

ACME Working Group
Internet-Draft
Intended status: Standards Track
Expires: 11 June 2026

B. Weeks
G. Mallaya
S. Rajala
8 December 2025

Automated Certificate Management Environment (ACME) Device Attestation
Extension
draft-ietf-acme-device-attest-00

Abstract

This document specifies new identifiers and a challenge for the Automated Certificate Management Environment (ACME) protocol which allows validating the identity of a device using attestation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 June 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Conventions and Definitions	3
3. Permanent Identifier	3
4. Hardware Module	4
5. Device Attestation Challenge	4
6. Security Considerations	6
7. IANA Considerations	6
7.1. ACME Identifier Types	6
7.2. ACME Validation Method	7
7.3. New Error Types	7
8. References	7
8.1. Normative References	7
8.2. Informative References	8
Appendix A. Enterprise PKI	9
A.1. External Account Binding	9
Acknowledgments	9
Authors' Addresses	9

1. Introduction

The Automatic Certificate Management Environment (ACME) [RFC8555] standard specifies methods for validating control over identifiers, such as domain names. It is also useful to be able to validate properties of the device requesting the certificate, such as the identity of the device /and whether the certificate key is protected by a secure cryptoprocessor.

Many operating systems and device vendors offer functionality enabling a device to generate a cryptographic attestation of their identity, such as:

- * Android Key Attestation
(<https://source.android.com/security/keystore/attestation>)
- * Chrome OS Verified Access (<https://developers.google.com/chrome/verified-access/overview>)
- * Trusted Platform Module
(<https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/>)
- * Managed Device Attestation for Apple Devices
(<https://support.apple.com/en-om/guide/deployment/dep28afbde6a/web>)

Using ACME and device attestation to issue client certificates for enterprise PKI is anticipated to be the most common use case. The following variances to the ACME specification are described in this document:

- * Addition of permanent-identifier [RFC4043] and hardware-module [RFC4108] identifier types.
- * Addition of the device-attest-01 challenge type to prove control of the permanent-identifier and hardware-module identifier types.
- * The challenge response payload contains a serialized WebAuthn attestation statement format instead of an empty JSON object ({}).
- * Accounts and external account binding being used as a mechanism to pre-authenticate requests to an enterprise CA.

This document does not specify the attestation verification procedures. Section 13 of [WebAuthn] gives some guidance, however verification procedures are complex and may require changes to address future security issues.

Efforts are underway within the Remote ATtestation Procedures (RATS) working group to define a set of standard formats and protocols for attestation. An explicit aim of this document is to support vendor specific formats and protocols that are widely deployed at the time it was authored. In the future, an ACME challenge type based on these standards SHOULD be used instead of device-attest-01.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Permanent Identifier

A new identifier type, "permanent-identifier" is introduced to represent the identity of a device assigned by the manufacturer, typically a serial number. The name of this identifier type was chosen to align with [RFC4043], it does not prescribe the lifetime of the identifier, which is at the discretion of the Assigner Authority.

The identity along with the assigning organization can be included in the Subject Alternate Name Extension using the PermanentIdentifier form described in [RFC4043].

Clients MAY include this identifier in the certificate signing request (CSR). Alternatively if the server wishes to only issue privacy-preserving certificates, it MAY reject CSRs containing a `PermanentIdentifier` in the `subjectAltName` extension.

4. Hardware Module

A new identifier type, "hardware-module" is introduced to represent the identity of the secure cryptoprocessor that generated the certificate key.

The hardware module identity can be included in the Subject Alternate Name Extension using the `HardwareModuleName` form described in [RFC4108]. The `HardwareModuleName` is encoded as an `otherName` with the OID `id-on-hardwareModuleName` (1.3.6.1.5.5.7.8.4) and consists of:

- * `hwType`: An OBJECT IDENTIFIER that identifies the type of hardware module
- * `hwSerialNum`: An OCTET STRING containing the hardware module serial number

Clients MAY include this identifier in the certificate signing request (CSR). When included in a CSR, it MUST appear in an `extensionRequest` attribute [RFC2985] requesting a `subjectAltName` extension.

If the server includes `HardwareModule` in the `subjectAltName` extension the CA MUST verify that the certificate key was generated on the secure cryptoprocessor with the asserted identity and type. The key MUST NOT be able to be exported from the cryptoprocessor.

If the server wishes to issue privacy-preserving certificates, it MAY omit `HardwareModule` from the `subjectAltName` extension.

5. Device Attestation Challenge

The client can prove control over a permanent identifier of a device by providing an attestation statement containing the identifier of the device.

The `device-attest-01` ACME challenge object has the following format:

`type` (required, string): The string "device-attest-01".

`token` (required, string): A random value that uniquely identifies

the challenge. This value MUST have at least 128 bits of entropy. It MUST NOT contain any characters outside the base64url alphabet, including padding characters ("="). See [RFC4086] for additional information on randomness requirements.

```
{
  "type": "device-attest-01",
  "url": "https://example.com/acme/chall/Rg5dV14Gh1Q",
  "status": "pending",
  "token": "evaGxfADs6pSRb2LAv9IZf17Dt3juxGJ-PcT92wr-oA"
}
```

A client fulfills this challenge by constructing a key authorization (Section 8.1 of [RFC8555]) from the "token" value provided in the challenge and the client's account key. The client then generates a WebAuthn attestation object using the key authorization as the challenge.

This specification borrows the WebAuthn `_attestation object_` representation as described in Section 6.5.4 of [WebAuthn] for encapsulating attestation formats, but with these modifications:

- * The key authorization is used to form `_attToBeSigned_`. This replaces the concatenation of `_authenticatorData_` and `_clientDataHash_`. `_attToBeSigned_` is hashed using an algorithm specified by the attestation format.
- * The `_authData_` field is unused and SHOULD be omitted.

A client responds with the response object containing the WebAuthn attestation object in the "attObj" field to acknowledge that the challenge can be validated by the server.

On receiving a response, the server constructs and stores the key authorization from the challenge's "token" value and the current client account key.

To validate a device attestation challenge, the server performs the following steps:

1. Perform the verification procedures described in Section 6 of [WebAuthn].
2. Verify that key authorization conveyed by `_attToBeSigned_` matches the key authorization stored by the server.

```
POST /acme/chall/Rg5dV14Gh1Q
Host: example.com
Content-Type: application/jose+json
```

```
{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://example.com/acme/acct/evOfKhNU60wg",
    "nonce": "SS2sSl1PtspvFZ08kNtzKd",
    "url": "https://example.com/acme/chall/Rg5dV14Gh1Q"
  }),
  "payload": base64url({
    "attObj": base64url(/ * WebAuthn attestation object */),
  }),
  "signature": "Q1bURgJoEslbD1c5...3pYdSMLio57mQNN4"
}
```

The webauthn payload MAY contain any identifiers registered in "WebAuthn Attestation Statement Format Identifiers" and any extensions registered in "WebAuthn Extension Identifiers" [IANA-Webauthn], [RFC8809].

6. Security Considerations

See Section 13 of [WebAuthn] for additional security considerations related to attestation statement formats, including certificate revocation.

Key attestation statements may include a variety of information in addition to the public key being attested. While not described in this document, the server MAY use any policy when evaluating this information. This evaluation can result in rejection of a certificate request that features a verifiable key attestation for the public key contained in the request. For example, an attestation statement may indicate use of an unacceptable firmware version.

7. IANA Considerations

7.1. ACME Identifier Types

The "ACME Identifier Types" registry is to be updated to include the following entries:

Label	Reference
permanent-identifier	RFC XXXX
hardware-module	RFC XXXX

Table 1

7.2. ACME Validation Method

The "ACME Validation Methods" registry is to be updated to include the following entry:

Label	Identifier Type	ACME	Reference
device-attest-01	permanent-identifier	Y	RFC XXXX

Table 2

7.3. New Error Types

This document adds the following entries to the ACME Error Type registry:

Type	Description	Reference
badAttestationStatement	The attestation statement is unacceptable (e.g. not signed by an attestation authority trusted by the CA)	RFC XXXX

Table 3

* Change Controller:

- W3C Web Authentication Working Group (public-webauthn@w3.org)

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, DOI 10.17487/RFC2985, November 2000, <<https://www.rfc-editor.org/rfc/rfc2985>>.
- [RFC4043] Pinkas, D. and T. Gindin, "Internet X.509 Public Key Infrastructure Permanent Identifier", RFC 4043, DOI 10.17487/RFC4043, May 2005, <<https://www.rfc-editor.org/rfc/rfc4043>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/rfc/rfc4086>>.
- [RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, DOI 10.17487/RFC4108, August 2005, <<https://www.rfc-editor.org/rfc/rfc4108>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <<https://www.rfc-editor.org/rfc/rfc8555>>.
- [RFC8809] Hodges, J., Mandyam, G., and M. Jones, "Registries for Web Authentication (WebAuthn)", RFC 8809, DOI 10.17487/RFC8809, August 2020, <<https://www.rfc-editor.org/rfc/rfc8809>>.
- [WebAuthn] Hodges, J., Jones, J., Jones, M. B., Kumar, A., and E. Lundberg, "Web Authentication: An API for accessing Public Key Credentials Level 2", April 2021, <<https://www.w3.org/TR/webauthn-2/>>.

8.2. Informative References

[IANA-Webauthn]

"IANA Registries for Web Authentication (WebAuthn)", n.d.,
<<https://www.iana.org/assignments/webauthn/webauthn.xhtml>>.

Appendix A. Enterprise PKI

ACME was originally envisioned for issuing certificates in the Web PKI, however this extension will primarily be useful in enterprise PKI. The subsection below covers some operational considerations for an ACME-based enterprise CA.

A.1. External Account Binding

An enterprise CA likely only wants to receive requests from authorized devices. It is RECOMMENDED that the server require a value for the "externalAccountBinding" field to be present in "newAccount" requests.

If an enterprise CA desires to limit the number of certificates that can be requested with a given account, including limiting an account to a single certificate. After the desired number of certificates have been issued to an account, the server MAY revoke the account as described in Section 7.1.2 of [RFC8555].

Acknowledgments

TODO acknowledge.

Authors' Addresses

Brandon Weeks
Email: me@brandonweeks.com

Ganesh Mallaya
Email: ganesh.mallaya@appviewx.com

Sven Rajala
Email: sven.rajala@keyfactor.com