Hochschule RheinMain
Fachbereich DCSM
Studiengang Master of Science - Informatik

Masterthesis
zur Erlangung des akademischen Grades
Master of Science - M.Sc.1.

Entwicklung eines dezentralen Identitätsmanagementsystems basierend auf Distributed Ledger Technology (DLT)

vorgelegt von **Robert DAVIDOFF**
Matrikelnummer 1108804
Innsbrucker Straße 34
55246 Mainz-Kostheim

am 10.11.2023

Referent: Prof. Dr. Philipp SCHAIBLE
Korreferent: Prof. Dr.Marc-Alexander ZSCHIEGNER

Kompiliert am 03.12.2023 um 17:21:02 - Erstellt mit LaTeX

Formales

Erklärung gem. BBPO 4.4.5 (3)

Ich versichere, dass ich die Master-Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum Unterschrift Studierender

Hiermit erkläre ich mein Einverständnis mit den im Folgenden aufgeführten Verbreitungsformen dieser Master-Arbeit:

Verbreitungsform	**ja**	**nein**
Veröffentlichung des Titels der Arbeit im Internet	X	
Veröffentlichung der Arbeit im Internet	X	

Ort, Datum Unterschrift Studierender

Abstract

Deutsch

Die vorliegende Masterarbeit beschäftigt sich mit dem Konzept der Self-Sovereign-Identity (SSI) und hat das Ziel einen Prototypen für ein Identitätsmanagementsystem zu implementieren, welches auf Distributed-Ledger-Technology (DLT) basiert. Diese Arbeit führt mehrere SSI-Lösung (Sovrin, Dock, ShoCard, Luniverse, PolygonId) auf und vergleicht diese im Anschluss miteinander. Als Ergebnis des Vergleichs tritt PolygonId als beste Option hervor. Demnach wird der Prototyp basierend auf PolygonId implementiert. Die anschließende Analyse ergibt, dass die Anwendung voll funktional ist und Operationen (DID-Erstellung, Claim-Erstellung, Erhalten des Claims als QR-Code, Identitäten lesen und Claims widerrufen) eine Laufzeit von 1 bis 1.5 Sekunden haben. Zudem ergab die anschließende STRIDE-Analyse, dass die Anwendung resistent gegen eine Vielzahl an Angriffskategorien ist.

English

The present master's thesis focuses on the concept of Self-Sovereign Identity (SSI) and aims to implement a prototype for an identity management system based on Distributed Ledger Technology (DLT). This work introduces several SSI solutions (Sovrin, Dock, ShoCard, Luniverse, PolygonId) and subsequently compares them. The comparison results highlight PolygonId as the optimal choice. Accordingly, the prototype is implemented based on PolygonId. The subsequent analysis reveals that the application is fully functional, with operations (DID creation, claim creation, receiving the claim as a QR code, reading identities, and revoking claims) having a runtime of 1 to 1.5 seconds. Additionally, the subsequent STRIDE analysis indicates that the application is resilient against a variety of attack categories.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Hintergrund und Motivation

Das Internet hat sich als disruptive Technologie erwiesen, die die Art und Weise, wie Menschen kommunizieren, Informationen teilen und Geschäfte abwickeln, revolutioniert hat. Im Zeitalter des Internets spielt die Identität im virtuellen Raum eine wichtige Rolle. Normalerweise erfordert die Nutzung eines Online-Dienstes eine einmalige Registrierung und im Anschluss für jede Verwendung eine Anmeldung unter Angabe der zuvor festgelegten Login-Daten. Neben den Login-Daten werden meist auch personenbezogene Daten abgefragt. Wenn ein Nutzer nun X verschiedene Online-Dienste verwendet, so werden X mal identische Daten zur Person gespeichert (Adresse, Vorname, Nachname, Geschlecht, etc.). Dieses Verhalten verursacht die Entstehung von Datensilos, die mit mehreren Problemen einhergehen. Nach [MS15] summieren sich die Kosten für die Identitätsdatenspeicherung in den UK auf knapp 4 Billionen Euro und in den USA hochgerechnet auf 22 Billionen Euro.

Ein weiteres Problem ist die Benutzerfreundlichkeit für den Anwender. Dieser ist gezwungen für jeden Online-Dienst sichere Login-Daten zu selektieren. Sind diese immer identisch, so stellt dies ein Sicherheitsrisiko dar, denn wenn einmalig ein Password kompromittiert ist, sind alle anderen Dienste in Gefahr. Besser wäre demnach für jeden Online-Dienst unterschiedliche Login-Daten zu verwenden, was jedoch das Merken schwer macht.

Darüber hinaus stellt die Identitätsproblematik im Internet einen potenziellen Angriffsvektor dar. Cyberkriminelle können Schwachstellen in den Authentifizierungssystemen ausnutzen, um unbefugten Zugriff auf Konten zu erlangen oder Identitätsdiebstahl zu begehen. Dies birgt Risiken für die Privatsphäre und Sicherheit der Nutzer. In den USA werden 25 Personen pro Minute Opfer von Identitätsdiebstahl, wobei sich die durchschnittlichen Kosten für einen Online-Händler pro gestohlenem Datensatz personenbezogener/sensibler Daten auf 164 USD belaufen (2023) [Id6c]. Drei Jahre zuvor lag dieser Wert noch bei 146 USD.

Statistiken [Sta12] zeigen, dass 82% der Unternehmen unter gefälschten Nutzerkonten leiden. Diese Fake-User verursachen nicht nur finanzielle Schäden, sondern können auch den Ruf eines Unternehmens schädigen. Darüber hinaus werden etwa 18% der Einkaufswagen aufgrund von Problemen mit den Anmeldedaten aufgegeben. Dies führt zu Umsatzeinbußen für Unternehmen und frustriert potentielle Kunden.

Ein weiteres Problem ist, dass ein Nutzer im Status Quo keine Macht über seine Daten besitzt. Er ist stets davon abhängig dem Anbieter zu vertrauen, dass die Daten bei Anfrage gelöscht werden, sicher gespeichert sind, nicht ungefragt weitergegeben werden, etc. Ebenso ist keinerlei Transparenz darüber gegeben, wofür die Daten im Detail verwendet oder wofür sie gebraucht werden. Alles in einem besteht keine Autonomie für den Nutzer über die Daten, die er bei einem Online-Service angeben muss.

Angesichts dieser Herausforderungen ist die Notwendigkeit einer verbesserten Identitätsverwaltung im Internet offensichtlich. Es werden Lösungen erforscht, die auf dezentralen Identitätsplattformen und Blockchain-Technologie basieren. Solche Ansätze könnten dazu beitragen, die Sicherheit, Privatsphäre und Benutzerfreundlichkeit im Internet zu verbessern, indem sie eine effizientere und sicherere Möglichkeit bieten, Identitäten zu verwalten und zu überprüfen.

1.2 Zielsetzung der Arbeit

Als Ziel soll ein Konzept erarbeitet werden, dass dem Nutzer erlaubt Herrscher seiner Daten zu sein. Er soll eigenständig in der Lage seine Informationen hinzuzufügen, zu teilen oder Anfragen zu beantworten. Als technologische Grundlage soll DLT (Distributed Ledger Technology) verwendet werden. Es soll ein Prototyp entwickelt werden, der obere Logik implementiert. Es sollen Funktionen zur Verfügung stehen zum:

- Identitäten erstellen
- Dokumente erstellen
- Dokumente überprüfen
- Dokumente widerrufen

Ein möglicher Anwendungsfall wäre, das ein Nutzer ein Dokument von einer Behörde ausgestellt bekommt und dieses dadurch besitzt. Das Wichtige hierbei ist, dass nur der Nutzer Zugriff zum Dokument und den darin enthaltenen Informationen hat. Zudem muss die Möglichkeit bestehen, dass eine andere Instanz die Korrektheit oder Attribute des Dokumentes überprüfen kann.

1.3 Forschungsfragen

Folgende Forschungsfragen werden in dieser Arbeit beantwortet:

1. Wie erfolgt die Speicherung verschiedener Typen an Daten innerhalb eines Identitätsmanagementsystems?
   (a) Wie kann die technische Machbarkeit erreicht werden, Daten sowohl öffentlich zugänglich als auch privat zu speichern?
   (b) Welche kryptografischen Technologien, wie zum Beispiel Hashing oder Verschlüsselung, sind optimal für die sichere Speicherung von Identitätsdaten geeignet?
   (c) Welche Sicherheitsmaßnahmen sind im Falle einer Datenkompromittierung zu ergreifen, und welche Wiederherstellungsoptionen sind verfügbar?
   (d) Wie lässt sich die sichere und effektive Ungültigmachung von Informationen (Revokation) gewährleisten?
2. Welchen Nutzer oder Service-Mehrwert generiert ein dezentrales Identitätsmanagementsystem basierend auf DLT?
   (a) Welche konkreten Vorteile ergeben sich für Nutzer und Online-Dienste durch die Implementierung des vorgeschlagenen Identitätsmanagementsystems?
   (b) Inwiefern trägt das System zur Lösung der Problematik von Fake-Usern bei?
3. Wie erfolgt die Identitätszuordnung innerhalb des Identitätsmanagementsystems?
   (a) Wie kann eine verlässliche Zuordnung von Identitäten zu Personen auf hohem Sicherheitsniveau erreicht werden?
   (b) Welche Ansätze können entwickelt werden, um das Problem zu lösen, dass Nutzer möglicherweise verschiedene Identitäten auf verschiedenen Plattformen verwenden möchten?
4. Welche technologischen Spezifika sind im Bezug auf die Blockchain als DLT im Identitätsmanagementsystem zu entscheiden?
   (a) Welcher Konsensus-Algorithmus ist am besten geeignet, um die Anforderungen des entwickelten Identitätsmanagementsystems zu erfüllen?
   (b) Wie können private Schlüssel sicher gespeichert werden, um unbefugten Zugriff zu verhindern?
   (c) Inwiefern können Identitätsdokumente erfolgreich als NFTs gespeichert werden, und welche Implikationen ergeben sich daraus?
   (d) Welche Rolle spielen Zero-Knowledge-Proofs bei der Entwicklung eines sicheren Identitätsmanagementsystems?

1.4 Aufbau der Arbeit

Zunächst werden die Grundlagen dieser Arbeit gesetzt, indem auf die Historie von Identitätsmanagementsystemen eingegangen wird. Dabei werden der zentrale, föderierte, Nutzer-zentrierte und selbstbestimmte Ansatz erläutert. Daraufhin werden die Grundlagen von Distributed Ledger Technology erläutert und deren Bedeutung für Identitätsmanagementsysteme angeführt. Im Anschluss werden die Anforderungen formuliert - die in funktionale/nicht-funktionale und technische Anforderungen gegliedert sind - und existierende Lösungen zunächst vorgestellt und im nächsten Schritt verglichen. Im Fokus stehen hierbei Luniverse, Dock, Sovrin, ShoCard und PolygonId. Nachdem erklärt wurde, warum Polygon die passende Plattform ist, wird ein System-Design vorgestellt, welches im nächsten Schritt implementiert wird. Hierbei wird zunächst das Grobe Design skizziert und im Anschluss jede Komponente einzeln illustriert. Im vorletzten Schritt werden Metriken festgelegt (Laufzeit,Sicherheit) und evaluiert. Zum Abschluss wird ein Fazit verfasst, es wird überprüft, ob alle Anforderungen erfüllt wurden und eine Diskussion findet statt. Auch werden zukünftige Forschungsrichtungen aufgeführt und der Beitrag zur Forschung illustriert.

Kapitel 2

Grundlagen

2.1 Historie von Identitätsmanagementsystemen und deren Status Quo

Die Historie von Identitätsmanagementsystemen ist geprägt von verschiedenen Ansätzen, darunter die zentralisierte Identität (centralized Identity), die föderierte Identität (federated Identity), die nutzerzentrierte Identität (user-centric Identity) und die selbstbestimmte Identität (self-sovereign Identity). Die angegeben Identitätssysteme schließen sich nicht gegenseitig aus und vor allem die dezentrale Identität wird in modernen dezentralen Identitätsmanagementsystemen in Kombination mit seinen Vorgängern implementiert.

Zentralisierte Identitätssysteme waren lange Zeit vorherrschend, bei denen Identitätsinformationen in zentralen Datenbanken gespeichert wurden. Organisationen und Behörden kontrollierten den Zugriff auf diese Daten und verwalteten die Identitäten der Benutzer. Dabei authentifiziert sich ein Nutzer mit einer Nutzeridentifikation und einem Password. Dieser Ansatz führte jedoch zu Fragmentierung, Ineffizienz und möglichen Sicherheitsrisiken, wenn Nutzer nicht unterschiedliche Login-Daten für jeden Online-Dienst verwenden.

Mit der Einführung der föderierten Identitätssysteme wurde versucht, diese Probleme zu lösen. Hierbei können Benutzer über einen Identitätsanbieter, wie beispielsweise ein soziales Netzwerk oder ein Unternehmenskonto, auf verschiedene Dienste zugreifen. Der Identitätsanbieter fungiert als Vermittler und ermöglicht den nahtlosen Zugriff, ohne dass Benutzer separate Anmeldeinformationen für jeden Dienst bereitstellen müssen. Das Konzept hinter der föderierten Identität lautet *Single-Sign-On (SSO)*. Dabei gibt der Nutzer pro Sitzung seine Login-Daten einem *Identitätsanbieter* (Google, Facebook, etc), welcher im Gegenzug ein signiertes Token ausstellt, welches für kommende Logins verwendet wird. Die dabei verwendeten Technologien sind beispielsweise *SAML* [Loc+] oder *OpenID Connect* [Id1e].

Die nutzerzentrierte Identität rückt den Benutzer in den Mittelpunkt des Identitätsmanagements. Bei diesem Ansatz behalten Benutzer die Kontrolle über ihre Identitätsdaten und können sie in einer sicheren Umgebung speichern. Sie können ihre Daten selektiv freigeben und verwalten, was zu mehr Privatsphäre und Kontrolle führt. Eine Implementierung hierfür ist BrowserID[Id1c]. Durchgesetzt hat sich diese Technologie jedoch nicht, da es an Akzeptanz und Integration durch Webseiten mangelte.

Die selbstbestimmte Identität oder Self-Sovereign Identity (SSI) stellt den neuesten Ansatz dar. Bei SSI behalten Benutzer die vollständige Kontrolle über ihre Identitätsdaten, indem sie kryptografische Schlüssel verwenden. Die Identitätsdaten werden dezentralisiert und auf der Blockchain oder anderen verteilten Systemen gespeichert. Benutzer können selektiv Informationen freigeben und verifizieren, wodurch ihre Privatsphäre und Sicherheit gestärkt werden.

2.2 Self-Sovereign-Identity

2.2.1 Das Konzept hinter SSI

Das Konzept der Self-Sovereign Identity [TR17] basiert auf den folgenden Prin

zipien:

1. Benutzerkontrolle: Der Benutzer hat die ultimative Kontrolle über seine Identität und die damit verbundenen Daten. Der Benutzer kann bestimmen, welche Informationen er teilen möchte, mit wem und zu welchen Bedingungen.

2. Dezentralisierung: Die Identitätsdaten sind nicht an eine zentrale Institution oder Datenbank gebunden. Stattdessen werden sie dezentral auf verschiedenen Plattformen, Geräten oder Blockchains gespeichert. Der Benutzer hat die Möglichkeit, seine Identitätsdaten an einem sicheren Ort seiner Wahl zu speichern.

3. Interoperabilität: SSI strebt nach Interoperabilität zwischen verschiedenen Identitätsplattformen und -systemen. Das bedeutet, dass Identitätsdaten zwischen verschiedenen Diensten und Organisationen ausgetauscht und verifiziert werden können, ohne dass eine zentrale Instanz benötigt wird.

4. Vertrauensmodelle: SSI nutzt kryptografische Technologien, wie digitale Signaturen und Blockchains, um die Integrität und Vertrauenswürdigkeit von Identitätsdaten zu gewährleisten. Es ermöglicht auch das Prinzip der Verifizierung von Informationen, bei dem die Authentizität bestimmter Daten von anderen Parteien bestätigt werden kann.

5. Datenschutz und Privatsphäre: SSI legt großen Wert auf Datenschutz und Privatsphäre. Der Benutzer hat die Kontrolle darüber, welche Informationen freigegeben werden und welche nicht. Es ermöglicht auch selektive Offenlegung, bei der

nur die notwendigen Informationen für einen bestimmten Zweck oder Kontext offengelegt werden.

Das Ziel von Self-Sovereign Identity ist es, die Verwaltung von Identitätsdaten für Benutzer transparenter, sicherer und benutzerzentrierter zu gestalten. Es bietet die Möglichkeit, Identitätsinformationen nahtlos zwischen verschiedenen Diensten und Organisationen zu nutzen, während die Kontrolle über die eigenen Daten in den Händen des Benutzers bleibt.

2.2.2 Identität

Im Kontext der Self-Sovereign Identity (SSI) gibt es verschiedene Konzepte, die verschiedene Aspekte der Identität und Kontrolle berücksichtigen. Zwei solcher Konzepte sind die *Weak/Nym Identity* und die *Partial/Strong Identity*.

Die Weak/Nym Identity bezieht sich auf eine Identität, die nur begrenzte Informationen über den Benutzer enthält. Bei dieser Identität wird bewusst darauf verzichtet, persönliche Informationen oder Details preiszugeben, die zur Identifizierung des Benutzers verwendet werden könnten. Stattdessen wird ein Pseudonym oder ein Alias verwendet, um die Privatsphäre des Benutzers zu schützen. Die Weak/Nym Identity ermöglicht es dem Benutzer, Transaktionen durchzuführen und Dienste zu nutzen, ohne seine wahre Identität preiszugeben.

Im Gegensatz dazu bezieht sich die Partial/Strong Identity auf eine Identität, die umfassendere Informationen über den Benutzer enthält. Diese Identität kann persönliche Daten wie Name, Adresse, Geburtsdatum und andere relevante Informationen enthalten. Die Partial/Strong Identity ermöglicht eine genauere Identifizierung und Authentifizierung des Benutzers, was in einigen Situationen erforderlich sein kann, beispielsweise bei behördlichen Anforderungen oder bei Zugang zu sensiblen Diensten. Die Partial/Strong Identity erfordert eine sorgfältige Verwaltung der Identitätsdaten, um sicherzustellen, dass sie sicher und geschützt bleiben.

Beide Identitätskonzepte haben ihre eigenen Vor- und Nachteile in Bezug auf Datenschutz, Sicherheit und Benutzerkontrolle. Die Entscheidung für eine bestimmte Identität hängt von den individuellen Anforderungen, dem Kontext und den Präferenzen des Benutzers ab. SSI strebt jedoch nach Flexibilität und Wahlfreiheit für Benutzer, um die Identität zu wählen, die ihren Bedürfnissen am besten entspricht und gleichzeitig die Sicherheit und den Schutz ihrer Daten gewährleistet.

In den folgenden Kapiteln wird in der Konzeption und Implementierung ein Minimum an Daten preisgegeben, also eine schwache Identität. Jedoch werden auch Anwendungsfälle berücksichtigt, wo Informationen zur Identität benötigt werden.

2.2.3 Technische Grundlagen

Um das Konzept der Self-Sovereign Identity (SSI) aus technologischer Sicht zu verstehen und anzuwenden, sind folgende Kernkonzepte erforderlich [Id1d] [Ish20]:

1. Trust-Registries: Diese dienen als gemeinsame und vertrauenswürdige Aufzeichnung bestimmter Informationen. Mit anderen Worten fungieren sie als 'Vertrauensebene' und 'einzige Quelle der Wahrheit'. Eine mögliche Realisierung einer Trust-Registry ist ein dezentraler Speicher (Distributed Ledger), der alle Aktivitäten in Transaktionen speichert.

2. Kryptografische Schlüssel: Diese übertragen die Kontrolle über digitale Identitäten und ermöglichen grundlegende Funktionen wie Verschlüsselung und Authentifizierung. Hierbei handelt es sich um klassische private/öffentliche Schlüsselpaare, die im Falle von der Bitcoin-Blockchain 48 Byte / 256 Bit lang sind [Id1b] und dem Verschlüsseln/Entschlüsseln/Signieren von Daten dienen.

3. Dezentrale Identifikatoren (DIDs): DIDs sind globale und einzigartige Identifikatoren, die keine zentralen Register zur Speicherung benötigen. Sie unterscheiden sich zu UUIDs in dem Sinne, dass DIDs auf sog. DID-Documents zurückzuführen sind und mit kryptographischen Mechanismen Eigentumsverhältnisse zeigen. DID's sind aufgebaut wie folgt:
   "did:"⟨Methodenname⟩":"⟨methodenspefizische-ID⟩
   Beispiel:"did:btcr:abcd-1234-wxyz:789"
   Dabei zeigt ein DID immer auf ein DID-Dokument, welches im JSON-Format Metadaten wie öffentliche Schlüssel oder Authentifizierungsmethoden beinhaltet.

   Ein DID-Document sieht wie folgt aus:

```
{
  "id": "did:ion:EiClkZMDxPKqC9c-umQfTkR8vvZ9JPhl_xLDI9Nfk38w5w",
  "@context": [
  "https://www.w3.org/ns/did/v1",
  {
    "@base": "did:ion:EiClkZMDxPKqC9c-umQfTkR8vvZ9JPhl_xLDI9Nfk38w
        5w"
  }
  ],
  "service": [
  {
    "id": "#linkedin",
    "type": "linkedin",
    "serviceEndpoint": "linkedin.com/in/henry-tsai-6b884014"
  },
  {
    "id": "#github",
    "type": "github",
    "serviceEndpoint": "github.com/thehenrytsai"
  }
  ],
  "verificationMethod": [
  {
    "id": "#someKeyId",
```

```
24      "controller": "did:ion:EiClkZMDxPKqC9c-umQfTkR8vvZ9JPhl_xLDI9
            Nfk38w5w",
25      "type": "EcdsaSecp256k1VerificationKey2019",
26      "publicKeyJwk": {
27        "kty": "EC",
28        "crv": "secp256k1",
29        "x": "WfY7Px6AgH6x-_dgAoRbg8weYRJA36ON-gQiFnETrqw",
30        "y": "IzFx3BUGztK0cyDStiunXbrZYYTtKbOUzx16SUK0sAY"
31      }
32    }
33    ],
34    "authentication": [
35    "#someKeyId"
36    ]
37  }
```

Es ist zu erkennen, dass dieses DID-Document festlegt für welche Services dieses Dokument die Authentifikation definiert (in diesem Falle LinkedIn und Github). Unter 'verificationMethod' wird der Typ 'EcdsaSecp256k1VerificationKey2019' angegeben, was einer Public-Key-Authentifikation entspricht, welche Elliptic-Curve-Kryptografie verwendet.

4. Verifizierbare Nachweise (VCs): VCs sind digitale Identitätsdokumente, die von jedem auf ihre Gültigkeit, Integrität, Authentizität und Herkunft hin überprüft werden können. Sie beinhalten sog. *Claims*, also Informationen/Behauptungen über die Entität (beispielsweise den Namen, Geburtsdatum, etc.). Wichtig ist, dass VCs aus Datenschutz- und Compliance-Gründen niemals unverschlüsselt auf einer Blockchain gespeichert werden. Ein VC kann wie folgt aussehen:

```
1  {
2    "@context": [],
3    "id": "e9ea3429-b32f-44ad-b481-b9929370bb90",
4    "type": [ "VerifiableCredential", "ExampleCredential" ],
5    "issuer": { "id": "did:btcr:2d28bb79-87a9-4224-8c63-d28b29716b67
         " },
6    "issuanceDate": "2022-01-01T00:00:00Z",
7    "credentialSubject": {
8      "id": "did:example:7564cb9c-165c-4857-a887-bfc2460af867",
9      "birth_date": "1970-01-01"
10   },
11   "expirationDate": "2023-01-01T00:00:00Z",
12   "proof": {<SignatureOfIssuer>}
13 }
```

Es ist zu erkennen, dass in dem VC unter anderem Claims ('credentialSubject' genannt) enthalten sind (in diesem Falle das Geburtsdatum), der Issuer des VC, ein Auslaufdatum und ein 'Proof', also eine digitale Signatur des Issuer's, um die Integrität des VC's zu überprüfen.

5. Wallets: Wallets speichern unsere Schlüssel und VCs und ermöglichen die Verwaltung und Nutzung unserer digitalen Identitäten und Daten über benutzerfreundliche Anwendungen.

Diese Kernkonzepte bilden die Grundlage der SSI-Technologie. Sie ermöglichen es Ein-

zelpersonen, die Kontrolle über ihre digitalen Identitäten zu haben, verifizierbare Nachweise sicher zu teilen und vertrauenswürdige Interaktionen mit anderen durchzuführen.

Das grobe Zusammenspiel der Komponenten sieht dabei wie folgt aus:

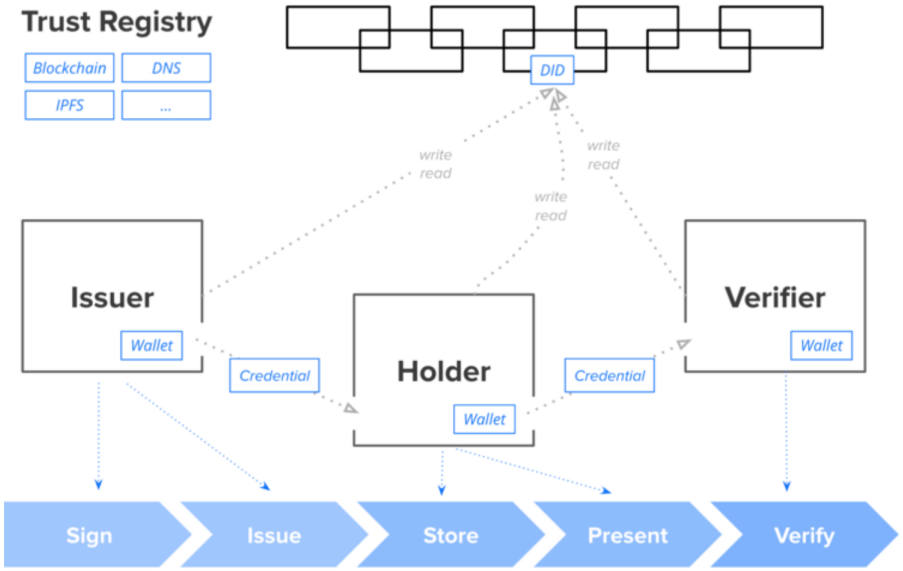

Abbildung 2.1: Zusammenspiel von Issuer, Holder und Verifier [Id6a]

Es ist zu erkennen, dass sowohl Issuer, Holder und Verifier Eigentümer von einem Wallet sind. Der Issuer (beispielsweise eine Bank) stellt VC's aus, die der Holder (Beispielsweise eine Privatperson) in seinem Wallet speichert. Muss sich dieser nun ausweisen, so reicht er dem Verifier (Beispielsweise der Arbeitgeber) eine VC-Representation ein. Diese VC-Representation (oder auch Verifiable Presentation *VP* genannt) beinhaltet in der Regel ein VC, kann jedoch in komplexeren Szenarien mehrere VC's enthalten. Zudem sind alle Akteure (insbesondere der Verifier) in der Lage die Authentizität des VP zu überprüfen.

Issuer, Holder und Verifier können jeweils alle drei Rollen annehmen, besitzen eine DID und sind DID-Subjects. Folgende Zusammenhänge existieren zwischen dem DID-Subject, DID, DID-URL, DID-Controller, DID-Document und der Verifiable Data Registry:

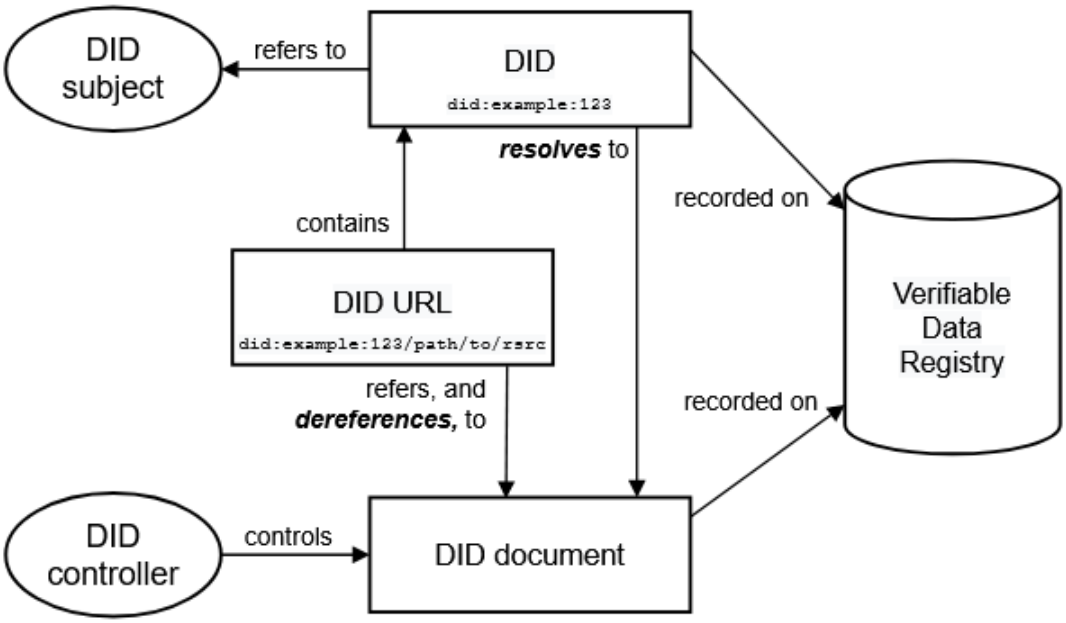

Abbildung 2.2: DID-Objekte und deren Zusammenhänge [Id1a]

Die DID-URL beinhaltet die DID und erweitert die Syntax um die URI-Komponenten wie Pfade oder Anfrage-Parameter. Sowohl DID's als auch DID-Documents werden auf einem Verifiable Data Registry persistiert. Diese werden beispielsweise als Datenbanken oder dezentrale Dateisysteme realisiert. In diesem Kontext sind jedoch distributed Ledger die Speicher der Wahl. Das DID-Dokument wird durch die DID-URL (de-)referenziert und durch die DID auf sich verwiesen. Der DID-Controller ist die Entität, die das DID-Document modifizieren kann. In der Regel ist diese Entität das DID-Subject der DID, es kann jedoch auch ein oder mehrere andere DID-Subjekte sein.

2.3 Politik, Recht und Ethik in Bezug auf SSI

2.3.1 Politik

Aus politischer Sicht spielt SSI eine wichtige Rolle, da das Potential besteht das Monopol des Staates in Bezug auf die Ausstellung, Aufrechterhaltung und Entzug von Ausweisdokumenten zu verlieren. Durch die Einführung von digitalen Entitäten - mit denen man sich auch in der analogen Welt ausweisen könnte - können staatliche Institutionen nicht mehr uneingeschränkt über den genannten Prozess verfügen. Betroffen sein können unter anderem die Digitalisierung der Grenzkontrollen oder das Migrationsmanagement, indem Pässe, Visa und ähnliche Dokumente als VC zur Verfügung gestellt werden.
Eine weitere Auswirkung ist, dass durch SSI das aktuelle Monopol im Identitätsmanagement von Unternehmen wie Meta (Facebook) und Google durch föderierte Iden-

titätsmanagementsysteme abgelöst wird. Dadurch haben letztere Unternehmen nicht mehr die Kontrolle über die Daten ihrer Nutzer, was auch monetäre Auswirkungen hat. Die zuvor für Marketingzwecke verwendeten Daten stehen demnach nicht mehr zu Verfügung für personalisierte Werbung und Ähnlichem.
Insgesamt hat SSI das Potenzial, das bestehende Identitätsmanagement-System zu revolutionieren und die Kontrolle über digitale Identitäten in die Hände der Nutzer zu legen. Dies wirkt sich auf verschiedene Bereiche aus, darunter die Politik, das Monopol des Staates, die Digitalisierung von Grenzkontrollen und die Monetarisierung von Daten.

2.3.2 Recht

Die Einführung von SSI wirft auch rechtliche Fragen und Aspekte auf. Eine zentrale Frage betrifft die rechtliche Anerkennung von digitalen Identitäten und die damit verbundenen Rechte und Pflichten. Da SSI die traditionelle Vorstellung von staatlich ausgestellten Ausweisdokumenten und Identitätsnachweisen herausfordert, müssen rechtliche Rahmenbedingungen geschaffen werden, um die Verwendung und den Schutz digitaler Identitäten zu regeln. Dies könnte die Festlegung von Standards für digitale Identitäten, Datenschutzbestimmungen, Haftungsfragen und den Zugriff auf und die Verwaltung von persönlichen Daten umfassen. Zudem muss auch geklärt werden, wie SSI in bestehende Rechtssysteme und -strukturen integriert werden kann, beispielsweise in Bezug auf Verträge, Gerichtsverfahren oder behördliche Angelegenheiten. Die rechtlichen Aspekte von SSI sind daher von großer Bedeutung, um die rechtliche Sicherheit, den Schutz der Privatsphäre und die Gewährleistung der Rechte und Pflichten aller beteiligten Parteien zu gewährleisten.
Weitere zu beachtende Aspekte sind die juristische Verantwortlichkeit oder Datenschutz. Vor allem Ersteres spielt eine Rolle, wenn es sich um die Verwendung von Identitätsinformationen, Identitätsdiebstahl oder Fälschungsversuchen handelt. Zweiteres ist relevant in Bezug auf die rechtlichen Anforderungen die mit Datenverwaltung einhergehen.

2.3.3 Ethik

Auch auf ethnischer Sicht zeigt sich die Relevanz von SSI. Gerade das Konzept der Dezentralisierung und die Tatsache, dass keine Instanz mehr Macht über die Identitäten hat als andere wirft ethnische Fragestellungen auf. In [Ish20] beschreibt Ishmaev das Paradoxon, dass einerseits SSI zuletzt genannte Eigenschaften fordert, jedoch andererseits verschiedene Levels an 'Vertrauen' an Entitäten in der analogen Welt existieren. So sind beispielsweise Dokumente, die von einer staatlichen Autorität zugewiesen werden bedeutender als das Sportabzeichen für Grundschüler. Dieser Zustand wird in dem SSI-Konzept jedoch nicht berücksichtigt und zeigt den Kompromiss, den SSI-

Identitätsmanagementsysteme implementieren müssen.
Ein weiterer problematischer Aspekt, ist die von Ishmaev genannte Tatsache, dass die Macht über die Freigabe der Daten zwar bei dem Nutzer liegt, die Macht über die Nutzung eines Dienstes liegt jedoch weiterhin bei dem Anbieter. So kann ein Großkonzern für die Nutzung eines Dienstes eine unmoralische Menge an Informationen fordern. Dadurch wird klar, dass weiterhin eine problematische Machtrelation existiert.

Ein weiterer Punkt ist die Diskrepanz zwischen SSI im sozialen und im technischen Kontext. SSI-Systeme unterscheiden nicht zwischen den handelnden Entitäten, ob es sich um Privatpersonen, Institutionen oder - im Rahmen von Internet-of-Things - Hardware handelt. Gerade dadurch weißt die Interpretation von 'Vertrauen' in sozialen oder cybersecurity Bereich Unterschiede auf. Es gibt eine Vielzahl an Definitionen für "Vertrauen", wobei diese meist in Abhängigkeit zu dem Kontext stehen, in dem sie verwendet werden. Eine allgemeine Definition wurde von McKnight und Chervany (1996) festgelegt: Vertrauen ist der Grad zu welchem eine Partei einwilligt abhängig zu etwas oder jemanden in einer Situation zu sein mit einem Gefühl von Sicherheit. Hierbei werden explizit und implizit folgende drei Bestandteile von Vertrauen dargestellt [Jø+05]:

- Abhängigkeiten zwischen Parteien
- Zuverlässigkeit einer Partei
- Risiko, dass eine Partei nicht wie erwartet agiert

Alle diese drei Charakteristika unterscheiden sich, je nachdem ob es sich um IoT-Geräte, Software oder Menschen handelt.

2.4 Merkmale und Vorteile von DLT

Distributed Ledger ist - wie der Titel bereits beschreibt - eine , für diese Arbeit, bedeutende Technologie. Dabei sind folgende Merkmale und Vorteile der DTL relevant [KAN+00]:

- DLT ermöglicht das Betreiben einer hochverfügbaren Datenbank (eines 'Ledgers'), da nicht eine zentrale Instanz für die Verfügbarkeit verantwortlich ist, sondern die Gesamtheit der Knoten in dem Netzwerk.
- Ebenso ermöglicht die Dezentralität des DLT eine verteilte Speicherung und Verarbeitung.
- Manipulationsresistenz wird durch kryptographische Verfahren innerhalb der Blockchain sichergestellt. Im Fall der Blockchain sind diese in der Regel asymmetrische

Verfahren, was bedeutet, dass private/öffentliche Schlüssel zum Ent- oder Verschlüsseln der Daten verwendet werden, wobei 'Integer Factorization', "Discrete Logarithm' oder 'Elliptic Curves' verwendet werden können [Bas17]

- Zensurresistenz kann gewährleistet, indem beispielsweise alle Knoten die gleichen Berechtigungen haben und somit keine machthabende Instanz existiert. Alle Teilnehmer im Netzwerk werden als Knoten (Nodes) bezeichnet und besitzen jeweils eine lokale Kopie des Ledgers. Änderungen werden nun auf der Kopie ausgeführt und im Anschluss in dem Netzwerk synchronisiert. Das Netzwerk gilt als 'untrustworthy' (nicht vertrauenswürdig), wenn willkürliche einzelne Knoten sog. 'Byzantine-Failures' [LSP82] [Sun19] erzeugen können. Dies bedeutet, dass versucht wird beliebig falsches Verhalten im System zu erzeugen (unauthentische Daten, Zusammensturz des Systems, etc). Der Resistenzgrad des Netzwerks gegenüber diesen Angriffen wird als 'Byzantine-Toleranz' bezeichnet und wird in der Regel durch Abstimmungen im Netzwerk (Beispielsweise Konsensus-Algorithmen) verhindert. Beispiele im Blockchain-Kontext sind 'Proof-of-Work' oder "Proof-of-Stake' [Min+17] angewandte Algorithmen.
- Möglichkeit zur 'Demokratisierung' von Daten: Durch DLT kann ermöglicht werden, dass Individuen und/oder Organisationen kooperativ Kontrolle über Daten ausüben

2.5 Anwendung von DLT im Bereich der digitalen Identität

Die oben genannten Eigenschaften sind für das Betreiben eines Identitätsmanagementsystems optimal, da diese hochverfügbar sein sollten, mit möglichst kurzen oder nicht existierenden Downtimes. Auch ist eine verteilte Speicherung und Verarbeitung eine effiziente Möglichkeit große Menge an Anfragen zu bearbeiten oder eine Vielzahl an Identitätsdaten zu speichern. Zusätzlich ist Manipulationsresistenz von großer Bedeutung, da die Identitätsdaten stets authentisch sein müssen, um beispielsweise Dokumentenfälschung oder Identitätsdiebstahl zu vermeiden. Ebenso können finanzielle Transaktionen hiermit abgewickelt werden, was in der analogen Welt oft im Zusammenhang mit der Dokumentenausstellung stattfinden. Ein Beispiel hierfür sind die Gebühren beim Beantragen eines Reisepasses oder die Strafgebühr für das zu späte Neubeantragen eines abgelaufenen Ausweises. Die Möglichkeit sog. 'smart contracts' - also eigene Programme - zu schreiben ist eine Eigenschaft, die nicht in allen DLT's gegeben ist. Dennoch wird diese Eigenschaft an dieser Stelle erwähnt, da einige Blockchains wie Ethereum Letzteres unterstützen und somit einem Software-Entwickler die Chance geben fehlende Software im Identitätsmanagementsystems zu implementieren.

Diese Merkmale von DLT machen es zu einer idealen Technologie für die Umsetzung von Self-Sovereign-Identity. Sie ermöglicht eine sichere, vertrauenswürdige und selbst-

bestimmte Verwaltung von Identitätsinformationen, wodurch Benutzer die Kontrolle über ihre Identität zurückerlangen und die Notwendigkeit von zentralen, vertrauenswürdigen Dritten verringert wird.

2.6 Warum SSI Lösungen überlegen sind

Es gibt zahlreiche Argumente für das Verwenden von SSI-Lösungen (Sovrin, uPort, etc), da diese viele Nachteile von herkömmlichen (internationalen [Google, Facebook, Amazon, etc] oder lokalen [Verimi, netID, etc]) Identitätsplattformen kompensieren [Id2a]

- Kontrolle: Sowohl lokale als auch internationale IDP's geben dem Nutzer kaum Kontrolle/Einfluss über seine Daten, während SSI Lösungen dem Nutzer die alleinige Kontrolle geben und dieser somit in der Lage ist seine Daten beliebig zu modifizieren oder Zugriff einzuschränken.
- Datenablage: Sowohl bei internationalen als auch bei nationalen IDP's werden die Daten zentral beim IDP abgelegt. Der Unterschied hierbei ist, das bei nationalen IDP's die Daten innerhalb der EU liegen, da diese an rechtliche Rahmenbedingungen gebunden sind. Bei SSI-IDP's liegen die Daten global verteilt in den Knoten des Netzwerks.
- Sicherheit: Bei herkömmlichen IDP's würde ein erfolgreicher Angriff sämtliche Daten kompromittieren, was bei SSI-IDP's aufgrund der Dezentralität nicht (oder nur sehr schwer) möglich ist.
- Datenschutz: Bei internationalen IDP's wird die DSGVO nicht eingehalten, was bei nationalen IDP's und SSI-IDP's nicht zutrifft.
- Standards: Alle gegebene IDP's erfüllen in der Regel Standards.
- Vertrauen: Ist lediglich bei SSI-IDP's vollständig gegeben, wenn der Nutzer informiert ist.

Es ist zu erkennen, dass SSI-IDP's in den meisten Fällen überlegen sind, jedoch gibt es auch einen großen Vorteil: Der Nutzer muss sich bereit erklären auf neue Technologien zuzugreifen und diese auch zu verstehen, um nicht Opfer von Angriffen zu werden. Dazu gehört das Verwenden von Wallets, Verständnis von Prozessen über das Anfragen/Austellen von VC's und erstellen von VP's. Auch sollte nicht vernachlässigt werden, dass das Erstellen von Transaktionen (Schreiben in der Blockchain) in der Regel mit Gebühren einhergeht. In den folgenden Subkapiteln werden verschiedene SSI-Lösungen vorgestellt und im Anschluss miteinander verglichen.

Kapitel 3

Anforderungsanalyse

3.1 Use-Case

Der zu implementierende Use-Case ist, dass eine Autorität in der Lage sein muss einen Führerschein an einen Nutzer auszustellen. Der Führerschein darf für Niemand anderen als den Nutzer einsehbar sein. Es muss jedoch die Möglichkeit geben, dass eine andere Autorität den Führer auf Authentizität überprüft oder herausfinden kann, ob der Führerschein für die richtige Fahrzeugart ist oder wann der Fahrer den Führerschein erhalten hat. Von Bedeutung ist, dass nicht der ganze Führerschein gezeigt werden muss, sondern dass selektiv entweder einzelne Attribute einsehbar gemacht werden oder Anfragen bejaht/verneint werden. Sollte der Fahrer Fehlverhalten zeigen, so kann die Autorität den Führerschein wieder entziehen.

Folgende Anforderungen gelten für das Identitätsmanagementsystem und entsprechen den von der OECD festgelegten Eigenschaften [Id2d] [Id2b].

3.2 Funktionale Anforderungen

- Widerruf: Nachdem ein Credential ausgestellt wurde muss die Möglichkeit existieren diesen zu widerrufen. Dies kann durch den Holder oder Issuer des Credentials geschehen. Dazu gehört auch, dass Credentials nach einer bestimmten automatisch ungültig sein können.
- Überprüfbarkeit: Es muss möglich sein, dass ein Verifier Credentials überprüfen kann. Auch muss nachverfolgbar sein wann der Credential durch wen erstellt wurde. Wenn beispielsweise ein Führerschein als Credential ausgestellt wurde, so muss ein Kontrolleur in der Lage zu sein diesen auf Gültigkeit und korrekte Attribute zu prüfen.
- Selektive-Veröffentlichung: Dem Nutzer muss es möglich sein lediglich einzelne Claims zu veröffentlichen. In der analogen Welt zeigt der Kunde beispielsweise

den kompletten Ausweis, obgleich nur das Alter überprüft werden möchte. Demnach soll die Funktion existieren ausschließlich eine Submenge von Informationen eines Credentials offen zu legen.

3.3 Nicht-Funktionale Anforderungen

- Vertraulichkeit: Credentials müssen vor unbefugten Zugriffen geschützt sein. Gemeint ist hiermit, dass unberechtigte Nutzer Credentials weder schreiben noch lesen sollten.
- Integrität: Credentials dürften nur autorisiert modifiziert werden. Dies betrifft Claims eines Credentials oder den gesammten Credential.
- Non-Replay: Operationen dürfen nicht erneut ausführbar sein. Dies bedeutet, dass beispielsweise die Credential-Zuweisung nicht mehrfach ausgeführt werden kann, da ansonsten ein Holder entweder doppelte Credentials erhält oder ein zweiter (und somit unberechtigter) Holder ebenso den Credential fälschlicherweise empfangen kann.
- Nichtabstreitbarkeit: Die Transaktionen müssen dokumentiert sein, sodass Aktivitäten nicht abgesprochen werden können. Beispielsweise sollte der Issuer nicht in der Lage sein abzustreiten, dass ein Credential erstellt wurde.

3.4 Technische Anforderungen

Als technische Anforderung wird lediglich festgelegt, dass eine DLT verwendet werden soll, was in dieser Arbeit durch die Blockchain realisiert wird. Bei den Templates für die Credentials, die beschrieben welche Attribute ein Dokument hat (z.B. Fahrzeugtyp im Führerschein) muss sich an den *W3C Standard für Verifiable Credentials* [1] gehalten werden.

[1] https://www.w3.org/TR/vc-data-model/

Kapitel 4

Darstellung existierender Lösungen

4.1 Luniverse

Luniverse [Id2e] ist eine Firma, die im Mai 2018 gegründet wurde und BaaS (Blockchain as a Service) anbietet, indem ein umfassendes Portfolio an Blockchain Lösungen angeboten wird, die verschiedene Problematiken der "Blockchain-Umstellung"lösen. 2022 bedient Luniverse nach eigenen Angaben über 2000 Firmenkunden. Dabei gibt es vier Kategorien, die allesamt das Sidechain-Konzept implementieren.
Dabei ist eine Sidechain [Id2c] [Id3f] eine separate Blockchain, die

- parallel zur ursprünglichen Blockchain läuft. Dies bedeutet, dass die Knoten der Hauptkette und Knoten der SideChain unabhängig voneinander agieren.
- bidirektional mit dem Mainnet verknüpft ist. Dies hat den Vorteil, dass Informationen zwischen der SideChain und der Hauptkette in beide Richtungen transportiert werden können. Letzteres passiert beispielsweise dadurch, dass die Hauptkette Vermögenswerte blockiert (’lockt’), um diese auf der SideChain zugänglich zu machen. Diese Aktion kann in beide Richtungen erfolgen.
- ihren eigenen Konsensus-Algorithmus hat. Ein Beispiel hierfür ist, dass die Hauptkette Proof-of-Stake verwendet und die SideChain Proof-of-Authority.
- ihre Sicherheit "**nicht**" von der Elternkette erbt (hängt unter anderem mit dem oberen Punkt zusammen)
- erweiterte Funktionen implementieren kann. So ist es möglich, dass die SideChain beispielsweise ’zero-knowledge-proofs’ anbietet auch wenn die Hauptkette diese Funktion nicht besitzt.
- neue Anwendungsfälle umsetzt. Während die Hauptkette evtl. nur ein Gerüst mit den Standard-Funktionen anbietet (Transaktionen von Vermögenswerten) kann es sein, dass die SideChain sich auf Supply-Chain-Management speziallisiert oder Finanzoperationen abbildet (Vermögenswerte leihen, Zinsen, etc)

ohne dabei die Elternkette zu beeinträchtigen.

1. Luniverse NFT
   Mit diesem Dienst wird versprochen, dass die 'Luniverse-Multichain-NFTs' sowohl die Emissionskosten entfernen als auch die Umweltprobleme lösen. Dabei finden die Transaktionen zunächst nur auf der Luniverse-Sidechain statt, wobei die NFTs auch auf das Ethereum-Ökosystem übertragen. Es wird das Erstellen ('minten') von NFTs nach dem ERC721 Interface, ein Marktplatz für NFTs, geteiltes Eigentum von NFT, eine Datenbank für Metadaten und vieles mehr angeboten. Hierbei wird eine sehr hohe Energieeffizienz durch das Verwenden des LPOA-Konsensus-Algorithmus garantiert, welcher keine Transaktionskosten generiert. Weitere Vorteile der Luniverse-Sidechain sind: Mehr Transaktionen pro Sekunde, Anbieten einer REST-API, ein CLI und viele weitere

2. Loyalty Point
   Dieser Dienst ist das Blockchain-Äquivalent von Treuepunkten. Es wird angeboten Treuepunkte dezentral zu organisieren, wodurch eine bessere Kundenakquise, eine stärkere Bindung von Kunden und das Übertragen von Treuepunkten zwischen Unternehmen angeboten wird.

3. Trace
   Dieser Dienst ist dafür zuständig jegliche Aktivität auf der Blockchain aufzuzeichnen, um im Anschluss Datenintegrität, Verlaufsaufzeichnungen in Zeitreihendatenbanken und Datenverfolgung anzubieten.

4. DID
   Unter diesem Dienst wird der ganze Prozess rund um DID's, DID-Dokumenten, Claims, etc. abgebildet. Es wird eine REST-API zur Verfügung gestellt, jedoch wird auch das UI benötigt. Beispielsweise werden Templates für Credentials oder für Verifier im UI erstellt. Davon abgesehen existieren HTTP-Endpunkte für das Überprüfen der Stati von widerrufenen Credentials, das Ausstellen, das Widerrufen und das Verifizieren von Credentials. Endpunkte für das Generieren von DIDs, DID-Docs und Ähnlichem stehen nicht zur Verfügung.

Bemerkenswert ist, dass die oben genannten Dienste sich gegenseitig ergänzen. So kann die SSI oder NFT Aktivität mittels dem Trace-Dienst analysiert werden. Vor allem NFT's, die auch das Konzept des Eigentums implementieren könnten für ein umfangreiches IDP von Bedeutung sein.
Auf technischer Sicht arbeitet die API mit nur einem Parameter in den Anfragen: *did-ProjectId*. Dieser ist ein Pfad zu einer in der UI erstellen Ressource (Beispielsweise Templates für das Anfragen oder Verifizieren von Credentials). Davon abgesehen verwendet Luniverse eigene DID-Methoden, was an folgender beispielhaften Holder-DID deutlich wird: 'did:ethr:lunvs:0x1111111111111111111111111111111111111111'

4.2 Dock

'Dock Certs (certs.dock.io)' ist ein Webdienst, der es ermöglicht Credentials anzufragen, Templates für Credentials, Templates für die Verification von Credentials, das Erstellen von Issuer-Profilen und das visuelle Gestalten von VC im PDF-Format.
Um Credentials anzufragen, zu erstellen oder zu verifizieren muss zunächst ein Template erstellt werden. Dies passiert durch das UI. Hier kann man entweder Templates als JSON importieren (auch mittels öffentlichen URL's von *public ledgern* wie IPFS [https://ipfs.tech]) oder im UI jedes Attribut einzeln konfigurieren. Anschließend muss ein Issuer-Profil erstellt werden. Dieses besteht aus einem öffentlichen Namen, einer optionalen Beschreibung und einem 'DID-Type', welcher entweder vom Typ 'dock' ist oder 'polygonid' (siehe nächste Lösung). Im Anschluss können Credentials ausgestellt werden, die zuvor erstellen Templates entsprechen müssen. Daraufhin kann der Nutzer alle Empfänger des VC entweder manuell eintragen oder als Excel importieren lassen. Hierbei kann auch eine Email angeben werden, sodass der Rezipient eine Email erhält, um das VC zu akzeptieren. In der Email befindet sich ein QR-Code, welcher beispielsweise mit der PolygonID-App gescannt werden kann, wodurch das VC auf den Wallet übertragen wird. Ebenso lassen sich im UI Verifikationstemplates erstellen, die beispielsweise festlegen welcher Claim (zum Beispiel eine Note) vorhanden sein muss.
Alle diese Funktionen lassen sich auch über die REST-API ausführen. Zusätzlich (und dies ist nicht über das UI möglich) lassen sich DID's und DID-Docks erstellen und Löschen, VC widerrufen, etc.
Es wird angegeben, dass alle Formate W3C-Standard-konform sind.

Anders als bei Luniverse handelt es sich bei Dock nicht um eine Sidechain, sondern eine eigenständige Blockchain. Dock wirbt damit eine Blockchain entwickelt zu haben, die für die dezentrale Identität optimiert ist [Id3a]. Der verwendete Konsensus-Algorithmus lautet GRANDPA (GHOST-based Recursive Ancestor Deriving Prefix Agreement). Eine wichtige Eigenschaft für diesen Kontext ist, dass die Blockproduktion und das Bestätigen von Transaktionen möglichst effizient gestaltet ist. Zudem werden Validato ren (oder auch Nominatoren/Staker) benötigt, die eine bestimmte Menge der Kryptowährung der Blockchain als Kaution hinterlegen müssen, um am Konsensprozess teilzunehmen. Ein Validator ist ein sog. "full node", also ein Knoten des Netzwerks der eine vollständige Kopie der gesamten Blockchain enthält. Hierbei arbeiten bis zu 50 Validatoren parallel, um die Integrität des Netzwerks zu bewahren. Zudem unterstützt die Blockchain das Konzept des Stakings, was bedeutet, dass Teilnehmer dem Netzwerk eine Menge an Kryptowährung hinterlegen, um das Netzwerk sicherer zu gestalten. Als Belohnung erhält der Teilnehmer an bestimmte Menge an 'Dock' (so lautet die Einheit der Kryptowährung).

4.3 PolygonId

PolygonId (https://polygon.technology/polygon-id) ist eine Plattform, die einem Entwickler die Möglichkeit gibt 'eine vertrauenswürdige und sichere Beziehung zwischen Nutzern und dApps (dezentrale Applikationen) zu bauen, die den Prinzipien von SSI und **privacy by default** folgen' [Id3b]. 'Privacy by default' beschreibt hierbei, dass Claims mittels ZK-proofs (zero-knowledge) überprüft werden können.

4.3.1 Polygon

PolygonId dient als Identitätsinfrastruktur auf der Polygon-Blockchain, welche wiederum eine Layer-2-Lösung ist [RAN+23], was bedeutet, dass

- Ethereum (Layer 1) das übergeordnete Netzwerk ist
- Polygon direkt mit der Hauptkette (Ethereum Mainnet) verbunden ist und dessen Sicherheit und Konsensmechanismen nutzt
- Polygon darauf abzielt die Transaktionsfrequenz zu erhöhen, indem Transaktionen auf außerhalb der Hauptkette ausgelagert werden
- Polygon als Skalierungslösung zk-Rollups verwendet. Demnach wird eine große Menge an Transaktionsdaten auf der - vom Polygon-Framework zur Verfügung gestellten - Sidechain in ein Batch verpackt und mittels zk-Proofs auf der Hauptkette verifiziert, ohne die Details der Transaktionen zu kennen. Eine alternative zu zk-Rollups sind sog. 'optimistic rollups', was bedeutet, dass das Mainnet den Batch 'beglaubigt' [Id3g]. In dem Falle, dass sich Fehler oder Unstimmigkeiten innerhalb des Batches befinden kann eine sog. challange eingereicht werden, wodurch Transaktionen überprüft werden durch Teilnehmer des Netzwerks.

Zusätzlich bietet das Polygon weitere Mechanismen an, wobei im Folgenden nur solche genannt werden, die für das Implementieren von SSI-IDPs von Bedeutung sind oder werden könnten:

- Polygon POS Chain [Id3c]: Ist die Hauptkomponente der Polygon-Plattform und ist eine Proof-of-Stake Blockchain (zuvor auch Matic Network genannt). In der Kryptowährung Matic (MATIC) werden die Transaktionsgebühren gezahlt. Die über PolygonID getätigten Transaktionen finden hier statt und profitieren von Skalierbarkeit, hoher Geschwindigkeit, Benutzerfreundlichkeit und Konnektivität zu Ethereum. Auch wenn es sich hierbei um eine Sidechain handelt ist es bisher **nicht** möglich DIDs von Polygon auf Ethereum zu übertragen, was bedeutet, dass VP's nur innerhalb von Polygon existieren und andere Ethereum-basierende Lösung diese nicht verifizieren können. Am 21.Juli 2023 wurde angekündigt, dass

durch eine Partnerschaft PolygonId auch auf der Ethereum-Blockchain verwendet werden kann [Id3c].

- Polygon Bridge: Ist ein Mechanismus um Vermögenswerte (also Kryptowährungen) zwischen Ethereum und Polygon-Sidechains zu bewegen. Dies kann von Bedeutung sein, wenn beispielsweise beim Anfordern eines VC eine Gebühr bezahlt werden muss (was in der analogen Welt regelmäßig stattfindet) und die Vermögenswerte nicht auf einer Blockchain abgesperrt werden sollen.
- Zusätzlich gibt es mehrere Tools für Software-Entwickler eigene Sidechains zu implementieren (Polygon SDK) oder ein Testnetz, um Smartcontracts und ähnliches zu deployen und zu testen bevor auf dem Mainnet gearbeitet wird.

All die oben genannten Punkte sind für PolygonId von Bedeutung, da alle Schreib und Verifizierungsvorgänge (das Registrieren oder Updaten von DID's, Verifizieren von Claims, etc.) auf der Polygon-Blockchain stattfinden.

4.3.2 Usecases und technische Daten über PolygonId

Nach eigenen Angaben sind Usecases von PolygonId

- Digitale Demokratie (eine Person, eine Stimme)
- Passwordless Login
- private Zugangskontrolle
- weitergehende technische Features, wie 'Sybil-Proof'[1]-Protokolle oder Frameworks, die die Entwicklung von SSI-Applikationen vereinfachen.

Auf technischer Sicht bietet PolygonId ein Framework um folgendes Dreieck zu implementieren:

Es ist zu erkennen, dass Polygon SDK's für Entwickler zur Verfügung stellt um jeweils (Identity-) Holder, Verifier und Issuer zu implementieren. Für Wallets gibt es zudem eine Wallet-App. Für Issuer müssen Issuer Node gehostet werden, wodurch VC's ausgestellt oder entfernt werden können oder *Identity States* on-chain veröffentlicht werden können. Nach der Dokumentation [Id3d] muss der Node zusammen mit

- der Applikation
- einem 'Vault' für Key-Management-Services (Beispielsweise zum Speichern des privaten Schlüssels des Issuers)

[1] Attacke wo Identitäten gefälscht werden - gefährlich bei Mehrheitsabstimmungen oder zum Verlangsamen des Netzwerks

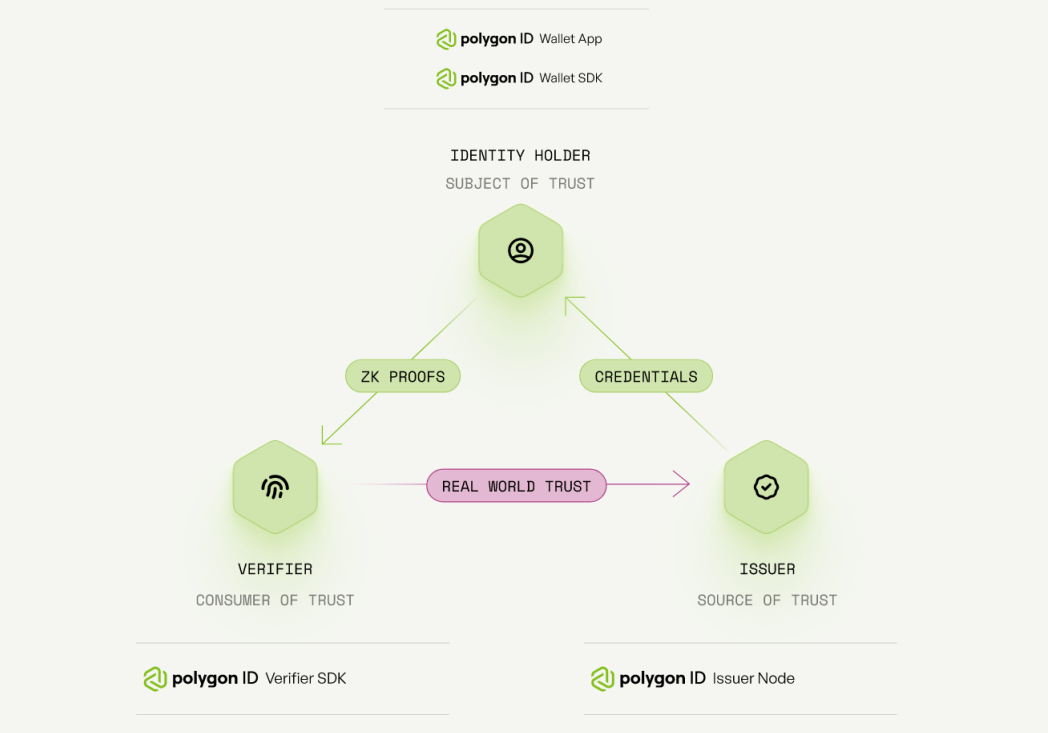

Abbildung 4.1: Zusammenspiel von Issuer, Holder und Verifier auf technischer Sicht in PolygonId [Id3b]

- einem Cache-Service (Bespielsweise zum Zwischenspeichern von Templates, die einmalig von IPFS gedownloaded werden)
- einer Datenbank zum persistieren aller operativen Daten

Die Identitätsdaten bestehen aus folgenden drei Informationen, die jeweils in sog. 'Sparse Merkle Trees' gespeichert werden:

1. **Claims Tree**: Ist ein Baum, der Informationen über ausgestellte Claims einer Identität **öffentlich** speichert.
2. **Revocations Tree**: Ist ein Baum, der die Noncen (zuvor zufällig generierte Zahlen) einer Widerrufung von Claims **privat** speichert.
3. **Roots Tree**: Speichert **öffentlich** die Historie der Wurzeln der Claim Trees.

Der Identitätsstatus lässt sich somit wie folgt definieren:

$$IdState = Hash(ClR||ReR||RoR) \tag{4.1}$$

wobei:

- Hash einer Hash-Funktion entspricht
- ClR der Wurzel des Claims Tree entspricht

- ReR der Wurzel des Revocation Tree entspricht
- ReR der Wurzel des Roots Tree entspricht

Der oben genannte Identitätsstatus wird als einziges Datum in der Blockchain gespeichert.

4.3.3 zk-Proof von PolygonId

Ein Feature von PolygonId, dass es von anderen SSI-Frameworks hervorhebt ist, dass es Verifikation mittels sog. *zero-knowledge-proofs* zur Verfügung stellt. Die Idee hierbei ist, dass dem Verifier die Richtigkeit eines Claims verifizieren kann **ohne** Informationen über den eigentlichen Claim zu erhalten. Als Beispiel wird folgender **nicht anonymisierter / nicht verschlüsselter/ selektiv freigegebener** Claim betrachtet:

```
[...]
  "credentialSubject": {
    "id": "did:example:456",
    "degree": {
      "type": "Gehalt",
      "name": "ArbeitgeberName",
      "frequence": "monthly",
      "value": 4000
    }
  }
[...]
```

Sollte nun beispielsweise ein Kreditinstitut (der Verifier) überprüfen wollen, ob der Kunde mehr als 2000€ verdient, so ist es in erster Linie irrelevant, ob die Person 5000€ oder 10000€ verdient und würde nur unnötig die Privatsphäre beeinträchtigen. Polygon bietet hierzu eine Lösung und implementiert ein zero-knowledge-Konzept, wo der Verifier über eine Query-Sprache lediglich über den Besitz des Credentials informiert wird ohne tatsächlich Informationen zu erhalten. Die Anfrage kann on-chain und off-chain formuliert und sieht wie folgt aus (erstellt mit dem ZK-QueryBuilder[2] von PolygonId):

```
[...]
    const proofRequest: protocol.ZKPRequest = {
    id: 1,
    circuitId: 'credentialAtomicQuerySigV2',
    query: {
      allowedIssuers: ['*'],
      type: 'EmployeeData',
      context: 'https://raw.githubusercontent.com/0xPolygonID/tutorial-
          examples/main/credential-schema/schemas-examples/employee-data
          /employee-data.jsonld',
      credentialSubject: {
        monthlySalary: {
          $gt: 2000,
        },
      },
```

[2] https://schema-builder.polygonid.me/query-builder

```
14      },
15    };
16  [...]
```

Das Ergebnis der Anfrage ist ein Objekt, welches 'zero-knowledge-proof-information' speichert und dem Verifier zusichert, dass der zu verifizierende Nutzer den Claim tatsächlich besitzt.

4.4 Sovrin

Sovrin (Foundation) [3] ist eine gemeinnützige Organisation, die etabliert wurde, um das Governance-Framework 'Sovrin Network' zu administrieren, welches ein öffentlicher Dienstleistungsservice ist zum Ermöglichen der SSI [Id3e]. Dabei ist die Funktion der Sovrin Foundation das Sicherstellen des öffentlichen und global zugänglichen Sovrin-Identity-Systems. Der verteilte Speicher des Netzwerks ist die eigen entwickelte Blockchain, die vollständig darauf abzielt SSI zu implementieren. Folgende vier Eigenschaften seien für ein erfolgreiches SSI-System notwendig und wurden in Sovrin Network eingebaut [Id4d]:

- Governance: Alle Stakeholder können dem Netzwerk vollständig vertrauen
- Performance: Das Netzwerk soll auf dem gleichen Level skalieren wie das Internet selber
- Zugänglichkeit: Das Netzwerk ist für jeden zugänglich
- Privatsphäre: Die sichersten Standards werden implementiert

Ein Mechanismus zum verbessern der Privatsphäre ist das 'pseudonymous by default'-Konzept, welches Sovrin umsetzt. Hierbei wird für jede Relation (beispielsweise PersonX->Arbeitgeber, PersonX->Verkäufer, etc) eine eigene DID verwendet. Somit ist mehr Privatsphäre gegeben und auch die Sicherheit ist verbessert.
Die Sovrin-Blockchain ist eine sog 'permissioned' Blockchain, was bedeutet, dass ein Knoten, der Transaktionen tätigen oder validieren möchte (also ein sog. 'Validator-Knoten') eine Erlaubnis braucht. Organisationen, die diese Erlaubnis erhalten haben nennen sich bei Sovrin 'Stewards'. Diese sind global verteilt und aktuell gibt es 50 Stewards in 13 Ländern und 6 Kontinenten [Id4c].

4.4.1 Technische Grundlagen

Sovrin versucht sich an DNS[4] zu orientieren, da DNS knapp eine Milliarde Einträge hat und über 100 Milliarden anfragen pro Tag bearbeitet [Id4e]. Übertragen auf den

[3] https://sovrin.org/
[4] Domain Name System

Identitätskontext - unter der Annahme dass Identitäten mehrere DID's besitzten - muss das Netzwerk möglicherweise täglich Trillionen von Anfragen bearbeiten. Während DNS keinen Konsensus-Algorithmus verwendet ist dies bei der Sovrin-Blockchain jedoch nicht der Fall. Um dennoch eine gute Skalierung zu implementieren werden zwei verschiedene Arten von Knoten im Netzwerk verwendet:

- **Validator-Knoten**: Ist eine kleine Mengen an Knoten im Netzwerk deren Funktion es ist Transaktionen zu akzeptieren
- **Observer-Knoten**: Eine größere Menge an Knoten, die Lese-Anfragen bearbeiten

Issuer, Verifier und Holder erreichen diese Knoten über Agenten. Diese Agenden können beispielsweise Mobilanwendungen sein und die Aufgabe haben mit dem Sovrin-Network in Verbindung zu stehen. Agenten können Identitätstransaktionen stellvertretend für den Identitätsträger tätigen und kommunizieren direkt mit anderen Agenten. Dies funktioniert, da der Agent Zugriff auf den privaten Schlüssel hat. Demnach können beispielsweise DID-Dokumente modifiziert werden oder Transaktionen getätigt werden. Zudem werden private Daten (wie Claims) ebenso auf dem Agenten gespeichert, während öffentliche Daten auf der Blockchain abgelegt werden.
Ähnlich wie bei PolygonId kann von ZK-Proofs Gebrauch gemacht werden oder Anfragen modelliert werden.

4.5 ShoCard

Die letzte hier vorgestellte Lösung basiert auf der Bitcoin Blockchain. Das besondere hierbei ist, dass Bitcoin keine Möglichkeit bietet Smartcontracts oder ähnliches zu implementieren. Dennoch gibt es einen Prozess, der SSI in der Blockchain implementiert. Hierbei verschachtelt Shocard die DID, ein existierenden Credential und zusätzliche Identitätsattribute in einer Bitcoin-Transaktion [DP18]. Shocard verwendet einen zentralen Server der folgende drei Vorgänge implementiert:

- **Bootstrapping**: Ist der Prozess bei dem eine neue Shocard erstellt wird. Hierbei erstellt die Shocard-App ein neues asymmetrisches Schlüsselpaar und scannt die Credentials über die Kamera. Die Daten werden im Anschluss verschlüsselt und auf dem Gerät gespeichert. Ein signierter Hash wird im Anschluss in einer Bitcoin Transaktion gespeichert, wobei die erhaltene Transaktionsnummer zu der ShoCardID des Anwenders wird. Ebenso wird diese Information in der App gespeichert. Ist dieser Prozess abgeschlossen ,so kann der Identitätsträger mit Issuern kommunizieren, um weitere Identitätsattribute anzufragen. Dieser Prozess nennt sich 'certification'.
- **Certification**: Um Identitätsattribute zu erhalten muss der Identitätsträger nachweisen, dass er die Daten kennt, die den Hash generiert hatten (wurde in der App

persistiert) und den Schlüssel, der zur Signatur verwendet wurde. Als Ergebnis liegt eine neue Transaktion vor, die die Attribute und die ShoCardID gehasht enthält. Da der Provider die Transaktion getätigt hat, muss er die Transaktionsnummer zusammen mit dem signierten Klartext der neuen Attribute mit dem Nutzer teilen. Diese werden erneut in der Applikation lokal gespeichert. Da der Nutzer die Credentials jedoch nicht verlieren möchte im Falle, dass der Zugriff zur Applikation verloren geht, bietet ShoCard die Möglichkeit Credentials verschlüsselt in einem 'Envelop' zu speichern (ein Speicher den ShoCard verwaltet). Der Envelop kennt den Schlüssel zur Verschlüsselung nicht.

- Validation: Dieser Prozess findet statt, wenn ein Identitätsträger den Besitzt von Credentials nachweisen möchte. Hierbei gibt der Nutzer die Referenz des Envelops und den zur Verschlüsselung verwendeten Schlüssel. Somit kann der Verifier die Korrektheit überprüfen. Somit wird klar, dass kein ZK-Proof vorliegt, da dem Verifier sämtliche Informationen des Credentials vorliegen.

4.5.1 Probleme von ShoCard

Mit der Verwendung von ShoCard gehen jedoch auch Probleme einher. Beispielsweise wird offensichtlich, dass - wenn der **ShoCard central server** verwendet wird (der die Envelops speichert) - eine Abhängigkeit zu ShoCard existiert. Sollte das Unternehmen verschwinden und die Server offline nehmen, so verlieren Nutzer ihren Zugang zu den Credentials, wenn sie keine lokale Kopie besitzen. Diese Eigenschaft nimmt ShoCard einen Teil seiner Dezentralität. Zudem existiert das Problem, dass ShoCardIDs nur unidirektional identifizieren (Also Identität -> ShoCardId). Es gibt keine dezentrale Registry, die von einer ShoCardID auf eine DID oder etwas Vergleichbarem auflöst. Zudem gibt es noch mehrere weitere Probleme [DP18].

4.6 Vergleich existierender Lösungen

In der Folgenden Tabelle werden die Lösungen von **Luniverse**, **Dock**, **Polygon** und **Sovrin** miteinander verglichen. Betrachtet wird dabei die Blockchain und dessen Konsensus-Algorithmus, ob ein Knoten ohne zusätzliche Erlaubnis ein Validator werden kann, ob ZK-Proofs verfügbar sind und wo die Credentials gespeichert werden. Im Anschluss werden die Transaktionskosten betrachtet.

- Luniverse
  - Blockchain: Luniverse-Blockchain
  - Konsensus-Algorithmus: LPOA
  - Permissionless?: No

  - ZK-Proofs?: Yes
  - Speicherung: Wallet, Blockchain

- Dock
  - Blockchain: Dock-Sidechain
  - Konsensus-Algorithmus: GRANDPA
  - Permissionless?: No
  - ZK-Proofs?: Yes
  - Speicherung: Wallet, Blockchain

- PolygonId
  - Blockchain: Polygon PoS
  - Konsensus-Algorithmus: Proof-of-Stake
  - Permissionless?: Yes
  - ZK-Proofs?: Yes
  - Speicherung: Wallet oder WalletSDK, Blockchain

- Sovrin
  - Blockchain: Sovrin-Network
  - Konsensus-Algorithmus: Plenum
  - Permissionless?: No
  - ZK-Proofs?: Yes
  - Speicherung: WalletSDK, Blockchain

- Shocard
  - Blockchain: Bitcoin-Blockchain
  - Konsensus-Algorithmus: Proof-of-work
  - Permissionless?: Yes
  - ZK-Proofs?: No
  - Speicherung: Blockchain, Shocard central server, App

4.7 Transaktionskosten

Transaktionen auf der Blockchain sind Prozesse, die in den verteilten Speicher schreiben. Da Transaktionen von Validator-Knoten überprüft werden, muss an das Netzwerk eine Gebühr gezahlt werden. Diese variiert je nach Blockchain, Zustand der Blockchain und Auslastung.

- Luniverse: Luniverse gibt an, dass keine Transaktionsgebühren existieren.
- Dock: Gibt an, dass keine Gebühren anfallen für die Credentialerstellung. Transaktionen für das Erstellen von Schemas oder Widerrufen von Credentials seien sehr gering [Id4a]
- Polygon: Bei Polygon lassen sich die Transaktionskosten sehr genau berechnen. Dabei hängt es von zwei Faktoren ab, wie teuer die Transaktion wird [Id4b]:
  - Menge an Gas: Dies ist eine Metrik für die Leistung, die das Netzwerk für das Ausführen der Transaktion zur Verfügung stellt. Im Falle vom Polygon wird die 'Ethereum Virtual Machine' (EVM) verwendet, welche Platten-Verwendung, CPU-Verwendung, etc. misst und so die Menge an Gas berechnet.
  - Gaspreis: Dieser Faktor hängt von der Netzwerkauslastung ab. Prinzipiell gibt es drei Modelle zwischen denen man entscheiden muss: Standard, Fast und Rapid. Dabei wird aufsteigend die Transaktionsdauer geringer (30-10 bei Standard und 5-10 bei Rapid). Analog dazu steigt auch der Gaspreis. Auf PolygonScan[5] können die Gaspreise historisch betrachtet werden.
- Sovrin: Credential und DID-Erstellung sind kostenlos. Für alles andere (also Revokation, Schemas, etc) gibt es jeweils für TestNet und MainNet eigene Preise.
- ShoCard: Transaktionskosten auf der Bitcoin Blockchain werden in Satoshi ($1 * 10^{-6} Bitcoin$) pro Byte berechnet. Demnach kostet die Transaktion mehr, je nachdem viele Daten in die Blockchain geschrieben werden. Im Jahr 2023 betrugen die Kosten für eine Transaktion im Durchschnitt zwischen einen und drei Euro [Id4f]. Es ist jedoch anzunehmen, dass Transaktionen, die Data-Anchoring betreiben, mehr kosten, da mehr Daten geschrieben werden.

4.8 Konsensus-Algorithmus

Konsensus-Algorithmen werden verwendet, um einen einheitlichen Zustand des Netzwerk zwischen den Knoten festzulegen. Hierbei gibt es verschiedene Ausführungen, die im Folgenden betrachtet werden:

- LPOA: Dieses Akronym steht für "Luniverse's Proof of Authority"[Id5d]. Hierbei handelt es sich um einen Proof-of-Authority-Algorithmus, wobei ein Validator nicht anhand seiner zur Verfügung gestellten Rechenkraft oder Kryptowährung bemessen wird, sondern an seiner Identität. Potentielle Validatoren (beispielsweise Unternehmen, Institutionen, Regierungen, etc.) müssen sich bei einer Einrichtung oder Organisation, die das PoA-Netzwerk betreibt, bewerben oder eingeladen werden. Diese entscheiden anhand der Reputation und der Vertrauens-

[5] https://polygonscan.com/gastracker

würdigkeit des Bewerbers, ob dieser als Validator fungieren darf. Sollte Letzteres erlaubt werden, so werden Verträge verfasst, um die Verantwortlichkeiten im Netzwerk festzulegen. Im Anschluss kann mit dem Validieren von Blöcken begonnen werden. Sollte Fehlverhalten des Validators festgestellt werden, kann dieser aus dem Netzwerk ausgeschlossen werden [Id5c].

- GRANDPA: Dieser Algorithmus lässt sich ebenso als Proof-of-Authority-Algorithmus klassifizieren

- Proof-of-Stake: Proof of Stake (PoS) ist ein Konsensusmechanismus in Blockchain-Netzwerken, der verwendet wird, um Transaktionen zu validieren und neue Blöcke zur Blockchain hinzuzufügen. Im Gegensatz zu Proof of Work (PoW), bei dem Miner rechenintensive Aufgaben lösen müssen, um Blöcke zu erstellen, basiert PoS auf dem Konzept des 'Stakings' oder des Einsatzes von Kryptowährungen.

  1. **Validatoren und Staking:** In einem PoS-Netzwerk gibt es keine Miner im herkömmlichen Sinne. Stattdessen gibt es Validatoren. Um ein Validator zu werden, müssen Benutzer eine bestimmte Menge der Kryptowährung des Netzwerks als Einsatz hinterlegen. Dieser dient als Garantie dafür, dass der Validator korrekt arbeitet.
  2. **Blockvalidierung:** Wenn eine Transaktion im Netzwerk eingereicht wird, wird ein Validator zufällig ausgewählt, um die Transaktion zu validieren und einen neuen Block hinzuzufügen. Die Wahrscheinlichkeit, ausgewählt zu werden, hängt oft von der Menge des gestakten Vermögens ab, was bedeutet, dass Benutzer mit größeren Stakes eine höhere Chance haben, ausgewählt zu werden.
  3. **Belohnungen:** Validatoren, die korrekt arbeiten und Transaktionen ordnungsgemäß validieren, erhalten Belohnungen in Form von Transaktionsgebühren und neuen Kryptowährungseinheiten, die dem System hinzugefügt werden. Diese Belohnungen werden oft proportional zur Höhe des gestakten Vermögens des Validators verteilt.
  4. **Bestrafungen:** Wenn ein Validator betrügerisches Verhalten zeigt oder versucht, das Netzwerk zu schädigen, kann er seine gestakten Vermögenswerte verlieren.

  PoS [Id5b]bietet mehrere Vorteile, darunter eine geringere Umweltauswirkung im Vergleich zu PoW, da keine rechenintensiven Aufgaben erforderlich sind und eine höhere Skalierbarkeit. Trotz der vielen Vorteile von Proof-of-Stake gibt es auch einige Nachteile [Id5b]:

  1. **Zentralisierungstendenzen:** PoS kann zu einer gewissen Zentralisierung führen, da Teilnehmer mit großen Stakes mehr Einfluss haben und wahrscheinlicher ausgewählt werden, um Transaktionen zu validieren und Blö-

cke hinzuzufügen. Dies könnte zu einer Konzentration der Netzwerkvalidierungsmacht führen.

2. **Reichtumsungleichheit:** PoS belohnt Benutzer proportional zu ihren gestakten Vermögenswerten. Dies könnte die bestehende Reichtumsungleichheit in Kryptowährungen weiter verstärken, da reichere Benutzer größere Stakes halten können und somit mehr Belohnungen erhalten.
3. **Gefahr von Auslagerung (Staking as a Service):** Einige Benutzer könnten ihre Staking-Aktivitäten an Dritte auslagern, um die Belohnungen zu maximieren. Dies könnte dazu führen, dass reiche Benutzer Dritte beauftragen, um ihre Stakes zu verwalten, was die Dezentralisierung gefährden könnte.
4. **Geringe Anreize für Aktivität:** Einige PoS-Netzwerke könnten Schwierigkeiten haben, Benutzer dazu zu ermutigen, aktiv am Netzwerk teilzunehmen, da sie bereits gestakte Vermögenswerte besitzen und möglicherweise keine zusätzlichen Anreize sehen, aktiv Transaktionen zu validieren.
5. **Schwierigkeiten bei der Wahl der Validatoren:** Die Auswahl von zuverlässigen und ehrlichen Validatoren kann eine Herausforderung darstellen. Es müssen Mechanismen implementiert werden, um sicherzustellen, dass betrügerische oder bösartige Validatoren erkannt und bestraft werden.
6. **Sicherheitsprobleme bei geringer Beteiligung:** PoS-Netzwerke könnten anfällig für Angriffe sein, wenn die Beteiligung gering ist und nur wenige Validatoren vorhanden sind. In solchen Fällen könnten Angreifer leichter die Kontrolle über das Netzwerk erlangen.
7. **Verlust von gestakten Vermögenswerten:** Bei fehlerhaftem Verhalten oder betrügerischen Handlungen können Validatorn Strafen auferlegt werden, einschließlich des Verlusts ihrer gestakten Vermögenswerte.

- Proof of Work (PoW) ist ein Konsensusmechanismus, der in mehreren Blockchain-Netzwerken verwendet wird. Er dient dazu, Transaktionen zu validieren, neue Blöcke zur Blockchain hinzuzufügen und das Netzwerk vor verschiedenen Arten von Angriffen zu schützen. Im Folgenden sind die Grundlagen von PoW zu betrachten:

  1. **Transaktionen validieren:** Im Netzwerk werden Transaktionen von Benutzern gesammelt und in einen Pool gestellt, der darauf wartet, in einen neuen Block aufgenommen zu werden. Diese Transaktionen müssen validiert werden, um sicherzustellen, dass sie den Regeln des Netzwerks entsprechen.
  2. **Rätsellösung:** PoW erfordert von den sogenannten Minern, mathematische Rätsel zu lösen, die als "Proof of Work"bezeichnet werden. Diese Rätsel sind so konzipiert, dass sie eine erhebliche Rechenleistung erfordern und gleichzeitig leicht zu überprüfen sind, sobald sie gelöst sind. Minen ist also ein

wettbewerbsfähiger Prozess, bei dem die Miner darum konkurrieren, das Rätsel zu lösen.

3. **Wettbewerb und Belohnungen:** Die Miner verwenden ihre Rechenleistung, um das Rätsel zu lösen. Der erste Miner, der das Rätsel erfolgreich löst, kann einen neuen Block erstellen und ihn mit den validierten Transaktionen füllen. Dieser neue Block wird dann der Blockchain hinzugefügt. Als Belohnung für ihre Arbeit erhalten die Miner neue Kryptowährungseinheiten (z. B. Bitcoin) sowie die Transaktionsgebühren, die von den Benutzern für die Validierung ihrer Transaktionen gezahlt werden.
4. **Sicherheit und Dezentralisierung:** PoW schützt das Netzwerk, indem es für Angreifer sehr teuer macht, die Mehrheit der Rechenleistung im Netzwerk zu kontrollieren. Je mehr Rechenleistung ein Angreifer benötigt, desto schwieriger wird es, das Netzwerk zu übernehmen. Dies trägt zur Sicherheit und Dezentralisierung bei, da viele Miner weltweit am PoW-Prozess teilnehmen.
5. **Schwierigkeitsanpassung:** Das PoW-System passt die Schwierigkeit des zu lösenden Rätsels automatisch an die Gesamtrechenleistung des Netzwerks an. Dies stellt sicher, dass die Zeit zwischen der Erstellung neuer Blöcke ungefähr gleich bleibt, unabhängig davon, wie viele Miner im Netzwerk aktiv sind.

Obgleich PoW mit vielen Vorteilen einhergeht stehen auch Kritikpunkte im Raum: Unter anderem ist PoW höchst ineffizient, da wie bereits erwähnt nur der erste Miner, der das Rätsel löst belohnt wird, obwohl viele andere Miner gleichzeitig am selben Problem arbeiteten. Dadurch werden große Mengen an Rechenleistung verschwendet.

Kapitel 5

System-Design und Implementierung

5.1 Architektur

5.1.1 Von Anforderungen zum System-Design

Den Anforderungen nach soll ein dezentrales Identitätsmanagementsystem entwickelt werden, dass die Möglichkeit bietet Credentials (wie z.B. Ausweisdokumente) auszustellen, zu überprüfen und selektiv zu veröffentlichen unter den - in Kapitel 3.1 genannten - Bedingungen. Daher werden zwei der drei Komponenten (Issuer, Verifier, Holder) implementiert. Der Issuer wird als CLI (Command-Line-Interface) implementiert und der Verifier als REST-API. Der Holder muss nicht implementiert werden werden, da die Plattformen jeweils Identity-Wallets anbieten. Als Use-Case wird die Erstellung eines Führerscheins illustriert. Dieser Führerschein (der Credential) wird von einer Autorität (Issuer) ausgestellt und kann von einem beliebigen Kontrolleur/Polizist/etc. (Verifier) überprüft werden. Die in Kapitel 4 angeführten Lösungen können alle diese Funktionen implementieren, daher wird anhand anderer Faktoren entschieden, welches Framework verwendet wird.

5.1.2 Entscheidung über Framework

Um sich für eine Architektur festzulegen, muss zunächst die Entscheidung über die darunterliegende Plattform getroffen werden. Dabei stehen die in Kapitel 5 betrachteten Lösungen zur Auswahl: Luniverse, Dock, PolygonId, Sovrin und Shocard. Für die Entwicklung des Prototypen wird PolygonID aus folgenden Gründen verwendet:

- Polygon (als unterliegende Blockchain) zeigt folgende Vorteile auf [Id5a]:
  - Die Blockchain skaliert hervorragend mit einer steigenden Anzahl von Trans-

aktionen
  - Geringe Transaktionskosten
  - Hohe Interoperabilität mit Ethereum
  - Etablierte Plattform und weite Verbreitung im Markt
- PolygonId:
  - Möglichkeit zum Widerruf von Informationen gegeben
  - Informationen sind überprüfbar
  - Selektive-Disclosure implementierbar
  - Alle nicht-funktionalen Anforderungen implementierbar
  - Unterstützt W3C Standard für VC's
  - Credential Exchange erfolgt nach Identity-Foundation Standard

5.1.3 Grobe Architektur

Prinzipiell besteht die Architektur aus drei Komponenten: Einem Verifier, einem Issuer und dem Holder. Jede dieser drei Komponenten laufen unabhängig voneinander und können mit fremd-implementierten Instanzen kommunizieren. Das hier dargestellte Szenario ist das Erstellen, Übertragen und Verifizieren von einem digitalen Führerschein auf der Blockchain. Die System-Architektur kann in der Grafik 5.1 betrachtet werden.

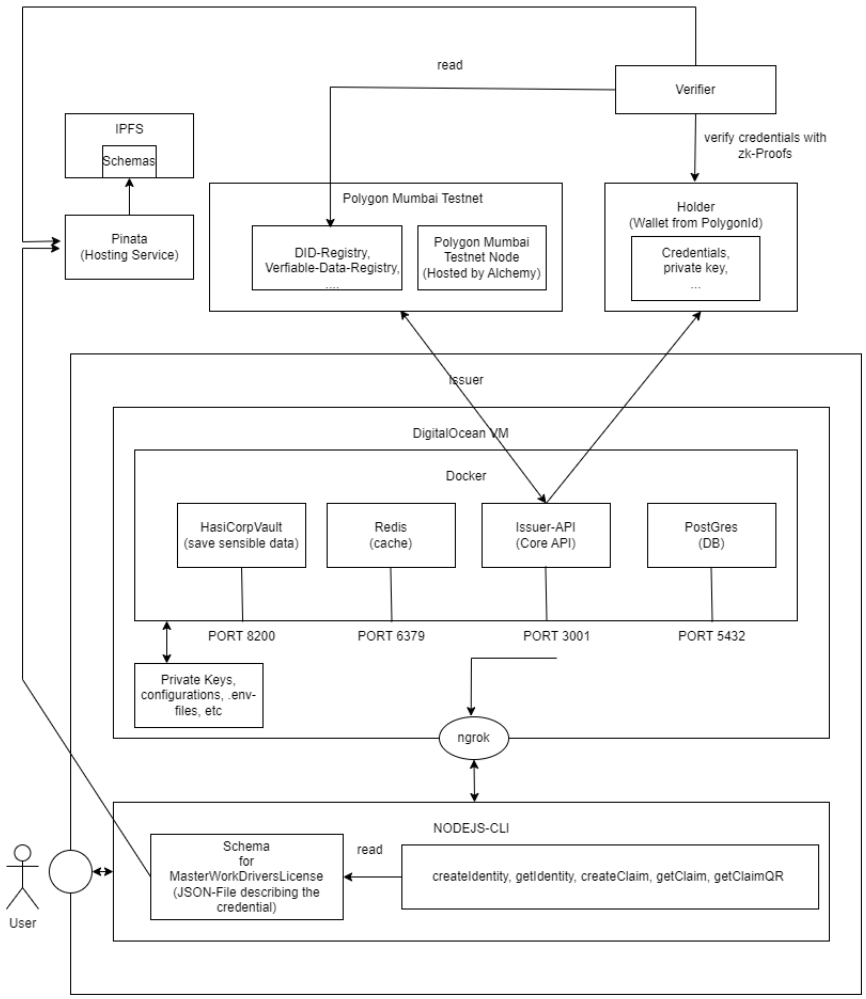

Abbildung 5.1: System-Design des Prototyps

Es ist zu erkennen, dass der Issuer die zentrale Komponente ist. Um den Issuer zu hosten wird eine virtuelle Maschine von DigitalOcean [1] verwendet. VMs werden bei DigitalOcean auch Droplets genannt.

Auf dem Droplet läuft eine Docker-Instanz, die alle in der Grafik darstellen Komponenten orchestriert. Das einzige Image, das auch von extern erreichbar sein muss ist das 'issuer-api'-Image, da dieses die REST-API Anfragen erhält. Für Letzteres wird 'ngrok' [2] verwendet. Ngrok hat die Funktion die lokale IP-Addresse (127.0.0.1) öffentlich durch einen Tunnel zugänglich zu machen. Dieser Tunnel wird vom NodeJs-Cli verwendet, um Anfragen über die API an den - von Alchemy [3] gehosteten - Netzwerk-Knoten weiterzuleiten. Alchemy wird benötigt, da mit Knoten des Polygon-Netzwerks kommuniziert werden muss und Alchemy diese zur Verfügung stellt. Die Schemas müssen öffentlich zugänglich sein, da sie unter anderem im Verifier und Issuer referenziert werden. Um dem ursprünglichen Gedanken der Dezentralität treu zu bleiben werden auch die

[1] https://www.digitalocean.com/
[2] https://ngrok.com/
[3] https://www.alchemy.com/

Schemas dezentral - über IPFS [4] - gespeichert. Als Schnittstelle zwischen IPFS und der Anwendung wird Pinata [5] verwendet. IPFS ist eine Platform, die dem dezentralen Speichern von Inhalten dient und Pinata hostet IPFS-Knoten, damit diese nicht lokal aufgesetzt werden müssen. Das dort gehostete Schema ist in Grafik 5.1 illustriert.

```
    [...]
    "MasterWorkDriversLicense": {
      "@context": {
       [...]
        "TypeOfVehicle": {
          "@id": "polygon-vocab:TypeOfVehicle",
          "@type": "xsd:string"
          [...]

          "enum": [
            "Car",
            "Truck",
            "Scooter",
            "Motorcycle"
          ],

        },
        "YearOfReceipt": {
          "@id": "polygon-vocab:YearOfReceipt",
          "@type": "xsd:integer"
        }
      },
      [...]
    }
  }
  ]
}
```

Listing 5.1: Schema des Credentials

Es ist zu erkennen, dass der Führerschein für den Prototypen aus zwei Attributen besteht:

- YearOfReceipt: Ein Integer, der das Jahr darstellt an dem der Holder den Führerschein erworben hat
- TypeOfVehicle: Ein String, der den Typ des Vehicle beschreibt. Hierbei gibt es vier Typen:
  - Car
  - Truck
  - Scooter
  - Motorcycle

Zum Erstellen der Schemata stellt PolygonId einen Schema-Builder zur Verfügung [6].
Der Issuer stellt den Führerschein aus, indem er einen QR-Code erstellt, der die Informationen über den Credential enthält (also Typ und Jahr). Dieser QR-Code kann

[4] https://ipfs.tech/
[5] https://www.pinata.cloud/
[6] https://schema-builder.polygonid.me/

mit der PolygonID-App gescannt werden. Der Holder kann die Informationen über den Führerschein einsehen und akzeptieren. Im Anschluss befindet sich der Credential im Identity-Wallet. Details über diesen Prozess sind in den folgenden Kapitel beschrieben. Der nächste Schritt ist, dass ein Verifier einen Query definiert, der wie folgt aussehen könnte:

```
{[...]
  id: 1,
  circuitId: 'credentialAtomicQuerySigV2', // algorithmus zum Erstellen
      des zk-Proofs
  query: {
    allowedIssuers: ['*'],
    type: 'MasterWorkDriversLicense', // im Schema definierter Typ
    context: 'https://ipfs.io/ipfs/QmTSd6saivXHysRopQdM1yswp2qyFwobL7
        fwuFpkVTS8gd',
    credentialSubject: {
      YearOfReceipt: {
        $lt: 2022,
      },
    },
  },
  [...]
}
```

Der hier dargestellte Query überprüft primär, ob der Führerschein des Holders älter als vom Jahr 2022 ist. Indirekt fragt er ebenso ab, ob der Nutzer den beschriebenen Credential besitzt und ob dieser noch gültig ist. Ist dies der Fall, so wird der zk-Proof an den Verifier gesendet.

5.2 Kommunikation der Komponenten

Die Kommunikation der Komponenten ist in Abbildung 5.2 dargestellt.

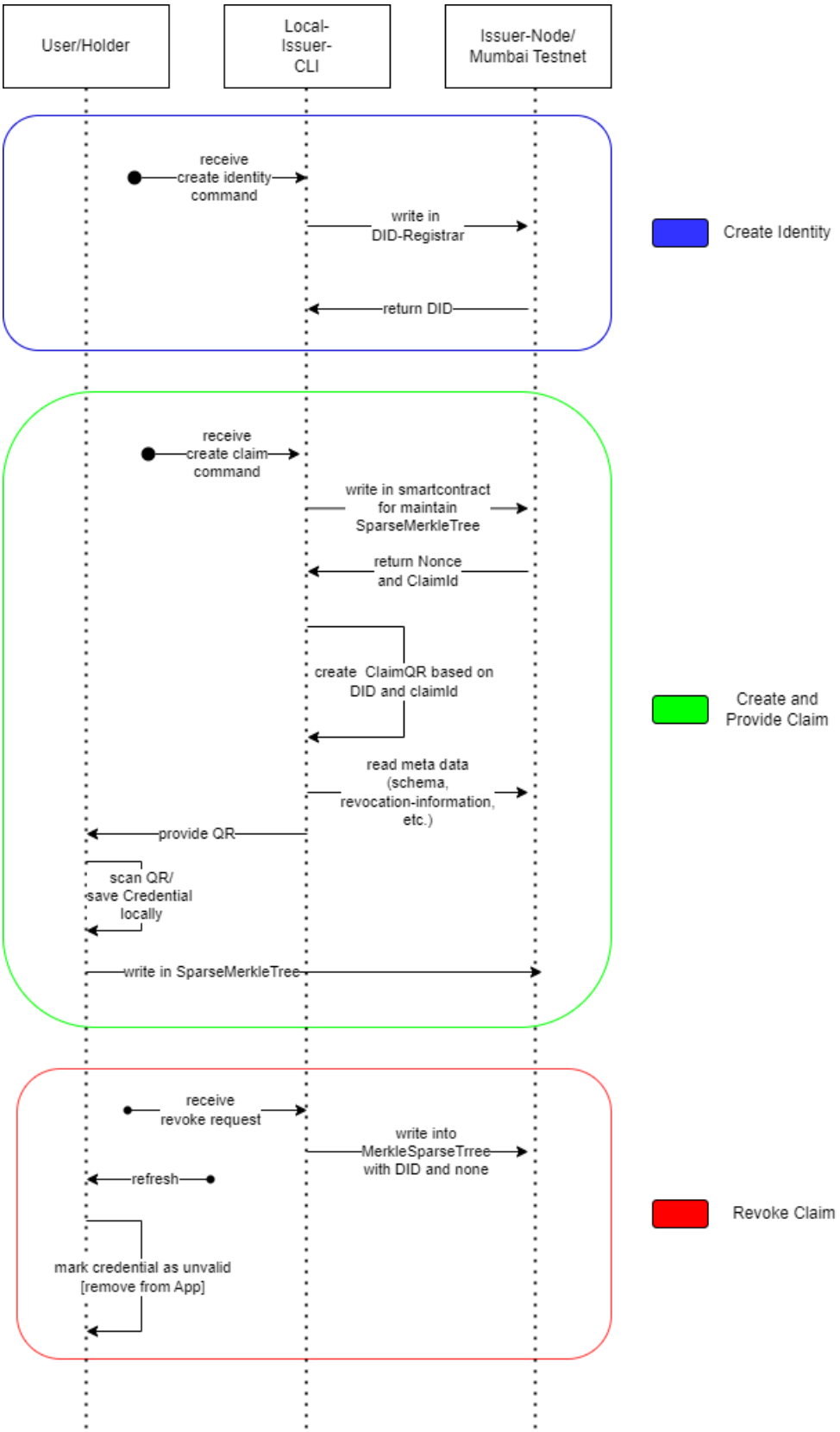

Abbildung 5.2: Kommunikationsdiagramm des Issuers, Holder und der Blockchain

Dabei sind jeweils die Prozesse zum Erstellen einer Identität, Erstellen und Zuwei-

sen von Credentials und das Widerrufen von Credentials illustriert. Die drei Akteure sind jeweils der User/Holder (in diesem Falle die PolygonID-App), das Issuer-CLI (mit Node-JS implementiert) und der Issuer-Node (über DigitalOcean gehostet). Es ist zu erkennen, dass jede der drei Aktivitäten durch das CLI initialisiert wird. In der 'create Identity' Aktivität wird eine Identität erstellt, wobei in das DID-Registrar geschrieben wird. Diese Komponente ist dafür zuständig DID's und DID-Documents zu verwalten und befindet sich in der Blockchain. Bei dem Erstellen und Zuweisen von Claims spielt die Blockchain insofern eine wichtige Rolle, als dass in den 'Sparse-Merkle-Tree' geschrieben wird (siehe 4.3.2). Auch auf das Widerrufen von Credentials trifft dies zu.

5.3 Der Issuer

Der Issuer kann im Polygon-Framework auf zwei Weisen realisiert werden:

- 'On-Chain': Das Ausgabe der Credentials geschieht hierbei 'on-chain', also innerhalb der Blockchain. Implementiert wird die Logik mittels Smart-Contracts, was bedeutet, dass die Logik beliebig erweitert werden kann. Die anzuwendende Programmiersprache ist Solidity [7], was der gleichen Sprache entspricht wie im Ethereum Ökosystem. Durch die Smartcontracts werden - die in Usecases und technische Daten über PolygonId erwähnten - Zustandsbäume gespeichert. Auch Identitäten können so generiert werden. In der Dokumentation werden zwei Szenarien besprochen: öffentliche und private Anwendungsfälle, die beide in der Blockchain realisiert werden können. Auch wenn der Issue-Prozess in der Blockchain stattfindet, so werden private Credentials nicht on-chain generiert. Dies passiert offline und ein Beweis für die Validität wird in der Blockchain gespeichert.
- 'Off-Chain': Hierbei werden die Credentials in einem Issuer-Knoten erstellt, der eine API zur Verfügung stellt. Um den Knoten zu hosten wird ein Droplet von 'Digital Ocean' verwendet, welches sich in Frankfurt befindet. Nachdem die Knoten-Software installiert wurde und alle Parameter konfiguriert wurden (privater Schlüssel von Issuer, CPU-Typ, URL's, Export von Variablen, RPC-Endpunkt-URL, usw.) kann der Knoten in Betrieb genommen werden. Für letzteren Endpunkt können öffentliche Knoten verwendet werden wie etwa 'https://rpc-mumbai.matic.today'. Dieser ist jedoch vergleichweise langsam und es gehen Vorteile von privaten Endpunkten verloren, wie Monitoring und bessere Debugging-Möglichkeiten. Aus diesen Gründen wird Alchemy (https://www.alchemy.com/) verwendet. Der private Schlüssel wird aus Sicherheitsgründen in einem Tool zur Speicherung sensibler Daten verwendet [8]. Redis [9] dient als Cache. Alle drei Komponenten befinden

[7] https://soliditylang.org/
[8] Vault by HashiCorp
[9] https://redis.io/

sich jeweils in einem Container in Docker [10]. In einem vierten Container befindet sich ein Server, der über eine REST-API Endpunkte zur Verfügung stellt. Die für diesen Prototypen relevanten Aktivitäten sind das Generieren und Lesen von Identitäten, das Erstellen und Schreiben von Claims und das Generieren von QR-Codes, die der potentielle Holder mit seiner Wallet-App scannen kann. Alle die zuletzt genannten Funktionen wurden in einem CLI (Command Line Interface) in JavaScript zur Verfügung gestellt. Dabei werden in Abbildung 5.3 die Befehle mit einem jeweiligen Beispiel illustriert.

```
<C:\Users\robert.davidoff@sap.com\Desktop\masterthesis\Issuer> node app
                                -a createIdentity
Create identity request received
  {
    identifier: 'did:polygonid:polygon:mumbai:2q...", // did of identity
    state: {[...]}
  }

<C:\Users\robert.davidoff@sap.com\Desktop\masterthesis\Issuer> node app
                                -a getIdentity
'did:polyganid:polygon:mumbai:2q...'
'did:polygonid:polygon:mumbai:3z...'

<C:\Users\robert.davidoff@sap.com\Desktop\masterthesis\Issuer> node app
                                -a createClaim
                                -t Motorcycle -y 2017
                                -i did:polyganid:polygan:mumbai:2q...
{id: '76219657-7445-11ce-b5e3-82423128885" } // id of claim

<C:\Users\robert.davidoff@sap.com\Desktop\masterthesis\Issuer> node app
                                -a getClaim - 76219657-7445-11cc-b5e3-8242ac128885
                                -i did:polyganid:polygon:mumbai:2q...
[...]
credentialschema: {
  id: 'https://ipfs.io/ipfs/Qmd6v3nfTusgcnotwTnmphwKEsmo/VzrNsmxcKxgGNMZd",
  type: 'JsonSchema2023'
  credentialstatus: {[...]},
  credentialSubject: {
  TypeofVehicle: 'Motorcycle',
  YearOfReceipt: 2017,
  id: 'did:polygonid:polygon:mumbai:2qEJURNEGK6wxJRk4Q1BGj5j5Rz1cTRp7EvEoPaTr', // did of issuer
  type: 'MasterWorkDriversLicense
}
[...]
```

Abbildung 5.3: CLI Beispiele

Alle Implementierungen verwenden intern die REST-API des Issuer-Knoten. Alternativ ist es möglich dem Droplet eine 'reserved IP' zuzuweisen. Mittels PolygonScan (https://mumbai.polygonscan.com) und dem Monitoring Tool bei Alchemy sind die Aktivitäten auch als Transaktionen einsehbar. Nachdem der QR-Code gescannt wurde, kann man in der App den Credential einsehen, was in Abbildung 5.4 ersichtlich wird.

[10] https://www.docker.com/

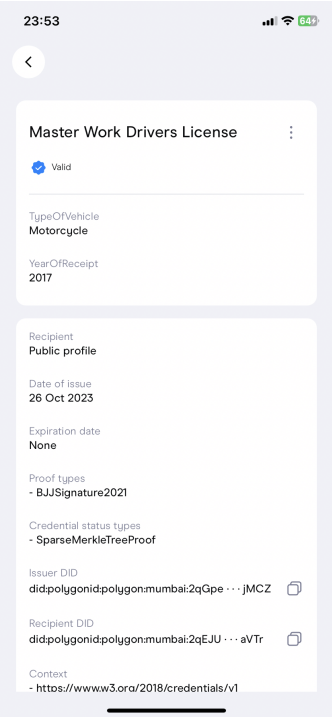

Abbildung 5.4: App Credential Beispiel

Es ist zu erkennen, dass die DID vom Issuer/Receiptient und die im CLI angegeben Attribute im Credential nachzuverfolgen sind.

5.4 Der Verifier

Der Verifier hat die Funktion die Credentials zu überprüfen. Hierbei wird zunächst überprüft, ob der Nutzer den Credential besitzt und ob er noch gültig ist, im Falle dass ein Auslaufdatum angegeben wurde oder der Credential widerrufen wurde. Drei Typen von Anfragen können gestellt werden:

- Ohne Query: Entspricht einer Anfrage, die lediglich überprüft, ob der Credential existiert und valide ist
- Mit Query:
  - Für Abfrage-Operatoren ungleich ’eq’ (lt , gt, in, etc)
  - Für Abfrage-Operation gleich ’eq’(entspricht ==). Dieser Typ von Anfragen wird auch als ’Selective Disclosure’ bezeichnet.

Gebaut werden können diese Anfragen mittels einem vom Polygon zur Verfügung gestellten Tool [Id5f]. Hierbei wird das Schema des Credentials geladen und über ein UI können die Attribute und Operatoren ausgewählt werden und ein JSON-Objekt wird als Ergebnis zurückgegeben.

Der oben beschriebe Query wird statisch im Programmcode festgehalten.
Über eine REST-API (gehostet auf port 8080) werden zwei Schnittstellen zur Verfügung gestellt:

- /api: Dieser Endpunkt ist dafür zuständig den Request-Body zu bauen und zu speichern (wird später erneut benötigt). Zudem wird ein QRCode von dem Request abgespeichert, der mit der PolygonId-App gescannt werden kann. Der Request-Body sieht wie folgt aus:

```
{
  "from": "did:polygonid:polygon:mumbai:2qF57iujBWKeAGc2koCV56yW5S
     1SfPtFsCgDHzGRdW",
  "typ": "application/iden3comm-plain-json",
  "type": "https://iden3-communication.io/authorization/1.0/
     request",
  "body": {
    "reason": "Check if year of receipt is older than 2020",
    "callbackUrl": " https://df4c-165-1-191-123.ngrok-free.app/api
       /callback?sessionId=1",
    "scope": [
    {
      "id": 1,
      "circuitId": "credentialAtomicQuerySigV2",
      "query": {
        "allowedIssuers": ["*"],
        "type": "MasterWorkDriversLicense",
        "context": "https://ipfs.io/ipfs/QmTSd6saivXHysRopQdM1yswp
           2qyFwobL7fwuFpkVTS8gd",
        "credentialSubject": {
          "YearOfReceipt": {
            "$lt": 2020
          }
        }
      }
    }
    ]
  }
}
```

- /qr: Dient als Endpunkt, um die QR-Codes zu lesen
- /callback: Dieser Endpunkt von dem Identity-Wallet kontaktiert, um einen zk-Proof entgegenzunehmen und diesen zu verifizieren.

Nachdem der QR-Code von der App gescannt wurde, wird die Anfrage illustriert. Ebenso wird die URL des Issuers angezeigt und der Name des Credentials. Der Holder initiert die Verifizierung durch das Drücken des ÄpproveButtons. An dieser Stelle wird ein zk-Proof generiert und an den Verifier geschickt. Der Holder schickt hierfür einen POST-Request an die angegebene 'callbackURL'. An dieser Stelle führt der Server die tatsächliche Verifikation des zk-Proofs aus. Der Ablauf hierfür wird im folgenden Code beschrieben:

```
26    const ethURL = 'https://polygon-mumbai.g.alchemy.com/v2/
          PRIVATE_API_KEY';
27    const contractAddress = "0x134B1BE34911E39A8397ec6289782989729807a4"
          //public verification contract adress
28
29    const ethStateResolver = new resolver.EthStateResolver(
30    ethURL,
31    contractAddress,
32    );
33
34    const resolvers = {
35      ['polygon:mumbai']: ethStateResolver,
36    };
37
38
39    // fetch authRequest from sessionID
40    const authRequest = requestMap.get(`${sessionId}`);
41
42    // EXECUTE VERIFICATION
43    let path_full = path.join(__dirname, './circuits-dir')
44    const verifier = await auth.Verifier.newVerifier(
45    {
46      stateResolver: resolvers,
47      circuitsDir: path_full,
48      ipfsGatewayURL:"https://app.pinata.cloud/gateway/amethyst-official-
          duck-350?pinataGatewayToken=
          F5FDkLi66xtMWFQ0BCjZ1EGceaWSvbQ1uvkioaYqi9Iq4lSc8CRpMi-2EXVSa1f"
           // gateway used to save the ZK-Proof
49    }
50  );
```

Man kann erkennen, dass Pinata nicht nur verwendet um die Schemata zu speichern, sondern auch als Gateway um die zk-Proofs abzulegen. Auch ist zu erkennen, dass eine URL für einen Polygon-Knoten notwendig ist, da zum einen auf Widerruf geprüft wird und zum anderen der unter 'contractAddress' gespeicherte Smart-Contract kontaktiert wird. Der Holder erhält im Anschluss die Information, dass mit Erfolg verifiziert wurde.

5.4.1 On-Chain Verifikation

An dieser Stelle sollte kurz erwähnt werden, dass es ebenso möglich wäre die Verifikation 'on-chain', also als Smart-Contract in der Blockchain ablaufen zu lassen. Analog dazu gibt es die Alternative auch den Issuer 'on-chain' zu implementieren. Hierbei wird - der in Solidity geschriebene - Smartcontract in der Blockchain deployed. Über einen Polygon-Knoten (beispielsweise gehostet über Alchemy) wird der - vom Holder erstelle - zk-Proof an den Smartcontract gesendet welcher die Richtigkeit überprüft. Der Vorteil hierbei wäre, dass der Gedanke der Dezentralität intensiviert wird und mehr Transparenz über die Verifikation gegeben wird. Als Nachteil ergibt sich, dass für das Deployment Vermögenswerte (MATIC) gezahlt werden müssen und Solidity weniger Schnittstellen bietet als Javascript.

5.5 Der Holder

Der Holder ist die Komponente mit der geringsten Relevanz für diesen . Es gibt jeweils eine App für Android und IOS, die folgende Funktionen haben [Id5g]:

- SSI implementieren
- Credential entgegennehmen, speichern und warten
- zk-Proofs erstellen
- mit Issuer und Verifier kommunizieren
- Recovery-Funktion mittels 'seed-phrase'

Wenn ein Entwickler seinen eigenen 'Identity Wallet' implementieren möchte, hat er die Auswahl zwischen der Flutter-SDK (https://flutter.dev/) und einer Android SDK. Auch steht eine REST-API verschiedener Anbieter zur Verfügung, die Funktionen anbieten, die der Wallet benötigt, wie das Erstellen von Identitäten oder zk-Proofs. Für diesen Prototypen wurde kein eigener Identity-Wallet implementiert, da der von Polygon-ID zur Verfügung gestellte Wallet bereits allen Anforderungen genügt und eine eigene Implementierung keinen Mehrwert liefert.

5.6 Interaktion zwischen den Komponenten

Um die Interaktion zwischen den Komponenten zu illustrieren wird folgende Grafik verwendet 5.5.

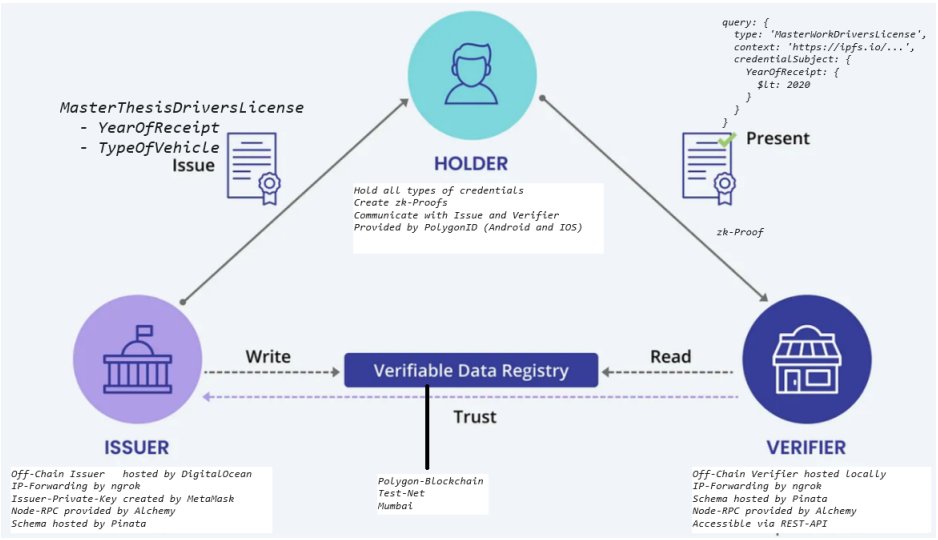

Abbildung 5.5: Interaktion zwischen den Komponenten [Id6e]

Bei der Abbildung 5.5 handelt es sich zunächst um das Konzept 'Triangle of Trust', dass mit den Details des Prototypen erweitert wurde. Unterhalb jeder der drei Komponenten werden technische Details zur Realisierung angegeben. Es ist zu erkennen, dass das Konzept abhängig davon ist, dass der Issuer korrekte Credentials angibt. In dem Szenario, dass der Issuer kompromittiert wurde oder bösartig ist würde folgende Situation eintreten:

- Es werden fehlerhafte, unglaubwürdige erstellt.
- Das Schreiben von Credentials in die Blockchain gilt als Transaktion und daher würden Kosten entstehen
- Der Holder ist im Besitzt dieser falschen Credentials und könnte (evtl. sogar unwissentlich Identitätsdiebstahl begehen)
- Der Verifier vertraut der 'Verifiable Data Registry' und würde fehlerhafte Credentials als richtig attestieren

Daher ist die Korrektheit des Issuer von elementarer Bedeutung.
Ein schematischer Ablauf von der Erstellung der Identität des Issuer hin zur erfolgreichen Verifikation eines Credential des Holder sieht wie folgt aus 5.6.

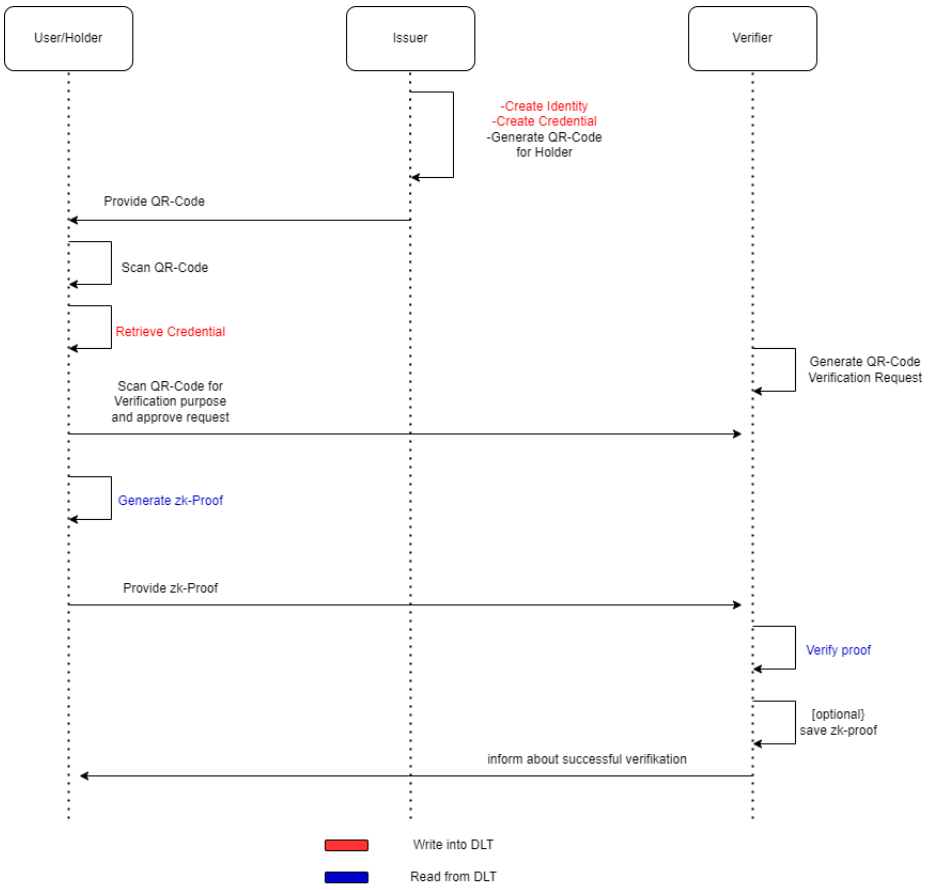

Abbildung 5.6: Schematischer Ablauf des Prozesses

Rote Aktivitäten beinhalten Transaktionen in der Blockchain und blaue Aktivitäten beinhalten Schreibprozesse. Außerdem ist zu sehen, dass viele Prozesse auf anderen Speichern arbeiten, wie z.B. verteile Speicher oder der lokale App-Speicher. Ein Unterschied zu Abbildung 5.2 ist, dass der Verifier ebenso dargestellt ist und zwischen Lese- und Schreibzugriffen auf dem DLT unterschieden wird.

Kapitel 6

Evaluation

6.1 Festlegen der Evaluations Metriken

Folgende Metriken werden im folgenden Betrachtet:

- Laufzeit
- Sicherheit

6.1.1 Metrik - Laufzeit

Um die Laufzeit der Operationen zu messen wurde ein Python-Script entwickelt, welches die Operationen 'createIdentity', 'getIdentity', 'createClaim', 'getClaimQR' und 'revokeClaim' jeweils 100 mal ausführt. Jeder Aufruf startet einen eigenen Subprozess mit dem Python Framework 'subprocess', welcher mit dem Framework 'timeit' gemessen wird. Gemessen wird der Durchschnitt und der Median bei jeweils 100 Ausführungen einer Operation. Die gemessenen Werte sehen wie folgt aus:

Operation_Name	Durchschnitt	Median
createClaim	1.2098522	1.0454559
createIdentity	1.1128388	1.1019622
getClaimQR	1.5566253	1.2646243
getIdentity	1.1031939	1.0355741
revokeClaim	1.4271623	1.526894

Es ist zu erkennen, dass der Median in jedem Fall geringer ausfällt als der Durchschnitt. Dies muss damit zusammenhängen, dass ein Teil der Aufrufe unverhältnismäßig länger dauern.

Der folgenede Code zeigt wie die Messung der Laufzeit stattgefunden haben.

```
1  def runMeasurement(command, cache):
2    runs = 100
3    key = command.split()[3]  # e.g. "node app -a createIdentity" ->
         createIdentity
4    try:
5      runtime = timeit(stmt="subprocess.call('{}')".format(command),
           setup="import subprocess", number=runs, )
6      durations = repeat(stmt="subprocess.call('{}')".format(command),
           setup="import subprocess", number=1, repeat=runs)
7
8      cache[key + "_median"] = calcMedian(durations)
9      runtime = checkisFloat(runtime)
10     if runtime:
11       cache[key + "_average"] = runtime
12   except:
13     pass  # the error here can be ignored
14
15 cache = {}
16 runMeasurement(create_identity_command, cache)
17 runMeasurement(get_identities_command, cache)
18 runMeasurement(create_claim_command.format("Truck", "2000",
       first_identifier), cache)
19 runMeasurement(get_claim_command.format(first_claim, first_identifier
       ), cache)
```

6.2 Sicherheitsevaluierung

Die Komponente, die den größten Schutz in der Implementierung benötigt ist der Issuer. Das SSI-Konzept ist abhängig davon, wie vertrauenswürdig Letzterer ist. Daher findet eine Security-Analyse nach dem STRIDE-Modell [Id5e] statt..

6.2.1 Spoofing

Dieser Angriff beschreibt unberechtigten Zugriff auf die Komponente. Hiergegen wurde implementiert, dass der Zugriff (der über ngrok stattfindet) nur authentifiziert passieren kann: –basic-auth 'ngrok:issecure' –basic-auth='username:password'. Zusätzlich besitzt die verwendete REST-API eine Basic-Authentifikation. Daher ist diese Komponente doppelt geschützt. Es wird stark empfohlen, dass die Passwörter den Richtlinien des BSI entsprechen [Id6b].

6.2.2 Tampering

Dieser Angriff beschreibt die ungewollte Manipulation von Daten. Prinzipiell gibt können zwei Komponenten Daten verändern:

1. Der Holder: Als Besitzer der Daten hat der Holder die Möglichkeit seine Credentials zu löschen. Dieser Prozess würde über die Applikation passieren, welche zunächst über ein potentielles Password des Geräts und zum anderen durch das Passwort der PolygonId-App geschützt ist.

2. Der Issuer: Der Issuer kann ebenso Credentials revoken (widderufen). Aber wie bereits im Kapitel 6.2.1 beschrieben ist der Issuer geschützt.

3. Der Verifier: Hat keine Möglichkeit Credentials zu modifizieren und ist daher nicht weiter zu betrachten

6.2.3 Repudiation

Dieser Angriff beschreibt, dass ein Nutzer eine Aktivität abstreiten kann. Gegen diese Attacke schützt die Blockchain, die jede Transaktion abspeichert. Durch das Monitoring-Tool von Alchemy oder im Blockchain-Browser können diese Transaktionen betrachtet werden. Eine solche Transaktion kann wie folgt aussehen:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "eth_call",
  "params": [
  {
    "from": "0x0000000000000000000000000000000000000000",
    "to": "0x134b1be34911e39a8397ec6289782989729807a4",
    "data": "0x7c1a66de0a79f724bb72300544255781fc350952acb21cb77ea9a719
       c8eebb7d1a055ad0"
  },
  "latest"
  ]
}
```

Es ist zu sehen, dass sowohl übertragene Daten, als auch involvierte Adressen gespeichert werden. Das Schreiben ist ebenso kryptographisch geschützt.

6.2.4 Information disclosure

Dieser Angriff beschreibt das ungewollte Veröffentlichen von Daten. In diesem Prototypen gibt es vier Typen von Daten:

1. Daten im Issuer: Diese Daten sind streng geheim und werden unter anderem in '.env' Dateien oder in einem Vault gespeichert. Darunter sind private Schlüssel, API-Schlüssel für Alchemy, Nutzernamen und Passwörter für UI und API. Zusätzlich liegen alle diese Daten in einer virtuellen Maschine, die mit 2-Faktor-Authentifizierung und einem 20-Stellen Root-Passwort geschützt sind.

2. Daten in der Blockchain: Diese Daten sind bereits öffentlich.

3. Daten im Holder: Die Daten werden in der Wallet-App gespeichert. Darunter sind private Schlüssel und die Credentials, welche durch ein Passwort in der Polygon-App geschützt sind.

4. Daten im Verifier: Der Verifier benötigt lediglich Zugang zu den Schemas, welche öffentlich gespeichert sind, und einen RPC-Node, welcher in Umgebungsvariablen lokal gespeichert ist.

Daher lässt sich erkennen, dass private Daten sicher gespeichert sind und nicht veröffentlicht werde können.

6.2.5 Denial of service

Denial of Service beschreibt eine Attacke, in die komplette Software oder Teile davon ungewollt außer Betrieb genommen werden. Dieser Angriff findet in dem Prototypen nur Anwendung in dem Verifier, da andere Komponenten entweder lokal sind, oder nur der Nutzer Zugriff hat. Im Verifier könnte ein Angreifer probieren den RPC-Node zu überlasten. Jedoch hat Alchemy hiergegen Mechanismen entwickelt [Id6d].

6.2.6 Evelation of priviledge

Dieses Konzept beschreibt, dass ein Nutzer ungewollt seine Rechte auf ein höheres Level setzten kann, um autorisiert zu sein neue Aktivitäten auszuführen. Diese Art der Attacke findet jedoch keine Anwendung, da es keine Levels an Rechten gibt.

6.2.7 Zusammenfassung - STRIDE

KATEGORIE	Abgesichert?
Spoofing	✓
Tampering (Manipulation)	✓
Repudiation (Nichtanerkennung)	✓
Information disclosure (Veröffentlichung von Informationen)	✓
Denial of service	✓
Evelation of priviledge (Erhöhung von Rechten)	✓

Es ist zu erkennen, dass der Prototyp sicher ist im Rahmen des STRIDE-Modells.

6.3 Beantworten der Forschungsfragen

Im Folgenden werden die in Kapitel 1.3 gestellten Fragen beantwortet:

1. Wie erfolgt die Speicherung verschiedener Typen an Daten innerhalb eines Identitätsmanagementsystems?

(a) Wie kann die technische Machbarkeit erreicht werden, Daten sowohl öffentlich zugänglich als auch privat zu speichern? - Es ist möglich Daten privat und öffentlich zu speichern. Dies passiert zum einen bei Shocard in dem Prozess des Bootstrapping (siehe 4.5.1) und zum andere bei zk-Proofs in PolygonId. Jedoch ist es zu empfehlen, private Daten wie Credentials auch privat zu speichern, wie es bereits in verschiedenen Lösungen angewandt wird.

(b) Welche kryptografischen Technologien, wie zum Beispiel Hashing oder Verschlüsselung, sind optimal für die sichere Speicherung von Identitätsdaten geeignet? - Sowohl Hashing als auch (asymmetrische) Verschlüsslung werden für die Speicherung von Identitätsdaten verwendet, je nachdem welche Lösung angewandt wird.

(c) Welche Sicherheitsmaßnahmen sind im Falle einer Datenkompromittierung zu ergreifen, und welche Wiederherstellungsoptionen sind verfügbar? - Sollte der Nutzer den privaten Schlüssel seines Identity-Wallets verlieren, so existiert eine Recovery-Option, bei welcher der User - die bei der Erstellung angezeigten - Passwörter erneut angeben muss. Sollte der Nutzer den privaten Schlüssel an einen Angreifer preisgeben, so ist der Wallet unbrauchbar.

(d) Wie lässt sich die sichere und effektive Ungültigmachung von Informationen (Revokation) gewährleisten? - Die Revokation kann auf drei Weisen geschehen.

- Es kann bei der Credential-Zuweisung eine maximale Gültigkeitsdauer angegeben werden. Läuft diese aus, so wird der Credential widerrufen
- als Issuer besteht die Möglichkeit Credentials zu wiederrufen
- als Holder besteht die Möglichkeit Credentials aus dem Identity-Wallet zu löschen und somit zu widerrufen

2. Welchen Nutzer oder Service-Mehrwert generiert ein dezentrales Identitätsmanagementsystem basierend auf DLT?

(a) Welche konkreten Vorteile ergeben sich für Nutzer und Online-Dienste durch die Implementierung des vorgeschlagenen Identitätsmanagementsystems? - Als Vorteil entpuppt sich für den Nutzer, dass er Herrscher seiner Daten ist und diese eigenständig verwalten kann. Dazu gehört, dass der Nutzer selber entscheiden kann, welche Daten er freigibt. Als Vorteil ergibt sich für einen Online-Dienst, dass er keine eigene personenbezogenen Daten speichern muss und Attribute von Nutzern überprüfen kann, die beispielsweise zum Kauf berechtigen oder Vergünstigungen anbieten.

(b) Inwiefern trägt das System zur Lösung der Problematik von Fake-Usern bei? - Ja das Problem der Fake-User kann hiermit gelöst werden. Beispielsweise

kann bei der Registrierung die Verifizierung eines Credentials erfolgen, wobei dieser Credential nur von einer autorisierten Instanz ausgestellt werden kann.

3. Wie erfolgt die Identitätszuordnung innerhalb des Identitätsmanagementsystems?

   (a) Wie kann eine verlässliche Zuordnung von Identitäten zu Personen auf hohem Sicherheitsniveau erreicht werden? - Diese Aufgabe liegt bei dem Issuer. Dieser muss sicherstellen, dass er keine Credentials zuweist an Autoritäten, denen er nicht vertrauen kann. Um Vertrauen aufzubauen kann beispielsweise ein Know-Your-Customer-Prozess durchgeführt werden.

   (b) Welche Ansätze können entwickelt werden, um das Problem zu lösen, dass Nutzer möglicherweise verschiedene Identitäten auf verschiedenen Plattformen verwenden möchten? - Dieser Faktor stellt sich nicht als Problem, sondern sogar als Feature heraus. Nutzer (auch Issuer) können mehrere DIDs besitzen. Bei Sovrin (siehe 4.4) wird pro Interaktion eine neue DID verwendet, um maximale Privatsphäre zu gewährleisten.

4. Welche technologischen Spezifika sind im Bezug auf die Blockchain als DLT im Identitätsmanagementsystem zu entscheiden?

   (a) Welcher Konsensus-Algorithmus ist am besten geeignet, um die Anforderungen des entwickelten Identitätsmanagementsystems zu erfüllen? - Proof-of-Stake stellte sich als passender Algorithmus herraus (siehe 4.3.1)

   (b) Wie können private Schlüssel sicher gespeichert werden, um unbefugten Zugriff zu verhindern? - Private Schlüssel des Issuers befinden sich in einem Vault (siehe 8) und private Schlüssel des Users befinden sich in der Applikation des Mobilgeräts.

   (c) Inwiefern können Identitätsdokumente erfolgreich als NFTs gespeichert werden, und welche Implikationen ergeben sich daraus? - Ja, die Credentialausstellung ist leicht erweiterbar mit der Zustellung eines NFTs. Dies könnte sowohl on-chain im smartcontract geschehen oder in der REST-API des Issuers. Beide Komponenten sich beliebig erweiterbar.

   (d) Welche Rolle spielen Zero-Knowledge-Proofs bei der Entwicklung eines sicheren Identitätsmanagementsystems? - ZK-Proofs werden vom Holder erstellt und an den Verifier gesendet, welcher diese überprüfen kann. Dies hat den Vorteil, dass der Verifier einen Beweis für den Besitzt eines Claims erhält ohne Details zu kennen.

6.4 Erfüllung der Anforderungen

Die in Kapitel 3.1 gestellten Anforderungen werden die folgt erfüllt:

1. Widerruf: Wird durch den Holder und Issuer ermöglicht
2. Überprüfbarkeit: Der Holder kann zk-Proofs ausstellen, die der Verifier überprüfen kann
3. Selektive Veröffentlichung: Wird zum einen durch 'selective disclosure' und zum anderen durch speichern von Credentials im öffentlichen Profil ermöglicht.
4. Vertraulich: Mittels Passwörter und kryptographischen Verfahren ist diese Anwendung vertraulich
5. Non-Replay: Jede Transaktion/Operation erfolgt authentifiziert und ist daher nicht reproduzierbar
6. Nichtabstreitbarkeit: Ist gegeben (siehe 6.2.3)
7. Diebstahlschutz: Ist gegeben (siehe 6.2.1 und 6.2.4
8. W3C-Standard : Wird erfüllt [1]

Es wird also ersichtlich, dass alle Anforderungen erfüllt wurden.

[1] auch ersichtlich im Schema 'xsd: http://www.w3.org/2001/XMLSchema'

Kapitel 7

Fazit

7.1 Zusammenfassung der Arbeit

Diese Thesis nimmt eine tiefgehende Betrachtung der Problematik vor, die mit herkömmlichen Identitätsmanagementsystemen in der heutigen vernetzten Welt einhergeht. Sie identifiziert und analysiert den wachsenden Bedarf für SSI und legt einen besonderen Fokus auf die technischen Grundlagen dieses Ansatzes.

Zu Beginn wurden die Grundlagen und die historische Entwicklung von Identitätsmanagementsystemen beleuchtet. Diese umfassen sowohl zentrale als auch dezentrale Ansätze, wobei letztere im Kontext von SSI von besonderem Interesse sind. Die Thesis beschäftigt sich ausführlich mit den technischen Grundlagen von SSI, einschließlich der Konzepte von Holder (Identitätsinhaber), Issuer (Identitätsaussteller) und Verifier (Identitätsüberprüfer). Dieser theoretische Rahmen bildet die Grundlage für das Verständnis und die Implementierung von SSI.

Ein zentraler Aspekt dieser Arbeit ist die Bedeutung von Distributed Ledger Technology (DLT) für die Umsetzung von SSI. Die Verwendung von DLT ermöglicht es, Identitätsdaten sicher und dezentral zu speichern, wodurch die Kontrolle über persönliche Informationen wieder in die Hände der Nutzer gelegt wird. Die Thesis untersucht verschiedene DLT-Plattformen und deren Eignung für die Implementierung von SSI, wobei die Wahl auf Polygon als Technologie fällt.

Das Design des SSI-Prototyps wird im Detail erläutert, wobei die Interaktion zwischen den verschiedenen Komponenten des Systems eine entscheidende Rolle spielt. Ein spezifisches Szenario wird ausgewählt und umgesetzt, um die praktische Anwendung von SSI aufzuzeigen.

Eine umfassende Evaluierung des Prototyps erfolgt anhand verschiedener Metriken, darunter die Laufzeit und Sicherheit. Neben den Metriken wird auch die Erfüllung der Anforderungen im Detail betrachtet. Es kristalliert sich heraus, dass die ich Kapitel 1.2 gesetzte Zielsetzung vollständig erfüllt wurde. Es wurde - wie beschrieben - ein dezen-

trales Identitätsmanagementsystem entwickelt, welches DLT implementiert. Hierbei ist - wie gefordert - der Nutzer Herrscher über seine Daten. Sowohl die funktionalen/nicht-funktionalen als auch technischen Anforderungen wurden erfüllt. Zudem sind Sicherheit, Kontrollierbarkeit und Skalierbarkeit berücksichtigt worden. Auch das vorgestellte Szenario kann ausgeführt werden - implementiert wurde es doch Anhand eines Führerschein-Beispiels.

Eine weiterfolgende Analyse findet statt und ergibt, dass eine Anwendung implementiert wurde, die zum einen sicher ist gegen verschiedene Typen an Attacken ist und zum anderen Performant, wobei erwähnt werden muss, dass die Performance größtenteils abhängig ist von der Performance der Issuer-Node-API und der Transaktionsgeschwindigkeit der Blockchain. Davon abgesehen sind alle Security-Richtlinien, Funktionalen/Nicht-Funktionalen/Technischen Anforderungen erfüllt.

7.2 Diskussion

In dieser Arbeit gab es mehrere Optionen das in der Zielsetzung beschriebene Ziel zu implementieren. Wäre die Anwendung beispielsweise nicht im Polygon Framework implementiert worden, so wäre Dock eine valide Alternative gewesen. Der Vorteil hierbei ist, dass keine eigenen Issuer-Knoten oder ähnliches gehostet werden müssen und viele Dienste bereits implementiert wurden, wie das Zuschicken von Credentials über Email - direkt an mehrere Empfänger. Auch ist ein Vorteil, dass die Dock-REST-API viele SSI-Methoden bereits anbietet, wie das Erstellen und Widerrufen von Credentials, Erstellen von Schemas, etc.
Auch wäre es in der implementierten Lösung bedenkenswert den Issuer oder Verifier nicht off-chain sondern on-chain zu implementieren. Hierbei befände sich die Logik im Smartcontract und gibt die Möglichkeit Operationen direkt in der Blockchain durchzuführen (Token-Transfer, Verwendung anderer Smart-Contracts, bessere Nachverfolgbarkeit, etc.). Auch wäre es denkbar einen eigenen Identity-Wallet zu implementieren und zu verwenden. In der hier vorgestellten Lösung wurde die PolygonId-App verwendet, jedoch stehen mehrere SDK zur Verfügung, womit ein Entwickler seinen eigenen Wallet erstellen kann.
Zusätzlich gibt es das Konzept der Non-Fungible-Token, die mit dem Proof-Of-Ownership-Gedanken dem SSI-Konzept ähneln. Daher wäre es abwägbar, ob eine Integration von NFT's in eine weiterführende Arbeit von Bedeutung wäre.

7.3 Zukünftige Forschungsrichtungen

Der Blick in die Zukunft des Forschungsfelds im Bereich SSI und deren Implementierung mithilfe von Distributed Ledger Technology (DLT) verspricht faszinierende Ent-

wicklungen. In den kommenden Jahren werden Forscher voraussichtlich verstärkt die Skalierbarkeit von SSI-Systemen erforschen, um diese für breitere Anwendungsbereiche tauglich zu machen. Ein Schwerpunkt könnte dabei auf der Integration von SSI in bestehende digitale Infrastrukturen und Plattformen liegen, um die nahtlose Interoperabilität zu gewährleisten.

Des Weiteren wird die Verbesserung der Sicherheitsaspekte von SSI von entscheidender Bedeutung sein. Forschung wird sich auf fortschrittliche Kryptographie, Authentifizierungsmethoden und Datenschutzkonzepte konzentrieren, um das Vertrauen in diese Systeme zu stärken und Datenschutzverletzungen zu minimieren.

Die Entwicklung von internationalen Standards und Protokollen für SSI wird eine weitere wichtige Forschungsrichtung sein. Diese Standards sind entscheidend, um die weltweite Akzeptanz und Anwendung von SSI-Systemen zu fördern und Interoperabilität zwischen verschiedenen Implementierungen sicherzustellen.

Schließlich werden auch soziale und ethische Aspekte der Selbstsouveränen Identitäten verstärkt in den Fokus rücken. Forschung wird sich auf Fragen der Akzeptanz, der Bildung und Sensibilisierung der Nutzer, sowie auf die ethischen Implikationen in Bezug auf Identitätsmanagement und Datenschutz konzentrieren.

Insgesamt versprechen zukünftige Forschungsrichtungen auf dem Gebiet der Selbstsouveränen Identitäten und DLT eine spannende und vielversprechende Entwicklung, die sowohl technische als auch gesellschaftliche Herausforderungen angehen wird, um die Vision von selbstsouveränen und sicheren Identitäten in einer digitalisierten Welt voranzubringen.

7.4 Beitrag zur Forschung

Der Beitrag zur Forschung beinhaltet unter anderem, dass eine vertiefte Analyse der Problematik herkömmlicher Identitätsmanagementsysteme stattfindet. Auch werden die technischen Grundlagen und Implementierung von SSI aufbereitet und detailliert an den Leser gebracht. Zudem wird diskutiert, welche alternativen Implementierungsoptionen es gibt und welche Felder weiterhin zu erforschen sind. Zudem wurde durch die Implementierung der Beweis erbracht, dass die theoretischen Konzepte in der Praxis Anwendung finden.

Literatur

[Id2a] "Blockchain Autumn School 2020". In: (2022). URL: https://esatus.com/wp-content/uploads/Actionlog_BAS_SSI_in_der_praktischen_Nutzung_20201009.pdf (besucht am 23. 08. 2023).

[Id2b] "Digital Identity Management". In: (2011). URL: https://www.oecd.org/sti/ieconomy/49338380.pdf (besucht am 23. 07. 2023).

[Id3f] "Use Ethereum". In: (2020). URL: https://ethereum.org/en/developers/docs/scaling/plasma/ (besucht am 26. 08. 2023).

[Id4d] "Sovrin". In: (27. Aug. 2023). URL: https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf.

[Ish20] Georgy Ishmaev. "Sovereignty, privacy, and ethics in blockchain-based identity management systems". In: *Ethics and Information Technology* (30. Nov. 2020), 239–252.

[Jø+05] Audun Jøsang u. a. "Trust Requirements in Identity Management". In: (1. Jan. 2005).

[LSP82] Leslie Lamport, Robert Shostak und Marshall Pease. "The byzantine generals problem". In: *ACM Trans. Program.Lang. Syst. 4, 3* (1982).

[Min+17] Du Mingxiao u. a. "A Review on Consensus Algorithm of Blockchain". In: (5. Okt. 2017), S. 152–189.

[MS15] Alan Mitchell und Jamie Smith. "Economics of Identity". In: (2015).

[Sun19] Ali Sunyaev. "Distributed ledger technology. In Internet Computing: Principles of Distributed Systems and Emerg- ing Internet-based Technologies". In: (2019).

[TR17] Andrew Tobin und Drummond Reed. "The Inevitable Rise of Self-Sovereign Identity". In: (2017), S. 1–23.

Online-Quellen

[Id1a] *Decentralized Identifiers (DIDs) v1.0*.

[Id1b] *Die Chain Key Technologie: Schlüssel des Internet Computers*. URL: https://internet-computer.de/wissen/chain-key-technologie-kryptographie/ (besucht am 23. 11. 2023).

[Id1c] *Introducing BrowserID – easier and safer authentication on the web*. 21. Juli 2011. URL: https://hacks.mozilla.org/2011/07/introducing-browserid-easier-and-safer-authentication-on-the-web/.

[Id1d] *Introduction to Self-Sovereign Identity (SSI)*. 2015. URL: https://walt.id/white-paper/self-sovereign-identity-ssi (besucht am 23. 11. 2023).

[Id1e] *OpenID*.

[Id2c] *Ethereum*. 2015. URL: https://ethereum.org/de/developers/docs/scaling/sidechains/ (besucht am 23. 08. 2023).

[Id2d] *Identity Management System Requirements*. 2015. URL: https://ebrary.net/24577/computer_science/identity_management_system_requirements#gads_btm (besucht am 23. 07. 2023).

[Id2e] *Luniverse*. 20. Aug. 2023. URL: https://luniverse.io.

[Id3a] *Dock*. 2020. URL: https://www.dock.io/feature/blockchain (besucht am 23. 08. 2023).

[Id3b] *Polygon*. 2020. URL: https://polygon.technology/polygon-id (besucht am 24. 08. 2023).

[Id3c] *Polygon*. 2020. URL: https://polygon.technology/polygon-pos (besucht am 26. 08. 2023).

[Id3d] *Polygon ID Documentation tutorials*. 2017. URL: https://0xpolygonid.github.io/tutorials/issuer-node/issuer-node-overview/ (besucht am 26. 08. 2023).

[Id3e] *Sovrin*. 2016. URL: https://sovrin.org/ (besucht am 27. 08. 2023).

[Id3g] *Use Ethereum*. 2020. URL: https://ethereum.org/de/developers/docs/scaling/optimistic-rollups/ (besucht am 26. 08. 2023).

[Id4a] *Dock Blog — How Dock Compares to Blockcerts*. 2023. URL: https://blog.dock.io/how-dock-compares-to-blockcerts/ (besucht am 02. 09. 2023).

[Id4b] *Polygonscan Support Center*. 2023. URL: https://support.polygonscan.com/support/solutions/articles/69000793833-what-is-gas-fee-%E2%9B%BD (besucht am 02. 09. 2023).

[Id4c] *Sovrin*. 2019. URL: https://sovrin.org/faqs/ (besucht am 01. 09. 2023).

[Id4e] *Statista*. 27. Aug. 2023. URL: https://www.statista.com/statistics/264473/number-of-internet-hosts-in-the-domain-name-system/.

[Id4f] *YCharts*. 2005. URL: https://ycharts.com/indicators/bitcoin_average_transaction_fee (besucht am 05. 09. 2023).

[Id5a] *Bitpanda*. 2020. URL: https://www.bitpanda.com/academy/de/lektionen/konsens-algorithmen-proof-of-work/#:~:text=Proof%20of%20Work%20ist%20der%20Konsens%2DAlgorithmus%2C%20der%20der%20Bitcoin,Regel%20durch%20Rechenleistung%20%2D%20verrichten%20m%C3%BCssen. (besucht am 30. 09. 2023).

[Id5b] *BTC-Echo*. 24. Sep. 2023.

[Id5c] *Coinmerce*. 2020. URL: https://coinmerce.io/de/lernen/was-ist-proof-of-authority/ (besucht am 24. 09. 2023).

[Id5d] *Luniverse Nova*. 24. Sep. 2023.

[Id5e] *Microsoft Threat Modeling Tool-Bedrohungen*. 2016. URL: https://learn.microsoft.com/de-de/azure/security/develop/threat-modeling-tool-threats (besucht am 29. 10. 2023).

[Id5f] *Polygon ID Documentation tutorials*. 2020. URL: https://0xpolygonid.github.io/tutorials/verifier/query-builder/#:~:text=In%20Polygon%20ID%2C%20this%20useful,being%20at%20a%20certain%20age. (besucht am 27. 10. 2023).

[Id5g] *Polygon ID Documentation tutorials*. 28. Okt. 2023. URL: https://0xpolygonid.github.io/tutorials/wallet/wallet-sdk/polygonid-app/.

[Id6a] *Adnovum*. 2015. URL: https://www.adnovum.com/blog/self-sovereign-identity-ssi-switzerland (besucht am 03. 11. 2023).

[Id6b] *BSI*. 2022. URL: https://www.bsi.bund.de/EN/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehl
Accountschutz/Sichere-Passwoerter-erstellen/sichere-passwoerter-erstellen_node.html (besucht am 29. 10. 2023).

[Id6c] *Cost per leaked record in data breaches worldwide from 2014 to 2023*. 2023. URL: https://www.statista.com/statistics/799396/worldwide-cost-effects-data-record-breaches/#:~:text=As%20of%202023%2C%20the%20cost,was%204.45%20million%20U.S.%20dollars. (besucht am 30. 11. 2023).

[Id6d] *Security at Alchemy*. 2020. URL: https://www.alchemy.com/security (besucht am 29. 10. 2023).

[Id6e] *The Missing Layer of the Internet: (Self-Sovereign) Identity*. 2023. URL: https://medium.com/@umutyorulmaz11/the-missing-layer-of-the-internet-self-sovereign-identity-1f26b201c024 (besucht am 09. 11. 2023).

[KAN+00] NICLAS KANNENGIEßER u. a. *Trade-offs between Distributed Ledger Technology Characteristics*. 2000. URL: https://www.w3.org/TR/did-core/ (besucht am 01. 05. 2023).

[Loc+] Hal Lockhart u. a. *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. URL: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html.

[Sta12] Statista. *Why Do Shoppers Drop Out of an Online Purchase*. 2012. URL: https://www.statista.com/topics/7868/online-checkout-behavior-and-e-commerce-conversions-worldwide/#topicOverview.