ACME Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 November 2026

B. Weeks
Google Inc
G. Mallaya
AppViewX Inc.
S. Rajala
Keyfactor
C. Bonnell
DigiCert, Inc.
R. Hurst
Peculiar Ventures

5 May 2026

Automated Certificate Management Environment (ACME) Device Attestation Extension
draft-ietf-acme-device-attest-04

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 6 November 2026.

Copyright Notice

Table of Contents

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 will be a 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 publication time of this specification.

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. Additionally, the assigner of the identifier MAY also be specified. The name of this identifier type was chosen to align with [RFC4043]. This specification does not prescribe the lifetime of the identifier, which is at the discretion of the Assigner Authority.

Although [RFC4043] permits any valid UTF-8 string to be used as the identifier, this specification mandates that identifiers MUST NOT contain the forward-slash "/" (UTF-8: U+002F) character. This restriction is required to make the ABNF production rule for the permanent-identifier-value unambiguous.

3.1. Representation in Order resources

The identifier’s value field contains a UTF-8 string representation of the identity of the device. In addition to the value being a valid UTF-8 string, the value MUST match the permanent-identifier-value production rule as defined in this ABNF [RFC5234] syntax:

```
assigner-value = ("0" / "1" / "2")  1*("." 1*DIGIT)
device-identifier-value = 1*(%x00-2E / %x30-FF)

permanent-identifier-value = device-identifier-value ["/" assigner-value]
```

A valid permanent-identifier-value value is a UTF-8 string that contains an identity consisting of one or more characters without any forward-slash "/" (UTF-8: U+002F) characters. Optionally, a forward-slash "/" character and "dotted-decimal" object identifier identifying the assigner may follow the identity. The assigner-value is a dotted-decimal representation of an ASN.1 OBJECT IDENTIFIER.

The Server MUST verify that identifier values in newOrder requests conform to the permanent-identifier-value production rule and MUST reject requests containing non-conforming values with a "malformed" error.

Example identifier without an assigner:

```
{
  "type": "permanent-identifier",
  "value": "ABCDEF123456"
}
```

Example identifier with an assigner:

```
{
  "type": "permanent-identifier",
  "value": "ABCDEF123456/1.2.3.4"
}
```

3.2. Representation in Certificate Signing Requests (CSRs) and X.509 Certificates

This section describes the X.509 representation of the permanent-identifier. Other credential types may use the same identifier values with representations appropriate to those credential types.

The identity is included in the Subject Alternative Name Extension using the identifierValue field of the PermanentIdentifier form described in [RFC4043]. Although [RFC4043] permits the requester to include the identifierValue in a serialNumber subject attribute, this specification mandates that the identifierValue field of the PermanentIdentifier MUST be present and MUST contain the identifier.

The value of the identifierValue field of the PermanentIdentifier MUST be an octet-for-octet match of the device-identifier-value value as encoded in the Order resource. If the assigner-value value is included in the identifier as encoded in the Order resource, then the assigner field of the PermanentIdentifier MUST be the encoding of the "dotted-decimal" object identifier encoded as the assigner-value value.

This strict matching requirement ensures that the SAN in the issued certificate appears exactly as it appeared during proof-of-ownership validation, preventing identifier malleability. Serial number allocation schemes may be case-sensitive or otherwise sensitive to exact byte representation, so no normalization or transformation is permitted.

To ensure that the identifier as presented in the Order resource and CSR match, the Server MUST perform the logical equivalent of extracting the device-identifier-value and assigner-value values from the CSR and reconstructing the UTF-8 representation of the identifier. The Server MUST then ensure that the UTF-8 representation and the identifier presented in the Order resource are an octet-for-octet match and reject the Order otherwise. Servers that derive identifier values directly from verified attestation evidence and construct the certificate SAN from that evidence, provided the derived values are verified against the attested device identity in the attestation statement, satisfy the intent of this requirement.

[RFC8555] section 7.4 mandates that "The CSR MUST indicate the exact same set of requested identifiers as the initial newOrder request". However, there are some environments where the Server requires validation of the identifier but does not include the identifier in certificates due to privacy concerns. To support privacy-preserving certificates, Clients MAY omit this identifier in the certificate signing request (CSR). Similarly, if the Server wishes to issue privacy-preserving certificates, it MAY reject CSRs containing a PermanentIdentifier in the subjectAltName extension. See the Section 7 for more information.

4. Hardware Module

A new identifier type, hardware-module is introduced to represent the identity of the secure crypto-processor that generated the certificate key. The identity is modeled after the HardwareModuleName form described in [RFC4108]. It consists of two components: an OBJECT IDENTIFIER to represent the type of hardware module, and a serial number that identifies the specific hardware module.

Although [RFC4108] specifies that serial numbers can be represented as any sequence of bytes, this specification requires that serial numbers MUST be representable as valid UTF-8 strings consisting of at least one code point and MUST NOT contain a forward-slash "/" (UTF-8: U+002F) character. This restriction ensures that serial numbers can be included in hardware-module identifier string values and that the ABNF production rule for the value is unambiguous.

4.1. Representation in Order resources

The identifier’s value field contains a UTF-8 string representation of the identity of the hardware module. In addition to the value being a valid UTF-8 string, the value MUST match the hardware-module-value production rule as defined in this ABNF [RFC5234] syntax:

```
hw-type-value = ("0" / "1" / "2")  1*("." 1*DIGIT)
hw-serial-num-value = 1*(%x00-2E / %x30-FF)

hardware-module-value = hw-serial-num-value ["/" hw-type-value]
```

A valid hardware-module-value value is a UTF-8 string that contains a serial number consisting of one or more characters without any forward-slash "/" (UTF-8: U+002F) characters. Optionally, a forward-slash "/" character and "dotted-decimal" object identifier identifying the hardware type may follow the serial number.

The Server MUST verify that identifier values in newOrder requests conform to the hardware-module-value production rule and MUST reject requests containing non-conforming values with a "malformed" error.

Example identifier with the type of the hardware module represented using the OBJECT IDENTIFIER "1.2.3.4" and a serial number of "ABCD":

```
{
  "type": ‘hardware-module‘,
  "value": "ABCD/1.2.3.4"
}
```

Example identifier with no type specified and a serial number of "ABCD":

```
{
  "type": ‘hardware-module‘,
  "value": "ABCD"
}
```

4.2. Representation in Certificate Signing Requests and X.509 Certificates

This section describes the X.509 representation of the hardware-module identifier. Other credential types may use the same identifier values with representations appropriate to those credential types.

The hardware module identity is 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

The value of the hwSerialNum field of the HardwareModuleName MUST be an octet-for-octet match of the hw-serial-num-value value as encoded in the Order resource. If the hw-type-value value is included in the identifier as encoded in the Order resource, then the hwType field of the HardwareModuleName MUST be the encoding of the "dotted-decimal" object identifier encoded as the hw-type-value value.

This strict matching requirement ensures that the SAN in the issued certificate appears exactly as it appeared during proof-of-ownership validation, preventing identifier malleability. Serial number allocation schemes may be case-sensitive or otherwise sensitive to exact byte representation, so no normalization or transformation is permitted.

To ensure that the identifier as presented in the Order resource and CSR match, the Server MUST perform the logical equivalent of extracting the hw-serial-num-value and hw-type-value values from the CSR and reconstructing the UTF-8 representation of the identifier. The Server MUST then ensure that the UTF-8 representation and the identifier presented in the Order resource are an octet-for-octet match and reject the Order otherwise. Servers that derive identifier values directly from verified attestation evidence and construct the certificate SAN from that evidence, provided the derived values are verified against the attested device identity in the attestation statement, satisfy the intent of this requirement.

[RFC8555] section 7.4 mandates that "The CSR MUST indicate the exact same set of requested identifiers as the initial newOrder request". However, there are some environments where the Server requires validation of the identifier but does not include the identifier in certificates due to privacy concerns. To support privacy-preserving certificates, Clients MAY omit this identifier in the certificate signing request (CSR). Similarly, if the Server wishes to issue privacy-preserving certificates, it MAY reject CSRs containing a HardwareModuleName in the subjectAltName extension. See the Section 7 for more information.

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.

```
{
  "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.

* Some attestation formats use an external attestation authority that issues a certificate binding the challenge to the device before the Client's account key is available. In these formats, _attToBeSigned_ is formed from the token alone rather than the full key authorization, because the external authority signs at attestation time before the account key thumbprint can be incorporated. The token construction provides freshness. The key authorization construction additionally binds the attestation to a specific account key. The Server MUST consult format-specific documentation to determine which construction applies and MUST verify accordingly. Attestation formats whose signing procedure does not incorporate _attToBeSigned_ cannot be used to satisfy this challenge type.

* The _authData_ field carries browser-context data (including the RP ID hash) that has no meaning in the ACME context 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. Clients MAY include additional fields beyond "attObj" in the response object. Servers MUST ignore unrecognized fields in the challenge response.

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 _attToBeSigned_ contains the key authorization or the token, according to the construction required by the attestation format, and that the value matches what the Server stored.

3. Verify that the attestation statement contains a device identifier and that it matches the identifier in the Order. The means by which the identifier is encoded in the attestation statement are specific to the attestation format.

```
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. Operational Considerations

Although this document focuses guidance on implementing new identifier types and a challenge for certificate issuance using ACME, it does not define a new protocol, a protocol extension, or an architecture.

6.1. Enterprise PKI

ACME was originally envisioned for issuing certificates in the Web PKI, however this extension will primarily be useful in enterprise PKI.

6.1.1. 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].

6.1.2. Attestation Posture

Enterprise deployments often consist of heterogeneous device fleets where not all devices are capable of hardware attestation. A Server MAY offer device-attest-01 alongside other challenge types within a single authorization, allowing capable devices to complete device-attest-01 while other devices complete an alternative challenge. This posture allows operators to observe fleet attestation coverage before enforcing policy and is compatible with phased deployments.

Servers MAY rely on other authorization mechanisms, such as external account binding or pre-authorized accounts, to establish device identity instead of completing the device-attest-01 challenge.

6.1.3. Multiple Challenge Types

[RFC8555] permits a Server to offer multiple challenge types within a single authorization, with any one being sufficient to complete it. Servers MAY offer device-attest-01 alongside other challenge types for the same authorization, allowing capable devices to attest while other devices use an alternative challenge type.

6.2. Attestation Trust

Attestation formats differ in the authority that enforces the boundary around the attested key and in the claims that authority can make. At one end, dedicated security hardware (such as a TPM or HSM) provides manufacturer-backed guarantees that the key is generated and stored within the hardware and cannot be exported. At the other end, OS-enforced isolation boundaries (such as platform keystores protected by the operating system kernel) provide meaningful key protection guarantees without discrete security hardware. Intermediate cases include TEE-based attestation where a hypervisor or trusted execution environment acts as the authority.

Servers SHOULD consider the trust properties of each attestation format when establishing issuance policy, including the nature of the authority making the attestation and the key protection guarantees it can assert. The key authorization construction described in Section 5 also contributes to this trust model: formats that use the full key authorization as _attToBeSigned_ bind the attestation to a specific account key, while formats that use the token alone provide freshness without account key binding.

7. Privacy Considerations

This section analyzes the privacy implications of the permanent-identifier and hardware-module identifier types introduced in this document. The guidance here is informed by the threat taxonomy defined in [RFC6973] and is intended to help implementers make informed decisions about whether and when to include these identifiers in certificate requests and issued certificates.

Both identifier types represent unchanging hardware-bound properties of a device. Unlike domain names or other identifiers whose lifetime is bounded by operational changes, these identifiers typically persist across the entire operational life of a device and cannot be rotated or revoked by the device owner. This permanence has material privacy consequences that implementers must weigh carefully.

The privacy analysis below addresses the two phases in which these identifiers appear: the attestation exchange between the Client and server during challenge validation, and the optional embedding of identifiers in the issued certificate.

7.1. Identification and Correlation

The permanent-identifier type encodes a manufacturer assigned device identity, typically a serial number. The hardware-module type encodes the identity of the secure cryptoprocessor that generated the certificate key. In both cases, the identifier is globally unique within its assigner scope and unchanging for the lifetime of the device or hardware module.

From the perspective of [RFC6973] Section 5.2.2, such identifiers enable direct identification of a device across protocol interactions, deployments, and time. Any entity that receives or observes these identifiers, including the server, intermediary infrastructure, and any relying party that processes the issued certificate acquires an observable reference that can be used to track the device’s certificate issuance history, renewal patterns, and operational context.

When the same permanent-identifier or hardware-module value appears across multiple certificate requests (as it will in any recurring renewal workflow), it enables [RFC6973] correlation: an observer with access to server logs can reconstruct the full lifecycle of a device’s certificate activity. Similarly, when such identifiers are included in issued certificates, logging issued certificates in a central location (in certificate transparency logs, etc.) produces a persistent device audit trail regardless of whether the log operator intends to maintain one.

Implementers SHOULD assess whether the operational benefit of unchanging device identification outweighs this correlation exposure. In deployments where device anonymity or pseudonymity is a requirement, such as systems handling sensitive workloads on behalf of individuals, implementers SHOULD consider whether alternative validation mechanisms that do not bind the certificate to a permanent hardware identifier are more appropriate.

7.2. Fingerprinting via Attestation Payloads

The device-attest-01 challenge response carries a WebAuthn attestation object that may contain significantly more information than the identifier value alone. Depending on the attestation format, this payload may include device model, firmware version, bootloader state, hardware security level, and operating system version. Even when the resulting certificate is issued in a privacy-preserving form that omits the identifier from the subjectAltName extension (see Section 3.2 and Section 4.2), the attestation payload itself is transmitted to and evaluated by the server during challenge validation.

This constitutes a fingerprinting surface as defined in [RFC6973] Section 3.2. The combination of a hardware serial number, hardware type OID, and firmware attestation attributes may uniquely identify not just the device model but the specific device unit, even in the absence of an explicit permanent-identifier value. Implementers operating servers may consider applying data minimization principles to attestation payload handling by limiting only the attributes necessary to make the authorization decision should be evaluated, and the full attestation payload SHOULD NOT be retained beyond the duration of the challenge validation exchange unless there is a specific, documented operational requirement to do so.

Implementers operating ACME Clients SHOULD be aware that the attestation format selected may expose more device state than is necessary to satisfy the server’s authorization policy. Where multiple attestation formats are available, Clients SHOULD prefer formats that minimize the set of disclosed attributes.

7.3. Secondary Use of Attestation Data

The server receives attestation data in the context of authorizing a certificate issuance request. [RFC6973] Section 5.2.3 identifies secondary use as the processing of data for purposes beyond the original collection context and as a distinct privacy threat.

Attestation payloads received during challenge validation may contain information about device health, software configuration, and hardware capability that is operationally useful beyond certificate issuance. For example, for asset inventory, compliance monitoring, or security posture assessment. Implementers operating servers must clearly define and document the purposes for which attestation data is processed and must not process attestation data for purposes materially different from authorization of the certificate request without explicit policy disclosure to the device owner or operator.

Implementers integrating ACME device attestation into enterprise PKI platforms should publish a clear attestation data handling policy that specifies what attributes are evaluated, how long they are retained, and whether they are shared with other systems.

7.4. Privacy-Preserving Certificate Issuance

This document provides an explicit mechanism to decouple attestation-based validation from identifier disclosure in the issued certificate. Clients MAY omit the permanent-identifier or hardware-module from the CSR, and servers MAY issue certificates that do not contain these identifiers in the subjectAltName extension, even when those identifiers were used to authorize the request.

Implementers should treat this privacy-preserving mode as the default posture unless there is a specific operational requirement for the identifier to appear in the certificate. The following considerations apply to this decision:

* If the issued certificate will be presented to relying parties outside the issuing organization's trust boundary, embedding a permanent-identifier or hardware-module value in the certificate enables those relying parties to correlate certificate presentations with specific physical hardware. This may be acceptable in closed enterprise environments but is likely inappropriate in any context where the certificate is presented to external services, counter-parties, or public infrastructure.

* If the certificate is used for mutual TLS in a workload identity context, embedding an unchanging hardware identifier couples the cryptographic identity of the workload to the physical device rather than to the logical identity of the workload. This can impede key rotation, device replacement, and workload migration, in addition to creating the correlation risks described above. In such cases, implementers should prefer logical workload identifiers (such as SPIFFE URIs) in the issued certificate and treat the hardware attestation as a bootstrap authorization mechanism only.

* If the certificate is intended for use in certificate transparency logs, implementers MUST consider that embedding a permanent-identifier or hardware-module value will make that identifier permanently and publicly discoverable, indexed by issuance time, issuer, and subject. This constitutes an irreversible disclosure under [RFC6973] Section 5.2.4 and should be avoided unless public discoverability of the device identifier is an explicit operational requirement.

7.5. Stored Data and Account Binding

This document recommends the use of externalAccountBinding to pre-authenticate device requests to an enterprise server. When an ACME account is persistently bound to a device identity, the server's account store contains a durable mapping between the cryptographic account credential and the physical device. Per [RFC6973] Section 5.1.2, this stored association constitutes a target for compromise: an attacker who obtains the account store gains not only account credentials but a historical record of device-to-identity mappings across all certificate issuances.

Implementers operating servers SHOULD store account-to-device bindings using the minimum fidelity necessary for authorization decisions. Where the operational requirement is only to confirm that a given device is authorized to request certificates, it may be sufficient to store a hash or other one-way transformation of the device identifier rather than the identifier itself. Implementers should also define and enforce retention limits on historical account-to-certificate linkage records.

7.6. Implementer Decision Guidance

Implementers considering whether to include permanent-identifier or hardware-module in CSRs and issued certificates SHOULD work through the following questions before enabling these identifiers:

* Is unchanging hardware identity in the certificate necessary for the relying party to make authorization decisions, or is it sufficient for the server to have validated it at issuance time? If the latter, prefer privacy-preserving certificate mode.

* Will the certificate be logged to a certificate transparency log or otherwise made publicly accessible? If so, embedding a permanent hardware identifier creates an irrevocable, publicly indexed disclosure and should be avoided unless explicitly required.

* Will the certificate be presented to parties outside the issuing organization’s administrative control? If so, consider whether those parties should have visibility into the device’s hardware identity.

* Does the deployment have requirements for device replacement or key rotation without service interruption? Binding the certificate’s identity to a specific hardware module OID and serial number complicates these operational scenarios and may require reissuance policies that expose additional identifier churn in logs.

* What is the attestation data handling policy of the server operator? If this is not documented or auditable, device operators SHOULD treat the attestation exchange as a full disclosure of all attributes present in the attestation payload.

8. Security Considerations

Please reference [RFC8555] for other 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.

The "token" value MUST have at least 128 bits of entropy. It MUST NOT contain any characters outside the base64url alphabet, including padding characters ("="). See [I-D.ietf-tls-rfc8446bis], Appendix C.1 for additional information on randomness requirements.

The binding between the certified public key and the device identifier is established through the attestation statement rather than through the CSR alone. The attestation authority cryptographically binds the public key to the device identity, either by signing the attestation statement directly, by issuing an attestation certificate, or by other cryptographic means specific to the attestation format. The Server verifies this chain: the attestation is produced by a trusted attestation authority, the public key in the attestation matches the public key in the CSR, and the device identifier in the attestation matches the identifier in the Order. This three-way binding is the basis on which the Server can associate a certified public key with a particular device.

9. IANA Considerations

9.1. 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

9.2. ACME Validation Method

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

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

Table 2

9.3. 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

10. References

10.1. Normative References

[I-D.ietf-tls-rfc8446bis]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8446bis-14, 13 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-rfc8446bis-14>.

[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>.

[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>.

[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>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/rfc/rfc6973>.

[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/>.

10.2. Informative References

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

Acknowledgments

We thank the participants on the ACME Working Group mailing list for their insightful feedback and comments. In particular, the authors extend sincere appreciation to Mike Ounsworth, Deb Cooley, Aaron Gable, Richard Barnes, and Herman Slatman for their reviews and suggestions, which greatly improved the quality of this document.

Authors' Addresses

Brandon Weeks
Google Inc
Email: me@brandonweeks.com

Ganesh Mallaya
AppViewX Inc.
Email: ganesh.mallaya@appviewx.com

Sven Rajala
Keyfactor
Email: sven.rajala@keyfactor.com

Corey Bonnell
DigiCert, Inc.
Email: corey.bonnell@digicert.com

Ryan Hurst
Peculiar Ventures
Email: ryan.hurst+ietf@peculiarventures.com