OAuth 2.0 Attestation-Based Client Authentication

- Refresher
- Updates since IETF 121
- Discussion points
- Q&A

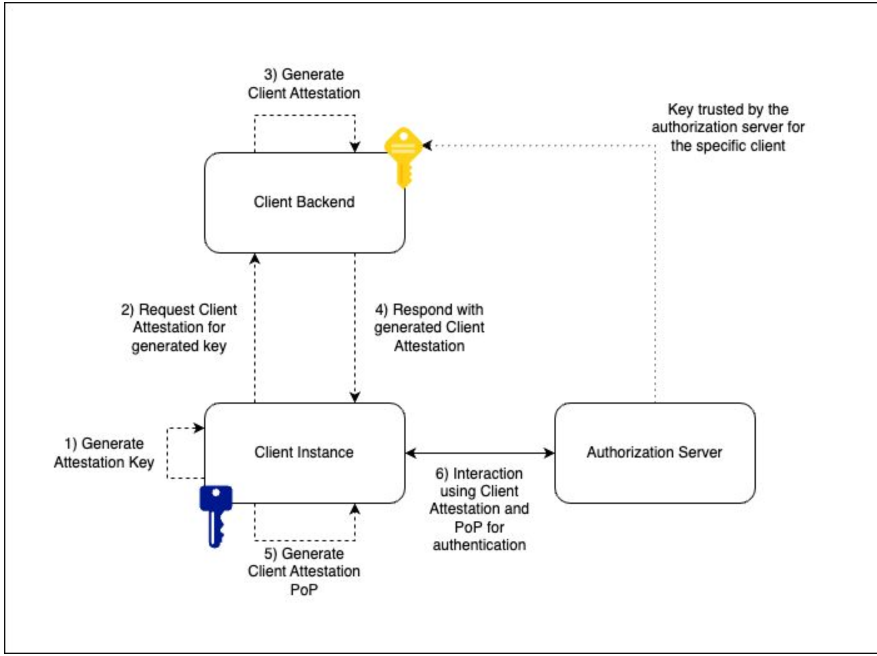
Tobias Looker, Paul Bastian, Christian Bormann

A Refresher - Motivation

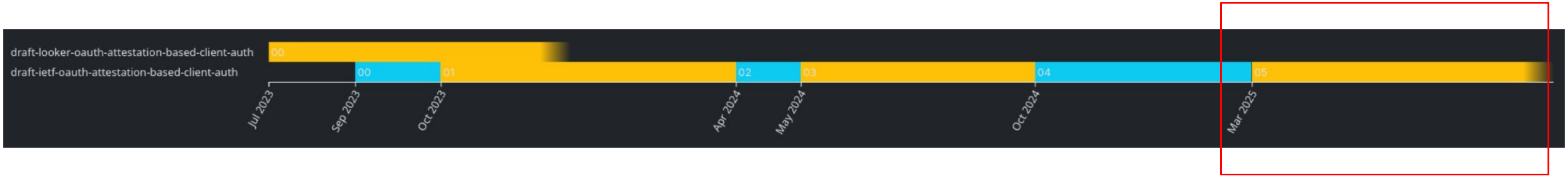
- Backend vs frontend channel authentication
  - Classical OAuth Security model requires client authentication through a backend channel, which also causes the transaction to be conducted through the backend channel
  - In the context of OpenID4VCI and the Issuer-Holder-Verifier model, this creates significant 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 overview

- 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
- Avoids the client instance from having to register with the AS via DCR
    - May still work in conjunction with DCR



Changes since IETF 121



Changes: -05

- **add nonce endpoint**
- add metadata entry for nonce
- **improve introduction**
- rename client backend to client attester
- fix missing typ header in examples

Rewritten Introduction to emphasize Rationale

- [...] this approach acknowledges the evolving landscape of OAuth 2 deployments, where the ability for public clients to authenticate securely and reliably has become increasingly important.
- Leveraging platform mechanisms to validate a client instance, e.g. for mobile native apps, enables secure authentication that would otherwise be difficult with traditional OAuth client authentication methods. Transforming these platform-specific mechanisms into a common format as described in this specification abstracts this complexity to minimize the efforts for the Authorization Server.
- This **primary purpose of this specification is the authentication of a client instance enabled through the client backend attesting to it.** The client backend may also attest further technical properties about the hardware and software of the client instance.

Nonce fetching

- Was discussed at OSW 2025
- Simple solution using HTTP OPTIONS with header at the relevant endpoint

8. Nonce Retrieval

This specification defines header fields that allow a Client to request a fresh nonce value to be used in the OAuth-Client-Attestation-PoP. The nonce is opaque to the client.

An Authorization Server compliant with this specification **SHOULD** signal via the metadata entry `client_attestation_pop_nonce_required` which endpoints support and expect a server-provided nonce. The client **MUST** retrieve a nonce before other calls to this endpoint and **MUST** use this nonce for the Client Attestation PoP.

A Request to an endpoint supporting the server-provided nonce **MUST** include the `attestation-nonce-request` field name with the value `true` and use the HTTP method of type OPTIONS (without payload) to actively request a nonce. The server answers with an HTTP Response with status code 200 without body, but sets the header field `attestation-nonce` to the nonce.

The client **MUST** use this nonce in the OAuth-Attestation-PoP as defined in (#client-attestation-pop-jwt).

The following is a non-normative example of a request:

```
OPTIONS /as/par HTTP/1.1
Host: as.example.com
attestation-nonce-request: true
```

the following is a non-normative example of a response:

```
HTTP/1.1 200 OK
Host: as.example.com
```

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

Questions?