

Self-Sovereign Identity (SSI)

Version 1.01, 30.01.2023

Inhaltsverzeichnis

1. SSI	2
1.1. Austellung und Verifizierung von VC	3
1.2. Teilnehmer	4
2. Aufbau	6
3. DIDComm (Decentralized Identity Communication)	8
3.1. Was ist DIDComm?	8
3.2. DIDComm Ablauf	8
3.3. Wie funktioniert DIDComm?	9
3.3.1. DID Erzeugung	9
3.3.2. Was ist OIDC?	10
3.3.3. Erweiterung des OAuth 2.0 Authorization Request	10
3.3.4. Die Teilnehmer	13
3.3.5. Einfacher Beispiel Ablauf	14
3.3.6. Einordnung von OIDC	14
3.4. OIDC4VC/VP	15
3.5. OIDC4VC	15
3.6. OIDC4VP	15
3.7. Szenarien	16
3.8. Komponenten	17
3.9. Technische Umsetzung	18
3.10. Ergebnis	19
3.11. Beurteilung	33
3.11.1. SSI für Personen	33
3.11.2. SSI für Organisationen	33
3.11.3. SSI für Objekte	34

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
6. Umfang der Ausarbeit

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.

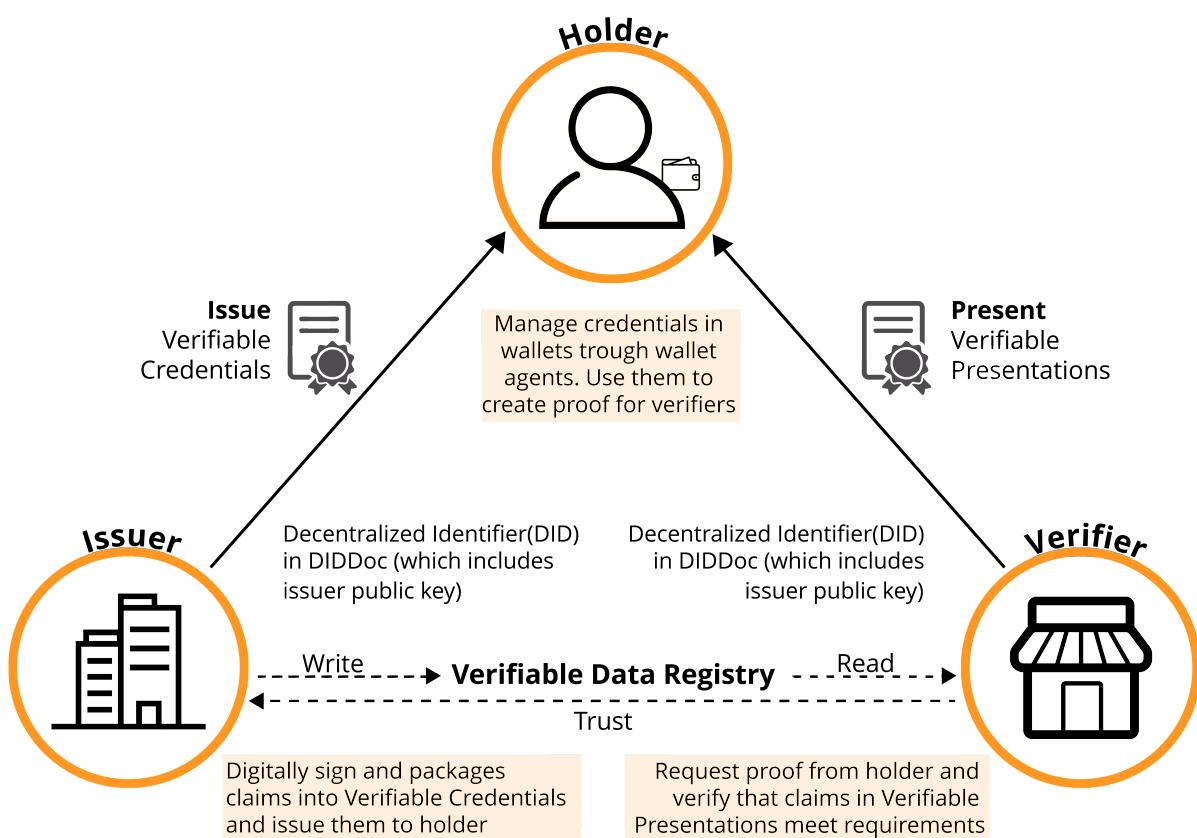
Zuletzt wollen wir durch ein PoC die technische Realisierbarkeit von SSI nachweisen und einige Vorteile und Anwendungsfälle näher erläutern, sowie aber auch einzelne Kritikpunkte aufführen.

Chapter 1. SSI

Self-Sovereign Identity ist ein Konzept, welches den Nutzer in den 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, zu 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.

Um das Konzept hinter Self-Sovereign Identity umsetzen zu können gibt es ein "SSI-Trust Triangle", welches aufzeigt, wie die Echtheit einer Identität bewiesen werden kann.



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 Daten zugänglich macht. Dieser erhält und hält 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.

1.1. Austellung und Verifizierung von VC

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 vertrauenswürdigem 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 durch einen nicht autorisierten Issuer ausgestellt wurden, so kann der Holder nicht authentifiziert werden.

Einführung in VC

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

Mögliche Attribute eines Verifiable Credentials:

- Informationen über die ausstellende Behörde (z.B. eine Stadtverwaltung, eine nationale

Behörde oder eine Zertifizierungsstelle)

- 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 Issuer einfacher die Authentizität und Integrität der Verifiable Credentials zu überprüfen. Zum Beispiel muss ein Arbeitgeber eine vorliegende Bachelorurkunde manuell auf Echtheit prüfen, indem die betreffende Hochschule angefragt wird. Im Anschluss daran muss auf Antwort der Hochschule gewartet werden, was mehrere Tage dauern kann. Durch die Nutzung von Verifiable Credentials wäre es möglich, die Echtheit innerhalb weniger Sekunden festzustellen indem ein Verifiable Credential in Form eines QR-Codes auf dem Handy des Bewerbes eingescannt wird.

1.2. Teilnehmer

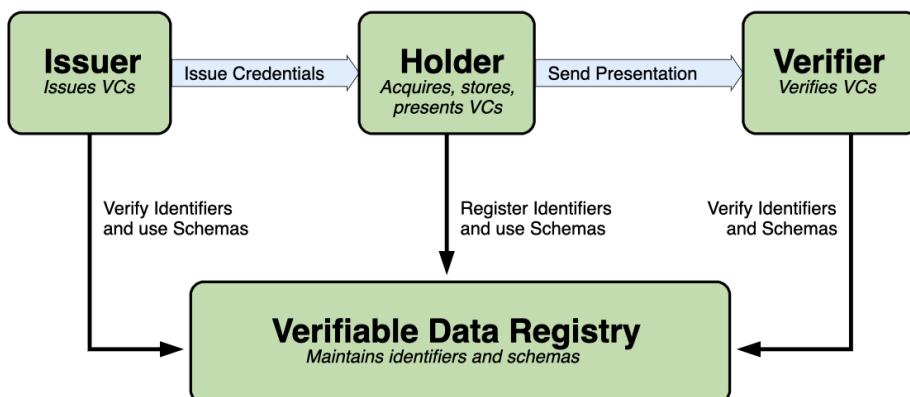


Figure 1 The roles and information flows forming the basis for this specification.

Holder

Eine Entität welche digitale Nachweise (VC) erstellen und verwalten kann. Holder könnten sein: * Personen * Organisationen * Dinge

Issuer

Eine Instanz welche die Fähigkeit besitzt Verifiable Credentials auszustellen. Beispiele dafür wären:
* Universität zum Ausstellen von Abschlussurkunden * Bürgeramt zum Ausstellen eines Personalausweises

Verifier

Die Instanz welche die vorgelegten Credentials der Holder prüft. Dies könnten sein: * Arbeitgeber * Verkäufer * Polizei

Claims

Bei Claims handelt es sich um Behauptungen welche der Holder über sich selbst aufstellt. Diese sollen durch den Issuer in Form von Verifiable Credentials ausgestellt und somit mit der Signatur des Issuers, welche die Echtheit beweist, versehen werden.

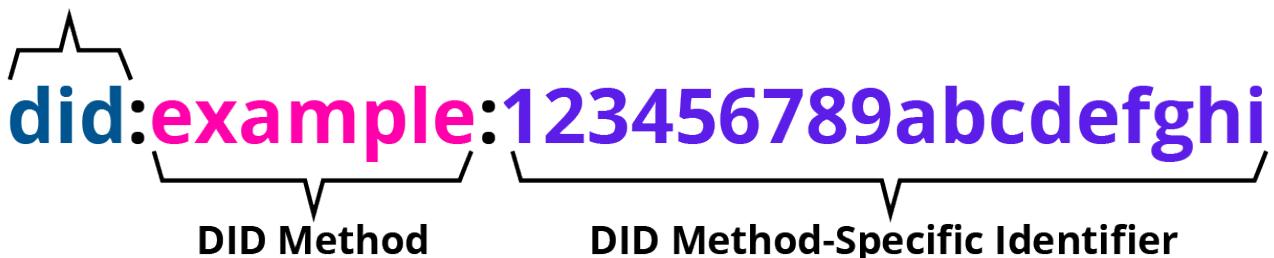
Ein möglicher Lösungsansatz von SSI kann durch Decentralized Identifier (DID) erreicht werden.

Decentralized Identifiers sind eine Art von eindeutigen Identifikatoren, welche es Entitäten ermöglichen, sich selbst zu identifizieren. Dabei fungiert ein DID als Adresse auf ein DID-Dokument in beispielsweise einer Blockchain, welches beschreibt, auf welche Art ein Agent ansprechbar ist.

Eine DID ist eine einfache Textzeichenfolge, die aus drei Bestandteilen besteht:

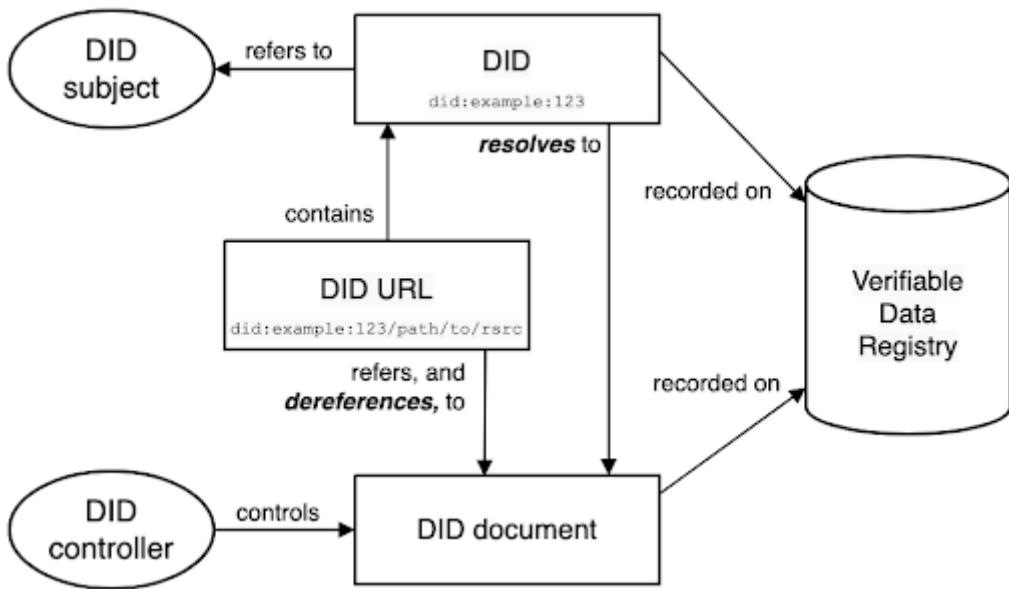
- dem did-URI-Schema-Bezeichner
- dem Bezeichner für die DID-Methode
- dem DID-methodenspezifischen Bezeichner.

Scheme



Das obige DID-Beispiel führt zu einem DID-Dokument. Ein DID-Dokument enthält Informationen wie öffentliche Schlüssel und "Endpoints" zur Kommunikation.

Chapter 2. Aufbau



DID and DID URL

Ein dezentraler Bezeichner (Decentralized Identifier, DID) ist ein URI, die aus drei Teilen besteht, dem Schema did, einem Methodenbezeichner, einem eindeutigen, methodenspezifischen Bezeichner der durch die DID-Methode festgelegt wird. DIDs sind in DID-Dokumente auflösbar. 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-Dokuments oder eine Ressource außerhalb des DID-Dokuments.

`did:example:123456789abcdefghi/path/to/rsrcc`

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 Documentes zu ändern. Ein Controller kann auch das Subject sein.

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 Blockchains.

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 sind der Mechanismus, mit dem eine bestimmte Art von DID und das zugehörige DID-Dokument erstellt, aufgelöst, aktualisiert und deaktiviert werden.

DID Resolver and 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.

Chapter 3. DIDComm (Decentralized Identity Communication)

3.1. Was ist DIDComm?

DIDComm ist ein Kommunikationsprotokoll, 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 als

- sicheres
- offenes
- standardbasiertes

Nachrichtenprotokoll konzipiert, das ein

- zuverlässiges
- vertrauenswürdige

Kommunikationsverbindung zwischen Anwendungen erstellt. DIDComm bietet eine Möglichkeit für Anwendungen und Benutzer, miteinander zu interagieren und Daten, Dokumente und andere Informationen auszutauschen, während sie 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.

3.2. DIDComm Ablauf

[pascal didcomm flow]

1. Verbindungsauftbau: Austausch der DIDs durch eine Out-Of-Band-Nachricht. Diese kann in Form einer Email, SMS oder auch eines QR-Codes gesendet werden.

2. Akzeptanz: Bestätigt der Empfänger die Kommunikation, so wird ein Kommunikationskanal initialisiert.

3. Nachricht verschlüsseln und senden: Eine Partei verschlüsselt ihre Nachricht mit Hilfe des öffentlichen Schlüssels(pulic key) der empfangenden Partei und sendet diese über den vorher aufgebauten Kommunikationskanal.

4. Nachricht empfangen und entschlüsseln: Der Empfänger entschlüsselt die Nachricht nach Empfang mit seinem private Key.

5. Ende des Protokolls: Die beiden Parteien schließen den sicheren Kanal und geben alle verwendeten Ressourcen frei.

3.3. Wie funktioniert DIDComm?

Das DIDComm Protokoll basiert auf einer Reihe von verschlüsselten Nachrichten, die zwischen verschiedenen Parteien ausgetauscht werden, wobei die dezentralen Identifikatoren (DIDs) der Parteien als Adressen für die Nachrichten verwendet werden.

Um zu verstehen, wie das funktioniert, muss man zunächst wissen, was ein dezentraler Identifikator (DID) ist. Ein DID ist ein eindeutiger, Blockchain-basierter Identifikator, der zur Identifizierung von Personen, Organisationen oder anderen Einheiten in einem dezentralen System verwendet wird. Er ähnelt einem herkömmlichen Identifikator, wie z. B. einem Benutzernamen oder einer E-Mail-Adresse, ist jedoch dezentralisiert, d. h. er wird nicht von einer zentralen Behörde kontrolliert. Stattdessen wird die DID vom Eigentümer der Kennung kontrolliert, der die volle Kontrolle über seine eigene Identität und seine Daten hat.

```
did:example:1234567890
```

In diesem Beispiel ist "did" der Bezeichner für eine dezentrale Kennung, "example" ist der Methodenbezeichner (der den Typ der DID angibt) und "1234567890" ist der spezifische DID-Bezeichner. Diese DID wäre für den Besitzer der Kennung eindeutig und kann zur Identifizierung innerhalb eines dezentralen Systems verwendet werden.

Bei DIDComm haben die an einem Kommunikationsaustausch beteiligten Parteien jeweils ihre eigenen DIDs, die als Adressen für die Nachrichten verwendet werden. Um eine Nachricht zu versenden, verschlüsselt der Absender die Nachricht mit seinem privaten Schlüssel. Der Empfänger kann dann die Nachricht mithilfe des öffentlichen Schlüssels des Absenders entschlüsseln. Dadurch wird sichergestellt, dass der Inhalt der Nachricht nur für die vorgesehenen Parteien zugänglich ist und von niemandem sonst gelesen werden kann.

3.3.1. DID Erzeugung

Das DIDComm-Protokoll ist ein sicheres Kommunikationsprotokoll für dezentralisierte Identitätssysteme, das sich nicht direkt mit der Erstellung oder Erzeugung von DIDs befasst. Stattdessen ist das Protokoll so konzipiert, dass es den Austausch von Nachrichten und Daten zwischen verschiedenen Parteien unter Verwendung von DIDs als Adressen für die Nachrichten erleichtert.

Das DIDComm-Protokoll enthält 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 einer DID zu überprüfen.

Wenn beispielsweise eine Partei eine neue DID erstellen möchte, kann sie das DIDComm-Protokoll verwenden, um ein DID-Dokument zu erstellen, das die erforderlichen Informationen wie die DID selbst, die mit der 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 verteilten 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 Veröffentlichung neuer DIDs eingesetzt werden können.

Quellen:

1. https://www.windley.com/archives/2020/11/didcomm_and_the_self-sovereign_internet.shtml
2. <https://medium.com/decentralized-identity/understanding-didcomm-14da547ca36b>
3. <https://identity.foundation/didcomm-messaging/spec>

3.3.2. 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, der/die 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

3.3.3. 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.

Scope

Scope ist ein Konzept in OIDC, das die Zugriffsberechtigungen eines Access Tokens auf bestimmte Ressourcen zb. die eines Nutzers beschreibt. Es ist eine Liste von Claims, die angeben, für welche Bereiche ein 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, das "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 Anwendung alle ihre Nutzerdaten lesen und verarbeiten dürfen und sich dessen nicht bewusst. Dadurch entsteht ein Verlust der Datenkontrolle auf Seiten des Endnutzers.

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/Reyling 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 könnten 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.

ID-Token

Um einen Nutzer zu identifizieren, wird ein ID-Token verwendet. Es handelt sich hierbei um ein signiertes und optional verschlüsseltes Datenelement, das **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, die das Token ausgestellt hat (OpenID Provider und/oder Identity Provider)

"sub" (Subject)

Eindeutige Identifikation des Benutzers, für den das Token ausgestellt wurde (Resource Owner)

"aud" (Audience)

Empfänger des Tokens (Reyling Partie)

"exp" (Expiration Time)

Ablaufdatum des Tokens

"iat" (Issued At)

Zeitpunkt, zu dem das Token ausgestellt wurde

und weitere Informationen über den Benutzer.

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, das **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

3.3.4. Die Teilnehmer

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

- **Identity Provider (IdP):** Ein IdP ist ein Dienst (zb. Google oder Facebook), der 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 Partie (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, die im System ausgeführt wird.

- **OpenID Provider (OP):** Ein OP ist ein Dienst, der 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, die 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.

3.3.5. Einfacher Beispiel Ablauf

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

[Ablauf OIDC] | ./4_Schlussfolgerungen/img/Ablauf_OIDC.png

1. Ein Endbenutzer (Resource Owner) besucht eine Website (Relying Partie), die OpenID Connect unterstützt und klickt zB. 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 Partie mit einem ID-Token und einem Access Token zurück.
5. Die Relying Partie überprüft den ID-Token, um sicherzustellen, dass dieser gültig ist und dass der Benutzer derjenige ist, der er vorgibt zu sein.
6. Die Relying Partie kann nun das Access Token verwenden, um im Namen des Benutzers API-Aufrufe an den OpenID Provider zu tätigen.
7. Die Website kann dadurch dem Nutzer ein personalisiertes Erlebnis auf der Website auf Grundlagen seiner persönlichen Daten bieten.

3.3.6. Einordnung von OIDC

OIDC ist der derzeitige Standard für die Identifizierung, Authentifizierung und Autorisierung im Web 2.0 ist und wird unter anderem am häufigsten eingesetzt. Dabei treten gewisse Herausforderungen und Bedenken auf, die im ersten Abschnitt bereits erläutert wurden.

Ein neuer konzeptioneller Ansatz diese Herausforderungen zu lösen, bietet SSI (Self-Sovereign Identity). Dieses Konzept schiebt den Nutzer in den zentralen Mittelpunkt der Verwaltung seiner eigenen Daten und stellt eine Verschiebung der zentralen digitalen Identität zur dezentralen digitalen Identität dar.

Im nächsten Abschnitt wollen wir das Konzept von SSI genauer erläutern und kurz auf die technischen Grundrahmenbedingungen eingehen. SSI kann durch verschiedenen 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.

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

1. **Architektur:** DIDComm ist ein dezentralisiertes Protokoll für die Kommunikation zwischen

DIDs (Decentralized Identifiers), während OIDC ein zentralisiertes Protokoll ist, das von einem Identity Provider (IdP) bereitgestellt wird.

2. **Kontrolle über Identitätsdaten:** In DIDComm haben Benutzer volle Kontrolle über ihre Identitätsdaten und Credentials, während bei OIDC der Identity Provider die Kontrolle hat.
3. **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.
4. **Verifizierbare Credentials:** DIDComm unterstützt die Verwendung verifizierbarer Credentials und Präsentationen, um Identitäten zu verifizieren, während OIDC dies nicht unterstützt.
5. **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, das einfacher zu integrieren ist, aber 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 Issuer's.

**OpenID4VC als Erweiterung von OIDC erfüllt diesen Ansatz.

3.4. OIDC4VC/VP

Als Erweiterung des OpenID Connect Protokolls gibt es die Möglichkeit der Verifiable Credentials und Verifiable Presentation. Diese wurden basierend auf dem Konzept von SSI umgesetzt.

3.5. OIDC4VC

Das Grundkonzept beschreibt die Bereitstellung der Identität in Form von Verifiable Credentials für den Resource Owner (Nutzer) durch den Identity Provider. Die digitalisierte Identität wird in der Wallet des RO gespeichert und ist für keine Dritten zugänglich. Somit liegt die Datenhoheit beim Nutzer.

3.6. OIDC4VP

Im nächsten Schritt muss der Resource Owner in der Lage sein, sich gegenüber einer Relying Party

auszuweisen. Die Grundlage dafür bilden die Verifiable Credentials, welche durch "Selective-Attribute Disclosure" zu einer Verifiable Presentation zusammengefasst werden. Dabei werden nur die notwendigen, vom Nutzer ausgewählten, Attribute präsentiert. Der Identity Provider rückt bei der Datenübergabe in den Hintergrund. Die Verifiable Presentation beinhaltet, für die spätere Überprüfung der Identität, immer die DID des Identity Providers. Die Relying Party ist somit in der Lage die Herkunft und Echtheit der Verifiable Presentation zu authentifizieren. Möchte der Resource Owner jedoch kein Attribut seiner Identität übermitteln, besteht die Möglichkeit für einen Zero-Knowledge Proof (ZKP). Dieser beschreibt eine kryptografische Methode durch diese ein Resource Owner gegenüber einer Relying Party ein Attribut beweisen kann ohne dieses offen zu legen. Beispielsweise übergibt man bei der Überprüfung der Volljährigkeit an Stelle des Alters nur einen Beweis, mindestens 18 Jahre alt zu sein. Das Ziel dabei ist es, so wenig Informationen über die eigene Identität preiszugeben wie möglich.

sources: https://www.windley.com/archives/2021/11/zero_knowledge_proofs.shtml
https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0-10.html

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 erleichtert, 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. 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, die auf SSI basieren.

Um das Ganze anschaulicher zu Demonstrieren, wollen dazu die [Alice Gets a Mobile Agent!](#) Demo von Hyperledger Aries verwenden. In dieser Demo werden wir 2 Standardszenarien simulieren.

3.7. Szenarien

(1.) Das Ausstellen von Credentials

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

(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, ihr Credentials zuzusenden. Alice kann den Umfang der geforderten Daten vorab einsehen und diese bestätigen. Ein **Selective Attribute Disclosure** wird in dieser Demo nicht gezeigt.

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.

3.8. 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 Agent kommuniziert.

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. Verbindungsauflaufbau zu einem anderen Agent (Hochschule/Arbeitgeber)
3. Die Verwaltung von Verbindungen
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 betrachten werden.

Hochschule

Damit wir unseren Issuer, die Hochschule, simulieren können, müssen wir diese als Agent lokal installieren und hosten. Der Agent verfügt dann 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 2 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 Reyling 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.

⇒ Daraus ergibt sich der folgender technische Aufbau

3.9. 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 ein 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 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, das Teil der Demos sein soll.

--aip 10

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

--revocation

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

--events

gibt an, dass das 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.

3.10. Ergebnis

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
```

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

(2.) Provisionierung eines Agents und einer Wallet

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

```
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 --ain 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.bcovrin.vonx.io/register
Faber      | nym_info: {'did': 'EcagwahExWPRRmJmCeFA7e', 'seed': 'd_00000000000000000000000000000000133130', 'verkey': '8RMCzwEDjRU4BWLz9dM9Gee9z7VGFw3eAKJbpwD6PhJn'}
Faber      | Registered DID: EcagwahExWPRRmJmCeFA7e
Created public DID
```

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

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

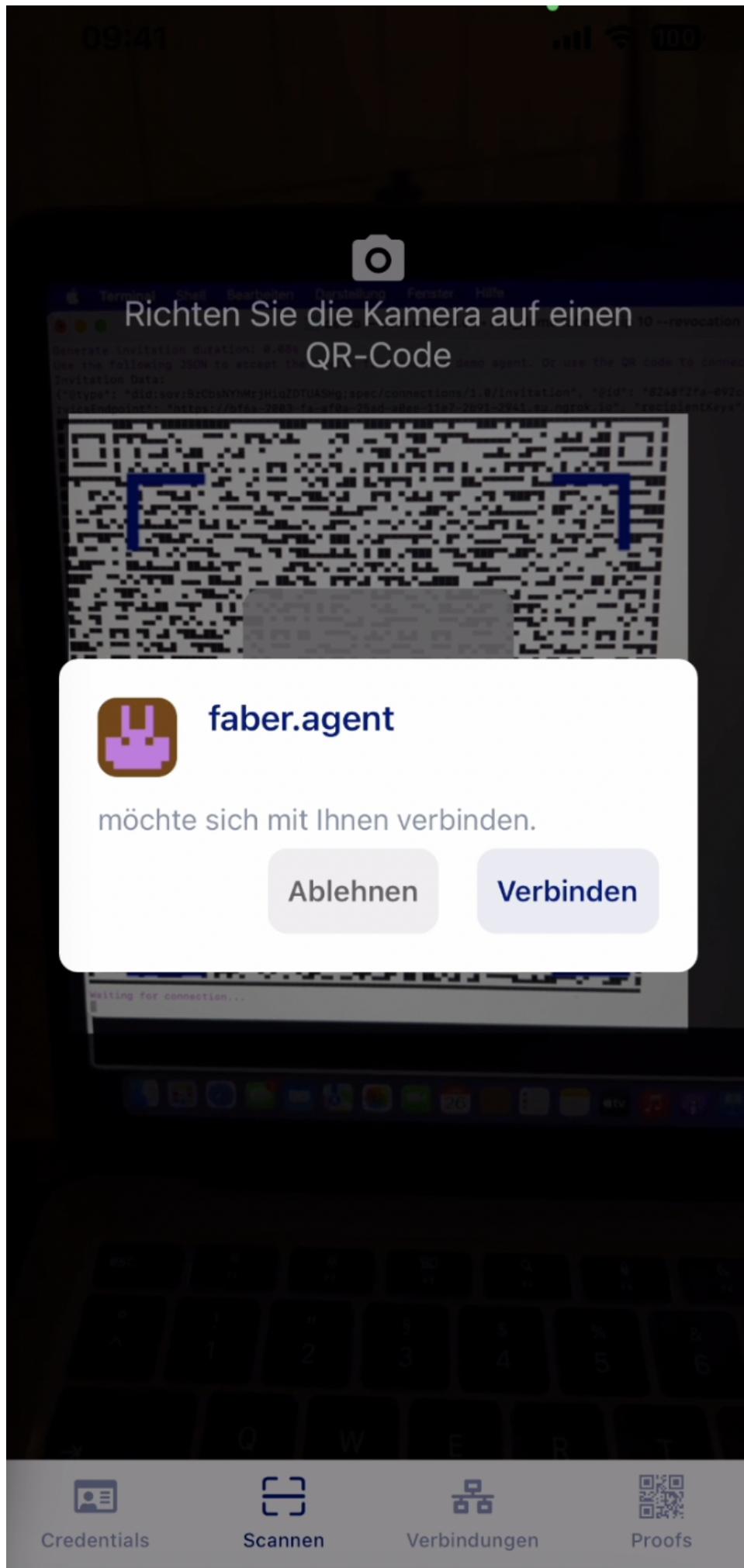
Schema:

```
{  
  "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

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



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

09:41

100%

Verbindungen



faber.agent



faber.agent

26.01.2023

Angeboten

02:11

faber.agent.degree_schema

birthdate_dateint	19990126
date	2018-05-28
degree	Maths
name	Alice Smith
timestamp	1674695507

Löschen

Akzeptieren



Credentials



Scannen



Verbindungen



Proofs

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 Datenhohheit.

(8). Speicherung der Credentials



faber.agent.degree_schema



Info ^

faber.agent.degree_schema

birthdate_dateint	19990126
date	2018-05-28
degree	Maths
name	Alice Smith
timestamp	1674695507
Ausgestellt von	faber.agent
Ausgestellt am	26.01.2023

Credential Definition:

EcagwahExWPRRmJmCeFA7e:3:CL:668113:faber.agent.degree_schema

Schema: EcagwahExWPRRmJmCeFA7e:2:degree schema:38.96.33



Verlauf ^

Credential löschen 


 Credentials


 Scannen


 Verbindungen


 Proofs

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



faber.agent.degree_schema



Info ^

faber.agent.degree_schema

birthdate_dateint	19990126
date	2018-05-28
degree	Maths
name	Alice Smith
timestamp	1674695507
Ausgestellt von	faber.agent
Ausgestellt am	26.01.2023

Credential Definition:

EcagwahExWPRRmJmCeFA7e:3:CL:668113:faber.agent.degree_schema

Schema: EcagwahExWPRRmJmCeFA7e:2:degree schema:38.96.33



Verlauf ^

Credential löschen 


 Credentials


 Scannen


 Verbindungen


 Proofs

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.

3.11. 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 Perspektive.

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

3.11.1. 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.

3.11.2. 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 zb. gegenüber einen Clienten oder als juristischer Vertreter seiner Institution ausweisen.

3.11.3. 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.

...