Clarification of RFC7030 CSR Attributes Definition
draft-ietf-lamps-rfc7030-csrattrs(-05)

Michael Richardson* <mcr+ietf@sandelman.ca>
Dan Harkins (Industrial Lounge)
David von Oheimb (Siemens)
Owen Friel (Cisco)

(exactly 365 days since IETF114)
The Story so far

- RFC7030 was unclear about CSR attributes
- RFC8994 (ACP) and RFC8995 (BRSKI) made an assumption that values could be provided
  - (RFC7030 and RFC8995 have one author in common)
- Discussed at Philadelphia (IETF114), a virtual interim, and a plan to revise to include extensionRequests
- New examples were made, running code, and -05 published.
  - could be finished now
    - examples from MCR, Dan, see the document
- Proposal from David for greenfield uses, less complexity.
CertificationRequest ::= SEQUENCE {
    certificationRequestInfo CertificationRequestInfo,
    signatureAlgorithm AlgorithmIdentifier{{ SignatureAlgorithms }},
    signature          BIT STRING
}

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

Attributes { ATTRIBUTE:IOSet } ::= SET OF Attribute{ { IOSet } }

Attribute { ATTRIBUTE:IOSet } ::= SEQUENCE {
    type   ATTRIBUTE.&id({IOSet}),
    values SET SIZE(1..MAX) OF ATTRIBUTE.&Type({IOSet}{@type})
}
CsrAttrs as defined in RFC 7030:

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})
}

These resemble basically the Attributes of PKCS#10.

They allow providing patterns for attributes such as X.509 extensions and also allow to give individual OIDs with unclear ad-hoc interpretation, but do not allow giving patterns for the "subject" and "subjectPKInfo" fields.
Requested CSR attributes, such as X.509 extensions, can then be given in straightforward and simple way, namely as patterns. Also patterns for the subject and subjectPKInfo (i.e., key) fields can be given this way.

For backward compatibility with RFC 7030 we cannot change the type of CsrAttrs, but since the attribute values are entirely flexible we can define a new attribute OID for CsrAttrs with the associated value type being CertificationRequestInfo and then require that CsrAttrs MUST contain just one Attribute of that form.

The embedded CertificationRequestInfo structure contains a partially filled-in CSR.
New approach: CertificationRequestInfo pattern example

```plaintext
version: 0
subject: "CN=,serialNumber=4711"
subjectPKInfo:

  AlgorithmIdentifier:
    OID: id-ecPublicKey,
    parameters: secp384r1
  subjectPublicKey: empty BIT STRING # zero length because key value always to be filled in by client

attributes: SEQUENCE {
  Attribute:
    OID: challengePassword
    values: { NULL }, # indicating value to be filled in
  Attribute:
    OID: extensionRequest
    values: {
      OID: subjectAltName
      critical: true
      extnValue: email:potato@example.com
    }
}```
Is this new mechanism worth having?

**PRO:** allows specification of SubjectDN, which is important for provisioning of Initial Device Identity (IDevID) certificates.

Could be much simpler if other mechanisms do not need to be coded. May leverage existing CSR libraries.

**CON:** it is new code, do we need both old and new methods? Maybe not for existing uses, but only for greenfields?
Discussion/Questions

Are we done?