OAuth 2.0 Attestation-Based Client Authentication

- Refresher
- Updates since IETF 119
- Discussion points
- Q&A

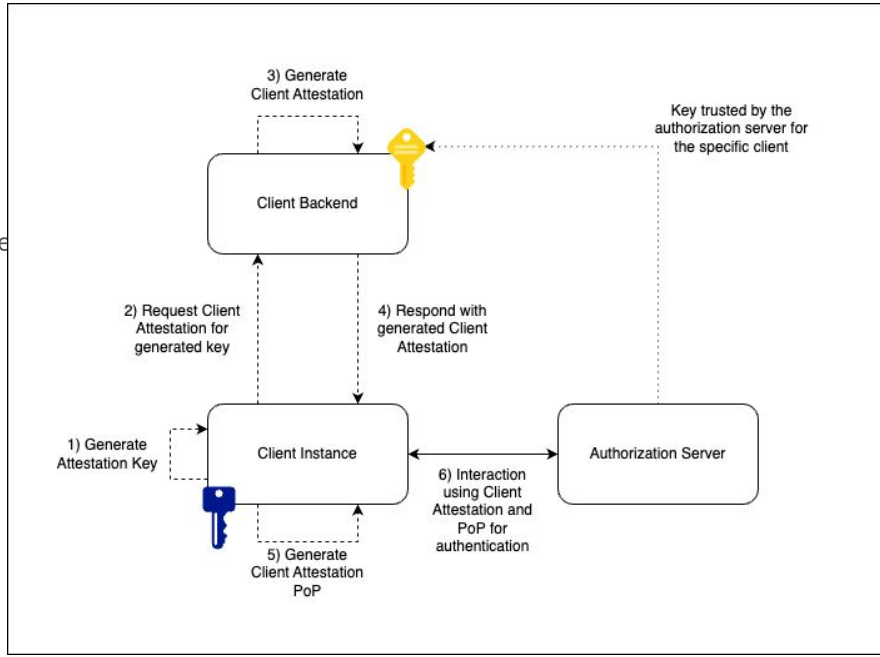
Tobias Looker, Paul Bastian
(Adding Christian Bormann to editors)

A Refresher - Motivation

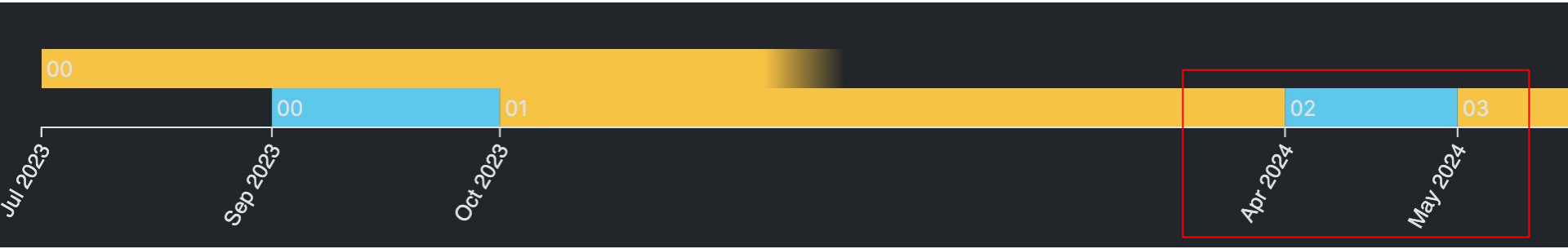
- Backend vs frontend channel authentication
  - Classical OAuth Security model requires client authentication through a backend channel, which also causes the transaction to be conducted through the backend channel
  - In the context of OpenID4VCI and the Issuer-Holder-Verifier model, this creates significant privacy issues, as the backend gets to see all credentials/tokens
  - Establish **a mechanism for backend-attested client authentication through a front-end channel**
- Establish a mechanism to enable technologically independent attestations that give AS assurances about the security level of the Client

Architecture

- Differentiate Client and Client Instance
- **Client Backend attests a Client Instance with a Client Instance Key and provides a key-bound Client Attestation JWT**
- Client Backend may perform any number of security checks before issuing a key-bound Client Attestation JWT to the client instance, however, steps 2 and 4 are out of scope
- **Client Instance generates a Proof of Possession and utilize both to authenticate towards AS**
- Trust mechanism for the Client Backend public key is out of scope
- Avoids the client instance from having to register with the AS via DCR
    - May still work in conjunction with DCR



Changes since IETF 119 (Brisbane)



Changes: -02

- add text on the inability to rotate the Client Instance Key

Changes: -03

- remove usage of RFC7521 and the usage of client_assertion
- **add new header-based syntax introducing Oauth-Client-Attestation and OAuth-Client-Attestation-PoP**
- add Client Instance to the terminology and improve text around this concept

Previous Token Request

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-client-attestation&
client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.
eyJpc3Mi[...omitted for brevity...].
cC4hiUPo[...omitted for brevity...]~
eyJzI1NiIsImtphbGciOimtpZCI6IjIyIn0.
IjIyIn0[...omitted for brevity...].
iOiJSUzI1[...omitted for brevity...]
```

New assertion type

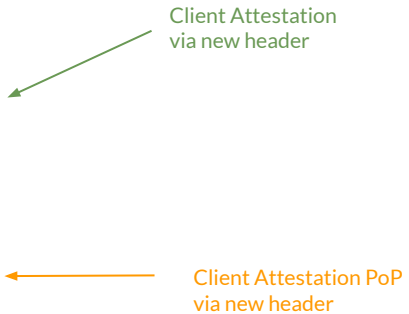
Two JWTs concatenated via a '~' character
- Client Attestation
- Client Attestation PoP

deprecated

New Header-based Syntax

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
OAuth-Client-Attestation: eyJhbGciOiAiRVMyNTYiLCJraWQiOiAiMTEifQ.eyJ\
pc3MiOiJodHRwczovL2NsaWVudC5leGFtcGxlLmNvbSIsInN1YiI6Imh0dHBzOi8vY2x\
pZW50LmV4YW1wbGUuY29tIiwibmJmIjoxMzAwODE1NzgwLCJleHAiOjEzMDA4MTkzODA\
sImNuZiI6eyJqd2siOnsia3R5IjoiRUMiLCJ1c2UiOiJzaWciLCJjcnYiOiJQLTI1NiI\
sIngiOiIxOHdITGVJZ1c5d1ZONlZEMVR4Z3BxeTJMc3pZa01mNko4bmpWQWlidmhNIiw\
ieSI6Ii1WNGRTNFVhTE1nUF80Zlk0ajhpcjdjbDFUWGxGZEFnY3g1NW83VGtjU0EifX1\
9.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiJ9.eyJpc3MiOiJodHRwc\
zovL2NsaWVudC5leGFtcGxlLmNvbSIsImF1ZCI6Imh0dHBzOi8vYXMuZXhhbXBsZS5jb\
20iLCJuYmYiOjEzMDA4MTU3ODAsImV4cCI6MTMwMDgxOTM4MH0.coB_mtdXwvi9RxSMz\
bIey8GVVQLv9qQrBUqmc1qj9Bs

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4
```



Discussion - DPoP Optimization

The main trade off with this proposal is that the DPoP key and the Client instance key MUST be the same. Which can viewed as both a useful simplification or a constraint.

Does the working group have an opinion on this? Are there practical use cases where the DPoP key and Client instance key need to be different? Or is the complexity it creates more hassle than it is worth?

DPoP Optimization

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
DPoP:
eyJ0eXAiOiJkcG9wK2p3dCIsImp3ayI6eyJhbGciOiJFUzI1NiIsImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiLCJ4IjoiaThReW03NFRNUHVLQXV
KUGlZczFSZlVsYTVjemNxelVobEpmRHNMdzd0NCIsInkiOiJGQjlUY2ZmeVZDSEpFQjJjejc4NTE2MUE0SmxlTkh2cG44bXhHRldZMlNjIn0sIm
FsZyI6IkVTMjU2In0.eyJqdGkiOiIzNTc2ODI5Ny1kZWM1LTQ2ZjYtODVlNS1iNzU4MjE2YWI1ZmYiLCJodG0iOiJQT1NUIiwiaHR1IjoiaHR0c
HM6Ly9hcy5leGFtcGxlL3Rva2VuIiwiaWF0IjoxNzAwODEyODAwLCJub25jZSI6ImV5SjdTX3pHLmV5SkgwLVouSFg0dy03diJ9.5VuDrkd8RhM
Raps_AzJBs2p-_UXXWT4dVHITBHiQxe31GeDq81otnIh3HBQN8_XjS1diHPq1tti1pn55eZdI5g
OAuth-Client-Attestation: eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.eyJpc3Mi[...omitted for brevity...].
cC4hiUPo[...omitted for brevity...]

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
```

Client Attestation PoP via DPoP syntax

Client Attestation via new header

Draft Naming Discussion

- Feedback from Torsten and others
- The word “Attestation” is already used certain contexts like RATS
  - See Issue#60
  - However, Attestation in context of eIDAS actually fitting
- Torsten’s proposal: “Key-bound JWT Client Authentication” , see Issue#71
- Opinions?

Discussion to nonce fetching

- Section 7.1 on Replay Attack Detection recommends nonce over jti
- However, explicit nonce fetching mechanism is not described yet
  - Consider reusing DPoP mechanism, yet DPoP integration is thought to be optional
  - Require independent mechanism
- Thoughts so far:
  - As we move to header syntax, request nonce via headers
  - DPoP mechanism may result in nonce being too old and AS requesting new DPoP proof with fresh nonce
    - This may be very costly in the context of this draft when using hardware-backed crypto(external or remote HSM)
    - Otherwise requesting a new nonce with HTTP 400 result
  - Enable new mechanism for the Client to explicitly request a new nonce, i.e. new header OAuth-Client-Attestation-Nonce
    - Especially useful for Client Authentication at PAR endpoint without prior interaction

Questions?