

Bachelor Thesis

Unlinkability of Verifiable Credentials in a practical approach

June 5, 2024

Joel Robles | TI

Inhaltsverzeichnis

- ▶ Ziel
- ▶ Self-sovereign Identity
- ▶ Verifiable Credentials
- ▶ Verifiable Presentations
- ▶ Sicherheitsüberlegungen von VC/VPs
- ▶ OpenID Connect for Verifiable Presentations
- ▶ Sicherheitsüberlegungen von OIDC4VP
- ▶ Fazit
- ▶ Ausblick

Ziel

Was ist das Ziel?

Die Analyse, ob eine echt-welt implementation von Verifiable Credentials mit dem BBS Signature Scheme, unverknüpfbarkeit beibehält

Self-sovereign Identity

Self-sovereign Identity (SSI)

- Ist ein konzept wo eine Person (**Holder**) entscheiden kann, wer was über sie wissen darf
- Holders dürfen wählen was sie offenbaren und was nicht, auch bekannt als **selective disclosure**
- Erstes Problem:
 - Holder zeigt eine Staatliche ID
 - Ist eine Menge von Daten oder eine Menge von **attributes**
 - Die person welche verifiziert sieht alle attribute
- Zweites Problem:
 - Holder zeigt attribute einer person die diese verifizieren will, bekannt als **verifier**
 - Holder zeigt die gleichen attribute einem zweiten **verifier**
 - Der kann ge-**linked** werden
- Heutiger stand - Holder haben keine kontrolle über ihre Attribute
- Zukünftiger stand dank SSI - Holder haben volle kontrolle über ihre Attribute

Trust Triangle

- Wie weiss ein verifier das eine Menge von Attributen (**credential**) valid ist?
- Er vertraut dem issuer!
- Beispiel: Schweizer ID hat hologramme

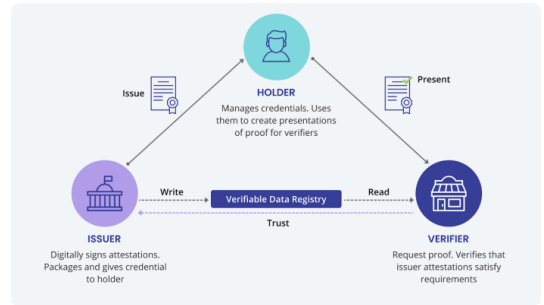


Abbildung: Trust triangle

Verifiable Credentials

Verifiable Credentials (VC)

- Verifiable Credentials sind eine digitale repräsentation von physischen Credentials
- JSON-LD repräsentiert attribute als **key-value pairs**
- Beispiel:
 - ▣ Vorname auf einer ID
 - ▣ Repräsentiert als {first_name: "John"}
 - ▣ first_name ist der key und "John" ist der value

```
1  {
2    "@context": [
3      "https://www.w3.org/ns/credentials/v2",
4      "https://w3id.org/security/data-integrity/v2",
5      "https://raw.githubusercontent.com/robl...
6    ],
7    "type": ["VerifiableCredential"],
8    "credentialSubject": {
9      "first_name": "John",
10     "last_name": "Doe",
11     "birth_date": "1.1.1970"
12   }
13 }
```

Abbildung: Beispiel VC


VCs and BBS

- Warum werden sie **Verifiable** Credentials genannt?
- Der verifier kann ein VC, welches ihm präsentiert wurde (**Verifiable Presentation**), verifizieren, wegen Kryptographischen Signaturen
- Diese zeigen, dass das credential seit der ausstellung nicht verändert wurde
- Wir nutzen das BBS Signature Scheme (**BBS**)
- Diese Schema bietet **selective disclosure** and **unlinkability**
- Aber wie unlinkability? - Der Verifierbraucht die Signatur
- BBS kann **proofs** generieren
- Diese beweisen das der Holder die Signatur kennt, ohne diese zu offenbaren
- Weiter sind die proof unlinkable zwischen jeder generierung

Verifiable Presentations

Verifiable Presentation (VP)

- Ein holder würde gerne ein VC präsentieren
- Dafür werden **Verifiable Presentations** genutzt
- BBS kann nur staments signieren
- Der **RDF** canonicalization Algorithmus, welcher staments aus key-value pairs generiert



```
1 _:b0 <https://schema.org/birthDate> \"1.1.1970\" .\n2 _:b0 <https://schema.org/familyName> \"Doe\" .\n3 _:b0 <https://schema.org/givenName> \"John\" .\n4 _:b1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://www.w3.org/2018credentials#VerifiableCredential> .\n5 _:b1 <https://www.w3.org/2018/credentials#credentialSubject> _:b0 .\n
```

Abbildung: Beispiel canonicalized VP

Der RDF Algorithmus

aus einer unsortierten geschachtelten json struktur werden staments gemacht
staments sortiert basierend auf ihren hash wert deterministische sortierte folge von
staments (kanonisch)

1. Erzeuge eine Map zwischen den blank node identifiers (z.B. eo) und den attributen
2. Kreiere ein hash der blank node identifiers
3. Kreiere eine Map zwischen den hashes und den blank node identifiers
4. Sortiere die Map von klein zu gross basierend auf den hashes
5. Kreiere canonical identifiers (z.B. c14no) für jede blank node identifier, basierend auf der position in der hash-identifier map

Sicherheitsüberlegungen von VC/VPs

Permutation von statements

- Holder präsentiert ein VP mit verborgenen Zivilstands-Attributen
- Holder heiratet bekommt ein neues VP mit geändertem Zivilstand
- Holder präsentiert das aktualisierte VP mit verborgenem Zivilstand
- **Datenleck:** Der Verifier kann herausfinden, dass sich der Zivilstand geändert hat
- Damit das passieren kann, muss der issuer immer die statements zufällig permutieren
- Der issuer muss die Permutation dem holder bekannt geben, nicht dem verifier

Verknüpfbarkeit von Identifikatoren & Metadaten

Test

OpenID Connect for Verifiable Presentations

Transport zwischen holder und verifier

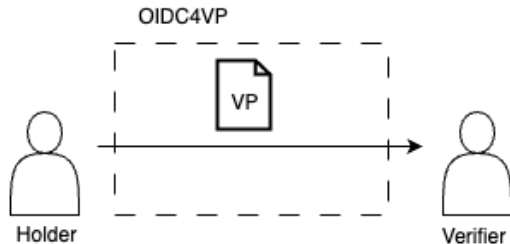


Abbildung: OpenID connect for Verifiable Presentations

OIDC4VP Fluss

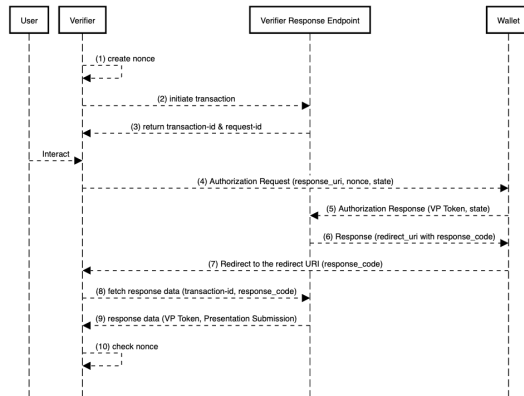


Abbildung: OpenID connect for Verifiable Presentations Fluss

Sicherheitsüberlegungen von OIDC₄VP

Replay attack

- Der Holder sendet dem Verifier ein VP
- Ein *Man in the Middle* speichert die Vorstellung
- Der *Man in the Middle* kann das gespeicherte VP wiederverwenden
- Um dieses Problem zu umgehen, wird eine random nummer genutzt (challenge-response)

Fazit

Fazit

Ausblick

Ausblick

- Link Secrets und Blind BBS Signatures für linkability und selective disclosure analysieren
- Implementieren und testen