Decentralised Identifiers and KILT

Antonio Antonino

Substrate Seminar - 01/06/2021

project supported by
web3 foundation
grants program

@KILTprotocol
antonio@kilt.io

#Substrate #DID

§ Legal Note

1. The **purpose of this technical presentation is to present the KILT Protocol and views about the current and future technical infrastructure around it to an interested public**. The information set forth should not be considered exhaustive and does not imply any elements of a contractual relationship. Its sole purposes are to provide relevant and reasonable information to an interested public as well as to any developers who think about integrating the KILT Protocol, to any blockchain developers who wish to contribute to the KILT Community and to potential implementation partners who want to get an insight into the current state of the KILT project.

2. **Nothing in this presentation shall be deemed to constitute a prospectus of any sort or a solicitation for investment**, nor does it, in any way, pertain to an offering or a solicitation of an offer to buy any securities in any jurisdiction. The document is not composed in accordance with, and is not subject to, laws or regulations of any jurisdiction which are designed to protect investors.

3. This **presentation shows today's views and today's vision of the project which are both subject to constant change**. Certain statements, estimates, and financial information contained within this presentation constitute forward-looking, or pro-forma statements, and information. Such statements or information involve estimates and opinions on potential future developments as well as known and unknown risks and uncertainties which may cause actual events or results to differ materially from the estimates or the results implied or expressed in such forward-looking statements, even if such statements are not specially marked as unknown or uncertain by an explicit remark or by the grammar or tense used.

4. **Nothing published by the BOTLabs GmbH should be interpreted as investment advice**. BOTLabs GmbH is in no way providing trading or investment advice. Please consult with your appropriate licensed professional before making any financial transactions, including any investments related to ideas or opinions expressed, past, present, or future by the aforementioned entities and any future entities that may operate under the parent entities. BOTLabs GmbH does not intend to express financial, legal, tax, or any other advice and any conclusions drawn from statements made by, or on, BOTLabs GmbH shall not be deemed to constitute advice in any jurisdiction. Information is provided for educational and amusement purposes only.

Imprint

BOTLabs GmbH
Keithstr. 2-4
10787 Berlin

Germany Commercial Court:
AG Charlottenburg
HRB 193450 B
USt-IdNr.: DE316284270

info@botlabs.org
https://botlabs.org

Managing Director:
Ingo Rübe
[Requirements according to § 5 TMG (Germany)]

Agenda

- What are Decentralised Identifiers (DID)
- Verifiable Credentials and DIDs
- DIDs in KILT
- Demo

What are Decentralised Identifiers?

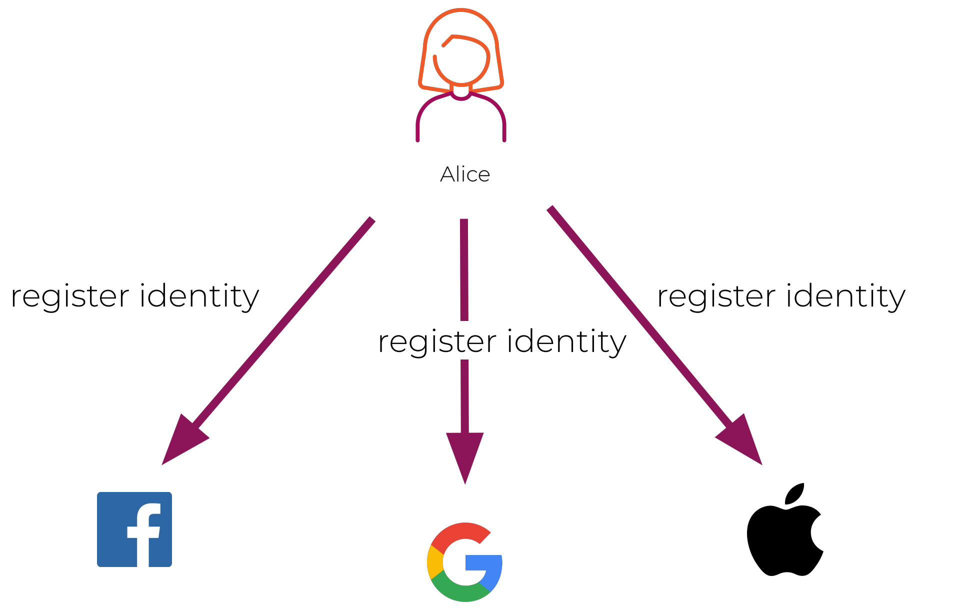
Decentralised Identifiers (DID) are:

*"a new type of identifier that enables **verifiable**, **decentralised** digital identity."**

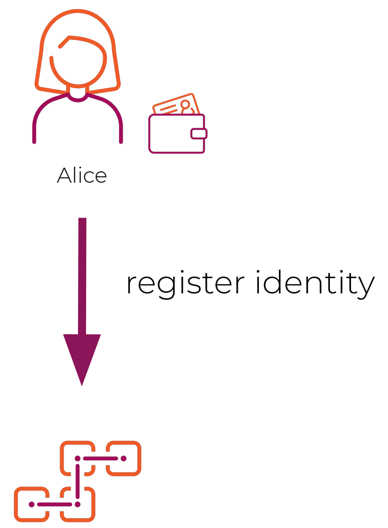
- **Verifiable**: the identity subject can use various cryptographic techniques to prove ownership of the identity
- **Decentralised**: identity management (creation, management, resolution) does not depend on any centralised registry/controller

**W3C official specification v1.0:* [*https://w3c.github.io/did-core/*](https://w3c.github.io/did-core/)

Centralised vs. Decentralised Identifiers (visual representation)



- **Full control by identity providers** over Alice's information and **full visibility** over when and where the information is used
- **Reliance on third-parties** for identity resolution
- **Very expensive** to create multiple identities (e.g., a different Google address for each website) -> **activity tracking** over long periods of time



- **Single source** of truth for Alice's identity
- **Full control by Alice** over her own information updates
- Often **inexpensive** to create multiple identities -> **reduced** chances of **tracking** interactions of a given subject

Structure of a DID

did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw

Structure of a DID

did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw

Structure of a DID

did:**kilt**:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw

W3C DID methods registry:
https://w3c.github.io/did-spec-registries/#did-methods

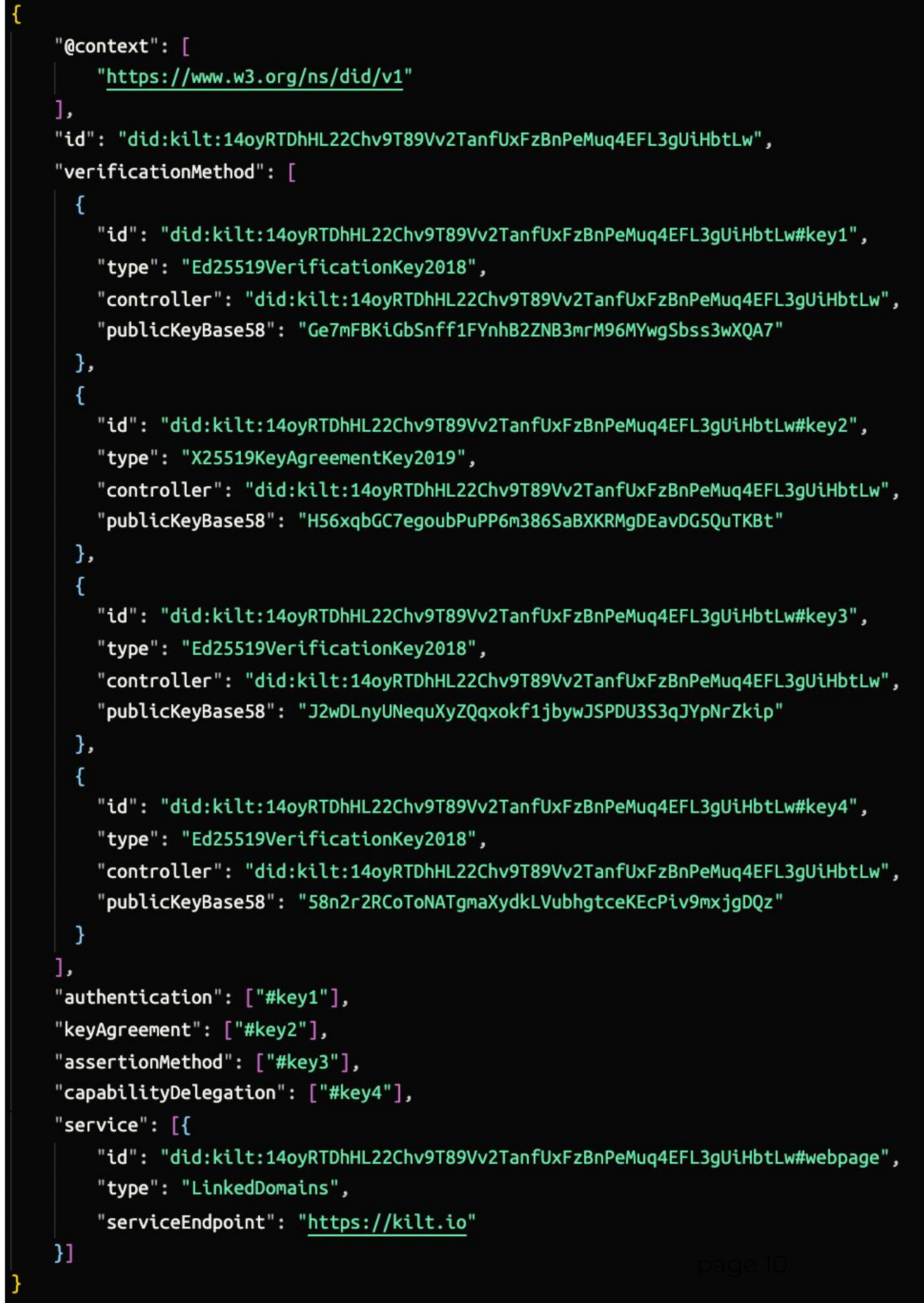
Structure of a DID

did:kilt:**14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw**

DID Resolution

->

did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw

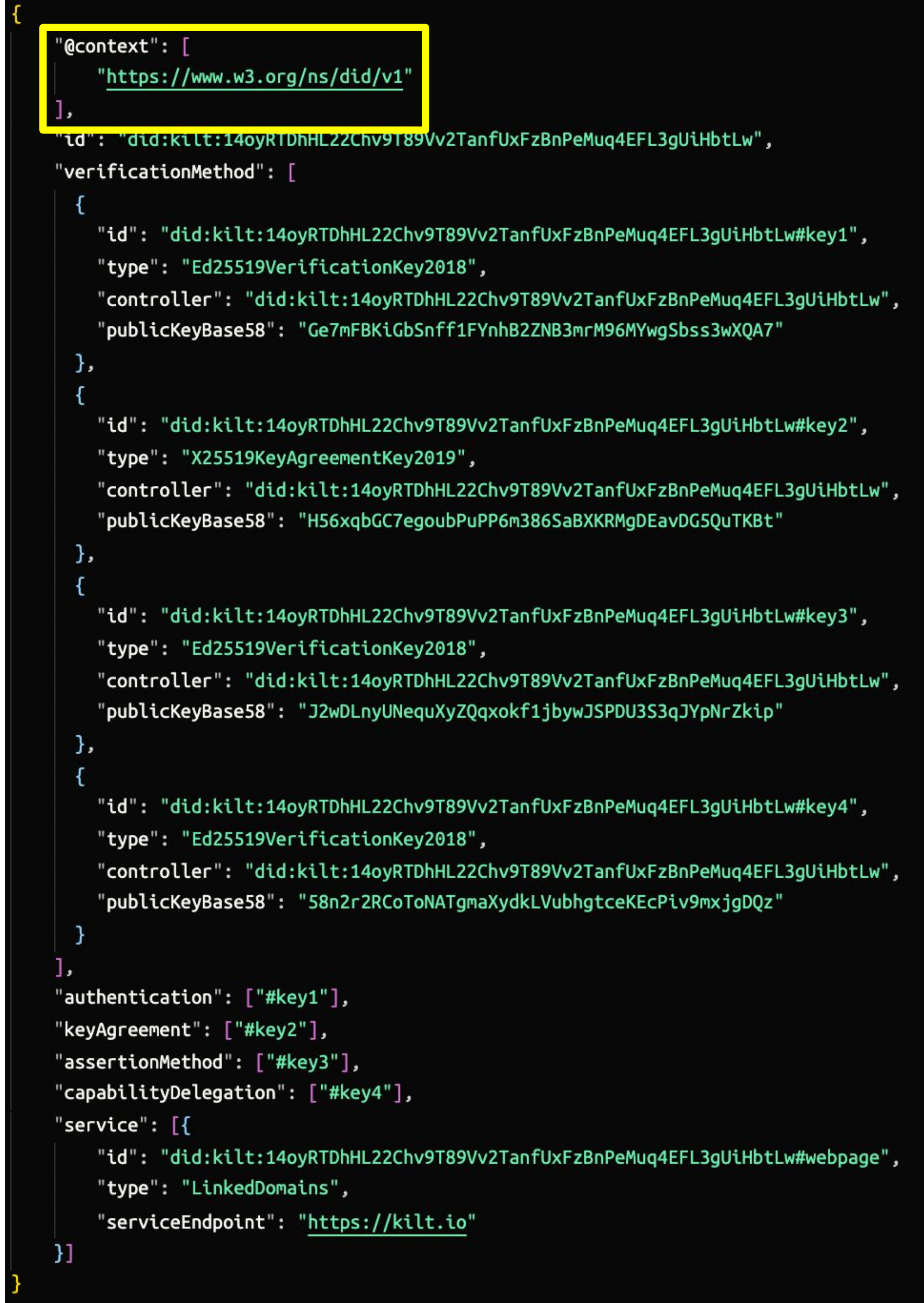


```
{
  "@context": [
      "https://www.w3.org/ns/did/v1"
  ],
  "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
  "verificationMethod": [
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
    }
  ],
  "authentication": ["#key1"],
  "keyAgreement": ["#key2"],
  "assertionMethod": ["#key3"],
  "capabilityDelegation": ["#key4"],
  "service": [{
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
      "type": "LinkedDomains",
      "serviceEndpoint": "https://kilt.io"
  }]
}
```

Universal DID resolver by DIF:
https://resolver.identity.foundation/

DID Document - Context

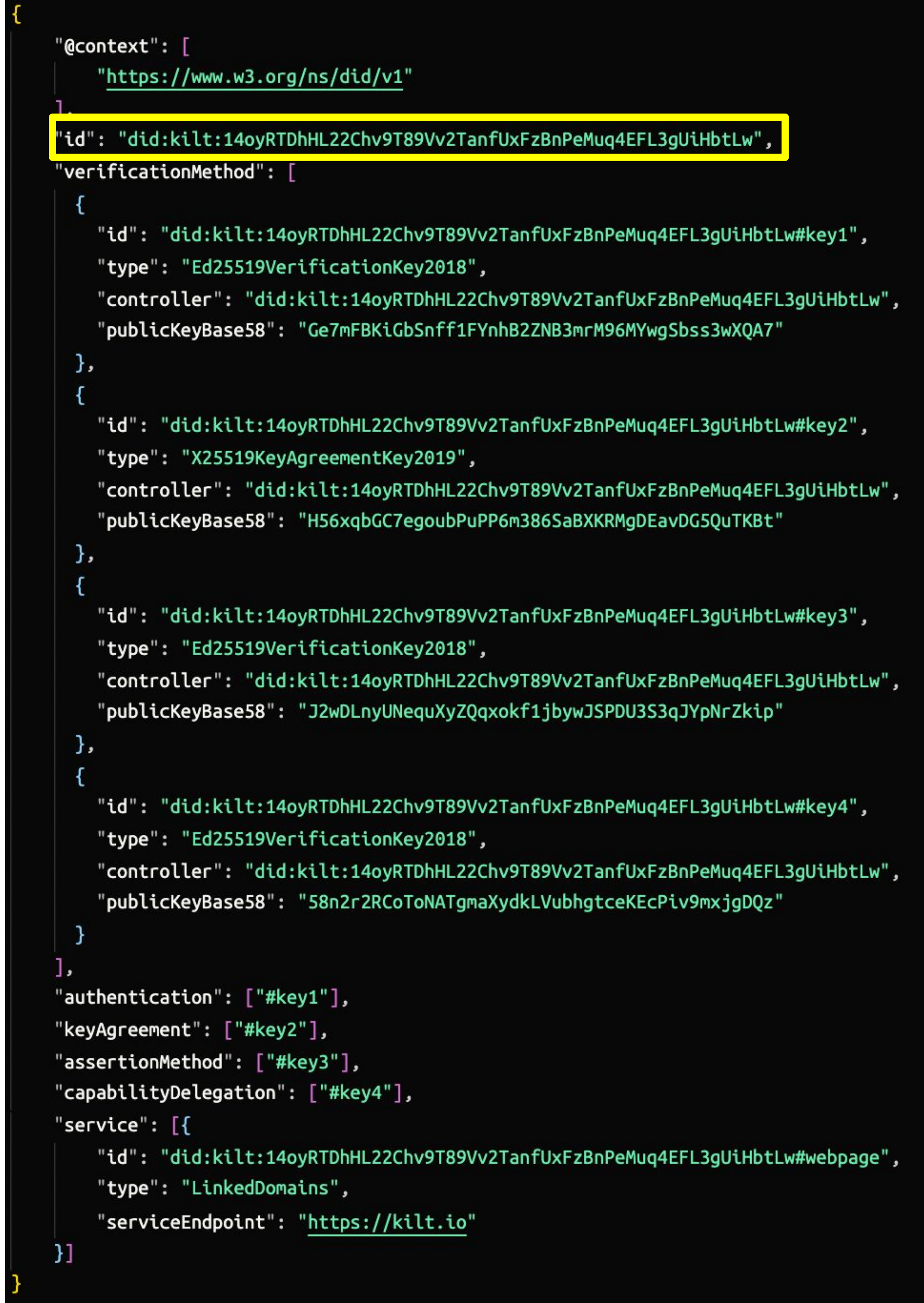
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
  "@context": [
      "https://www.w3.org/ns/did/v1"
  ],
  "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
  "verificationMethod": [
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
    }
  ],
  "authentication": ["#key1"],
  "keyAgreement": ["#key2"],
  "assertionMethod": ["#key3"],
  "capabilityDelegation": ["#key4"],
  "service": [{
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
      "type": "LinkedDomains",
      "serviceEndpoint": "https://kilt.io"
  }]
}
```

DID Document - Identifier

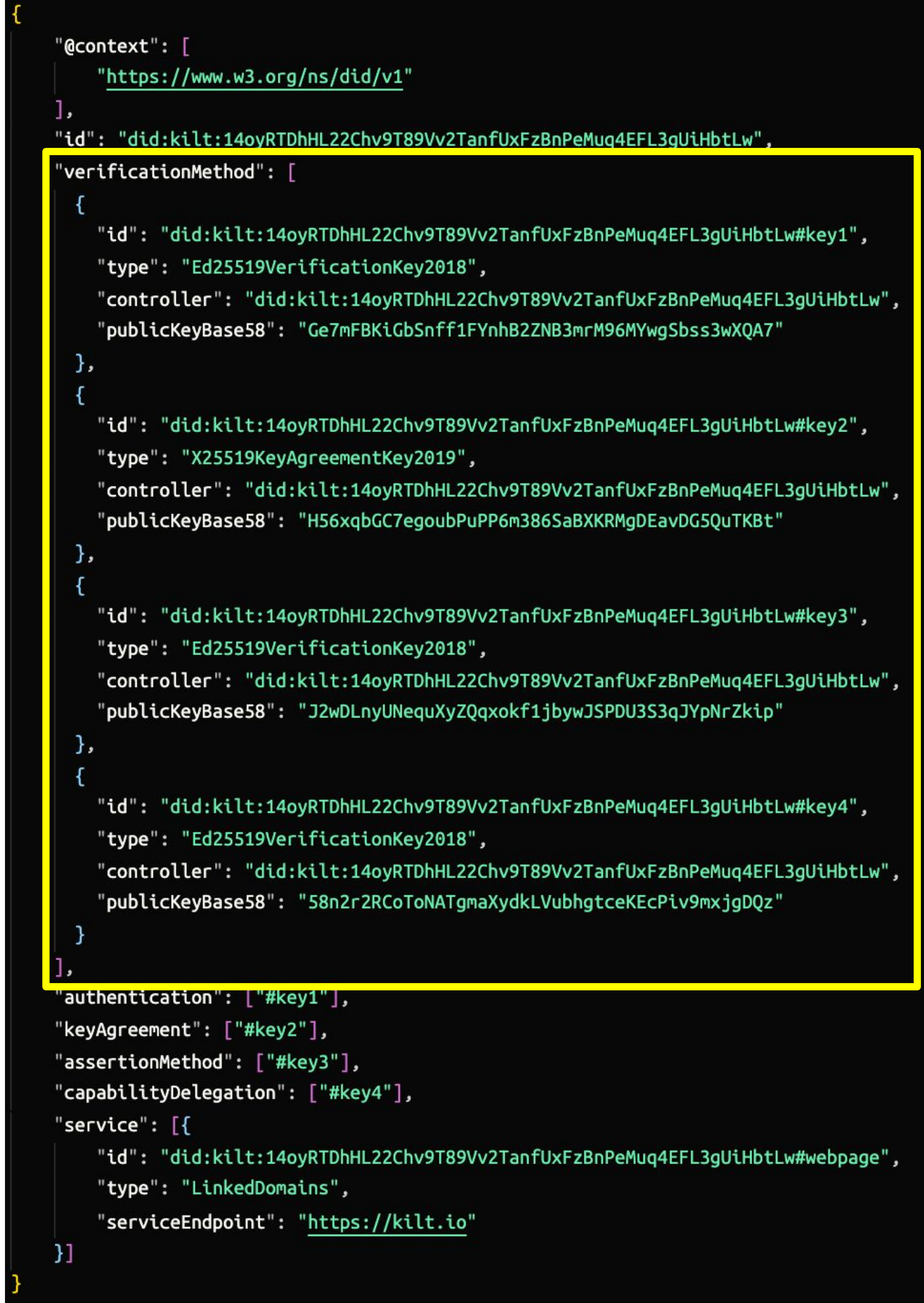
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
  "@context": [
    "https://www.w3.org/ns/did/v1"
  ],
  "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
  "verificationMethod": [
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
    }
  ],
  "authentication": ["#key1"],
  "keyAgreement": ["#key2"],
  "assertionMethod": ["#key3"],
  "capabilityDelegation": ["#key4"],
  "service": [{
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
    "type": "LinkedDomains",
    "serviceEndpoint": "https://kilt.io"
  }]
}
```

DID Document - Verification Methods

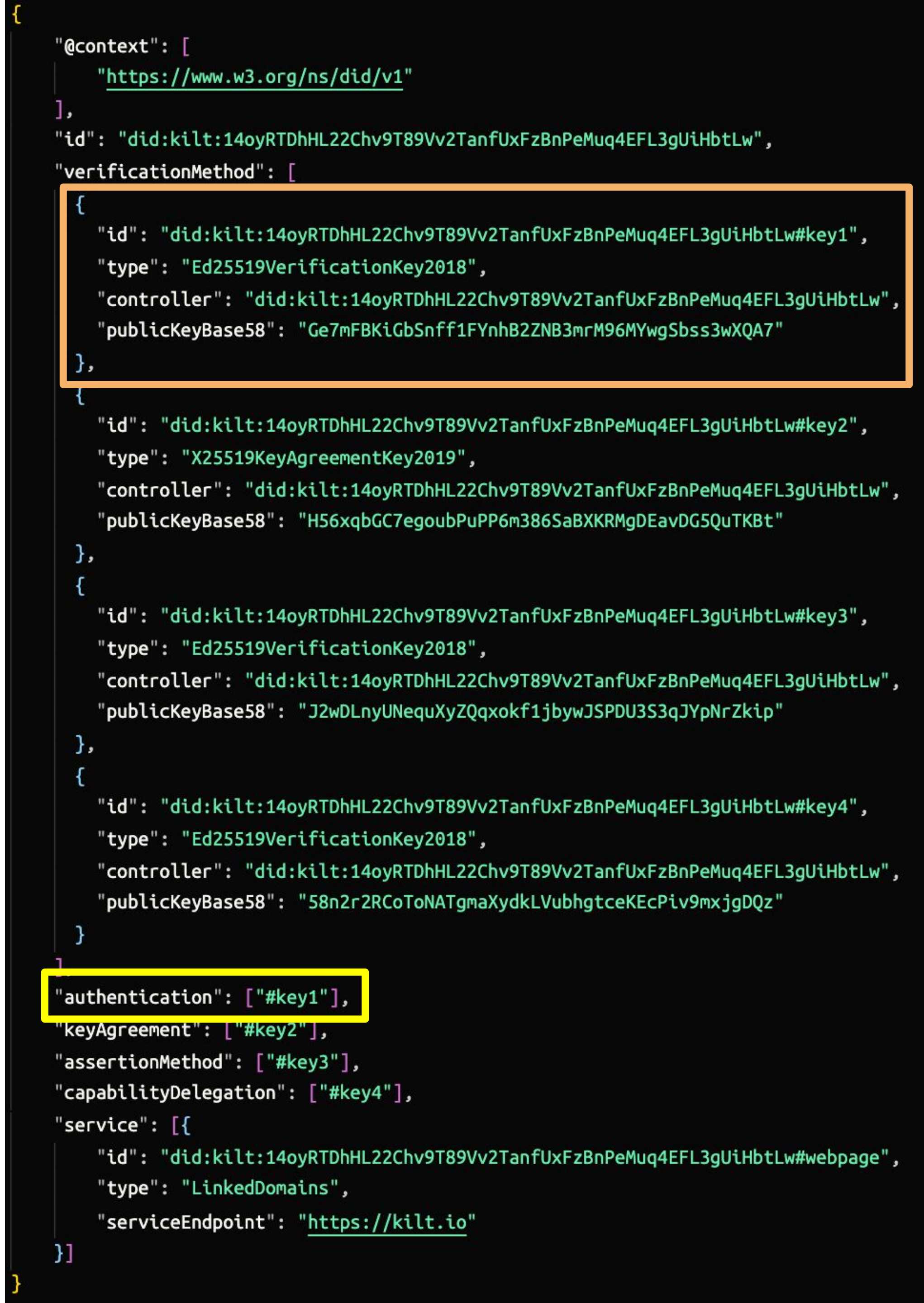
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
  "@context": [
    "https://www.w3.org/ns/did/v1"
  ],
  "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
  "verificationMethod": [
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
    }
  ],
  "authentication": ["#key1"],
  "keyAgreement": ["#key2"],
  "assertionMethod": ["#key3"],
  "capabilityDelegation": ["#key4"],
  "service": [{
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
    "type": "LinkedDomains",
    "serviceEndpoint": "https://kilt.io"
  }]
}
```

DID Document - Authentication

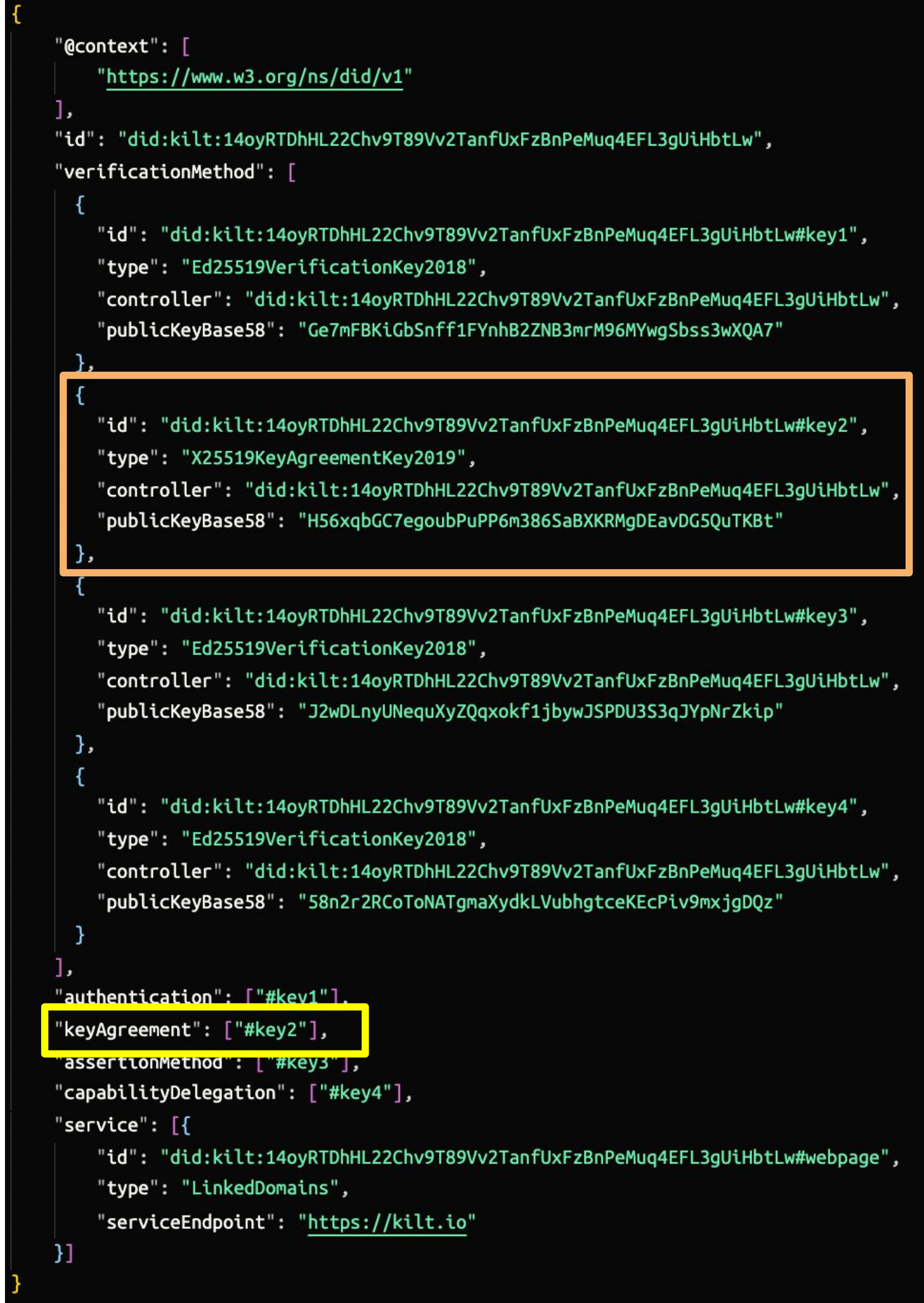
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
  "@context": [
    "https://www.w3.org/ns/did/v1"
  ],
  "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
  "verificationMethod": [
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
    }
  ]
  "authentication": ["#key1"],
  "keyAgreement": ["#key2"],
  "assertionMethod": ["#key3"],
  "capabilityDelegation": ["#key4"],
  "service": [{
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
    "type": "LinkedDomains",
    "serviceEndpoint": "https://kilt.io"
  }]
}
```

DID Document - Key Agreement

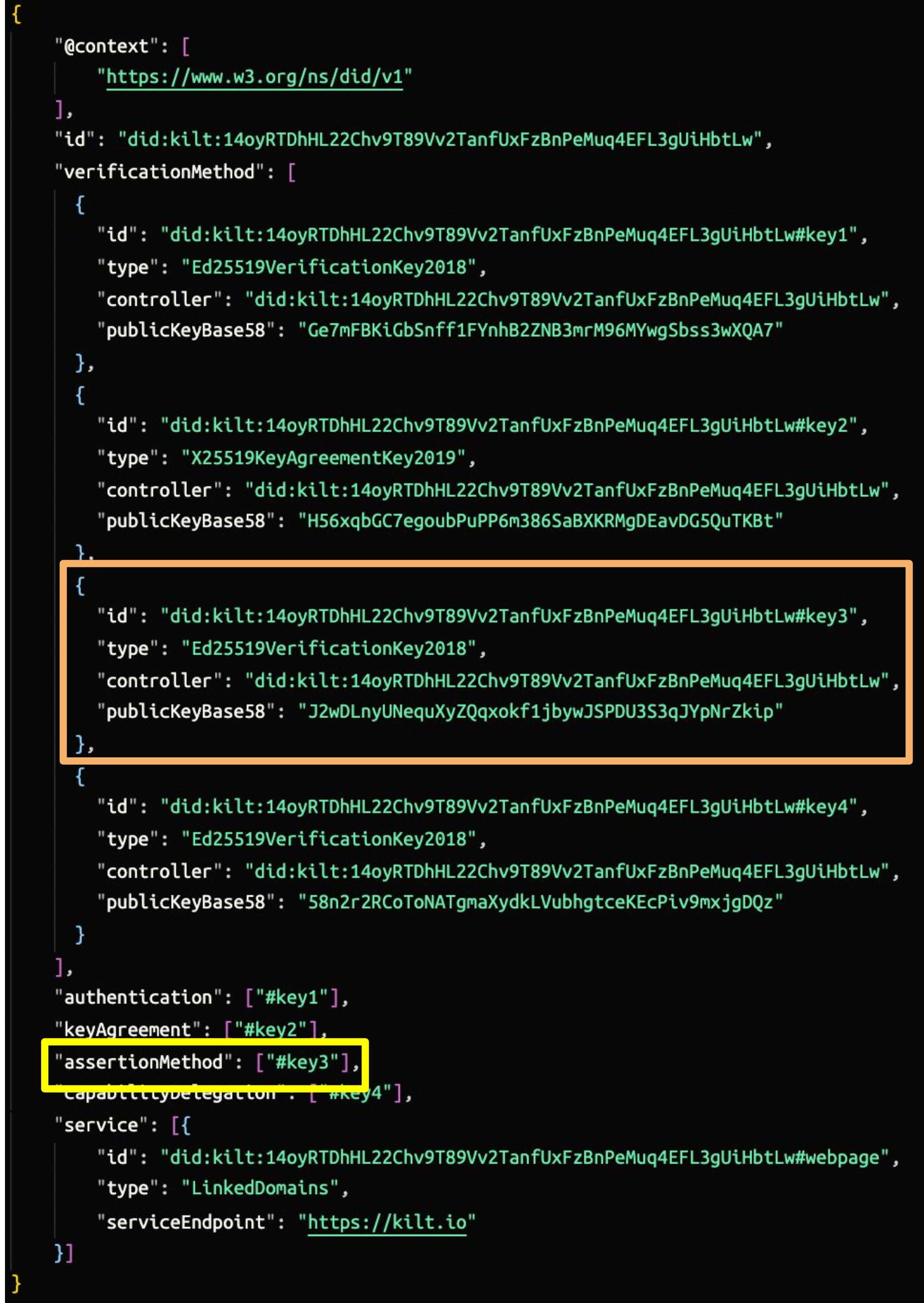
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
    "@context": [
        "https://www.w3.org/ns/did/v1"
    ],
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
    "verificationMethod": [
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
        "type": "Ed25519VerificationKey2018",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
      },
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
        "type": "X25519KeyAgreementKey2019",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
      },
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
        "type": "Ed25519VerificationKey2018",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
      },
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
        "type": "Ed25519VerificationKey2018",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
      }
    ],
    "authentication": ["#key1"],
    "keyAgreement": ["#key2"],
    "assertionMethod": ["#key3"],
    "capabilityDelegation": ["#key4"],
    "service": [{
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
        "type": "LinkedDomains",
        "serviceEndpoint": "https://kilt.io"
    }]
}
```

DID Document - Assertion

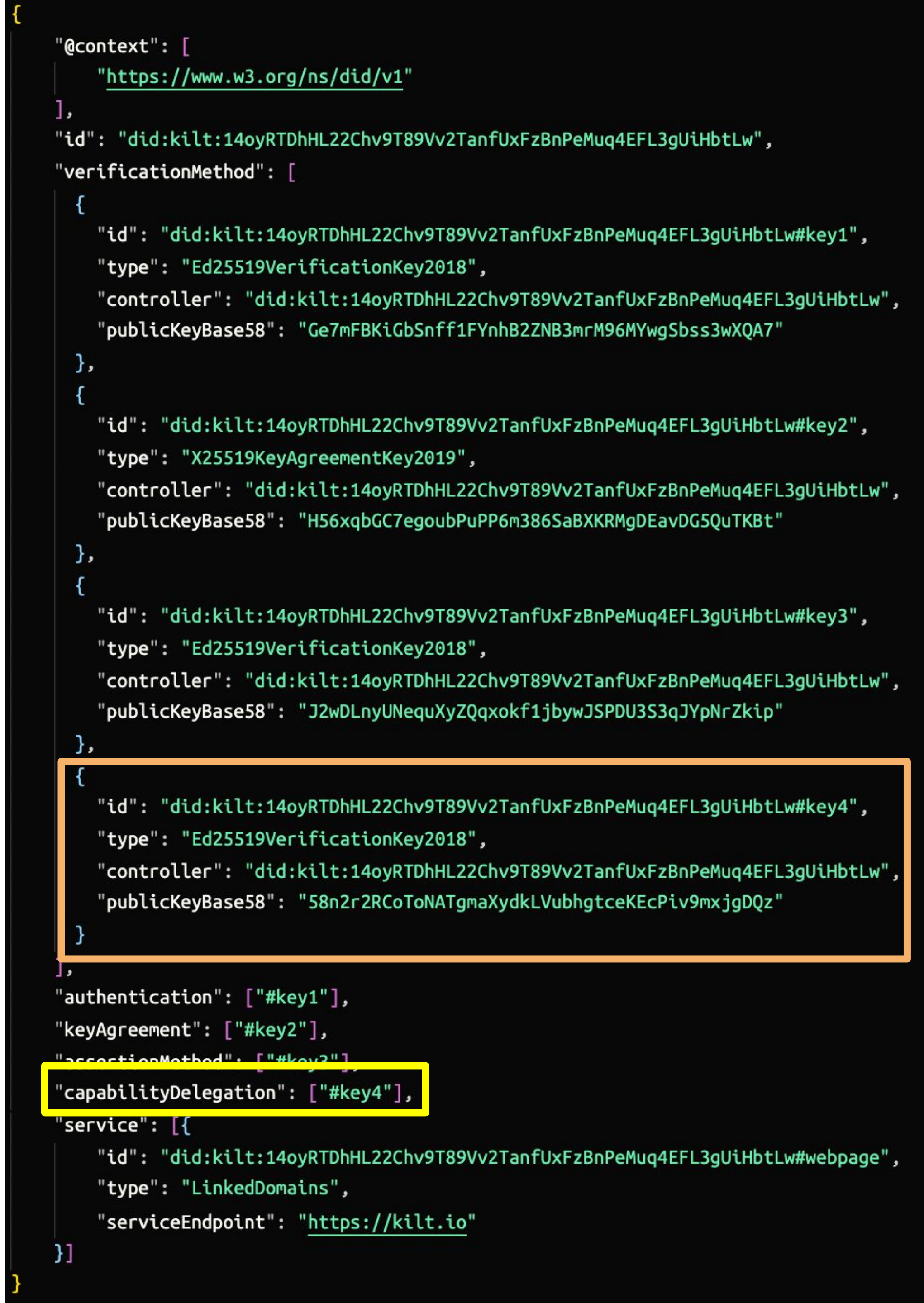
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
    "@context": [
        "https://www.w3.org/ns/did/v1"
    ],
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
    "verificationMethod": [
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
        "type": "Ed25519VerificationKey2018",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
      },
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
        "type": "X25519KeyAgreementKey2019",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
      },
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
        "type": "Ed25519VerificationKey2018",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
      },
      {
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
        "type": "Ed25519VerificationKey2018",
        "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
        "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgteKEcPiv9mxjgDQz"
      }
    ],
    "authentication": ["#key1"],
    "keyAgreement": ["#key2"],
    "assertionMethod": ["#key3"],
    "capabilityDelegation": ["#key4"],
    "service": [{
        "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
        "type": "LinkedDomains",
        "serviceEndpoint": "https://kilt.io"
    }]
}
```

DID Document - Delegation

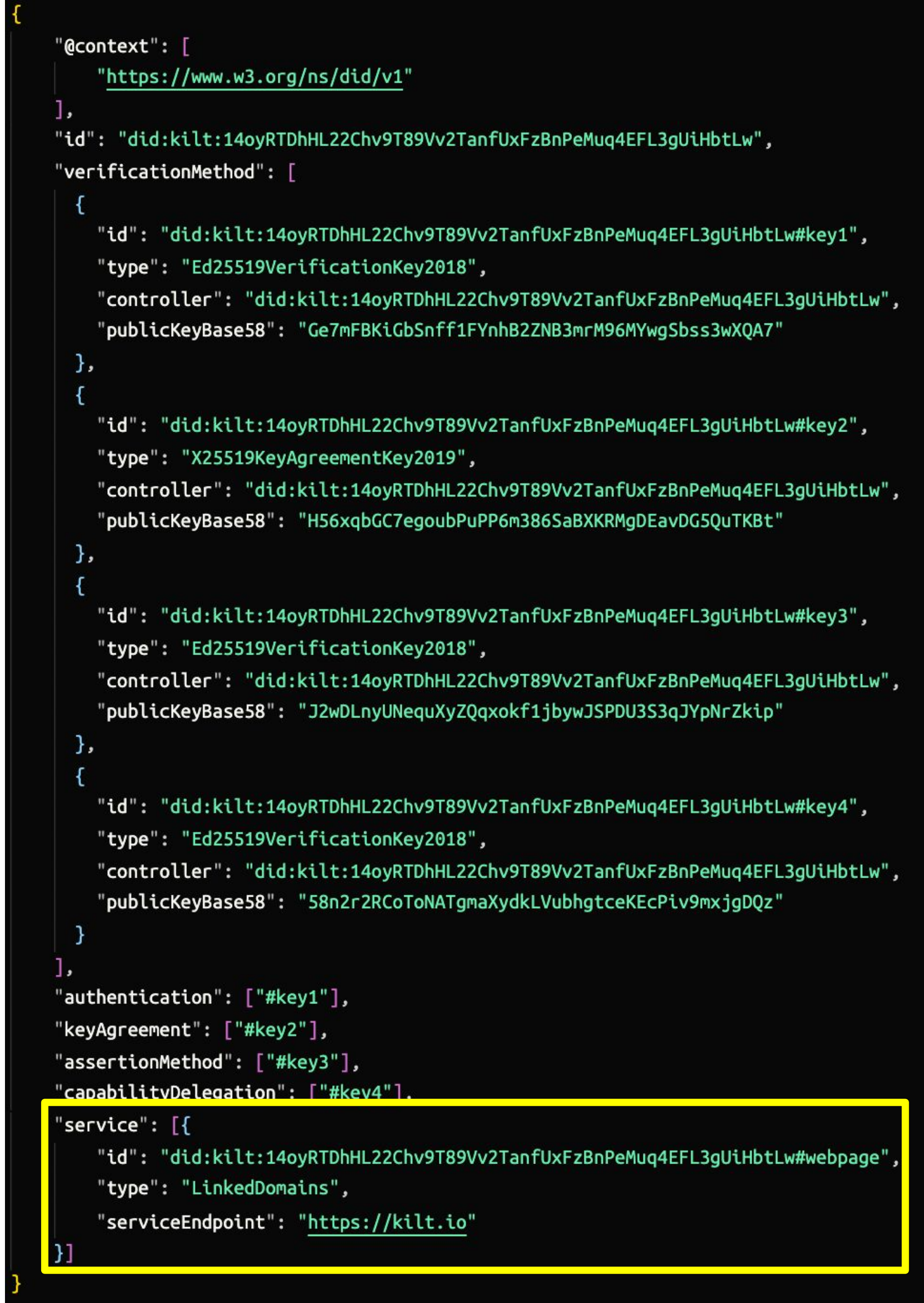
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
  "@context": [
    "https://www.w3.org/ns/did/v1"
  ],
  "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
  "verificationMethod": [
   {
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
    "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
   },
   {
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
    "type": "X25519KeyAgreementKey2019",
    "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
    "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
   },
   {
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
    "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
   },
   {
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
    "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
   }
  ],
  "authentication": ["#key1"],
  "keyAgreement": ["#key2"],
  "assertionMethod": ["#key3"],
  "capabilityDelegation": ["#key4"],
  "service": [{
    "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
    "type": "LinkedDomains",
    "serviceEndpoint": "https://kilt.io"
  }]
}
```

DID Document - Services

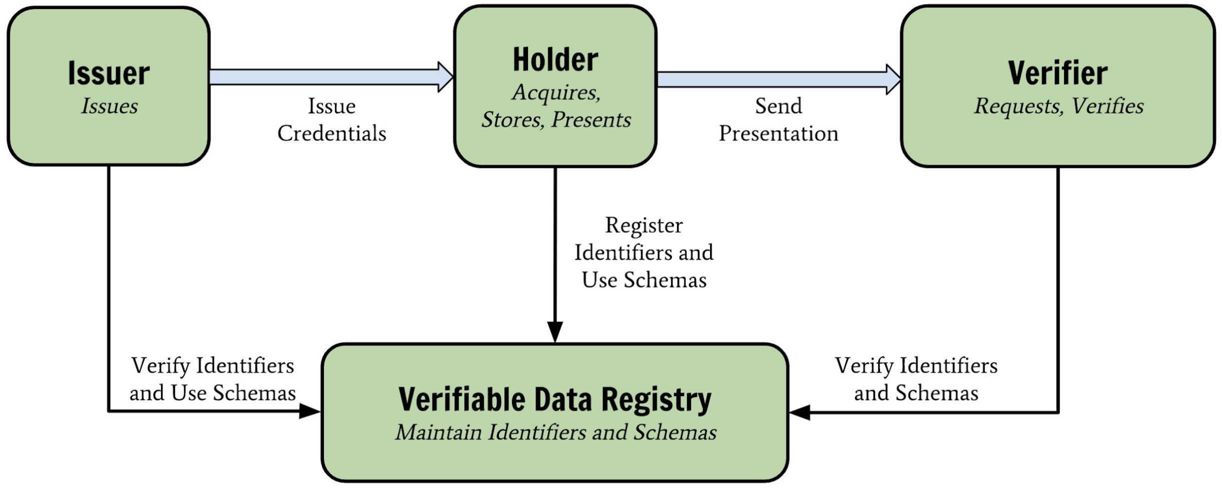
did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw



```
{
  "@context": [
    "https://www.w3.org/ns/did/v1"
  ],
  "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
  "verificationMethod": [
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key1",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "Ge7mFBKiGbSnff1FYnhB2ZNB3mrM96MYwgSbss3wXQA7"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key2",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "H56xqbGC7egoubPuPP6m386SaBXKRMgDEavDG5QuTKBt"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "J2wDLnyUNequXyZQqxokf1jbywJSPDU3S3qJYpNrZkip"
    },
    {
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#key4",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw",
      "publicKeyBase58": "58n2r2RCoToNATgmaXydkLVubhgtceKEcPiv9mxjgDQz"
    }
  ],
  "authentication": ["#key1"],
  "keyAgreement": ["#key2"],
  "assertionMethod": ["#key3"],
  "capabilityDelegation": ["#key4"],
  "service": [{
      "id": "did:kilt:14oyRTDhHL22Chv9T89Vv2TanfUxFzBnPeMuq4EFL3gUiHbtLw#webpage",
      "type": "LinkedDomains",
      "serviceEndpoint": "https://kilt.io"
  }]
}
```

Verifiable Credentials and DIDs

A Verifiable Credential (VC) is a **set of claims** that an **issuer** makes about a **subject**, and that a **verifier** can use to validate certain properties about the subject.



Issuer
Issues
Issue Credentials
Holder
Acquires, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verify Identifiers and Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas

Verifiable Credentials and DIDs

Problem

How does the verifier make sure that the credential was issued to the subject presenting it?

Verifiable Credentials and DIDs

Problem

How does the verifier make sure that the credential was issued to the subject presenting it?

Solution

- Using **bearer credentials** -> valid for whomever owns/presents them

Verifiable Credentials and DIDs

Problem

How does the verifier make sure that the credential was issued to the subject presenting it?

Solution

- Using **bearer credentials** -> valid for whomever owns/presents them
- Using **proof of possession** -> the credential owner must prove to be the owner of the identity to which the credential was issued

Verifiable Credentials and DIDs

Problem

How does the verifier make sure that the credential was issued to the subject presenting it?

Solution

- Using **bearer credentials** -> valid for whomever owns/presents them
- Using **proof of possession** -> the credential owner must prove to be the owner of the identity to which the credential was issued -> the identity must be verifiable

Verifiable Credentials and DIDs

Problem

How does the verifier make sure that the credential was issued to the subject presenting it?

Solution

- Using **bearer credentials** -> valid for whomever owns/presents them
- Using **proof of possession** -> the credential owner must prove to be the owner of the identity to which the credential was issued -> the identity must be verifiable -> DIDs fit very well in this context

DIDs in KILT

KILT DID Features

Lightweight DID

Full DID

KILT DID Features

Lightweight DID

- Created and resolved **offline**, with no blockchain interaction required
- Created from a **KILT account address**
- Do not involve DID documents and do not support key rotation
- Start with **did:kilt:0<kilt_account_address>**
- Suitable for **credential holders** and **verifiers**, with no public identity needed

Full DID

KILT DID Features

Lightweight DID

- Created and resolved **offline**, with no blockchain interaction required
- Created from a **KILT account address**
- Do not involve DID documents and do not support key rotation
- Start with **did:kilt:0<kilt_account_address>**
- Suitable for **credential holders** and **verifiers**, with no public identity needed

Full DID

- Written **on chain**, needs the chain in order to be resolved
- Created from **ed25519** or **sr25519 keypairs**, soon also **ecdsa on secp256k1**
- Supports multiple keys and **key rotation**
- Start with **did:kilt:1<kilt_account_address>**
- Suitable for **credential issuers** and publicly recognised entities

Lightweight KILT DIDs

Generation

Key resolution

*possible only with keys that support this class of operation

Lightweight KILT DIDs

Generation

1. **Generate a KILT account** using a seed or an existing supported keypair
2. Create a **KILT DID** from the account in the form *did:kilt:0<kilt_account>*
    - DID authentication key is the account signing key
3. **Derive key agreement key*** from the authentication key using EC scalar multiplication, resulting in a **x25519** key

Key resolution

*possible only with keys that support this class of operation

Lightweight KILT DIDs

Generation

1. **Generate a KILT account** using a seed or an existing supported keypair
2. Create a **KILT DID** from the account in the form *did:kilt:0<kilt_account>*
    - DID authentication key is the account signing key
3. **Derive key agreement key*** from the authentication key using EC scalar multiplication, resulting in a **x25519** key

Key resolution

1. From a DID like *did:kilt:0<kilt_account>* extract the **KILT account**
2. Compute the **public verification key** for signature verification
3. Derive the **public key agreement key** for payload encryption

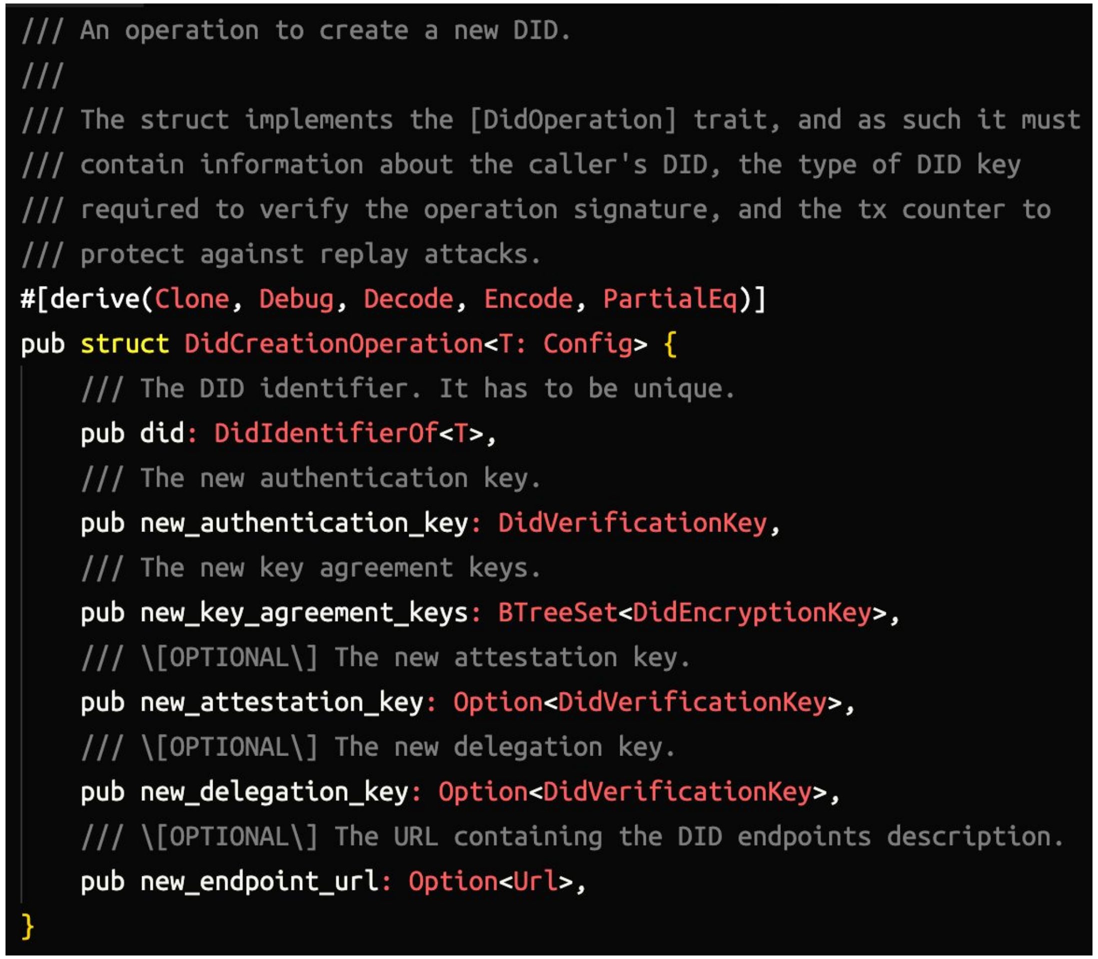
*possible only with keys that support this class of operation

Full KILT DIDs - Generation

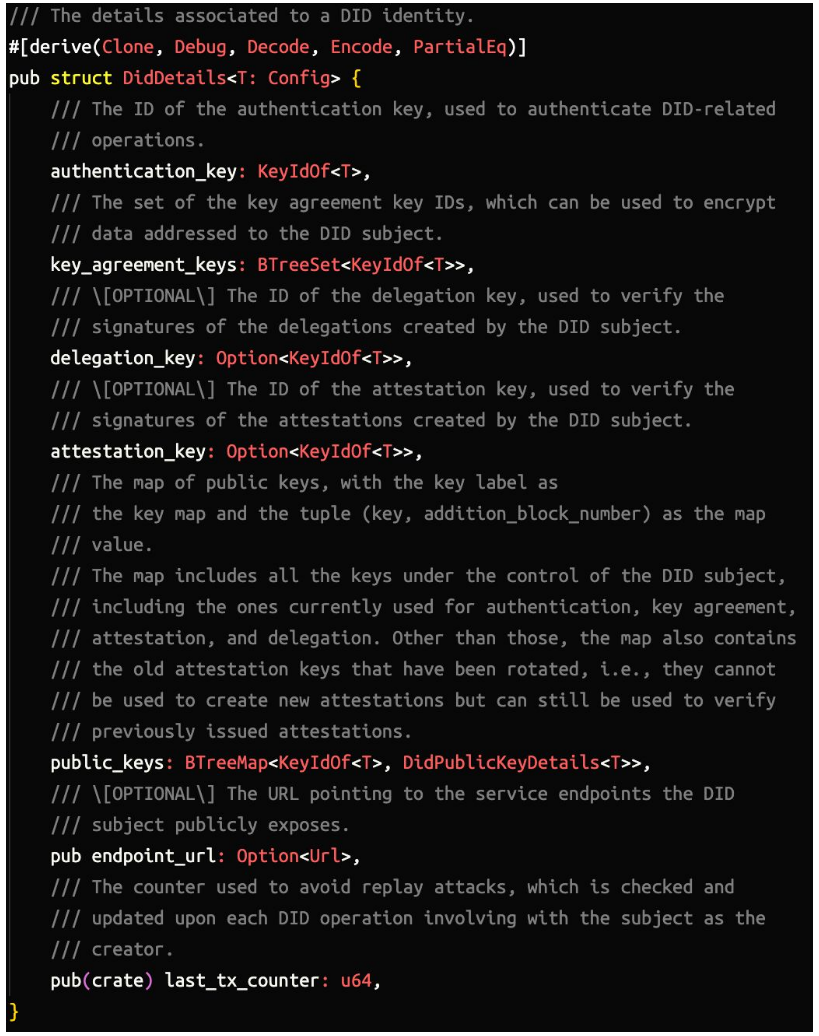
1. Required **authentication** keypair
2. Optional set of **key agreement** keypairs
3. Optional **attestation** keypair -> *assertionMethod* DID key
4. Optional **delegation** keypair -> *capabilityDelegation* DID key
5. Optional URL pointing to **endpoint services***
6. **Encode** the operation **and sign** with given authentication keypair
7. Wrap the encoded operation in a **Substrate tx** and submit to the chain**

*stored off-chain, with the SDK performing additional checks on the document referenced by the URL. Currently supports *HTTP*, *FTP*, and *IPFS* addresses.

**any KILT account can submit the tx, so the DID owner is not required to pay the transaction fees.

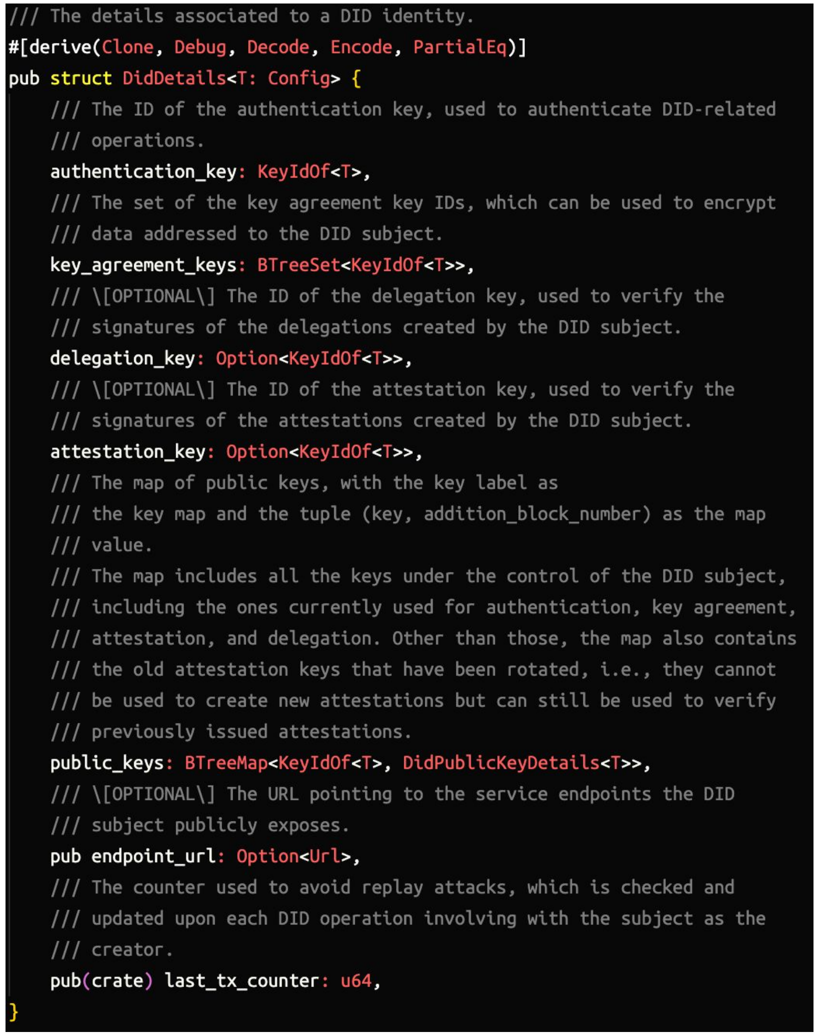


```rust
/// An operation to create a new DID.
///
/// The struct implements the [DidOperation] trait, and as such it must
/// contain information about the caller's DID, the type of DID key
/// required to verify the operation signature, and the tx counter to
/// protect against replay attacks.
#[derive(Clone, Debug, Decode, Encode, PartialEq)]
pub struct DidCreationOperation<T: Config> {
    /// The DID identifier. It has to be unique.
    pub did: DidIdentifierOf<T>,
    /// The new authentication key.
    pub new_authentication_key: DidVerificationKey,
    /// The new key agreement keys.
    pub new_key_agreement_keys: BTreeSet<DidEncryptionKey>,
    /// \[OPTIONAL\] The new attestation key.
    pub new_attestation_key: Option<DidVerificationKey>,
    /// \[OPTIONAL\] The new delegation key.
    pub new_delegation_key: Option<DidVerificationKey>,
    /// \[OPTIONAL\] The URL containing the DID endpoints description.
    pub new_endpoint_url: Option<Url>,
}
```



Full KILT DIDs - Resolution

1. From a DID like *did:kilt:1<kilt_account>* extract the **KILT account**, i.e., the DID identifier
2. **Retrieve** from the KILT chain **the details** associated with the DID subject
3. Optionally, **build a DID document** from the information returned, to maintain interoperability with other ecosystems

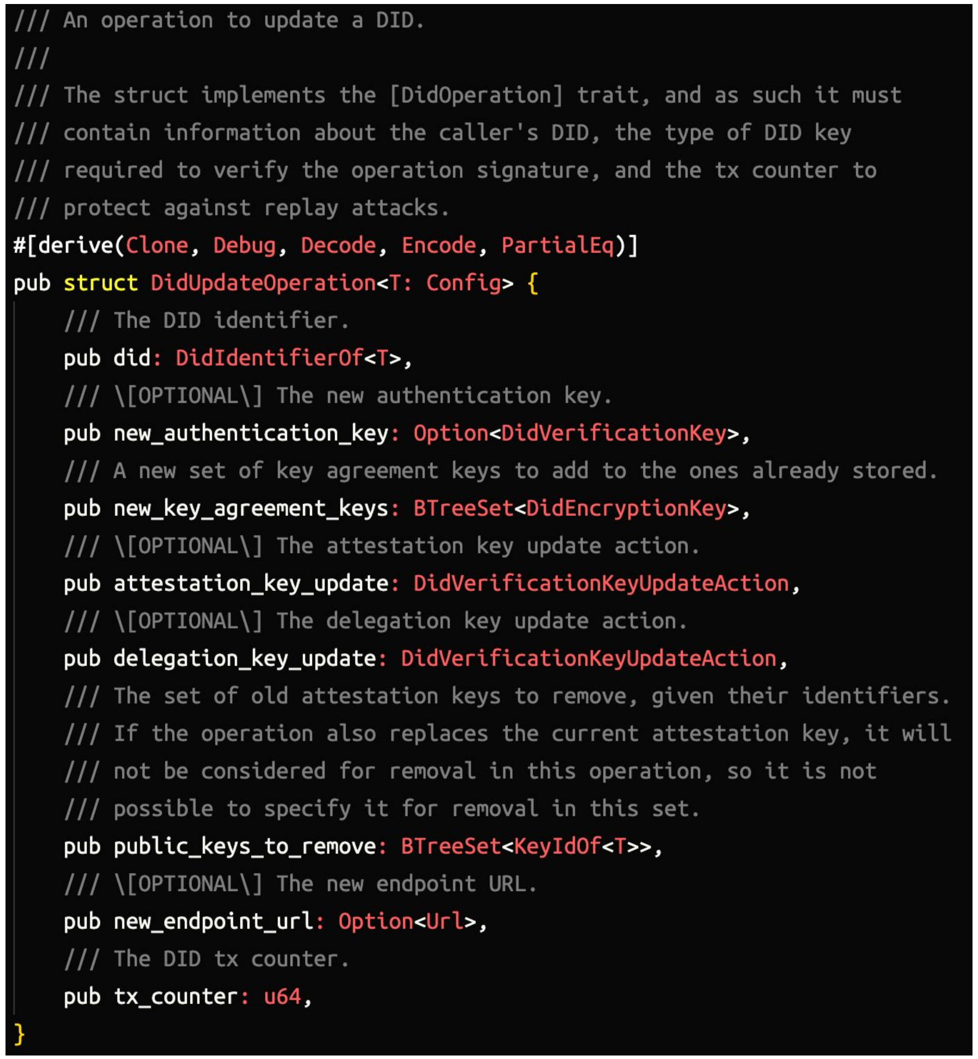


```rust
/// The details associated to a DID identity.
#[derive(Clone, Debug, Decode, Encode, PartialEq)]
pub struct DidDetails<T: Config> {
    /// The ID of the authentication key, used to authenticate DID-related
    /// operations.
    authentication_key: KeyIdOf<T>,
    /// The set of the key agreement key IDs, which can be used to encrypt
    /// data addressed to the DID subject.
    key_agreement_keys: BTreeSet<KeyIdOf<T>>,
    /// \[OPTIONAL\] The ID of the delegation key, used to verify the
    /// signatures of the delegations created by the DID subject.
    delegation_key: Option<KeyIdOf<T>>,
    /// \[OPTIONAL\] The ID of the attestation key, used to verify the
    /// signatures of the attestations created by the DID subject.
    attestation_key: Option<KeyIdOf<T>>,
    /// The map of public keys, with the key label as
    /// the key map and the tuple (key, addition_block_number) as the map
    /// value.
    /// The map includes all the keys under the control of the DID subject,
    /// including the ones currently used for authentication, key agreement,
    /// attestation, and delegation. Other than those, the map also contains
    /// the old attestation keys that have been rotated, i.e., they cannot
    /// be used to create new attestations but can still be used to verify
    /// previously issued attestations.
    public_keys: BTreeMap<KeyIdOf<T>, DidPublicKeyDetails<T>>,
    /// \[OPTIONAL\] The URL pointing to the service endpoints the DID
    /// subject publicly exposes.
    pub endpoint_url: Option<Url>,
    /// The counter used to avoid replay attacks, which is checked and
    /// updated upon each DID operation involving with the subject as the
    /// creator.
    pub(crate) last_tx_counter: u64,
}
```

Full KILT DIDs - Update

1. Any **new keys** to add, change, or remove
2. An optional **new URL** for the services endpoints
3. A **counter** against replay attacks
4. **Encode** the operation **and sign** with the authentication keypair stored on chain
5. Wrap the encoded operation in a **Substrate tx** and submit to the chain

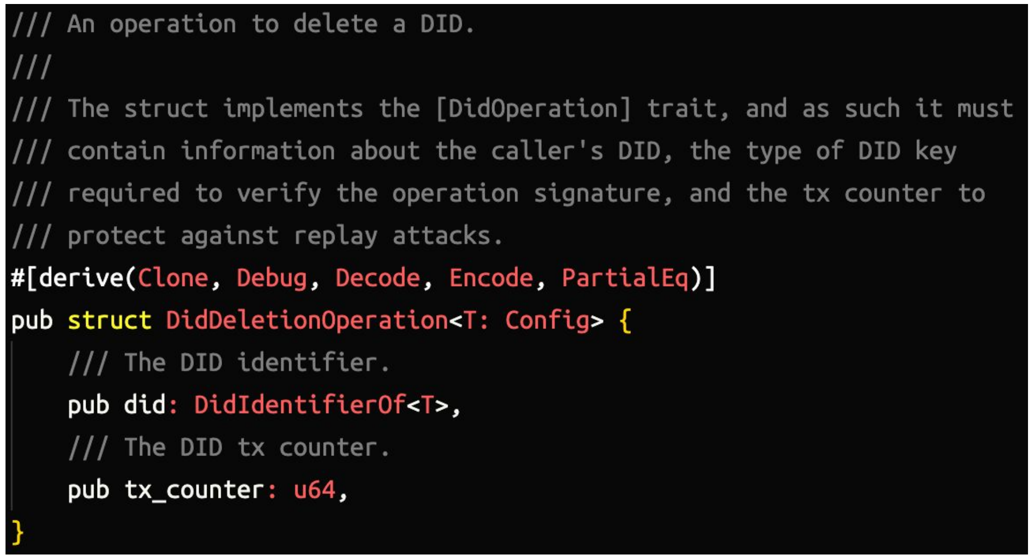
Since **attestation keys** are also used by external entities to verify attestation signatures, once they are replaced they **are still kept in the set of public keys**, unless explicitly deleted by the DID subject -> future verifications of signatures generated with those keys will fail.



```rust
/// An operation to update a DID.
///
/// The struct implements the [DidOperation] trait, and as such it must
/// contain information about the caller's DID, the type of DID key
/// required to verify the operation signature, and the tx counter to
/// protect against replay attacks.
#[derive(Clone, Debug, Decode, Encode, PartialEq)]
pub struct DidUpdateOperation<T: Config> {
    /// The DID identifier.
    pub did: DidIdentifierOf<T>,
    /// \[OPTIONAL\] The new authentication key.
    pub new_authentication_key: Option<DidVerificationKey>,
    /// A new set of key agreement keys to add to the ones already stored.
    pub new_key_agreement_keys: BTreeSet<DidEncryptionKey>,
    /// \[OPTIONAL\] The attestation key update action.
    pub attestation_key_update: DidVerificationKeyUpdateAction,
    /// \[OPTIONAL\] The delegation key update action.
    pub delegation_key_update: DidVerificationKeyUpdateAction,
    /// The set of old attestation keys to remove, given their identifiers.
    /// If the operation also replaces the current attestation key, it will
    /// not be considered for removal in this operation, so it is not
    /// possible to specify it for removal in this set.
    pub public_keys_to_remove: BTreeSet<KeyIdOf<T>>,
    /// \[OPTIONAL\] The new endpoint URL.
    pub new_endpoint_url: Option<Url>,
    /// The DID tx counter.
    pub tx_counter: u64,
}
```

Full KILT DIDs - Deletion

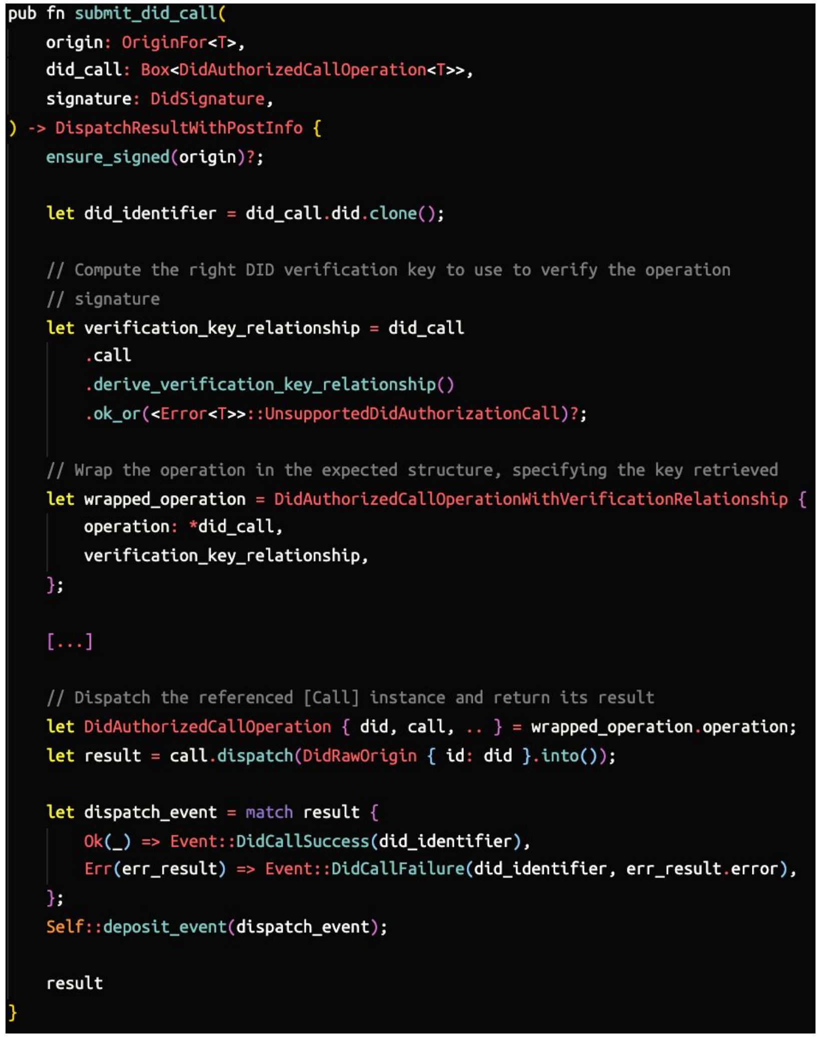
1. A **counter** against replay attacks
2. **Encode** the operation **and sign** with old authentication keypair
3. Wrap the encoded operation in a **Substrate tx** and submit to the chain



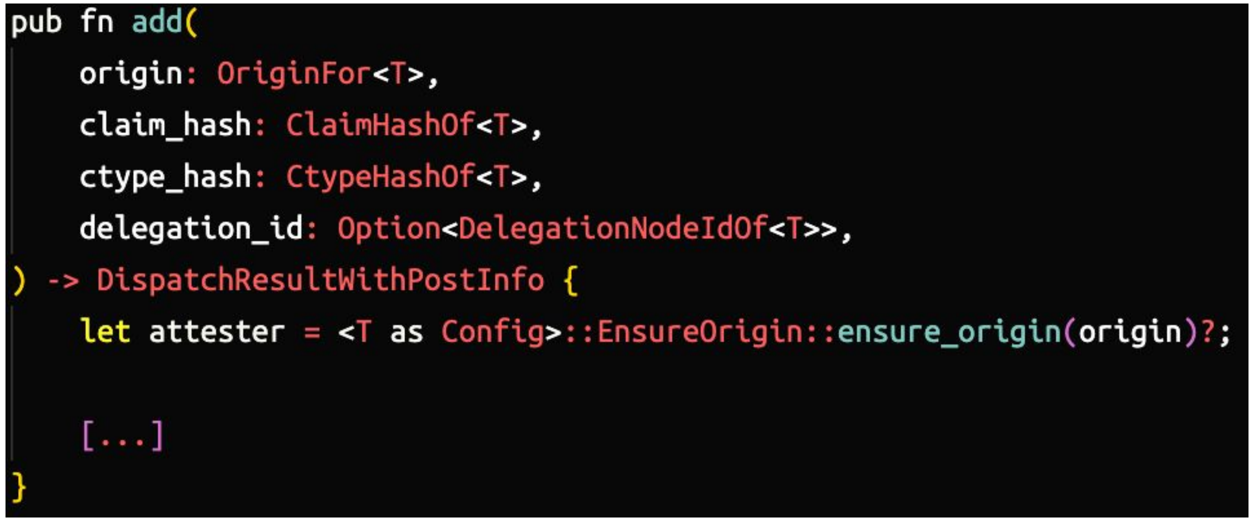
```rust
/// An operation to delete a DID.
///
/// The struct implements the [DidOperation] trait, and as such it must
/// contain information about the caller's DID, the type of DID key
/// required to verify the operation signature, and the tx counter to
/// protect against replay attacks.
#[derive(Clone, Debug, Decode, Encode, PartialEq)]
pub struct DidDeletionOperation<T: Config> {
    /// The DID identifier.
    pub did: DidIdentifierOf<T>,
    /// The DID tx counter.
    pub tx_counter: u64,
}
```

Full KILT DIDs - DID-authorised calls

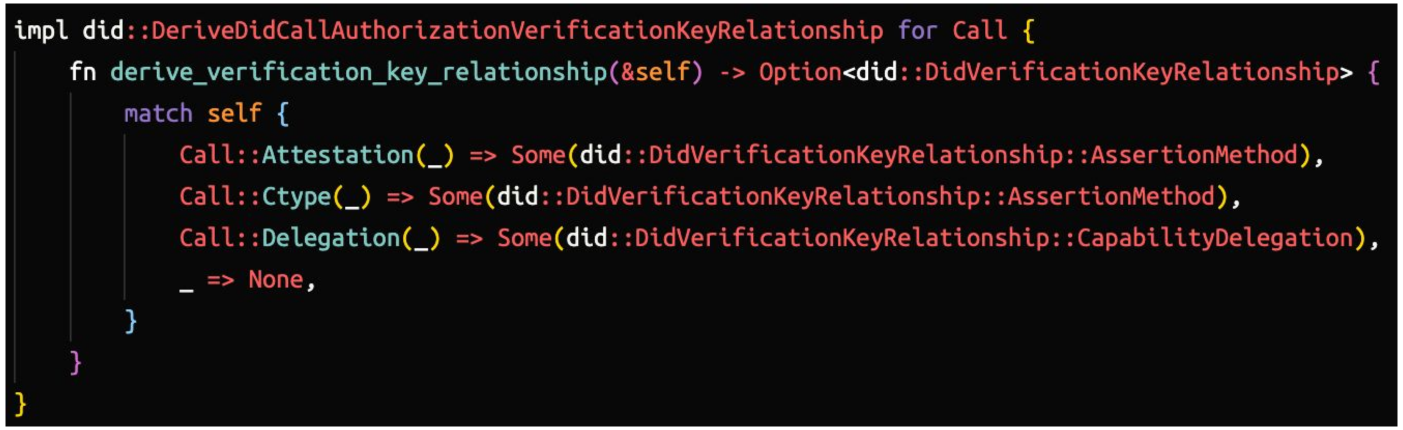
KILT



```rust
pub fn submit_did_call(
    origin: OriginFor<T>,
    did_call: Box<DidAuthorizedCallOperation<T>>,
    signature: DidSignature,
) -> DispatchResultWithPostInfo {
    ensure_signed(origin)?;

    let did_identifier = did_call.did.clone();

    // Compute the right DID verification key to use to verify the operation
    // signature
    let verification_key_relationship = did_call
        .call
        .derive_verification_key_relationship()
        .ok_or(<Error<T>>::UnsupportedDidAuthorizationCall)?;

    // Wrap the operation in the expected structure, specifying the key retrieved
    let wrapped_operation = DidAuthorizedCallOperationWithVerificationRelationship {
        operation: *did_call,
        verification_key_relationship,
    };

    [...]

    // Dispatch the referenced [Call] instance and return its result
    let DidAuthorizedCallOperation { did, call, .. } = wrapped_operation.operation;
    let result = call.dispatch(DidRawOrigin { id: did }.into());

    let dispatch_event = match result {
        Ok(_) => Event::DidCallSuccess(did_identifier),
        Err(err_result) => Event::DidCallFailure(did_identifier, err_result.error),
    };
    Self::deposit_event(dispatch_event);

    result
}
```



```rust
pub fn add(
    origin: OriginFor<T>,
    claim_hash: ClaimHashOf<T>,
    ctype_hash: CtypeHashOf<T>,
    delegation_id: Option<DelegationNodeIdOf<T>>,
) -> DispatchResultWithPostInfo {
    let attester = <T as Config>::EnsureOrigin::ensure_origin(origin)?;

    [...]
}
```



```rust
impl did::DeriveDidCallAuthorizationVerificationKeyRelationship for Call {
    fn derive_verification_key_relationship(&self) -> Option<did::DidVerificationKeyRelationship> {
        match self {
            Call::Attestation(_) => Some(did::DidVerificationKeyRelationship::AssertionMethod),
            Call::Ctype(_) => Some(did::DidVerificationKeyRelationship::AssertionMethod),
            Call::Delegation(_) => Some(did::DidVerificationKeyRelationship::CapabilityDelegation),
            _ => None,
        }
    }
}
```

Demo

Resources

KILT

- DID spec: https://www.w3.org/TR/did-core/
- DID spec: https://www.w3.org/TR/vc-data-model/
- DIDComm spec: https://identity.foundation/didcomm-messaging/spec/
- JSON-LD spec: https://json-ld.org/
- DIF Universal Resolver: https://resolver.identity.foundation/
- KILT on GitHub: https://github.com/KILTprotocol
- KILT DID pallet: https://github.com/KILTprotocol/mashnet-node/tree/develop/pallets/did

Thank you!

github.com/KILTprotocol

KILT.io

@KILTprotocol

#kilt-general:matrix.org