

# Self Sovereign Identity (SSI)

Version 1.01, 30.01.2023

# Inhaltsverzeichnis

|  |    |
|--|----|
| 1. Was ist eine digitale Identität?                            | 2  |
| 1.1. Praktisches Beispiel                                      | 2  |
| 2. OpenID Connect (OIDC)                                       | 4  |
| 2.1. Was ist OIDC?   | 4  |
| 2.2. Erweiterung des OAuth 2.0 Authorization Request           | 4  |
| 2.2.1. Scope   | 4  |
| 2.2.2. Claims  | 5  |
| 2.2.3. ID-Token  | 5  |
| 2.2.4. Access Token  | 6  |
| 2.3. Die Teilnehmer  | 7  |
| 2.3.1. Einfacher Beispiel Ablauf                               | 7  |
| 2.4. Einordnung von OIDC                                       | 8  |
| 3. Self Sovereign Identity (SSI)                               | 10 |
| 3.1. SSI Trust Triangle  | 10 |
| 3.1.1. Austellung und Verifizierung von Verifiable Credentials | 11 |
| 3.2. Decentralized Identifiers (DID)                           | 12 |
| 3.2.1. Komponenten   | 12 |
| 3.3. Verifiable Credential (VC)                                | 14 |
| 3.3.1. Architektur   | 14 |
| 4. DIDComm   | 16 |
| 4.1. Was ist DIDComm?  | 16 |
| 4.2. DIDComm v2  | 16 |
| 4.3. DIDComm Ablauf (vereinfacht!)                             | 16 |
| 4.3.1. DID Erzeugung   | 17 |
| 5. Technischer Vergleich zwischen DIDComm und OIDC             | 18 |
| 6. OIDC4/VC und OIDC4/VP                                       | 19 |
| 6.1. OIDC4/VC  | 19 |
| 6.2. OIDC4/VP  | 19 |
| 6.3. Selective Attribute Disclosure                            | 19 |
| 6.4. Zero-Knowledge Proof                                      | 20 |
| 7. Proof-of-Concept (SSI)                                      | 23 |
| 7.1. Szenarien   | 23 |
| 7.2. Komponenten   | 25 |
| 7.3. Technische Umsetzung                                      | 28 |
| 7.4. Ausführung  | 29 |
| 8. Beurteilung   | 36 |
| 8.1. SSI für Personen  | 36 |
| 8.2. SSI für Organisationen                                    | 36 |

|                                 |    |
|---------------------------------|----|
| 8.3. SSI für Objekte . . . . .  | 36 |
| 8.4. Kritiken . . . . .         | 37 |
| 9. Quellenverzeichnis . . . . . | 39 |

## **Unser Team**

Studenten der HTW Dresden:

- Pascal Thielemann
- Ludwig Schönthier
- Kenneth-Raphael Keil
- Johannes Schuster
- Roman Patzig

Betreuung durch T-Systems MMS:

- Prof. Dr. Frank Schönefeld
- Dr. Ivan Gudymenko

Ein besonderes Dankeschön möchten wir an dieser Stelle auch an Danil Tolonbekov geben, der uns bei der Umsetzung des PoC geholfen hat.

## **Aufgabenstellung**

Unsere Aufgabenstellung für das Praxisprojekt lautete wie folgt:

1. A technical comparison of DIDComm and OIDC
2. If OIDC already exists, then why OIDC for Verifiable Credentials?
3. Different between traditional OIDC and OpenID4VC / VP. How does OpenID4VC / VP support the idea of SSI?
4. Does OpenID4VC / VP supports selective attribute disclosure / ZKP?
5. Working POC

## **Umfang der Projektarbeit**

Diese Arbeit dient dem Einstieg in die Thematik "Self-Sovereign Identity". Sie soll die Grundlage für das Verständnis selbstbestimmter Identitäten bilden und weiterhin einen Ausblick über technologische Umsetzungsmöglichkeiten geben.

Dabei wird an aktuelle Autorisierungs- und Authentifizierungsprotokolle und Web-Standards wie OIDC herangeführt, um ein Verständnis für die derzeitige Problematik und Situation im Web zu entwickeln. Im Anschluss wenden wir den Blick in eine mögliche Lösungsalternative, welche durch "Decentralized Identifier" (DID) und das Protokoll "DIDComm" versucht das Konzept von SSI technisch zu realisieren. Wir widmen uns im Zuge dessen den einzelnen Fragen aus unserer Aufgabenstellung und versuchen diese im Kontext von SSI und OICD/DIDComm zu erklären.

Zuletzt wollen wir durch ein PoC die technische Realisierbarkeit von SSI nachweisen und einige Vorteile und Anwendungsfälle näher erläutern, sowie aber auch einzelne Kritikpunkte aufführen.

# Chapter 1. Was ist eine digitale Identität?

Eine digitale Identität ist eine Online-Repräsentation einer Person oder einer Organisation, die verwendet wird, um ihre Präsenz und Interaktionen im digitalen Raum zu verwalten. Im Web 2.0 wird die digitale Identität durch die Verwendung von E-Mail-Adressen, sozialen Netzwerken und anderen Online-Konten abgebildet.

Die wichtigsten Standards und Möglichkeiten zur Abbildung einer digitalen Identität im Web 2.0 sind:

- **OAuth:** ein Protokoll zur Berechtigungsdelegation, das es Benutzern ermöglicht, ihre Online-Konten bei einem Dienstanbieter zu nutzen, um Zugriff auf andere Anwendungen zu erlangen, ohne ihre Anmeldeinformationen an jedes Mal anzugeben.
- **OpenID:** ein Single-Sign-On-Protokoll, das Benutzern ermöglicht, sich mit einem einzigen Konto bei mehreren Websites anzumelden.
- **SAML:** ein Standard zur Übertragung von Authentifizierungs- und Autorisierungsdaten, der es Organisationen ermöglicht, Benutzerauthentifizierung und -autorisierung zwischen verschiedenen Anwendungen und Websites zu koordinieren.
- **Self-Sovereign Identity (SSI) / Decentralized Identifier (DID):** Ein neuer Ansatz zur Abbildung von digitalen Identitäten, bei dem Benutzer die volle Kontrolle über ihre Identitätsdaten haben und diese dezentral gespeichert sind, ohne dass eine zentrale Autorität oder ein Unternehmen für ihre Verwaltung verantwortlich ist.

Im Rahmen dieser Ausarbeitung wollen wir uns näher mit OpenID (OIDC) und dem Konzept der Self-Sovereign Identity (SSI) / Decentralized Identifier (DID) beschäftigen.

## 1.1. Praktisches Beispiel

Ein praktisches Beispiel für die digitale Identität ist der Einsatz von OIDC für die Anmeldung bei einer Web-Anwendung mit einem Google- oder Facebook-Account. Der Benutzer wird aufgefordert, sich bei Google oder Facebook anzumelden, und dann wird seine Identität an die Web-Anwendung weitergeleitet, anstatt dass der Benutzer ein neues Konto erstellen oder seine Anmeldeinformationen eingeben muss.

Eines der größten Bedenken im Zusammenhang mit OIDC ist die Datenschutzfrage. Wenn Benutzer ihre Identität bei einem Autorisierungsdienstanbieter verifizieren, geben sie ihm Zugang zu ihren persönlichen Daten, einschließlich ihres Namens, ihrer E-Mail-Adresse und anderer Informationen, die der Dienstanbieter sammeln kann. Dies kann zu einer Überwachung und Verwendung dieser Daten durch den Dienstanbieter führen, wodurch die Privatsphäre und die digitale Identität des Benutzers gefährdet werden können.

Weitere mögliche Nachteile von OIDC können sein, dass es möglicherweise keine umfassende Kontrolle über die Art und Weise gibt, wie Benutzerdaten gespeichert, verarbeitet und verwendet werden, und dass es ein hohes Maß an Abhängigkeit von Dritten gibt, die die Authentifizierung und Autorisierung durchführen.

Um OIDC besser verstehen zu können und um eine Frage darauf zu finden, warum SSI einen neuen

Ansatz für die digitale Identität darstellt, der versucht zukünftig einige der Probleme von OIDC zu lösen, beschäftigen wir uns im nächsten Abschnitt intensiver mit dem Protokoll von OIDC.

# Chapter 2. OpenID Connect (OIDC)

## 2.1. Was ist OIDC?

OpenID Connect ist ein Authentifizierungsprotokoll und eine Erweiterung des OAuth 2.0 Standards. Es ermöglicht Clients die Überprüfung der Identität des Endbenutzers auf der Grundlage der von einem Autorisierungsserver durchgeführten Authentifizierung, sowie den Erhalt grundlegender Profilinformationen über den Endbenutzer in einer interoperablen und REST-ähnlichen Schnittstelle. Mit OpenID Connect können Clients aller Art, einschließlich webbasierter, mobiler und JavaScript-Clients, Informationen über authentifizierte Sitzungen und Endbenutzer anfordern und empfangen.

Bei einer OpenID-Connect-Anfrage wird der Benutzer an einen Identitätsanbieter, wie zB. Google oder Facebook weitergeleitet, der die angeforderten Informationen authentifiziert und autorisiert und dann mit einem ID-Token und optional einem Zugriffstoken an die anfordernde Anwendung (zB. einer App) zurückgeschickt.



**Authentifizierung:** *"Bist du das wirklich?"* Authentifizierung ist die Möglichkeit, zu beweisen, dass ein Benutzer oder eine Anwendung wirklich die Person oder die Anwendung ist, welche die Anwendung beansprucht.



**Autorisierung:** *"Bist du berechtigt das zu tun?"* Autorisierung bezeichnet das initiale Zuweisen und das wiederholte Überprüfen von Zugriffsrechten mittels spezieller Methoden bezüglich interessierter Systemnutzer zu Daten und zu Diensten.

**UseCase:** zB. SSO (Single Sign-On) für Konsumenten Apps

## 2.2. Erweiterung des OAuth 2.0 Authorization Request

Ursprünglich ist OIDC eine Erweiterung des OAuth 2.0 Standards um die Authentifizierung des Nutzers. Dabei wird der **OAuth 2.0 Authorization Request** um den Parameter **Scope** erweitert. Hiermit kann festgelegt werden, auf welche Informationen der Client vom Nutzer zugreifen darf. Diese Informationen werden in Form eines **Claims** zurückgesendet.

### 2.2.1. Scope

Scope ist ein Konzept in OIDC, welches die Zugriffsberechtigungen eines Access Tokens auf bestimmte Ressourcen zB. die eines Nutzers beschreibt. Es ist eine Liste von Claims, die angeben, für welche Bereiche eine Anwendung berechtigt ist, wenn diese eine Anfrage an eine Ressource stellt.

Ein Beispiel für den Einsatz von Scope könnte sein, dass eine Anwendung, die auf Gesundheitsdaten zugreifen möchte, über einen Access Token verfügt, welches "read:patient" berechtigt. Das bedeutet, dass sie Zugriff auf die Patientendaten hat, aber keinen Zugriff auf administrative Daten oder andere Daten, die nicht Teil des "read:patient" Scopes sind.

In OIDC werden Scopes verwendet, um die Zugriffsberechtigungen für Benutzerdaten und -ressourcen zu beschreiben und zu regeln. Dies hilft, sicherzustellen, dass Benutzerdaten geschützt bleiben und, dass Anwendungen nur die Berechtigungen erhalten, die sie zur Ausführung ihrer Funktionen benötigen. Häufig stimmen Nutzer jedoch bei der Anmeldung zu, dass Anwendungen alle ihre Nutzerdaten lesen und verarbeiten dürfen, ohne sich dessen bewusst zu sein. Dadurch entsteht ein Verlust der Datenkontrolle seitens des Endnutzers.

## 2.2.2. Claims

Claims sind in diesem Kontext Aussagen über eine Person oder eine Sache, die in einem Token enthalten sind. Sie beschreiben Informationen, die einem Empfänger des Tokens (häufig der Anwendung/Webseite/Relying Party) bekannt gegeben werden sollen. In OIDC werden Claims verwendet, um Benutzerdaten und Informationen über die Authentifizierung und Autorisierung bereitzustellen.

Ein Beispiel für ein Claim könnte sein:

```
"name": "Jane Doe"  
"email": "janedoe@example.com"  
"birthdate": "0000-10-31"
```

In diesem Beispiel beschreiben die Claims den vollständigen Namen, die E-Mail-Adresse und das Geburtsdatum einer Person. Diese Claims können in einem OIDC-Token enthalten sein und an eine Anwendung weitergegeben werden, um die Identität eines Benutzers zu bestätigen und dessen Daten zu verwenden. Es hängt von der Implementierung und Konfiguration ab, welche Claims in einem Token enthalten sind.

## 2.2.3. ID-Token

Um einen Nutzer zu identifizieren, wird ein ID-Token verwendet. Es handelt sich hierbei um ein signiertes und optional verschlüsseltes Datenelement, welches **Informationen über den Benutzer** enthält, wie z.B. den Namen, die E-Mail-Adresse und andere identifizierende Merkmale. ID-Tokens werden zur Authentifizierung des Benutzers für eine einmalige Sitzung verwendet.

Ein Beispiel für einen ID-Token im JSON-Format:

```
{  
  "iss": "http://my-domain.auth0.com",  
  "sub": "auth0|123456",  
  "aud": "my_client_id",  
  "exp": 1311281970,  
  "iat": 1311280970,  
  "name": "Jane Doe",  
  "given_name": "Jane",  
  "family_name": "Doe",  
  "gender": "female",  
  "birthdate": "0000-10-31",  
  "email": "janedoe@example.com",
```

```
"picture": "http://example.com/janedoe/me.jpg"  
}
```

Diese Felder bezeichnen die Claims im Kontext von OIDC. Die einzelnen Teilnehmer werden im nächsten Abschnitt nochmal genauer erläutert.

- **"iss" (Issuer)**: Identifiziert die Partei, welche den Token ausgestellt hat (OpenID Provider und/oder Identity Provider)
- **"sub" (Subject)**: Eindeutige Identifikation des Benutzers, für den der Token ausgestellt wurde (Resource Owner)
- **"aud" (Audience)**: Empfänger des Tokens (Reyling Partie)
- **"exp" (Expiration Time)**: Ablaufdatum des Tokens
- **"iat" (Issued At)**: Zeitpunkt, zu dem der Token ausgestellt wurde

und weitere Informationen über den Benutzer.

## 2.2.4. Access Token

Um Zugriff auf die API oder weitere Ressourcen zu erhalten, wird ein Zugriffstoken verwendet. Es handelt sich dabei um ein signiertes und optional verschlüsseltes Datenelement, welches **Informationen über den Client und den Ressourcenserver**, sowie eine **Berechtigungserteilung** enthält. Zugriffstoken werden für den Zugriff auf geschützte Ressourcen im Namen eines autorisierten Benutzers verwendet.

Ein Beispiel für einen Access Token im JSON-Format

```
{  
  "iss": "https://my-domain.auth0.com/",  
  "sub": "auth0|123456",  
  "aud": [  
    "https://example.com/health-api",  
    "https://my-domain.auth0.com/userinfo"  
,  
  "azp": "my_client_id",  
  "exp": 1311281970,  
  "iat": 1311280970,  
  "scope": "openid profile read:patients"  
}
```

Interessant ist vor allem der letzte Parameter **Scope**, welcher die wesentliche Erweiterung von OAuth zu OIDC darstellt und das vorherige Beispiel nochmals aufgreift. In diesem Beispiel wird folgendes Recht zum Lesen von Patientendaten vergeben:

read:patients -> erlaubt den Zugriff auf Patientendaten

## 2.3. Die Teilnehmer

In OIDC werden 4 Teilnehmer benötigt, um einen Authentifizierungsprozess durchzuführen. Diese Teinehmer sind der Identity Provider (IdP), die Relying Party (RP), der OpenID Provider (OP) und der Resource Owner (RO).

- **Identity Provider (IdP):** Ein IdP ist ein Dienst (zb. Google oder Facebook), welcher Benutzer authentifiziert und nach erfolgreicher Authentifizierung einen Identitäts-Token ausstellt. Der IdP ist für die sichere Authentifizierung von Benutzern, sowie für die Verwaltung und den Schutz ihrer Identitäten verantwortlich. Die Nutzerdaten werden dabei auf den Servern des IdP's gesichert und gespeichert.
- **Relying Party (RP):** Ein RP ist ein Dienst oder Client (zb. eine App), der sich auf den IdP verlässt, um seine Benutzer authentifizieren zu können. Der RP ist dafür verantwortlich, die Identitäts-Token vom IdP zu verbrauchen, um seine Benutzer zu identifizieren.



**Identifizierung:** "*Wer bist du?*" Identifikation ist die Fähigkeit, eindeutig einen Benutzer eines Systems oder einer Anwendung zu identifizieren, welche im System aufgeführt wird.

- **OpenID Provider (OP):** Ein OP ist ein Dienst, welcher eine OpenID Connect-Schnittstelle zwischen dem IdP und der RP bereitstellt. Der OP ist verantwortlich für die sichere Bereitstellung der notwendigen Protokoll- und Kommunikationsinfrastruktur, um den Authentifizierungsprozess zu erleichtern.
- **Resource Owner (RO):** Ein RO ist die Entität, welche den Zugriff auf eine geschützte Ressource (zb. Personeninformationen) gewährt, z.B. ein Endnutzer. Der RO ist für die Autorisierung des Zugriffs auf die Ressource verantwortlich und kann entweder der Endbenutzer selbst oder eine Entität sein, der die Autorität übertragen wurde, im Namen des Endbenutzers handeln zu dürfen.

### 2.3.1. Einfacher Beispiel Ablauf

Der Ablauf des OIDC-Protokolls wird im Folgenden aus einfacher Sicht der Teilnehmer dargestellt, ohne in die Tiefe technischer Details zu gehen.

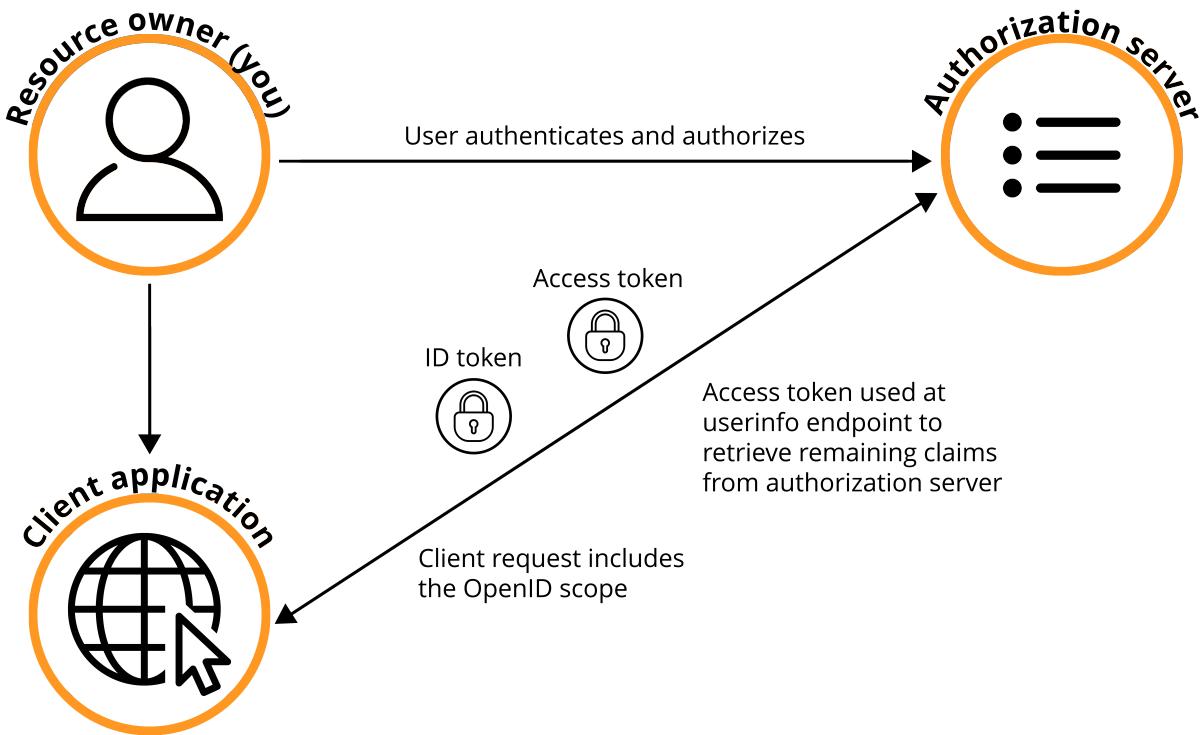


Abbildung: OpenID Connect Flow, auf Basis (Referenz) von ([https://images.pingidentity.com/image/upload/f\\_auto,q\\_80,w\\_778,c\\_scale/ping\\_dam/content/dam/ping-6-2-assets/images/resources/articles/oidc-diagram.png](https://images.pingidentity.com/image/upload/f_auto,q_80,w_778,c_scale/ping_dam/content/dam/ping-6-2-assets/images/resources/articles/oidc-diagram.png)) (Quelle)

1. Ein Endbenutzer (Resource Owner) besucht eine Webseite (Relying Party), welche OpenID Connect unterstützt und klickt z.B. auf eine Schaltfläche "Anmelden über [IdP]".
2. Die Webseite (auch Client genannt) leitet den Endbenutzer an den OpenID Provider (OP), häufig gleichzeitig auch den Identity Provider, mit einer Anfrage zur Authentifizierung des Benutzers weiter.
3. Der Identity Provider authentifiziert den Benutzer, indem er ihn auffordert, seinen Benutzernamen und sein Passwort einzugeben. Dabei müssen die Daten die angefordert werden, bereits vor der Abfrage deklariert sein.
4. Nach erfolgreicher Authentifizierung sendet der OpenID Provider eine Authentifizierungsantwort an die Relying Party mit einem ID-Token und einem Access Token zurück.
5. Die Relying Party überprüft den ID-Token, um sicherzustellen, dass dieser gültig ist und dass der Benutzer derjenige ist, der er vorgibt zu sein.
6. Die Relying Party kann nun das Access Token verwenden, um im Namen des Benutzers API-Aufrufe an den OpenID Provider zu tätigen.
7. Die Webseite kann dadurch dem Nutzer ein personalisiertes Erlebnis auf der Webseite auf Grundlagen seiner persönlichen Daten bieten.

## 2.4. Einordnung von OIDC

OIDC ist der derzeitige Standard für die Identifizierung, Authentifizierung und Autorisierung im

Web 2.0. Dabei treten gewisse Herausforderungen und Bedenken auf, die im ersten Abschnitt bereits erläutert wurden.

Im nächsten Abschnitt wollen wir das Konzept von SSI genauer erläutern und kurz auf die technischen Grundrahmenbedingungen eingehen. SSI kann durch verschiedene Methoden und Protokolle implementiert werden. Eines der am häufigsten verwendeten Protokolle ist dabei DIDComm, welches wir später nochmal konkreter im Vergleich zu OIDC betrachten werden.

# Chapter 3. Self Sovereign Identity (SSI)

Self Sovereign Identity ist ein Konzept, welches den Nutzer in den zentralen Mittelpunkt bei der Verwaltung seiner eigenen digitalen Identität stellt und damit die Autonomie des Nutzers gewährleistet.

Die Self-Sovereign Identity (SSI) wurde als Lösung für die Herausforderungen des digitalen Identitätsmanagements entwickelt. Sie ermöglicht es Menschen, Daten über sich selbst zu verwalten, kontrollieren und zu teilen, ohne auf eine zentrale Behörde oder ein Unternehmen angewiesen zu sein. Damit können Menschen ihre Daten besser schützen und sicherstellen, dass sie nur dann verwendet werden, wenn sie es wünschen.

## 3.1. SSI Trust Triangle

Um das Konzept hinter Self-Sovereign Identity umsetzen zu können, gibt es das Prinzip des "SSI-Trust Triangle", welches aufzeigt, wie die Echtheit einer Identität und ihrer Daten (Behauptungen) in einer digitalen Welt bewiesen werden kann.

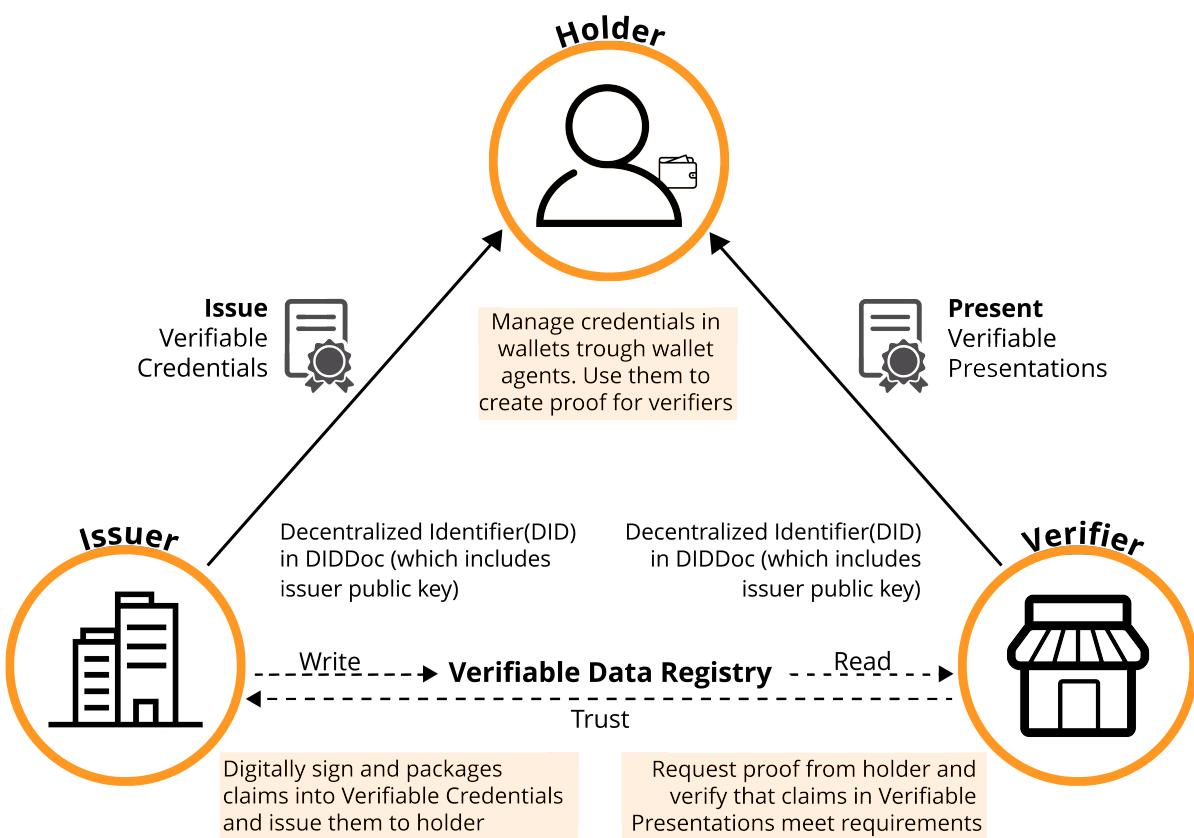


Abbildung: SSI Trust Triangle, auf Basis (Referenz) von ([https://courses.edx.org/asset-v1:LinuxFoundationX+LFS178x+3T2022+type@asset+block@LFS178x\\_2022\\_Chapter3Images\\_Graphic\\_9.png](https://courses.edx.org/asset-v1:LinuxFoundationX+LFS178x+3T2022+type@asset+block@LFS178x_2022_Chapter3Images_Graphic_9.png)) (Quelle)

### Holder

Eine Person oder Organisation, die über eine digitale Identität verfügt und diese verwaltet. Ein Holder hat die Kontrolle über seine Daten und entscheidet, wem er diese zugänglich macht.

Dieser erhält und besitzt Verifiable Credentials, welche wiederum in seiner digitalen Wallet gespeichert sind.



**Wallet/Agent:** Eine Software-Anwendung, die es einem Holder ermöglicht, seine digitale Identität und Verifiable Credentials zu verwalten und zu nutzen. Ein Wallet/Agent dient als Schnittstelle zwischen dem Holder und anderen Teilnehmern im SSI-System und ist für die Verwaltung der privaten Schlüssel des Holders und die Übertragung von Daten zuständig. Es hilft dem Holder, seine digitale Identität sicher zu speichern und zu nutzen und ermöglicht es ihm, Verifiable Credentials an andere Teilnehmer zu übertragen.

## Issuer

Eine Instanz welche autorisiert ist, Verifiable Credentials auszustellen. Dabei handelt es sich beispielsweise um eine öffentliche Einrichtung wie eine Hochschule oder ein Bürgeramt. Die ausgestellten Credentials werden mit der DID des Issuers versehen, um die Daten später überprüfbar zu machen.

## Verifier

Eine Entität, welche Daten anfordert und nach Erhalt prüft. Der Holder möchte sich gegenüber dem Verifier authentifizieren.

## Verifiable Data Registry

Eine zentrale oder dezentrale Datenbank, in welcher die DID des Issuer's gespeichert wird.



Auch eine Blockchain kann als Verifiable Data Registry im Kontext von Self-sovereign Identity (SSI) verwendet werden. Eine Blockchain-basierte VDR bietet ein dezentrales und sicheres System der DID-Speicherung. Durch die Verwendung einer Blockchain kann sichergestellt werden, dass die Daten in der VDR unveränderlich und sicher sind, da jede Transaktion in der Blockchain durch Kryptographie geschützt ist. Außerdem stellt eine Blockchain-basierte VDR eine Überprüfbarkeit und Integrität der Daten sicher, da jeder Teilnehmer im Netzwerk Zugriff auf eine einheitliche Version der Daten hat.



In der VDR wird die DID von öffentlichen Institutionen oder Organisationen gespeichert, welche Verifiable Credentials ausstellen. Wiederum werden keine privaten DIDs von einzelnen Personen gespeichert.

### 3.1.1. Austellung und Verifizierung von Verifiable Credentials

Im Folgenden werden die einzelnen Schritte beschrieben, die in einem Austellungs- und Verifizierungsprozess für Verifiable Credentials innerhalb des SSI-Trust-Triangles stattfinden:

1. **Ausstellung:** Ein Issuer erstellt Verifiable Credentials, die Informationen über eine Person enthalten und stellt diese Credentials an den Holder aus.
2. **Speicherung:** Der Holder speichert die ausgestellten Verifiable Credentials in seinem Wallet/Agent.
3. **Identitätsbereitstellung:** Der Holder möchte eine Dienstleistung des Verifiers in Anspruch

nehmen. Um seine Identität zu bestätigen, fordert der Verifier die Verifiable Credentials vom Holder an.

4. **Überprüfung:** Der Verifier überprüft, ob die Verifiable Credentials des Holders von einem vertrauenswürdigen Issuer ausgestellt wurden.

a. **Überprüfung erfolgreich:** Wenn die Verifiable Credentials von einem vertrauenswürdigen Issuer ausgestellt wurden, kann der Verifier die Identität des Holders bestätigen und die gewünschte Dienstleistung erbringen. (Durch DID referenziert)

b. **Überprüfung fehlgeschlagen:** Wird erkannt, dass die Verifiable Credentials nicht durch einen autorisierte Issuer ausgestellt wurden, so kann der Holder nicht authentifiziert werden.

## 3.2. Decentralized Identifiers (DID)

Eine DID (Decentralized Identifier) ist eine spezifische Art von Identifikator in einer SSI (Self-Sovereign Identity) Lösung, die es einer Person, Organisation oder einem Gegenstand ermöglicht, seine Identität unabhängig und dezentral zu verwalten. Eine DID kann in einer z.B. in einer dezentralen Datenbank (z.B. einer Blockchain) erstellt, verwaltet und gespeichert werden, was wiederum bedeutet, dass es keine zentrale Stelle gibt, die über die Kontrolle oder Verwaltung dieser Identität verfügt. Stattdessen liegt die Kontrolle der DID direkt beim Inhaber der Identität. Dabei fungiert die DID als ein Adressverweis auf ein DID-Dokument.

Die DID ist eine Textzeichenfolge, die aus drei Bestandteilen besteht:

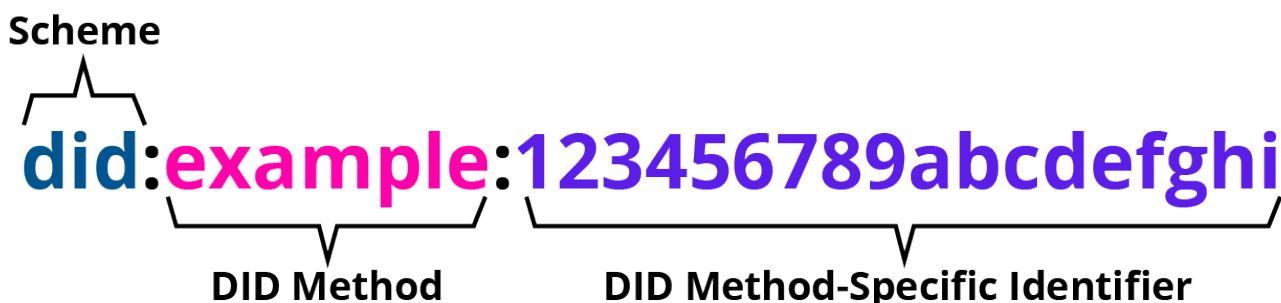


Abbildung: *DID* Beispiel ([https://courses.edx.org/asset-v1:LinuxFoundationX+LFS178x+3T2022+type@asset+block@LFS178x\\_2022\\_Chapter3Images\\_Graphic\\_10.png](https://courses.edx.org/asset-v1:LinuxFoundationX+LFS178x+3T2022+type@asset+block@LFS178x_2022_Chapter3Images_Graphic_10.png)) (Quelle)

### 3.2.1. Komponenten

Im folgenden werden die einzelnen Komponenten der DID und deren Beziehungen zueinander grafisch dargestellt und erläutert.

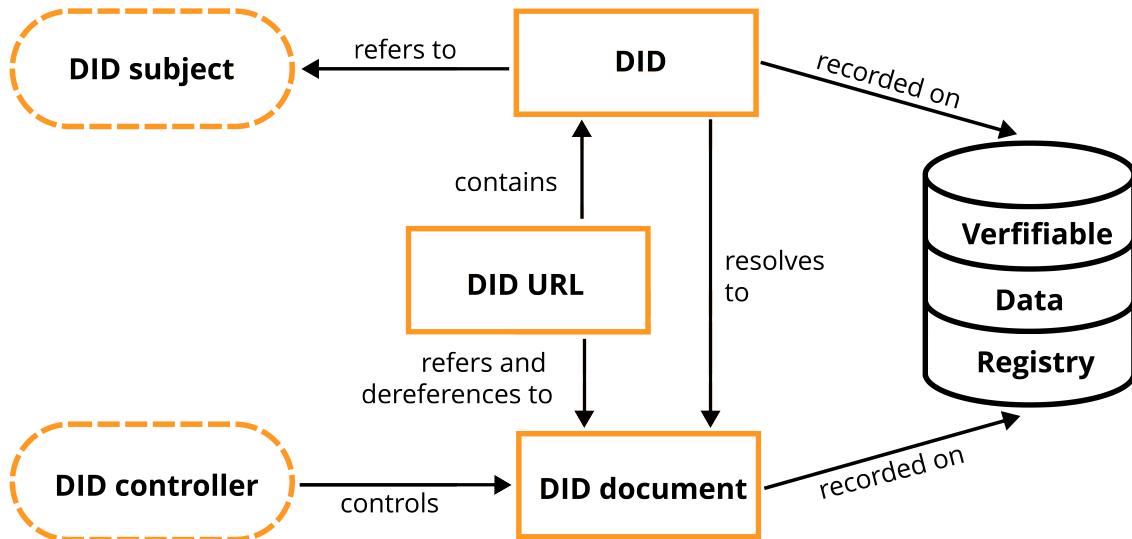


Abbildung: DID Architektur Überblick, auf Basis (Referenz) von ([https://www.w3.org/TR/did-core/diagrams/did\\_brief\\_architecture\\_overview.svg](https://www.w3.org/TR/did-core/diagrams/did_brief_architecture_overview.svg)) (Quelle)

## DID and DID URL

Ein dezentraler Bezeichner (Decentralized Identifier), welcher in ein DID-Dokument aufgelöst wird. Eine DID-URL erweitert die Syntax eines grundlegenden DID um andere Standard-URI-Komponenten, wie Pfad, Abfrage und Fragment, um eine bestimmte Ressource zu lokalisieren - beispielsweise einen kryptografischen öffentlichen Schlüssel innerhalb eines DID-Dokumentes oder eine Ressource außerhalb des DID-Dokumentes. Eine Beispiel DID URL:

did:example:123456789abcdefghi/path/to/rsrC

## DID Subject

Die Entität, auf welche sich eine DID bezieht wird als "DID Subject" bezeichnet. Dabei handelt es sich um eine Person, Organisation oder einen Gegenstand.

## DID Controller

Ein DID Controller besitzt die Fähigkeit, Informationen eines DID Documents zu ändern und kann auch das DID Subject sein, muss es aber nicht.



Ein Beispiel dafür, dass ein DID Controller nicht gleichzeitig das DID Subject ist, wäre eine Eltern-Kind Beziehung, wo die Eltern Rechte (Controlling) über das Kind (Subjekt) verfügen.

## Verifiable Data Registry

Damit DIDs in DID-Dokumente aufgelöst werden können, werden sie in der Regel in einem zugrunde liegenden System oder Netzwerk gespeichert. Dabei handelt es sich beispielsweise um

Blockchain oder sonstige Datenbank.

## DID Document

DID-Dokumente enthalten Informationen, die mit einer DID verbunden sind. Sie stellen Informationen bereit, welche für die Interaktionen mit dem DID-Subjekt relevant sind. Darunter fallen zum Beispiel public keys oder Endpoints, welche zum Verschlüsseln und Übermitteln von Nachrichten benötigt werden.

## DID Methode

DID-Methoden beschreiben den Mechanismus, mit welchem eine bestimmte Art von DID und das zugehörige DID-Dokument erstellt, aufgelöst, aktualisiert und deaktiviert werden kann.

## DID Resolver und DID Resolution

Ein DID-Auflöser ist eine Systemkomponente, die eine DID als Eingabe annimmt und ein konformes DID-Dokument als Ausgabe erzeugt. Dieser Prozess wird als DID-Auflösung bezeichnet.

## DID URL Dereferencers

Ein DID-URL-Dereferencer ist eine Systemkomponente, die eine DID-URL als Eingabe annimmt und eine Ressource als Ausgabe produziert.

## 3.3. Verifiable Credential (VC)

Bei Verifiable Credentials handelt es sich um digitalisierte Nachweise, welche man verschiedenen Instanzen zur Verifizierung seiner Identität vorlegen kann.

Mögliche Attribute von Verifiable Credentials können sein:

- Informationen über die ausstellende Behörde (z.B. eine Stadtverwaltung, eine nationale Behörde oder eine Zertifizierungsstelle) (**Pflicht, mindestens in Form der DID!**)
- Informationen über die Art des Nachweises (z.B. Personalausweis, Bachelorzertifikat, Krankenkassenkarte oder auch ein Fahrzeugschein)
- Informationen zu bestimmten Eigenschaften, die von der ausstellenden Behörde über die betreffende Person behauptet werden. (Geburtsdatum, Abschlussnote, Versichertennummer oder Fahrzeugklasse)

Im Gegensatz zu analogen Ausweisen ist es für den Verifizierer einfacher, die Authentizität und Integrität der Verifiable Credentials zu überprüfen, da er durch die Aufschlüsselung der DID des Austellers (Issuers) dessen Authentizität durch den Datenbankeintrag im Ledger nachweisen kann. Zum Beispiel muss ein Arbeitgeber (Verifier) eine vorliegende Bachelorurkunde manuell zuerst auf Echtheit prüfen, indem er die betreffende Hochschule anfragt. Durch die Nutzung von Verifiable Credentials wäre es möglich, die Überprüfung in nahezu Echtheitzeit vorzunehmen.

### 3.3.1. Architektur

Die folgende Darstellung stellt nochmal grafisch die Beziehung zwischen **Issuer**, **Holder** und **Verifier** im Hinblick auf Verifiable Credentials dar.

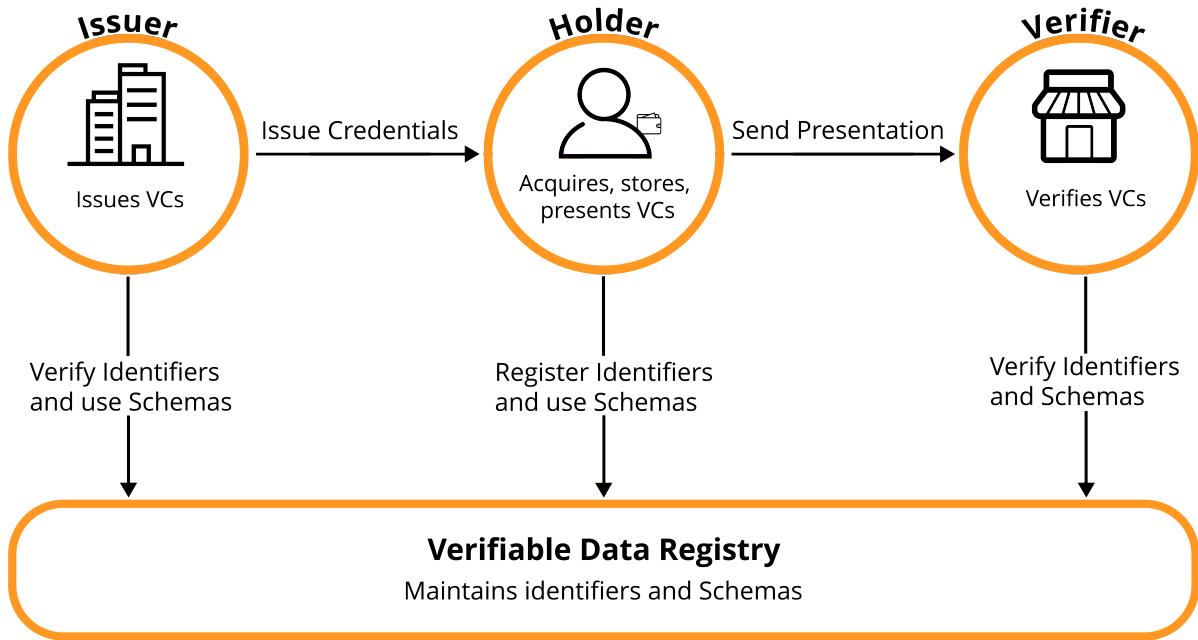


Abbildung: Überblick über das Ökosystem von VC, auf Basis (Referenz) von (<https://www.w3.org/TR/vc-data-model/diagrams/ecosystem.svg>) (Quelle)



Ein Schema beschreibt das Format und die Struktur von Verifiable Credentials. Es legt fest, welche Eigenschaften und Daten ein bestimmter Typ von Credentials enthalten sollte, beispielsweise ein Führerschein, ein Abschlusszeugnis oder ein Pass. Es dient als Standard, der die Interoperabilität und Übertragbarkeit der Credentials sicherstellt.

# Chapter 4. DIDComm

## 4.1. Was ist DIDComm?

DIDComm ist ein Kommunikationsprotokoll basierend auf dem Konzept SSI, das von der Decentralized Identity Foundation (DIF) entwickelt wurde, um sichere Peer-to-Peer-Interaktionen zwischen Anwendungen (bzw. Benutzern) mit dezentraler Identität (DID) zu ermöglichen. Es ist ein Satz an Funktionen, Vorgehensweisen und Methoden die spezifisch implementiert werden können. Es ist als offenes, sicheres, standardbasiertes Nachrichtenprotokoll so konzipiert, das es eine zuverlässige und vertrauenswürdige Kommunikationsverbindung zwischen Anwendungen herstellt. DIDComm bietet eine Möglichkeit für Anwendungen und Benutzer, miteinander zu interagieren und Daten, Dokumente und andere Informationen auszutauschen, während diese gleichzeitig kryptografisch verschlüsselt sind.



**Peer-to-Peer:** Eine Netzwerkarchitektur, bei der jeder Computer (oder Knoten) sowohl als Client, als auch als Server fungieren kann, so dass die Benutzer Ressourcen direkt miteinander teilen können, ohne einen zentralen Server zu benötigen.

## 4.2. DIDComm v2

DIDComm v2 ist eine Weiterentwicklung von DIDComm. Die Neuerungen von DIDComm v2 umfassen:

- Verbesserte Skalierbarkeit und Leistung durch eine bessere Verwaltung von Verbindungen und Datenströmen
- Unterstützung für neue Kryptografie-Standards wie Ed25519 und secp256k1
- Unterstützung für eine Vielzahl von Messaging-Kanälen, einschließlich IPFS und anderen Netzwerken
- Erweiterte Integrationsmöglichkeiten mit anderen Tools und Diensten
- Verringerte Komplexität und verbesserte Benutzerfreundlichkeit durch eine bessere Dokumentation und ein verbessertes Ökosystem

## 4.3. DIDComm Ablauf (vereinfacht!)

Bob und Alice wollen miteinander über ein DIDComm Protokoll kommunizieren. Folgender Ablauf (vereinfacht) entsteht dabei:

1. **DID Generierung:** Alice generiert eine DID.
2. **Verbindungsauftbau** Alice teilt ihre DID durch eine Out-Of-Band-Nachricht (sicherer Kommunikationskanal) dem Empfänger Bob mit. Dies kann z.B. in Form einer Email, SMS oder auch eines QR-Codes geschehen.
3. **Kommunikationsaufforderung:** Bob sendet eine Nachricht an Alice, die ihre Zustimmung für die Kommunikation benötigt, um eine sichere Verbindung herzustellen.

4. **Akzeptanz:** Alice bestätigt die Verbindung und gibt Bob Zugriff auf ihre Verifizierungsdokumente.
5. **Verifizierung:** Bob verwendet die Verifizierungsdokumente, um Alice zu authentifizieren und sicherzustellen, dass er mit der richtigen Person kommuniziert.
6. **Erfolgreiche Initialisierung:** Bob und Alice können nun sicher miteinander kommunizieren und Transaktionen durchführen die verschlüsselt sind, ohne dass eine zentrale Partei involviert ist.

#### 4.3.1. DID Erzeugung

Auch wenn sich das Protokoll nicht direkt mit der Erstellung oder Erzeugung von DIDs befasst, enthält das DIDComm-Protokoll jedoch bestimmte Funktionen und Mechanismen, die für die Verwaltung und Überprüfung von DIDs verwendet werden können, z.B. die Möglichkeit, die Echtheit von DID-Dokumenten und die Identität des Inhabers eines DID zu überprüfen.

Wenn beispielsweise eine Partei einen neuen DID erstellen möchte, kann sie das DIDComm-Protokoll verwenden, um ein DID-Dokument zu erstellen, das die erforderlichen Informationen wie den DID selbst, die mit dem DID verbundenen öffentlichen Schlüssel und alle anderen relevanten Daten enthält. Das DID-Dokument kann dann mit dem privaten Schlüssel des Besitzers signiert werden, was als Beweis für die Authentizität des Dokuments und die Identität des Besitzers dient.

Nach der Erstellung des signierten DID-Dokuments, kann es in einem dezentralen Register veröffentlicht werden, z.B. in einer Blockchain oder einem Distributed Ledger. Andere Parteien können dann das DIDComm-Protokoll verwenden, um die Authentizität des DID-Dokuments und die Identität des Eigentümers anhand der DID und der zugehörigen öffentlichen Schlüssel zu überprüfen.

Obwohl das DIDComm-Protokoll die Erstellung oder Erzeugung von DIDs nicht direkt abwickelt, enthält es Funktionen und Mechanismen, die zur Verwaltung und Überprüfung von DIDs verwendet werden können, die bei der Erstellung und zur Veröffentlichung neuer DIDs eingesetzt werden können.

# Chapter 5. Technischer Vergleich zwischen DIDComm und OIDC

DIDComm und OpenID Connect (OIDC) sind beides Protokolle für die Verwaltung von Identitäten, haben aber einige Unterschiede:

- Architektur:** DIDComm ist ein dezentralisiertes Protokoll für die Kommunikation zwischen DIDs (Decentralized Identifiers), während OIDC ein zentralisiertes Protokoll ist, welches von einem Identity Provider (IdP) bereitgestellt wird.
- Kontrolle über Identitätsdaten:** In DIDComm haben Benutzer die volle Kontrolle über ihre Identitätsdaten und Credentials, während bei OIDC der Identity Provider die Kontrolle hat.
- Authentifizierung:** OIDC verwendet eine Client-Server-Architektur und einen Auth-Code-Flow zur Authentifizierung von Benutzern, während DIDComm eine Peer-to-Peer-Kommunikation nutzt, um direkt zwischen den Endpunkten zu authentifizieren.
- Verifizierbare Credentials:** DIDComm unterstützt die Verwendung verifizierbarer Credentials und Präsentationen, um Identitäten zu verifizieren, während OIDC dies nicht unterstützt.
- Interoperabilität:** DIDComm nutzt ein dezentralisiertes Identitätsmodell und ist dadurch in der Lage, mit anderen dezentralisierten Systemen zu kommunizieren, während OIDC auf eine zentralisierte Infrastruktur angewiesen ist.

Zusammenfassend ist DIDComm ein dezentralisiertes Protokoll, das eine größere Kontrolle und Datensicherheit für Benutzer bietet und eine starke Unterstützung für verifizierbare Credentials hat, während OIDC ein zentralisiertes Protokoll ist, welches einfacher zu integrieren ist, jedoch weniger Kontrolle und Datensicherheit für Benutzer bietet.

| Kriterien           | DIDComm         | OIDC          |
|---------------------|-----------------|---------------|
| Entitäten           | 2*              | 3             |
| Basis-Technologie   | DID             | OAuth 2.0     |
| Datenverwaltung     | Dezentralisiert | Zentralisiert |
| SSI-Ansatz erfüllt? | Ja              | Nein**        |

\*korrekterweise benötigt es zu Beginn eine dritte Entität, welche die Credentials einmalig ausstellt. Im Nachhinein kann aber ein Peer-to-Peer Kommunikation stattfinden, ohne Einbezug eines Issuers.

\*\*OpenID4VC als Erweiterung von OIDC erfüllt diesen Ansatz.

# **Chapter 6. OIDC4/VC und OIDC4/VP**

Als Erweiterung des OIDC Protokolls wurde OIDC4/VC und OIDC4/CP eingeführt und um die Möglichkeit von Verifiable Credentials und Verifiable Presentations ergänzt. Diese wurden basierend auf dem SSI Konzept umgesetzt.

## **6.1. OIDC4/VC**

OIDC4/VC steht für "OpenID Connect for Verifiable Credentials". Es ist ein Protokoll, das Verifiable Credentials (VCs) in die Authentifizierungs- und Autorisierungsprozesse von OIDC (OpenID Connect) integriert.

In der Praxis wird OIDC4/VC durch die Verwendung von VCs als Claims in einem ID-Token implementiert, das von einem OpenID-Provider ausgestellt und an eine Anwendung geliefert wird. Die Anwendung kann dann die in den VCs enthaltenen Informationen verwenden, um eine authentische und verifizierbare Überprüfung der Identität des Benutzers durchzuführen.

Das Grundkonzept beschreibt die Bereitstellung der Identität in Form von Verifiable Credentials für den Resource Owner (Nutzer) durch den Identity Provider. Die verifizierte digitale Identität wird in der Wallet des Resource Owners gespeichert und ist für keine Dritte Partei öffentlich zugänglich. Somit liegt die Datenhoheit beim Nutzer, was dem Gedanken von SSI entspricht.

## **6.2. OIDC4/VP**

OIDC4/VD steht für "OpenID Connect for Verifiable Presentations". Es ist ein Protokoll, das es Anwendungen ermöglicht, Verifiable Presentations von einem Relying Party anzufordern und zu empfangen, die von einem Issuer ausgestellt wurden. Es ist eine Technologie, die darauf abzielt, eine einfache und sichere Übertragung von Verifiable Presentations im Kontext von SSI (Selbstsovereign Identity) zu ermöglichen.

Dieses Protokoll wird genutzt, um Verifiable Presentations zu verwalten und zu übertragen. Es nutzt OAuth2- und OpenID Connect-Mechanismen, um eine sichere Kommunikation zwischen Relying Party und Issuer zu gewährleisten.

OIDC4/VD setzt OAuth2-Flows um, um die Übertragung von Verifiable Presentations zu steuern und zu sichern. Es wird eine OAuth2-basierte Interaktion genutzt, bei der der Issuer einer Verifiable Presentation eine Zustimmungserklärung an die Relying Party sendet, die die Übertragung von Verifiable Presentations genehmigt. Die Relying Party kann dann eine Verifiable Presentation abrufen, indem sie eine Anfrage an den Issuer sendet. Die Interaktion endet, wenn die Relying Party die Verifiable Presentation erhält.

## **6.3. Selective Attribute Disclosure**

Die selektive Offenlegung von Attributen oder im engl. "Selective Attribute Disclosure" ist eine Funktion zur Verbesserung der Privatsphäre in souveränen Identitätssystemen (Self-Sovereign Identity), bei der eine Person sich dafür entscheiden kann, nur eine bestimmte Gruppe von Attributen oder Informationen über ihre Identität preiszugeben, anstatt alle ihre persönlichen

Daten teilen zu müssen.

"Selective Attribute Disclosure" wird in OIDC4VC und OIDC4VP durch das Konzept der "Requested Claims" realisiert. Ein Relying Party kann bei der Übermittlung von Verifiable Credentials nur die relevanten Attribute anfordern um eine Reduktion auf die notwendige Daten zu erreichen. OIDC4VC und OIDC4VP ermöglichen es außerdem einem Resource Owner, nur bestimmte Informationen, die für diese Interaktion erforderlich sind, auszuwählen und zu teilen, anstatt alle Informationen Preisgeben zu müssen.

Requested Claims werden in OIDC konkret durch den "claim"-Parameter im OIDC-Authorization-Request realisiert. Dieser Parameter enthält eine JSON-Struktur, die definiert, welche Claims z.B. für die Interaktion mit der Anwendung benötigt werden. Ein Beispiel (Ausschnitt aus der JSON-Struktur) für den Claim-Parameter könnte wie folgt aussehen:

```
{  
  "userinfo": {  
    "given_name": {"essential": true},  
    "family_name": {"essential": true}  
  }  
}
```

Dabei ist anzumerken, dass die Ablehnung essentieller Attribute ggf. zur nicht Nutzung eines Dienstes führen kann. Eine selektive Offenlegung gewährleistet nicht gleichzeitig auch den vollen Umfangsumfanges eines Services, jedoch müssen nur die tatsächlich notwendige Informationen und Daten übertragen werden, die zur eigentlichen Nutzung des Dienstes essentiell sind.



## 6.4. Zero-Knowledge Proof

Im Kontext der Self-Sovereign Identity (SSI) ist ein Zero-Knowledge Proof (ZKP) eine kryptografische Technik, die es einer Person ermöglicht, die Authentizität bestimmter Informationen gegenüber einer anderen Partei zu beweisen, ohne die Informationen selbst preiszugeben. In der SSI werden ZKPs verwendet, um eine Authentifizierung unter Wahrung der Privatsphäre zu ermöglichen und um überprüfbare Aussagen über die Identität einer Person zu machen.

Zum Beispiel möchte Alice, Bob beweisen, dass sie über 18 Jahre alt ist, aber sie möchte ihr Geburtsdatum geheim halten. Zu diesem Zweck verwendet sie einen Zero-Knowledge Proof. Der Prozess sieht folgendermaßen aus:

1. Alice sendet ihr Geburtsdatum zur Überprüfung an eine vertrauenswürdige Stelle. Die Behörde oder Institution bestätigt, dass Alice tatsächlich über 18 Jahre alt ist und fügt den Daten eine kryptografische Signatur hinzu, um zu beweisen, dass sie überprüft wurden.
2. Alice sendet die signierten Daten, die nun ein Hash ihres Geburtsdatums sind, an Bob.
3. Bob verwendet die Signatur der vertrauenswürdigen Stelle, um zu überprüfen, ob die Daten verifiziert wurden. Er kann jedoch Alices tatsächliches Geburtsdatum nicht sehen, da es gehasht

wurde und nur zur Bestätigung der Verifizierung der Daten verwendet werden kann.

Auf diese Weise kann Alice Bob beweisen, dass sie über 18 Jahre alt ist, ohne ihr tatsächliches Geburtsdatum preiszugeben und so einen Zero-Knowledge Proof erstellen.

OpenID4VC/VP unterstützt die Verwendung von Zero-Knowledge-Proofs. Ein gängiger Ansatz ist die Verwendung eines Zero-Knowledge Proof-Beweisprotokolls wie z.B. ZK-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge). Dabei handelt es sich um eine hocheffiziente Beweiskonstruktion, mit der der Besitz bestimmter Informationen, z.B. eines geheimen Schlüssels, bewiesen werden kann, ohne dass diese Informationen offengelegt werden und ohne dass es zu einer Interaktion zwischen dem Beweisführer und dem Überprüfer kommt.

ZK-SNARKs kann in OIDC wie folgt verwendet werden:

1. Der Benutzer erzeugt ein Paar aus privatem und öffentlichem Schlüssel, und der öffentliche Schlüssel ist in das OIDC-Identitäts-Token eingebettet.
2. Wenn der Benutzer einer vertrauenden Partei beweisen möchte, dass er über 18 Jahre alt ist, sendet er eine Anfrage an den OIDC-Anbieter, einschließlich seines öffentlichen Schlüssels und eines ZK-SNARK-Beweises, dass der Hash seines Geburtsdatums größer ist als das für die Transaktion erforderliche Mindestalter.
3. Der OIDC-Anbieter überprüft den ZK-SNARK-Beweis anhand des öffentlichen Schlüssels. Ist der Beweis gültig, stellt er ein OIDC-Identitäts-Token aus, das eine überprüfbare Behauptung enthält, die besagt, dass der Nutzer über 18 Jahre alt ist.
4. Die vertrauende Partei kann die Behauptung im Identitäts-Token mit Hilfe des öffentlichen Schlüssels und des ZK-SNARK-Beweises verifizieren und bestätigen, dass der Benutzer über 18 Jahre alt ist, ohne jemals sein tatsächliches Geburtsdatum zu erfahren.

Dies ist nur ein Beispiel dafür, wie Zero-Knowledge-Proofs im Rahmen von OIDC implementiert werden können. Es gibt neben den ZK-SNARKs noch mehrere andere Beweisprotokolle, die für die Implementierung von Zero-Knowledge Proofs verwendet werden können. Einige der anderen populären Beweisprotokolle sind:

- **ZK-STARKs (Zero-Knowledge Succinct Transparent Argument of Knowledge):** Eine Beweiskonstruktion, die ähnliche Funktionen wie ZK-SNARKs bietet, jedoch mit verbesserter Transparenz und Sicherheit.
- **Bulletproofs (Non-Interactive Zero-Knowledge Proofs for Confidential Transactions):** Eine hocheffiziente Beweiskonstruktion, mit der die Korrektheit von Berechnungen auf vertraulichen Daten bewiesen werden kann.
- **Sigma-Protokolle:** Eine Familie von Beweiskonstruktionen, die auf rechnerischen Annahmen beruhen und bei der Implementierung von Zero-Knowledge-Beweisen weit verbreitet sind.
- **Interaktive Zero-Knowledge (iZK)-Beweise:** Eine Beweiskonstruktion, die eine Interaktion zwischen dem Prover und dem Verifier beinhaltet, um die Gültigkeit des Beweises festzustellen.

Jedes dieser Beweisprotokolle hat seine eigenen Stärken und Schwächen, und die Wahl eines Beweisprotokolls hängt von den spezifischen Anforderungen des Anwendungsfalls ab, wie Effizienz, Sicherheit und Datenschutz. Wenn das Ziel beispielsweise darin besteht, ein Maximum an Effizienz und Skalierbarkeit zu erreichen, sind ZK-SNARKs die beste Wahl, während ZK-STARKs die

beste Wahl sind, wenn Transparenz und Sicherheit im Vordergrund stehen.

# Chapter 7. Proof-of-Concept (SSI)

Um das Konzept von SSI in seiner Machbarkeit und seinem Potenzial zur Umsetzung zu belegen, wollen wir eine praktische Demonstration dieser Technologie umsetzen. Es handelt sich dabei um einen einfachen experimentellen Ansatz, um die Funktionsfähigkeit dieses Konzeptes zu testen. Der Zweck dieses PoC ist es, ein besseres Verständnis für die technischen und geschäftlichen Auswirkungen von SSI zu gewinnen.

Dazu wird das Hyperledger Aries Framework verwendet. Es ist eine Bibliothek von Protokollen, Formaten und Komponenten, die es Entwicklern erleichtern, Anwendungen für Verifiable Credentials und die Übertragung von Informationen in einer SSI-Umgebung zu erstellen. Das Framework basiert auf dem DIDComm Protokoll und beinhaltet dessen technologischen Umsetzung.

Hyperledger Aries unterstützt zudem eine Vielzahl von Identity-Systemen, einschließlich Decentralized Identifier (DID) und Verifiable Credentials (VC). Es bietet eine einheitliche API für die Übertragung von Daten zwischen verschiedenen Identity-Systemen, die Interoperabilität und Austauschbarkeit gewährleisten. Das Hyperledger Aries-Framework bietet auch Funktionen wie Identitätsmanagement, Übertragung von Daten, Schlüsselverwaltung und mehr, die für die Entwicklung von Anwendungen erforderlich sind und auf SSI basieren.

Um das Ganze anschaulicher zu demonstrieren, verwenden wir dazu die [Alice Gets a Mobile Agent!](#) Demo von Hyperledger Aries. In dieser Demo werden wir zwei Standardszenarien simulieren.

## 7.1. Szenarien

### (1.) Das Ausstellen von Credentials

In einem ersten Szenario baut Alice eine Verbindung zu Ihrer Hochschule durch die Annahme einer Einladung in Form eines QR-Codes auf. Im zweiten Schritt sendet ihr die Hochschule Verifiable Credentials (VC) zu, welche das Datum ihres Abschlusses und das Studienfach beinhalten. Das Szenario ist im Folgenden als Ablauf grafisch dargestellt.

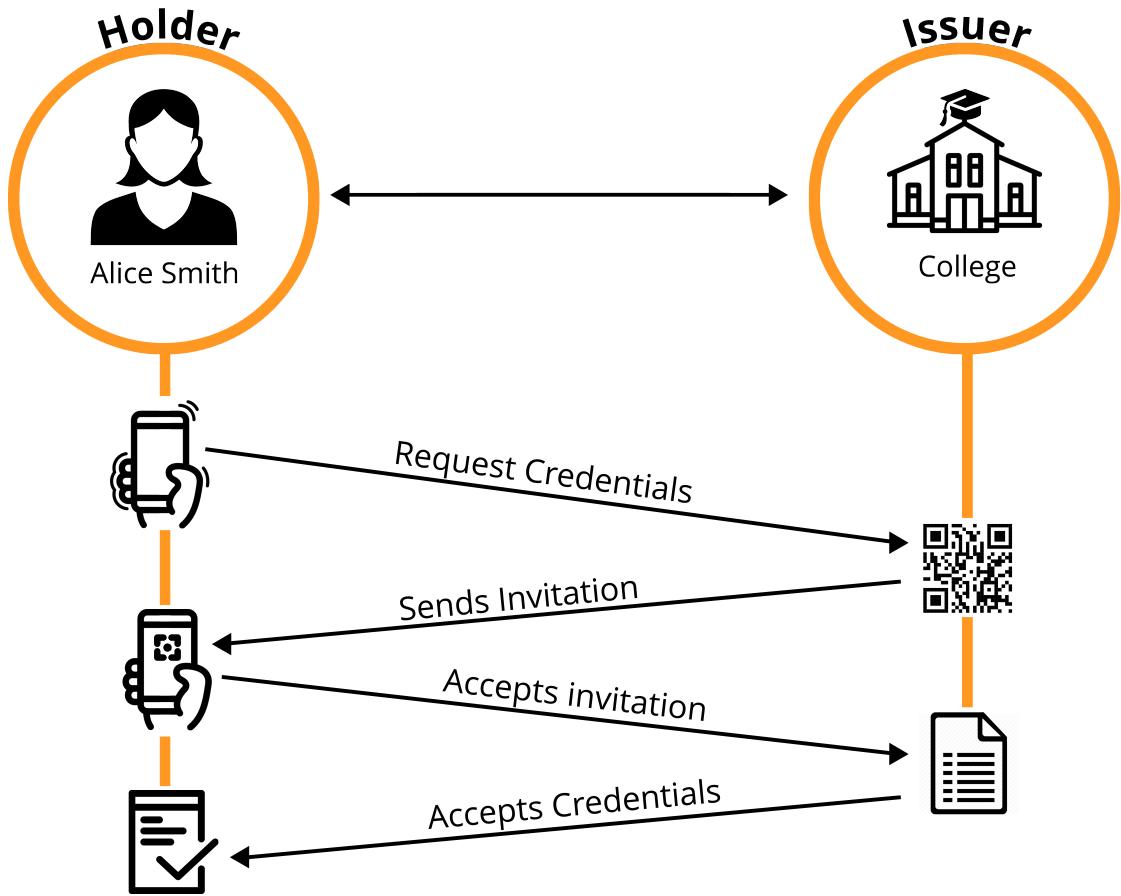


Abbildung: Kommunikationsfluss Diagramm für das Ausstellen von Credentials (Eigenstellung)

## (2.) Verifizierung der Credentials

Im zweiten Szenario möchte sich Alice bei einem neuen Arbeitgeber bewerben. Dazu muss sie nachweisen, dass sie tatsächlich über einen Abschluss in Mathematik verfügt. Ihr Arbeitgeber fordert Sie dazu auf, ihre Credentials zuzusenden. Alice kann den Umfang der geforderten Daten vorab einsehen und diese bestätigen. Ein **Selective Attribute Disclosure** wird in dieser Demo nicht demonstriert.

Diese Credentials werden wiederrum durch den Arbeitgeber, mit dem Abgleich der DID der Hochschule, als Issuer der Credentials, aus dem Verifiable Data Register verifiziert und bestätigt.

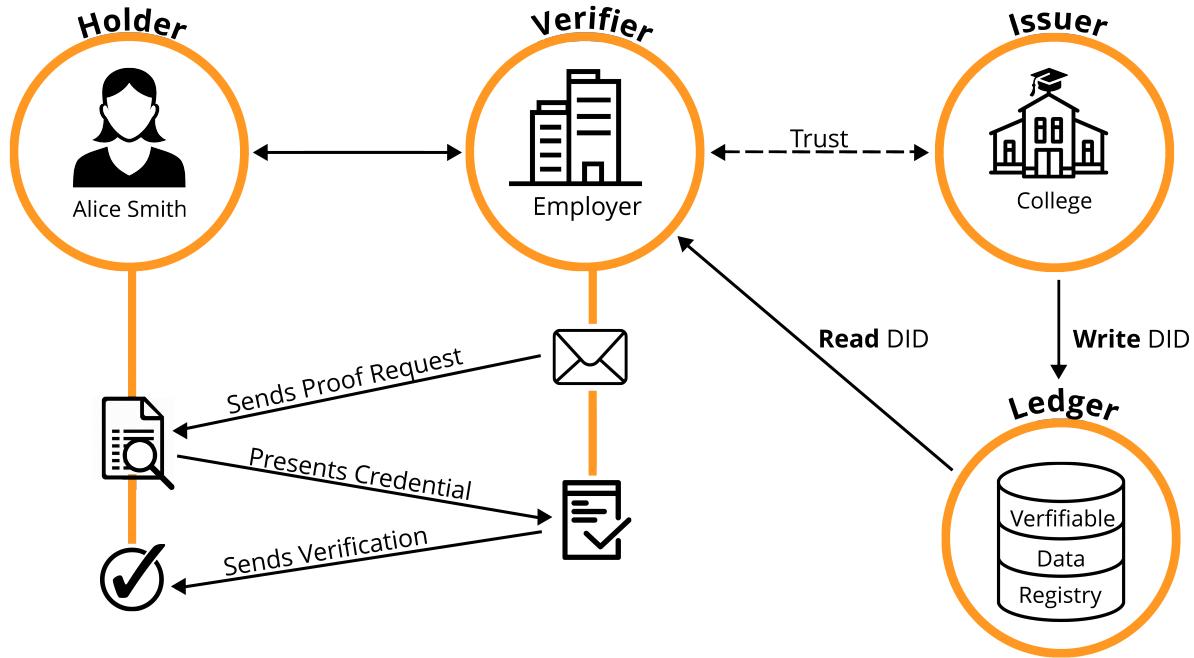


Abbildung: Kommunikationsfluss Diagramm für die Verifizierung der Credentials (Eigenerstellung)

## 7.2. Komponenten

Um beide Szenarien technisch umsetzen zu können, benötigen wir mehrere Komponenten, die wir erstellen und simulieren müssen.

### Alice

Damit wir Alice simulieren können, verwenden wir ein geeignetes Smartphone (iPhone 13) auf dem ein Wallet installiert ist, welches über seinen eigenen besitzt Agent.

### Wallet/Agent

Beim Wallet haben wir uns für das **esatus Wallet** entschieden. Esatus ist eine Eigenentwicklung der esatus AG, welche ein Treiber von Self-Sovereign Identity (SSI) in Deutschland und einer der globalen Technologieführer ist. Das Wallet bietet für die Demonstration folgenden Funktionsumfang:

1. Das Scannen eines QR-Codes (Einladungen der Hochschule)
2. Verbindungsauftbau zu einem anderen Agent (Hochschule/Arbeitgeber)
3. Die Verwaltung von Verbindungen (Speicherung)
4. Die Möglichkeit ein Test-Ledger auszuwählen (BCGov Test Ledger/BCovrin)
5. Das Halten, Speichern und Vorzeigen von Credentials

→ Darüber hinaus werden auch noch weitere Funktionen angeboten, die wir im Umfang dieser Demonstration jedoch nicht weiter betrachten werden.

### Hochschule

Damit wir unseren Issuer, die Hochschule, simulieren können, müssen wir diese als Agent lokal

installieren und gleichzeitig hosten. Der Agent verfügt über folgende Funktionen:

- Issue Credential
- Send Proof Request
- Send **Connectionless** Proof Request
- Send Message
- Create New Invitation
- Revoke Credential
- Publish Revocations
- Toggle tracing in credential/proof exchange

→ Für die Simulation verwenden wir hauptsächlich die ersten zwei Funktionen.

## Arbeitgeber

Der Arbeitgeber wird auf die gleiche Weise wie die Hochschule simuliert. Wir beschränken uns dabei auf **EINEN localhost**, welcher gleichzeitig Arbeitgeber und Hochschule simuliert. In einem Real Case Szenario wären diese zwei getrennte Parteien. Der Issuer (Hochschule) und der Verifier (Arbeitgeber) der im OIDC Kontext die Relying Party darstellt.

## ngrok

Ngrok ist ein Tunneling-Service, mit dem Entwickler lokale Anwendungen und Dienste über das Internet verfügbar machen können, ohne sie direkt bereitstellen zu müssen. Ngrok ermöglicht es, eine lokale Anwendung über eine öffentliche URL erreichbar zu machen, was beispielsweise für die Überprüfung von Anwendungen durch andere Benutzer oder das Debugging von Problemen nützlich sein kann. Wir verwenden Ngrok um ein Tunneling-Service nach außen für unseren localhost Agent für die "**Alice Gets a Mobile Agent!**" Demo zu bauen.

## BCovrin (Ledger)

[BCoverin](#) ist ein Test-Ledger (Verifiable Data Register), welches bei der Initialisierung unserer **"Alice Gets a Mobile Agent"** Anwendung als Ledger dient.

## Indy Tails Server

Indy Tails ist ein Projekt innerhalb der Hyperledger Indy Community, das sich auf die Bereitstellung von Anonymität und Datenschutz in der Identitätsbranche konzentriert. Ein [Indy Tails Server](#) ist ein Server, der Teil des Indy Tails-Projekts ist und Anwendern Anonymität bei der Verwendung von Hyperledger Indy-basierten Identitätslösungen bietet. Es handelt sich dabei um einen anonymen Proxy-Server, der eine Verbindung zwischen Anwendern und einer Hyperledger Indy-basierten Identitätslösung herstellt. Durch die Verwendung eines Indy Tails-Servers können Anwender ihre Identitätsdaten und ihre Online-Aktivitäten schützen, indem sie ihre echte Identität verborgen halten und stattdessen eine anonyme Identität verwenden. Zudem werden dadurch datenschutzfreundliche Widerrufsprozesse (Revocations) abgebildet.

## Ergänzung:

Im Kontext des Hyperledger Indy-Projekts bezieht sich der Begriff Revocation (Widerruf) auf den Prozess der Ungültigmachung eines zuvor ausgestellten Credentials oder den Widerruf des Zugriffs

auf ein bestimmtes Attribut innerhalb eines Credentials. Hyperledger Indy verwendet dazu eine dezentralisierte Architektur und die Indy Tails-Server spielen eine Schlüsselrolle bei datenschutzfreundlichen Widerrufsprozessen. Indy Tails-Server sind für die Speicherung von Sperrinformationen und den sicheren Zugriff auf diese Informationen für Sperrprüfungen verantwortlich.

Bei einem typischen Widerrufsprozess widerruft der Aussteller eines Credentials ein bestimmtes Attribut oder das gesamte Credential. Die Widerrufsinformationen werden in einer "tails" genannten Datenstruktur gespeichert, die von den Indy Tails-Servern verwaltet wird. Die Tails-Datei enthält die Widerrufsinformationen für alle vom Aussteller ausgestellten Berechtigungsnachweise und ist durch kryptografische Signaturen und Hash-Ketten geschützt, um ihre Integrität und Authentizität zu gewährleisten.

Wenn eine vertrauende Partei die Gültigkeit eines Berechtigungsnachweises überprüfen muss, fragt sie die Indy Tails-Server nach den neuesten Widerrufsinformationen ab. Die Indy Tails-Server geben die entsprechenden Informationen aus der Tails-Datei zurück, so dass die vertrauende Partei feststellen kann, ob der Berechtigungsnachweis oder ein bestimmtes Attribut innerhalb des Berechtigungsnachweises widerrufen wurde.

Diese dezentrale Architektur und die Verwendung von Indy Tails-Servern ermöglichen einen datenschutzfreundlichen Widerrufsprozess, der die Privatsphäre der Benutzer schützt und gleichzeitig den vertrauenden Parteien die notwendigen Informationen zur Verfügung stellt, um die Gültigkeit von Berechtigungsnachweisen zu überprüfen. Die Tails-Datei kann von Datenschutzbehörden geprüft werden, um sicherzustellen, dass die Widerrufsinformationen korrekt sind und die Privatsphäre der Nutzer geschützt ist.

⇒ Daraus ergibt sich der folgender technische Aufbau

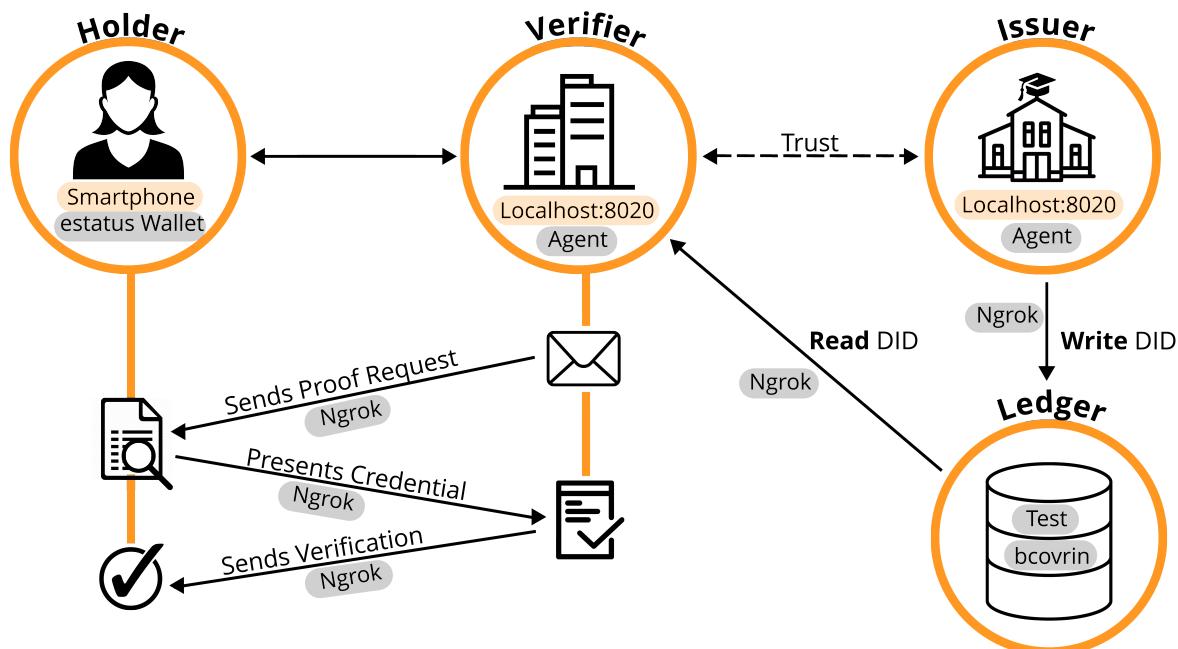


Abbildung: Technischer Aufbauplan für die PoC Demonstration (Eigenerstellung)

## 7.3. Technische Umsetzung

### (1.) Initialisierung Indy Tails Server

Zuerst muss der Indy Tails Server initialisiert werden. Dazu wird ein Dockerimage gebaut, welches wiederum später von unserem lokalen Agent als Serverinstanz genutzt werden kann.

### (2.) Aufbau einer Bridge (ngrok)

Um später lokal mit unserem Hochschul Agenten nach außen zu Alice kommunizieren zu können, müssen wir vorab über ngrok eine http Bridge aufbauen. Dabei findet ein Portmapping des *localhost* auf eine öffentliche URL statt.

```
https://bf6a-2003-fa-af0a-25ad-a0ee-11e7-2b91.eu.ngrok.io -> http://localhost:8020
```

### (3.) Initialisierung des lokalen Agents

Über das Tails Netzwerk wird ein Agent initialisiert, welcher wiederum BCovrin als Test Ledger nutzt. Der folgende Befehl zeigt die detaillierte Initialisierung des Agenten.

```
TAILS_NETWORK=docker_tails-server LEDGER_URL=http://test.bcovrin.vonx.io ./run_demo  
faber --aip 10 --revocation --events
```

#### Auflistung der einzelnen Parameter:

##### **TAILS\_NETWORK=docker\_tails-server**

Legt die Art des Tails-Netzwerks fest, auf dem das Demo ausgeführt wird. Hier wird "docker\_tails-server" angegeben, so dass ein Docker-Container als Tails-Server verwendet wird.

##### **LEDGER\_URL=http://test.bcovrin.vonx.io**

Legt die URL des Ledger fest, mit dem Faber kommunizieren soll. Hier wird eine Test-URL angegeben, die auf einen vonx.io-Ledger-Server verweist.

##### **/run\_demo**

Ist der Befehl, der die Demo startet.

##### **faber**

Ist das Argument, das an den Befehl übergeben wird und das Faber-System angibt, welches Teil der Demos sein soll.

##### **--aip 10**

Gibt an, dass die Demo gemäß den Anforderungen des AIP (Agent Interaction Protocol) 10 ausgeführt werden soll.

##### **--revocation**

Gibt an, dass die Demo die Verwendung von Widerrufskomponenten beinhalten soll.

## -events

gibt an, dass die Demo die Überwachung von Ereignissen unterstützen soll.

## (4.) Installation des esatus Wallets

Zuletzt müssen wir noch ein Wallet auf unserem Smartphone installieren. Die Installation findet klassisch über den App-Store oder Play-Store statt. Dabei sind keine Spezifischkeiten zu beachten. Nach erfolgreicher Installation muss das Wallet auf das BCGov Test Ledger (BCovrin) umgestellt werden.

# 7.4. Ausführung

Nachdem wir die technischen Grundlagen für unsere Demonstration geschaffen haben, wollen wir zuletzt einen Auszug aus der "Alice Gets a Mobile Agent!" Demo zeigen.

## (1.) Start der ngrok Bridge

```
ngrok
Add Single Sign-On to your ngrok dashboard via your Identity Provider: https://ngrok.com/dashSSO

Session Status          online
Session Expires        1 hour, 59 minutes
Terms of Service       https://ngrok.com/tos
Version                3.1.0
Region                 Europe (eu)
Latency                -
Web Interface          http://127.0.0.1:4040
Forwarding             https://bf6a-2003-fa-af0a-25ad-a0ee-11e7-2b91-2941.eu.ngrok.io -> http://localhost:8020

Connections            ttl     opn      rt1      rt5      p50      p90
                        0       0       0.00    0.00    0.00    0.00
```

Abbildung: Ausschnitt aus dem Session Status von Ngrok (Screenshot)

Zu erkennen ist vor allem das Mapping der lokalen Adresse `localhost:8020` auf eine öffentlich zugängliche URL.

## (2.) Provisionierung eines Agents und einer Wallet

```
ngrok tails service name [ngrok-tails-server]
Fetching endpoint from ngrok service
Fetched ngrok tails server endpoint [http://4e0f-217-237-115-37.ngrok.io]
Starting [faber] agent with args [--port 8020 --aip 10 --revocation]
Initializing demo agent faber with AIP 10 and credential type indy

#1 Provision an agent and wallet, get back configuration details
Started webhook listener on port: 8022
Faber  | Registering faber.agent ...
using ledger: http://test.bcoverin.vonx.io/register
Faber  |nym_info: {'did': 'EcagwahExWPRRmJmCeFA7e', 'seed': 'd_00000000000000000000000000000000133130', 'verkey': '8RMCzwEDjRU4BWLz9dM9Gee9z7VGFw3eAKJbpwD6phJn'}
Faber  | Registered DID: EcagwahExWPRRmJmCeFA7e
Created public DID
```

Abbildung: Ausschnitt aus der Agent-Initialisierung über die Kommandozeile (Screenshot)

Als nächstes wird der Tails Server gestartet und die Endpunkte des Servers mit ngrok gefetcht, so dass dieser einen neuen Endpunkt für die Kommunikation nach außen mit ngrok als Verbindungsbrücke erhält.

Danach werden wiederum Agent und Wallet provisioniert, indem eine neue DID erzeugt wird, welche wiederum im Test Ledger (test.bcoverin) registriert wird.

## (3.) Erfolgreiche Provisionierung

```

Faber | :::::::::::::::::::::|:::
Faber | :: faber.agent ::|:::
Faber | :: ::|:::
Faber | :: Inbound Transports: ::|:::
Faber | :: - http://0.0.0.0:8020 ::|:::
Faber | :: ::|:::
Faber | :: Outbound Transports: ::|:::
Faber | :: ::|:::
Faber | :: - http ::|:::
Faber | :: - https ::|:::
Faber | :: ::|:::
Faber | :: Public DID Information: ::|:::
Faber | :: ::|:::
Faber | :: - DID: EcagwahExWPRRmJmCeFA7e ::|:::
Faber | :: ::|:::
Faber | :: Administration API: ::|:::
Faber | :: ::|:::
Faber | :: - http://0.0.0.0:8021 ::|:::
Faber | :: ::|:::
Faber | :: :: ver: 1.0.0-rc1 ::|:::
Faber | :: ::|:::
Faber | :: Listening... ::|:::
Faber | :: ::|:::
Startup duration: 15.79s
Admin URL is at: http://host.docker.internal:8021
Endpoint URL is at: https://bf6a-2003-fa-af0a-25ad-a0ee-11e7-2b91-2941.eu.ngrok.io

```

*Abbildung: Ausschnitt aus der Agent-Konfiguration über die Kommandozeile (Screenshot)*

Nach erfolgreicher Provisionierung werden die Daten für den Agent und dem Wallet einschließlich der öffentlichen DID Information ausgegeben und gespeichert.

#### (4.) Schema und weitere Einstellungen

```

Schema:
{
  "sent": {
    "schema_id": "EcagwahExWPRRmJmCeFA7e:2:degree schema:38.96.33",
    "schema": {
      "ver": "1.0",
      "id": "EcagwahExWPRRmJmCeFA7e:2:degree schema:38.96.33",
      "name": "degree schema",
      "version": "38.96.33",
      "attrNames": [
        "name",
        "timestamp",
        "birthdate_dateint",
        "degree",
        "date"
      ],
      "seqNo": 668113
    }
  },
  "schema_id": "EcagwahExWPRRmJmCeFA7e:2:degree schema:38.96.33",
  "schema": {
    "ver": "1.0",
    "id": "EcagwahExWPRRmJmCeFA7e:2:degree schema:38.96.33",
    "name": "degree schema",
    "version": "38.96.33",
    "attrNames": [
      "name",
      "timestamp",
      "birthdate_dateint",
      "degree",
      "date"
    ],
    "seqNo": 668113
  }
}

Schema ID: EcagwahExWPRRmJmCeFA7e:2:degree schema:38.96.33

```

*Abbildung: Ausschnitt aus dem Ladeprozesses des Schemas über die Kommandozeile (Screenshot)*

Im Anschluss wird das Schema geladen, welches später für die Erzeugung der Credentials verwendet wird. Es werden zusätzliche Konfigurationen abgeschlossen, die in dieser Darstellung jedoch nicht abgebildet sind.

## (5.) Erstellung einer Einladung

```
Generate invitation duration: 0.08s
Use the following JSON to accept the invite from another demo agent. Or use the QR code to connect from a mobile agent.
Invitation Data:
{
  "@type": "did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/connections/1.0/invitation",
  "@id": "8248f2fa-892c-4ec9-95cc-8574d4c29407",
  "label": "faber.agent",
  "serviceEndpoint": "https://bf6a-2003-fa-a0a-25ad-a0ee-11e7-2b91-2941.eu.ngrok.io",
  "recipientKeys": ["HCR1NCYHTAHnmrM21jgyX9uEcopk6zGjSNfssttRBHc"]
}
```



Abbildung: Digitale Einladung (QR-Code) für einen Verbindungsauflauf zwischen Hochschule und Alice (Screenshot)

Nach erfolgreichen Abschluss aller Konfigurationen wird eine einmalige Einladung erstellt, die über einen mobilen Agenten (vorzugsweise den von Alice) gescannt werden kann. Danach hat Alice die Möglichkeit die Verbindung anzunehmen oder abzulehnen.

## (6.) Aufbau einer Verbindung zwischen Alice und dem Hochschul Agenten

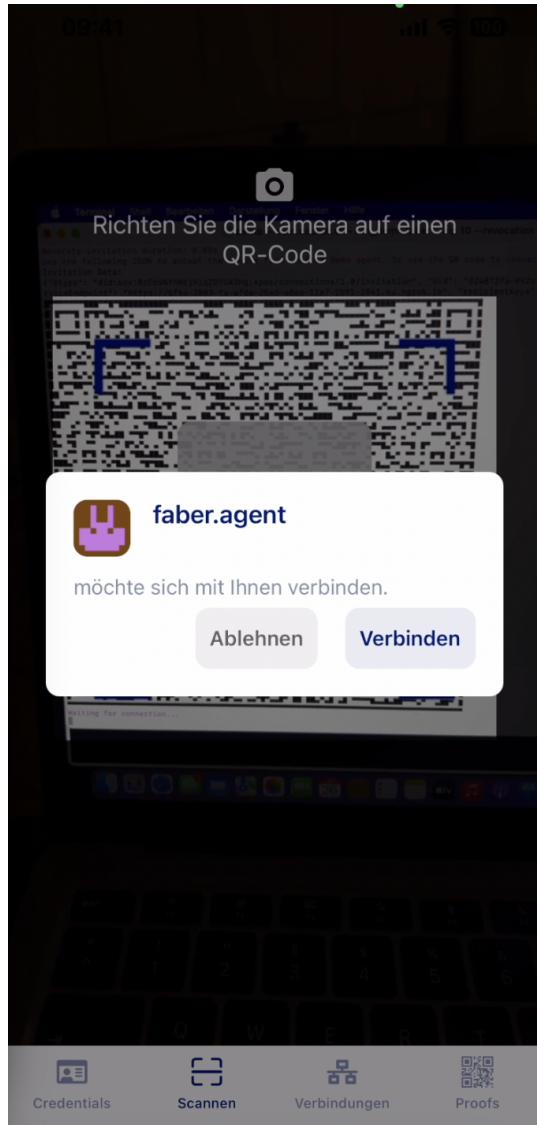


Abbildung: Annahme der Einladung aus der Sicht von Alice über ihr Wallet/Agent (Screenshot)

Nachdem Alice die Einladung angenommen hat, wird eine Peer-to-Peer Verbindung zwischen dem Agenten von Alice und dem Agenten der Hochschule hergestellt.

#### (7.) Austellung der Credentials

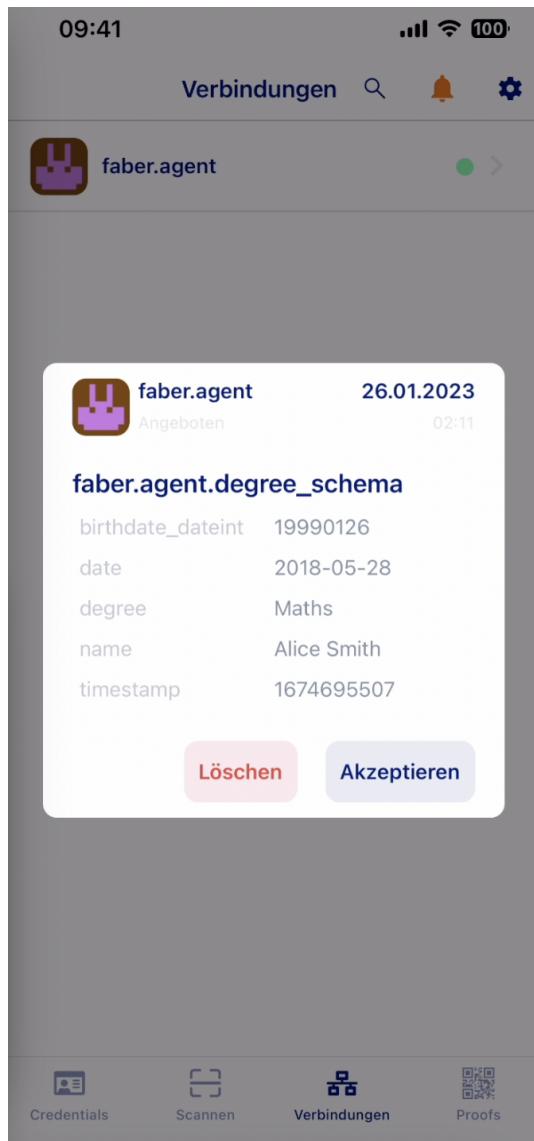


Abbildung: Annahme der Credentials aus der Sicht von Alice über ihr Wallet/Agent (Screenshot)

Sowohl Alice hat nun die Möglichkeit Credentials anzufordern, als auch die Hochschule besitzt die Möglichkeit diese eigenständig auszustellen. Dabei kann Alice die Korrektheit dieser Daten überprüfen und diese auch jederzeit ablehnen. Alice ist nicht dazu gezwungen die Credentials auch annehmen zu müssen. Damit behält Alice stets ihre Datenhöheit.

#### (8). Speicherung der Credentials

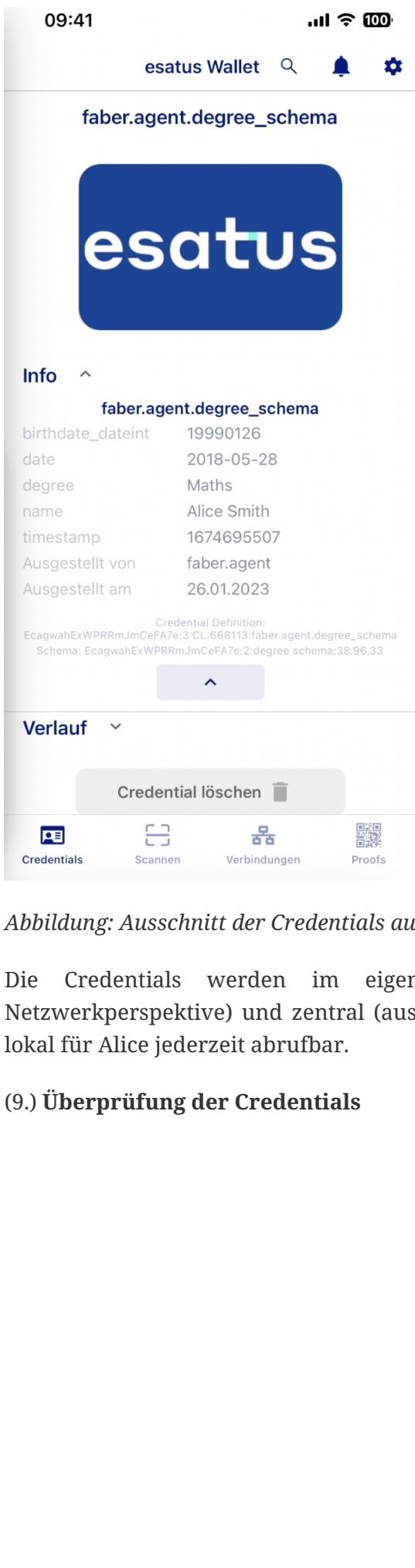


Abbildung: Ausschnitt der Credentials aus dem Wallet von Alice (Screenshot)

Die Credentials werden im eigenen persönlichen Wallet von Alice dezentral (aus Netzwerkperspektive) und zentral (aus ihrer eigenen Perspektive) gespeichert und sind dadurch lokal für Alice jederzeit abrufbar.

#### (9.) Überprüfung der Credentials

### Proof of Education

faber.agent erbittet die folgenden Informationen von Ihnen:

name

Alice Smith

esatus

date

2018-05-28

esatus

degree

Maths

esatus

birthdate\_dateint <= 2005...

19990126

esatus

Bitte nehmen Sie unsere Datenschutzerklärung zur Kenntnis (<https://esatus.com/dataprivacy>), besonders Abschnitt 2.2. Mit "Senden" bestätigen Sie, dass die ausgewählten Daten übertragen werden an faber.agent.

**Abbrechen** **Senden**

*Abbildung: Anforderung zur Übermittlung zur Überprüfung der Credentials durch die Hochschule aus der Sicht von Alice über ihr Wallet/Agent (Screenshot)*

Wichtig ist dabei vor allem die Überprüfung der Credentials. Dies geschieht durch die Zertifizierung des Ausstellers. In diesem Demo Beispiel ist Aussteller (Issuer bzw. die Hochschule) und Verifizierer (Verifier bzw. der Arbeitgeber) ein und der selbe Agent. In einem Real-Case-Szenario findet die Verifizierung durch die Überprüfung der DID des Ausstellers im Daten Register (Ledger) statt.

# Chapter 8. Beurteilung

Das PoC stellt eine vereinfachte Implementierung von SSI dar. Das Grundkonzept wird damit veranschaulicht, jedoch werden nicht alle Parteien über verschiedene Agenten hinweg simuliert. Auch wird keine detaillierte Überprüfung der DID im Ledger aufgeführt, was wiederum ein wesentlicher Bestandteil für die Verifizierung von Credentials ist. Diese Vorgehensweise bleibt in einer Blackbox verborgen. Dennoch kann die Demo als ein Grundverständnis für die Vorgehensweise mit SSI genutzt werden.

Über das Hyperledger Aries Framework lassen sich weitere spezifischere Anforderungen implementieren, programmieren, erweitern und auf Real-Case-Szenarien anpassen. Dazu braucht es jedoch eine intensive Einarbeitung in das DIDComm Protokoll, ein tiefes Verständnis für SSI und einen starken technologischen Bachground. Die Implementierung von SSI ist daher wesentlich aufwendiger als das standardmäßige Arbeiten mit OICD, jedoch lohnen sich die Vorteile aus verschiedenen Perspektiven.

Da nicht nur Personen von SSI profitieren können, sondern gleichermaßen auch Organisationen oder Objekte einen Vorteil durch SSI erhalten, wollen wir im letzten Abschnitt ein Szenario je Gruppe vorstellen, welches die Implementierung von SSI sinnvoll macht.

## 8.1. SSI für Personen

Die wesentlichen Vorteile für die Implementierung von SSI für Personen haben wir bereits ausführlich erläutert. Zusammenfassend lässt sich jedoch sagen, dass durch die Implementierung von SSI, Nutzer die Möglichkeit erhalten, ihre eigenen Daten eigenständig zu speichern, zu verwalten und nachweisen zu können, ohne dass es die wiederholte Interaktion oder die Speicherung durch eine Dritte Partei benötigt. Ein solches Szenario wird in unserem PoC beschrieben.

## 8.2. SSI für Organisationen

Auch Unternehmen oder Organisationen können von SSI profitieren. Ein mögliches Szenario wäre z.B. die Lieferanten Beziehung zweier Unternehmen, die zum derzeitigen Zeitpunkt noch keine Geschäftsbeziehung besitzen. Wie lässt sich digital die Echtheit und Glaubhaftigkeit eines Unternehmens, dass z.B. im Ausland sitzt, überprüfen?

Auch können Unternehmen durch SSI ihre Echtheit, z.B. beim Online-Shopping auf weniger bekannten Webseiten oder Plattformen, nachweisen und somit Vertrauen gegenüber potentiellen Kunden gewinnen.

Ein weiterer Anwendungsfall wäre, dass dadurch eine Möglichkeit entsteht, wie Rechte von Unternehmen auf natürliche Personen übertragen werden können. Ein Bankvertreter kann sich somit z.B. gegenüber einen Clienten oder als juristischer Vertreter seiner Institution ausweisen.

## 8.3. SSI für Objekte

Das SSI Konzept lässt sich auch auf Objekte übertragen. Wann hat die letzte TÜV Prüfung

stattgefunden? Wie ist die Lieferkette einer Maschine aufgebaut? Woher stammen die Rohstoffe oder Materialien? Durch SSI lassen sich all diese Punkte eindeutig durch verifizierbare Credentials nachweisen.

## 8.4. Kritiken

Zuletzt wollen wir einen kritischen Blick auf SSI und mögliche Schwachstellen und Szenarien werfen, die Kritikpunkte aufwerfen.

### Der Verlust des Wallets

Ein Wallet, das für SSI und DIDComm verwendet wird, speichert die Verifikations- und Kryptoschlüssel einer DID (Decentralized Identifier). Der Verlust des Wallets bedeutet in diesem Kontext den Verlust des Zugangs zu den Identitätsinformationen und den damit verbundenen Daten und Aktivitäten, die damit verknüpft sind. Dies kann für eine Person oder ein Unternehmen erhebliche Folgen haben, insbesondere wenn es sich um sensiblen oder vertraulichen Informationen handelt.

Dies ist ein möglicher Kritikpunkt von SSI und DIDComm, da der Verlust des Wallets eine Barriere für den Zugang zu den Identitätsinformationen darstellt und die Datensicherheit gefährdet.

### Weitere mögliche Kritikpunkte und Schwachstellen sind:

**Interoperabilitätsprobleme:** Obwohl es einen gemeinsamen Standard für DIDs und DIDComm-Nachrichten gibt, kann es Probleme bei der Interoperabilität zwischen verschiedenen Implementierungen und Plattformen geben.

**Datenschutzprobleme:** Da DIDs und DIDComm-Nachrichten über ein dezentralisiertes Netzwerk übertragen werden, ist es schwierig, die Datenschutzrichtlinien und -anforderungen für verschiedene Anwendungen und Gerichtsbarkeiten einzuhalten und zu überwachen.

**Skalierbarkeitsprobleme:** Die Verarbeitung von DIDComm-Nachrichten und die Speicherung von DID-Informationen kann eine Herausforderung für dezentralisierte Netzwerke sein, die häufig nicht so leistungsstark wie zentrale Netzwerke sind. Gründe dafür sind:

1. Umfang der Nachrichten: Obwohl es sich bei der Verarbeitung von DIDComm-Nachrichten um ein P2P-Verfahren handelt, kann das Volumen der zu verarbeitenden Nachrichten beträchtlich sein, insbesondere in dezentralen Netzen mit einer großen Anzahl von Knoten. Dies kann zu Problemen bei der Skalierbarkeit führen, da jeder Knoten in der Lage sein muss, die DID-Informationen zu verarbeiten und speichern zu können.
2. Komplexität der Nachrichten: DIDComm-Nachrichten können komplexe Datenstrukturen enthalten, wie z.B. Verifiable Credentials und Zero-Knowledge-Proofs, deren Validierung und Speicherung erhebliche Rechenleistung erfordert.
3. Begrenzte Ressourcen: Dezentrale Netze bestehen in der Regel aus Knoten mit begrenzten Ressourcen wie Speicher und Verarbeitungsleistung, was bei der Verarbeitung und Speicherung großer Mengen von DID-Informationen eine Einschränkung darstellen kann.
4. Schutz der Daten: In einem dezentralen Netzwerk speichert jeder Knoten seine eigene Kopie der DID-Informationen, was zu Problemen mit dem Datenschutz führen kann, wenn die

Informationen nicht ordnungsgemäß geschützt sind. Dies kann eine Herausforderung darstellen, wenn es um die Verarbeitung und Speicherung sensibler Daten, wie z.B. persönlicher Identitätsdaten geht.

Um diesen Herausforderungen zu begegnen, müssen dezentrale Netzwerke effiziente und sichere Methoden zur Verarbeitung und Speicherung von DID-Informationen implementieren, die wiederum Rechenleistung benötigen und somit aufwendiger zu skalieren sind. So können Knoten beispielsweise Techniken zur Aufteilung und Partitionierung von Daten einsetzen, um die Verarbeitungs- und Speicherlast zu verteilen und kryptografische Algorithmen verwenden, um sensible Informationen zu schützen. Darüber hinaus können die Knoten die Blockchain-Technologie nutzen, um ein sicheres und ein fälschungssicheres Main Ledger (Hauptbuch) für DID-Informationen zu erstellen.



Die reine Peer-to-Peer (P2P) Kommunikation in Form von einzelnen Standard Nachrichten ist dagegen als einzelner Prozess betrachtet jedoch leistungseffektiv, da zur Verarbeitung der Informationen keine Drittpartei notwendig ist und die Verschlüsselung auf ein einfaches symmetrisches Verfahren stattfinden kann, welches deutlich weniger Rechenleistung als die asymmetrische Verschlüsselung in Anspruch nimmt.

Insgesamt hängt die Skalierbarkeit dezentraler Netzwerke für die Verarbeitung von DIDComm-Nachrichten und die Speicherung von DID-Informationen von der spezifischen Implementierung und den getroffenen Abwägungen zwischen Effizienz, Sicherheit und Datenschutz ab.

**Benutzerfreundlichkeitsprobleme:** Es kann schwierig sein, Benutzern ein einfaches und intuitives Verständnis von DIDs und DIDComm zu vermitteln, und die Verwendung von Wallets und anderen SSI-Tools kann für viele Benutzer eine Herausforderung darstellen

# Chapter 9. Quellenverzeichnis

1. [https://www.windley.com/archives/2021/11/zero\\_knowledge\\_proofs.shtml](https://www.windley.com/archives/2021/11/zero_knowledge_proofs.shtml)
2. [https://openid.net/specs/openid-connect-4-verifiable-presentations-1\\_0-10.html](https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0-10.html)
3. [https://www.windley.com/archives/2020/11/didcomm\\_and\\_the\\_self-sovereign\\_internet.shtml](https://www.windley.com/archives/2020/11/didcomm_and_the_self-sovereign_internet.shtml)
4. <https://www.w3.org/TR/vc-data-model/>
5. <https://www.w3.org/TR/did-core/>
6. <https://medium.com/decentralized-identity/understanding-didcomm-14da547ca36b>
7. <https://identity.foundation/didcomm-messaging/spec>
8. <https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS178x+3T2022/home>
9. <https://www.oose.de/blogpost/oauth-openid-connect-und-jwt-wie-haengt-das-alles-zusammen-teil-1/>
10. <https://www.oose.de/blogpost/oauth-openid-connect-und-jwt-wie-haengt-das-alles-zusammen-teil-2/>
11. <https://openid.net/connect/>