draft-bweeks-acme-device-attest-00

Brandon Weeks, Google
tl;dr

- Describes how WebAuthn attestation statements can be included in a challenge response payload to attest to the identity of the requesting device along with the key generation parameters.
- Primary use case is issuing client certificates.
Why ACME?

- SCEP, despite its flaws, remain the primary certificate enrollment protocol used for client certificate enrollment.
- ACME has an extensible design that permits inclusion of attestation with few changes.
- Ubiquitous library support.
Why now?

- Attestation schemes have matured and become ubiquitous:
  - Android Key Attestation (Android)
  - Managed Device Attestation (iOS, macOS soon?)
  - Chrome Verified Access (Chrome OS)
  - RATS Entity Attestation Token (eventually?)
  - Trusted Platform Module (Linux, Windows)
WebAuthn attestation statement format usage

- **In the wild**
  - Apple [App Attest](#)
  - WebAuthn :)

- **IETF drafts**
  - [draft-fossati-tls-attestation-00](#) (tls)
  - [draft-wallace-lamps-key-attestation-ext-00](#) (lamps)

- Ubiquitous library support for CBOR, COSE, and WebAuthn.
ACME extension

- **device-attest-01 challenge**
  - Challenge response payload contains the attestation statement, instead of an empty JSON object.
  - Key authorization is used as the WebAuthn nonce.

- **Identifiers**
  - permanent-identifier (RFC 4043)
  - hardware-module (RFC 4108)

- **EAB for pre-authorization to the CA**
Implementations

- Demonstration CA / client
  - [https://github.com/brandonweeks/acme-device-attest-demo](https://github.com/brandonweeks/acme-device-attest-demo)
  - Upstream: [https://github.com/smallstep/certificates/pull/977](https://github.com/smallstep/certificates/pull/977)

- iOS 16
Open questions

- Is this the right document to specify how key properties should be reflected in issued client certificates?
- Verification procedures and trust anchor selection is complex and poorly specified. Where should the procedures be specified?