framework.

You're suggesting

For example, the Verifier might authenticate using a x.509 Certificate (through a cert chain) issued by a trusted CA. Other technologies, like ETSi trusted list or OpenID Federation might be used as well. Those mechanisms are differentiated by different client id schemes.

The wallet then decodes and processes the request. Here is an example of a decoded OID4VP request requesting a mDL:

```
{
 "client_id": "verifier.example.com",
 "aud": "https://self-issued.me/v2",
 "client_id_scheme": "x509_san_dns",
 "nonce": "n-0S6_WzA2Mj",
 "client_metadata": {
```

```
  "authorization_encrypted_response_alg": "ECDH-ES",
  "authorization_encrypted_response_enc": "A256GCM",
  "vp_formats": {
     "mso_mdoc": {
       "alg": [
         "ES256"
       ]
     }
  }
```

response.

Tobias Looker
9:27 PM Yesterday

I think it is worthwhile mentioning what we discussed on the DCP WG call today. In my opinion if this browser API is anything like Web Authn the browser will provide the origin of the verifier to the wallet which can be used as the basis for identification and authentication. This can then be supplemented by additional mechanisms such as an X.509 certificate chain in the cases where the origin provided by the browser isn't sufficient assurance.

Show less

Tom Jones
9:55 AM Today

should i take this to mean that the protocol will only work with browsers and only when the TLS data is available to the application

Reply or add others with @