Using OpenID Connect Self-Issued to Achieve DID Auth

a white paper from Rebooting the Web of Trust VIII

by Ivan Basart, Egido Casati, Michael B. Jones, Andrés Junge,
David Stark, Oliver Terbu, and Dmitri Zagidulin

ABSTRACT

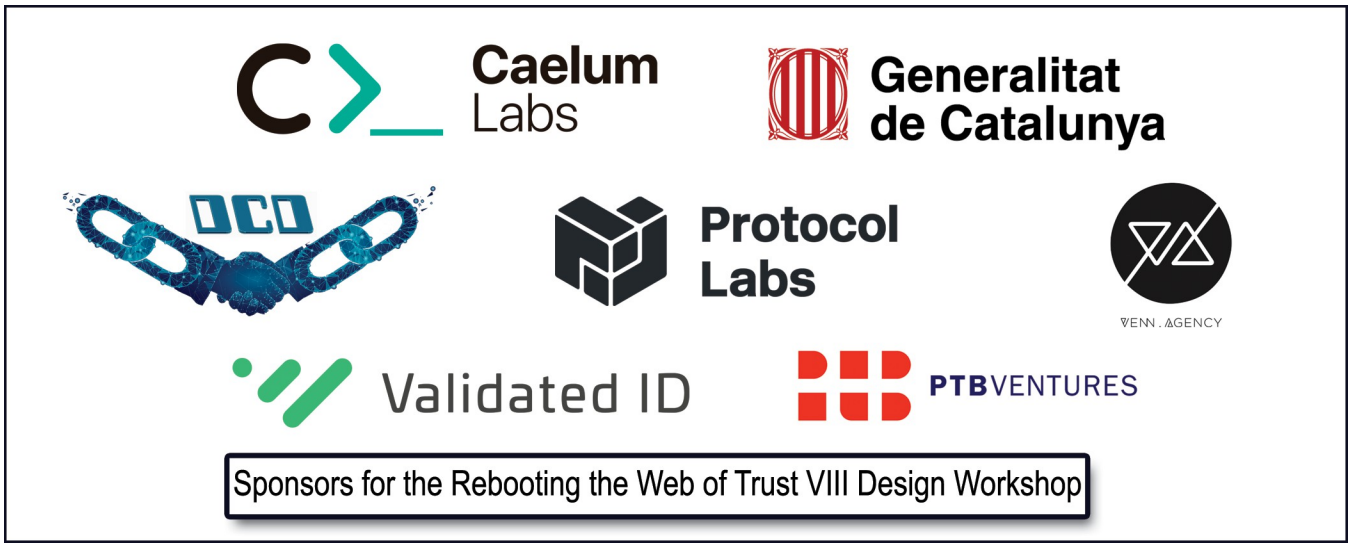
Proving control of a DID requires proving ownership of a private key corresponding to a public key for the DID. Of course, this could be done with a new DID-specific protocol. However, standard protocols for proving ownership of a public/private key pair already exist.

This paper describes how to reuse the Self-Issued OpenID Connect (SIOP) specification and related protocol messages to prove control of a DID. It describes both why and how to do this. Related topics, such as release of claims, are also touched upon.

TERMINOLOGY

The following terminology is used in this paper:

- OP - OpenID Connect Identity Provider. An entity that issues authentication-related assertions (such as ID Tokens).
- SIOP - Self-Issued OpenID Connect Provider. A personal, self-hosted Identity Provider that issue self-signed assertions.
- RP - Relying Party, an application or service that relies on the Identity Provider's assertions.
- DID - Decentralized Identifier.



BACKGROUND AND MOTIVATIONS

OpenID Connect is a widely used JSON/REST-based identity protocol. Section 7 of the OpenID Connect Core specification defines how to authenticate using an identity that you control yourself, which is represented by a public key. The authentication protocol messages prove that you are in possession of the private key corresponding to the public key.

We believe that using the SIOP functionality to prove ownership of a DID has multiple advantages. It reuses existing functionality, possibly accelerating adoption relative to approaches utilizing new custom protocols.

USE CASES

This paper addresses the following use cases:

- Authenticate with (sign into) an RP (relying party / client app)
- Authenticate with (sign into) an RP with a DID
- Presentation of claims

NON-GOALS / OUT OF SCOPE

The following are Non-Goals for this paper:

- Using DID Auth for *local authentication* of user to the Self-Issued OpenID Provider (should be self-evident, no spec needed)
- Exchange of verifiable claims (merits separate paper / specs)
- Zero Knowledge Proofs (could be added by experts in the field)
- Issuing access tokens to resources

NATURE OF THE DESIGN DECISIONS

This paper is guided by the following set of motivations and design decisions:

- Reuse existing specs whenever possible.
- Remain compatible with existing specs that are reused.
- Layer DID Auth functionality such that it can be added incrementally without breaking existing uses of the self-issued functionality.

HOW TO PROVE CONTROL OF A DID

Keys Used

For purposes of DID Authentication, the self-issued key (see the `sub_jwk` in the Authorization Response below) used in the response must be present in the Authentication Section of the DID Document.

Authorization Request

The protocol for making an Authorization Request is almost identical to the one outlined in the SIOP Request spec.

To indicate that the client wants a DID in the response, the request `scope` parameter must include `did_authn` (in addition to the required `openid` scope).

Request Example:

```
openid://?response_type=id_token
    &client_id=https%3A%2F%2Frp.example.com%2Fcb
    &response_type=id_token&client_id=https%3A%2F%2Fmy.rp.com%2Fcb
    &scope=openid%20did_authn
    &nonce=n-0S6_WzA2Mj&scope=openid%20did_authn
```

An example of a JWT payload of a Request Object:

```
{
   "alg": "ES256K",
   "typ": "JWT",
   "kid": "did:example:0xab#veri-key1"
}
{
    "iss": "did:example:0xab",
    "response_type": "id_token",
    "client_id": "https://my.rp.com/cb",
    "scope": "openid did_authn",
    "nonce": "n-0S6_WzA2Mj",
    "registration" : {
        "request_object_signing_alg" : "ES256K",
        "jwks_uri" : "did:example:0xab",
        "id_token_signed_response_alg" : [ "ES256K", "Ed25519", "RS256" ],
    }
}
```

Authorization Response

The format for the SIOP DID Auth response is almost identical to the one described in the SIOP Response definition, with the following additional considerations:

1. The ID Token will have an additional `did` claim, which will contain the DID of the subject.
2. Note that the `sub` claim is a `base64url` encoding of the JWK Thumbprint, as required by OpenID Connect Core Section 7.

Authorization Response Example:


```
{
   "alg": "ES256K",
   "typ": "JWT"
}
{
   "iss": "https://self-issued.me",
   "nonce": "n-0S6_WzA2Mj",
   "exp": 1311281970,
   "iat": 1311280970,
   "sub_jwk" : {
      "crv":"P-256K",
      "kid":"did:example:0xcd#verikey-1",
      "kty":"EC",
      "x":"7KEKZa5xJPh7WVqHJyUpb2MgEe3nA8Rk7eUlXsmBl-M",
      "y":"3zIgl_ml4RhapyEm5J7lvU-4f5jiBvZr4KgxUjEhl9o"
   },
   "sub": "9-aYUQ7mgL2SWQ_LNTeVN2rtw7xFP-3Y2EO9WV22cF0",
   "did" : "did:example:0xcd"
}
```


DID Validation

Note: This section is specifically for the DID Auth use case, and is implemented as an additional separate layer above the regular SIOP flow.

When using this flow the client must validate the response as follows:

1. Verify that the response is a valid SIOP Response definition, and perform the required Self-Issued ID Token Validation steps.
2. Verify that the ID Token returned contains the additional attribute `did` and that it is a valid DID format. (Conforms to the ABNF rules of the DID spec). Note: If the ID Token does not contain a `did` attribute, it may still be a valid SIOP ID Token, but it is not a valid DID Auth response.
3. Perform a DID Resolution operation. (The specifics vary by individual DID method, but typically involves using a resolver to retrieve the DID document specified by the `did` claim in the ID Token.)
4. (Optional, DID method specific) Check that the DID Document is not expired or revoked.
5. Verify that the self-issued key used to sign the response (specified in the `sub_jwk` claim) is present in the `authentication` attribute of the DID Document and is not expired or revoked. The key specified in the DID Document can be in various formats and they must be converted to a common format to compare. This conversion is out of scope of this document.
6. Verify that the algorithm used to sign the ID Token (specified in the ID Token header `alg` claim) matches the algorithm of the corresponding key in the DID Document. Note that the `alg` attribute and the DID Document key may be represented using different serialization formats, and will have to be converted to a common format before doing a comparison. This conversion is out of scope of this document.

RELATIONSHIP TO OTHER OPENID CONNECT FEATURES

Provider Selection via DID Document

A DID Document could optionally contain (in the Service Endpoints section) the user's preferred OpenID Connect Provider (or a set of preferred providers). This is useful for situations where a user's DID is already known by the RP (or transmitted out of band).

Example of an OpenID Connect Provider in the `service` endpoint section of a DID Document:

```
{
  …
  "service": [
    {
       "id": "did:example:123456789abcdefghi#openid",
       "type": "OpenIdConnectVersion1.0Service",
       "serviceEndpoint": "https://openid.example.com/"
    }
  ]
}
```

Provider Discovery

The SIOP Provider Discovery process uses a static configuration document.

Although this static configuration spec only mentions the RS256 crypto algorithm, implementers should keep in mind that other algorithms, such as Ed25519 and ES256K, are more commonly used in the SSI community and may be used in the SIOP DID Auth flow.

Providing Client Meta-Data

Certain use cases require that the RP (client app) provides the OP (identity provider) with additional metadata relevant to the authorization process, such as to exchange the RP's public encryption keys (for encryption of ID Tokens). Since the SIOP flow does not have a Dynamic Registration step, the RP can provide this metadata during the Authorization Request using the optional `registration` parameter.

- If the RP (client app) has its own DID, this may be contained in the `iss` claim of the Authorization Request.
- If the RP has a DID, keys used to encrypt content to that DID could be advertised in the RP's `jwk_set` or `jwk_uri` parameters if encrypted ID Tokens are being used.
- Other client meta-data can be provided using the registration request parameter to indicate the crypto algorithm that the client supports.

Presentation of Claims

The presentation of claims is already supported by OpenID Connect spec, and can include any claims pertinent to a use case.

As specified in OpenID Connect Core, aggregated and distributed claims can be used in responses, including in Self-Issued ID Tokens. For instance, National eID claims, such as claims from EIDAS claim providers, could be represented in that way.

If relevant to the use case, some claims can use JWT claim definitions from W3C Verifiable Credentials spec. Note that OpenID Connect does not define validation rules for claims issued by third parties, as they are, in general, specific to the use case.

Signed Request Objects

The SIOP specification provides the option to encode the authorization request as a request object. The request object could be used by the SIOP to authenticate the RP. In that case, the RP could choose to include a `did` claim in the request object. The signing key of the request object must correlate with one of the keys in the DID Document associated with the DID in the `did` claim.

Non-SIOP OpenID Connect Flows

Although currently specified non-self-issued OpenID Connect authentication flows (either Auth Code or Implicit) cannot be used to prove control of a DID (since they lack the public key as the identifier of the subject of the identity), they can still be useful in associating a particular subject/user with a DID. Specifically, a `did` claim could be returned as a UserInfo or ID Token claim, which means that the OP is asserting that the DID is associated with the End User.

CONCLUSIONS

Reuse of the OpenID Connect Self-Issued mechanism to prove control of a DID is effective and straightforward. The authors believe that reusing this mechanism will be important in promoting DID adoption by defining a bridge to the successful OpenID Connect protocol.

OPEN QUESTIONS

To be explored and specified further:

- Use case where the Self-Issued flow is being initiated from a Desktop browser (or other environment that does not have a protocol handler installed). Current deployed or proposed solutions:
  - ⇨ This is currently being solved by methods like uPort via a combination of QR code which contains a JWT request that includes the response callback of the RP.
  - ⇨ This potentially could be solved by an agreed-upon site hosted by the community, “Log in with SSI”, that would redirect desktop user to a page that shows the QR code.

BIBLIOGRAPHY / PREVIOUS PAPERS AND PROPOSALS

- OIDC Profile for SSI by Oliver Terbu, Andres Junge
- Proof of Key Ownership with OpenID Connect Self-Issued Identities by Michael B. Jones
- Introduction to DID Auth by Markus Sabadello, Kyle Den Hartog, Christian Lundkvist, Cedric Franz, Alberto Elias, Andrew Hughes, John Jordan, Dmitri Zagidulin
- Open ID v. FIDO v. SSI

Related Specs and References

- https://openid.net/specs/openid-connect-core-1_0.html , [OIDC]
- https://www.youtube.com/watch?v=tTeN_SxQ_OI , [NSvideo18]
- https://nat.sakimura.org/2018/12/11/todo-list-for-self-issued-op/ , [NSblog18]
- https://iiw.idcommons.net/OIDC_DID-Auth_Profile , [II18]
- https://w3c.github.io/vc-data-model/#json-web-token , [W3C]

Additional Credits

Lead Authors: Oliver Terbu and Dmitri Zagidulin

Authors: Ivan Basart, Egido Casati, Michael B. Jones, Andrés Junge, and David Stark

---

Sample APA Citation:

Basart, I., Casati, E., Jones, M.B., Junge, A., Stark, D., Terbu, O. and Zagidulin, D. (2019). Using OpenID Connect Self-Issued to Achieve DID Auth. *Rebooting the Web of Trust VIII*. Retrieved from https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/final-documents/did-auth-oidc.pdf.

About Rebooting the Web of Trust

This paper was produced as part of the Rebooting the Web of Trust VIII design workshop. On March 1st to 3rd, 2019, over 80 tech visionaries came together in Barcelona, Spain to talk about the future of decentralized trust on the internet with the goal of writing at least 5 white papers and specs. This is one of them.

RWOT Board of Directors: Christopher Allen, Joe Andrieu, Kim Hamilton Duffy

Silver Sponsors: Caelum Labs, Digital Contract Design, Generalitat de Catalunya, Protocol Labs, Venn Agency

Additional Sponsors: Validated ID, PTB Ventures

Community Sponsors: Blockchain Commons, Digital Bazaar, In Turn Information Management Consulting,

Learning Machine, Legendary Requirements

Workshop Credits: Christopher Allen (Founder), Joe Andrieu (Producer and Facilitator), Shannon Appelcline (Editor-in-chief), and Carlotta Cataldi (Graphical Recorder)

Thanks to our other contributors and sponsors!

What's Next?

The design workshop and this paper are just starting points for Rebooting the Web of Trust. If you have any comments, thoughts, or expansions on this paper, please post them to our GitHub issues page:

https://github.com/WebOfTrustInfo/rwot8/issues

The ninth Rebooting the Web of Trust design workshop is scheduled for September 3rd-6th 2019 in Prague, The Czech Republic. If you'd like to be involved or would like to help sponsor the event, email:

rwot-leadership@googlegroups.com
