges of credentials like age verification, diplomas, or employment [8]. This is one of the problems that ESSIF aims to solve.

Roles Introduced by ESSIF Apart from using the classical roles from classical DiD systems, ESSIF introduces other important ones. For the ESSIF to be compliant with eIDAS and to support legal enforceability, it has introduced the notion of trusted parties, which issue specific types of credentials or, in some instances, oversee the onboarding of entities. ESSIF *Onboarding Service* as defined in [24] serves as a gatekeeper into the ESSIF ledger. It checks whether the given natural person has required authorization based on classical identification means to get the permission to onboard the system. The other important role defined in [24] is *Trusted Issuer*,

which refers to an organization that has the authorization to issue certain types of VCs, e.g., driving licenses. Lastly, [24] introduces *Trusted Accreditation Organization*, which is authorized to issue accreditation to other parties to issue specific types of VCs.

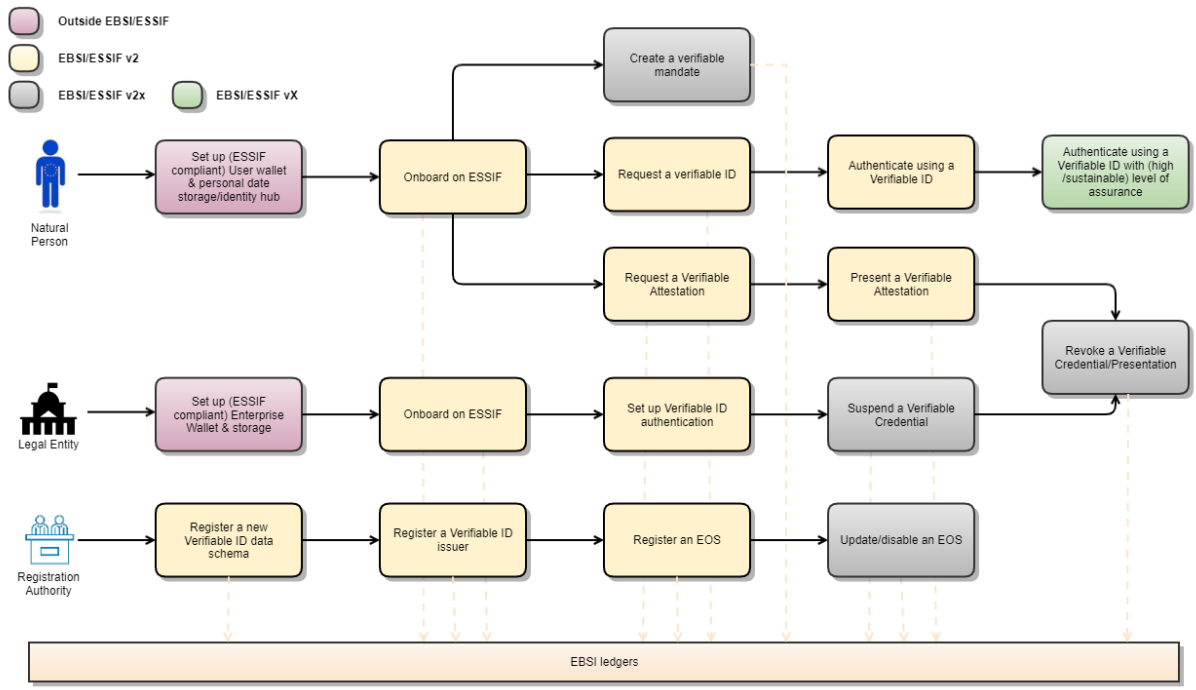

Figure 3.2 General ESSIF Usecase [24]

General ESSIF Usecase

- *Setup*: entity creates a DID and stores it on the blockchain. This step would be executed through some ESSIF-compliant wallet. After this step, the entity can start receiving and sharing VCs. [24]
- *Onboarding*: creating and registering a DID will not be enough for the entity to interact with governments and other authorities. For that, it is needed to obtain a verifiable id from a legal entity [24]. After the onboarding, the entity can start fully benefiting from the ecosystem because it can identify and authenticate itself to the legal entities [24]. It can request verifiable claims from legal entities, for example, a verifiable claim about a finished curriculum on a university.
- *Presenting attestation*: entity can provide a verifiable presentation of attestation (a form of a VC) to the requesting party, for example, proof of age.
- *Revocation*: legal entity can revoke a VC it issued to some entity after it is no longer valid. For example, an employer can revoke a VC confirming employment after a contract ends.

3.4 Supply Chains and DiD

When it comes to supplying goods, companies have to comply with many standards—social, technical, or environmental and, as such, need to prove compliance with them and control the

quality as a whole. Companies may be mandated to comply with Know Your Supplier (KYS) programs. Such programs are intended to ensure that human rights are not being violated (for example child labor), that the quality of certain products is ensured (for example health care), or that state's embargos are obeyed. Blockchains and DiD can together provide a solution for those problems in supply chain management (SCM). Some of the possible use cases are:

1. proofs of authenticity and quality,
2. transparency in the supply chain and origin tracking,
3. identification and traceability of faulty items.

Current Solutions for Item Identification The most common way to identify trade items is to assign them a Global Trade Item Number (GTIN). GTIN is a number that allows companies to uniquely identify their trade items[1] (TI) [25]. An international nonprofit organization— GS1— develops the GTIN. It is common for an item's identification number to be encoded using barcodes, or 2D barcodes, known as QR codes, which GS1 also standardizes. GS1 is starting to investigate the DiD ecosystem and is partnering with technology companies to leverage the VCs and DIDs to embed trusted data to the identifiers [26].

Proofs of authenticity and quality According to research performed by the World Health Organization, 1 in 10 medical products in low- and middle-income countries are either substandard or falsified [27]. Using the decentralized identity ecosystem and its integration with the supply chain, the customers could have the opportunity to verify the authenticity and quality of the products.

Trade items (TI) or batches of TI could be assigned a GTID, which would be based on DIDs. Such TI could receive VCs that could be used for providing any verifier with quality assurance, proof of authenticity, proof of standards compliance, etc. A network of certifiers could be built up, and individual companies could pay those certifiers for audits. In case of a successful audit, the certifier would issue a VC for the company or an individual TI, which the customers could verify.

The verification could involve scanning the GTID with a specialized mobile application and verifying the associated VCs. Using such system the customer would be able to conveniently check each supplier in the supply chain or the quality certifications of the product.

It is needed to ensure that the product identifiers will not be cloned. Technologies like RFID tags could be used. There are two possible approaches to this problem, as mentioned in [28]:

1. development of secure tag distribution schemes, so an attacker cannot copy the contents of the tag,
2. development of cloned tag detection systems.

3.5 Peer-To-Peer Communication over DIDComm

DIDComm is a message-based, asynchronous, transport-agnostic, and simplex communication protocol. DIDComm aims to provide secure and private communication channels built atop of DIDs." [29]

DIDComm represents a secure network overlay and provides a substrate upon which other protocols can run [30]. The protocol's core is an interaction between 2 entities identified by their DIDs. Alternatively, as [30] states, "DIDComm is a standard way to **interact with** an identity.". After exchanging the DIDs the two entities can look up the relevant communication endpoints

[1] "A trade item is a product or service priced, ordered or invoiced at any point in the supply chain." [25]

listed in the DID document and establish connections secured by the DID associated private and public keys.

Existing applications like WhatsApp or Signal already have good security and privacy properties, but they have a common problem. They are not interoperable, often do not provide the ability to integrate to other protocols (for example using Signal over email), and thus silo the users [31]. Additionally, the users are not the owners of the communication channels, but rather consumers [31]. Those are significant limitations that go against the principles of sovereignty and interoperability of the DiD.

DIDComm isn't just about the security, additional features according to [30] are:

- *transport-agnosticity*: can leverage different transport protocols like http(s) or WebSocket,
- *asynchronicity*: isn't request-response by nature,
- *peer-to-peer*: can operate without centralized servers,
- *offline*: can run just via bluetooth or wi-fi,
- *works without registration*: user's just have to exchange DIDs.

3.5.1 Messages Types and Security

The DIDComm messages usually have a layered format: plaintext, signed and encrypted [29]. The **plaintext** messages are JWMs with some set of shared attributes (time, from, to), as such, they lack the confidentiality and integrity guarantees [29]. The **signed** part adds another layer to the plaintext message in the form of a non-repudiable signature. However, the signature is optional and can be added only where needed. The signature can degenerate the protocol's security in specific scenarios [29]. Sharing a signed secret with a third party is a security risk because the third party can share the secret with anyone else [32]. That is problematic because signatures are non-repudiable. A better alternative might be to use authenticated encryption, where two parties share a secret, e.g., a symmetric key, so both parties can repudiate the shared messages [32]. The **encrypted** message is the message format that will be transported over the network. It provides confidentiality, integrity, and authenticated/anonymous encryption. For authenticated encryption, the encrypted form of JWM—the JWE—is used in the form of ECDH-1PU [29]. For anonymous encryption, i.e., the message's recipients will not authenticate the sender, the JWA of ECDH-ES should be used [29].

DIDComm and Man in the Middle The DIDComm messaging happens between two DIDs and is encrypted. The only weak point that can be susceptible to man-in-the-middle attack is the exchange of DIDs. The two communicating parties must know who is the controller of the DID they are communicating with. This authentication can happen via the exchange of VCs that bind the entity to the DID (similar to the current PKI model) or via some third party secure channel, e.g., using Signal.

3.5.2 Roles and Routing

There are three main roles in the protocol: sender, recipient, and mediator. The communication of two peers is usually simplex and is routed through a mediator. The mediator is responsible for forwarding messages from the sender to the recipient, and the sender models it [33]. For the sender to know where to send the message, he has to resolve the DID document of his peer based on the provided DID and lookup the relevant communication endpoints.

Users can have a self-hosted mediator. However, it is expected that mediator agents would be provided as a cloud service [33]. The mediator can be stateless (simple forwarding) and stateful

(e.g., retry delivery logic) [29]. The mediator receives messages of type *forward* on behalf of the recipient. The recipient is responsible for coordination with the mediator, e.g., to set its correct communication endpoint [29]. The sender can send decorated messages of type *forward* to the mediator. The mediator decrypts them (not the actual message, just the *forward* wrapper) and, based on the decorators, decides what to do with them next. Usually, he forwards them to the recipient based on the *to* parameter of the message [29]. The mediator can rewrap the message in a new forward message and thus provide a capability for dynamic routing [29].

As can be seen, the mediator does not have access to the contents of the transported messages. Recipients can create a new connection with different mediators for different peer-to-peer interactions. Also, DIDComm is based on DIDs, which can be easily generated for every new connection. The architecture of DIDComm ensures that the users are not dependent on any central authority and that they can communicate securely and anonymously.

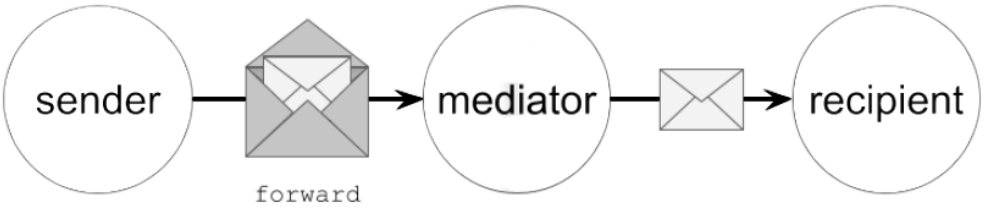

Figure 3.3 DIDComm Roles [29]

3.6 Healthcare

This section is concerned with use cases of DiD in healthcare. It discusses representing electronic health records (EHR) with tools provided by DiD. It also discusses the relationship between DiD and selected medical standards.

The healthcare industry uses many standards—for formatting EHR, medical terminology, or safe manipulation of health data. The DiD cannot replace all the standards or be integrated into all healthcare processes. However, the VCs are flexible enough to capture the semantics of many existing EHR types. The DiD can also provide an alternative way to store and exchange medical data in the more straightforward use cases.

3.6.1 Standards

Standards in healthcare are various, but they mainly involve terminology, content, and security [34]. The terminology standards involve the representation of medical concepts in unambiguous manners. Such standards standardize codes for diseases, symptoms, terminology for naming drugs or supplements [34]. The content standards involve the definition of structure and organization of medical documents [34]. And lastly, the security standards involve data collection, sharing, access, and encryption [34].

Terminology Verifiable credentials contain three main parts: metadata, claims, and proofs. The claims can be modeled so that they contain almost arbitrary data[2]. Therefore, the VCs do not limit terminology, and the DiD is fully compatible with the terminology standards.

[2]They are mainly limited by the amount of information they can contain, such that the VC can be still conveniently shared, e.g., using QR codes. However, they can contain links to external storage, where the primary payload might be stored.

Content The current W3C VC standard standardizes the VCs in JSON/JSON-LD formats. That alone causes interoperability issues. For example, common standards HL7[3] V2, HL7 V3, and HL7 CDA, used to exchange medical data between healthcare providers and patients, use XML. However, a newer standard by HL7 called FHIR supports both XML and JSON. FHIR is based on so-called resources. Resources have an identity—a URL—and contain structured data as described by the resource type [36]. Presenting clinical information modeled with FHIR using credentials very similar to VCs was already shown in [37].

Security The most prominent standards and regulations for dealing with healthcare (or personal) data are HIPAA in America in GDPR in Europe. Because the DiD is built around privacy, security, and user consent, it fits into the health ecosystem from a security standpoint. In DiD system user's data should be stored in his private wallet or encrypted via his private keys on cloud storage. VCs can also support proving claims in zero knowledge.

3.6.2 Healthcare Usecases

Interoperability Many of the current medical institutions do not have interoperable information systems. It may be due to the issuance of EHR in different formats, or the impossibility of seamless cross-institutional electronic communication, e.g., lack of standard APIs.

DiD is user-centric, and a user should have the majority of his data stored by himself, or the data should be easily accessible on his cloud storage. Therefore the problem of cross-institutional communication could be bypassed. Electronic information exchange between a patient and a doctor could become peer-to-peer. The incompatibility of data formats could be solved by following the W3C VC standard and using common VC schemas as discussed in Meaning of Fields of VCs.

The exchange of sensitive data would happen in the domain of patient-doctor. NFC and Bluetooth extensions of the DIDComm protocol are being developed, and therefore the whole exchange could happen offline. Alternatively, the exchange could happen via scanning a VC encoded as a QR code.

Preventing Falsification The healthcare industry still relies on the exchange of information on paper. In the case of prescriptions or recommendations for specialized health examinations, those documents are accompanied by a hand signature or a stamp, which are much easier to forge than a digital signature. Additionally, digital signatures provide non-repudiation and integrity. Thus the issuers—the doctors—can be held accountable in case of a policy violation. Because of integrity, the medical VCs can not be tampered with.

Permanent Health Record All patients can continuously build and maintain their health records by storing VCs that they received from doctors. Such health records could be analyzed by specialized algorithms and provide health suggestions. It would also provide valuable information to doctors and enable them to create more precise diagnoses. Families could share their health data and thus create a detailed case history.

Verification of Specialization With the usage of DiD, doctors would be able to display their education, attestations or attendance at conferences in the form of VCs. Doctors could also apply for verification of their attestations by some international organization, e.g., WHO. Such verified VCs would then have worldwide validity. That would enable patients to verify their competence easily, even when abroad. Additionally, such verification could be helpful in the private sector, where the government or hospital does not directly guarantee the attestation.

[3] "HL7 is a non-profit, ANSI-accredited organization that develops standards for the exchange, integration, sharing and retrieval of electronic health information." [35]

Chapter 4

Security and Privacy

This chapter covers the security and privacy aspects of the decentralized identity. It focuses on zero-knowledge proofs, authentication, key management, and correlation.

4.1 Zero-knowledge Proofs

This section provides an overview of zero-knowledge proofs from the standpoint of DiD. Basics of interactive, non-interactive proofs and their relation are introduced. The special signatures of VCs that allow for zero-knowledge proofs in the DiD are discussed.

Introduction "Zero-knowledge proof (ZKP) system is a system that has properties of being convincing and yielding nothing beyond the validity of the assertion." [38] It is equivalent to being told by a trusted third party that the assertion holds [38]. The usefulness of ZKP for identity management is evident as they provide the holders of ZKP-friendly VCs an ability to retain their private information in secret and reveal nothing beyond the validity of assertions they are trying to make.

Notion of a Proof In mathematics, the notion of proof is that it is static, i.e., some fixed number of statements derived from previous statements or axioms so that this derivation process is self-evident or commonly agreed to [38]. Zero-knowledge proofs (ZKP) can be both dynamic and static. There are two major types of ZKP: interactive (dynamic) and non-interactive (static).

Properties of Proofs There are two fundamental entities in a proof system. A prover who provides the proof and a verifier who does a verification process. Proof systems have two basic properties: completeness and soundness. [39] defines those properties as:

▶ **Definition 4.1.** *A **proof system for membership** in language L is a verification algorithm V: ∀ x the following properties hold:*

- *completeness: if $x \in L$, then ∃ a proof p, such that V(x, p) = ACCEPT,*
- *soundness: if $x \notin L$, then ∀ proofs p holds V(x, proof) = REJECT.*

The soundness property relates to the verifier and the fact that he cannot be tricked into accepting false statements. The completeness property relates to the prover's ability to persuade the verifier of true statements [38].

4.1.1 Interactive Proof Systems

In the previous definition, a verification procedure was introduced. It verifies proofs from outside, e.g., from a prover [38]. In the interactive proof systems, this verification procedure may include *interaction with the outside* as long as it happens in probabilistic polynomial time [38]. In such a case, the verification procedure yields a probabilistic result bounded by some error probability, which can be decreased by repeating the procedure [38]. The verification procedure V and the prover interact for k rounds, and the messages sent by V depend on his polynomially long random string [39]. At the end of the interaction, V checks the received message and decides whether to accept or reject the proof [39]. [39] defines an interactive proof system as:

▶ **Definition 4.2.** *An **interactive proof system** for language L is a probabilistic-polynomial time algorithm V and a function P such that* $\forall$ *x*

- *completeness: if* $x \in L$*, then verifier accepts with probability:* $Pr[(P,V) \text{ accepts } x] >= \frac{1}{2} + \frac{1}{poly(|x|)}$,
- *soundness: if* $x \notin L$*,* $\forall$ P^* *the verifier accepts with probability:*[1] $Pr[(P^*,V) \text{ accepts } x] <= \frac{1}{2} - \frac{1}{poly(|x|)}$.

And finally, a definition of the zero-knowledge system, also from [39]:

▶ **Definition 4.3.** *An interactive proof (P, V) for L is (honest-verifier) **zero-knowledge** if* $\exists$ *probabilistic-polynomial turing machine S (a simulator) such that* $\forall$ $x \in L$ $S(x) \sim (P,V)(x)$.

S(x) $\sim$ (P,V)(x) means that the distribution of the simulator is the same as the distribution of the view of the verifier, i.e., the verifier's random string and potential messages of the prover have the same probability in the simulation, and real interaction [39].

The definition can be extended for *perfect* zero-knowledge. It assumes an arbitrary verifier and requires that for each such verifier, a PPT S exists so that the distributions are the same.

Role of the Simulator The simulator gets no special knowledge at all. It resides in an alternative world that is secure by definition [40]. "Nonetheless, it must be able to convince every verifier into believing that the statement is true while producing a transcript that's statistically identical to the output of a real prover." [41] Because the simulator doesn't have any special knowledge, the verifier can't extract any information by interacting with it. If the existence of a statistically identical simulator for a given protocol is shown, then such protocol is zero-knowledge.

Randomness in IP The fact that interactive proofs are more powerful than NP-proof systems is based on them being rooted in randomization, i.e., the fact that the verification procedure is randomized [42]. Without the randomization, the interaction could be determined beforehand (without the interaction occurring) and transcribed deterministically [42]. As such, it would have the same power as the static mathematical proofs discussed at the beginning of this section. If the verifier behaves deterministically, then the prover can provide its answers to the verifier's predictable questions [42]. The verifier would then just check that the answers are convincing [42].

4.1.2 Transformation to Non-interactive Proofs

Problems with IP Although very powerful, the interactive ZKPs are not very practical. They require much back and forth messaging, and both parties to be online (or unbounded

[1] It is to be emphasized that the definition allows for an arbitrary prover P^*, even a malicious one.

delays may arise if they are not) [43]. As such, the interactive ZKPs are not used much in applied cryptography [43]. Techniques for minimizing the interaction (or even getting rid of it entirely) exist. One such technique is called the Fiat-Shamir heuristic (FSh)

Fiat-Shamir Transform FSh is a simple template that can be used to design non-interactive argument schemes [44]. [44] describes it as follows:

1. Let *pi* be an interactive proof, where the verifier's messages are based on his random string, and let x be an input from the prover.
2. Compile *pi*:
   a. The verifier sends a description of a hash function h[2].
   b. The prover responds with a transcript of an emulated execution of *pi*, where:
      - input x is included,
      - all messages between the prover and verifier are included,
      - each message of the emulated verifier is the value of h **applied to the transcript so far**.
   c. The verifier checks that the received transcript is consistent with the outputs of the hash function h.

To illustrate the technique, a conversion of a simplified interactive Schnorr identification protocol to the Schnorr signature—which is non-interactive—is provided in 4.1. Description of the identification algorithm, which is taken from [43] is provided below:

1. Alice wants to prove that she knows x in $Y = g^x \mod p$.
2. Alice commits to a random value $R = g^k$ and sends it to Bob.
3. Bob responds with a random challenge.
4. Alice computes $s = k + c * k$.
5. Bob verifies that $g^s = Y^c * R$.

The protocol can be made non-interactive by allowing Alice to supply the randomness by herself using a hash function[3].

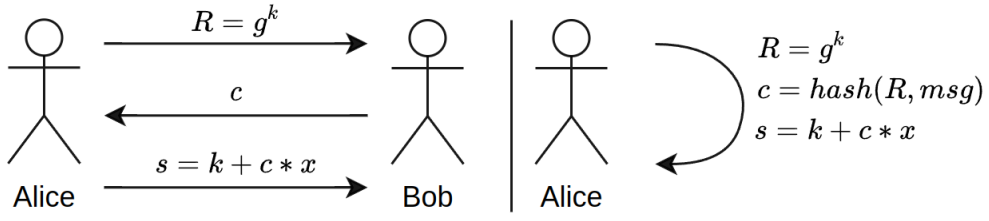

Figure 4.1 Converting Interactive Protocol to Non-Interactive

[2]The hash function h can be chosen ahead of time to remove the interactive step.
[3]In 4.1 the hash can contain a msg. Therefore Alice can both prove that she knows some secret and commit to a message, i.e., she can create a signature [43].

4.1.3 ZKP in DiD

To use ZKPs with VCs, the issuer has to issue the VC in a special format, in particular, the VC has to be signed with a special ZKP-friendly signature. A holder of a VC that has such a special signature can prove the validity of the signature without revealing the signed values and even the signature itself. The DID ecosystem converged on using two types of such special signatures:

1. Camenisch-Lysyanskaya Signature (CL signatures) [45]
2. BBS+ signatures [46]

The CL signatures are based on the Strong RSA assumption [45]. The CL signatures are used in a VC scheme called Anonymous Credentials. On the other hand, the BBS+ signatures are pairings-based. Company Mattr created VCs based on linked-data proofs and the BBS+ signatures.

Table 4.1 Comparisson of ZKP VCs [47]

Feature	ZKP-CL	BBS+
Selective disclosure	yes	yes
Signature binding	yes	yes
Private holder binding	yes	yes
Predicate proofs, e.g., >= 21	yes	no
Compound proofs (combine VCs)	yes	yes
Small & efficient	no	yes
Compatible with LD sigs	no	yes
W3C standard compliant	no	yes

Anonymous Credentials vs LD-proofs ZKP with BBS+ signatures Both schemes have advantages and disadvantages, as shown in the table: 4.1. However, a leading expert on ZKP in DiD Brent Zundel, argues in [48] that the community should converge on the LD-proofs with BBS+ signatures. He states that such credentials have two main benefits:

1. possibility to have a rich and hierarchical set of attributes
2. breaking the dependency on a distributed ledger, as the MATTR's implementation of the scheme discarded the credential schema [4]

BBS+ BBS signatures utilize pairing-friendly elliptic curves. This property of the EC allows for the construction of signatures with the following properties:

1. *aggregation*: signing multiple messages while producing one output [50] (it allows for selective disclosure),
2. *ZKP disclosure.*

Those signatures also allow for generating proofs of knowledge of the signature itself [50]. Using this property allows generating a unique proof for each verifier a user interacts with. This is very useful because if the same signature is shown to multiple different verifiers, it could be used as a correlating identifier, and the user's privacy could be compromised.

In [50], a signature proof of knowledge-generating algorithm that creates a zero-knowledge proof of knowledge of a signature is described. In the algorithm, the Fiat-Shamir heuristic for creating a NIZK is used.

[4] All the required data is embedded into the VC itself using Linked Data Contexts [49]. In the case of Anonymous Credentials the data is stored on a ledger.

4.2 Mitigating Password Database Breaches Using DiD

This section discusses password database breaches and how DiD can help to mitigate them. Preventing database breaches is a direct consequence of DiD authentication, presented in 3.1. This section is solely concerned with the security aspects and the security implications.

Authentication on the Web The majority of web services use the combination of a username and a password for authenticating their users. A user is represented by his username. By presenting the knowledge of the password that corresponds to the username he proves his identity and is thus authenticated.

However, such an approach is not optimal for two reasons —user experience (UX) and security. Because most web services are run independently, users have to create a new pair of username and password for each of them.

Some users use password managers, but the password managers have to be synchronized across multiple devices, which might be harder to do, like in the case of KeePassXC or managed by some trusted third party. Another problem with password managers is that the autofill feature sometimes does not work. This can be caused by the website that actively blocks modifications on the page. Alternatively, the login might be coded in some non-standard way that is not recognized by the manager [51].

Companies like Google, Facebook, or Microsoft offer an alternative solution to managing passwords. They provide solutions for federated identity, which is another option for users to manage their digital identities and passwords. However, such an approach increases the amount of personal information the corporations have, making the users more dependent on them.

The passwords on the web can be stored safely, and the impact of attacks on databases can be minimized, but the techniques for securing them are not trivial. Passwords should be stored in a hashed form, ideally hashed with salt to prevent attacks like the rainbow table attack. Also, the inputs to the databases should be well sanitized and escaped, and principles of least privilege should be employed.

Because of the mentioned inconveniences, many users opt to use low entropy passwords and reuse those passwords across multiple services, resulting in severe security threats. Apart from that, it is not easy for service providers to securely store the credentials in databases. Consequently, the primary authentication technique on the web is inherently insecure.

How DiD Can Help As explained in 3.1, by using DiD for authentication, a user acts as his identity provider. In DiD all users are identified by their DIDs, and the users themself are the only ones who can prove the ownership of their DID.

By incorporating DiD authentication, the web service can relay the authentication to the users themselves. Therefore the web services do not have to store any passwords. Instead, a web service can provide an option for DiD authentication.

The security of such an authentication scheme would depend on the wallet security that the users use for managing their DiD, because the wallet would be responsible for proving the DID ownership. This is a much better alternative to the current scheme because a security compromise of one user's wallet does not influence the security of other users. Such a decentralized authentication scheme would prevent global password database breaches.

4.3 Key Management

One of the challenges in decentralized systems based on asymmetric cryptography is key management. The private keys that correspond to DIDs are the central points to his digital identity. It is through the private keys that users control their identity. Key compromise or key loss can

have dire consequences for the user. This section serves as an introduction to key management in DiD.

4.3.1 Rotation and Revocation

Key rotation and revocation are important defensive security techniques that help lower the probability of a key compromise and lower the impact of the potential key compromise.

Key Rotation Key rotation is useful mainly for two reasons:

1. to provide an option to rotate keys and use a more secure form of encryption in case of technology advances [12].
2. to mitigate passive attacks (an attacker is for example eavesdropping and the user does not know it) [12].

Additionally, rotation lowers the value of compromising a single key and thus lowers the probability of an attack [5]. The keys connected to a DID can be changed via an UPDATE method defined by the corresponding DID method. The keys in the private connections have different rules for rotation and are described in 4.4.2. It is important to note that the key that signs the UPDATE transaction can be different from the key corresponding to the key that is rotated. If this would not be the case, then an attacker could also perform this operation.

Rotating a key does not invalidate the proofs made before the rotation [5]. However, to validate such a signature, a historical value of a DID document might be necessary as not all the DID documents have a key log directly built-in.

Key Revocation Key revocation deactivates a selected key pair, i.e., any new proofs generated by the private key should not be considered valid [5]. It is expected to be used in case of a key compromise. Revocation signals others not to trust and use the associated public key. Revocation is not retroactive. It is only concerned with the future use of the keys [5]. However, several precautions have to be made for such a technique to be safe. The state of the DID document at the time of making a cryptographic assertion has to be known [5]. Additionally, the state should be reflected inside the VC. Otherwise, an attacker could use a compromised key pair and manipulate the date of making the assertion such that it would seem that it happened before the revocation.

4.3.2 Recovery Methods

Losing keys in a mature SSI wallet can have severe consequences for a user as it can prohibit him from acting in the digital world (and by extension also in the real world) for months [8]. The keys in DiD ecosystem are typically stored in a mobile wallet. Key recovery might be necessary for multiple scenarios—the wallet gets either stolen, lost, or corrupted [8]. If such an event happens and no key recovery method is established, all the user's VCs have to be revoked (if possible) and reissued. All the connections a user made have to be reestablished. On the other hand, if the wallet used a periodical backup and the user has key recovery method, his digital identity can be recreated. This section covers multiple key recovery methods that can help users prevent losing their digital identity.

Mnemonic Phrase Schemes One of the most used recovery methods is a so-called mnemonic. Mnemonic is a sequence of easy-to-remember words that can be used for generating or recovering wallets [52]. It was initially proposed as a Bitcoin Improvement Proposal BIP39. It provides a more human-friendly way to capture the value of the seed that is used for generating the wallets.

Using mnemonics is a de facto standard in the whole cryptocurrency world. It is used in Bitcoin, Ethereum, Cardano, and many other protocols.

It works by first generating a random number of a specific length. This random number is then split into groups of a length of 11 bits. Part of the algorithm is a carefully crafted wordlist. The wordlist has a length of 2048 words. The 11-bit groups are individually converted to numbers, and those numbers are used for indexing the words in the wordlist. The mnemonic usually has either 12 or 24 words. Finally, the mnemonic is converted to seed using a key derivation function. The BIP39 proposes the usage of PBKDF2. Apart from supplying the mnemonic to the key derivation function, a user can also supply a passphrase. The passphrase is used as a security measure if the mnemonic is compromised. The seed can be used for generating wallets.

However, the mnemonic can be unwieldy to use as it requires reproducing all the secret words in their exact form and the exact order [53].

Secret Sharing Schemes and Social Recovery Secret sharing schemes are based on splitting a piece of secret information into multiple parts (shards) and distributing those parts amongst multiple entities (or locations) to eliminate a single point of failure. The secret can later be reconstructed by taking the shards back together (or a subset of them). The most famous technique for splitting a secret is Shamir Secret Sharing (SSS). The secret sharing techniques are helpful because they allow users to split a secret (for example, a mnemonic) so that if some of the involved parties get compromised, the key can still be reconstructed. At the same time, the attacker gets no information about the original secret. Secret sharing can be incorporated into the DiD wallets, and users can share their shards with trusted parties (DIDs) over end-to-end encrypted channels, e.g., via DIDComm.

The drawback is that when a secret is reconstructed, it can get compromised, at which point the whole system collapses. Apart from that, the trusted parties can collude and reconstruct the original secret without the owner of the secret information. Additionally, the shards are long and not human-friendly.

4.4 Privacy

Privacy is firmly connected to the concept of DiD. Privacy can have various forms. In this thesis, privacy is regarded as an individual's ability to decide by himself when and with whom to share his personal information. There are two sides to the DiD—technical and ideological. The technical aspects described in this work cannot by themself solve the problem of privacy on the internet. In certain aspects, they can make it even worse. To arrive at more private internet, the ideological aspects have to be also adopted, both by the users on one side and governments and service providers on the other. In a model in which users need to exchange their personal information to be granted access to some resource or service, their privacy will eventually get violated, even if techniques from DiD are used. This section discusses the problems mentioned in this paragraph and the privacy aspects and tools of DiD more generally. It discusses both the benefits and also the possible drawbacks.

4.4.1 Verifiable Credentials and Privacy

VCs and Personal Information VCs can (and often will) contain personally identifying information (PII). Sharing such credentials can lead to de-anonymization and loss of privacy [10]. That is more so amplified by the fact that the credentials are verifiable and thus give the verifiers great assurance about the authenticity of the presented data. This aspect, therefore, can have a very negative impact on user privacy. This problem can be alleviated in several ways:

1. Issuers minimize PII in VCs to a bare minimum to proceed with the task for which the VC is being issued for.

2. Issuers use predicates instead of concrete values, e.g., using a predicate *subject's bank balance is greater than x* rather than revealing the exact balance.

3. Issuers create multiple simple VCs instead of one VC that contains a bundle of information, i.e., the data is separated into multiple VCs (only when ZKP signature schemes cannot be used [10].

4. Holders use selective disclosure with ZKPs to reveal only specific fields of the VC instead the whole VC.

5. Holders verify the verifier, e.g., they share VCs only with verifiers that they verified and that they trust [10].

6. Wallets notify holders when they share strongly identifying information and ask for confirmation.

Identifier and Signature-Based Correlation Holders of VCs are identified by the *id* (DID) field in the *credentialSubject* property of the credential, reusing this DID across many different VCs can lead to correlation [10]. If a holder shares one field with one verifier and a different field with another verifier, those verifiers can collude and de-anonymize the holder [10]. For example, suppose a holder shares a postal code with one verifier and a birthdate with another. Those verifiers can cooperate, and by sharing the obtained data, they can de-anonymize the holder.

Similarly to identifier-based correlation, holders can be correlated using the signature proof, or related metadata [10]. The majority of the values in the proof field can be used for correlation, be it the date of creation of the proof, the proof value itself, or the verification method.

Again, there are some techniques to alleviate those problems:

1. Issuers periodically reissue the credentials.

2. Holders generated new DIDs for each connection, where it is possible.

3. Holders create new ad-hoc credentials using ZKP for presenting proofs.

Correlation is a very powerful technique to de-anonymize holders and violate their privacy. One of the most powerful techniques to prevent correlation is the usage of ZKPs. ZKPs can help to avoid correlation in multiple ways:

1. Masking holder's DID—a holder can prove in zero-knowledge that he owns the DID to which a VC was issued [54].

2. Selective disclosure—a holder can choose only specific fields of a VC to be disclosed.

3. Creating proofs of knowledge of signatures of VCs and not revealing the actual signatures themselves [54].

It is up to the whole ecosystem to converge to ZKP credentials. If issuers use non-ZKP signature schemes, then the holders cannot use those techniques to protect themselfs.

Bearer Credentials Bearer credentials are special VCs that do not contain the *id* field in the *credentialSubject* property [10]. Such credentials entitle the holder to access specific resources by having the credential, i.e., being the bearer [10]. Bearer credentials are used in low-risk situations, where sharing it should not result in any losses to the issuer. Such credentials would, for example, represent cinema tickets [10]. Bearer credentials should be used anywhere, where possible, to enhance the privacy of their holders.

Revocation Checks It often is necessary for issuers to incorporate an option for VC revocation. Revocation can be implemented in multiple ways: using centralized servers operated by the issuer, publishing revocation metadata to a distributed ledger, or directly embedding the revocation details into a VC, e.g., using timestamps.

During the verification process, the verifiers check whether a presented VC was revoked or not. However, such checks can compromise the privacy of the holder. If, for example, the verifier directly contacts the issuer, then the issuer obtains the information that the specific holder is trying to use the VC. On the other hand, if the revocation data is published on a ledger (and the data is anonymized correctly) or in a publicly available revocation list, then a verifier can query the ledger or the list. Therefore the issuer obtains no information that a holder uses the issued VC.

This issue again ties to the ideology of the DiD. For the ecosystem to be privacy-preserving, the principles must be adopted both by the holders and the issuers.

4.4.2 Did:peer Method

The *peer:did* method is focused on establishing and retaining private and secure peer-to-peer connections. It is a suitable DID method for peer-to-peer communication via the DIDComm protocol [29]. It is based on the premise that if two peers want to interact, they should be the only ones who should care [55]. To enhance the scalability and privacy of this method, the DID document associated with *peer:did* is not anchored on any ledger (that might also be a benefit from a regulatory standpoint as no PII is anchored). Therefore such DIDs are not publicly discoverable, and the connections are visible only to the communicating parties.

The *did:peer* method can be seen as complementary to the ledger-based methods. While the ledger-based methods are convenient for the *issuer/holder/verifier* paradigm, the *did:peer*, on the other hand, provides an option for lightweight and scalable private connections. The ledger-based methods bypass the need to connect to the issuer (to retrieve a public key) and allow the issuer to update his keys easily. This is not the case for *did:peer*, where the main objective is not to exchange VCs but to directly connect with other peers.

Connecting Peers To establish a peer-to-peer connection, the peers first have to exchange their DID documents. For the exchange, the Hyperledger community developed two RFCs—DID exchange protocol and Out-of-Band protocol to connect two agents [56], [57]. For that to work, the peers have to have a secure preestablished connection, e.g., in-person exchange (Bluetooth, NFC), TLS, or Signal.

Using *did:peer* is a useful anti-correlation method. Peers can generate a new peer DID for each new connection. Because they are identified in each connection by a different DID, it lowers the probability of correlation [5]. This technique could be used to establish connections with all online services. That would greatly enhance the anonymity of users but at the expense of a lack of personalization.

Updating Connection Parameters The *did:peer* method has a very simple DID document. Inside the DID document are the peer's public key and a privilege model. The model prescribes rules for changing the DID document—what and by whom can be changed. The rules are mainly concerned with key rotation, i.e., who and under what conditions can change the keys. Each peer enforces the rules for the another peer because it is in the best interest of each of the peers to keep to connection secure.

Chapter 5
Case Study

The COVID-19 pandemic brought up the issue of how to issue, verify and hold medical information about individuals in a decentralized fashion. There was[1] a need for individuals to present their health status to various authorities. The information that individuals need to share in the context of the COVID-19 pandemic can be split into three main categories: testing credentials, vaccination credentials, and antibodies credentials. This chapter provides a case study that focuses on vaccination credentials. However, the concepts discussed are generally applicable to a broader spectrum of health credentials.

This chapter provides an overview of how W3C verifiable credentials and DIDs can be used to issue COVID-19 medical credentials. Additionally, it showcases how the credentials can be issued and verified and how a holder can store them. Finally, a comparison of the current approaches to COVID-19 credentials is made.

5.1 Design of Vaccination Credentials

This section provides an overview of possible approaches to COVID-19 VCs. Firstly the design of the VCs is explained. Secondly, operations atop those credentials (issuance, holding, verification) using Hyperledger Aries Cloud Agent are shown. Two types of vaccination credentials are introduced. The first type is a FHIR compatible VC based on SMART Health Cards [58]. The second is a VC based on the W3C vaccination vocabulary [59].

5.1.1 W3C VC with FHIR

W3C verifiable credentials can be modeled with FHIR. This section presents the syntax of such credentials and how they can be issued, verified, and incorporated into the DiD ecosystem.

5.1.1.1 FHIR

FHIR is a standard from the healthcare industry concerned with medical information exchange, healthcare terminologies, and secure electronic health records exchange [60]. All major vendors like Apple, Microsoft, or Google use it [61], [62]. FHIR operates atop FHIR resources. Resources are packets of information that can be combined into FHIR bundles, and thus semantic richness can be reached. Resources can represent individual health or administrative data elements. The DiD ecosystem can interoperate with FHIR in two ways. Firstly it can leverage FHIR's

[1] In 2022, this problem still holds. Moreover, it can be expected that a similar need might arise again in the future.

rich terminologies and semantics to express various health credentials. Secondly, health wallets can use FHIR APIs to communicate with FHIR-enabled issuers and mediate the exchange and issuance of VCs [58].

5.1.1.2 Design of W3C FHIR VCs

The groundwork for FHIR-based vaccine credentials was layed out in SMART Health Cards (SHC) [58]. SHC is becoming widely adopted. They are recognized in many countries like New York or California [63]. SHC has similar design principles as W3C VCs—data minimization, privacy, and cryptographic verifiability [58]. SHC are encoded as JWS and are shared as QR codes [58]. The most important component that SHC lack are DIDs. Individual vaccine recipients are not identified by their DID but by their real names. That has several disadvantages—vaccine recipients cannot link their SHC to their other VCs issued with their DID. Additionally, SHC are not syntactically compatible with the W3C VCs and thus cannot be easily used with the tooling used in DiD.

This section shows how the SHC can be issued and verified in the DiD ecosystem. W3C VCs have various flavors. Here the JSON-LD credentials are used. For the conversion of SHC to the W3C VC 3 steps must be taken:

1. Modify the syntax to comply with the W3C standard.
2. Provide a context for the JSON-LD credentials.
3. Use identification based on DIDs.

To conform with the syntax the terms *issuer* and *id* of a *credentialSubject* need to be added. Next, the *type* property needs to be added. Without it the credential will not be verifiable [10]. Because JSON-LD credentials are used, the context of individual terms of the credentials needs to be addressed. Terms outside the *fhirBundle* are mapped to the context of W3C VC. Terms inside the *fhirBundle* are mapped to the context provided by [58].

Last step is to add DIDs to the *issuer* and the *credentialSubject* fields. For the issuer of the vaccination credential the *did:key* method was chosen. There are two reasons for that—*did:key* function independently of any ledger and it supports ZKP. The resulting (shortened) credential is shown in 5.1.

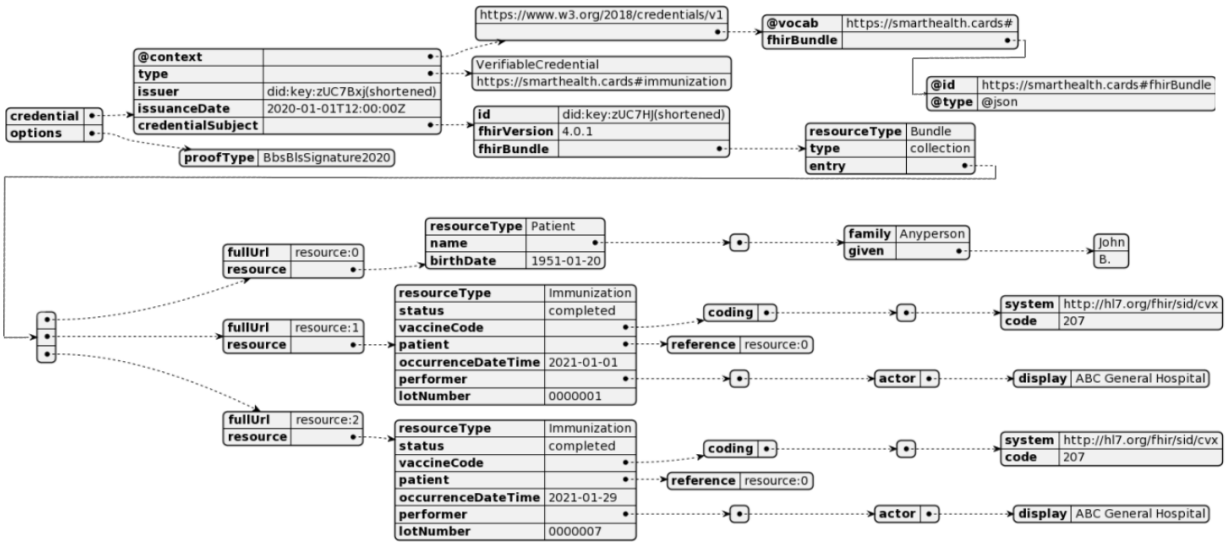

Figure 5.1 Structure of FHIR VC

5.1.2 W3C VC with W3C Vaccination Vocabulary

Another approach for designing a vaccination credential is to use a W3C vaccination vocabulary [59]. For the event of vaccination, it has the same level of semantic expressiveness as FHIR-based VC, but it uses a different syntax. Compared to FHIR, it lacks in terms of adoption. It is mentioned here mainly because it is well suited to be combined with the technique of selective disclosure and data minimalization.

5.2 Implementation

The syntactical structure and the design choices of the VCs were explained. However, for the VC templates to become useful, they must be used for issuing, holding, and verification. The following sections show how a vaccination credential can be issued (with actual data and accompanied by a cryptographic signature), held (inside a wallet), and verified (both regular verification and ZKP verification).

5.2.1 Technology Stack

The whole case study is built atop the Hyperledger technology stack. The hyperledger DiD community is closely aligned with the values expressed in the previous sections and also adheres to the W3C specifications and standards. The Hyperledger ecosystem supports almost all the functionalities mentioned throughout this work - VCs, DIDs (peer and ledger-based), DIDComm, ZKPs, ledgers, wallets, or agents. The tooling is deployed under an open-source Apache license.

Hyperledger Foundation Hyperledger Foundation is a non-profit organization developing open-source enterprise blockchains and accompanying infrastructure. The blockchain solution chosen for this case study is Hyperledger Indy, which was purposefully built for the management and development of DiD. Indy comes with a custom ledger implementation and an SDK for ledger interactions, wallet management, etc. Hyperledger also develops a cryptographic library, Hyperledger Ursa, that is interoperable across different Hyperledger projects and used in Indy. The last relevant Hyperledger project is Hyperledger Aries, which provides protocols for p2p interactions, VC management, and overall, a set of protocols for interactions in DiD.

Hyperledger Indy Indy is based on Indy Node, Indy SDK, and Hyperledger Ursa. Node is an extension of Indy Plenum. Plenum is a distributed ledger based on the RBFT consensus protocol [64]. The ledger is permissioned and has multiple running implementations like Sovrin MainNet or Indicio MainNet.

Indy SDK constitutes mainly of libindy and libindy wrappers. It additionally provides Indy CLI and other periphery libraries. Through the libraries, Indy supports ledger interactions and DiD operations. The DiD operations include: anchoring and resolution of DIDs (which are based on the *did:sov* method), VCs, did:peer, or ZKPs [65]. Overall the operations are built with anti-correlation in-mind [65].

Hyperledger Aries Hyperledger Aries is built mainly for p2p interactions. It provides protocols for p2p communication, VC exchange, verification, and key management protocols. Additionally, Aries provides agent frameworks that use the mentioned protocols and are operated by users. One such framework is Hyperledger Cloud Agent Python, used in this case study.

Hyperledger Aries Cloud Agent - Python Aries Cloud Agent - Python (ACA-py) is a server application designed to be run in the cloud. It utilizes the Aries protocols and enables the development of DiD services with any language supporting HTTP requests [49].

[49] lists functionalities of ACA-py, the most important ones are:

1. issue, hold and verify VCs (both AnonCreds and W3C VCs),
2. support for various DID methods: did:sov, did:key, did:peer,
3. mediator service,
4. DIDComm transport,
5. secure storage using indy-wallet and Aries Askar.

To use ACA-py, a controller that communicates with the ACA-py server over HTTP has to be developed [49]. The controllers can express arbitrary business logic. According to [49], possible applications are an authentication service based on VCs, p2p messaging, or service for issuing, holding and verifying VCs.

An API key or JWT can be used to secure the communication with ACA-py. ACA-py can host multiple wallets for multiple users at once. In such cases, usage of JWTs is necessary [49].

5.2.2 Alternative Technologies & Interoperability

Hyperledger provides an advanced DiD ecosystem built on the DiD principles. However, it has some drawbacks. The Indy ledger is a permissioned ledger, which might be concerning when building a truly open DiD system. Additionally, the existing ledgers based on Indy have a low level of decentralization. Sovrin MainNet currently runs with 17 nodes [66]. Apart from that, if the problem of decentralization was to be fixed, the existing BFT protocols do not scale well with an increasing number of nodes [67].

As explained in Blockchain Interoperability, different ledger-based identity solutions should be interoperable via resolvers. Additionally, it is one of the building blocks of DiD that users should be able to migrate freely between systems. Thus choosing Hyperledger should not be decisive for users as they can migrate in case of a better alternative.

Also, a lot of DiD functionality works off-ledger (did:peer, did:key, mediation, etc.), which is well supported by Hyperledger. ACA-py can provide better interoperability options by adding support for more did:methods.

[9] provides a list with over 100 did:methods in development, most of which are based on distributed ledgers. Apart from Hyperledger, there are other popular projects building the DiD ecosystem. Such projects include ESSIF, mentioned in 3.3.1, Sidetree and did:ion method developed by the Identity foundation, Mattr Global, which provides new ZKP solutions or Veramo that supports Ethereum DIDs and many others.

5.3 Issuing, Holding, and Verification of the Vaccination Credentials

This section describes how the proposed COVID19 vaccination credentials can be used. Issuing, holding, and verification are shown. Verification is done using both the non-ZKP and ZKP methods. For handling the VC operations, ACA-py is used. For the case study, two simple controllers were implemented. One represents a doctor performing the vaccination. The second one represents a patient receiving the vaccination. To showcase that an issuer can act as a verifier simultaneously, the doctor controller can also initiate a verification of the credentials. The controllers are based on a DEMO controller developed in [49]. The architecture of the case study is shown in 5.2.

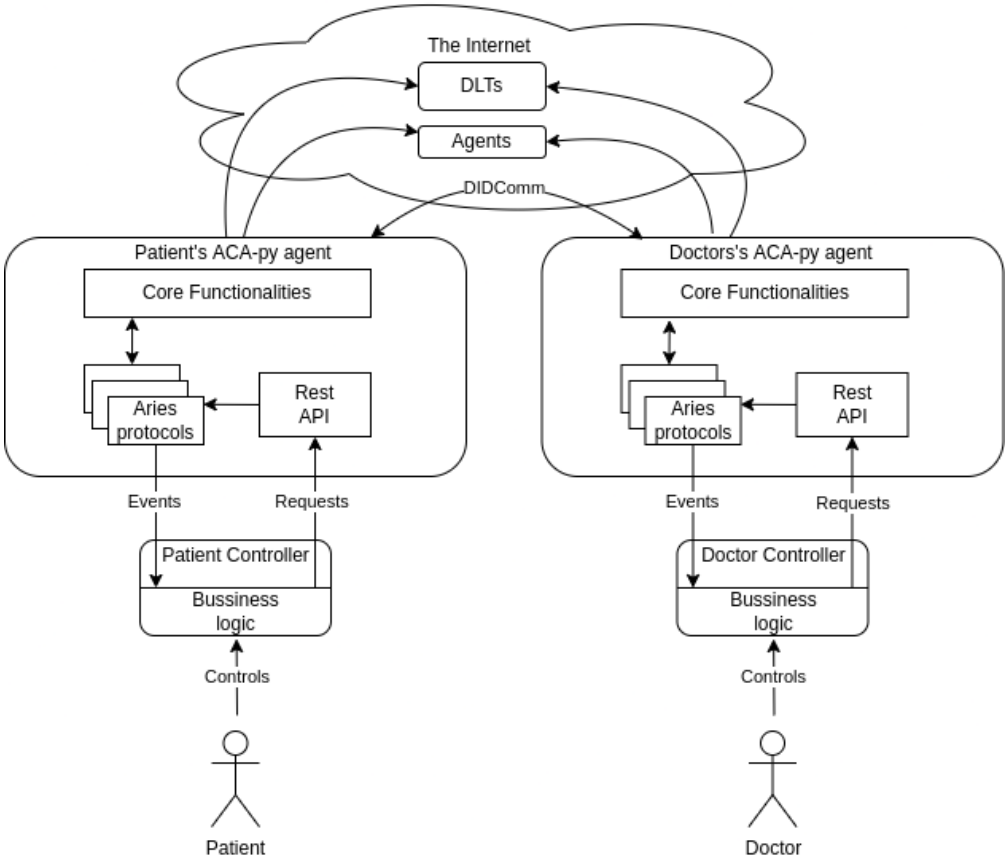

Figure 5.2 Architecture of the case study [49]

5.3.1 Controllers for ACA-py

For simplicity, the doctor controller will be referred to as *DC* and the patient controller as *PC*. ACA-py fits into the cloud layer as showcased in 2.2. The controllers, on the other hand, fit into the edge layer. The controllers are directly operated by users and, on their behalf, connect and send requests to the ACA-py REST API. ACA-py is an agent that acts on behalf of the users. In this case study, ACA-py is responsible for establishing connections with other agents and for VC management.

The doctor and the patient have their unique running instance of ACA-py. ACA-py has support for multitenancy, i.e., it can act as an agent for multiple users at once. However, this functionality is more common when the users are from the same domain, e.g., all doctors from one hospital could use one ACA-py instance.

The controllers are written in Python and can be controlled from CLI. **DC's** CLI provides the following functionalities:

1. *Issue Vaccination Credential*: issues a vaccination credential with predefined values to *PC*.
2. *Send Proof Request (with ZKP)*: sends a ZKP proof request to *PC*.
3. *Send Proof Request (without ZKP)*: sends a non-ZKP proof request to *PC*.
4. *Send Message*: sends a message to the other peer.
5. *Create New Invitation*: creates a new invitation to create a new connection.
6. *Get proof value*: fetches the last proof value.

PC has a more limited set of functionalities:

1. *Send Message*: sends a message to the other peer.
2. *Input New Invitation*: accepts an invitation for a connection.

Both the ACA-py instances are configured such that most interactions are automated. In normal circumstances, all the operations that affect users' privacy would be set in a way that would require users' consent. For example, when *DC* requests proof of vaccination, the proof is automatically provided by the patient's agent. Usually, this would require consent from the controller administered by the patient.

How the controllers work The controllers use an event loop and an *Agent class* provided in DEMO in [49]. The event loop runs until a terminate signal is provided by the user. The controllers read the user input from CLI and make a corresponding request to the ACA-py server based on the input. The *Agent class* represents the ACA-py server and provides an interface for sending requests to the server. The VCs and proof requests are hardcoded inside the code because both of the requests require a lot of parameters, which is inconvenient to do inside CLI.

The controllers do not do the actual cryptographic operations like signature generation and verification. The controllers only supply relevant data to ACA-py, and ACA-py uses the data to perform the cryptographic operations by itself.

The calls made by the controllers are asynchronous. ACA-py uses webhooks to notify the controllers about events. The controllers listen for the events and display the communication in the terminal. The webhook functionality is gotten from the ACA-py DEMO.

Running ACA-py and the controllers Both the ACA-py and the controllers run inside a Docker container. The controllers are run using python3 interpreter. Inside the controller scripts, an ACA-py agent is spawned as a sub-process.

5.4 Security Comparison to EU COVID-19 Certificates

This last section compares the security and privacy aspects of the proposed VCs to the EU vaccination certificates. The EU member states use a common approach for uniform, and interoperable certificates called eHealth Network [68].

EU Digital COVID Certificate EU Digital COVID Certificate (DCC) is digital proof that a person was either vaccinated for COVID-19, recovered from COVID-19, or was tested negative for COVID-19. The specification of DCC was published by eHealth Network based on Directive 2011/24/EU of the European Parliament and the Council, and can be found in [69].

DCC is represented as JSON and its format is shown in 5.3. The diagram was made from an existing DCC issued in the Czech Republic. The DCC was originally represented as a QR code and for this thesis decoded using [70] to the JSON format.

The decoded certificate can be split into three main parts:

1. *personally identifiable information*: date of birth (dob), first name (fn), given name (gn),
2. *information about the issuer*: issuer of the QR code (label 1), the issuer of the certificate (is),
3. *information about the vaccine certificate*: expiration date (label 4), issuance date (label 6), unique certificate identifier (ci), and vaccine-related information like tg and vp.

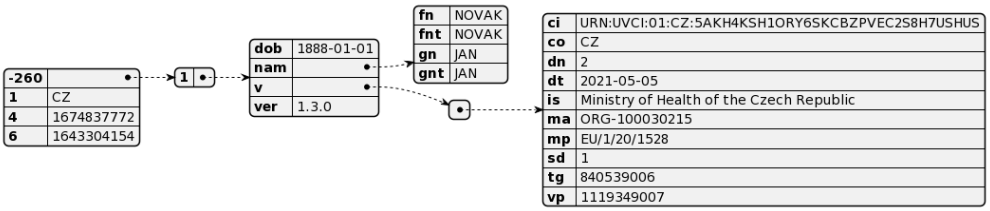

Figure 5.3 Decoded EU Digital COVID-19 Certificate

Security comparison The most important field of the DCC from the security standpoint is the **unique certificate identifier**, which is used to prevent certificate forgery. The specification specifies two signing algorithms for creating the signature: ECDSA and RSA, where ECDSA is the preferred algorithm [69]. Both of the signing algorithms should be used with SHA-256, which is used to produce a digest of the certificate which is then signed [69].

The cryptographic algorithms used in DCC are standard and, when used properly, should be secure to use. However, ECDSA is known to be fragile[2], especially in the nonce generation, and can easily break if not used properly [72]. However, incorrect implementation of BBS+ signatures can also leak secret information. The specification for BBS+ signatures in DiD emphasizes using trusted sources of randomness and not reusing nonces [50].

The DCC itself does not contain verification keys. In the case of the Czech Republic, the public keys for verification can be downloaded from an openly accessible API [73]. VCs proposed in the previous section have the public keys of the issuers embedded. However, the origin of the keys cannot be trusted in itself. The keys have to be obtained through some trusted channels before the verification, and thus the two methods are very similar.

In a DCC, a person is identified by his given name, family name, and date of birth. In the case of the VCs, a person is identified by his DiD. If multiple persons share the same name and

[2]Recently (2022) an ECDSA-related vulnerability was discovered in Java versions 15 and above. It allowed an attacker to easily forge ECDSA signatures [71].

date of birth, one DCC would be valid for all of them. On the other hand, VCs are bound to one DID, and thus the persons would have to share the keys too for the fraud to be possible.

To sum up, both the proposed VCs and DCC are designed such that the vaccination certificates are not forgeable in a feasible time. In both cases, the weakest link in the chain are humans, who can misuse the access to the signing keys and issue a certificate to an unauthorized person. Other vulnerabilities can arise when implementing cryptographic algorithms.

Privacy comparison The main advantage of the proposed VCs is that they are based on ZKP-friendly signature schemes. DCCs are not compatible with ZKPs. When a DCC is shared, all the fields in 5.3 are shared too, i.e., DCCs do not offer selective disclosure. Any time a holder shares the DCC with a verifier (for example in a restaurant or a cinema), the verifier gains immediate access to the personal information contained in the DCC. It is not necessary to share the date of birth, the number of doses, or nationality in some cases. VCs based on ZKP friendly signature schemes would allow excluding those fields that are not necessary for the given use case.

The Czech Republic's revocation of DCCs is handled using public revocation registry [74]. All revoked certificates are identified by their unique certificate identifier (ci). Anyone who has access to the database can check whether some *ci* was revoked. In the case of BBS+ signatures, pairing-based accumulators can prove set membership (or non-membership) in zero-knowledge, i.e., a person can prove in zero-knowledge that his certificate was not revoked.

Conclusion The digital certificates for COVID-19 currently used in the EU are secure, as they use proven standardized cryptography. However, they lack privacy compared to the VCs proposed in the previous section. A significant advantage of using the ECDSA algorithm is that it is NIST approved, which is not the case for BBS+ signatures. That can have an impact on government adoption.

Conclusion

The main goals of this thesis were to describe possible use cases of decentralized identity, review its privacy and security aspects, and showcase the usage of decentralized identity in a case study. The following paragraphs describe how the goals were achieved.

Firstly, the building blocks of the decentralized identity were introduced—the thesis described the intrinsic components like decentralized identifiers and verifiable credentials and the essential infrastructure like blockchains, agents, and wallets. Relationships of those components and their importance were explained. This thesis followed design principles and values described in standards and drafts by W3C and aligned communities like the Identity foundation community and Hyperledger Indy community.

The thesis introduced seven use cases. The use cases include authentication and authorization, secure communication and mitigating database breaches, credentials in healthcare, using DiD for tracking goods in supply chains, and finally, one general use case on how DiD can be used in the European Union blockchain infrastructure.

Regarding security and privacy, the thesis concentrated on privacy aspects of verifiable credentials, zero-knowledge proofs, and key management. Additionally, the security and privacy aspects were reviewed where it was relevant for a given use case.

Zero-knowledge proofs can be very conveniently used to enhance users' privacy. Thus, multiple sections are dedicated to the theory of ZKPs and their usage in decentralized identity. A decentralized identity ecosystem will heavily lie on asymmetric cryptography and key management. For that reason, a few sections on key management are provided. Regarding the verifiable credentials, the thesis was mainly concerned with the problem of correlation and how correlation can be prevented.

Next, a case study that explored the usage of verifiable credentials for COVID-19 vaccination was described. Two types of verifiable credentials were introduced. One type is compatible with a medical standard FHIR. The second one supports zero-knowledge proofs. Simple clients for a decentralized-identity-compatible server were programmed to showcase how the credentials can be exchanged, held, and verified.

Lastly, the proposed verifiable credentials were compared to the vaccination certificates used in the European Union. EU provides interoperable COVID-19 certificates based on eHealth Network. The security and privacy aspects of the EU certificates were considered and compared to the proposed verifiable credentials with the conclusion that the security level is similar and the privacy aspects of the verifiable credentials are better.

This thesis can be followed in several ways, for example:

1. Build applications based on EESSIF—the upcoming decentralized identity ecosystem built in the EU—for example, build diploma management for CTU.

2. Provide thorough research on zero-knowledge proofs, possibly with a review of cryptographic

libraries used in the ecosystem like Libursa.

3. Build a tool that would analyze the probability of correlation based on data in verifiable credentials and create guidelines for creating verifiable credentials based on the observations.
4. Do penetration tests on the infrastructure built in the ecosystem, for example, on the ACA-py server.
5. Extend ACA-py to provide compatibility with some unsupported blockchain, e.g., the Ethereum blockchain.

Bibliography

1. ALLEN, Christopher. *The Path to Self-Sovereign Identity* [online]. [N.d.] [visited on 2021-10-10]. Available from: `http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html`.
2. TOBIN, Andrew; REED, Drummond; WINDLEY, Phillip. The Inevitable Rise of Self-Sovereign Identity [online]. 2017 [visited on 2022-05-01]. Available from: `https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf`.
3. MÜHLE, Alexander et al. A Survey on Essential Components of a Self-Sovereign Identity. *Computer Science Review* [online]. 2018, vol. 30, pp. 80–86 [visited on 2021-09-20]. ISSN 15740137. Available from DOI: `10.1016/j.cosrev.2018.10.002`.
4. PON, Bryan; LOCKE, Chris; STEINBERG, Tom. *Private-Sector Digital Identity in Emerging Markets* [online]. Surrey: Caribou Digital, 2016 [visited on 2022-05-01]. Available from: `https://www.cariboudigital.net/wp-content/uploads/2019/01/Caribou-Digitial-Omidyar-Network-Private-Sector-Digital-Identity-In-Emerging-Markets.pdf`.
5. SPORNY, Manu et al. *Decentralized Identifiers (DIDs) v1.0* [online]. [N.d.] [visited on 2021-11-01]. Available from: `https://www.w3.org/TR/did-core/`.
6. HUGHES, Andrew; SPORNY, Manu; REED, Drummond. *A Primer for Decentralized Identifiers* [online]. [N.d.] [visited on 2022-04-23]. Available from: `https://w3c-ccg.github.io/did-primer/`.
7. HAMILTON-DUFFY, Kim; GRANT, Ryan; GROPPER, Adrian. *Use Cases and Requirements for Decentralized Identifiers* [online]. [N.d.] [visited on 2021-12-04]. Available from: `https://www.w3.org/TR/did-use-cases/`.
8. PREUKSCHAT, Alexander; REED, Drummond. *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*. Shelter Island: Manning, 2021. ISBN 978-1-61729-659-8.
9. STEELE, Orie; SPORNY, Manu. *DID Specification Registries* [online]. [N.d.] [visited on 2021-11-03]. Available from: `https://www.w3.org/TR/did-spec-registries/#did-methods`.
10. SPORNY, Manu; LONGLEY, Dave; CHADWICK, David. *Verifiable Credentials Data Model 1.0* [online]. [N.d.] [visited on 2022-04-04]. Available from: `https://www.w3.org/TR/vc-data-model/`.
11. COHEN, Gabe; STEELE, Orie. *Verifiable Credentials JSON Schema Specification* [online]. [N.d.] [visited on 2022-02-24]. Available from: `https://w3c-ccg.github.io/vc-json-schemas/v1/index.html#credential-schema`.

12. REED, Drummond et al. *Decentralized Key Management System - Design and Architecture V3* [Hyperledger]. 2018 [visited on 2022-04-09]. Available from: https://github.com/hyperledger/indy-sdk/blob/677a0439487a1b7ce64c2e62671ed3e0079cc11f/doc/design/005-dkms/DKMS%20Design%20and%20Architecture%20V3.md.

13. O'DONNELL, Darrell. The Current and Future State of Digital Wallets. 2019. Available also from: https://thewalletwars.s3.amazonaws.com/The-Current-and-Future-State-of-Digital-Wallets-v1.0-FINAL.pdf.

14. YAGA, Dylan et al. *Blockchain Technology Overview* [online]. Gaithersburg, MD, 2018-10 [visited on 2021-11-21]. Tech. rep., NIST IR 8202. National Institute of Standards and Technology. Available from DOI: 10.6028/NIST.IR.8202.

15. STEENBEEK, Gregory. *ESSIF Conceptual Reference for Architecture Definition - EBSI Documentation - CEF Digital* [online]. [N.d.] [visited on 2022-02-16]. Available from: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/%5Barchived%5DESSIF+Conceptual+Reference+for+Architecture+Definition.

16. PASTOR, Marta. *Trusted Schemas Registry API* [online]. [N.d.] [visited on 2022-02-24]. Available from: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Trusted+Schemas+Registry+API.

17. WANG, Jiahe; WEI, Songjie; LIU, Haozhe. Decentralized Identity Authentication with Trust Distributed in Blockchain Backbone. In: JOSHI, James; NEPAL, Surya; ZHANG, Qi; ZHANG, Liang-Jie (eds.). *Blockchain – ICBC 2019*. Cham: Springer International Publishing, 2019, pp. 202–210. Lecture Notes in Computer Science. ISBN 978-3-030-23404-1. Available from DOI: 10.1007/978-3-030-23404-1_14.

18. TERBU, Oliver. *Using OpenID Connect with Decentralized Identifiers* [online]. 2019 [visited on 2022-01-26]. Available from: https://medium.com/decentralized-identity/using-openid-connect-with-decentralized-identifiers-24733f6fa636.

19. TERBU, Oliver et al. *OpenID Connect for Verifiable Presentations* [online]. [N.d.] [visited on 2022-01-31]. Available from: https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html.

20. AUTH0. *What Is Authorization? - Examples and Definition* [online]. [N.d.] [visited on 2022-04-01]. Available from: https://auth0.com/intro-to-iam/what-is-authorization/.

21. JONES, M.; BRADLEY, J.; SAKIMURA, N. *JSON Web Token (JWT)* [online]. 2015-05 [visited on 2022-01-26]. Tech. rep., RFC7519. RFC Editor. Available from DOI: 10.17487/RFC7519.

22. STEENBEEK, Gregory. *General - EBSI Documentation - CEF Digital* [online]. [N.d.] [visited on 2021-12-04]. Available from: https://ec.europa.eu/cefdigital/wiki/display/EBSIDOC/General.

23. PASTOR, Marta. *What Is EBSI* [online]. [N.d.] [visited on 2022-04-03]. Available from: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=381517902.

24. PASTOR, Marta; STEENBEEK, Gregory. *High-level Scope of ESSIF* [online]. [N.d.] [visited on 2021-10-19]. Available from: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=379913698.

25. GS1. *Global Trade Item Number (GTIN) — GS1* [online]. [N.d.] [visited on 2022-01-08]. Available from: https://www.gs1.org/standards/id-keys/gtin.

26. TRANSMUTE. *Encoding Trust That Travels with Data — A New Product Introduction Case Study Powered by Solutions...* [Online]. 2021 [visited on 2022-01-08]. Available from: https://medium.com/transmute-techtalk/encoding-trust-that-travels-with-data-a-new-product-introduction-case-study-powered-by-solutions-a4be7e80bfdf.

27. WHO. *1 in 10 Medical Products in Developing Countries Is Substandard or Falsified* [online]. [N.d.] [visited on 2022-02-03]. Available from: https://www.who.int/news/item/28-11-2017-1-in-10-medical-products-in-developing-countries-is-substandard-or-falsified.
28. TOYODA, Kentaroh et al. A Novel Blockchain-Based Product Ownership Management System (POMS) for Anti-Counterfeits in the Post Supply Chain. *IEEE Access*. 2017, vol. 5, pp. 17465–17477. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2017.2720760.
29. CURREN, Sam et al. *DIDComm Messaging Specification* [online]. [N.d.] [visited on 2022-02-06]. Available from: https://identity.foundation/didcomm-messaging/spec/.
30. DANIEL HARDMAN. *DIDComm Mythconceptions* [online]. 2021 [visited on 2022-02-09]. Available from: https://www.youtube.com/watch?v=ovhSWg_E_54.
31. HARDMAN, Daniel. *Security, Silos, and Sovereignty* [online]. 2021 [visited on 2022-02-09]. Available from: https://daniel-hardman.medium.com/security-silos-and-sovereignty-522e30bb8eb4.
32. HARDMAN, Daniel. *Aries RFC 0049: Repudiation* [online]. 2019 [visited on 2022-04-04]. Available from: https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0049-repudiation/README.md.
33. HARDMAN, Daniel. *Aries RFC 0046: Mediators and Relays* [online]. 2019 [visited on 2022-05-01]. Available from: https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0046-mediators-and-relays/README.md.
34. HIMSS. *Interoperability in Healthcare — HIMSS* [online]. 2020 [visited on 2022-02-24]. Available from: https://www.himss.org/resources/interoperability-healthcare.
35. HL7. *Health Level Seven International - Homepage — HL7 International* [online]. [N.d.] [visited on 2022-02-24]. Available from: https://www.hl7.org/.
36. HL7. *Resource - FHIR v4.0.1* [online]. [N.d.] [visited on 2022-02-24]. Available from: https://www.hl7.org/fhir/resource.html.
37. MANDEL, Josh. *SMART Health Cards* [SMART on FHIR]. 2022 [visited on 2022-02-24]. Available from: https://github.com/smart-on-fhir/health-cards/blob/2dcf740db2fc7c464c38df2268679dff9d1606fb/docs/credential-modeling.md.
38. GOLDREICH, Oded. *Foundations of Cryptography: Volume 1, Basic Tools*. 1st edition. Cambridge: Cambridge University Press, 2008. ISBN 978-0-521-03536-1.
39. ROSEN, Alon. *ZERO-KNOWLEDGE (INTRO)* [online]. 2019 [visited on 2022-05-02]. Available from: https://cyber.biu.ac.il/wp-content/uploads/2018/08/WS-19-1-ZK-intro.pdf.
40. LINDELL, Yehuda. *How To Simulate It - A Tutorial on the Simulation Proof Technique* [online]. 2016 [visited on 2022-03-08]. Tech. rep., 046. Available from: http://eprint.iacr.org/2016/046.
41. GREEN, Matthew. *Zero Knowledge Proofs: An Illustrated Primer, Part 2* [online]. 2017 [visited on 2022-03-08]. Available from: https://blog.cryptographyengineering.com/2017/01/21/zero-knowledge-proofs-an-illustrated-primer-part-2/.
42. GOLDREICH, Oded. *Computational Complexity: A Conceptual Perspective*. 1st edition. Cambridge ; New York: Cambridge University Press, 2008. ISBN 978-0-521-88473-0.
43. WONG, David. *Real-World Cryptography*. Shelter Island, NY: Manning, 2021. ISBN 978-1-61729-671-0.
44. CANETTI, Ran et al. *Fiat-Shamir: From Practice to Theory — Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* [online]. 2019 [visited on 2022-03-08]. Available from: https://dl.acm.org/doi/pdf/10.1145/3313276.3316380.

45. CAMENISCH, Jan; LYSYANSKAYA, Anna. A Signature Scheme with Efficient Protocols. In: 2002, pp. 268–289. Available also from: https://cs.brown.edu/people/alysyans/papers/camlys02b.pdf.
46. LOOKER, Tobias; STEELE, Orie. *BBS+ Signatures 2020* [online]. [N.d.] [visited on 2022-03-08]. Available from: https://w3c-ccg.github.io/ldp-bbs2020/.
47. REED, Drummond; ZUNDEL, Brent. *What BBS+ Means For Verifiable Credentials* [online]. 2021 [visited on 2022-03-08]. Available from: https://www.evernym.com/bbs-webinar/.
48. ZUNDEL, Brent. *Why the Verifiable Credentials Community Should Converge on BBS+* [online]. 2021 [visited on 2022-03-08]. Available from: https://www.evernym.com/blog/bbs-verifiable-credentials/.
49. GLASTRA, Timo. *Hyperledger Aries Cloud Agent - Python* [Hyperledger]. 2022 [visited on 2022-04-11]. Available from: https://github.com/hyperledger/aries-cloudagent-python/blob/50772992cf354edbb2216de2659e2c44d4836576/JsonLdCredentials.md.
50. LODDER, M. et al. *The BBS Signature Scheme* [online]. [N.d.] [visited on 2022-04-07]. Available from: https://identity.foundation/bbs-signature/draft-bbs-signatures.html.
51. KOCKEN, Anthony. *Fix the Fill: Simple Solutions to Common Issues With Autofill* [online]. 2021 [visited on 2022-04-02]. Available from: https://blog.dashlane.com/troubleshoot-autofill/.
52. PALATINUS, Marek et al. *BIP 0039 - Bitcoin Wiki* [online]. 2013 [visited on 2022-04-07]. Available from: https://en.bitcoin.it/wiki/BIP_0039.
53. DINGLE, Pamela. *New Explorations in Secret Recovery* [online]. 2020 [visited on 2022-04-07]. Available from: https://techcommunity.microsoft.com/t5/identity-standards-blog/new-explorations-in-secret-recovery/ba-p/1441550.
54. YOUNG, Kaliya. Verifiable Credentials Flavors Explained. [N.d.], p. 21.
55. DEVENTER, Oskar et al. *Peer DID Method Specification* [online]. [N.d.] [visited on 2022-04-03]. Available from: https://identity.foundation/peer-did-method-spec/.
56. WEST, Ryan et al. *Aries RFC 0434: Out-of-Band Protocol 1.1* [Hyperledger]. 2022 [visited on 2022-04-03]. Available from: https://github.com/hyperledger/aries-rfcs/blob/c2bcc63fbf7d009dbf87ebec0f975f800e1900b9/features/0434-outofband/README.md.
57. WEST, Ryan et al. *Aries RFC 0023: DID Exchange Protocol 1.0* [Hyperledger]. 2022 [visited on 2022-04-03]. Available from: https://github.com/hyperledger/aries-rfcs/blob/c2bcc63fbf7d009dbf87ebec0f975f800e1900b9/features/0023-did-exchange/README.md.
58. *SMART Health Cards Framework* [online]. [N.d.] [visited on 2022-03-31]. Available from: https://spec.smarthealth.cards/#healthwalletissuevc-operation.
59. LOOKER, Tobias; STEELE, Orie; PROROCK, Michael. *Vaccination Certificate Vocabulary v0.1* [online]. 2021 [visited on 2022-03-31]. Available from: https://w3c-ccg.github.io/vaccination-vocab/.
60. HL7. *Modules - FHIR v4.0.1* [online]. [N.d.] [visited on 2022-03-31]. Available from: https://www.hl7.org/fhir/modules.html.
61. GOOGLE. *FHIR — Cloud Healthcare API* [online]. [N.d.] [visited on 2022-03-31]. Available from: https://cloud.google.com/healthcare-api/docs/concepts/fhir.
62. APPLE. *Healthcare - Health Records* [online]. [N.d.] [visited on 2022-03-31]. Available from: https://www.apple.com/healthcare/health-records/.

63. PIETERSE, Louise. *SMART Health Card Launches in Some US States* [online]. 2021 [visited on 2022-03-31]. Available from: https://www.covidpasscertificate.com/smart-health-card/.

64. BOYD, Michael. *Overview of the System — Indy Plenum Documentation* [online]. 2018 [visited on 2022-04-16]. Available from: https://hyperledger-indy.readthedocs.io/projects/plenum/en/latest/main.html.

65. BOYD, Michael. *Hyperledger Indy — Hyperledger Indy 1.0 Documentation* [online]. 2018 [visited on 2022-04-16]. Available from: https://hyperledger-indy.readthedocs.io/en/latest/index.html.

66. SOVRIN, Foundation. *SSI Metrics Dashboards* [online]. [N.d.] [visited on 2022-04-17]. Available from: https://sovrin.org/ssi-metrics-dashboards/.

67. NEIHEISER, Ray; MATOS, Miguel; RODRIGUES, Luís. The Quest for Scaling BFT Consensus through Tree-Based Vote Aggregation. *arXiv:2103.12112 [cs]* [online]. 2021 [visited on 2022-04-17]. Available from arXiv: 2103.12112 [cs].

68. *eHealth and COVID-19* [online]. [N.d.] [visited on 2022-04-19]. Available from: https://ec.europa.eu/health/ehealth-digital-health-and-care/ehealth-and-covid-19_en.

69. *Official Journal of the European Union* [online]. [N.d.] [visited on 2022-04-07]. Available from: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32021D1073&from=EN.

70. BÖCK, Hanno. *Vacdec* [online]. 2022 [visited on 2022-04-21]. Available from: https://github.com/hannob/vacdec/blob/ec06f1270a7659dc79ad039f36368bdc049175a1/vacdec.

71. GOODIN, Dan. *Major Cryptography Blunder in Java Enables "Psychic Paper" Forgeries* [online]. 2022 [visited on 2022-04-21]. Available from: https://arstechnica.com/information-technology/2022/04/major-crypto-blunder-in-java-enables-psychic-paper-forgeries/.

72. MILLER, James. *ECDSA: Handle with Care* [online]. 2020 [visited on 2022-04-21]. Available from: https://blog.trailofbits.com/2020/06/11/ecdsa-handle-with-care/.

73. *Czechia Signing Certificates* [online]. [N.d.] [visited on 2022-04-21]. Available from: https://dgcverify.mzcr.cz/api/v1/verify/NactiPodpisoveCertifikaty.

74. *Czechia Revocation Registry* [online]. [N.d.] [visited on 2022-04-21]. Available from: https://dgcverify.mzcr.cz/api/v1/verify/NactiRevokovaneCertifikaty.

Contents of Enclosed CD

```
root
├── README.MD ..... description of the contents of the CD
├── src
│   ├── case-study ..... source code of the case study
│   └── thesis ..... source code of the thesis in LaTeX
└── text ..... text of the thesis
    └── thesis.pdf ..... the thesis in PDF format
```