Stream: Internet Engineering Task Force (IETF)
RFC: 9908
Updates: 7030, 9148
Category: Standards Track
Published: January 2026
ISSN: 2070-1721
Authors:
M. Richardson, Ed. — *Sandelman Software Works*
O. Friel — *Cisco*
D. von Oheimb — *Siemens*
D. Harkins — *The Industrial Lounge*

RFC 9908
Clarification and Enhancement of the CSR Attributes Definition in RFC 7030

Abstract

This document updates RFC 7030, "Enrollment over Secure Transport" (EST), clarifying how the Certificate Signing Request (CSR) Attributes Response can be used by an EST server to specify both CSR attribute Object Identifiers (OIDs) and CSR attribute values, particularly X.509 extension values, that the server expects the client to include in a subsequent CSR request. RFC 9148 is derived from RFC 7030 and is also updated.

RFC 7030 is ambiguous in its specification of the CSR Attributes Response. This has resulted in implementation challenges and implementor confusion because there was no universal understanding of what was specified. This document clarifies the encoding rules.

This document also provides a new straightforward approach: using a template for CSR contents that may be partially filled in by the server. This also allows an EST server to specify a subject Distinguished Name (DN).

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9908.

Copyright Notice

Table of Contents

1. Introduction

This document updates RFC 7030 and clarifies how the Certificate Signing Request (CSR) Attributes Response can be used by an EST server to specify both CSR attribute OIDs and CSR attribute values. In particular, the server needs to be able to specify X.509 extension values that it expects the client to include in the subsequent CSR.

"Enrollment over Secure Transport" [RFC7030] has been used in a wide variety of applications. In particular, [RFC8994] and [RFC8995] describe a way to use it in order to build out an Autonomic Control Plane (ACP) [RFC8368].

When bootstrapping the ACP, there is a requirement that each node be given a very specific subjectAltName. In [RFC8994], the ACP specification, the EST server is specified to make use of the CSR Attributes ("/csrattrs") Request (specified in [RFC7030], Section 2.6) to convey the actual subjectAltName to the EST client that needs to go into its CSR and thus ultimately into its End Entity (EE) certificate.

As a result of some implementation challenges, it came to light that this particular way of using the CSR Attributes was not universally agreed upon, and it was suggested that it went contrary to [RFC7030], Section 2.6.

[RFC7030], Section 2.6 says that the CSR Attributes "can provide additional descriptive information that the EST server cannot access itself". This is extended to describe how the EST server can provide values that it demands be used.

After significant discussion, it has been determined that Section 4.5 of [RFC7030] is sufficiently difficult to read and ambiguous to interpret, so clarification is needed.

Also, [RFC7030], Section 4.5.2 is extended to clarify the use of the existing ASN.1 syntax [X.680] [X.690].

This covers all uses and is fully backward compatible with existing use, including addressing the needs of [RFC8994] and [RFC8995].

2. Terminology

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. CSR Attributes Handling

3.1. Extensions to RFC 7030, Section 2.6

This document replaces the second paragraph in Section 2.6 of [RFC7030] with the following text:

> These attributes can provide additional information that the EST server cannot access itself, such as the Media Access Control (MAC) address of an interface of the EST client. The EST server can also provide concrete values that it tells the client to include in the CSR, such as a specific X.509 Subject Alternative Name extension. Moreover, these attributes can indicate the type of the included public key or which crypto algorithms to use for the self-signature, such as a specific elliptic curve or a specific hash function that the client is expected to use when generating the CSR.

3.2. Extensions to RFC 7030, Section 4.5.2

The ASN.1 syntax for CSR Attributes as defined in EST ([RFC7030], Section 4.5.2) is as follows:

```
CsrAttrs ::= SEQUENCE SIZE (0..MAX) OF AttrOrOID

AttrOrOID ::= CHOICE (oid OBJECT IDENTIFIER, attribute Attribute }

Attribute { ATTRIBUTE:IOSet } ::= SEQUENCE {
     type   ATTRIBUTE.&id({IOSet}),
     values SET SIZE(1..MAX) OF ATTRIBUTE.&Type({IOSet}{@type}) }
```

This remains unchanged, such that bits-on-the-wire compatibility is maintained.

Key parts that were unclear were which OID to use in the '`type`' field and that the '`values`' field can contain an entire sequence of X.509 extensions.

The OID to use for such attributes in the '`type`' field **MUST** be `id-ExtensionReq`, which has the value 1.2.840.113549.1.9.14. Note that this is the same as `pkcs-9-at-extensionRequest` defined in PKCS#9 [RFC2985]. There **MUST** be only one such attribute.

The '`values`' field of this attribute **MUST** contain a set with exactly one element, and this element **MUST** be of type `Extensions`, as per Section 4.1 of [RFC5280]:

```
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID
     }
```

An Extension comprises the OID of the specific X.509 extension (extnID), optionally the 'critical' bit, and the extension value (extnValue).

An Extensions structure, which is a sequence of elements of type Extension, **MUST NOT** include more than one element with a particular extnID.

When not using the template-based approach of Section 3.4, specifying the requirement for a public key of a specific type and optionally its size and other parameters **MUST** be done as follows: Include exactly one Attribute with the `type` field being the OID specifying the type of the key, such as `ecPublicKey` or `rsaEncryption`. The '`values`' field **MAY** be empty to indicate no further requirements on the key. Otherwise, it **MUST** contain suitable parameters for the given key type, such as a singleton set containing the OID of an elliptic curve (EC) (e.g., `secp384r1`) or containing an integer value for the RSA key size (e.g., 4096). Many examples for this are given in Section 5.

3.3. Update to RFC 9148

The updates to EST in this document equally apply when using the Constrained Application Protocol (CoAP) as a transport mechanism as described in [RFC9148]. This document therefore adds the following paragraph after the second paragraph of [RFC9148], Section 1:

> EST over CoAP as specified in [RFC9148] applies unchanged to [RFC7030], which is updated by RFC 9908. Hence, all references to [RFC7030] in [RFC9148] are assumed to indicate that [RFC7030] is updated by RFC 9908.

3.4. Use of CSR Templates

As an alternative to the unstructured inclusion of CSR Attributes specified in Section 4.5.2 of [RFC7030] with its limitations and ambiguities, Appendix B of [RFC8295] describes an approach using a CSR template. An entire CSR object is returned with various fields filled out and other fields waiting to be filled in. In that approach, a pKCS7PDU attribute includes a PKIData content type [RFC5272] and that, in turn, includes a CSR [RFC2986] or a Certificate Request Message Format (CRMF) formatted request (for details, see 5 or 9 of [RFC6268], respectively).

One drawback to that approach, particularly for the CSR, is that some unused fields have to be included; specifically, the '`signature`' field on the CSR is faked with an empty bit string.

A similar method has been defined in "Internet X.509 Public Key Infrastructure -- Certificate Management Protocol (CMP)" [RFC9810] and "Lightweight Certificate Management Protocol (CMP) Profile" ([RFC9483], Section 4.3.3) using a CSR template as defined for CRMF [RFC4211]. Like the approach mentioned before, this method does not properly deal with absent Relative Distinguished Name (RDN) values, as it would encode them as invalid empty strings. Also, encoding an absent '`subjectPublicKey`' value as an empty `BIT STRING` and an absent X.509 extension value as an empty `OCTET STRING` can cause issues with strict ASN.1 parsing and decoding.

These drawbacks are avoided as follows:

This specification defines a new Certificate Request Information Template attribute for `CsrAttrs` (as given in Section 3.2) that is essentially a partially filled-in PKCS#10 CSR minus the signature wrapper:

```
CertificationRequestInfoTemplate ::= SEQUENCE {
    version       INTEGER { v1(0) } (v1, ... ),
    subject       NameTemplate OPTIONAL,
    subjectPKInfo [0] SubjectPublicKeyInfoTemplate
                              {{ PKInfoAlgorithms }} OPTIONAL,
    attributes    [1] Attributes{{ CRIAttributes }}
}
```

Appendix A contains all details.

The CertificationRequestInfoTemplate closely resembles the CertificationRequestInfo from [RFC5912], Section 5:

```
CertificationRequestInfo ::= SEQUENCE {
  version       INTEGER { v1(0) } (v1,...),
  subject       Name,
  subjectPKInfo SubjectPublicKeyInfo{{ PKInfoAlgorithms }},
  attributes    [0] Attributes{{ CRIAttributes }}
}
```

with the following differences:

- The 'subject' field has been made **OPTIONAL**. It **MUST** be present if the server places any requirements on the RDNs of the subject name; otherwise, it **MUST** be absent.
- RDNs in the 'subject' fields are allowed to have no value, which has been achieved by adding **OPTIONAL** to the 'value' field of SingleAttributeTemplate. If the client is expected to provide an RDN of a certain type such as commonName, the respective RDN **MUST** be present in the 'subject' field; otherwise, it **MUST** be absent. In addition, if the server gives an RDN value, this means that the client is expected to use this value for the RDN; otherwise, the client is expected to fill in a suitable value. The example at the end of this section has a 'subject' field that contains both forms of RDN specifications.

```
SingleAttributeTemplate {ATTRIBUTE:AttrSet} ::= SEQUENCE {
    type      ATTRIBUTE.&id({AttrSet}),
    value     ATTRIBUTE.&Type({AttrSet}{@type}) OPTIONAL
}
```

- The 'subjectPKInfo' field has been made **OPTIONAL**. The field **MUST** be absent if the server places no requirements on the key; otherwise, it **MUST** be present, and the 'algorithm' field specifies the type of key pair the client is expected to use.
- The 'subjectPublicKey' field contained in SubjectPublicKeyInfoTemplate has been made **OPTIONAL** because it is usually not needed. In case the server requires use of an RSA key and needs to specify its size, the field **MUST** be present and contain a placeholder public key value of the desired RSA modulus length; otherwise, the subjectPublicKey **MUST** be absent.

```
SubjectPublicKeyInfoTemplate{PUBLIC-KEY:IOSet} ::= SEQUENCE {
    algorithm        AlgorithmIdentifier{PUBLIC-KEY, {IOSet}},
    subjectPublicKey BIT STRING OPTIONAL
}
```

- A new OID `id-aa-extensionReqTemplate` and the related `ExtensionTemplate` structure is defined where the '`extnValue`' field has been made **`OPTIONAL`**. This is only needed to enable specifying partial extensions with values to be filled in by the client; otherwise, the `id-ExtensionReq` OID and the respective value of type `ExtensionReq` **MUST** be used for specifying requirements on X.509 extensions.

For each extension of type `Extension` or `ExtensionTemplate` provided by the server, the client is expected to include an extension of the type given by the `extnID`. If the '`critical`' field is present, the client **SHOULD** use it in the extension as well. If the '`extnValue`' is present (which is always the case when type `Extension` is used), the client **SHOULD** use the given extension value in its CSR. When the type `ExtensionTemplate` is used, the '`extnValue`' can be absent, and then the client **SHOULD** provide an extension value in an `Extension` with the given `extnID`. For instance, if the server includes an `ExtensionTemplate` with the `extnID` '`subjectAltName`' but without an `extnValue`, the client **SHOULD** include a SAN extension with a suitable value.

In case the server includes an `ExtensionTemplate` with the `extnID` '`subjectAltName`' and a partially filled-in `extnValue`, such as a '`directoryName`' choice containing the `NULL-DN` (i.e., an empty sequence of RDNs) or the '`iPAddress`' choice with an empty `OCTET STRING`, it means that the client **SHOULD** fill in the respective `GeneralName` value.

```
ExtensionTemplate {EXTENSION:ExtensionSet} ::= SEQUENCE {
   extnID      EXTENSION.&id({ExtensionSet}),
   critical    BOOLEAN DEFAULT FALSE,
   extnValue   OCTET STRING (CONTAINING
               EXTENSION.&ExtnType({ExtensionSet}{@extnID}) OPTIONAL
               --  contains the DER encoding of the ASN.1 value
               --  corresponding to the extension type identified
               --  by extnID when present
  }
```

The '`version`' field of the `CertificationRequestInfoTemplate` **MUST** contain v1 (0).

The '`attributes`' field **MUST NOT** contain multiple `id-aa-extensionReqTemplate` attributes and **MUST NOT** contain both `id-ExtensionReq` and `id-aa-extensionReqTemplate` attributes.

The '`values`' field of an `id-aa-extensionReqTemplate` attribute **MUST** contain a set with exactly one element, and this element **MUST** be of type `ExtensionTemplate`.

Suppose the server requires that the CSR will contain:

- the '`subject`' field with a common name to be filled in by the EE and two organizational unit names with given values "myDept" and "myGroup",

- the '`publicKey`' field with an Elliptic Curve Cryptography (ECC) public key on curve `secp256r1`,
- the 'subjectAltName' extension with two entries; one DNS entry with name "www.myServer.com" and IP entry that is empty for the IP address to be filled in,
- the '`keyUsage`' extension marked critical with the values digitalSignature and keyAgreement, and
- the '`extKeyUsage`' extension with the value to be filled in by the EE.

Then, the `CertificationRequestInfo` structure constructed by the server will be as follows:

```
SEQUENCE {
  INTEGER 0
  SEQUENCE {
    SET {
      SEQUENCE {
        OBJECT IDENTIFIER commonName (2 5 4 3)
        }
      }
    SET {
      SEQUENCE {
        OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
        UTF8String "myDept"
        }
      }
    SET {
      SEQUENCE {
        OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
        UTF8String "myGroup"
        }
      }
    }
  [0] {
    SEQUENCE {
      OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
      OBJECT IDENTIFIER secp256r1 (1 2 840 10045 3 1 7)
      }
    }
  [1] {
    SEQUENCE {
      OBJECT IDENTIFIER id-aa-extensionReqTemplate
                        (1 2 840 113549 1 9 62)
      SET {
        SEQUENCE {
          SEQUENCE {
            OBJECT IDENTIFIER subjectAltName (2 5 29 17)
            OCTET STRING, encapsulates {
              SEQUENCE {
                [2] "www.myServer.com"
                [7] ""
                }
              }
            }
          SEQUENCE {
            OBJECT IDENTIFIER keyUsage (2 5 29 15)
```

```
            BOOLEAN TRUE
            OCTET STRING, encapsulates {
              BIT STRING 3 unused bits
                "10001"B
              }
            }
          SEQUENCE {
            OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
            }
          }
        }
      }
    }
  }
```

4. Coexistence with Existing Implementations

EST servers with legacy clients **MAY** continue to use the unstructured list of attribute/value pairs as described in [RFC7030] and **MAY** also include the template style described in Section 3.4 for newer clients. Clients that understand both **MUST** use the template only, and ignore all other `CsrAttrs` elements. Older clients will ignore the new CertificationRequestInfoTemplate element.

5. Examples Using the Original Approach in RFC 7030

Each example has a high-level (English) explanation of what is expected. Some mapping back to the Attribute and Extension definitions above are included. The base64 DER encoding is then shown. The output of "dumpasn1" [dumpasn1] is then provided to detail what the contents are.

5.1. Require an RFC 8994 / ACP subjectAltName with Specific otherName

A single subjectAltName extension is specified in a single `CsrAttrs` attribute [RFC7030] with an OID '`id-ExtensionReq`' indicating type `Extensions`. This is what might be created by a Registrar [RFC8995] that is asking for AcpNodeName [RFC8994] with format '`otherNames`'.

5.1.1. Base64-Encoded Example

The Base64:

```
MGgwZgYJKoZIhvcNAQkOMVkwVzBVBgNVHREBAf8ESzBJoEcG
CCsGAQUFBwgKoDsWOXJmYzg5OTQrZmQ3MzlmYzIzYzM0NDAx
MTIyMzM0NDU1MDAwMDAwMDArQGFjcC5leGFtcGxlLmNvbQ==
```

5.1.2. ASN.1 DUMP Output

There is a single subjectAltName Extension with an Attribute with Extension type.

```
   <30 68>
 0 104: SEQUENCE {
   <30 66>
 2 102:   SEQUENCE {
   <06 09>
 4   9:     OBJECT IDENTIFIER
      :       extensionRequest (1 2 840 113549 1 9 14)
      :       (PKCS #9 via CRMF)
   <31 59>
15  89:     SET {
   <30 57>
17  87:       SEQUENCE {
   <30 55>
19  85:         SEQUENCE {
   <06 03>
21   3:           OBJECT IDENTIFIER
      :             subjectAltName (2 5 29 17)
      :             (X.509 extension)
   <01 01>
26   1:           BOOLEAN TRUE
   <04 4B>
29  75:           OCTET STRING, encapsulates {
   <30 49>
31  73:             SEQUENCE {
   <A0 47>
33  71:               [0] {
   <06 08>
35   8:                 OBJECT IDENTIFIER '1 3 6 1 5 5 7 8 10'
   <A0 3B>
45  59:                 [0] {
   <16 39>
47  57:                   IA5String
      :                   'rfc8994+fd739fc23c34401122334455'
      :                   '00000000+@acp.example.com'
      :                   }
      :                 }
      :               }
      :             }
      :           }
      :         }
      :       }
      :     }
      :   }
```

5.2. Original Example in RFC 7030

In this example, taken from Section 4.5.2 of [RFC7030], a few different attributes are included. The original encoding of the '`macAddress`' part in the example is NOT CORRECT. It was not aligned with the definition of the Extension Request attribute as specified in Section 5.4.2 of [RFC2985]. The revised encoding given here does not use an '`id-ExtensionReq`' attribute because the MAC Address is not an X.509 certificate extension by itself and because the server provides its OID without a value, which is not allowed syntactically within a structure of type '`Extension`'.

5.2.1. Base64-Encoded Example

The Base64:

```
MDIGCSqGSIb3DQEJBzASBgcqhkjOPQIBMQcGBSuBBAAiBgcr
BgEBAQEWBggqhkjOPQQDAw==
```

5.2.2. ASN.1 DUMP Output

The CsrAttrs structure contains:

1. The challengePassword attribute to indicate that the CSR should include this value.
2. An ecPublicKey OID with the value secp384r1 to indicate what kind of public key should be submitted.
3. The macAddress OID 1.3.6.1.1.1.1.22 to indicate that the CSR is expected to include (in a subjectDirectoryAttributes extension) a MAC address value.
4. The ecdsaWithSHA384 OID to indicate what kind of hash is expected to be used for the self-signature in the PKCS#10 CSR.

```
   <30 32>
 0  50: SEQUENCE {
   <06 09>
 2   9:   OBJECT IDENTIFIER challengePassword (1 2 840 113549 1 9 7)
     :     (PKCS #9)
   <30 12>
13  18:   SEQUENCE {
   <06 07>
15   7:     OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
     :       (ANSI X9.62 public key type)
   <31 07>
24   7:     SET {
   <06 05>
26   5:       OBJECT IDENTIFIER secp384r1 (1 3 132 0 34)
     :         (SECG (Certicom) named elliptic curve)
     :       }
     :     }
   <06 07>
33   7:   OBJECT IDENTIFIER '1 3 6 1 1 1 1 22'
   <06 08>
42   8:   OBJECT IDENTIFIER ecdsaWithSHA384 (1 2 840 10045 4 3 3)
     :     (ANSI X9.62 ECDSA algorithm with SHA384)
     :   }
```

5.3. Require a Specific subjectAltName Extension

This example is the same as the previous one except that instead of the OID for a macAddress, a subjectAltName is specified as the only Extension element.

5.3.1. Base64-Encoded Example

The Base64:

```
MEUGCSqGSIb3DQEJBzASBgcqhkjOPQIBMQcGBSuBBAAjBgkq
hkiG9w0BCRQGCgmSJomT8ixkAQUGA1UEBQYIKoZIzj0EAwQ=
```

5.3.2. ASN.1 DUMP Output

The CsrAttrs structure contains:

1. The challengePassword attribute to indicate that the CSR should include this value.
2. An ecPublicKey OID with the value secp521r1 to indicate what kind of public key should be submitted.
3. An extensionRequest container with a subjectAltName value containing the name potato@example.com.
4. The ecdsaWithSHA512 OID to indicate the SHA-512 hash is expected to be used for the self-signature in the PKCS#10 CSR.

```
   <30 45>
 0  69: SEQUENCE {
   <06 09>
 2   9:   OBJECT IDENTIFIER challengePassword (1 2 840 113549 1 9 7)
      :     (PKCS #9)
   <30 12>
13  18:   SEQUENCE {
   <06 07>
15   7:     OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
      :       (ANSI X9.62 public key type)
   <31 07>
24   7:     SET {
   <06 05>
26   5:       OBJECT IDENTIFIER secp521r1 (1 3 132 0 35)
      :         (SECG (Certicom) named elliptic curve)
      :       }
      :     }
   <06 09>
33   9:   OBJECT IDENTIFIER friendlyName (for PKCS #12)
                            (1 2 840 113549 1 9 20)
      :     (PKCS #9 via PKCS #12)
   <06 0A>
44  10:   OBJECT IDENTIFIER '0 9 2342 19200300 100 1 5'
   <06 03>
56   3:   OBJECT IDENTIFIER serialNumber (2 5 4 5)
      :     (X.520 DN component)
   <06 08>
61   8:   OBJECT IDENTIFIER ecdsaWithSHA512 (1 2 840 10045 4 3 4)
      :     (ANSI X9.62 ECDSA algorithm with SHA512)
      :   }
```

5.4. Require a Public Key of a Specific Size

The CSR requires an RSA public key of a specific size.

5.4.1. Base64-Encoded Example

The Base64:

```
MCkGCSqGSIb3DQEJBzARBgkqhkiG9w0BAQExBAICEAAGCSqG
SIb3DQEBCw==
```

5.4.2. ASN.1 DUMP Output

Provide a CSR with an RSA key that's 4096 bits and use SHA256 as the hash algorithm within the signature.

```
    <30 29>
  0  41: SEQUENCE {
    <06 09>
  2   9:   OBJECT IDENTIFIER challengePassword (1 2 840 113549 1 9 7)
       :     (PKCS #9)
    <30 11>
 13  17:   SEQUENCE {
    <06 09>
 15   9:     OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
       :       (PKCS #1)
    <31 04>
 26   4:     SET {
    <02 02>
 28   2:       INTEGER 4096
       :       }
       :     }
    <06 09>
 32   9:   OBJECT IDENTIFIER sha256WithRSAEncryption
                             (1 2 840 113549 1 1 11)
       :     (PKCS #1)
       :   }
```

5.5. Require a Public Key of a Specific Curve

The CSR requires an ECC public key with a specific curve.

5.5.1. Base64-Encoded Example

The Base64:

```
MC4GCSqGSIb3DQEJBzASBgcqhkjOPQIBMQcGBSuBBAAiBgNV
BAUGCCqGSM49BAMD
```

5.5.2. ASN.1 DUMP Output

Provide a CSR with an ECC public key from p384, include your serial number, and use SHA384 as the hash algorithm within the signature.

```
    <30 2E>
 0  46: SEQUENCE {
    <06 09>
 2   9:   OBJECT IDENTIFIER challengePassword (1 2 840 113549 1 9 7)
       :     (PKCS #9)
    <30 12>
13  18:   SEQUENCE {
    <06 07>
15   7:     OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
       :       (ANSI X9.62 public key type)
    <31 07>
24   7:     SET {
    <06 05>
26   5:       OBJECT IDENTIFIER secp384r1 (1 3 132 0 34)
       :         (SECG (Certicom) named elliptic curve)
       :       }
       :     }
    <06 03>
33   3:   OBJECT IDENTIFIER serialNumber (2 5 4 5)
       :     (X.520 DN component)
    <06 08>
38   8:   OBJECT IDENTIFIER ecdsaWithSHA384 (1 2 840 10045 4 3 3)
       :     (ANSI X9.62 ECDSA algorithm with SHA384)
       :   }
```

5.6. Require Specific Extensions and Attributes

The CSR is required to have an ECC public key, include a serial number, include a friendly name, include a favorite drink [favoritedrink] [OID 0.9.2342.19200300.100.1.5], and use SHA512 as the hash algorithm within the signature.

5.6.1. Base64-Encoded Example

The Base64:

```
MEUGCSqGSIb3DQEJBzASBgcqhkjOPQIBMQcGBSuBBAAjBgkq
hkiG9w0BCRQGCgmSJomT8ixkAQUGA1UEBQYIKoZIzj0EAwQ=
```

5.6.2. ASN.1 DUMP Output

Provide a CSR with an ECC public key from sha521 and include your serial number, friendly name, and favorite drink, and hash it with SHA512.

```
   <30 45>
 0  69: SEQUENCE {
   <06 09>
 2   9:   OBJECT IDENTIFIER challengePassword (1 2 840 113549 1 9 7)
      :     (PKCS #9)
   <30 12>
13  18:   SEQUENCE {
   <06 07>
15   7:     OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
      :       (ANSI X9.62 public key type)
   <31 07>
24   7:     SET {
   <06 05>
26   5:       OBJECT IDENTIFIER secp521r1 (1 3 132 0 35)
      :         (SECG (Certicom) named elliptic curve)
      :       }
      :     }
   <06 09>
33   9:   OBJECT IDENTIFIER friendlyName (for PKCS #12)
                            (1 2 840 113549 1 9 20)
      :     (PKCS #9 via PKCS #12)
   <06 0A>
44  10:   OBJECT IDENTIFIER '0 9 2342 19200300 100 1 5'
   <06 03>
56   3:   OBJECT IDENTIFIER serialNumber (2 5 4 5)
      :     (X.520 DN component)
   <06 08>
61   8:   OBJECT IDENTIFIER ecdsaWithSHA512 (1 2 840 10045 4 3 4)
      :     (ANSI X9.62 ECDSA algorithm with SHA512)
      :   }
```

6. Security Considerations

The security considerations from [RFC7030], Section 6 are unchanged.

6.1. Identity and Privacy Considerations

An EST server may use this mechanism to instruct the EST client about the identities it should include in the CSR it sends as part of enrollment. The client may only be aware of its Initial Device Identifier (IDevID) Subject, which includes a manufacturer serial number. The EST server can use this mechanism to tell the client to include a specific fully qualified domain name in the CSR in order to complete domain ownership proofs required by the CA. Additionally, the EST server may deem the manufacturer serial number in an IDevID as personally identifiable information and may want to specify a new random opaque identifier that the pledge should use in its CSR. This may be desirable if the CA and EST server have different operators.

7. IANA Considerations

For the ASN.1 module in Appendix A, IANA has assigned the following OID in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry:

Decimal	Description	Reference
82	id-mod-critemplate	RFC 9908

Table 1

For the Certification Request Information Template and Extension Request Template attributes in Appendix A, IANA has assigned the following OIDs in the "SMI Security for S/MIME Attributes (1.2.840.113549.1.9.16.2)" registry:

Decimal	Description	Reference
61	id-aa-certificationRequestInfoTemplate	RFC 9908
62	id-aa-extensionReqTemplate	RFC 9908

Table 2

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/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/info/rfc5911>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[RFC6268] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <https://www.rfc-editor.org/info/rfc6268>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

[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/info/rfc8174>.

[RFC9148] van der Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol", RFC 9148, DOI 10.17487/RFC9148, April 2022, <https://www.rfc-editor.org/info/rfc9148>.

[X.680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680>.

[X.690] ITU-T, "Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690>.

8.2. Informative References

[dumpasn1] Gutmann, P., "Dump ASN", 22 April 2021, <https://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c>.

[favoritedrink] OID Repository, "drink(5) [other identifier: favouriteDrink]", OID 0.9.2342.19200300.100.1.5, 4 July 2019, <https://oid-base.com/get/0.9.2342.19200300.100.1.5>.

[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/info/rfc2985>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <https://www.rfc-editor.org/info/rfc5272>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC8295] Turner, S., "EST (Enrollment over Secure Transport) Extensions", RFC 8295, DOI 10.17487/RFC8295, January 2018, <https://www.rfc-editor.org/info/rfc8295>.

[RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <https://www.rfc-editor.org/info/rfc8368>.

[RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <https://www.rfc-editor.org/info/rfc8994>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

[RFC9483] Brockhaus, H., von Oheimb, D., and S. Fries, "Lightweight Certificate Management Protocol (CMP) Profile", RFC 9483, DOI 10.17487/RFC9483, November 2023, <https://www.rfc-editor.org/info/rfc9483>.

[RFC9810] Brockhaus, H., von Oheimb, D., Ounsworth, M., and J. Gray, "Internet X.509 Public Key Infrastructure -- Certificate Management Protocol (CMP)", RFC 9810, DOI 10.17487/RFC9810, July 2025, <https://www.rfc-editor.org/info/rfc9810>.

Appendix A. ASN.1 Module

This appendix provides an ASN.1 module [X.680] for the Certification Request Information Template attribute and its sub-template structures. It follows the conventions established in [RFC5911], [RFC5912], and [RFC6268].

```
CRITemplateModule
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-9(9) smime(16) modules(0) id-mod-critemplate(82) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS -- from [RFC5912]

SupportedAttributes
 FROM PKIX1Explicit-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51)}

ATTRIBUTE, EXTENSION
 FROM PKIX-CommonTypes-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57) }

PUBLIC-KEY, AlgorithmIdentifier{}
 FROM AlgorithmInformation-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}

CertExtensions
 FROM PKIX1Implicit-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}
```

```
Attributes{}, CRIAttributes, PKInfoAlgorithms
 FROM PKCS-10
  { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7)
    id-mod(0) id-mod-pkcs10-2009(69) }
;

aa-certificationRequestInfoTemplate ATTRIBUTE ::=
  { TYPE CertificationRequestInfoTemplate IDENTIFIED BY
    id-aa-certificationRequestInfoTemplate }

id-aa-certificationRequestInfoTemplate OBJECT IDENTIFIER ::=
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) aa(2) id-aa-certificationRequestInfoTemplate(61) }

--  like CertificationRequestInfo but OPTIONAL subject, subjectPKInfo
CertificationRequestInfoTemplate ::= SEQUENCE {
    version       INTEGER { v1(0) } (v1, ... ),
    subject       NameTemplate OPTIONAL,
    subjectPKInfo [0] SubjectPublicKeyInfoTemplate
                            {{ PKInfoAlgorithms }} OPTIONAL,
    attributes    [1] Attributes{{ CRIAttributes }}
}


--  like Name, but with OPTIONAL RDN values
NameTemplate ::= CHOICE { -- only one possibility for now --
    rdnSequence  RDNSequenceTemplate }

RDNSequenceTemplate ::= SEQUENCE OF RelativeDistinguishedNameTemplate

RelativeDistinguishedNameTemplate  ::= SET SIZE (1 .. MAX)
    OF SingleAttributeTemplate { {SupportedAttributes} }

--  like Attributes, but with OPTIONAL value
SingleAttributeTemplates{ATTRIBUTE:AttrSet} ::= SEQUENCE OF
    SingleAttributeTemplates{ {AttrSet} }

--  like SingleAttribute, but with OPTIONAL value
SingleAttributeTemplate{ATTRIBUTE:AttrSet} ::= SEQUENCE {
    type      ATTRIBUTE.&id({AttrSet}),
    value     ATTRIBUTE.&Type({AttrSet}{@type}) OPTIONAL
}

--  like SubjectPublicKeyInfo, but with OPTIONAL subjectPublicKey
SubjectPublicKeyInfoTemplate{PUBLIC-KEY:IOSet} ::= SEQUENCE {
    algorithm        AlgorithmIdentifier{PUBLIC-KEY, {IOSet}},
    subjectPublicKey BIT STRING OPTIONAL
}

id-aa-extensionReqTemplate OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
  smime(16) aa(2) id-aa-extensionReqTemplate(62) }

--  like extensionRequest, but with OPTIONAL Extension extnValues
--  original definition was in PKCS#9 RFC 2985, Section 5.4.2
at-extensionReqTemplate ATTRIBUTE ::= {
    TYPE ExtensionReqTemplate IDENTIFIED BY
```

```
                  id-aa-extensionReqTemplate }

ExtensionReqTemplate ::= ExtensionTemplates{{CertExtensions}}

--  like Extensions, but with OPTIONAL extnValue
ExtensionTemplates{EXTENSION:ExtensionSet} ::=
    SEQUENCE SIZE (1..MAX) OF ExtensionTemplate{{ExtensionSet}}

--  like Extension, but with OPTIONAL extnValue
ExtensionTemplate{EXTENSION:ExtensionSet} ::= SEQUENCE {
    extnID    EXTENSION.&id({ExtensionSet}),
    critical  BOOLEAN
  --                  (EXTENSION.&Critical({ExtensionSet}{@extnID}))
                    DEFAULT FALSE,
    extnValue OCTET STRING (CONTAINING
              EXTENSION.&ExtnType({ExtensionSet}{@extnID})) OPTIONAL
              --  contains the DER encoding of the ASN.1 value
              --  corresponding to the extension type identified
              --  by extnID when present
}

END
```

Acknowledgments

Corey Bonnell crafted Example 2 using a different tool, and this helped debug other running code.

Carl Wallace provided major parts of the CertificationRequestInfoTemplate syntax declaration.

Russ Housley conducted many reviews of the ASN.1 module and suggested many fixes.

Deb Cooley conducted the usual Area Director Review.

Authors' Addresses

Michael Richardson (EDITOR)
Sandelman Software Works
Email: mcr+ietf@sandelman.ca

Owen Friel
Cisco
Email: ofriel@cisco.com

Dr. David von Oheimb
Siemens
Email: dev@ddvo.net

Dan Harkins
The Industrial Lounge
Email: dharkins@lounge.org